



Universidad Popular Autónoma del Estado de Puebla
Departamento de Ingenierías
Facultad de Tecnologías de Información

**Cubecraft Game: Framework de desarrollo de
videojuegos para dispositivos móviles**

Tesis que para obtener el grado de Licenciado en Tecnologías de Información
Presenta

Manuel Romero Rivera

Asesor: Dr. Jorge Rafael Aguilar Cisneros



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Dedicatoria

A ti señor:

Que nada detienes, para cumplir su misión. Que me has hecho hombre sensato para reconocer tu sabiduría como regalo y al conocimiento como mi única alternativa de educación. Dones con los que iluminaste mi vida estudiantil hasta lograr esta mi primer meta profesional misma que te ofrezco con todo mi corazón.

A mis padres:

Dr. Manuel Romero Cordero
Dra. María Eugenia Rivera Téxon

Que con su honestidad, rectitud, impetuosa voluntad responsable e inmenso amor me enseñaron a través del ejemplo el poder del principio. Por confiar en mí durante todos estos años y darme la fortaleza para seguir adelante. Con todo mi cariño les ofrezco este humilde testimonio de mi gratitud.

A mis hermanos:

María Teresa y Alvaro

Por haber compartido estos años de esfuerzo, alegrías y tristezas y al ver coronados mis anhelos, esto, es también un logro de ustedes.

A mis abuelos:

Manuel Romero Beranza
Elvira Cordero Pastrana
Abel Rivera Garcia (qepd)
Altagracia Texon Galicia

Por todo su amor y confianza, sus oraciones y consejos que me ayudaron a hacer posible mi formación profesional. Para ustedes, mi agradecimiento eterno.

A mis amigos:

Principalmente a Mijail Vicencio Jiménez

Gracias por la amistad de tantos años, por los consejos y los ánimos en los momentos difíciles de ésta tesis. Esto también es un logro tuyo.

Agradecimientos

M.C. Lourdes Hernandez Hernandez

Por el apoyo otorgado en el planteamiento y análisis de los datos estadísticos que utilice en esta tesis. Mi agradecimiento eterno.

Dr. Jorge Aguilar Cisneros

Por haber creído en mí desde el principio de la carrera, por ser mi consejero y guía profesional. Gracias por todos los años de arduo trabajo, logros y amistad que me ha otorgado.

Dr. Lorna Veronica Rosas Tellez

Le agradezco haberme enseñado a superar mis límites, es una enseñanza de vida que mantendré de ahora en adelante y por el apoyo moral que me otorgó durante el desarrollo de esta tesis.

Índice

Dedicatoria	1
Agradecimientos.....	2
Índice.....	3
Índice de Imágenes	7
Índice de Códigos.....	8
Índice de Gráficas	8
Índice de Análisis Estadísticos	9
Introducción	11
Capítulo 1. Problemática y Motivación	13
1.1 Problemática	13
1.2 Motivación	13
1.3 Objetivo General	14
1.4 Objetivos Específicos	14
1.5 Alcances y Limitaciones	14
1.6 Herramientas a Utilizar	15
Capítulo 2. Marco Teórico	17
2.1 Framework.....	17
2.1.1 Kodu.....	18
2.1.3 Greenfoot.....	20
2.1.4 Alice	21
2.1.5 LUA.....	22
2.1.6 AndEngine	23
2.2 Videojuegos	24
2.2.1 La Historia de los Videojuegos (Belli & Lopez, 2008).....	24
2.3 Ciclo de Vida de los videojuegos	26
2.3.1 Scrum	27
2.3.1.1 Las Reuniones	28
2.3.1.2 Los elementos.....	29
2.3.1.3 Ventajas de Scrum sobre otras metodologías ágiles	29
2.3.2 Huddle.....	29
2.3.3 SUM	30
2.3.4 MCMI	31

2.4 Dispositivos Móviles	32
2.4.1 Generaciones de la Telefonía Celular	33
2.4.1.1 Generación Cero (0G)	33
2.4.1.2 Primera Generación (1G)	34
2.4.1.3 Segunda Generación (2G)	34
2.4.1.4 Tercera Generación (3G)	35
2.4.2 Sistemas Operativos Móviles	36
2.4.2.1 Windows Mobile	36
2.4.2.2 Symbian	37
2.4.2.3 iPhone iOS	37
2.4.2.4 Android	38
2.5 Usabilidad e Iteración Humano Computadora	39
Capítulo 3. Desarrollo	43
3.1 Product Backlog	43
3.2 Primer Sprint	44
3.3 Segundo Sprint	47
3.4 Tercer Sprint	56
3.4.1 Generando los proyectos Android	56
3.4.2 Generando clases del proyecto	58
3.4.3 Compilando el proyecto utilizado	60
3.4.4 Firmando la aplicación	61
3.4.5 Iniciando un emulador	62
3.4.6 Instalando la aplicación	62
3.5 Pruebas	64
Capítulo 4. Aplicación del Experimento	67
4.1 Objetivo	67
4.2 Grupos de Enfoque	67
4.3 Hipótesis	67
4.4 Preguntas de Investigación	67
4.5 Experimento	68
4.6 Encuestas	69
4.7 Método de Análisis	71
4.8 Tratamiento del Análisis	72
4.9 Historias para la creación de videojuegos	73

Capítulo 5. Resultados.....	75
5.1 Recolección de Información.....	76
5.2 Interpretación de Resultados.....	76
5.2.1 PI1. ¿La herramienta es intuitiva para el usuario?	76
5.2.1.1 Primaria.....	77
5.2.1.2 Preparatoria.....	79
5.2.2 PI2. ¿Cuál es la aceptación que tiene el sistema entre los usuarios objetivo?	82
5.2.2.1 Primaria.....	82
5.2.2.2 Preparatoria.....	84
5.2.3 PI3. ¿Qué flexibilidad puede llegar a otorgar el sistema para el desarrollo de videojuegos?	87
5.2.3.1 Primaria.....	87
5.2.3.2 Preparatoria.....	88
5.3 Comentarios en las encuestas	91
Capítulo 6. Conclusiones y Trabajo a Futuro	93
6.1 Conclusiones	93
6.2 Trabajo a Futuro.....	94
Apéndice A. Arquitectura de Software	97
A.1 Framework Conceptual	98
A.1.1 Vista de Casos de Uso.....	99
A.1.2 Vista Lógica	103
A.1.3 Vista de Desarrollo	110
A.1.4 Vista Física.....	111
A.2 Arquitectura del Sitio Web	112
A.2.1 Vista de Casos de Uso.....	113
A.2.2 Vista Lógica	117
A.2.3 Vista de Desarrollo	121
A.2.4 Vista Física.....	123
A.2.5 Vista de Procesos	124
A.3 Diseño de la Base de Datos.....	125
Apéndice B. Estándar de Codificación C#.....	127
B.1 Estándar de Escritura de Nombre de Métodos y Variables.....	127
B.2 Identación y Espacios	128
B.3 Buenas Prácticas de Programación	129

B.4 Comentarios.....	130
B.5 Manejo de Excepciones	130
Apéndice C. Estándar de Codificación Java	133
C.1 Organización de Archivos	133
C.2 Identación.....	134
C.3 Longitud y División de Líneas.....	134
C.4 Declaraciones.....	135
C.5 Declaración de Clases e Interfaces.....	135
C.6 Sentencias.....	135
C.7 Espacios en Blanco	136
C.8 Nomenclatura de Identificadores	136
C.9 Prácticas de Programación	137
C.10 Documentación	137
Apéndice D. Administración del Proyecto.....	139
D.1 User Stories	139
D.2 Product Backlog	149
D.3 Product Backlog Primer Sprint.....	157
D.4 Product Backlog Segundo Sprint.....	161
D.5 Product Backlog Tercer Sprint	163
Apéndice E. Manual Técnico	167
E.1 Requerimientos Mínimos	167
E.2 Instalación del Sistema	167
E.2.1 Instalación de Cubecraft Game	167
E.2.2 Instalación de Java SDK.....	168
E.2.3 Instalación de Android SDK	170
E.2.3.1 Instalación de la herramienta.....	170
E.2.3.2 Instalación de la Versión de Android	171
E.2.3.3 Creación y Primer arranque del Emulador	172
E.2.4 Configuración del config.ini.....	173
E.2.4.1 Usuarios con permisos de Administrador	173
E.2.4.2 Usuarios sin permisos de Administrador	174
E.3 Ambiente de Trabajo	174
E.4. Configuración de dispositivo móvil para ejecutar la aplicación desde el sistema	175
Apéndice F. Manual de Usuario.....	177

F.1. Pantalla de Inicio	177
F.2. Pantalla Principal.....	178
F.3. Generar un Nuevo Proyecto	180
F.4. Abrir un proyecto existente	181
F.5. Agregar funcionalidad al área de trabajo	181
F.6. Configurando cubos.....	182
F.7. Compilando la aplicación	183
F.8 Ejemplo de Aplicación en Cubecraft Game.....	185
Apéndice G. Derechos de Autor	191
Referencias	193

Índice de Imágenes

Figura 1: Avance de Alumnos con Kodu	19
Figura 2: Interfaz de Scratch.....	19
Figura 3: Interfaz de Greenfoot	21
Figura 4: Arquitectura de Alice	22
Figura 5: Ciclo de Vida de Scrum	28
Figura 6: Distribución de Tecnologías	43
Figura 7: Tiempo Estimado vs Tiempo Real Primer Sprint	46
Figura 8: Interfaz Cubecraft Game.....	48
Figura 9: Interfaz Nuevo Proyecto	51
Figura 10: Interfaz de Compilación	52
Figura 11: Interfaz para Exportar como Imagen.....	52
Figura 12: Cubos de Funcionalidad	53
Figura 13: Tiempo Estimado vs Tiempo Real Segundo Sprint	56
Figura 14: Tiempo Estimado vs. Tiempo Real del Tercer Sprint	63
Figura 15: Tiempo de respuesta del inicio del sistema	64
Figura 16: Tiempo de respuesta al crear un proyecto	65
Figura 17: Rendimiento al insertar un cubo	65
Figura 18: Tiempo de respuesta a la configuración de cubos	65
Figura 19: Tiempo de respuesta para la compilación del juego	66

Índice de Códigos

Código 1: Creación Proyecto Android	57
Código 2: Importación de la Librería.....	58
Código 3: Análisis de Cubos Maestros	58
Código 4: Asignación de la configuración a la funcionalidad de los cubos	59
Código 5: Detección de la pantalla principal del juego.....	59
Código 6: Compilación de archivos .class utilizando javac.....	60
Código 7: Creación de archivos DEX utilizando el comando DX	60
Código 8: Generando el archivo APK sin firmar.....	61
Código 9: Firmando aplicaciones en Android.....	61
Código 10: Optimización de los proyectos APK.....	62
Código 11: Mostrando un Emulador Android en pantalla	62
Código 12: Desinstalando aplicación Android	62
Código 13: Instalando aplicación Android.....	63
Código 14: Iniciando aplicaciones en Android.....	63

Índice de Gráficas

Gráfica 1: Facilidad de uso en Primaria.....	77
Gráfica 2: Sencillez del Sistema en Primaria.....	78
Gráfica 3: Facilidad de Uso en Preparatoria	79
Gráfica 4: Distribución de la pantalla para estudiantes de Preparatoria	80
Gráfica 5: Dinámica de la actividad en Preparatoria	81
Gráfica 6: Aceptación de los colores de los Cubos en Primaria.....	82
Gráfica 7: Estadísticas del gusto por el sistema en primaria.....	83
Gráfica 8: Gusto hacia el aspecto del cubo en preparatoria	84
Gráfica 9: Estadísticas de la facilidad para desarrollar juegos en preparatoria	85
Gráfica 10: Aceptación de los colores empleados en Preparatoria	86
Gráfica 11: Aceptación de las funciones para juegos en primaria.....	87
Gráfica 12: Datos sobre las funcionalidades para juegos en preparatoria	88
Gráfica 13: Datos sobre la funcionalidad necesaria para crear videojuegos en preparatoria	89
Gráfica 14: Facilidad para configurar un juego en preparatoria	90

Índice de Análisis Estadísticos

Análisis Estadístico 1: Facilidad para Usar el Programa en Primaria	77
Análisis Estadístico 2: Sencillez del Sistema en Primaria	78
Análisis Estadístico 3: Facilidad de Uso en Preparatoria	79
Análisis Estadístico 4: Distribución de los Elementos en Pantalla en Preparatoria	80
Análisis Estadístico 5: Dinámica de la actividad en Preparatoria	81
Análisis Estadístico 6: Aceptación de los colores de los cubos en Primaria	83
Análisis Estadístico 7: Gusto por el sistema en primaria.....	83
Análisis Estadístico 8: Gusto por el aspecto del cubo en Preparatoria	84
Análisis Estadístico 9: Facilidad para Desarrollar Juegos en Preparatoria	85
Análisis Estadístico 10: Aceptación de los colores empleados en Preparatoria	86
Análisis Estadístico 11: Aceptación de las funciones para juegos en Preparatoria	88
Análisis Estadístico 12: La funcionalidad disponible en los cubos para preparatoria	89
Análisis Estadístico 13: Datos sobre la funcionalidad necesaria para crear videojuegos en preparatoria	90
Análisis Estadístico 14: Facilidad para encontrar y configurar un juego en Preparatoria.....	90

Para empezar un gran proyecto, hace falta valentía
Para terminar un proyecto, hace falta perseverancia
-- Anónimo

Introducción

Un videojuego es un sistema de software destinado al entretenimiento que puede funcionar en distintos dispositivos como computadoras, consolas especializadas, teléfonos celulares, entre otros, integrando audio y video que permitan al jugador disfrutar de experiencias que, en muchos casos, serían difíciles de hacer en el mundo real. Además en estos días, los videojuegos son más que productos informáticos, son también una herramienta de trabajo, información e incluso un fenómeno social. (García, 2005)

A pesar de que no existen investigaciones definitivas, mucho de los trabajos hechos tanto cualitativos como cuantitativos demuestran que muchos juegos mejoran ciertas habilidades que ayudan al desarrollo intelectual de la persona. Es decir, la inteligencia, al contrario de lo que se ha dicho en algunas investigaciones, no parece degradarse cuando se utiliza este tipo de entretenimiento digital. Por el contrario, parece que favorece el desarrollo de determinados aspectos, especialmente los espaciales. Desde el punto de vista educativo, los videojuegos permiten una enseñanza eficaz, en algunos casos de una manera mejor que el sistema escolar actual. (García, 2005)

Desde el punto de vista de la teoría de enseñanza social, algunos de los factores que fomentan la motivación intrínseca de los videojuegos son: Un aprendizaje de carácter lúdico, la dificultad de los problemas y sus habilidades para resolverlos aumenta progresivamente adaptándose al ritmo de cada uno, reconocimiento social de los logros adquiridos y estimulación simultánea en múltiples niveles como visual, auditivo, entre otros. (Rodríguez, 2005)

Algunos autores reconocen que los videojuegos aumentan la adquisición y desarrollo de algunas estrategias fundamentales para el aprendizaje como la resolución de problemas, el aprendizaje de secuencias, razonamiento deductivo y la memorización. (Lavilla, 2005)

Buscando una manera que los estudiantes dejen de ser usuarios de los juegos y se vuelvan sus propios desarrolladores, se propone la realización de un framework donde los temas de programación se vuelvan transparentes. Un framework es una herramienta incompleta y configurable que permite que los usuarios desarrollen aplicaciones diferentes teniendo como base un patrón de diseño.

Existen muchos frameworks destinados a la creación de aplicaciones y videojuegos para cada uno de los sistemas operativos de las computadoras y dispositivos móviles. Pero entre toda la lista, no existe uno que sea especializado en la creación de videojuegos para los sistemas operativos móviles más comunes en el mercado (Rogers, 2011). Es por esta razón que se propone un framework destinado a niños que cursan quinto o sexto año de primaria y jóvenes que se encuentren en su último semestre de preparatoria o en preparación pre-universitaria que les permita interactuar con los diferentes lenguajes de programación y con varios dispositivos de una forma lúdica.

La capacidad consiste en lo que se es capaz de hacer.

La motivación determina lo que se hace.

La actitud determina lo bien que se hace.

-- Lou Holtz

Capítulo 1. Problemática y Motivación

Los videojuegos se han vuelto una de las industrias más importantes del mundo, incluso más importante que el cine. Pero existe un problema, se necesitan amplios conocimientos en el desarrollo de software para que puedan realizarse.

En este capítulo se habla sobre esa problemática y la propuesta que se hace, además de definir los alcances y limitaciones que tiene el sistema basado en su complejidad y se definen las herramientas que se utilizarán basados en sus ventajas.

1.1 Problemática

La mayoría de las personas que juegan videojuegos se han preguntado alguna vez: ¿Cómo se hace un videojuego?, ¿Qué tengo que saber?, ¿Por dónde empiezo?, y muchas preguntas más. Muchos han intentado hacerlos sin resultado alguno, otros en cambio han podido crear sus primeros juegos con un poco de esfuerzo, pero aun así siempre quedan preguntas pendientes por responder y bastante por aprender (Albornoz, 2006).

La problemática principal es que para poder desarrollar los videojuegos es necesario tener conocimientos en el desarrollo de software, y entre más alto sea éste conocimiento, mayor es el avance que se tiene en el tema, haciendo que el desarrollo para personas sin conocimiento alguno sea nulo.

Es por esta razón que se propone el desarrollo de un framework que permita crear videojuegos para dispositivos móviles sin necesidad que los usuarios sean expertos en el área de sistemas computacionales o que tengan experiencia con lenguajes de programación.

1.2 Motivación

Se planteó el desarrollo de este framework por diferentes razones, entre las que se encuentran:

- Los videojuegos tienen más usuarios hoy en día que cualquier otra industria digital
- Los juegos permiten una enseñanza eficaz que en ocasiones es mejor que el actual sistema escolar.
- Contienen un carácter lúdico de los aprendizajes
- Algunos jugadores se identifican con héroes o personajes que fomentan la imitación
- Ayudan a procesar información recibida de varias fuentes simultáneamente.

- Permite que los jugadores verifiquen hipótesis y conocimientos aprendidos de forma empírica o en el salón de clases.
- Tienen un importante potencial educativo ya que estimulan la capacidad lógica, el desarrollo de estrategias encaminadas a la resolución de problemas y tolerancia ante el fracaso.
- Permite desarrollar la coordinación visual-manual, la capacidad de concentrarse en una tarea y de mantener la atención sobre ella.
- Favorecen el contacto social con el grupo de iguales y la participación en actividades comunes
- Además de entretener, son una introducción a la informática y a las nuevas tecnologías.
- Utilizar la experiencia en móviles obtenida para que los jugadores dejen de ser usuarios y se vuelvan creadores.

1.3 Objetivo General

Desarrollar un framework para facilitar el desarrollo de videojuegos para dispositivos móviles que utilicen el Sistema Operativo Android con las nuevas tecnologías que se encuentran en el mercado para que niños en edades escolares básicas puedan crearlos solo uniendo cubos.

1.4 Objetivos Específicos

- Desarrollar un framework en C# que permita desarrollar videojuegos para dispositivos Android y Symbian.
- Desarrollar el framework para niños utilizando el lenguaje de programación C# .NET para que desarrollen videojuegos de una forma sencilla.
- Diseñar las bases de un sitio web utilizando herramientas que vuelva la navegación fluida para que los usuarios del framework puedan publicar sus aplicaciones y alentar la colaboración.
- Proporcionar la herramienta desarrollada a estudiantes de nivel primaria y preparatoria para conocer la usabilidad, la aceptación del sistema y la forma de poder mejorarla para su uso.

1.5 Alcances y Limitaciones

Basados en el tiempo de desarrollo necesario para generar un framework de este tipo, se plantea que se tengan los siguientes alcances:

- Desarrollar el framework funcionando en la versión 1.5 con las siguientes funcionalidades:
 - Detección de cubos organizados en archivos
 - Agregar cubos haciendo Drag&Drop
 - Permitir configurar funciones desde una interfaz amigable

- Mostrar lista de funciones agregadas y errores.
- Desarrollar las librerías de Android para la creación de juegos que incluyen el manejo de dispositivos periféricos y el uso de pantallas predefinidas.
- Compilación y verificación de los juegos de forma transparente al usuario
- Desarrollo de herramientas que permitan a los expertos en el área de Sistemas Computacionales generar sus propias funciones.

Se tiene una lista de limitaciones del framework. Estas limitaciones quedarán marcadas como un trabajo a futuro, pero las bases quedarán cimentadas en este trabajo.

- Desarrollo de la librería para Symbian y iOS
- Creación de un sitio web donde los usuarios puedan publicar sus aplicaciones
- Sistema de Actualizaciones vía internet

1.6 Herramientas a Utilizar

Se analizaron varios frameworks que existen en el mercado para el desarrollo del proyecto, eligiendo las siguientes por sus beneficios:

- MySQL Server 5.5: Este tipo de servidor es utilizado en todos los paneles de control de servidores web. Al ser utilizado primero de forma local para realizar el diseño e implementación de la base de datos, permite que al emigrarla al panel web no cause ningún conflicto de funcionamiento.
- PHP versión 4: Esta es una versión de Java utilizada comúnmente para el desarrollo de Web Services en servidores web sin soporte para JSP o tecnologías .NET. Al ser utilizada como un lenguaje estructurado, permite que el mantenimiento del código sea sencillo.
- C# Framework .NET 4.0: Esta versión de framework es soportada por todos los equipos que cuenten con una versión de Windows Vista o superior, permitiendo que el sistema pueda ser ejecutado en cualquier ambiente Windows sin ninguna restricción.
- Android SDK 2.1 Eclair: Aunque esta versión de Android ya casi no es utilizada, es necesaria debido a que se quiere que los juegos desarrollados sean compatibles con varios modelos de teléfonos móviles Android, sin importar la modernidad y potencia de los equipos.

La investigación se asemeja a los largos meses de gestación,
Y la solución del problema, al día del nacimiento

-- Mao Tse-Tung

Capítulo 2. Marco Teórico

En este capítulo se analizan las ventajas y desventajas de cada los frameworks para el desarrollo de videojuegos más comunes, con la finalidad de tomar ideas para el desarrollo de nuestro framework.

Más adelante se definen conceptos básicos sobre los dispositivos móviles como su historia, las características de cada una de las generaciones que existen y las ventajas y desventajas de cada sistema operativo en el mercado, con la finalidad de conocer la forma en la que funcionan y las diferencias entre los modelos existentes en el mercado.

El capítulo finaliza con un análisis de las diferentes metodologías ágiles y tradicionales para el desarrollo de videojuegos, donde se toma SCRUM como metodología ágil con sus respectivas modificaciones para adaptarlas al objetivo final.

2.1 Framework

En general, con el término Framework, nos referimos a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación (Gutierrez, 2006). Usualmente se encuentran conformados por programas de soporte, compiladores, librerías de código, una interfaz gráfica amigable y herramientas que ayudan a utilizar todos los componentes de una forma fácil y rápida en el desarrollo de un proyecto o solución (Maxxes System, 2010).

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión artificial, para el desarrollo de videojuegos, y para cualquier ámbito que pueda ocurrírseles.

En otras palabras, un framework se puede considerar como una aplicación genérica, incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta (Gutierrez, 2006).

Los objetivos principales que persiguen un framework son:

- Acelerar el proceso de desarrollo
- Reutilizar código ya existente
- Promover buenas prácticas de desarrollo como el uso de patrones.

Aunque en el mercado se ofrecen múltiples frameworks, todos ellos se encuentran dentro de un marco de tipologías que son (Gutierrez, 2006):

- Orientado a la interfaz de Usuario
- Orientados a aplicaciones de publicación de documentos
- Orientados al control de eventos.
- Entre otros.

En los siguientes apartados se habla de los frameworks de desarrollo más utilizados en el mercado para el desarrollo de videojuegos. De cada framework se analizan sus principales ventajas y la forma en la que funcionan, que servirán de base para el desarrollo de nuestra herramienta.

2.1.1 Kodu

Kodu es una herramienta de programación visual especializada para videojuegos que son generados a partir de una idea. Está diseñado para ser accesible a niños y divertir a cualquiera.

Las características principales de ésta plataforma son:

- Ofrece un entorno creativo end-to-end.
- Utiliza un lenguaje de alto nivel que incorpora en el mundo real algunos conceptos primitivos como la colisión, el color y la visión.
- Utiliza un Xbox Game Controller para la entrada (no requiere un teclado).
- La herramienta y los juegos pueden ser ejecutados en una consola Xbox 360 o en PC.
- Trae integrado un editor de terrenos interactivo para crear mundos de forma arbitraria y tamaño.
- Integra un total de 20 personajes diferentes con capacidades distintas (Knowledge Bank: Next Generation Research, 2010).

Actualmente Kodu es utilizado en países como Malasia y Singapur como una herramienta educativa, ya que enseña mucho más que a programar:

- Es una herramienta que mejora la creación de historias.
- Demuestra que la programación puede ser un medio para que los niños puedan demostrar su creatividad.
- Ayuda a los niños con la resolución de problemas difíciles dividiéndolos en pasos.
- Introduce condiciones y secuencias que enseña el concepto de “Causa – Efecto”.
- Los estudiantes aprenden acerca de la cooperación, la lógica y la creatividad, además de aprender a programar.

La figura 1 se muestra el avance y entrega de los alumnos en un instituto educativo en Australia, los cuales arrojan que las actividades realizadas aumentaron considerablemente el compromiso con la educación (Knowledge Bank: Next Generation Research, 2010).

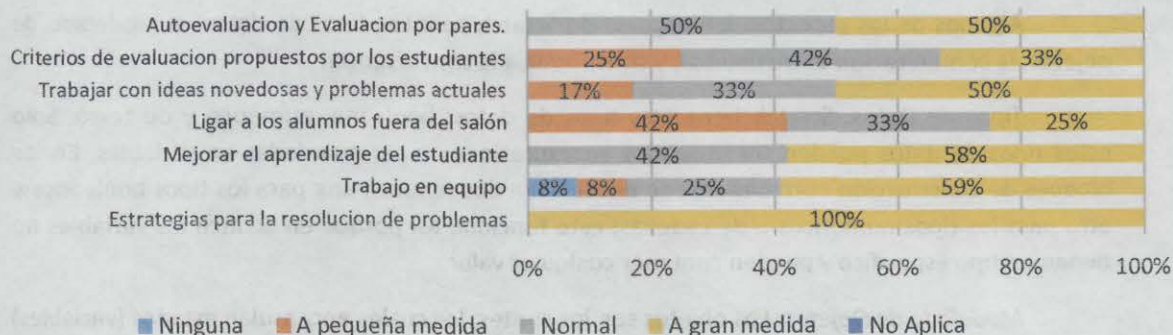


Figura 1: Avance de Alumnos con Kodu

2.1.2 Scratch

El proyecto Scratch empezó en el año 2003, en el MIT (Massachusetts Institute of Technology) pero el software y el sitio web fue liberado al público en el año 2007. Actualmente Scratch se encuentra traducido en aproximadamente 50 lenguajes y tiene más de dos millones de descargas, esto se debe a la facilidad con la que se pueden desarrollar animaciones, juegos, noticias online, reportes de libros, cartas de felicitaciones, simulaciones, entre otros. (Maloney, Resnick, Rusk, & Eastmond, 2010)

El diseño original de Scratch fue motivado por las necesidades e interés de las personas jóvenes en un rango de edad entre 8 a 16 años, que aplica algunas actividades de manipulación de medios que son populares entre la cultura de los jóvenes.

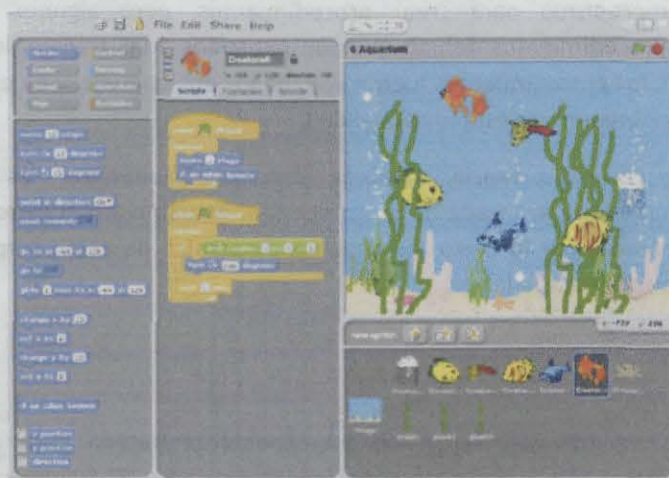


Figura 2: Interfaz de Scratch

La figura 2 muestra la ventana de Scratch, que tiene cuatro paneles principales. El panel izquierdo es la paleta de comandos con botones para seleccionar algunas acciones, el panel central muestra el script del sprite seleccionado, mientras que el panel más largo en la esquina superior derecha es un campo donde se puede visualizar el funcionamiento del script y el panel

inferior derecho muestra las pre visualizaciones de todos los sprites que se encuentran en el proyecto.

Algunos de los aspectos del lenguaje de Scratch son: Los tipos de datos, el modelado de objetos, la comunicación entre objetos, y su enfoque a la concurrencia.

Tipos de Datos: Scratch tiene tres tipos de datos: booleano, numérico y de texto. Solo estos tipos de datos pueden ser utilizados en expresiones o ser guardadas en variables. En los bloques de construcción normalmente se encuentran dos ranuras, una para los tipos booleanos y otro para los tipos numéricos o de cadenas; esto funciona así porque en Scratch las variables no tienen un tipo específico y pueden contener cualquier valor.

Modelado de Objetos: Los objetos son los sprites, los cuales encapsulan estados (variables) y funcionamiento (scripts), pero como no tiene clases, solo se trata de un lenguaje basado en objetos más no uno orientado a objetos. Cada sprite tiene su propia colección de scripts independiente, este diseño implica un equilibrio entre la facilidad para entender la funcionalidad de cada sprite y la localización de los cambios en los códigos sin afectar los demás elementos del proyecto.

Comunicación entre sprites: Como se ha mencionado anteriormente, los sprites no pueden llamar el script de otro directamente, por lo tanto Scratch use un mecanismo de transmisión para permitir la comunicación y sincronización, cualquier sprite puede transmitir un mensaje normalmente hecho como una cadena de texto arbitraria. El modelo de transmisión de Scratch es “uno a varios”, porque una sola transmisión puede disparar varios scripts (posiblemente en múltiples objetos); de acoplamiento flexible, porque no importa cuántos receptores haya, y asíncrono, porque el mensaje de transmisión no espera hasta que los scripts disparados se completen.

Procedimientos: Versiones anteriores de Scratch (antes de ser liberado al público en general) tenía un mecanismo para crear procedimientos, pero algunos usuarios fueron confundidos por los procedimientos debido que son parecidos a las transmisiones. Pero por el interés de simplicidad, los procedimientos fueron eliminados del lenguaje, y de esta forma Scratch ha tenido un buen recibimiento sin ésta funcionalidad.

Concurrencia: La concurrencia, también llamada “Multi-hilos” es frecuentemente considerada como una técnica avanzada de programación. Actualmente nuestro mundo tiene un alto índice de concurrencia, por lo tanto, los usuarios de Scratch no están sorprendidos que un sprite pueda realizar varias cosas a la vez.

2.1.3 Greenfoot

Greenfoot es un software integrado educacional preparado para el aprendizaje y enseñanza de la programación a gente sin conocimientos previos. El framework está diseñado para un público de edad mínima de 14 años y es muy común en cursos introductorios a la educación universitaria. (Kolling, 2010)

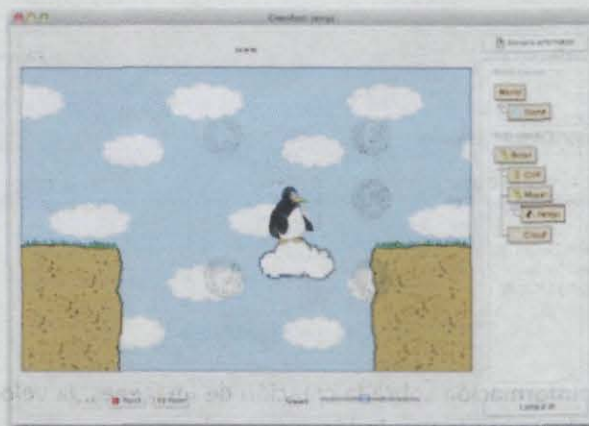


Figura 3: Interfaz de Greenfoot

La figura 3 muestra la interfaz de usuario principal de Greenfoot con un escenario (término utilizado para referirse a un proyecto). La parte principal de la ventana muestra el mundo de Greenfoot, y es el área donde el programa se ejecuta. El mundo es variable, definido por el usuario y contiene los objetos en el escenario. A la derecha se muestra un diagrama de clases que visualiza las clases usadas en el escenario y sus relaciones.

Uno de los objetivos de Greenfoot es un diseño que explícitamente se visualiza principales conceptos de la programación orientada a objetos. Y aunque se puede modificar el código fuente, los estudiantes no deben iniciar manipulándolo.

Un número importante de objetivos de diseño apuntalan el diseño de Greenfoot. A un alto nivel, pueden ser resumidos en dos puntos de diferentes perspectivas: (Kolling, 2010)

1. Desde la perspectiva del estudiante, el objetivo es hacer la programación atractiva, creativa y satisfactoria
2. Desde la perspectiva del maestro, el objetivo es que el ambiente de desarrollo es ayudar a enseñar aspectos importantes sobre la programación.

2.1.4 Alice

El proyecto Alice es un proyecto a gran escala que empezó en 1992 con la visión de crear un sistema interactivo de gráficos en 3D para apoyar en la exploración de la programación.

A pesar de todos los cambios internos de sus componentes, la arquitectura de Alice se mantiene más o menos estable. En términos de su última implementación, la arquitectura se muestra en la figura 4. (Conway, 1997)

En el nivel más bajo (capa 1), Alice depende de la librería grafica desarrollada por Microsoft llamada D3DRM (Direct 3D Retained-Mode), ésta capa provee de servicios básicos para el manejo de objetos en 3D y sus atributos, la textura de los mapas, la iluminación de la escena y el mantenimiento del árbol de coordenadas de los objetos 3D en la escena.

La siguiente capa (capa 2) es Ecila, que implementa un código fuente que permite ejecutar la librería Direct 3D (DLL para el lenguaje C) en Python. Ecila también incluye utilerías para crear

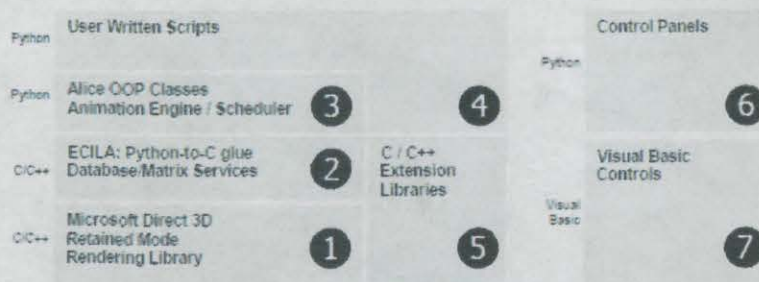


Figura 4: Arquitectura de Alice

nuevas escenas, obtener información sobre la creación de imágenes, la velocidad de cuadros por segundo y el número de objetos en la simulación.

Encima de la capa Ecila se encuentra la capa Alice (capa 3), ésta capa contiene los tres componentes más importantes de este sistema como las Definiciones del Paradigma Orientado a Objetos, El motor de Animación, y El Compilador.

La quinta capa es un traductor que permite convertir código C y C++ en funciones para ser llamadas en Python y viceversa; permitiendo un soporte inter-lenguajes entre C y Python. Un claro ejemplo del uso de ésta capa es la librería de Microsoft DirectPlay, la cual fue originalmente escrita en C y hace que sea ejecutable desde un código Python.

La sexta capa fue desarrollada para aquellos usuarios que deseen crear sus propios paneles 2D para sus aplicaciones 3D, este paquete se llama "Control Panels" y aunque su funcionalidad es limitada, tiene la ventaja de ser extremadamente simple y que toda la lógica detrás del panel de control está escrita en el mismo lenguaje que en toda la animación (Python) que controla los objetos 3D en pantalla.

2.1.5 LUA

LUA es un sencillo y a su vez poderoso lenguaje de propósito general concebido inicialmente para extender aplicaciones, es posible desarrollar sencillos videojuegos para Windows y consolas especializadas. (Pleguezuelos, 2007)

Este lenguaje es muy sencillo y muy usado en el campo de videojuegos. Así, juegos como "World of Warcraft" tienen módulos desarrollados con este lenguaje. Puesto que los bytecode generados por LUA corren sobre una máquina virtual, lo normal es ejecutar una aplicación sobre un PC, pero últimamente se han venido realizando máquinas virtuales para trabajar sobre videoconsolas portátiles comerciales.

LUA en un ambiente de desarrollo de videojuegos utilizando su aplicación LUAPlayer utiliza una técnica llamada "Double Buffer" que separa la interfaz gráfica del motor de la aplicación; esto se realiza para resolver un problema con las pantallas, ya que las pantallas de tubos de rayos catódicos (CRT) así como las de cristal líquido (LCD) pintan las imágenes pixel a pixel empezando

por la esquina superior izquierda y terminando en la esquina inferior derecha. A este proceso se le llama "refresh" y se hace con una determinada frecuencia. Puede ocurrir el caso (según la capacidad de la memoria RAM) que se inicie un nuevo refresh antes de que termine el otro. (Pleguezuelos, 2007)

Una desventaja que muestra LUA es que su máquina virtual solo puede cargar imágenes de formato JPG y PNG, pero para el desarrollo de videojuegos se utiliza comúnmente el segundo formato, ya que este permite el uso de transparencias.

2.1.6 AndEngine

Este framework fue creado a principios del 2010 por Nicolás Gramlich, basado en la disponibilidad de la potencia y motores para el desarrollo de videojuegos en 2D soportados por la plataforma Android.

Se creó con la finalidad de cumplir las necesidades de los desarrolladores por un framework que sea gratuito y de fácil uso, que sea capaz de permitir a los desarrolladores de videojuegos inexpertos un acceso rápido al mercado de videojuegos. En estos días, existen más de 2000 juegos utilizando AndEngine, actualmente la empresa Zynga lo implementa para sus desarrollos a nivel profesional. (Rogers, 2011)

AndEngine viene como un archivo .jar, para que sea agregado a los proyectos de desarrollo; se encuentra bajo la licencia GNU Lesser GPL, que permite usar el código fuente para cualquier propósito.

Los conceptos que utiliza AndEngine son muy parecidos a los empleados en el desarrollo de películas:

- Camera: Determina la vista del juego que se muestra a los jugadores, basándose en un espacio 2D. Tiene la facilidad de realizar movimiento y zoom para cambiar la perspectiva.
- Scene: En los juegos, como en las películas, consisten en una serie de escenas donde se lleva a cabo toda la acción.
- Layer: Las capas son gráficos sobrepuestos que permiten realizar efectos 3D en un ambiente 2D, su principal función es realizar el efecto de movimiento de cada una de las partes del juego moviendo solo una capa.
- Sprite: Los Sprites son la representación visual de los actores que pueden ser personas u objetos. Usualmente son animados y deben ser cargados de una sola imagen llamada "Sprite Sheet"
- Entity: Las entidades son las encargadas de dibujar en las pantallas (sprites, títulos, figuras geométricas, etc.) y tienen ciertas características como color, rotación, escala, posición, entre otras.

2.2 Videojuegos

A partir del 2007, los videojuegos son la puerta de entrada de niños y jóvenes en las tecnologías de información. Mediante el videojuego los niños adquieren capacidades y desarrollan habilidades diversas, las más importantes de las cuales son la familiarización con las nuevas tecnologías, su aprecio y su dominio. Por este motivo el videojuego es en estos momentos un elemento determinante para socializarse en el mundo de las nuevas tecnologías. (Belli & Lopez, 2008)

Jugamos a los videojuegos en espacios que pueden ser públicos o privados, en casa o en un cibercafé y sobre todo, en varios tipos de lugares y con diferentes modalidades de video jugar. Al compartir estos espacios para los videojuegos, entendiendo esos espacios como lugares para la socialización del conocimiento entre ellos y lugar donde tener la oportunidad de expresar las emociones reales en un contexto virtual.

El jugar crea lo inexistente, como por ejemplo, el sentido mismo del arte contemporáneo, crear algo que no existe: "El juego es simbólico, permite la aparición de nuevos significados y resinifica los ya existentes". (Gil & Vida, 2007)

2.2.1 La Historia de los Videojuegos (Belli & Lopez, 2008)

Los videojuegos han evolucionado rápidamente y de una manera excepcional tanto en gráficos como en sonido y aunque se puede decir que los videojuegos aparecieron formalmente hace unos 30 años, los orígenes de estos se remontan mucho más atrás, al igual que las compañías que lo fundaron. En la siguiente tabla se muestra la evolución de los videojuegos a partir de su cronología.

1952	Alexander S. Douglas desarrolla el Juego "Nought And Crosses", también llamado OXO, el cual era una versión computarizada del tres en línea y se ejecutaba sobre la EDSAC (Electronic Delay Storage Automatic Calculator) y permitía enfrentar un jugador humano contra la máquina.
1958	William Higginbotham crea, sirviéndose de un programa para el cálculo de trayectorias y un osciloscopio "Tennis for Two", que era un simulador de tenis de mesa y era utilizado por los visitantes del Brookhaven National Laboratory.
1962	Steve Russell desarrolla en seis meses utilizando gráficos vectoriales "SpaceWar!", donde dos jugadores controlaban la dirección y velocidad de dos naves espaciales que luchaban entre ellas. Este juego funcionaba sobre el procesador PDP-1.

- 1966 Ralph Baer empezó a desarrollar junto a Albert Maricon y Ted Dabney un proyecto de videojuego llamado "Fox and Hounds" dando inicio al videojuego doméstico. Este proyecto más tarde evolucionaría en la primer videoconsola llamada "Magnavox Odyssey".
- 1971 Nace la máquina recreativa "Pong", desarrollada por Al Alcorn para Nolan Bushnell en la recién fundada empresa Atari. Este juego era muy similar a "Tennis for Two" pero utilizada en lugares públicos como bares, aeropuertos, entre otros.
- 1972 Nace el videojuego Space Invaders, que se vuelve la piedra angular del videojuego como industria.
- 1983 Comienza la crisis del videojuego, la cual afecta a Estados Unidos y Canadá. Fue causado por la excesiva cantidad de videojuegos en las tiendas, los cuales tenían una baja calidad. Atari intenta solucionar el problema lanzando dos títulos que supuestamente iban a revolucionar el mercado:
- "E.T. El Extraterrestre: El Videojuego": Era y sigue siendo, sobre todo en la actualidad un videojuego pésimo, ya que la jugabilidad ni siquiera se entendía.
 - "Pac-Man": El juego que pusieron a la venta era un demo del mismo. Al creador apenas le dejaron acabarlo debido a las prisas.
- Durante esta crisis, se habla que Mattel y Coleco acumularon pérdidas cercanas a los 200 y 250 millones de dólares respectivamente. Grandes empresas como Magnavox se fue a la quiebra y en el caso de Atari, con más de 250 millones de dólares en pérdidas, fue vendida por Warner a Commodore por 240 millones de dólares.
- 1985 Acaba la crisis del videojuego y se adopta como principal sistema de videojuegos el NES (Nintendo Entertainment System).
- 1987 Comienzan a salir juegos clásicos como el Tetris de Alexey Pajitnov
- 1990 Las videoconsolas dan un salto técnico importante gracias a la llamada "Generación de 16 Bits" compuesta por la MegaDrive, el SNES, entre otros.
- 1994 Inicia la "Generación de 32 Bits" comandada por el Sega Saturn y el Sony PlayStation, años más adelante sale la competencia de Nintendo con Nintendo

64, entre otras.

- | | |
|------|---|
| 1998 | Se inicia la sexta generación de videojuegos llamada "Generación de 128 Bits" donde salen consolas como la Xbox de Microsoft, el Play Station 2 de Sony, el Game Cube de Nintendo, etc. |
| 2005 | Los videojuegos alcanzan su máximo esplendor con procesadores gráficos de 254 Bits, donde destacan el Xbox 360 de Microsoft y Nintendo DS. En este año nace la séptima generación de videojuegos. |
| 2006 | Sale al mercado las consolas de sobremesa como PlayStation 3 de Sony y Wii de Nintendo. Estas consolas pueden ejecutar juegos en alta definición e integran controles con sensores de movimiento. |
| 2010 | Viendo el éxito obtenido por Nintendo con sus controles con sensores de movimiento, Sony lanza el PlayStation Move, el cual incluye características de juego muy similares al Wii. Microsoft se une igualmente a Nintendo y lanza el Kinect, el cual vendió 8 millones de unidades en sus primeros 60 días en el mercado, ingresándolo en el Libro Guinness de los Records como el "más rápido en ventas de dispositivos electrónicos". |
| 2011 | Inicia la octava generación de los videojuegos, donde las consolas tienen un parecido a computadoras de alta gama. Las consolas sacadas al mercado en este año son PlayStation Vita de Sony y Nintendo 3DS XL. |
| 2012 | Como inicio de la octava generación en consolas de sobremesa, se lanza la consola WiiU, la cual contiene un procesador IBM PowerPC Tri-Core a 1.24 GHz y una tarjeta de gráficos AMD Radeon R700 serie 4xxx a una velocidad de 550MHz. |

2.3 Ciclo de Vida de los videojuegos

Los videojuegos como medio de entretenimiento han evolucionado de manera increíble en muy poco tiempo, se han convertido en una industria multimillonaria casi a la par con la industria del cine (Otter, 2008).

Actualmente las empresas, editoriales y grupos de desarrolladores independientes involucran a profesionales en campos como diseño, informática, música, marketing, etc. para lograr publicar un solo videojuego.

Durante muchos años, la industria se aferró a utilizar la metodología cascada y muchas compañías siguen creando productos de esta manera. Actualmente en el mercado no existen muchos procesos específicos para el desarrollo de videojuegos que sean públicos, aunque las grandes empresas de desarrollo de videojuegos tienen sus procesos “cerrados” en el sentido de que su información, modelos, plantillas y herramientas no están disponibles al público en general.

En las siguientes secciones presentaremos algunas metodologías utilizadas por empresas como Zynga para el desarrollo de videojuegos.

2.3.1 Scrum

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujiro Nonaka a mediados de los años 80. (Palacios, 2004)

En 1996 se presentó por Jeff Sutherland y Ken Schwaber como un proceso de desarrollo de software formal, en el OOPSLA 96. Más tarde, en 2001 serían dos de los promulgadores del Manifiesto Ágil.

El manifiesto ágil es un documento que resume en cuatro valores y doce principios las mejores prácticas para el desarrollo de software, basados en la experiencia de 17 industriales del software. En éste manifiesto se hace énfasis en cuatro valores principales que deben soportar el desarrollo de software: (Herrera & Valencia, 2007)

- Los individuos e interacciones por encima de los procesos y las herramientas: Para garantizar una mayor productividad, las metodologías ágiles valoran el recurso humano como el principal factor de éxito. Reconocen que contar con recurso humano calificado con capacidades técnicas adecuadas, facilidades para adaptarse al entorno, trabajar en equipo e interactuar convenientemente con el usuario, da mayor garantía de éxito que contar con herramientas y procesos rigurosos.
- Software funcionando por encima de la documentación: En este sentido, las metodologías ágiles respetan la importancia de la documentación como parte del proceso y del resultado de un proyecto de desarrollo de software, sin embargo, con la misma claridad hacen énfasis en que se deben producir los documentos estrictamente necesarios; los documentos deben ser cortos y limitarse a lo fundamental, dando prioridad al contenido sobre la forma de presentación.
- La colaboración del cliente por encima de la negociación del contrato: En este sentido, y complementando el valor que se da al trabajo en equipo, las metodologías ágiles incluyen de una manera directa y comprometida al cliente o usuario en el equipo de trabajo. Es un ingrediente más en el camino al éxito en un proyecto de desarrollo de software. Más que un ambiente de enfrentamiento en el cual las partes buscan su beneficio propio, evadiendo responsabilidades y procurando minimizar sus riesgos, bajo la filosofía de las metodologías ágiles se busca el beneficio común, el equipo de desarrollo y el del cliente.

La participación del cliente debe ser constante, desde el comienzo hasta la culminación del proyecto, y su interacción con el equipo de desarrollo, de excelente calidad.

- La respuesta al cambio por encima del seguimiento de un plan: Van desde ajustes sencillos en la personalización del software hasta cambios en las leyes, pasando por la aparición de nuevos productos en el mercado, comportamiento de la competencia, nuevas tendencias tecnológicas, entre otras. En este sentido, las metodologías tradicionales con frecuencia caen en la idea de tener todo completo y correctamente definido desde el comienzo. No se cuenta entre sus fortalezas la habilidad para responder a los cambios. Por el contrario, en las metodologías ágiles la planificación no debe ser estricta, puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para los siguientes meses.
- Scrum es una metodología de desarrollo muy simple, que requiere trabajo duro porque no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto.

La siguiente figura muestra de forma general la metodología Scrum para el desarrollo de un producto, cada una de las fases y sus características se describen a continuación.

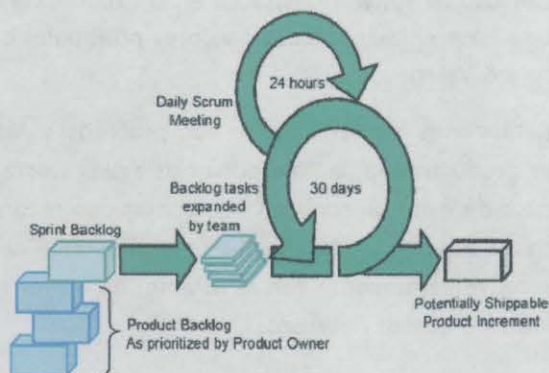


Figura 5: Ciclo de Vida de Scrum

2.3.1.1 Las Reuniones

- **Planificación de Sprint:** Jornada de trabajo previa al inicio de cada sprint en la que se determina cuál va a ser el trabajo y los objetivos que se deben cumplir en la iteración.
- **Reunión diaria:** Breve revisión del equipo de trabajo realizado hasta la fecha y previsión para el día siguiente.
- **Revisión de sprint:** Análisis y revisión del incremento generado.

2.3.1.2 Los elementos

- Pila del producto: Lista de requisitos de usuario que se origina con la visión inicial del producto y va creciendo y evolucionando durante el desarrollo.
- Pila del sprint: Lista de los trabajos que debe realizar el equipo durante el sprint para generar el incremento previsto
- Incremento: Resultado de cada sprint.

2.3.1.3 Ventajas de Scrum sobre otras metodologías ágiles

- Prácticas sencillas y fácil manejo de los productos de trabajo
- Resolución de problemas individuales y por equipo
- Requerimientos y desarrollo evolutivos e incrementales con un comportamiento adaptativo
- Participación directa y dirección del proyecto por parte del cliente
- Fácil combinación con otros modelos de administración de proyectos.
- Comunicación, aprendizaje y generación de valor por parte del equipo. (Larman, 2004)

2.3.2 Huddle

Su nombre nace a partir de la analogía con el fútbol americano, Huddle es una reunión que se realiza en el juego antes de cada jugada. La filosofía es que mediante breves reuniones de planeación a corto plazo, se planea cada “jugada” que se inicie; con esto se da un seguimiento más estrecho al avance del proyecto y es posible hacer correcciones tempranas a posibles desviaciones. (Morales, Nava, Fernández, & Rey, 2010)

Huddle es un proceso específico para el desarrollo de videojuegos que cumplan las siguientes especificaciones:

- Equipos multidisciplinarios de 5 a 10 personas
- Iterativo
- incremental
- Evolutivo

Este modelo de ciclo de vida está basado en la metodología Scrum aunque se adicionan algunos roles al proceso: Game Designer y Project Manager. El ciclo de vida de esta metodología se divide en 3 fases:

1. Preproducción: Tiene como objetivo migrar la idea del diseñador al Feature Log y al Sprint Plan; estos documentos darán la pauta a la planeación y producción del videojuego. Su propósito principal es el análisis del proyecto, es la fase en la que se revisará y se aceptará en su caso, la idea inicial y se realizará la planeación completa de la fase de producción.

2. Producción: Esta segunda etapa es la más importante y la más larga. Se apoya en herramientas del Scrum Framework (Reuniones diarias, los sprints y las estabilizaciones del Sprint) y en artefactos (Sprint Backlog y Burn-down Charts), aunque los roles cambian por un Project Manager y un Game Designer.

3. Postmortem: La etapa de postmortem consiste en generar un reporte cuyo propósito es describir a detalle las actividades específicas que fueron más efectivas para el proyecto desde el inicio del proceso hasta la entrega del producto; de igual manera, describe las actividades que llegaron a perjudicar el desarrollo junto con sugerencias para corregir dichos problemas.

Para realizar esta fase, el equipo debe realizar la última actividad llamada End-game Huddle en el cual se analizan los aspectos positivos y negativos del proyecto. Para el llenado de este artefacto se deben responder las siguientes preguntas:

- a. ¿Qué salió bien?
- b. ¿Qué salió mal?
- c. ¿Qué obstáculos se presentaron?

2.3.3 SUM

La metodología SUM para videojuegos tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia. Pretende obtener resultados predecibles, administrar eficientemente los recursos y riesgos del proyecto, y lograr una alta productividad del equipo de desarrollo. SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños (de 3 a 7 integrantes que trabajan en un mismo lugar físico o estén distribuidos), y para proyectos cortos (menores a un año de duración) con alto grado de participación del cliente.

La definición de la metodología se basa en el Software and Systems Process Engineering Metamodel Specification (SPEM) 2.0, un meta-modelo para describir procesos y metodologías desarrollado por el Object Management Group (OMG). Una ventaja de utilizar SPEM es que su estructura permite especificar el proceso de desarrollo de videojuegos sin mencionar prácticas específicas, lo que lo hace flexible y adaptable a cada realidad.

SUM adapta para videojuegos la estructura y roles de Scrum descrita por Ken Schwaber. Se utiliza ésta metodología, ya que brinda flexibilidad para definir el ciclo de vida y puede ser combinado fácilmente con otras metodologías para adaptarse a distintas realidades.

El ciclo de vida se divide en fases iterativas e incrementales que se ejecutan en forma secuencial con excepción de la fase de gestión de riesgos que se realiza durante todo el proyecto. Las cinco fases secuenciales son: concepto, planificación, elaboración, beta y cierre. Las fases de concepto, planificación y cierre se realizan en una única iteración, mientras que la elaboración y beta constan de múltiples iteraciones.

Las fases surgen como adaptación al desarrollo de videojuegos de las fases pre-game, game y post-game que presenta Scrum, donde las dos primeras coinciden con las fases de

planificación y elaboración, mientras que la tercera se corresponde con las fases de beta y cierre. Esta división se realiza ya que la fase beta tiene características especiales en la industria de videojuegos. La fase de concepto no corresponde con ninguna etapa de Scrum, y se agrega ya que cubre las necesidades específicas para el desarrollo de videojuegos. (Acetenza, Coppes, Mesa, Fernandez, Lorenzo, & Vallespir, 2009)

2.3.4 MCMI

Esta metodología para videojuegos inicia como un método para la creación de micro mundos inmersivos, de donde salen sus siglas. Para la formalización de la metodología, se aplicaron los estándares SPEM 2.0 definido por el consorcio OMG.

MCMI está basada en las metodologías ágiles de desarrollo de software, específicamente en Extreme Programing (XP), ya que XP tiene en cuenta aspectos relativos a la creación de videojuegos como: La colaboración continua, además de destacar la participación de los individuos pertenecientes al desarrollo del proyecto; se tiene en cuenta que los equipos de desarrollo al igual que los plazos de tiempo para obtener los productos, son bastante reducidos. (Valencia, Riascos, & Niño, 2011)

Esta metodología se definiría especialmente adecuada para proyectos donde los requisitos son imprecisos o cambiantes, además, de proyectos en los cuales exista un alto riesgo técnico y grupos reducidos para realizar el proyecto.

El objetivo principal de MCMI es tener un proceso estructurado de desarrollo que permita a los diseñadores y desarrolladores de videojuegos cambiar de opinión durante el proceso de desarrollo; que los videojuegos evolucionen a la par junto a la infraestructura tecnológica; que exista una estrecha colaboración entre el cliente y el equipo del proyecto, permitiendo así, la creación de videojuegos en los lapsos de tiempo estimados. (Valencia, Riascos, & Niño, 2011)

MCMI define seis fases: (Valencia, Riascos, & Niño, 2011)

- Fase 1.- Concepción: El propósito de esta fase, es tener una visión clara del contexto en el cual se va a enfocar el videojuego. Para ello, es necesario definir y comprender las necesidades del cliente.
- Fase 2.- Planeación: El objetivo de la fase de planeación es plantear los elementos necesarios para apoyar el proceso de enseñanza y aprendizaje, realizando dos tipos de diseños: El diseño del videojuego y el diseño educacional.
- Fase 3.- Iteración: El objetivo de la iteración, es desarrollar funcionalidades propias del videojuego, generando al finalizar de cada iteración, un entregable funcional de las historias de usuario realizadas en la fase de planeación.

El cliente en esta fase del proyecto, ha cumplido con su labor fundamental de escoger las historias de usuario que van a ser abordadas en la presente iteración, además, provee información detallada de características o datos que se deben tener en cuenta al realizar el análisis; una vez realizado el análisis de la historia de usuario, se realizan los diseños

artísticos y musicales necesarios para satisfacer las necesidades plasmadas en la historia de usuario actual.

- Fase 4.- Alfa: El objetivo de la fase Alfa es dar a conocer el producto a un grupo selecto de evaluadores, los cuales, ofrecerán retroalimentación y sugerencias referentes al videojuego. La información adquirida por el grupo de evaluadores, es necesaria para llevar a cabo cambios sustanciales sobre el producto. Ésta fase es necesaria para obtener información útil acerca de la experiencia global del videojuego, en aras de mejorar aspectos de enseñanza y aprendizaje, además de aspectos concernientes a la usabilidad y jugabilidad.
- Fase 5.- Beta: El objetivo de la fase Beta es liberar el producto a un público más amplio que en la fase Alfa, por lo general, con poco conocimiento acerca del videojuego realizado. El grupo al cual se libera el producto, necesariamente corresponde al público objetivo establecido en la fase de concepción.
- Fase 6.- Cierre: El objetivo de la fase de cierre es evaluar los sucesos significativos a lo largo del transcurso del proyecto y entregar la versión final del juego al cliente.

El uso de MCMI permite planificar un proceso estructurado de desarrollo a los diseñadores y desarrolladores de videojuegos. Por lo anterior, las actividades que ofrece MCMI en cada una de sus fases, permiten obtener documentación estrictamente pertinente en cada etapa del ciclo de vida, software funcional en lapsos cortos de tiempo, estrecho vínculo con los interesados del proyecto, evolución a la par con las nuevas tecnologías, logrando de esta forma, que los productos obtenidos al aplicar este método se caractericen por poseer practicas propias de los videojuegos.

2.4 Dispositivos Móviles

Hoy en día, el ser humano se mueve rápido. Vivimos en un mundo en el que el estrés se alza como claro dominador, un mundo en el que la pérdida de un segundo puede ser vital en muchos sentidos, un mundo en el que el hombre pretende llegar a todas partes, pero, ante todo, se trata de un mundo en el que el ansia de satisfacer todos estos requerimientos nos ha llevado a reducir al hombre y a la maquina en un mismo ser, como dos partes de un todo. (Felix, 2007)

Actualmente uno de los dispositivos que se han vuelto parte fundamental en la vida de las personas son los teléfonos móviles. Tan solo en México, el número de líneas de telefonía móvil pasó de 55.4 millones que se tenían en el 2006 a 97.6 millones durante el primer semestre de 2012. (AFP, 2012)

Definimos teléfono móvil o celular como un dispositivo electrónico de comunicación, normalmente de diseño reducido, sugerente y basado en la tecnología de ondas de radio (radiofrecuencias) que tiene la misma funcionalidad que cualquier teléfono de línea fija. Su rasgo característico principal es que se trata de un dispositivo portable e inalámbrico, esto es, que la realización de llamadas no es dependiente de ninguna terminal fija y que no requiere de ningún tipo de cableado para llevar a cabo la conexión a la red telefónica.

Además de ser capaz de realizar llamadas como cualquier otro teléfono convencional, un celular más o menos moderno suele incorporar un conjunto de funciones adicionales, tales como

mensajería instantánea (sms), agenda, juegos, etc., que aumentan la potencialidad de utilización de estos dispositivos. Es más, su desarrollo y exigencia ha llegado a tal punto, que ya se puede hablar de términos tales como memoria RAM para información de todo tipo (audio, video, texto, etc.), lo que hace de éstos dispositivos un complemento perfecto para la vida diaria. (Felix, 2007)

2.4.1 Generaciones de la Telefonía Celular

2.4.1.1 Generación Cero (0G)

La 1G de la telefonía móvil hizo su aparición en 1979 si bien proliferó durante los años 80. Introdujo los teléfonos "celulares", basados en las redes con múltiples estaciones de base relativamente cercanas unas de otras, y protocolos para el "traspaso" entre las celdas cuando el teléfono se movía de una celda a otra.

La transferencia analógica y estrictamente para voz son características identificadoras de la generación. Con calidad de enlaces muy reducida, la velocidad de conexión no era mayor a 2400 Bauds. En cuanto a la transferencia entre celdas, era muy imprecisa ya que contaban con una baja capacidad basadas en la tecnología FDMA (Frequency Division Multiple Access), lo que limitaba de forma notable la cantidad de usuarios que el servicio podía ofrecer en forma simultánea ya que los protocolos de asignación de canal estáticos padecen ésta limitación.

Con respecto a la seguridad, las medidas preventivas no formaban parte de esta primitiva telefonía celular. La tecnología predominante de esta generación es AMPS (Advanced Mobile Phone System), desarrollada principalmente por Bell. Si bien fue introducida inicialmente en los Estados Unidos, fue usada en otros países en forma extensiva. Otro sistema conocido como Sistema de Comunicación de Acceso Total (TACS) por sus siglas en inglés fue introducido en el Reino Unido y muchos otros países.

Si bien había diferencias en la especificación de los sistemas, eran conceptualmente muy similares. La información con la voz era transmitida en forma de frecuencia modulada al proveedor del servicio. Un canal de control era usado en forma simultánea para habilitar el traspaso a otro canal de comunicación de serlo necesario. La frecuencia de los canales era distinta para cada sistema. NMT usaba canales de 12.5 KHz, AMPS de 30 KHz y TACS de 25 KHz.

A su vez, el tamaño de los aparatos era mayor al de hoy en día; fueron originalmente diseñados para el uso en los automóviles. Motorola fue la primera compañía en introducir un teléfono realmente portátil.

Estos sistemas (NMT, AMPS, TACS, RTMI, etc.) fueron conocidos luego como la Primera Generación (G1) de teléfonos celulares.

En Septiembre de 1981 la primera red de telefonía celular con Roaming automático comenzó en Arabia Saudita; siendo un sistema de la compañía NMT. Un mes más tarde los países Nórdicos comenzaron una red NMT con Roaming automático entre países. (Molina, Maldonado, & Leidy, 2010)

2.4.1.2 Primera Generación (1G)

Si bien el éxito de la 1G fue indiscutible, el uso masivo de la propia tecnología mostró en forma clara las deficiencias que poseía. El espectro de frecuencia utilizado era insuficiente para soportar la calidad de servicio que se requería. Al convertirse a un sistema digital, ahorros significativos pudieron realizarse. Un número de sistemas surgieron en la década de los 90 debido a estos hechos, y su historia es tan exitosa como la de la generación anterior. La Segunda Generación (2G) de telefonía celular, como puede ser la GSM, IS-136 de TDMA, la iDEN & IS-95 de CDMA comenzó a introducirse al mercado.

La primera llamada digital entre teléfonos celulares fue realizada en Estados Unidos en 1990. En 1991 la primera red GSM fue instalada en Europa.

La generación se caracterizó por circuitos digitales de datos conmutados por circuito y la introducción de la telefonía rápida y avanzada a las redes. Usó a su vez acceso múltiple de tiempo dividido (TDMA) para permitir que hasta ocho usuarios utilizaran los canales separados por 200 MHz. Los sistemas básicos usaron frecuencias de banda de 900 MHz, mientras otros de 1800 y 1900 MHz.

Nuevas bandas de 850 MHz fueron agregadas en forma posterior. El rango de frecuencia utilizado por los sistemas 2G coincidió con algunas de las bandas utilizadas por los sistemas 1G desplazándolos rápidamente.

La introducción de esta generación trajo la desaparición de los "ladrillos" que se conocían como teléfonos celulares, dando paso a pequeñísimos aparatos que entran en la palma de la mano y oscilan entre los 80 y 200 gr. Se hicieron algunas mejoras en la duración de la batería empleando tecnologías de bajo consumo energético.

El sistema 2G utiliza protocolos de codificación más sofisticados y se emplea en los sistemas de telefonía celular actuales. Las tecnologías predominantes son: GSM (Global System for Mobile Communications), IS-136 (también conocido como ANSI-136) y CDMA (Code Division Multiple Access).

Los protocolos empleados en los sistemas 2G soportan velocidades de información por voz más altas, pero limitados en comunicación de datos. Se pueden ofrecer servicios auxiliares, como datos, fax y SMS (Short Message Service). La mayoría de los protocolos de 2G ofrecen diferentes niveles de cifrado. En Estados Unidos y otros países se le conoce a la generación 2G como PCS (Personal Communication Services). (Molina, Maldonado, & Leidy, 2010)

2.4.1.3 Segunda Generación (2G)

Una vez que la segunda generación se estableció, las limitantes de algunos sistemas en lo referente al envío de información se hicieron evidentes. Muchas aplicaciones para transferencia de información eran vistas a medida que el uso de laptops y el propio internet se fueron popularizando. Si bien la tercera generación estaba en el horizonte, algunos servicios se hicieron necesarios previa a su llegada. El GPRS (General Packet Radio Service) desarrollado para el sistema GSM fue de los primeros en ser visto. Hasta este momento, todos los circuitos eran dedicados en forma exclusiva a cada usuario. Este enfoque es conocido como "Circuit Switched", donde por

ejemplo un circuito es establecido para cada usuario del sistema. Esto era ineficiente cuando un canal transfería información solo en un pequeño porcentaje. El nuevo sistema permitía a los usuarios compartir un mismo canal, dirigiendo los paquetes de información desde el emisor al receptor. Esto permite el uso más eficiente de los canales de comunicación, lo que habilita a las compañías proveedoras de servicios a cobrar menos por ellos.

Aún más cantidad de mejoras fueron realizadas a la tasa de transferencia de información al introducirse el sistema conocido como EDGE (Enhanced Data rates aplicado a GSM Evolution). Éste básicamente es el sistema GPRS con un nuevo esquema de modulación de frecuencia.

Mientras GPRS y EDGE se aplicaron a GSM, otras mejoras fueron orientadas al sistema CDMA, siendo el primer paso el CDMA a CDMA2000 1x.

2.5G provee algunos de los beneficios de 3G (por ejemplo conmutación de datos en paquetes) y puede usar algo de la infraestructura utilizada por 2G en las redes GSM y CDMA. La tecnología más comúnmente conocida de 2.5G es GPRS, que provee transferencia de datos a velocidad moderada usando canales TDM no utilizados en la red GSM. Algunos protocolos, califican oficialmente como servicios "3G" debido a que su tasa de transferencia de datos supera los 144 Kbit/s, pero son considerados por la mayoría como servicios 2.5G o 2.75G, porque son en realidad varias veces más lentos que los servicios implementados en una red 3G.

Mientras los términos 2G y 3G están definidos oficialmente, no lo está 2.5G, ya que fue inventado solo con fines únicamente publicitarios.

Muchos de los proveedores de servicios de telecomunicaciones se moverán a las redes 2.5G antes de entrar masivamente a la red 3G ya que la tecnología 2.5G es más rápida y más económica para actualizar que la tecnología 3G.

2.4.1.4 Tercera Generación (3G)

No mucho luego de haberse introducido las redes 2G se comenzó a desarrollar los sistemas 3G. Como suele ser inevitable, hay variados estándares con distintos competidores que intentan que su tecnología sea la predominante. Sin embargo, en forma muy diferencial a los sistemas 2G, el significado de 3G fue estandarizado por el proceso IMT-2000. Este proceso no estandarizó una tecnología, sino una serie de requerimientos que se debían cumplir como 2Mbit/s de máxima tasa de transferencia en ambientes cerrados y 384 Kbit/s en ambientes abiertos. Hoy en día, la idea de un único estándar internacional se ha visto dividida en múltiples estándares bien diferenciados entre sí.

Existen principalmente tres tecnologías 3G. Para Europa existe la tecnología Universal Mobile Telecommunication System (UMTS) que funciona utilizando los protocolos CDMA de banda ancha (W-CDMA); este sistema provee transferencia de información de hasta 2 Mbps

Están a su vez las evoluciones de CDMA2000. La primera en ser lanzada fue CDMA2000 1xEV-DO, donde EV-DO viene de las siglas en inglés de Evolution Data Only. La idea atrás de este sistema era que muchas de las aplicaciones solo requirieran conexión de datos, como sería el caso si se usara el celular para conectar una computadora a internet de forma inalámbrica. En caso de requerir además comunicación por voz, un canal 1x estándar es requerido. Además de usar

tecnología CDMA, EV-DO usa tecnología TDMA para proveer de la velocidad de transferencia necesaria y mantener la compatibilidad con CDMA y CDMA2000 1x.

La siguiente evolución de CDMA2000 fue CDMA2000 1xEV-DV. Esto fue una evolución del sistema 1x totalmente distinto a CDMA2000 1xEV-DO, ofreciendo servicios totales de voz y datos. Este sistema también es compatible con CDMA y CDMA2000 1x y es capaz de ofrecer tasas de transferencia de 3.1 Mbps

Estos dos protocolos usaron lo que se conoce como FDD (Frequency Division Duplex), donde los links de ida y vuelta usan distintas frecuencias. Dentro de UMTS existe una especificación conocida como TDD (Time Division Duplex), donde los links poseen la misma frecuencia pero usan distintos segmentos de tiempo. Sin embargo, TDD no se implementará en los mercados por un tiempo.

Un tercer sistema 3G fue desarrollado en china que usa TDD, es conocido como TD-SCDMA (Time Division Synchronous CDMA), que usa un canal de 1.6 MHz y que está pensado para abarcar el mercado chino y de sus países vecinos.

Algunos de los sistemas 2.5G como CDMA2000 1x y GPRS, proveen de algunas de las funcionalidades de 3G sin llegar a los niveles de transferencia de datos o usos de multimedia de la nueva generación. Por ejemplo, CDMA2000 1x puede, en teoría, transferir información hasta a 307 Kbit/s. Justo encima de esto se encuentra el sistema EDGE, el cual puede en teoría superar los requerimientos de los sistemas 3G; aunque esto es por tan poco que cualquier implementación práctica quedaría probablemente por debajo del límite deseado.

Al comienzo del siglo XXI, sistemas 3G como UMTS y CDMA2000 1xEV-DO han comenzado a estar al alcance del público en los países de primer mundo, sin embargo, el éxito de estos sistemas aún está por probarse. (Molina, Maldonado, & Leidy, 2010)

2.4.2 Sistemas Operativos Móviles

Cada una de las plataformas tiene sus particularidades, no solo en cuanto al manejo del dispositivo por el usuario, sino también a la hora de desarrollar aplicaciones para las mismas.

El desarrollo de aplicaciones que tengan compatibilidad para todos los Sistemas Operativos que existen en el mercado es complejo, debido a que cada uno de ellos maneja su propio lenguaje de programación. Tomando en cuenta esto, en las siguientes secciones se presentan los sistemas operativos que tienen un mayor dominio del mercado en el mundo:

2.4.2.1 Windows Mobile

Es el sistema operativo en el que están basados la mayoría de PocketPCs. Este sistema se pretende vender como una versión muy aligerada de Windows, pero en realidad no tiene mucho que ver con él. El paradigma de funcionamiento es similar, basado en ventanas, aunque éstas se comportan de forma muy diferente a como lo hacen en su hermano mayor de escritorio.

El punto fuerte de estos dispositivos es que ofrecen funcionalidades similares a las de sus hermanos mayores. Por ejemplo, se pueden editar documentos de Word, hojas de cálculo de Excel, leer libros en formato PDF, enviar y recibir correos electrónicos, entre otras cosas. En definitiva, hace casi lo mismo que una PC, pero sin teclado y con la pantalla bastante pequeña.

El principal problema a la hora de desarrollar aplicaciones para estos dispositivos es el precio de las herramientas de desarrollo. Estas herramientas (Visual Studio, un entorno de desarrollo que soporta varios lenguajes como C#, C++, J#, JScript o ASP .NET) son de altísima calidad, pero de un precio que no está al alcance de todos los que se quieran aventurar a escribir una aplicación para PocketPC.

Por otra parte, el hecho de que Microsoft esté detrás de la plataforma, dándole todo su apoyo, se nota en la cantidad y calidad de la documentación disponible para los desarrolladores. Newsletters, una sección solo para dispositivos en MSDN, blogs de los ingenieros de Microsoft que trabajan en Windows Mobile, actualizaciones de la documentación en DVDs, periodicos, etc. (Taráguila, 2009)

2.4.2.2 Symbian

Symbian es un consorcio en el que participan los mayores fabricantes de teléfonos móviles, con Nokia a la cabeza. Soy Ericsson, BenQ, Fujitsu, Lenovo, Motorola, Panasonic, Samsung o Sharp están representados en Symbian, y son por lo tanto, partícipes en el desarrollo y la expansión del sistema.

De todos estos fabricantes, el que mayor cuota de mercado tiene es Nokia, que con algunas fluctuaciones, viene siendo desde hace años de alrededor de 40%, que además es el fabricante que primero apostó por este sistema operativo, y el que más uso hace de él.

Symbian es un sistema operativo escrito en C++, por lo que presenta muy bajo consumo de recursos del dispositivo, a la vez que se ejecuta con gran rapidez. El sistema operativo Symbian se presenta en varios "sabores". En concreto, Nokia divide sus dispositivos Symbian en tres familias, que se llaman respectivamente Serie 40, Serie 60 y Serie 80.

Aparte de la implementación que hace Nokia de Symbian, Sony Ericsson implementa otro sabor más de ese sistema, llamado UIQ, que se maneja por teclado o a través de una pantalla táctil. Estos dispositivos suelen implementar sistemas de reconocimiento de escritura, y lo más conocidos son, la serie P de Sony Ericsson. (Taráguila, 2009)

2.4.2.3 iPhone iOS

IOS, anteriormente llamado iPhone OS, es un sistema operativo móvil de Apple desarrollado originalmente por iPhone, siendo utilizado por los dispositivos iPhone, iPod Touch y iPad. Es un derivado de Mac OS X, que a su vez está desarrollado en Darwin BSD.

El sistema operativo iOS tiene cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de servicios principales, la capa de medios de comunicación y la capa de Cocoa Touch.

Todo el sistema se encuentra en la partición "root" del dispositivo, y ocupa aproximadamente la mitad de la capacidad del dispositivo.

Esto se realizó para poder soportar futuras aplicaciones de Apple, también como aplicaciones de terceros publicadas en la iTunes Store o App Store.

El iOS está en el mercado con muchas características interesantes y útiles. iOS 4 lleva más de 100 nuevas características, ajustes y mejoras. A continuación se mencionan las más importantes:

- **Multitarea (Multitasking):** Esta característica la habían publicado en el año 2007, pero había desaparecido. Ahora su publicación es oficial, y es capaz de navegar por internet, hacer tweets y actualizar el estado de Facebook al mismo tiempo.
- **Múltiples cuentas de Exchange:** Esta característica tiene la libertad de integrar cuentas de Microsoft Exchange como el usuario lo desee. También puede sincronizar cuentas de correo electrónico como GMail, MobileMe, etc.
- **Wi-Fi estable en modo de espera:** Es una gran mejora para las personas que utilizan dispositivos iPod Touch. Con ésta pueden recibir llamadas de Skype y otros servicios de VoIP en modo de suspensión.
- **Soporte de Bluetooth en el teclado:** iOS4 es compatible con el teclado Bluetooth Tethering.
- **iOS 4 SSL VPN:** Se abrió esta introducción de SSL VPN de acceso que antes no tenía, con la cual se asegura el acceso VPN del navegador y otras aplicaciones basadas en la red.

La ventaja del sistema operativo iOS es que tiene un muy buen diseño, funcionalidad, facilidad de usabilidad y una gran variedad en sus aplicaciones. Aunque sus desventajas son que al ser un sistema de Apple no se deja modificar, por lo que hay menos posibilidades de cambiar la forma de funcionar el dispositivo y tener un control más rígido de sus aplicaciones publicadas, además de que el iPhone tiene un costo bastante alto en el mercado, ya que tiene un solo fabricante y un modelo único en el mundo. (Aponte & Dávila, 2011)

2.4.2.4 Android

Es el sistema operativo más vendido a nivel mundial con un 52.5% de ventas mundiales por encima de Symbian y iOS de Apple, lo cual está revolucionando actualmente el mercado mundial del Smartphone, haciendo de este sistema operativo móvil un gigante de la industria de las telecomunicaciones.

El elemento más notable de Android es que es una plataforma de código abierto y cualquier elemento que haga falta o se pierda puede ser provisionado por un sinfín de usuarios que hacen parte de la comunidad Android a nivel mundial.

Los nombres que fueron asociados a sus diferentes versiones tales como Donut, Eclair, Froyo, Gingerbread, etc, lo hacen un sistema operativo especial dado la originalidad utilizada en los mismos.

Google compró la empresa Android Inc. en el inicio de 2005 para empezar el desarrollo de una nueva plataforma llamada Android y entre las piezas claves se incluyó a Andy Rubin, Nick Sears y Chris Blanco.

A finales de 2007, un grupo de líderes de la industria se reunieron alrededor de la plataforma Android y formaron la Alianza Open Handset de la cual era miembros importantes empresas como: Sprint Nextel, Toshiba, T-Mobile, Vodafone, Motorola, Google, Samsung, Intel, Sony Ericsson y Texas Instruments.

Parte del objetivo de la alianza era innovar con rapidez y responder mejor a las necesidades del consumidor y ese resultado fue la primera clave de la plataforma Android.

Android es un entorno de software integrado para dispositivos móviles y no es una plataforma de hardware como lo puede pensar mucha gente. Incluye un sistema Linux basado en el kernel del mismo sistema operativo. La interfaz de usuario es una plataforma muy rica en aplicaciones para el usuario final, bibliotecas de código, entornos de aplicaciones, soporte multimedia y mucho más, incluyendo por supuesto la funcionalidad de telefonía celular.

Una de las características importantes de la plataforma de Android es que no hay diferencia entre las aplicaciones incorporadas en el sistema operativo y las aplicaciones que se crean con el SDK, esto significa que el desarrollador puede escribir aplicaciones de gran alcance para aprovechar los recursos disponibles en el dispositivo.

Android es únicamente un software que aprovecha su kernel Linux para interactuar con el hardware y se ejecuta en diferentes dispositivos celulares de múltiples fabricantes. Generalmente las aplicaciones puestas en el mercado móvil son generadas por sus desarrolladores en Java. (Aponte & Dávila, 2011)

2.5 Usabilidad e Iteración Humano Computadora

En la actualidad se encuentra que la adquisición de productos o servicios depende en gran medida del nivel de calidad alcanzado por cada uno de ellos. En la industria de software se han definido varios aspectos que se deben medir a la hora de definir la calidad de un producto, tales como: Confiabilidad, Seguridad, Usabilidad y Rendimiento.

Hablando específicamente de usabilidad, existen factores principales que deben considerarse como la facilidad de aprendizaje, la efectividad de uso y la satisfacción con la que las personas son capaces de hacer sus tareas gracias al uso del sistema con el que están trabajando.

Se pueden asociar cada medida a las dimensiones o conceptos relacionados con la medición de la usabilidad, lo cual lleva a una clasificación de dichas medidas a un nivel de abstracción más general. Dicha clasificación arroja como resultado las categorías de aspectos de usabilidad que se deben medir:

A nivel de Eficiencia

- Eficiencia con respecto al tiempo
- Eficiencia de la interfaz
- Facilidad de Aprendizaje-comprensión

A nivel de la Efectividad

- Efectividad con respecto a las tareas
- Efectividad con respecto a los errores
- Efectividad de la interfaz

A nivel de Satisfacción

- Facilidad de uso
- Navegabilidad
- Nivel de uso
- Velocidad
- Calidad de la interfaz
- Calidad de los datos

Durante la búsqueda de patrones de usabilidad e Interacción Humano-Computadora se encontraron patrones de diseño de interfaces graficas que prometen un alto grado de usabilidad, los cuales especifican qué elementos deberían estar presentes en la interfaz y de qué manera; algunos otros patrones indican que elementos no deberían incluirse en una interfaz; otros patrones son un poco más explícitos e indican la manera de cómo se deben implementar, teniendo en cuenta la plataforma, el dispositivo y demás.

Ya que este proyecto propone un ambiente virtual, es necesario también tener en cuenta los patrones ajustables a los frameworks, que señalan ciertos elementos que deberían incluirse en su definición, de modo que sea fácilmente navegable por los usuarios y que, además de representar fielmente la metáfora, agregan mejoras para que la experiencia sea satisfactoria.

Algunos patrones HCI más utilizados en el mercado al momento de diseñar y construir interfaces graficas usables para el framework son (Fakhreddine, Alemzadeth, Saleh, & Arab):

➤ Aplicables a interfaces graficas en General

- *El Escudo*: El usuario puede accidentalmente seleccionar una función que tiene efectos irreversibles, este patrón aplica para administración de errores, se implementa normalmente con una ventana o conjunto de ventanas donde se pide al usuario la confirmación de la ejecución de estos procesos.
- *Filtrado Progresivo*: Permite al usuario alcanzar su objetivo mediante la aplicación de filtros secuenciales. Es decir, la información se puede categorizar y mostrarle al usuario ciertas opciones, esto se hace iterativamente hasta que finalmente llegue a la información que busca.
- *Progreso*: Muestra al usuario si aún se está ejecutando o no una tarea, en algunos casos se incluye información como los pasos restantes para completar una tarea o el tiempo que falta.

➤ Aplicables a interfaces gráficas 2D

- *Migajas de pan*: Ofrecen en la interfaz gráfica un rastro de navegación desde la ventana inicial hasta la actual.
- *Dispositivos de entradas unidireccionales*: Permiten que el usuario pueda utilizar el mouse o el teclado para acceder a las opciones y facilitar su uso.

➤ Utilización de Metáforas (Ferré, 2010)

- *Combinación de códigos comunicativos distintos*: visual (icónico), verbal, auditivo. El objetivo es conseguir una redundancia óptima en el proceso de comunicación.
- *Organización y estructuración de la interfaz en niveles jerárquicos*.
- *La utilización de metáforas en el proceso comunicativo mediante las posibilidades de los entornos gráficos*: emplear símbolos usuales en la vida cotidiana de los usuarios.

- En el diseño de las interfaces graficas de usuario, existen tantos modelos con los que trabajar como los que existen en situaciones de la vida real. Modelos de metáfora: un escritorio, una ciudad, etc. Aunque se necesita el diseño de la información para la interfaz más sencilla, los proyectos de la envergadura de una interfaz gráfica de usuario *de una ciudad se tienen que basar en el diseño de la información.*
- Otra estrategia alternativa es realizar un estudio exhaustivo de los conceptos implicados en un dominio dado, y proceder sistemáticamente a su representación icónica.
- Procurar ayudas textuales y gráficas, tanto generales como locales.
- Procurar ayudas procedimentales, simulando el asesoramiento de un experto, ya sea mediante procedimientos encubiertos o transparentes al usuario.

En la investigación es incluso más importante
El proceso que el logro mismo.

-- Emilio Muñoz

Capítulo 3. Desarrollo

En este capítulo se presenta el proceso de desarrollo empleado para Cubecraft Game, el cual se desarrolló utilizando la metodología ágil Scrum. El desarrollo tomó un tiempo aproximado de cuatro meses, donde se realizaron tres sprints con una duración de un mes cada uno, finalizando con un mes de pruebas de rendimiento, pruebas de estrés y estabilización del sistema.

Las tecnologías a emplear fueron seis: Java + XML, Java 2 Micro Edition, C# .NET, PHP 4, Ajax v. 0.5 y MySQL, con una distribución como la mostrada en la figura 6.



Figura 6: Distribución de Tecnologías

Se utilizó C# para el desarrollo del framework, donde se van a poder utilizar código desarrollado en Android o en J2ME para ensamblar las clases y generar el juego, aunque una de las funcionalidades que se tiene es puntajes en línea, por lo cual, se generó un Web Service en PHP que utilizando Ajax lo comunica con la base de datos, reduciendo el tiempo de bloqueo de las tablas.

3.1 Product Backlog

Para poder realizar el Product Backlog, se tuvieron que identificar las necesidades del cliente y crear las User Stories. Estas User Stories contemplan tareas que son un beneficio para el cliente o para el desarrollador.

En total se detectaron 30 User Stories, de las cuales seis son técnicas y 24 de usuario que tienen un total de 164 tareas. Para tener un control de las tareas a realizar por cada lenguaje de programación a emplear, se maneja un identificador formado por el nombre del lenguaje y

seguido por un número identificador consecutivo. Esto nos permite conocer cuántas tareas se tienen por cada lenguaje de programación y balancear cada sprint con tareas de todos los lenguajes.

Después de completar cada tarea, se debe realizar las pruebas unitarias, para comprobar que se cumplen sus objetivos. Las pruebas se llevan a cabo en dos fases, en la primera fase el desarrollador revisa si todos los casos de prueba son satisfactorios, mientras que en la segunda fase se le pide a una persona ajena al desarrollo que pruebe la funcionalidad sin ninguna guía de pruebas.

Una vez identificadas todas las User Stories, se planearon el número de sprints necesarios para desarrollar toda la funcionalidad y el tiempo que deben durar. Se tomó la decisión de manejar tres Sprints con una duración de un mes cada uno, donde el número de tareas a resolver cambia hasta formar el mes de trabajo.

En cada uno de los Sprint, la sumatoria del tiempo estimado de las tareas debe ser mayor o igual a 3500 minutos. Este tiempo corresponde aproximadamente a 7 días hombre de trabajo ininterrumpido, sin tomar en cuenta tiempos de pruebas para algunas tareas.

Se espera que se terminen las tareas del Product Backlog en un tiempo máximo de 11580 minutos que corresponden a un tiempo aproximado de 24 días hombre de desarrollo, y se espera tener un porcentaje de desviación no mayor al 5%.

Un día hombre corresponde a una jornada laboral de 8 horas diarias, pero debido a otras actividades, el factor de dedicación que se tiene para los Sprints es del 60%. Por lo cual el tiempo de duración del desarrollo será de aproximadamente 39 días hombre.

3.2 Primer Sprint

El primer sprint inicia el día 1 de Septiembre del 2012 y finaliza el 28 de Septiembre del mismo año, donde se realizaron un total de 67 tareas que corresponden a los lenguajes J2ME, Android, PHP y C#.

Las primeras tareas que se realizaron fueron las que se desprendieron de las User Stories Técnicas. Estas User Stories se planean al conocer los detalles de las herramientas de trabajo a utilizar, y consistían en la descarga, instalación y configuración de los ambientes de trabajo para cada lenguaje de programación.

El primer problema que se encontró al instalar las herramientas es la versión sobre las que trabaja. Este problema ocurrió con las plataformas para Symbian, debido a que cuando se encontraba en auge, los procesadores más empleados en el mercado eran de arquitectura de 32 bits.

Para poder llevar a cabo la instalación del ambiente de trabajo en procesadores con arquitectura de 64 bits, fue necesario instalar la versión del Kit de Desarrollo de Java para 32 bits, y con esto poder simular que el sistema trabaja bajo ésta plataforma.

La generación de la librería para Symbian estuvo dividida en tres fases. La primera fase consistía en la generación de librerías para el control de hardware del dispositivo, la segunda en la generación de herramientas de funcionalidad para el dispositivo con la finalidad de poder utilizar todas las características del teléfono y la última fase en la generación de interfaces gráficas y funciones que sean visibles en la pantalla del dispositivo.

En la primera fase, se escribieron clases abstractas para el control de las dimensiones y resolución de la pantalla, esta clase ayuda al dispositivo a conocer las capacidades de pantalla y la forma en la que debe ajustar los elementos cuando se utilice una resolución diferente a la usada en las pruebas.

La siguiente clase generada es un patrón de diseño, basado en el modelo Singleton que permitiera generar la información que se mostrará en la pantalla del dispositivo de una manera ordenada, al igual que se generó un "Garbage Collector" utilizando como base el nativo que trae Java, esto con la finalidad de permitir cerrar la aplicación y eliminar la información generada desde cualquier punto del juego y de esta forma ahorrar batería del dispositivo.

Una vez que se tenía la base de los juegos para Symbian, se generó una clase no abstracta que permitía funcionar el cualquier juego sin necesidad de inicializar, ya que el constructor recibe como firma un arreglo de pantallas que se pueden mostrar durante la ejecución del juego y la pantalla inicial de la aplicación.

Estas funciones no pueden probarse desde un principio, por lo cual, sus pruebas se llevaron a cabo al final del sprint con pruebas unitarias. En las pruebas unitarias, se tuvieron que corregir algunos problemas de buffering y de tiempo de refresh, debido a que la velocidad empleada era lenta, lo que daba la sensación al usuario de ir en cámara lenta.

En la segunda etapa, se generaron las utilerías como alertas independientes, fuentes, sockets de conexión HTTP, manejo de memoria estática, envío de mensajes, manejo de sonidos en formatos WAV y MP3, y utilerías de color y fuentes.

La clase de Utilerías de color, permite oscurecer y aclarar colores basados en la paleta de colores RGB. La paleta de colores RGB en Java se utiliza comúnmente en formatos numéricos hexadecimales, donde cada dos dígitos representa cada uno de los colores primarios (Rojo, Verde y Azul), el rango de valores aceptados es de 0 a F, donde F es igual a 255.

En la tercera etapa se manejaron métodos separados por cada funcionalidad que se quiera que haga el sistema. Con esto, se permite que el usuario pueda configurar la clase a su gusto de forma transparente, permitiendo que el código sea entendible y fácil de mantener.

Una vez finalizado el desarrollo de la primera funcionalidad para Symbian, se procedió a desarrollar la misma funcionalidad pero para dispositivos Android. Al trabajar de esta forma, se puede avanzar en ambas librerías al mismo tiempo, teniendo la misma funcionalidad una con otra.

Al igual que como se hizo con Symbian, el desarrollo de Android fue dividido en 3 fases. En la primera se escribieron controladores para los componentes periféricos del dispositivo, en la segunda se escribieron clases que optimizaban el uso de la capacidad del dispositivo y de sus

componentes, mientras que en la última etapa se desarrolló la información que se puede mostrar en la pantalla del dispositivo.

Una de las ventajas que se tuvo al desarrollar la funcionalidad de Android fue su capacidad de convertir el código desarrollado en librería, lo que permitió heredar configuración mantenida en el manifiesto XML hacia los proyectos posteriores. La configuración que se escribió en el manifiesto de la librería fue permitir que la aplicación pudiera utilizar la conexión a internet ya sea mediante 3G o Wi-Fi, y también el uso de pantalla completa en el dispositivo.

Durante la primera fase del desarrollo de la librería Android, se generaron clases de control para todos los componentes periféricos comunes en éstos dispositivos. La finalidad de hacer esto, es conocer la capacidad del dispositivo antes de iniciar la aplicación y permitir que otras aplicaciones puedan seguir utilizando los demás componentes si no van a ser usados por el juego.

Una vez terminados los controladores, se generaron clases que optimizan su uso y la obtención de información proveniente de éstos componentes. Para no gastar tantos recursos del dispositivo, primero se hace un análisis de la versión de Android que se utiliza, y a partir de ello indicar el número de información máxima que se guardará por componente.

Ya terminada la optimización de los controladores, se comienza a desarrollar las clases gráficas. Primero se generó el patrón de diseño Singleton para las pantallas, después se generaron los elementos gráficos y una clase genérica para mostrar de forma automática las pantallas generadas.

Ya generadas éstas dos librerías, se comenzó con el desarrollo del Sistema de Puntajes en Línea. Primero se analizó la información que se iba a poder guardar en el sistema y la información que se iba a manejar en el sitio web, debido a que la base de datos debe ser íntegra desde su conceptualización.

Una vez terminado el análisis de los datos, se generó un diagrama Entidad-Relación en la herramienta MySQL Workbench, donde se muestra de forma gráfica las relaciones que existen entre cada una de las tablas, para después normalizarla hasta tercera forma normal (3FN).

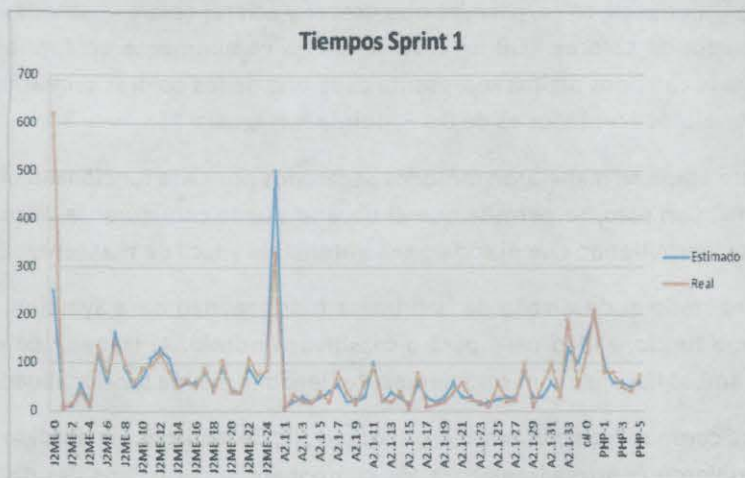


Figura 7: Tiempo Estimado vs Tiempo Real Primer Sprint

Debido a que se tratan de tareas que por sí mismas no pueden realizar una acción fue necesaria terminar la librería de cada uno de los lenguajes para realizar las pruebas. Durante estas pruebas se tuvieron que corregir algunos problemas de buffering y de tiempo de refresh, debido a que la velocidad empleada era lenta, lo que daba la sensación al usuario de ir en cámara lenta.

Ya finalizadas todas las User Stories, se tuvo una desviación en las estimaciones de 6.36%, que corresponde a un aproximado de 4 horas y 45 minutos, que debe tomarse en cuenta para el segundo Sprint (Figura 12). Esto se debe al mal cálculo de algunas tareas o complicaciones en otras como por ejemplo, en la configuración de los ambientes de trabajo.

3.3 Segundo Sprint

El segundo sprint comienza el 1ro de Octubre y termina el 31 de Octubre con un total de 47 tareas de los lenguajes PHP y C#.

La funcionalidad que se trabajó en PHP consiste en la realización de un socket de conexión entre MySQL y PHP, que permitirá obtener y guardar datos de los juegos del servidor de base de datos con el que se trabaja.

Una vez terminada esta tarea, se desarrollaron funciones que emplearan Ajax para obtener el nombre de un jugador y el puntaje que guardó. Esta User Story es necesaria para mostrarla al usuario cada vez que quiera consultar el puntaje mayor que tenga su juego.

Para comprobar que el Web Service se encuentra activo y listo para recibir peticiones, se hizo un método que hace le hace una petición. Si el Web Service se encuentra activo, responde la petición al dispositivo, en caso contrario, el dispositivo que realizó la petición informa al usuario de que no puede acceder al servicio y continúa su ejecución.

El Web Service se encuentra activo dentro del sitio web del proyecto (<http://cubegame.luminisoft.com.mx/webService.php>) y pide los siguientes valores pasados por medio del método post:

- Activity: En esta variable se indica el tipo de petición que se hará al sistema, las opciones disponibles son: "LoadScores" para pedir los puntajes y nombres de los jugadores con mayor puntaje, "saveScore" para guardar un puntaje dentro de la base de datos, "loadUser" para cargar un perfil de jugador, "saveUser" para generar un nuevo jugador dentro del juego y "ping" el cual se encarga de conocer el estado en el que se encuentra el Web Service.
- Game: En esta variable se espera el usuario indique el nombre del juego que desea cargar. Si el nombre del juego no existe, entonces el sistema regresa una lista vacía. Para poder dar de alta un juego, se debe de hacer directamente desde el sitio web.
- Limit: Esta variable permite indicar al Web Service el máximo de registros que quiere mostrar el usuario al jugador. En el caso de que el número de registros sea menor al límite que se indica, entonces se mandan todos los puntajes guardados.
- Score: Este valor es opcional, y es usado cuando un usuario desea ingresar un puntaje al sistema. El puntaje es ingresado sea mayor o no. Para que el sistema muestre los mejores

puntajes, se realiza una búsqueda ordenada por el puntaje de forma descendente en la base de datos, usando como límite la variable indicada anteriormente.

Ya teniendo todas las herramientas necesarias para poder desarrollar aplicaciones para Android, se comenzó con el desarrollo de la aplicación. Se tuvo que tomar la decisión de qué versión del framework de .NET se iba a utilizar, por lo cual se eligió la versión 4.0 por las siguientes razones:

- Se mejora el rendimiento de las aplicaciones, lo cual nos ayuda a liberar los recursos del equipo tan rápido como el proceso termine.
- Permite la utilización de enlace dinámico de clases, con lo que podemos utilizar la encapsulación de conceptos para su uso en poca memoria temporal.

Aunque en el momento de la creación del proyecto ya se encontraba activa la versión 4.5 del framework, se decidió mantenerse en la versión 4 debido a la compatibilidad que tiene esta versión en equipos con Sistemas Operativos anteriores a Windows Vista.

Antes de comenzar el desarrollo de la aplicación, es necesario tener un diseño de la interfaz y conocer la funcionalidad que tendrá cada uno de los elementos. Para obtener una interfaz gráfica estable, se utilizaron algunas de las buenas prácticas establecidas en los patrones de Interacción Humano-Computadora. Los patrones utilizados son: Interfaces Gráficas Generales, Interfaces Gráficas para aplicaciones 2D y el Uso de Metáforas.

Del patrón de Interfaces Gráficas Generales, se usaron la práctica de escudo y la práctica de línea de progreso. Por otra parte, de los patrones de las Interfaces Gráficas para aplicaciones 2D se utilizaron todas las prácticas nombradas y de los patrones de Uso de Metáforas se utilizó la combinación de códigos comunicativos distintos.

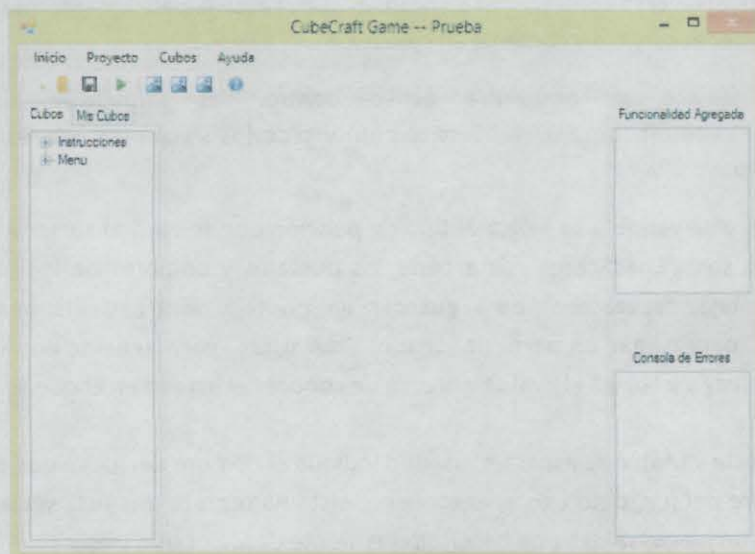


Figura 8: Interfaz Cubecraft Game

La primera pantalla con la que interactúa el usuario es la pantalla principal, y es la encargada de realizar todo el trabajo de diseño del videojuego. Su diseño se muestra en la figura 6, y se delimitan 4 zonas: el menú de opciones, Cubos de trabajo, Área de trabajo y las Consolas de Ayuda al Usuario.

En la zona de menú de opciones, se encuentran las barras de menú y de acceso rápido que permitirán al usuario realizar ciertas acciones. En ésta barra se tienen los siguientes elementos:

- Inicio: En este menú se encuentran las opciones básicas de todo sistema, como crear un nuevo proyecto, abrir un proyecto existente, guardar los cambios, exportar a una imagen y salir.
- Proyecto: Este menú se habilitará cuando el usuario abra o genere un proyecto nuevo. Aquí se encuentran las opciones de deshacer cambios, rehacer el último cambio, entrar a las propiedades del proyecto, compilar un proyecto y reiniciar el área de trabajo. Esto último pedirá confirmación, ya que una vez realizada la acción no podrá utilizarse las opciones de Deshacer o Rehacer para regresar las funcionalidades.
- Cubos: Al igual que el menú Proyecto, éste menú se habilitará hasta que el usuario genere o abra un proyecto. Aquí se pueden llevar a cabo acciones que permitan al usuario importar, eliminar o modificar cubos al sistema. Estas funcionalidades están destinadas para su uso por desarrolladores con experiencia en los lenguajes de programación soportados.
- Ayuda: En este menú se localizan todas las opciones de ayuda para el usuario, como lo es un manual de uso, una sección de Preguntas Frecuentes (FAQ's) e información sobre la versión del framework.

Las funcionalidades de abrir y crear, antes de mostrar un cuadro de dialogo al usuario, es necesario verifiquen se encuentren todos los archivos necesarios para su uso. En cualquiera de los casos, el sistema debe buscar archivos con extensión .CUB dentro del espacio de trabajo del framework. Este tipo de archivo contiene toda la información del proyecto y la configuración del framework para un óptimo funcionamiento del sistema.

La segunda sección son los cubos de trabajo, donde se pueden observar dos vistas: Cubos y Mis Cubos. En la primera opción se muestran como una lista contraíble, los cubos que trae por default el sistema o que se hayan obtenido de alguna actualización, por ejemplo, las opciones para hacer un menú, pantallas de configuración, pantallas informativas, herramientas de sonido, entre otras.

En la pestaña de Mis Cubos se muestran los cubos que el usuario haya agregado al framework y que coincidan con las características de proyecto. En esta pestaña se muestran todas aquellas funciones que el usuario haya desarrollado con el menú Agregar Cubo visto anteriormente o descargado desde el sitio oficial.

La tercera sección de la pantalla principal es el área de trabajo. Esta área comienza estando en blanco cuando el usuario crea un nuevo proyecto, y es el lugar donde los cubos podrán

conectarse unos con otros para generar la funcionalidad del juego. La retroalimentación que regresará el framework sobre la correcta elección de un cubo será por el curso del mouse.

Para poder agregar un cubo de la zona de cubos de trabajo al área de trabajo, el usuario debe seleccionar una de las opciones mantener presionado la opción y arrastrarla al área de trabajo. En caso que el usuario quiera soltar el cubo de funcionalidad sobre cualquier otra zona que no sea la destinada para ello, el cursor cambiará a una X y al soltar el cubo no ocurrirá nada.

La posición de conexión de los cubos no es específicamente en vertical u horizontal, sino que el usuario puede utilizar cualquiera de los lados para unirlos, el único lado en donde no se permite se haga unión de cubos es en su cara anterior, debido a que el sistema está diseñado para un uso 2D y no 3D.

La cuarta zona es la consola de ayuda al usuario, esta zona está dividida en 2 secciones: Ayuda al Usuario con la Funcionalidad Incluida en su Desarrollo y una Consola de Errores. Estas consolas están pensadas para dar retroalimentación suficiente al usuario y permitirle hacer su desarrollo de forma rápida y sencilla.

La consola de Funcionalidad Agregada se actualiza cada vez que el usuario agrega un cubo al área de trabajo, indicando el nombre del cubo que el usuario le asignó y el tipo de cubo que le corresponde, utilizando la misma connotación de los diagramas de objetos de UML, los cuales consisten en indicar primero el nombre del objeto seguido por dos puntos y después el tipo de objeto.

Cada vez que el usuario agrega funcionalidad a cada uno de los cubos raíz, como por ejemplo, las pantallas en blanco, la lista se volverá una vista de árbol, que le mostrará al usuario cada una de las funciones agrupadas según su cubo contenedor.

La consola de errores, como su nombre lo indica, permite al usuario saber los errores o posibles problemas que no permitan que se compile el juego o que no realice las acciones que debe realizar. Esta sección se implementa debido a la tendencia de los frameworks disponibles en el mercado de indicarle al desarrollador el lugar y el error que comete, y de esta forma ahorrarle tiempo en sus correcciones.

Para identificar si uno de los errores son críticos o solo advertencias, en la parte izquierda del aviso se muestra una imagen de color amarillo y una admiración para las advertencias y una imagen roja con una X para los errores.

Cuando se selecciona la opción de Nuevo Proyecto, ya sea mediante una de las opciones del menú o de los accesos rápidos, se muestra una pantalla como la mostrada en la figura 7, en la cual se puede escribir el nombre del proyecto y elegir el sistema operativo que se está empleando.

En esta ventana, se trata que la interfaz sea intuitiva para el usuario, ya que no se utilizan tecnicismos que pudieran confundirlo. Además, la ventana tiene la funcionalidad de indicarle al usuario si el nombre del proyecto existe o no.

Como se puede ver en la figura, no se muestra alguna opción para saber la dirección donde se guardará el proyecto, esto es porque el framework genera su propio ambiente de trabajo dentro de la carpeta "Mis Documentos".

En la parte superior del cuadro de texto del nombre del proyecto, se maneja retroalimentación si el proyecto existe o no. Cuando se abre la ventana, se muestra en letras rojas la frase "Indique un nombre de proyecto" lo cual significa que el campo del nombre permanece vacío.

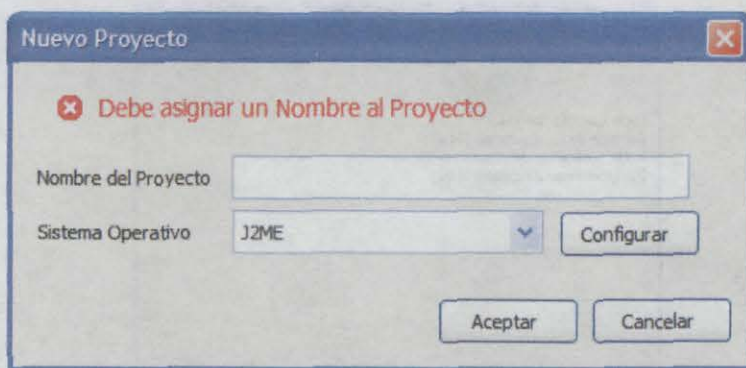


Figura 9: Interfaz Nuevo Proyecto

Cuando se comienza a escribir el nombre del proyecto, el sistema busca en el espacio de trabajo de disco proyectos que tengan ese nombre. En caso de que el nombre del proyecto ya exista, se marca con letras de color rojo y en fuente grande la frase "El Proyecto ya Existe" y deshabilitará el botón de Aceptar la creación de proyecto. Si el nombre de proyecto no existe, entonces se marca con texto en color verde la frase "Todo parece ir bien" y permitirá presionar el botón de Aceptar.

Una vez presionado el botón aceptar, el sistema genera una carpeta dentro del ambiente de trabajo con el nombre del proyecto. Dentro de ésta carpeta, se tienen las carpetas SRC y Entrega y un archivo con extensión .CUB que lleva el mismo nombre del proyecto.

La carpeta SRC contiene todas las clases generadas por el framework y que se compilan para generar el juego. Dependiendo de la complejidad y el sistema operativo del dispositivo elegido va a ser el número y las extensiones de los archivos que se generan.

En la carpeta Entrega se encuentra la aplicación ya compilada y lista para entrar al mercado, al igual que en la carpeta anterior, los archivos pueden tener varios tipos de extensión, entre los que destacan las extensiones .JAR para dispositivos Symbian y .APK para los dispositivos Android.

El archivo que se genera en la raíz del directorio contiene toda la información de configuración del proyecto, y es el que detecta el framework cuando se quiera abrir para modificarlo. Por razones de seguridad e integridad de los datos contenidos, el usuario debe abstenerse a modificarlo o eliminarlo, ya que puede causar que el proyecto no funcione más y que se pierda el trabajo realizado.

Una vez que el usuario ha terminado de agregar cubos al área de trabajo, puede probar la funcionalidad de su juego seleccionando la opción "Compilar" del menú o de la barra de accesos rápidos. Esta función mostrara una ventana parecida a la mostrada en la figura 8.

Esta pantalla de compilación cumple con la mayoría de las características del diseño de interfaces gráficas por metáforas, ya que muestra una barra de progreso con el avance de la compilación y ejecución del juego. Así mismo, se tiene un área de texto donde se especifican las acciones que se están llevando a cabo y la hora del sistema en la que comenzó o terminó el proceso.

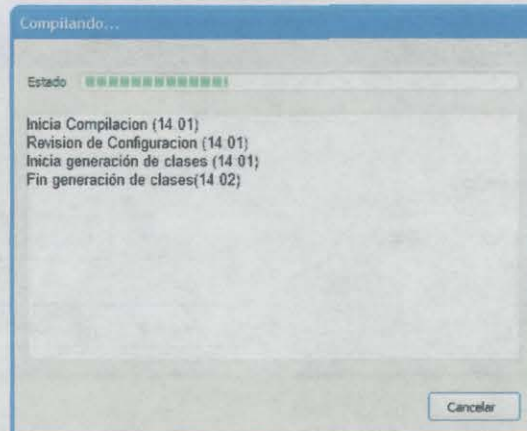


Figura 10: Interfaz de Compilación

La barra de progreso llegará a tres cuartas partes cuando termine la compilación del proyecto y la última cuarta parte corresponde a la configuración y ejecución del simulador según el sistema operativo.

El tiempo de ejecución de las tareas dependerá de las capacidades del equipo en el que se esté ejecutando el framework. Puede llegar a ocurrir el caso de que alguno de los simuladores no pueda funcionar debido a las capacidades del equipo. Cuando suceden estos casos, se generará el archivo ejecutable para ser publicado, pero no se iniciará el simulador y en la parte donde se indican las acciones del compilado se mostrará al usuario el problema que ocurrió.

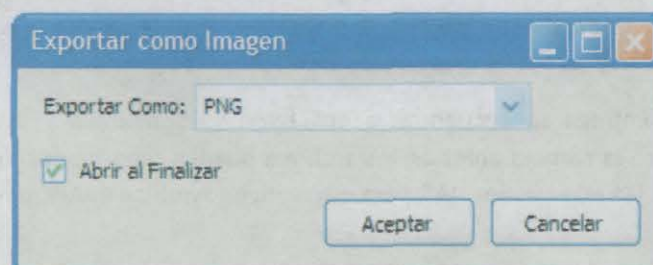


Figura 11: Interfaz para Exportar como Imagen

Siguiendo los ciclos de vida de los videojuegos, se agrega una funcionalidad al framework, el cual permite documentar sus desarrollos generando una imagen de los cubos utilizados. Al seleccionar la opción, se abre un cuadro de dialogo como se muestra en la figura 9, donde se tiene la opción de elegir el formato de la imagen, se plantea el uso de extensiones JPG o PNG debido a su popularidad entre los usuarios promedio. Además se agregará una opción de Visualización, la cual mostrará la imagen una vez generada y guardada en disco.

Al seleccionar la opción "Aceptar", se busca dentro de la carpeta del proyecto que exista una subcarpeta llamada "Imágenes". Dentro de esta carpeta, se mantienen todas las imágenes que se generen, cuyos nombres consisten en la palabra IMG seguido del número de imagen generada en el proyecto y la fecha y hora en la que se generó la imagen.

Los cubos de funcionalidad son cubos que representan cierta acción que se puede llevar a cabo durante la ejecución del juego. Se eligieron cubos debido a su facilidad para generar efectos de profundidad utilizando imágenes en dos dimensiones. La inclinación de cada uno de los cubos permite generar un efecto de ensamblado entre ellos.

Se manejan 3 colores de cubos basados en los colores de semáforos vehiculares (Figura 12). Se eligieron esos colores debido a su uso en la vida diaria de las personas, haciendo que no sea necesaria una explicación del porque se puso de cierto color.

Los cubos pueden cambiar de color dependiendo de las siguientes situaciones:

- Cubo Rojo: Cuando se pone el cubo de este color, quiere decir que hay un error grave en su implementación, que puede ir desde que no se encuentra un cubo inicial o que la configuración de la funcionalidad se encuentra incompleta.
- Cubo Amarillo: Este color indica que se tiene una advertencia que puede afectar la funcionalidad del juego. Esta advertencia normalmente se presenta cuando el desarrollador crea un cubo que no está conectado con ningún otro o cuando le falta configurar algún parámetro que no permita que se haga correctamente la acción.
- Cubo Verde: Cuando el cubo se pone de este color, quiere decir que todo se encuentra bien y sin problemas. Para llegar a este estado, es necesario haber conectado el cubo correctamente y no debe haber faltas de configuraciones del cubo.

Es fácil diferenciar entre los cubos raíz y los cubos de funcionalidad, con la información que muestran. Los cubos raíz normalmente solo piden el nombre del cubo, mientras que los cubos de funcionalidad tienen más campos para completar su configuración.

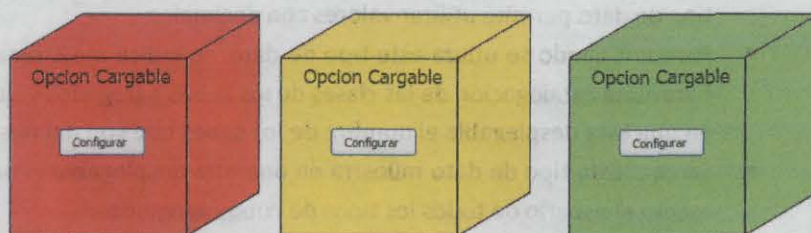


Figura 12: Cubos de Funcionalidad

Por estética de los cubos, se agregó un botón a la cara posterior del cubo con el nombre de "Configurar", que al presionarlo muestra una ventana emergente que se auto configura con los parámetros especificados en el archivo de funcionalidad.

Ya definida la versión de .NET a emplear y el diseño de la interfaz, se generaron las carpetas que contendrían toda la información del proyecto. La carpeta Clases contienen todas las

clases que se utilizan como tipo de dato general o que realizan una tarea repetitiva durante la ejecución del sistema, mientras que la carpeta Controls contiene todos los componentes que se crearon para facilitar el ingreso de datos al usuario final, y la carpeta Forms que contiene todos los formularios que serán cargados desde la ventana inicial.

Con la finalidad de darle al usuario todo el espacio posible para que pueda trabajar, se hizo que la pantalla inicial se abriera maximizada, esto hace que las áreas que se encuentran alrededor del área de trabajo se reajusten para darle el mayor espacio al usuario.

Debido a que se deben organizar por tipo las funcionalidades del sistema, se utilizó un TreeView en la zona de cubos, mientras que se utilizaron Listas para las consolas de errores y de funcionalidad. Para las listas, se agregó un manejador de excepciones, ya que si el usuario desea dar un clic sobre la lista y no hay elementos, puede causar un error crítico en el sistema.

Antes de comenzar a desarrollar la funcionalidad de catalogación de cubos, se tuvo que buscar un patrón para los archivos que contienen las funciones. Los elementos que tiene este diseño son los siguientes:

- Os: En este parámetro se debe indicar cuál es el sistema operativo que se va a utilizar. En todas las funciones donde se utilice este valor debe ser idéntico, para que el sistema no lo catalogue como un sistema operativo aparte.
- Class: El parámetro que recibe sirve para llevar a cabo la organización de los elementos, al igual que con el sistema operativo, el sistema es sensitivo.
- Function: Para las funciones se pide se indique el nombre con el que se quiera mostrar dentro del árbol de opciones dentro del sistema.
- Parameters: En los parámetros se piden tres datos, el primero es el nombre del parámetro y es la forma en la que se muestra al usuario, el segundo es el tipo de dato y el último es la prioridad del atributo.

Los tipos de datos que se encuentran disponibles son:

- String: Este tipo de dato permite que se ingrese el texto que el usuario quiera.
- Int: Se muestra un componente de selección donde solo puede ingresar valores numéricos.
- Float: Muestra una interfaz igual que el tipo int, con la diferencia que éste tipo de dato permite utilizar valores con decimales.
- Parent: Cuando se utiliza este tipo de dato, se indica al sistema que debe hacer una catalogación de las clases de los cubos agregados y que muestre en una lista desplegable el nombre de los cubos que son del mismo tipo.
- Screen: Este tipo de dato muestra en una lista desplegable el nombre que asignó el usuario de todos los tipos de cubos agregados.
- Color: Al usar este tipo de dato, se muestra un botón con el texto "Examinar" que abre una ventana de selección de color. Al seleccionar el color y presionar el botón "Aceptar" el fondo del botón cambia al color que se seleccionó.

- Image: Este tipo de dato muestra un cuadro de dialogo donde se debe seleccionar una imagen en formato PNG.
 - Padkey: Contiene una colección de los botones de un control básico que se muestra como una lista desplegable.
 - GameAction: Muestra en una lista desplegable los movimientos que se pueden realizar como por ejemplo mover hacia arriba o disparar.
 - Boolean: Gráficamente se muestra como un cuadro de selección donde se pueden elegir sus dos estados: True o False.
 - BarAlignment: Cuando se usa este tipo de dato, se muestra una lista desplegable donde se tienen alineaciones ya definidas.
 - EnemyAutoMove: Al igual que con BarAlignment, cuando es empleado se muestra una lista desplegable que contiene valores ya definidos.
- Code: En esta parte se indica la función tal y como se encuentra definida en la librería para que el sistema la utilice.

Para catalogar y mostrar al usuario las funcionalidades por clase, se utilizó un algoritmo de árboles binarios con podado alfa-beta, donde se catalogan las funciones primero por sistema operativo y después son catalogadas por clase, una vez terminada la catalogación y basados en la elección del sistema operativo del usuario, se elimina la información de memoria temporal del sistema operativo no usado y se muestra al usuario la parte del árbol que quedó.

Ya una vez que se permite mostrar el árbol de opciones al usuario, se desarrolló la parte de código que permite al usuario tomar una funcionalidad y arrastrarla al área de trabajo, para posicionar el cubo en la posición del cursor se tuvo que convertir la posición del cursor a posición de controlador. Esto es debido a que la posición de controlador está manejando una resolución diferente a la que utiliza el monitor de la computadora del usuario.

Ya que el cubo es posicionado, se permitió que el usuario al presionar el botón de "Configurar" o al dar clic derecho y seleccionar la opción "Configurar" mostrara una ventana donde se muestran los atributos del cubo. La generación de esta ventana es dinámica y depende de los datos leídos en el archivo de configuración.

Cuando un atributo no cumple con todos los requerimientos en el archivo de configuración o el tipo de dato indicado no existe, entonces ese atributo es ignorado y no se muestra al usuario en la ventana de configuración.

Una vez terminadas todas las tareas de este segundo sprint, se tuvo un tiempo de desarrollo ininterrumpido de 3731 minutos, lo cual es aproximadamente equivalente a ocho días hombre de trabajo. Comparándolo con el tiempo planeado de éste Sprint, se tiene una desviación de 211 minutos que corresponde al 5.99%.

La desviación de éste Sprint fue debido a una mala planeación de algunas tareas, que consumieron más tiempo del necesario y por el aprendizaje y adaptación de algunos algoritmos para el lenguaje c#.

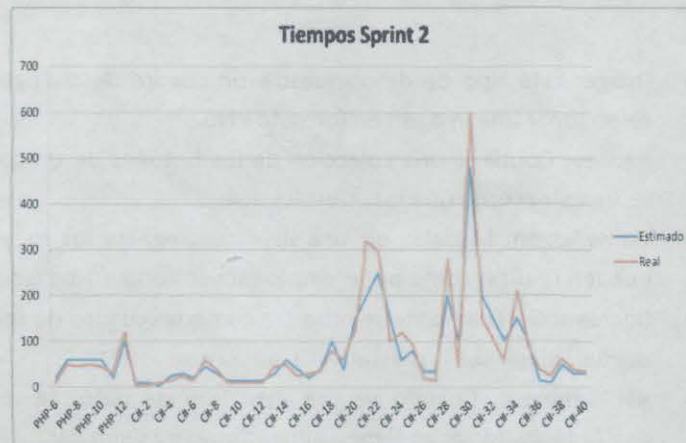


Figura 13: Tiempo Estimado vs Tiempo Real Segundo Sprint

3.4 Tercer Sprint

Para el tercer sprint que comprende del 1 al 30 de Noviembre, se realizaron las tareas que permiten generar el juego utilizando el framework. Estas tareas estuvieron estimadas en un tiempo de 3585 minutos, que corresponden a ocho días hombre de desarrollo ininterrumpido, completando un total de 50 funcionalidades.

Antes de comenzar la creación de los archivos de compilación, se tuvo que investigar el uso de Android desde la consola de sistema y los pasos que se debe seguir para la compilación exitosa de las aplicaciones. En Android, las herramientas con las que se cuenta se dividen en dos carpetas: Herramientas Generales (tools) que tiene las herramientas para la creación de proyectos y la forma de empaquetar los archivos APK, y Herramientas de la Plataforma (platform-tools) que contiene los elementos para la compilación de los archivos Dalvik (DEX) y el control de los emuladores.

Para la compilación de proyectos, se deben utilizar las herramientas de Java como de Android. Con las herramientas de Java se debe crear los archivos binarios soportados por las máquinas virtuales java y con las herramientas Android se deben convertir éstos archivos binarios a archivos DEX, el cual es la extensión que puede interpretar la máquina virtual Dalvik (DVM).

A continuación se explicará la forma de generar los distribuibles Android desde la concepción de la estructura, para ello se utilizarán nomenclaturas de comandos. Todos los comandos se encuentran negritas y con cursiva, si alguno de sus atributos se encuentra dentro de corchetes ([]) significa que es un valor opcional y puede omitirse sin causar algún error.

3.4.1 Generando los proyectos Android

Las herramientas generales de Android permiten generar el proyecto en la ruta que se indique. Un proyecto Android consta de archivos y carpetas que cumplen un objetivo en particular:

- **AndroidManifest.xml:** Este archivo presenta información esencial acerca de la aplicación al sistema Android, la cual debe ser cargada antes de la ejecución. Usualmente contiene información como: El nombre del paquete principal, los permisos que tiene la aplicación, los procesos que se ejecutarán y la forma, entre otros.
- **Src/:** Contiene los códigos que se encuentran en extensión Java. Estos archivos se encuentran organizados en carpetas llamados paquetes, los cuales usan un tipo de nomenclatura llamada DNS Inverso; esto consiste en escribir los nombres de los paquetes basándose en los nombres de dominios registrados.
- **Res/:** Contiene los recursos provistos por los usuarios como pueden ser las cadenas de caracteres, imágenes, menús o formatos de diseño.
- **Obj/:** Esta carpeta contendrá todos los archivos con terminación .class y todos aquellos archivos secundarios que genere las herramientas de Java.
- **Lib/:** Contendrá todas las librerías no pertenecientes a Android y que se encuentren en formato .jar.
- **Bin/:** Contendrá todos los archivos intermedios y los ejecutables finales que se producen con las herramientas de Android.
- **Docs/:** Este directorio es opcional, y contiene toda la documentación (javadoc) del proyecto. Si es generada esta carpeta, el archivo principal de la documentación debe ser *docs/index.html*.

Existen dos formas de crear estos archivos, creándolos manualmente o haciéndolo mediante los comandos de Android. Si los archivos son creados manualmente, se debe cuidar que se mantengan todos los lineamientos de nomenclatura utilizado por Android, ya que al cambiar algún nombre o la estructura de algún archivo puede causar que el proyecto no compile.

Para este proyecto, se utilizarán en todo momento las herramientas de Android ejecutadas desde línea de comandos (MS-DOS). El comando para crear un proyecto es ***android [opciones globales] action [opciones de acción]***.

Las opciones globales que puede tener este proyecto son: *-s* cuando se quiere solo se muestren los errores, *-h* para imprimir la lista de comandos que se pueden utilizar o *-v* para mostrar toda la información. La acción que se utilizará para la creación del proyecto es *create Project* y las opciones disponibles para su manejo son: *-n* para asignar el nombre del proyecto, *-t* para la versión de Android que se va a utilizar *-k* para indicar el paquete principal, *-a* para la actividad por default y *-p* para indicar la ruta donde se va a guardar el proyecto.

```
android -v create project -n MiJuegoEnAndroid -t "android-7" -a
MiJuego -k com.cubecraft -p c:/MisProyectos/MiJuegoEnAndroid
```

Código 1: Creación Proyecto Android

El código 1, muestra un ejemplo de la creación de un paquete Android, el cual recibe el nombre de MiJuegoAndroid, utilizando como versión de compilación la versión 7 de Android que corresponde a Android 2.1 Eclair y generando la actividad principal MiJuego. Si se desea tener una lista de las versiones android disponibles se puede utilizar el comando ***android list targets***.

3.4.2 Generando clases del proyecto

Ya teniendo el proyecto, se genera un archivo con terminación .java con el nombre de la actividad, que deberá modificarse con el código que se ejecutará. En el caso de nuestro sistema, primero se debe copiar la librería de Android hacia el proyecto y después generar el código que se ejecutará.

Se debe copiar la librería debido a que contiene todas las clases y funciones que Android utilizará en tiempo de ejecución, si estos archivos son omitidos entonces el dispositivo no tendría la forma de identificar y ejecutar los comandos.

Ya copiada la librería, deben generarse las carpetas que Android no generó y que se utilizarán para ligar y generar los archivos secundarios. Las carpetas que se generan son: Obj y Assets. Esta carpeta funciona como una segunda carpeta de recursos, donde las imágenes son tratadas por igual, no importando si el dispositivo es de alta definición o con pantalla de cierta resolución.

Teniendo las carpetas creadas y la librería copiada, se debe comenzar a generar el código fuente. La creación del código está dividida en cuatro partes:

- Creación de Cabeceras: Esta parte del código consiste en indicar a que paquete va a pertenecer la clase. Como en nuestro caso se va a utilizar la actividad, se debe indicar el nombre del paquete principal y para evitar algún problema de importación de librerías, se utilizarán todas las clases contenidas en la librería.

```
code.Add("package com.upaep.thesis;\n\n");
code.Add("import com.upaep.thesis.classes.*;\n");
code.Add("import com.upaep.thesis.framework.*;\n");
code.Add("import com.upaep.thesis.library.*;\n");
code.Add("import com.upaep.thesis.utils.*;\n\n");
```

Código 2: Importación de la Librería

- Creación de Objetos: Para generar los objetos, es necesario identificar los cubos maestros, asignarles un nombre y agregarlos al proyecto. Para identificar los cubos maestros se hace una validación donde el nombre de la función del cubo sea igual al nombre de la clase.

```
foreach (Cube cube in cubes)
{
    ClassTree information = cube.FunctionInformation;
    String value = searchParameter("name", cube);

    if(information.ClassName.CompareTo(information.FunctionName) == 0)
    {
        code.Add("private " + information.ClassName + " " +
            value + " = new " + information.ClassName + "(this);\n");
    }
}
```

Código 3: Análisis de Cubos Maestros

- Generación de Funcionalidad: Una vez generados los objetos, es necesario cargar la información que se tiene de cada cubo utilizando el atributo Pertenece. Para no causar errores en el formato de los tipos de datos, se hizo que cuando el tipo de dato fuera una cadena de caracteres o una imagen encerrara el valor otorgado por comillas dobles.

```
String function = searchParameter("pertenece", cube) + "." +
information.Code + "(";

...

if(parameter.Type.CompareTo("String") == 0 ||
parameter.Type.CompareTo("string") == 0)
{
function += "\"" + parameter.Value + "\"";
}
else if(parameter.Type.CompareTo("Image") == 0 ||
parameter.Type.CompareTo("image") == 0)
{
function += "\"" + Path.GetFileName(parameter.Value) + "\"";
}
```

Código 4: Asignación de la configuración a la funcionalidad de los cubos

- Asignación de inicio: En esta parte del código, se debe hacer una revisión de si cual cubo está marcado como cubo principal. Al igual que cuando se genera un cubo, se deben separar los cubos de funcionalidad a los cubos maestros y verificar su estado de principal. Cuando es elegido más de uno como cubo principal, se toma en cuenta el primero y los demás son ignorados.

```
String returnStatement = "return ";
code.Add("public Screen getStartScreen()\n{\n");

foreach (Cube cube in cubes)
{
if(cube.StartScreen)
{
returnStatement += searchParameter("name", cube) + ";";
break;
}
}

code.Add(returnStatement);
code.Add("\n}\n");
```

Código 5: Detección de la pantalla principal del juego

3.4.3 Compilando el proyecto utilizado

Debido a la versión de Android que se utiliza, es necesario utilizar Java 1.6.0 revisión 26, ya que Android 2.1 solo es compatible con esta versión de Java. Si se quiere hacer la compilación de las clases utilizando una versión más reciente de Java, los archivos .class generados no serán compatibles durante la conversión al formato DEX y el proyecto no funcionará.

Para la compilación se llevan a cabo dos pasos. El primer paso para la compilación de un proyecto Android es convertir los archivos con terminación .java a archivos con extensión .class. Para llevar a cabo esta acción se debe utilizar el comando **javac [opciones] [archivos fuentes] [argumentos]**. Donde las opciones a utilizar pueden ser **-classpath** para indicar la ruta donde se encuentran las librerías de compilación, **-sourcepath** indica donde se deben buscar las clases o los las librerías de ejecución, con el parámetro **-d** indicamos la ruta donde se quiere que se guarden los archivos .class generados y **-verbose** para indicar si se desea mostrar todo el proceso de compilación.

```
Javac -verbose -d "c:/MisProyectos/MiJuegoEnAndroid/obj" -classpath
"c:/Program Files/Android/Android-sdk/platforms/android-
7/android.jar";"c:/MisProyectos/MiJuegoEnAndroid/obj" -sourcepath
"c:/MisProyectos/MiJuegoEnAndroid/src" "MiJuegoEnAndroid"
```

Código 6: Compilación de archivos .class utilizando javac

En el ejemplo de código anterior, indicamos a Java que vamos a compilar las clases de Android utilizando sus librerías localizadas en las plataformas descargadas. Este paso solo lo hacemos con la clase que es la actividad Android, de esta forma se generarán los archivos .class de todas aquellas clases que se encuentren involucradas en la clase.

Una vez que se tienen los archivos .class, se comienza con el segundo paso de compilación, donde se deben convertir a un formato que pueda leer la máquina virtual de Android. Para hacer la conversión a este tipo de archivo se utiliza el comando **dx** ubicado en la carpeta **platform-tools** de Android.

Este comando tiene la siguiente estructura **dx [options][sourcefiles]** donde los parámetros más usados como opciones son: **-dex** permite convertir las clases a un solo archivo .dex, **-verbose** muestra todos los pasos de la creación del archivo y **-output** indica la ruta donde se quiere el archivo DEX; este archivo debe nombrarse **classes.dex** para que el sistema Android pueda localizarlo y ejecutarlo correctamente. Si se desea más información sobre este comando, se puede utilizar el parámetro **-help**.

```
dx -dex -verbose -output="c:/MisProyectos/MiJuegoEnAndroid/bin/classes.dex"
"c:/MisProyectos/MiJuegoEnAndroid/obj" "c:/MisProyectos/MiJuegoEnAndroid/libs"
```

Código 7: Creación de archivos DEX utilizando el comando DX

En este ejemplo se muestra el uso del comando **DX**, después del parámetro **output** se indican las rutas donde se encuentran los archivos .class y las librerías extras que se vayan a agregar al sistema.

3.4.4 Firmando la aplicación

La firma de la aplicación permite que se genere un archivo con extensión APK. Para poder firmar una aplicación es necesario generar una llave válida de Java que contenga la información del creador del sistema. Para este caso, se utilizará una llave que contiene los datos del sistema y que permitirá a los usuarios poder publicar sus juegos en el Android Market.

La firma del archivo conlleva tres pasos:

1. Generación del paquete APK sin firmar: Este archivo consiste en empacar todos los archivos en un ejecutable Android. Para realizar esta tarea, se utilizará el comando *aapt* con los siguiente parámetros: *-v* que indica que se quiere se muestren todos los pasos del proceso, *-f* con el que se indica que se quiere se sobrescriban los archivos ya existentes, *-M* donde se indica la ruta completa del *AndroidManifest* *-S* que es la ruta donde se encontrarán todos los recursos, *-A* indicando la ruta de los archivos utilizados como recursos secundarios, *-I* indicando el paquete principal del sistema y *-F* para indicar la ruta donde se quiere que se guarde el archivo generado.

```
aapt package -v -f -M "c:/MisProyectos/MiJuegoEnAndroid/AndroidManifest.xml" -S
"c:/MisProyectos/MiJuegoEnAndroid/res" -A "c:/MisProyectos/MiJuegoEnAndroid/assets" -I
"c:/Program Files/Android/Android-sdk/platforms/android-7/android.jar" -F
"c:/MisProyectos/MiJuegoEnAndroid/bin/MiJuegoEnAndroid.unsigned.apk"
"c:/MisProyectos/MiJuegoEnAndroid/bin"
```

Código 8: Generando el archivo APK sin firmar

2. Generación del paquete APK firmado: Al firmar la aplicación, se integran al archivo el directorio META-INF y dentro de ella la firma con extensión SF y el estándar de criptografía para las llaves públicas (RSA). Para llevar a cabo la firma se utiliza el comando *jarsigner* [opciones] *archivoSinFirmar* [usuarioFirma]. Que pide los siguientes parámetros en las opciones: *-verbose* para mostrar todo el proceso de firma, *-keystore* para indicar la ruta donde se tiene la llave, *-storepass* para indicar el password de la llave generada, *-keypass* que indica el password del usuario y *-signedjar* que indica la ruta donde se guardará el archivo.

```
Jarsigner -verbose -keystore "c:/MisProyectos/llaveAndroid.keystore" -storepass cubegame
-keypass cubegame -signedjar c:/MisProyectos/MiJuegoEnAndroid/bin/MiJuegoEnAndroid.signed.apk
"c:/MisProyectos/MiJuegoEnAndroid/bin/MiJuegoEnAndroid.unsigned.apk" CubecraftGame
```

Código 9: Firmando aplicaciones en Android

3. Alineación del archivo APK para dispositivos Android: Esta función como su nombre lo indica, permite alinear los archivos Android para optimizar sus datos y permitir que la descompresión durante la instalación sea satisfactoria. Esto ayuda a reducir la cantidad de RAM utilizada en el dispositivo. La estructura del comando es *zipalign [-f] [-v] [alineación] archivoEntrada.apk archivoSalida.apk*. donde *-f* permite que se sobrescriba el archivo y *-*

v indica si se quiere mostrar todo el proceso de alineación. La alineación que se está utilizando para este proyecto es de 4 bytes, lo cual permite que los archivos no sean pesados.

```
Zipalign -v -f 4 "c:/MisProyectos/MiJuegoEnAndroid/bin/MiJuegoEnAndroid.signed.apk"  
"c:/MisProyectos/MiJuegoEnAndroid/bin/MiJuegoEnAndroid.apk"
```

Código 10: Optimización de los proyectos APK

3.4.5 Iniciando un emulador

Los emuladores se encuentran definidos dentro de la herramienta AVD de Android, y su función es mostrar la funcionalidad que tiene una aplicación desarrollada. Android permite utilizar sus emuladores una vez que se tienen descargadas las imágenes de los sistemas operativos.

Para este trabajo, se generó un emulador llamado telefono2. Este dispositivo tiene la configuración de un Android 2.1 Eclair con una pantalla de un dispositivo Nexus S, que contiene un tamaño de pantalla estándar (4 pulgadas con resolución de 480 por 800 pixeles).

Para iniciar un emulador desde la consola, se utiliza el comando *emulator*. Este comando tiene la siguiente estructura: ***emulator -avd <nombre_dispositivo> [-option [valor]]***. Los valores que puede tener como opciones son: *-sdcard* con el que se le indica el archivo que contiene una imagen de una memoria SD, *-shell* para indicar si se quiere utilizar algún comando en la línea de comandos del núcleo, *-verbose* para mostrar todos los procesos que está realizando, etc.

Emulator -avd telefono2

Código 11: Mostrando un Emulador Android en pantalla

3.4.6 Instalando la aplicación

ADB (Android Debug Bridge) es un comando versátil que permite al desarrollador comunicarse con un emulador o con cualquier dispositivo Android activado en modo debug. Este servicio funciona como un servidor que se mantiene ejecutándose de forma oculta y está en continuo contacto con los dispositivos.

Cuando el servidor está funcionando, todo comando que se envíe a los dispositivos tendrán un tiempo de espera de un minuto. Si en el tiempo de espera no se puede enviar la información, entonces se detendrá la ejecución y se regresa a la línea de comandos.

Para poder llevar a cabo la instalación de la aplicación, es necesario asegurarse que no exista instalada una instancia del juego, por lo que se debe utilizar primero el comando *uninstall* seguido del nombre del paquete que se va a eliminar del dispositivo.

```
adb uninstall com.upaep.thesis
```

Código 12: Desinstalando aplicación Android

Ya desinstalado, se debe instalar la nueva versión de la aplicación, para ello se utiliza el comando *Install* seguido de la ruta del archivo APK a instalar. Este archivo ya debe encontrarse firmado y alineado, en caso contrario la línea de comandos marcará un error y no permitirá su instalación en el dispositivo.

```
adb install "c:/MisProyectos/MIJuegoEnAndroid/bin/MIJuegoEnAndroid.apk"
```

Código 13: Instalando aplicación Android

Cuando la instalación es satisfactoria y completa, se debe iniciar la aplicación de forma automática. Para llevar a cabo esta acción debemos utilizar la línea de comandos del nucleo de Android, el comando de arranque tiene la siguiente estructura ***adb Shell am start -a tipoActividad -n paquete/actividad***.

```
adb shell am start -a android.intent.action.MAIN -n com.upaep.thesis/.MIJuegoEnAndroid
```

Código 14: Iniciando aplicaciones en Android

Para realizar las pruebas de éste Sprint se ejecutaron los archivos Batch desde línea de comandos con diferentes valores y caracteres para conocer el comportamiento de Android al utilizar comandos válidos y no válidos.

Una vez hecha esta prueba, y revisando que se generan todos los juegos, se probó crear un juego desde la aplicación utilizando los archivos generados. Se descubrieron algunos errores de ruta los cuales fueron corregidos.



Figura 14: Tiempo Estimado vs. Tiempo Real del Tercer Sprint

Cuando el sistema pasó las pruebas, se tuvo un tiempo de 3760 minutos, que corresponden a ocho días hombre de trabajo. Con respecto a lo planeado, se tuvo una desviación de 175 minutos, que corresponden a una desviación del 4.88%

3.5 Pruebas

A pesar de que la metodología Ágil Scrum no maneja pruebas una vez que se terminó el desarrollo, se hizo una adaptación al modelo para realizarlas, con la finalidad de verificar la calidad del sistema y la cantidad de recursos que utiliza el sistema. Las pruebas se realizaron utilizando la herramienta de medición de rendimiento EQUATEC, herramienta de uso libre que mide los tiempos que tardan cada proceso del sistema.

Equatec es una empresa que ofrece diferentes sistemas para el análisis de rendimiento y velocidad de los sistemas desarrollados con tecnologías .NET. La herramienta que se utilizó para medir la velocidad del sistema es EQUATEC Profiler.

Esta herramienta sirve para calcular el tiempo que tarda cada una de las acciones de la aplicación, sin tomar en cuenta la cantidad de memoria utilizada. Esto es utilizado en el mercado para hacer que las aplicaciones sean rápidas y fluidas para el usuario.

Para realizar el análisis de los resultados, se tiene una tabla de valores definidos por Jakob Nielsen en el libro "Usability Engineering" donde indica tres límites importantes para el tiempo de respuesta. Estos límites estarán condicionados por la velocidad de procesamiento y la cantidad de memoria RAM que tiene cada equipo en el que se trabaje. (Tzancogg, Rodríguez, & Valeria)

Tiempo (Segundos)	Características
0.1	Es el límite en el cual el usuario siente que está "manipulando" los objetos desde la interfaz del usuario
1	Es el límite en el cual el usuario siente que está navegando libremente sin esperar demasiado una respuesta del sistema.
10	Es el límite en el cual se pierde la atención del usuario, si la respuesta tarda más de 10 segundos se deberá indicar algún mecanismo por el cual el usuario pueda interrumpir la operación

Las funcionalidades de la herramienta a analizar son:

- Inicio del Sistema: El tiempo que tarda el inicio del sistema es de 2 segundos y 651 milisegundos, lo cual basados en la tabla anterior, es un buen tiempo de espera para el usuario, lo cual muestra la velocidad con la que se realiza la revisión de los archivos de configuración.

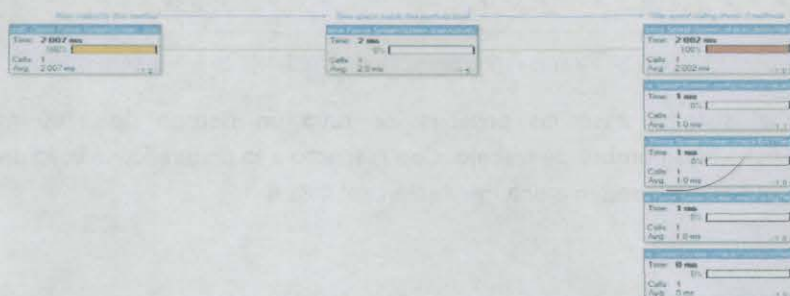


Figura 15: Tiempo de respuesta del inicio del sistema

- Creación de Proyectos: El tiempo aproximado que tarda en realizar todas las tareas el sistema cuando se quiere generar un proyecto es de 9 segundos y 517 milésimas. Este tiempo sigue estando debajo de los estándares, lo que demuestra una buena optimización de procesos durante el desarrollo.

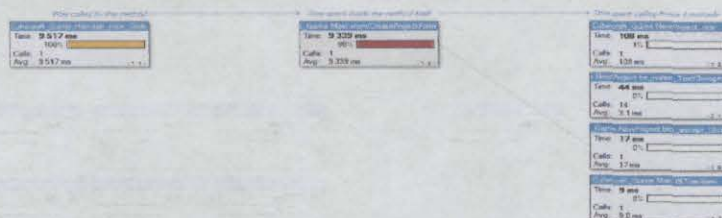


Figura 16: Tiempo de respuesta al crear un proyecto

- Inserción de Cubos: El tiempo de la generación del cubo es menor a un segundo, mostrando que la velocidad de creación de información y el uso de imágenes predefinidas es el correcto.

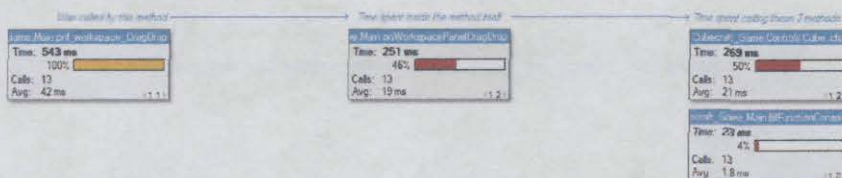


Figura 17: Rendimiento al insertar un cubo

- Configuración de cubos: Esta tarea tiene un tiempo de respuesta de 111 milésimas de segundo, este tiempo es necesario para cargar los componentes de usuario que se generaron y mostrárselos al usuario de una forma rápida. Basados en las definiciones de la tabla, el tiempo de espera es ideal y permite que los usuarios puedan utilizar esta funcionalidad de una forma fluida.

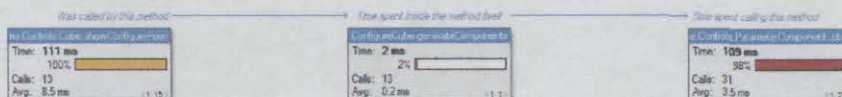


Figura 18: Tiempo de respuesta a la configuración de cubos

- **Compilación del juego:** La compilación del juego, es la tarea que más tiempo se lleva. Esto es debido a la cantidad de información que debe procesar y procesos en archivos batch que no pueden ser optimizados, ya que al utilizar métodos de terceros, se compromete el sistema al tiempo de respuesta preparado por sus desarrolladores. En este caso, como el tiempo sobrepasa la tabla propuesta teniendo 1 minuto 17 segundos y 433 milésimas. Tomando en cuenta las posibles acciones propuestas por la tabla, se permite que el usuario pueda interrumpir el proceso que se estaba llevando a cabo sin comprometer los demás procesos.

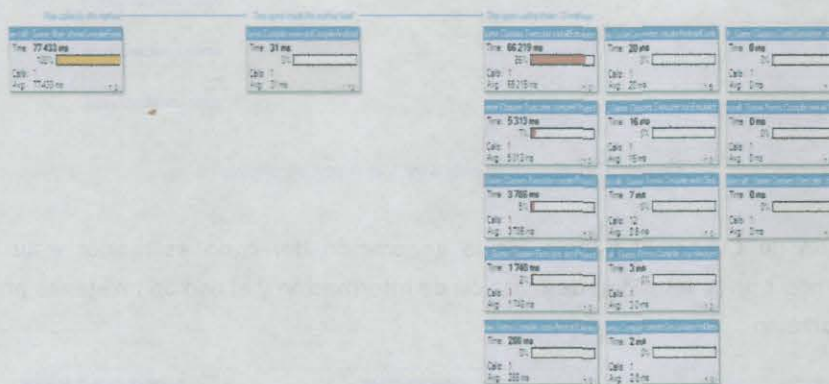


Figura 19: Tiempo de respuesta para la compilación del juego

Algo muy importante para el principiante es comprender
Que si los resultados van a ser tratados estadísticamente,
Es necesario tener en cuenta la estadística al planear el experimento
-- William Ian Beardmore Beveridge

Capítulo 4. Aplicación del Experimento

4.1 Objetivo

El objetivo principal del experimento es confirmar el cumplimiento de los atributos de calidad del framework, mediante la experimentación en dos grupos con diferentes características, con la finalidad de analizar si el framework puede ser empleada por personas de diferentes edades para el desarrollo de videojuegos y la aceptación que puede llegar a tener el sistema entre los usuarios a quienes va dirigido.

4.2 Grupos de Enfoque

En el primer grupo fue conformado por estudiantes de nivel primaria, que se encontraban en 5° o 6° de primaria y cuyas edades eran de 10 a 12 años. Para este grupo, se requirió tuvieran conocimientos básicos de computación para que pudieran utilizar el sistema sin ningún tipo de complicación.

En el segundo grupo de enfoque, se tuvieron estudiantes de preparatoria que cursan sus últimos dos semestres, con conocimientos básicos de computación y de lógica matemática. Los requisitos mínimos que se establecieron son debidos a la complejidad que pueden llegar a tener sus videojuegos durante su desarrollo.

4.3 Hipótesis

H1. El framework propuesto permite a los niños en edades de 10 a 12 años y adolescentes de 15 a 17 años a entender y desarrollar videojuegos para dispositivos móviles de una manera sencilla, sin importar que reciban o no capacitación sobre su uso.

4.4 Preguntas de Investigación

1. PI1: ¿La herramienta es intuitiva para el usuario?
2. PI2: ¿Cuál es la aceptación que tiene el sistema entre los usuarios objetivo?
3. PI3: ¿Qué flexibilidad puede llegar a otorgar el sistema para el desarrollo de videojuegos?

4.5 Experimento

El experimento se empleó de forma diferente para cada uno de los grupos de enfoque del framework. Para el grupo de usuarios de nivel primaria se preparó un experimento dividido en 3 fases. Cada uno de las fases tuvo un área de análisis en específico.

Fase 1: El objetivo de esta fase fue conocer si la interfaz propuesta es intuitiva para el grupo de personas a la que está dirigido. Para ello se pidió a los grupos que construyan un videojuego sin recibir una explicación previa del framework.

1. Se dividió al grupo en cuatro equipos, ubicándolos en lugares donde no puedan observar el trabajo de los demás equipos.
2. El responsable del experimento introdujo la historia del juego tomados del apartado 4.9 y la funcionalidad que debe tener para que los estudiantes puedan desarrollarlo.
3. Los alumnos armaron la dinámica mostrada en el sistema y la hicieron funcionar en el simulador.
4. El responsable de la actividad revisó la forma en la que armaron la dinámica y que toda la funcionalidad se cumpliera.

Fase 2: Durante la ejecución de ésta fase, se explicó el objetivo del framework y la forma de utilizar correctamente la interfaz del framework, y se pidió a los grupos que desarrollan una nueva dinámica de juego.

1. El responsable de aplicar el experimento explicó el sistema.
2. El responsable del experimento introdujo la historia del juego tomados del apartado 4.9 y la funcionalidad que debe tener para que los estudiantes pudieran desarrollarlo.
3. Los alumnos tuvieron que armar la dinámica mostrada en el sistema y hacerla funcionar en el simulador.
4. El responsable de la actividad revisó la forma en la que armaron la dinámica y que toda la funcionalidad se cumpliera.

Fase 3: En ésta fase se recabaron comentarios de todos los grupos de trabajo divididos en dos etapas, en la primera etapa se pidieron opiniones orales sobre el sistema, y en la segunda etapa se les otorgaron encuestas anónimas.

1. Aún por equipos, se pidió a los integrantes que dieran sus opiniones sobre el sistema, principalmente sobre los siguientes puntos:
 - a. Facilidad de Uso
 - b. Dinámica
 - c. Funcionalidad
2. Se desintegraron los equipos y se aplicó la encuesta especificada en el apartado 4.6, se le pidió a los estudiantes la contestasen con la mayor sinceridad posible, ya que los resultados obtenidos de las encuestas eran confidenciales.

Para el grupo de estudiantes de preparatoria, se aplicó un experimento de dos fases donde se esperaba que basado en sus conocimientos desarrollasen juegos simples que ayuden a entender algún tema académico o problema social.

Fase 1: En esta fase los usuarios debían realizar un videojuego sobre un tema en particular sin tener conocimiento del uso del sistema.

1. Los usuarios tuvieron que armar 4 equipos, y discutir probables temas académicos o sociales que pudieran ayudar a entender por medio de un videojuego.
2. Cada equipo debió elegir y desarrollar utilizando el framework una forma de juego de las que el responsable mostró y que mejor se adaptara para resolver el problema.
3. El responsable de la actividad debía revisar la forma en la que armaron la dinámica y revisar que cumpla con todas las características especificadas.

Fase 2: Durante esta fase se esperaba que los estudiantes conocieran los alcances y limitaciones que tiene el sistema, desarrollaran otro juego y al finalizar contestaran una encuesta de forma individual para conocer su satisfacción y el agrado que tuvieron hacia el sistema.

1. El responsable de la actividad explicó qué tipo de videojuegos se pueden desarrollar y hasta qué punto el sistema tiene flexibilidad para el desarrollo.
2. Se pidió a los usuarios que escucharan una historia del apartado 4.9 y eligieran una forma de juego para la historia.
3. Cada equipo tuvo que programar el juego utilizando las herramientas del sistema.
4. El responsable de la actividad revisó la forma en la que armaron la dinámica.
5. Una vez que se terminó la actividad se pidió a los usuarios contestaran con la mayor sinceridad posible una encuesta.

4.6 Encuestas

Para el grupo de primaria se empleó la siguiente encuesta:

Edad: _____ Grado: _____

Marca la cara que corresponda a las siguientes preguntas según tu experiencia con el programa



Me gustó mucho

Me gustó

Ni me gustó ni
me disgustó

No me gustó

Me disgustó

Característica

Calificación

1. ¿Qué tan fácil se te hizo utilizar el programa?



2. ¿Te gustaron los colores de los cubos?



3. ¿Tenías todas las funciones que querías para tu juego?



4. ¿Te gustó hacer juegos con este programa?



5. ¿Se te hizo sencillo poner los valores a los cubos?



Para el grupo de preparatoria se utilizó la siguiente encuesta:

Encuesta para determinar el rendimiento y usabilidad del framework Cubecraft Game: Entorno para el desarrollo de videojuegos.

Edad: _____ Semestre: _____

Califica las siguientes características de la actividad según el siguiente criterio:

E: Excelente M: Me Gusta D: Ni me gusta ni me disgusta N: No me gusta

a) Facilidad de uso	
b) Distribución de los elementos en pantalla	
c) Forma de la figura	
d) Facilidad para desarrollar los juegos	
e) Funcionalidad disponible en los cubos	
f) Atracción por el área de Sistemas Computacionales	

Contesta las siguientes preguntas eligiendo solo una de sus opciones.

1. ¿Te llamó la atención la dinámica de la actividad?

☐ Sí ☐ No ¿Por qué? _____

2. ¿Volverías a realizar la actividad?

☐ Sí ☐ No ¿Por qué? _____

3. ¿Qué te parecieron los colores empleados?

☐ Bien ☐ Regular ☐ Mal ¿Por qué? _____

4. ¿Cómo distribuirías los colores?

☐ Un color por cada tipo de acción

☐ Distribuir los colores como un semáforo para saber cuándo se equivoquen.

☐ Las dos anteriores. La intensidad del color indicaría si está correcto o no la opción.

5. Si pudieras elegir los colores, ¿Qué colores asignarías a cada una de las acciones?

6. Las acciones que necesitabas estaban completas

☐ Sí ☐ Casi Todas ☐ No ¿Por qué? _____

7. ¿Las acciones para cada uno de los juegos fueron fáciles de encontrar y configurar?

☐ Sí ☐ No ¿Por qué? _____

8. Si pudieras cambiar algo, ¿Qué cambiarías?

4.7 Método de Análisis

El método estadístico a utilizar será la Distribución Gaussiana, la cual fue elegida por las siguientes características: (Pertegas & Pita, 2001)

1. Tiene una única moda, que coincide con su media y su mediana
2. La curva normal es asintótica al eje de las abscisas. Por ello, cualquier valor entre $-\infty$ e ∞ es teóricamente posible. El área tota bajo la curva es, por tanto, igual a 1.
3. Es simétrica con respecto a su media. Según esto, para este tipo de variables existe una probabilidad de un 50% de observar un dato mayor que la media y un 50% de observar un dato menor.
4. La distancia entre la línea trazada en la media y el punto de inflexión de la curva es igual a una desviación típica. Cuanto mayor sea ésta desviación, mas aplanada será la curva de la densidad.
5. El área bajo la curva comprendida entre los valores situados aproximadamente a dos desviaciones estándar de la media es igual a 0.95. En concreto, existe un 95% de posibilidades de observar un valor comprendido en el intervalo $(\mu - 1.96\sigma; \mu + 1.96\sigma)$, donde μ es la media y σ es la desviación típica.
6. La forma de la campana de Gauss depende de los parámetros de la media y la desviación típica. La media indica la posición de la campana, de modo que para diferentes valores de μ la grafica es desplazada a lo largo del eje horizontal. Por otra parte, la desviación estándar determina el grado de apuntamiento de la curva. Cuanto mayor sea el valor de σ , más se dispersarán los datos en torno a la media y la curva será más plana. Un valor pequeño de este parámetro indica, por tanto, una gran probabilidad de obtener datos cercanos al valor medio de la distribución.

4.8 Tratamiento del Análisis

Las respuestas obtenidas del experimento una vez analizadas con el método de distribución gaussiana, ayudarán a responder las preguntas de investigación con relación a las preguntas:

- | | |
|---|---|
| 1. ¿La herramienta es intuitiva para el usuario? | Respuestas a las preguntas 1 y 5 de la encuesta para niños de primaria y las preguntas A,B y 1 |
| 2. ¿Cuál es la aceptación que tiene el sistema entre los usuarios objetivo? | Respuestas a las preguntas 2 y 4 de la encuesta para niños de primaria y las preguntas C,D,3 |
| 3. ¿Qué flexibilidad puede llegar a otorgar el sistema para el desarrollo de videojuegos? | Respuesta de la pregunta 3 en la encuesta para niños y de las preguntas E, 6 y 8 para adolescentes. |

Las respuestas de las preguntas 4 y 5 de la encuesta para adolescentes servirán para realizar la mejora continua del sistema y plantear nuevas funcionalidades como un trabajo a futuro.

Para aceptar o desechar la hipótesis se utilizará una distribución porcentual, donde el punto de partida es el 50% de la suma de los grupos de enfoque. Debido a que la hipótesis está dividida en 3 preguntas, ésta se validará cuando al menos dos de ellas tengan una distribución arriba del punto de partida.

Debido a que se trata de un trabajo de investigación, se requiere se tenga una confianza en el estudio del 95% por lo cual, para conocer el número de encuestas que se deben realizar se ocupa la siguiente formula:

$$n = \frac{N \sigma^2 z^2}{(N - 1)e^2 + \sigma^2 z^2}$$

Donde n es el número de encuestas a realizar para tener el grado de confianza deseado, N es el tamaño poblacional que se tiene, σ es la desviación estándar que para este experimento se utilizará .5 (50%), Z que es el nivel de confianza requerido para el experimento y e que es el grado de error que se espera.

Debido a que todavía no se conoce el tamaño de la población que se tiene, el número de encuestas será directamente proporcional al valor de N. Mientras que el nivel de confianza y el grado de error que se ocuparán para el proyecto serán de 95% (1.96) y 0.01% respectivamente, por lo cual el número de encuestas (n) queda de la siguiente manera:

$$n = \frac{N \sigma^2 z^2}{(N - 1)e^2 + \sigma^2 z^2}$$
$$n = \frac{N (0.5)^2 (1.96)^2}{(N - 1)(0.01) + (0.5)^2 (1.96)^2}$$

$$n = \frac{0.9604 N}{0.01N + 0.9504}$$

4.9 Historias para la creación de videojuegos

En ésta sección se especifican las historias con las que se pueden llevar a cabo el experimento, se especifica el inicio de la historia y los personajes a utilizar.

➤ Historia 1: Al rescate de la princesa

En una tierra lejana, existían dos pequeños reinos, la tierra de fuego y el reino de los bosques, los cuales nunca tuvieron armonía. Cierta día, uno de los guardias de la tierra de fuego, fue enviado por el rey a secuestrar a la princesa y así obligar al monarca del reino de los bosques a entregar todas sus tierras. En un acto desesperado, el monarca del reino de los bosques envía a su más poderoso general a rescatarla.

Personajes: Princesa, General, Guardia.

➤ Historia 2: Titín, La Avispa (Sacristán, 2010)

Titín volvía otra vez a casa sin comida. Como casi siempre, una de las avispas mayores se la había quitado, amenazándole con pegarle un coco. De camino, Titín paró en el parque y se sentó en un banco tratando de controlar su enfado y su rabia. Como era un chico sensible e inteligente, al poco rato lo había olvidado y estaba disfrutando de las plantas y las flores. Entonces, revoloteando por los rosales, vio una avispa y se asustó.

Personajes: Titín, Avispa.

➤ Historia 3: El laberinto de Kratón (Sacristán, 2010)

En lo más profundo de la gran montaña, en un gran laberinto de túneles creado por los grandes magos, vivía encerrada la peor de las fieras, una bestia horrible a quien todos conocían por el nombre de Kratón. Había sido encerrado allí tras aterrorizar a todos los pueblos, en un laberinto mágico con una única entrada y salida que cambiaba de lugar cada día.

Pero ocurrió que llegó a aquellas tierras un hombre de corazón malvado, perverso hasta el extremo, cuyo único deseo era someter a todos los hombres del reino. Se llamaba Jafa, y tanta prisa sentía por cumplir sus deseos, que al enterarse de la existencia de Kratón, pensó en liberarlo para que le ayudara a completar sus planes, por muy peligroso que fuera el monstruo.

Personajes: Kratón, Jafa.

➤ Historia 4: Juanija Lagartija (Sacristán, 2010)

Juanija Lagartija vivía entre unas piedras en el campo. Como a todas las lagartijas, le encantaba tomar tranquilamente el sol sobre una gran roca plana. Allí se quedaba tan a gusto, que más de una vez había llegado a dormirse, y eso fue lo que pasó el día que perdió su rabito: unos niños la atraparon, y Juanija solo pudo soltarse perdiendo su rabo y corriendo a esconderse.

Asustada oyó como aquellos niños reían al ver como seguía moviéndose el rabito sin la lagartija, y terminaban tirándolo al campo después de un ratito. La lagartija comenzó entonces a buscarlo por toda la zona, dispuesta a recuperarla como fuera para volver a colocarla en su sitio. Pero aquel campo era muy grande, y por mucho que buscaba, no encontraba ni rastro de su rabito.

Personajes: Juanija, Rabo de Juanija

Todo lo que somos es el resultado de lo que hemos pensado;
Esta fundado en nuestros pensamientos y está hecho
De nuestros pensamientos
-- Buda

Capítulo 5. Resultados

En este capítulo tomamos los resultados de la aplicación del experimento. Para llevarlo a cabo, contactamos un total de 23 preparatorias y primarias, de las cuales se nos autorizó realizar la investigación en institutos de educación primaria y de educación preparatoria ubicadas en el centro y poniente de la ciudad de Puebla.

Muchas de las escuelas visitadas no dieron su autorización debido a diferentes motivos, entre los que destacan el uso de videojuegos como herramientas de ocio. Algunos de los comentarios recolectados de los directores indican que: *“Actualmente en nuestro plantel estamos eliminando el uso de los videojuegos”, “Los niños actualmente juegan mucho y leen poco, permitir que desarrollen videojuegos va en contra el ideal del instituto”*.

Estas ideas expuestas por los directores lo tratamos de contrarrestar indicándoles que el uso de nuestra aplicación no era para ocio, sino que el objetivo principal era que los estudiantes fomentaran su creatividad e imaginación, y que lo expresaran por medio de desarrollos tecnológicos.

Las pruebas las realizamos el 18 y 19 de Febrero en la preparatoria y los días 5 y 11 de Marzo en la primaria. Con respecto a los grupos de preparatoria, pedimos que los grupos con los que trabajáramos no estuvieran en áreas de tecnologías, es por ello que trabajamos con los grupos de Artes y Medicina.

En la preparatoria, los equipos que utilizamos son clientes delgados de una máquina virtual que utiliza Linux, y que dirige el trabajo hacia un equipo físico con Sistema Operativo Windows 7, que distribuye las tareas por sesiones de usuario. Al utilizar este tipo de equipos, se complican las pruebas debido a que los procesos son conjuntos y no permiten que se ejecuten emuladores independientes, lo que resultó en algunas ocasiones en que la instalación de los juegos fallara.

En la primaria, se utilizaron equipos con procesador Pentium 4 y 512 Mb de RAM. Estos equipos no cumplen con los requerimientos mínimos necesarios para poder hacer funcionar Android correctamente, ya que el emulador utiliza procesos con alto consumo. Para realizar la prueba hicimos un bloqueo de la emulación del juego, permitiendo que el sistema generara el ejecutable y después probarlo en otra computadora de mayor capacidad.

5.1 Recolección de Información

El experimento lo aplicamos en la preparatoria con el grupo de Artes que cuenta con un total de 25 estudiantes. Este grupo no recibió ningún tipo de capacitación sobre el sistema, solo les explicamos cuál era el objetivo y el juego que debían desarrollar.

En el segundo día con preparatoria llevamos a cabo el experimento con el grupo de Medicina que cuenta con 37 estudiantes. A este grupo se les dio una plática con una duración aproximada de 20 minutos, donde les explicamos el objetivo del sistema y la forma correcta de utilizar el framework, haciendo un ejemplo práctico al final de la plática.

Con los grupos de primaria trabajamos con los grupos "A" y "B" de quinto y sexto grado. En el primer día, trabajamos con los grupos quinto, donde tuvimos un total de 16 estudiantes. Con este grupo, efectuamos una capacitación parcial, donde esperábamos que los estudiantes pudieran interactuar libremente con el sistema.

En el segundo día de primaria, trabajamos con los grupos de sexto. En estos grupos se tuvo un total de 49 estudiantes, a los que se les dimos una plática sobre el sistema y posteriormente los estudiantes desarrollaron un juego de forma conjunta.

Muchos de los resultados obtenidos de las encuestas se encuentran influenciados por varios factores externos, entre los que destacan: el gusto por los videojuegos, la atención prestada a la explicación previa y la motivación de los estudiantes para el aprendizaje de nuevas tecnologías, sin embargo, estas variables no fueron consideradas en el análisis de la información, pero, sin duda, será necesario considerarlas en investigaciones futuras.

5.2 Interpretación de Resultados

Para realizar la interpretación de los datos obtenidos, se utilizará una doble validación. La primera validación será utilizando el método estadístico descrito en el capítulo 4, y la segunda validación será por la interpretación de gráficas con los resultados.

Para el método estadístico, se toman como casos exitosos todos aquellos donde las personas hayan respondido que les gustó mucho o respondiendo afirmativamente a las preguntas de las encuestas. Para validar cada una de las preguntas de investigación, es necesario que la mayoría de las conclusiones de resultados cumplan con lo marcado en la hipótesis propuesta.

Para conocer los detalles de las fórmulas empleadas en el análisis estadístico y los criterios de interpretación de resultados, se puede revisar el apartado 4.8 Tratamiento del Análisis.

5.2.1 P11. ¿La herramienta es intuitiva para el usuario?

En esta pregunta se desea conocer qué tan fácil fue para los usuarios el desarrollo de los videojuegos, y si es necesario para todos los usuarios a los que va dirigido el sistema tener una capacitación previa de su uso.

5.2.1.1 Primaria

Para resolver la pregunta P1 en estudiantes de primaria, se utiliza los resultados de las preguntas uno y cinco de las encuestas aplicadas (Capítulo 4 apartado 4.9), de las cuales se obtienen las siguientes gráficas:



Gráfica 1: Facilidad de uso en Primaria

En la gráfica 1 se muestra la opinión de los estudiantes de primaria con respecto a la facilidad para usar el programa. Al tener la capacitación completa, los resultados de las encuestas son positivos en la mayoría de las encuestas aplicadas, mientras que con una capacitación parcial, las opiniones se encuentran divididas.

Este resultado desde el punto de vista estadístico, genera los siguientes resultados:

$$p'_1 = \frac{8}{16} \quad p'_2 = \frac{23}{49}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.5)(0.5)}{16} + \frac{(0.469)(0.5306)}{49}} \quad \sigma_{(p'_1 - p'_2)} = 0.143902$$

$$E = (1.96)(0.143902) \quad E = 0.2820$$

$$(0.5 - 0.46) - 0.2820 < P_1 - P_2 < (0.5 - 0.46) + 0.2820$$

$$-25.1\% < P_1 - P_2 < 31.3$$

Análisis Estadístico 1: Facilidad para Usar el Programa en Primaria

Basados en las posible interpretaciones marcadas por el método estadístico, el intervalo de confianza contiene entre sus posibles valores el 0, indicando que no hay diferencia entre los resultados de las dos poblaciones. Esto nos indica que para los estudiantes de primaria no es necesario que reciban una capacitación previa sobre el uso del framework.



Gráfica 2: Sencillez del Sistema en Primaria

En la gráfica 2 se muestra el número de personas que han dado casos de éxito a la sencillez del sistema. Al igual que en la gráfica anterior, a los estudiantes de primaria les gusta la sencillez con la que pueden desarrollar, haciendo que la validación por gráfica sea satisfactoria.

Realizando la validación por el método estadístico se obtienen los siguientes resultados:

$$p'_1 = \frac{8}{16} \quad p'_2 = \frac{25}{49}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.5)(0.5)}{16} + \frac{(0.5102)(0.4897)}{49}} \quad \sigma_{(p'_1 - p'_2)} = 0.143961$$

$$E = (1.96)(0.143961) \quad E = 0.2821$$

$$(0.5 - 0.5102) - 0.2821 < P_1 - P_2 < (0.5 - 0.5102) + 0.2821$$

$$-29.2\% < P_1 - P_2 < 27.2\%$$

Análisis Estadístico 2: Sencillez del Sistema en Primaria

En el intervalo que se obtuvo del análisis estadístico 2 sobre la sencillez del sistema podemos concluir que no importa si los alumnos recibieron una capacitación total o parcial, ya que el sistema es lo suficientemente intuitivo para ellos y se les facilita su uso.

Los resultados obtenidos en ambas encuestas cumplen con lo planteado en la hipótesis. Estos resultados nos indican que en niños de primaria, la herramienta si es intuitiva hacia los usuarios por lo que pueden o no recibir capacitación previa de su uso y aun así generar juegos.

5.2.1.2 Preparatoria

Para resolver la pregunta P1 de investigación con los estudiantes de preparatoria, se utilizaron las preguntas A y B de la primera sección y la pregunta 1 de la segunda sección de las encuestas aplicadas (Capítulo 4 apartado 4.9), donde se encontraron los siguientes resultados:



Gráfica 3: Facilidad de Uso en Preparatoria

En la gráfica 3 se muestra como las personas que recibieron una capacitación previa se les hizo fácil utilizar el programa, mientras que al no tener capacitación los estudiantes no sienten mucho agrado hacia el sistema. Lo que se pudo observar en ambos grupos, es la necesidad de los estudiantes de preparatoria de recibir una explicación de las herramientas que utilizarán, reduciendo su capacidad de exploración.

Visto desde el punto de vista estadístico, esta pregunta tiene los siguientes datos:

$$p'_1 = \frac{0}{25} \quad p'_2 = \frac{13}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0)(1)}{25} + \frac{(0.3513)(0.6486)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.0784$$

$$E = (1.96)(0.0784) \quad E = 0.1538$$

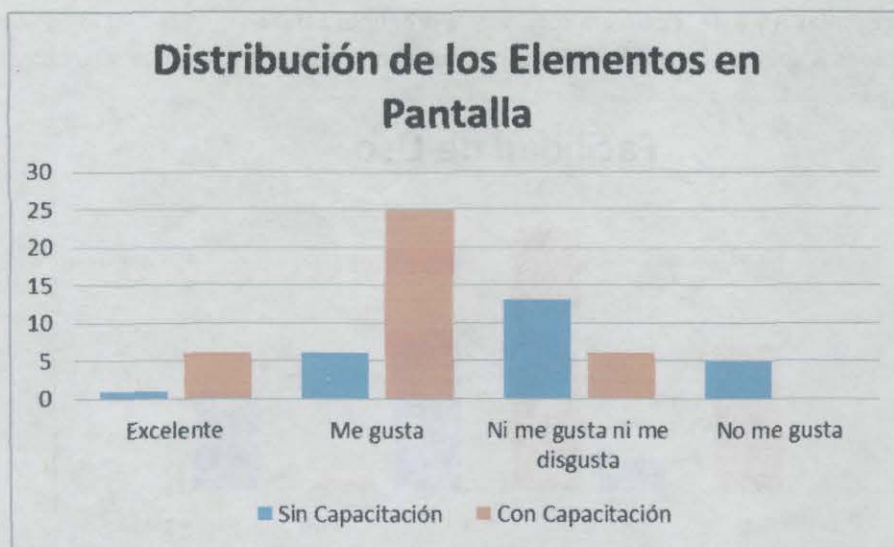
$$(0 - 0.3513) - 0.1538 < P_1 - P_2 < (0 - 0.3513) + 0.1538$$

$$-50.5\% < P_1 - P_2 < -19.75\%$$

Análisis Estadístico 3: Facilidad de Uso en Preparatoria

Estos resultados comprueban los obtenidos por medio de la interpretación de la gráfica. Los estudiantes de preparatoria prefieren tener una capacitación previa sobre el uso del sistema antes de desarrollar algún videojuego.

Sobre el tema de la distribución de la pantalla, los resultados se encuentran divididos entre todas las opiniones. Esto nos indica que se le debe agregar algo a la interfaz que permita al usuario realizar las acciones de forma fluida.



Gráfica 4: Distribución de la pantalla para estudiantes de Preparatoria

Analizando los resultados de esta parte de la encuesta utilizando el método estadístico se obtiene lo siguiente:

$$p'_1 = \frac{1}{25} \quad p'_2 = \frac{6}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.04)(0.96)}{25} + \frac{(0.1621)(0.8378)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.0721$$

$$E = (1.96)(0.0721)$$

$$E = 0.1414$$

$$(0.04 - 0.1621) - 0.1414 < P_1 - P_2 < (0.04 - 0.1621) + 0.1414$$

$$-26.4\% < P_1 - P_2 < 1.93\%$$

Análisis Estadístico 4: Distribución de los Elementos en Pantalla en Preparatoria

El intervalo de confianza de esta pregunta no tiene una variación que pueda considerarse significativa, lo cual que demuestra que a las personas con y sin capacitación creen que la interfaz está bien. Aunque tomando en cuenta lo visto en la gráfica, se deben realizar cambios para aumentar el número de éxitos de la interfaz.



Gráfica 5: Dinámica de la actividad en Preparatoria

Se les preguntó a los estudiantes si les gustó la dinámica de la actividad, con la finalidad de conocer que tanto les había gustado utilizar el sistema. Los resultados mostrados en la gráfica 5 se encuentran niveladas, donde la mayoría de las personas han dicho que volverían a realizar la actividad.

Haciendo el análisis por medio del método estadístico se tiene la siguiente información:

$$p'_1 = \frac{12}{25} \quad p'_2 = \frac{36}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.48)(0.52)}{25} + \frac{(0.9729)(0.0270)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.1034$$

$$E = (1.96)(0.1034)$$

$$E = 0.2026$$

$$(0.48 - 0.9729) - 0.2026 < P_1 - P_2 < (0.48 - 0.9729) + 0.2026$$

$$-69.6\% < P_1 - P_2 < -29.03\%$$

Análisis Estadístico 5: Dinámica de la actividad en Preparatoria

El resultado del método indica que los usuarios con capacitación son a los que más les gustó la actividad. Este resultado está influenciado por un factor importante: El grupo con capacitación pudo terminar su juego y probarlo en los emuladores.

Al analizar los resultados de las tres preguntas de la encuesta, se refleja que el objetivo no se cumple para la preparatoria. Los resultados demuestran que para este grupo de enfoque es necesario darles una capacitación previa, de la cual se debe buscar una estrategia para que sea autodidacta.

Analizando los resultados de ambos grupos de enfoque, se demuestra que la pregunta de investigación se cumple parcialmente. La respuesta que se infiere es: El sistema es fácil de usar

para alumnos de primaria, se les dé o no capacitación, mientras que para los alumnos de preparatoria, ya sea como una herramienta de diversión o de introducción a la carrera, es necesario que reciban una capacitación previa que les permita familiarizarse con los términos y funcionalidades con los que cuenta el sistema.

5.2.2 PI2. ¿Cuál es la aceptación que tiene el sistema entre los usuarios objetivo?

En esta pregunta de investigación, se desea conocer el gusto que tuvieron los estudiantes para generar juegos utilizando el sistema. El objetivo principal es medir el grado de adicción que puede causar, la cantidad de usuarios que puede llegar a tener el sistema y el número de veces que los usuarios utilizarían el sistema.

5.2.2.1 Primaria

A nivel primaria, las preguntas de la encuesta que se utilizan para responder esta pregunta son la 2 y 4 de las encuestas realizadas (Capítulo 4 apartado 4.9), de las cuales se obtuvieron los siguientes datos:



Gráfica 6: Aceptación de los colores de los Cubos en Primaria

Como se puede observar en la gráfica 6, a la mayoría de los usuarios les gustan los colores empleados, demostrando que el uso de colores para demostrar el estado de los cubos es una buena estrategia de retroalimentación. Los resultados del método estadístico son:

$$p'_1 = \frac{11}{16} \quad p'_2 = \frac{19}{49}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.6875)(0.3125)}{16} + \frac{(0.3877)(0.6122)}{49}}$$

$$\sigma_{(p'_1 - p'_2)} = 0.1351$$

$$E = (1.96)(0.1351)$$

$$E = 0.2649$$

$$(0.6875 - 0.3877) - 0.2649 < P_1 - P_2 < (0.6875 - 0.3877) + 0.2649$$

$$3.5\% < P_1 - P_2 < 56.47\%$$

Análisis Estadístico 6: Aceptación de los colores de los cubos en Primaria

Este resultado demuestra que las personas sin capacitación prefieren estos colores para conocer el estado de los cubos, mientras que el segundo grupo, aunque el número de éxitos es alto, existe una diferencia de entre el 3.5% y el 56.47% del gusto por los colores.



Gráfica 7: Estadísticas del gusto por el sistema en primaria

En la gráfica 7 se plantea el gusto por estar desarrollando videojuegos con el framework en ambos grupos de primaria, se tiene un alto número de éxitos, lo cual demuestra que la herramienta tiene una alta aceptación entre los estudiantes de primaria.

$$p'_1 = \frac{12}{16} \quad p'_2 = \frac{31}{49}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.75)(0.25)}{16} + \frac{(0.6326)(0.3673)}{49}} \quad \sigma_{(p'_1 - p'_2)} = 0.1283$$

$$E = (1.96)(0.1283)$$

$$E = 0.2514$$

$$(0.75 - 0.6326) - 0.2514 < P_1 - P_2 < (0.75 - 0.6326) + 0.2514$$

$$-13.4\% < P_1 - P_2 < 36.88\%$$

Análisis Estadístico 7: Gusto por el sistema en primaria

Analizando los datos sobre el gusto por el sistema en estudiantes de nivel primaria por el método estadístico, no existe diferencia entre los dos grupos. Esto nos indica que a nivel primaria, a los usuarios les gusta utilizar el sistema para desarrollar juegos tengan o no una capacitación previa.

Como resultado de la pregunta de investigación planteada, se puede inferir que a nivel primaria el sistema cumple con su objetivo, pero deben hacerse cambios que permitan aumentar el nivel de satisfacción de los usuarios.

5.2.2.2 Preparatoria

A nivel preparatoria, se utilizan las preguntas C y D de la primera sección y la pregunta 2 de la segunda sección (Capítulo 4 apartado 4.9) para responder la pregunta PI2. De las encuestas aplicadas se obtuvieron los siguientes datos:



Gráfica 8: Gusto hacia el aspecto del cubo en preparatoria

Este grupo de enfoque opina que el aspecto que tienen los cubos de funcionalidad no es lo ideal para ellos. El aspecto de los cubos está influenciado por la tarjeta gráfica con la que se cuenta, por lo que algunos de los equipos en donde se realizaron las pruebas se notaban borrosos o no se diferenciaba de los demás elementos. Haciendo el análisis por medio del método estadístico obtenemos:

$$p'_1 = \frac{0}{25} \quad p'_2 = \frac{10}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0)(1)}{25} + \frac{(0.2702)(0.7297)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.0730$$

$$E = (1.96)(0.0730)$$

$$E = 0.1430$$

$$(0 - 0.2702) - 0.1430 < P_1 - P_2 < (0 - 0.2702) + 0.1430$$

$$-41.3\% < P_1 - P_2 < -12.72\%$$

Análisis Estadístico 8: Gusto por el aspecto del cubo en Preparatoria

El intervalo de confianza obtenido comprueba lo inferido en el párrafo anterior. Los usuarios de preparatoria sienten un mayor gusto por la forma de los cubos cuando reciben una capacitación. Esta respuesta tiene varios sustentos, donde destacan: Se puede observar en diferentes equipos la calidad que tiene el cubo y se conoce el motivo de utilizar un cubo en lugar de otra figura geométrica.



Gráfica 9: Estadísticas de la facilidad para desarrollar juegos en preparatoria

Como se puede observar en la gráfica 9, no se tienen casos de éxito cuando las personas no reciben capacitación a nivel preparatoria. Esto se refleja de esta manera debido a una poca atracción hacia la indagación de nuevas tecnologías por parte de los adolescentes.

$$p'_1 = \frac{0}{25} \quad p'_2 = \frac{13}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0)(1)}{25} + \frac{(0.3513)(0.6486)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.0784$$

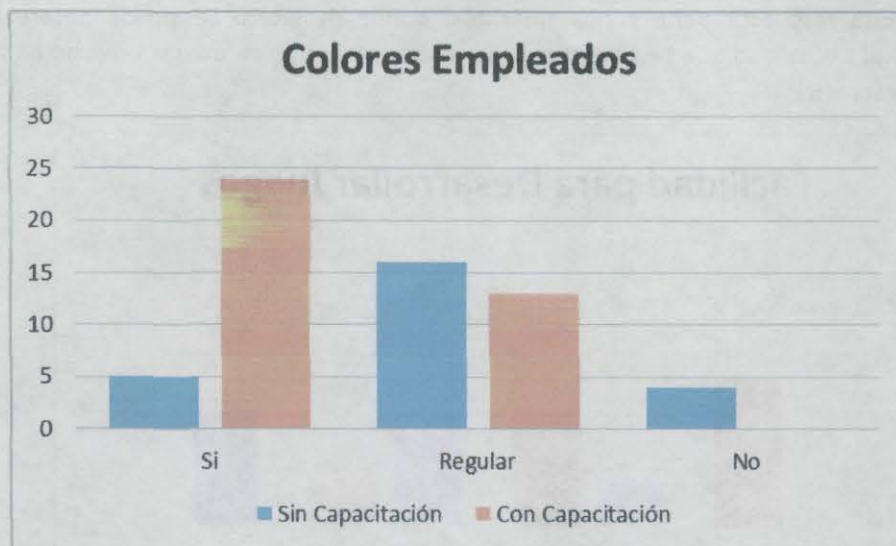
$$E = (1.96)(0.0784) \quad E = 0.1538$$

$$(0 - 0.3513) - 0.1538 < P_1 - P_2 < (0 - 0.3513) + 0.1538$$

$$-50.5\% < P_1 - P_2 < -19.75\%$$

Análisis Estadístico 9: Facilidad para Desarrollar Juegos en Preparatoria

El resultado de la inferencia comprueba la teoría marcada en el análisis de la gráfica, donde aproximadamente del 19% al 50% de las personas con capacitación les gusta más que a las personas sin capacitación desarrollar videojuegos con el sistema propuesto.



Gráfica 10: Aceptación de los colores empleados en Preparatoria

En la gráfica 10 se muestra que las personas con capacitación indican que les gustan los colores empleados en el sistema para indicar el estado de configuración de los cubos. Pero un alto porcentaje hace la recomendación de utilizar otra gama de colores para poder identificar las funcionalidades y los elementos con los que se deben trabajar.

Analizando por el método estadístico, se obtiene el mismo resultado. Donde se encuentra del 22.9% al 66.8% de diferencia en el gusto de los colores entre ambas poblaciones.

$$p'_1 = \frac{5}{25} \quad p'_2 = \frac{24}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.2)(0.8)}{25} + \frac{(0.6486)(0.3513)}{37}} \quad \sigma_{(p'_1 - p'_2)} = 0.1120$$

$$E = (1.96)(0.1120)$$

$$E = 0.2196$$

$$(0.2 - 0.6486) - 0.2196 < P_1 - P_2 < (0.2 - 0.6486) + 0.2196$$

$$-66.8\% < P_1 - P_2 < -22.90\%$$

Análisis Estadístico 10: Aceptación de los colores empleados en Preparatoria

De las tres preguntas de encuesta para preparatoria, se obtiene que los usuarios objetivos no desarrollaron un gusto por el sistema si antes no se les da una capacitación previa. Analizando todos los datos, la respuesta de ésta pregunta invalida la segunda parte de la hipótesis, arrojando la siguiente modificación: Para los usuarios que cursan educación primaria, el sistema se les hace intuitivo, pero se deben hacer cambios al diseño de los cubos para hacerlos más atractivos;

mientras que para los estudiantes de preparatoria, se debe buscar alguna estrategia que les permita recibir una capacitación auto-didacta.

5.2.3 PI3. ¿Qué flexibilidad puede llegar a otorgar el sistema para el desarrollo de videojuegos?

Con esta pregunta de investigación se planea conocer la calidad y la cantidad de videojuegos que los usuarios pueden desarrollar. El objetivo de conocer esto es conocer las modificaciones que se deben hacer a las librerías de videojuegos y las nuevas funcionalidades que deben desarrollarse.

5.2.3.1 Primaria

Para nivel primaria, se utiliza una sola pregunta de la encuesta (Capítulo 4 apartado 4.9). En esta pregunta se desea conocer sobre las funciones que tiene el sistema para desarrollar sus videojuegos, por lo que se obtiene la siguiente información:



Gráfica 11: Aceptación de las funciones para juegos en primaria

En la gráfica 11, la aceptación que tiene las funciones del juego es alta, lo que nos permitiría inferir que las funciones ya agregadas cumplen con los objetivos de los usuarios, y el desarrollo de nuevas funcionalidades debe estar basado en la tendencia que tienen los videojuegos hoy en día.

Al analizar los datos por el método estadístico se obtuvo que:

$$p'_1 = \frac{5}{16} \quad p'_2 = \frac{22}{49}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.3125)(0.6875)}{16} + \frac{(0.4489)(0.5510)}{49}} \quad \sigma_{(p'_1 - p'_2)} = 0.1359$$

$$E = (1.96)(0.1359)$$

$$E = 0.2664$$

$$(0.3125 - 0.4489) - 0.2664 < P_1 - P_2 < (0.3125 - 0.4489) + 0.2664$$

$$-40.3\% < P_1 - P_2 < 12.99\%$$

Análisis Estadístico 11: Aceptación de las funciones para juegos en Preparatoria

Los resultados del método estadístico comprueba que no hay diferencia entre los dos grupos de primaria, por lo que podemos deducir que el sistema si ofrece una buena flexibilidad para el desarrollo de videojuegos, y que contiene una cantidad de funciones que los usuarios consideran útiles para el desarrollo de sus aplicaciones.

5.2.3.2 Preparatoria

Para los alumnos de nivel preparatoria, las preguntas de la encuesta que servirán para responder la pregunta de investigación P3 son la E de la primera sección y la 6 y 7 de la segunda sección de las encuestas (Capítulo 4 apartado 4.9). De éstas preguntas se desprendieron los siguientes datos:



Gráfica 12: Datos sobre las funcionalidades para juegos en preparatoria

En la gráfica 12 se muestran pocos casos de éxito hacia la funcionalidad. El motivo de este bajo número de éxitos puede ser causado por el tipo de videojuegos que los adolescentes están acostumbrados a jugar. Por lo que para poder aplicar hacia este sector, se deben desarrollar librerías que permitan el uso de mejores gráficos.

$$p'_1 = \frac{2}{20} \quad p'_2 = \frac{10}{37}$$

$$\sigma_{(p'_1-p'_2)} = \sqrt{\frac{(0.1)(0.9)}{20} + \frac{(0.2702)(0.7297)}{37}} \quad \sigma_{(p'_1-p'_2)} = 0.0991$$

$$E = (1.96)(0.0991) \quad E = 0.1943$$

$$(0.1 - 0.2702) - 0.1943 < P_1 - P_2 < (0.1 - 0.2702) + 0.1943$$

$$-36.5\% < P_1 - P_2 < 2.41\%$$

Análisis Estadístico 12: La funcionalidad disponible en los cubos para preparatoria

Al analizar los datos estadísticamente, se comprueba lo visto en la gráfica. Ambos grupos tienen una buena aceptación hacia la funcionalidad para juegos que tiene el framework, pero entre los comentarios recolectados se muestra que se necesitan mejorar los gráficos que maneja el sistema.



Gráfica 13: Datos sobre la funcionalidad necesaria para crear videojuegos en preparatoria

Como interpretación de la gráfica 13 se puede decir que no todas las personas se encuentran de acuerdo que se tiene toda la funcionalidad completa para desarrollar un juego. La mayoría de estas personas son las que no recibieron una capacitación previa, comentando que al no conocer el sistema, no podían conocer qué acción realizaba cada funcionalidad.

Analizando éstos resultados por medio del método estadístico se obtiene que:

$$p'_1 = \frac{7}{20} \quad p'_2 = \frac{20}{37}$$

$$\sigma_{(p'_1-p'_2)} = \sqrt{\frac{(0.35)(0.65)}{20} + \frac{(0.5405)(0.4594)}{37}} \quad \sigma_{(p'_1-p'_2)} = 0.1344$$

$$E = (1.96)(0.1344) \quad E = 0.2635$$

$$(0.35 - 0.5405) - 0.2635 < P_1 - P_2 < (0.35 - 0.5405) + 0.2635$$

$$-45.4\% < P_1 - P_2 < 7.31\%$$

Análisis Estadístico 13: Datos sobre la funcionalidad necesaria para crear videojuegos en preparatoria

En éste resultado del método estadístico se observa que entre los casos exitosos que se tuvieron no hay diferencia. Esto nos demuestra que aunque la mayoría siente que falta funcionalidad, las personas que creen que la aplicación contiene toda la funcionalidad necesaria para crear un videojuego es balanceada, aunque hay que tomar en cuenta sus comentarios y comenzar a desarrollar nuevas funcionalidades que haga la aplicación más llamativa.



Gráfica 14: Facilidad para configurar un juego en preparatoria

En la gráfica 14 se muestra cómo los grupos que no recibieron capacitación creen que es difícil configurar cada uno de los parámetros. Estos resultados demuestran que para la mayoría de los estudiantes de preparatoria necesitan conocer todos los detalles de una nueva herramienta tecnológica antes de utilizarla.

$$p'_1 = \frac{7}{20} \quad p'_2 = \frac{34}{37}$$

$$\sigma_{(p'_1 - p'_2)} = \sqrt{\frac{(0.35)(0.65)}{20} + \frac{(0.9189)(0.0810)}{37}}$$

$$\sigma_{(p'_1 - p'_2)} = 0.1157$$

$$E = (1.96)(0.1157)$$

$$E = 0.2267$$

$$(0.35 - 0.9189) - 0.2267 < P_1 - P_2 < (0.35 - 0.9189) + 0.2267$$

$$-79.6\% < P_1 - P_2 < -34.21\%$$

Análisis Estadístico 14: Facilidad para encontrar y configurar un juego en Preparatoria

Con este resultado del método estadístico se demuestra que los usuarios de preparatoria deben recibir una introducción o capacitación previa hacia el sistema, para que puedan conocer mejor la forma en la que se configura cada uno de los cubos.

Al analizar todas las respuestas para resolver esta pregunta, se cumple el objetivo del sistema, que consiste en generar varios tipos de videojuegos utilizando una misma herramienta. Aunque se tienen que hacer algunas excepciones a nivel preparatoria por la capacitación que deben recibir.

Si analizáramos todas las respuestas dadas en las preguntas de investigación, obtenemos que la hipótesis solo se acepta hasta el punto de que sirva como una herramienta de introducción universitaria o para que los niños se vuelvan creadores, pero al no ser intuitivo todos los términos empleados se debe considerar preparar materiales digitales de capacitación.

5.3 Comentarios en las encuestas

Como una de las acciones de mejora continua, se necesita saber qué opinan los usuarios sobre el sistema y cómo proponen ellos mejorar tanto la interfaz como la funcionalidad con la que se cuenta.

Las preguntas que servirán para la mejora continua son la última parte de la encuesta para niños y las preguntas 4, 5 y 8 de la encuesta para adolescentes (Capítulo 4 apartado 4.9).

De las encuestas de los niños se obtuvo que los usuarios desean que el sistema contenga una librería de imágenes predefinida que permita desarrollar varios tipos de juegos, además de agregar otro tipo de funcionalidades para juegos con estilo Shooter, RPG o Aventuras. Hablando sobre gráficos, los estudiantes de primaria quieren gráficos más llamativos tanto en los videojuegos como en el framework.

Mientras que en las encuestas de los adolescentes, se tiene información donde piden que se facilite la forma en la que funciona el framework, cambiar la forma de trabajo o incluso preparar material virtual que puedan utilizar para conocer sobre el sistema. Uno de los cambios que más se piden es la generación de nuevas funcionalidades para los juegos.

Algunos de los comentarios recibidos por los usuarios fue: *"Cambiaría la distribución del framework y la representación de los cubos", "Me gustaría cambiar los colores y el diseño del cubo", "Me gustaría tener más acciones para juegos", "Mejorar las gráficas del cubo", "Sería buena idea poner antes un tutorial de su uso", "Quisiera tener una pre visualización del trabajo que estoy haciendo", "Poner sugerencias o tipos que se facilite y todos realmente lo podamos hacer", "Pondría una especie de guía en el mismo software", "Que el programa cuente con imágenes precargadas", "Pondría más imágenes y haría los cubos más pequeños", "Mejoraría un poco más los gráficos y daría más opciones para desarrollar mi juego", "Hacer más sencillo cuando tengo que poner los valores a los cubos".* Todos los comentarios que se obtuvieron fueron tomados en cuenta y se analizó su viabilidad para su desarrollo como un trabajo a futuro de esta investigación.

La vida es el arte de sacar conclusiones suficientes

A partir de datos insuficientes

-- Samuel Butler

Capítulo 6. Conclusiones y Trabajo a Futuro

Para concluir este trabajo de tesis, este capítulo se dedicará a mostrar las conclusiones y recomendaciones obtenidas a lo largo del trabajo en este proyecto, con el fin de dar continuidad al proyecto, así como mostrar los beneficios obtenidos.

6.1 Conclusiones

El objetivo principal de esta tesis era crear un framework mediante el cual los usuarios fueran capaces de crear videojuegos de manera rápida por medio de elementos gráficos que representaran funcionalidades para ser utilizado como una herramienta de introducción universitaria y para todas aquellas personas que quieran desarrollar videojuegos y no tuvieran la preparación para hacerlo.

El objetivo principal fue rebasado debido a la habilidad de las nuevas generaciones en la adaptación de las nuevas tecnologías. Al final de la tesis se obtuvo un sistema que permitió tanto a jóvenes como a niños desarrollar videojuegos de forma intuitiva.

Un aspecto importante de Cubecraft Game es la velocidad. La herramienta permite a los usuarios desarrollar un videojuego simple en un tiempo máximo de 20 minutos, donde el tiempo de respuesta del sistema para la generación de clases y la compilación de la aplicación hasta dejarla funcional es de aproximadamente 6 segundos, dependiendo de la velocidad de la computadora en la que se está ejecutando.

El tiempo de respuesta que muestra el sistema durante la compilación de juegos puede ser mejorado, por lo que se hacen algunas propuestas como trabajo a futuro en las secciones posteriores de éste capítulo.

Para poder concluir sobre la investigación de usabilidad y aceptación del framework planteada en ésta tesis es necesario analizar cada una de las preguntas de investigación planteadas.

De la primera pregunta de investigación *"¿La herramienta es intuitiva para el usuario?"*, podemos concluir que para los estudiantes de primaria si lo es, debido a la habilidad nombrada anteriormente. Para los grupos de preparatoria ésta pregunta de investigación no se cumple, por lo que se deben realizar modificaciones al framework que permita ayudar a éste grupo de enfoque a entender mejor su uso.

Para la segunda pregunta de investigación *"¿Cuál es la aceptación que tiene el sistema entre los usuarios objetivo?"*, el grupo de primaria se encuentra satisfecho, pero en un porcentaje menor del 36% de la población total, por lo que se deben buscar nuevas estrategias que permita aumentar el número de personas satisfechas con el uso del sistema. Con los grupos de

preparatoria ésta pregunta de investigación no se cumple, con lo que podemos concluir que al menos el 60% de los estudiantes prefieren una capacitación total de la tecnología.

Con la tercera pregunta de investigación “¿Qué flexibilidad puede llegar a otorgar el sistema para el desarrollo de videojuegos?”, los usuarios de primaria y preparatoria muestran una diferencia de opiniones. Para los estudiantes de primaria, el sistema otorga las herramientas suficientes para el desarrollo de cualquier tipo de juego, y para los estudiantes de preparatoria también se cumple el objetivo, con la única diferencia que ellos deben tener la capacitación para utilizar el framework.

De éstas tres preguntas de investigación, la hipótesis planteada “El sistema propuesto permite a los niños en edades de 10 a 12 años y adolescentes de 15 a 17 años a entender y desarrollar videojuegos para dispositivos móviles de una manera sencilla, sin importar que reciban o no capacitación sobre su uso” se cumple parcialmente. Podemos concluir a partir de los comentarios orales y escritos de los usuarios que el sistema se valida parcialmente debido a que éstas nuevas generaciones se encuentran acostumbrados a videojuegos con gráficos de alta definición desarrollados por grandes equipos de trabajo y muchos años de desarrollo.

Al igual que con el tiempo de respuesta con el que se cuenta, toda la información recabada para la mejora continua del sistema servirá como base para las propuestas de un trabajo a futuro en las siguientes secciones del capítulo.

6.2 Trabajo a Futuro

Como se explicó en las conclusiones, en este trabajo de tesis se tuvieron ligeras variaciones entre lo propuesto y los resultados obtenidos con los estudiantes. Es por ello que a partir de las conclusiones anteriores y de los comentarios de los estudiantes se planea realizar las siguientes actividades en un futuro:

- **Agregar tips y videos de uso dentro del framework:** Se tienen que generar ventanas de tips sobre el uso del sistema cada vez que el usuario inicie el framework. Con estos tips se espera que los usuarios puedan conocer usos del framework que pudieran ser complejos. En cuanto a los videos, se planea que sea una forma gráfica que los usuarios puedan visitar para conocer el uso correcto del framework; los videos deben contener ejemplos prácticos del desarrollo y explicaciones de cada uno de los pasos realizados.
- **Aumentar la calidad de imagen de los cubos (dpi) y reducir su tamaño:** Se deben cambiar la calidad de los cubos para que puedan ser mostrados en equipos que cuenten con cualquier tarjeta gráfica. Al reducir su tamaño, se espera que los usuarios puedan explotar al máximo el área de trabajo con el que cuentan.
- **Permitir que los cubos puedan unirse:** Uno de los comentarios de los usuarios es la retroalimentación sobre qué cubo de funcionalidad está ligado a qué cubo maestro. Para ello se debe buscar una forma de unir los cubos, ya sea mediante conexiones por medio de líneas o uniendo los cubos utilizando proyecciones isométricas.

- **Ingresar mayor número de funcionalidades al sistema:** Se deben mejorar y generar nuevas funcionalidades que puedan agregarse al sistema para permitir mejores juegos. Se deben realizar funcionalidades para juegos de estilo RPG, Multijugador o de primera persona; mientras que, como funcionalidad de configuración se debe permitir que se guarden preferencias y que se utilice el sistema de inicio de sesión del Web Service desarrollado.
- **Cambiar el sistema de gráficos a OpenGL:** Al cambiar el sistema de gráficos esperamos que la aceptación de los juegos aumente. Al utilizar gráficos OpenGL se tendrá que aumentar el peso de la aplicación y la cantidad de recursos utilizados del dispositivo, pero los gráficos serán más profesionales.
- **Desarrollar librerías para otros Sistemas Operativos:** Actualmente el sistema está preparado para trabajar con cualquier tipo de Sistema Operativo cuyo lenguaje de programación esté basado en Java. Es por esto que se deben comenzar a desarrollar nuevas librerías para cualquier tipo de sistema operativo sin importar que sea móvil o para PC.
- **Cargar imágenes pre-diseñadas para juegos:** Para algunos de los usuarios, la búsqueda de imágenes en internet para el desarrollo de sus videojuegos es difícil, debido a que debe cumplir ciertas características como el tipo y el tamaño de la imagen. Es por ello que se planea agregar una galería de imágenes pre-diseñadas originales de personajes para el desarrollo de juegos que cumpla con todas las características necesarias.
- **Manejar un sistema de licencias anuales:** En este momento el sistema es libre debido a que se encuentra en una versión de pruebas. Una vez que el sistema sea terminado y que toda la funcionalidad y la calidad de los gráficos de juegos sea mejorada se manejará un sistema de licencias que deberá ser analizado dependiendo de los tipos de usuario detectados.
- **Creación del sitio web "Cubegame":** Es necesario desarrollar toda la funcionalidad del sitio web, para que los usuarios puedan subir sus videojuegos desarrollados y recibir retroalimentación de cómo lo pueden mejorar, además de poder descargar actualizaciones del sistema o nuevas funcionalidades.

Como meta, la arquitectura debe proponernos la creación de relaciones nuevas
Entre el hombre, el espacio y la técnica.

— Hans Scharoun

Apéndice A. Arquitectura de Software

Este documento contiene el diseño elaborado, el cual es producto de un análisis minucioso de los requisitos del sistema, estos pueden ser satisfechos con las tecnologías y características detectadas.

Este documento de Arquitectura de Software, puede ser usado para comprender el diseño y construcción del framework, y sirve como base para que los desarrolladores de software puedan construir el bajo nivel de la aplicación utilizando el lenguaje que más les convenga.

La arquitectura está mostrada como una serie de vistas basadas en la arquitectura de software del modelo 4+1 de Kruchten. Estas vistas son:

- La vista de escenarios: Son de alguna manera, una abstracción de los requisitos más importantes. Su diseño se expresa mediante el uso de diagramas de escenarios y diagramas de iteración de objetos. Estas vistas es redundante con las otras, pero sirve a dos propósitos principales:
 - Como una guía para descubrir elementos arquitectónicos durante el diseño de arquitectura.
 - Como un rol de validación e ilustración después de completar el diseño de arquitectura, en el papel como punto de partida de las pruebas de un prototipo de la arquitectura.
- La vista lógica: Apoya principalmente los requisitos funcionales. El sistema se descompone en una serie de abstracciones clave, tomadas del dominio del problema en la forma de objetos o clases de objetos. Aquí se aplican los principios de abstracción, encapsulamiento y herencia. Esta descomposición no solo se hace para potenciar el análisis funcional, sino también sirve para identificar mecanismos y elementos de diseño comunes a diversas partes del sistema.
- La vista de procesos: Toma en cuenta algunos requisitos no funcionales tales como el rendimiento y la disponibilidad. Se enfoca en asuntos de concurrencia y distribución, integridad del sistema, de tolerancia a fallas. La vista de procesos también especifica en cual hilo de control se ejecuta efectivamente una operación de una clase identificada en la vista lógica.

- La vista de desarrollo: La vista de desarrollo se centra en la organización real de los módulos de software en el ambiente de desarrollo del software. El software se empaqueta en partes pequeñas que pueden ser desarrollados por uno o un grupo pequeño de desarrolladores. Los subsistemas se organizan en una jerarquía de capas, cada una de las cuales brinda una interfaz estrecha y bien definida hacia las capas superiores.
- La vista física: Toma en cuenta primeramente por requisitos no funcionales del sistema tales como la disponibilidad, confiabilidad (tolerancia a fallas), rendimiento y escalabilidad. El software se ejecuta sobre una red de computadoras o nodos de procesamiento (o tan solo nodos). Los variados elementos identificados requieren ser mapeados sobre los variados nodos. Esperando que diferentes configuraciones puedan usarse: algunas para desarrollo y pruebas, otras para emplazar el sistema en varios sitios para distintos usuarios. Por lo tanto, el mapeo del software en los nodos requiere ser altamente flexible y tener un impacto mínimo sobre el código fuente en sí.

No hay ninguna vista separada de una misma implementación, descrita en este documento, éstas vistas están hechas sobre el Lenguaje de Modelado Unificado (UML) en su versión 2.0.

La lógica de diseño arquitectónico que se aplicará a este documento, abre la posibilidad de que la implementación de bajo nivel sea efectuada con lenguajes que solamente cumplan con las características de un Lenguaje Orientado a Objetos. Y esto va a depender directamente de las características de los desarrolladores y su capacidad de aprendizaje. Si la implementación se desea desarrollar bajo lenguajes que no cumplan las características mencionadas, se deberá confeccionar una nueva arquitectura que cumpla con los requerimientos funcionales y no funcionales que los stakeholders solicitan.

A.1 Framework Conceptual

A continuación se definirán los Stakeholders, sus roles y sus preocupaciones en las que está basada ésta arquitectura.

Stakeholder	Descripción	Preocupaciones
Niños de 3ro de Primaria (8-9 años).	El Framework está dirigido a niños que oscilen entre 8 y 9 años de edad. El sistema deberá permitirles desarrollar videojuegos de una forma rápida y sencilla.	➤ Tener funcionalidades suficientes para crear juegos dinámicos ➤ Generar sus ejecutables para el dispositivo ➤ Tener una retroalimentación grafica del estado del juego. ➤ Fácil acceso a las características de la funcionalidad ➤ Acceso a imágenes y

Desarrolladores

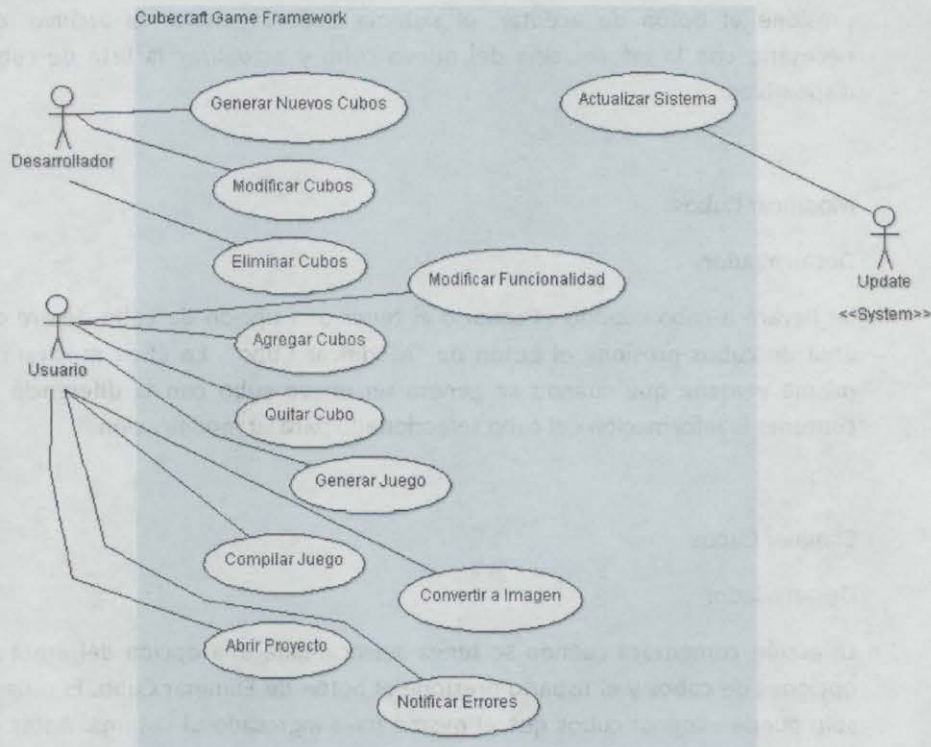
Deben conocer a profundidad el framework y la forma de desarrollar funcionalidades al sistema. Su función es generar mejores acciones de juego para los niños.

sonido para agregarlo al proyecto.

- Acceso al código fuente de las API's para generar mejoras
- Generación de código propio para la API
- Carga de códigos personales en todos los proyectos.

A.1.1 Vista de Casos de Uso

La vista de casos de uso indican gráficamente las acciones que puede llevar a cabo los stakeholders tomando en cuenta sus preocupaciones.



Nombre	Actualizar Sistema
Actores	Update (Sistema)
Descripción	Esta acción se lleva a cabo cada vez que el framework es abierto. Debe comunicarse con el módulo de UpdateSystem localizado en el servidor web y debe enviar una lista de las clases y cubos que se tienen instalados y la versión a la que corresponde, para descargar actualizaciones. Al terminar la descarga de los archivos deben ser posicionados en sus carpetas correspondientes. Una vez terminada la actualización, se le debe indicar al usuario que para poder usar las nuevas modificaciones, se debe reiniciar el framework.

Nombre	Generar Nuevos Cubos
Actores	Desarrollador
Descripción	Este caso de uso debe realizarse cuando el usuario, seleccione la opción de "Nuevo Cubo" en el sistema. Al ejecutarse, se mostrará una ventana donde se le pedirán datos para el cubo (Sistema Operativo en el que funciona, El nombre de la funcionalidad a la que corresponde y los atributos que se esperan en el constructor) y el código fuente de la funcionalidad. Una vez que el usuario presione el botón de aceptar, el sistema deberá generar el archivo .dat necesario con la información del nuevo cubo y actualizar la lista de cubos disponibles.

Nombre	Modificar Cubos
Actores	Desarrollador
Descripción	Se llevará a cabo cuando el usuario al tener una opción de cubo dentro del área de cubos presione el botón de "Modificar Cubo". En él se mostrará la misma ventana que cuando se genera un nuevo cubo con la diferencia de contener la información del cubo seleccionado para su modificación.

Nombre	Eliminar Cubos
Actores	Desarrollador
Descripción	La acción comenzará cuando se tenga seleccionada una opción del árbol de opciones de cubos y el usuario presione el botón de Eliminar Cubo. El usuario solo puede eliminar cubos que él mismo haya ingresado al sistema. Antes de llevar a cabo la eliminación el sistema debe mostrar una alerta indicándole al usuario que una vez eliminada la opción no podrá ser restaurada. En caso de que el usuario diga que sí, entonces se eliminará de la carpeta el archivo .dat

que corresponda a la opción.

Nombre	Agregar Cubos
Actores	Usuario
Descripción	Todo usuario puede agregar un cubo de funcionalidad que se encuentre en el árbol de cubos, para llevar a cabo la acción, el usuario debe seleccionar la opción que quiere del árbol y arrastrarla al área de trabajo. En caso de ser el primer cubo, el cubo se posicionará en el lugar donde se suelte el mouse, en caso contrario, el cubo se posicionará en una de las arista del cubo más cercano.

Nombre	Modificar Funcionalidad
Actores	Usuario
Descripción	Cuando el cubo se encuentre en el área de trabajo, se puede modificar las opciones de configuración. Dependiendo del tipo de variable que se tenga registrada en el archivo .dat, será el tipo del campo que se mostrara (campo de texto para un tipo String, una lista desplegable para objetos, etc). Según la sensibilidad del dato, el usuario recibirá retroalimentación por la coloración del cubo (rojo para error, amarillo para advertencia y verde para correcto).

Nombre	Quitar cubo
Actores	Usuario
Descripción	Cuando un cubo se encuentre en el área de trabajo, se puede eliminar presionando la tecla Suprimir del teclado o dando click derecho y seleccionando la opción "Quitar". Al llevarse a cabo éste caso de uso, el sistema mostrará una alerta al usuario confirmando la eliminación del cubo. Si el cubo se encontraba rodeado de otros, éstos se reorganizarán quitando el espacio dejado por el cubo eliminado.

Nombre	Generar Juego
Actores	Usuario
Descripción	Este caso de uso inicia cuando el usuario presiona el botón de "Verificar" en el menú Proyecto. Al ejecutarse el caso de uso, el framework genera las clases (según el sistema operativo) necesarias para compilarlo y generar el ejecutable.

Nombre	Compilar Juego
Actores	Usuario
Descripción	Este caso de uso es una extensión del caso de uso "Generar Juego". Una vez que el juego se encuentra compilado y se ha generado el ejecutable, el sistema hace un llamado al emulador del sistema operativo, instala el juego desarrollado y le permite al usuario interactuar con la funcionalidad que haya agregado de forma virtual.

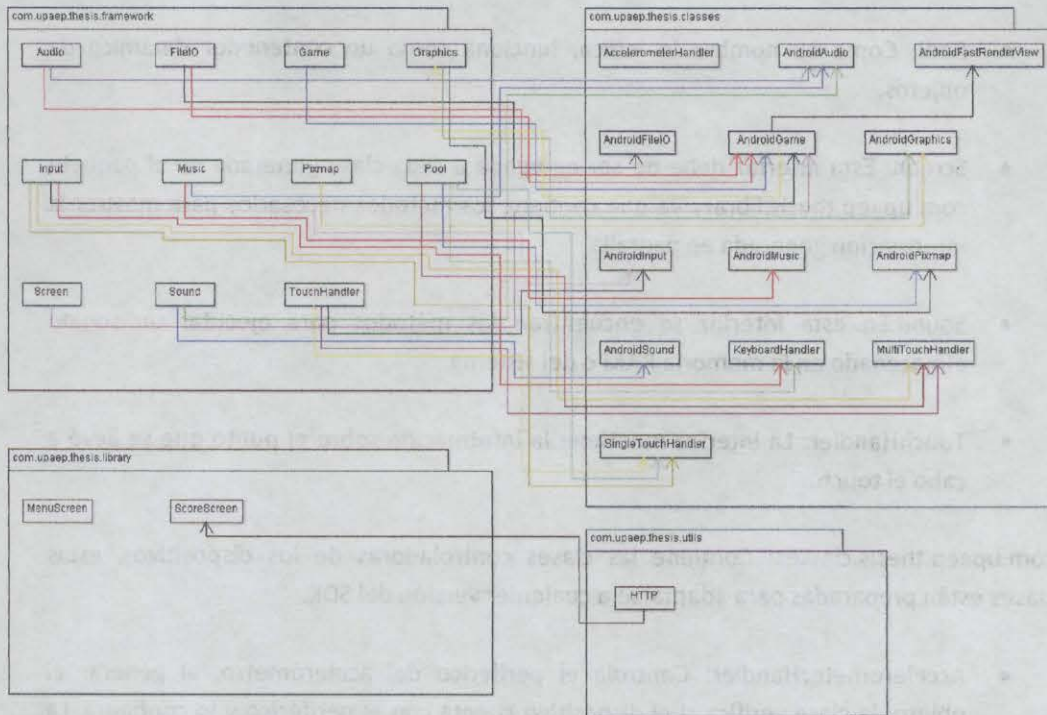
Nombre	Abrir Proyecto
Actores	Usuario
Descripción	El usuario al presionar el botón Abrir, el sistema deberá revisar la integridad de cada uno de los proyectos, los que se encuentren bien, el sistema los mostrará con un icono de un celular y una letra mayúscula inicial del sistema operativo (A o J). Al abrir un proyecto, si se encontraba otro proyecto abierto, el sistema pedirá confirmación de guardar el progreso.

Nombre	Convertir a Imagen
Actores	Usuario
Descripción	El caso de uso inicia cuando el usuario presiona la opción de "Exportar como Imagen". El framework generará una imagen en formato "png" de los cubos que se encuentren agregados en el proyecto. Una vez terminada la acción, guardará en disco (en la ubicación que el usuario elija) el archivo.

Nombre	Notificar Errores
Actores	Usuario
Descripción	Al estar trabajando con los cubos, en la consola de errores se muestra una retroalimentación escrita de los errores o advertencias que da el sistema. El caso de uso inicia cuando el usuario hace un doble clic encima de una de estas opciones, la cual deberá buscar y seleccionar el cubo que contiene ese error y permitirle al usuario cambiarlo correctamente.

A.1.2 Vista Lógica

En la vista lógica, se muestra de forma general las clases generadas en cada una de las capas:



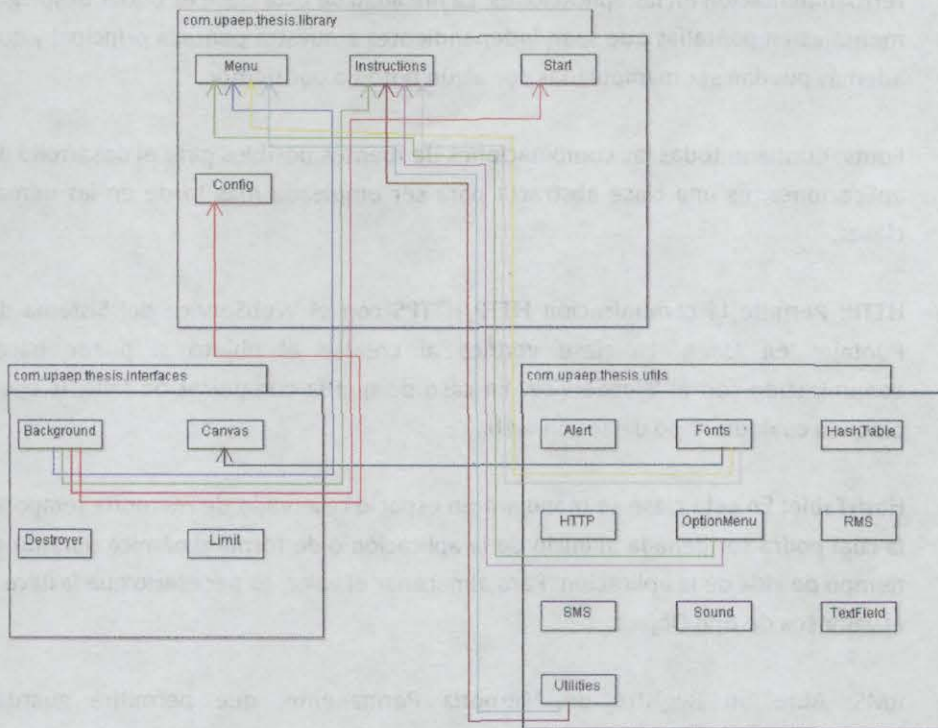
Para la API de Android, los paquetes y clases generadas son:

- **com.upaep.thesis.framework:** Este paquete contiene las interfaces de controladores. Más adelante, estas interfaces serán utilizadas como base para las clases controladoras de periféricos.
 - **Audio:** Esta interfaz será utilizada para la creación de elementos sonoros.
 - **FileIO:** Maneja la lectura/escritura de archivos en memorias externas SD o en memoria interna del teléfono.
 - **Game:** Controla la pantalla del dispositivo, al implementarse, debe indicar el cambio de estado y de información a mostrar.
 - **Graphics:** Cuando sea implementada, su función será funcionar como pincel dentro de las pantallas.
 - **Input:** La interfaz contiene los métodos necesarios para el control de los periféricos de entrada del dispositivo (teclado y touch).

- **Music:** Esta interfaz será utilizada para el control de archivos de audio (mp3,wav) desde un archivo.
 - **Pixmap:** Al implementarse, permite el control y generación de imágenes como mapas de bits.
 - **Pool:** Como su nombre lo indica, funciona como un contenedor dinámico de objetos.
 - **Screen:** Esta interfaz debe de ser agregada a cada clase generada en el paquete `com.upaep.thesis.library` ya que contiene los métodos necesarios para mostrar la información generada en pantalla.
 - **Sound:** En esta interfaz se encuentran los métodos para ejecutar un sonido almacenado en la memoria RAM o del sistema.
 - **TouchHandler:** La interfaz mantiene la información sobre el punto que se llevó a cabo el touch.
- **com.upaep.thesis.classes:** Contiene las clases controladoras de los dispositivos, estas clases están preparadas para adaptarse a cualquier versión del SDK.
- **AccelerometerHandler:** Controla el periférico del acelerómetro, al generar el objeto, la clase verifica si el dispositivo cuenta con el periférico y lo configura. La clase se queda en espera de algún cambio del periférico y lo guarda dentro de la clase para su uso futuro. Si el dispositivo cuenta con más de un acelerómetro, la clase elige el primero en la lista.
 - **AndroidAudio:** En esta clase se implementa la interfaz Audio, y contiene los constructores de objetos según el tipo de archivo a reproducir. Al iniciar, el sistema configura la Pool de sonidos y el control de volumen, en caso de que la configuración falle, la clase no permitirá generar sonido alguno.
 - **AndroidFastRenderView:** Contiene el hilo de ejecución que se encarga de actualizar los gráficos mostrados en la pantalla cada cierto tiempo o cuando se lleva a cabo alguna acción. Este hilo nace cuando la aplicación es llamada y no debe detenerse hasta que sea cerrada. Además, no debe permitir que la pantalla entre en estado de suspensión mientras se encuentre activa.

- **AndroidFileIO:** Al implementarse la interfaz FileIO, ésta clase contiene los métodos para abrir, guardar y leer los archivos localizados en el paquete APK o en la memoria SD del dispositivo. Al generarse el objeto, la clase busca la dirección de la memoria SD y de la memoria interna, dejando una variable de estado si la memoria externa se encuentra disponible o no.
- **AndroidGame:** Esta clase debe ser heredada hacia la clase principal del juego, en ella se encuentra el constructor de aplicaciones Android (onCreate) y su función en indicar al resto de las clases el ancho, alto y resolución de la pantalla, además de generar los objetos de todos los controladores descritos en éste paquete.
- **AndroidGraphics:** Contiene los métodos para dibujar en pantalla, las figuras geométricas disponibles para dibujar son: pixel, línea y rectángulo, pero también permite dibujar texto e imágenes dentro de un canvas determinado por el tamaño de la pantalla.
- **AndroidInput:** Funciona como un controlador y pool de los eventos realizados con los periféricos del dispositivo. La clase envía a los objetos del acelerómetro, teclado y touch configurarse e indicarle el tamaño disponible de toque, además de permitir obtener información sobre los eventos generados.
- **AndroidMusic:** Carga el sonido a reproducir en la memoria RAM del dispositivo y se prepara para reproducirlo. La clase puede configurarse para reproducir indefinidamente el sonido, o indicar el número de veces que se quiere se reproduzca el sonido. Cuando la aplicación se encuentra en modo "debug", si el sonido no ha podido ser cargado, se indica en consola el motivo por el cual falló.
- **AndroidPixmap:** Al generar un objeto de ésta clase, se indica el mapa de bits que se utiliza y el formato en el que se va a mostrar, éstos formatos pueden ser ARG8888 (colores de 32 bits), ARGB4444 (colores de 16 bits) o RGB565 (Colores de 16 bits guardados en 11 bits).
- **AndroidSound:** Ejecuta un sonido guardado dentro del paquete APK o que se encuentre dentro del sistema, este sonido no puede ser repetido indefinidamente, en caso de querer hacerlo, el sonido debe ser reproducido indefinidamente a nivel de código.
- **KeyboardHandler:** Contiene la pool con todos los eventos generados al presionarse las teclas del dispositivo, puede almacenar un máximo de 128 caracteres (numero máximo de teclas en un dispositivo Android) y permite conocer el estado en el que se encuentra (Presionado, No Presionado).

- **MultiTouchHandler:** Cuando el nivel de la API del dispositivo Android es mayor a 5 (Eclair) entonces se genera ésta clase. En ella se puede controlar los eventos generados por el manejo de toques múltiples.
 - **SingleTouchHandler:** Es generado solo cuando el nivel de la API de Android es menor de 5 y es para todos aquellos dispositivos cuyas pantallas aún no soportan el multitouch, la clase guarda la información del toque simple en la posición detectada.
- **com.upaep.thesis.utils:** Contiene utilerías para el framework, para agregar valor a los juegos.
- **HTTP:** La clase busca los puertos de internet activos en el dispositivo y trata de conectarse al Webservice del sistema. Para verificar si el dispositivo puede conectarse, envía un PING hacia el webservice y se recibe el estado del servidor. En caso el dispositivo no pueda comunicarse, entonces el sistema destruye el objeto y no permite su comunicación.
- **com.upaep.thesis.library:** Contiene todas las interfaces graficas que pueden ser utilizadas en el framework. Cada una de las funcionalidades deben ser llamadas desde estas clases.
- **Menu:** Esta clase muestra en pantalla un menú de opciones de 0 a n. Una vez ingresados, la clase adapta el número de opciones al tamaño y resolución de la pantalla, las opciones pueden contener texto o imagen, pero no ambas. Al crear el objeto, se pide el título del juego para mostrarlo en el menú, y se debe indicar el tipo de acción que tendrá cuando sea seleccionada; los tipos de acciones que pueden ser elegidos son: cargar, cambiar valor o silenciar
 - **Score:** Esta pantalla muestra una lista de puntajes, dependiendo el modo en el que se active, será la acción que tomara. En el modo Online, se genera un objeto HTTP y trata de comunicarse con el webservice. Una vez hecha la conexión y al cargarse los puntajes son mostrados en orden ascendente. El otro modo es el modo Offline, en él, los puntajes son guardados en un archivo de texto plano dentro de la memoria extraíble (si se encuentra disponible) y es leído cada vez que se solicita. Al igual que la clase MenuScreen permite ingresar un título a la pantalla y acepta solo una opción para regresar al menú anterior.



Para la librería de Symbian (J2ME) se tienen los siguientes paquetes y funcionalidades:

- **com.upaep.thesis.interfaces:** Contiene las interfaces y clases abstractas necesarias para la generación de la librería, su función es no repetir código dentro de las clases.
 - **Limit:** Se encarga de mantener los valores del tamaño de la pantalla y heredárselos a sus clases derivadas.
 - **Destroyer:** La función de esta interfaz es permitir finalizar la aplicación desde el hilo de ejecución.
 - **Background:** Con esta super-clase, todas las clases derivadas podrán tener las funciones necesarias para el control de la aplicación, así como también activar el sistema de límites para la pantalla. La funcionalidad de esta clase es para que el hilo de ejecución del Canvas no tenga que buscar la información, si no que todo esté en un patrón de diseño.
- **com.upaep.thesis.utils:** Contiene las herramientas utilizadas en la librería, además de algunas clases de funcionalidad no visible al usuario como el envío de mensajes y la conexión a internet.

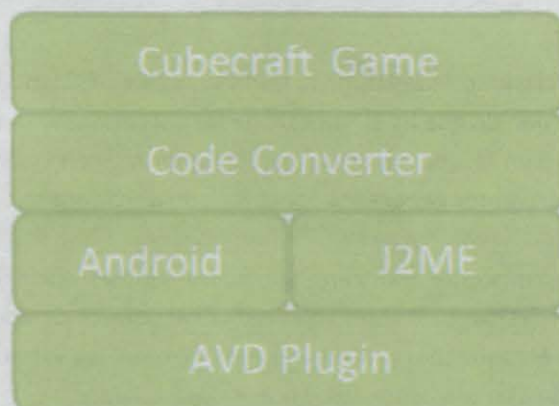
- **Alert:** En esta clase se crearán los métodos para poder utilizar las alertas de retroalimentación en las aplicaciones. La finalidad de esta clase es poder desplegar mensajes en pantallas que sean independientes a nuestra pantalla principal y que además puedan ser manipuladas por algún botón o por tiempo.
- **Fonts:** Contiene todas las combinaciones de fuentes posibles para el desarrollo de aplicaciones, es una clase abstracta para ser empleada más tarde en las demás clases.
- **HTTP:** Permite la comunicación HTTP/HTTPS con el Webservice del Sistema de Puntajes en Línea. La clase verifica al crearse el objeto si puede hacer comunicación con el Webservice. En caso de que la comunicación falle, la clase bloquea cualquier tipo de acción web.
- **HashTable:** En esta clase se manejará un espacio reservado de memoria temporal la cual podrá ser llenada al inicio de la aplicación o de forma dinámica durante el tiempo de vida de la aplicación. Para almacenar el valor, es necesario que la llave y el valor sea de tipo Object.
- **RMS:** Abre un Registro de Memoria Permanente, que permitirá guardar información de forma local en el dispositivo. En caso de que el registro ya se encuentre creado, la clase agregará valores al final del registro. Todos los valores guardados en él son de tipo byte.
- **SMS:** Esta clase, como su nombre lo indica, controla el envío de mensajes de texto cortos (SMS), el envío de los mensajes está limitado a que el dispositivo permita el control del sistema SMS y que se tenga servicio.
- **Sound:** Esta clase se encargará de manejar los diferentes formatos de sonido ya sean MP3, WAV, MIDI e incluso los sonidos predeterminados por el Sistema Operativo del dispositivo.
- **TextField:** Genera un cuadro de texto genérico de bajo nivel, con los bordes redondeados, para permitir a los usuarios interactúen los usuarios con el sistema. Al ser de bajo nivel, las acciones que se pueden llevar a cabo son limitadas.
- **Utilities:** Contiene varias herramientas que son necesarias para la mejora visual de la aplicación. Con esta clase se podrán realizar alertas personalizadas, textos sombreados e incluso aclarar u oscurecer colores.

➤ `com.upaep.thesis.library`: Contiene todos los módulos gráficos que pueden ser implementados en los juegos.

- **Canvas**: Esta clase no muestra nada gráficamente en la pantalla, pero es esencial en la librería, debido que es el “motor” de las aplicaciones. Su función es controlar la frecuencia con la que se actualizan los gráficos, el cambio de pantallas y el manejo de los eventos generados por el teclado del dispositivo.
- **Config**: Esta pantalla le mostrará al usuario algunas opciones para cambiar la configuración del juego. El objetivo es reducir el número de opciones de los menús y permitir un enfoque más profesional. Las opciones, los valores y las acciones que puede llevar a cabo son controladas por el desarrollador.
- **Start**: En esta clase se va a poder realizar el inicio del juego de una forma más profesional, donde se muestra el nombre de la aplicación, su logo y el nombre del desarrollador. El tiempo que se muestra ésta pantalla depende del elegido por el usuario.
- **Instructions**: Esta clase genera una pantalla donde se puede mostrar texto, sin un límite y una opción para alguna otra pantalla que se necesite (Regresar al menú anterior, Entrar al juego, etc.). La clase calcula el número de líneas necesarias para mostrar el texto y muestra una barra de desplazamiento como retroalimentación al usuario.
- **Menu**: Esta clase tiene todos los datos necesarios para poder crear un menú en el cual el desarrollador podrá agregar un número de opciones tales que puedan mostrarse en el tamaño de la pantalla.

A.1.3 Vista de Desarrollo

La vista de desarrollo del framework consiste en un modelo de capas donde se muestra el seguimiento del sistema para la creación de los ejecutables.



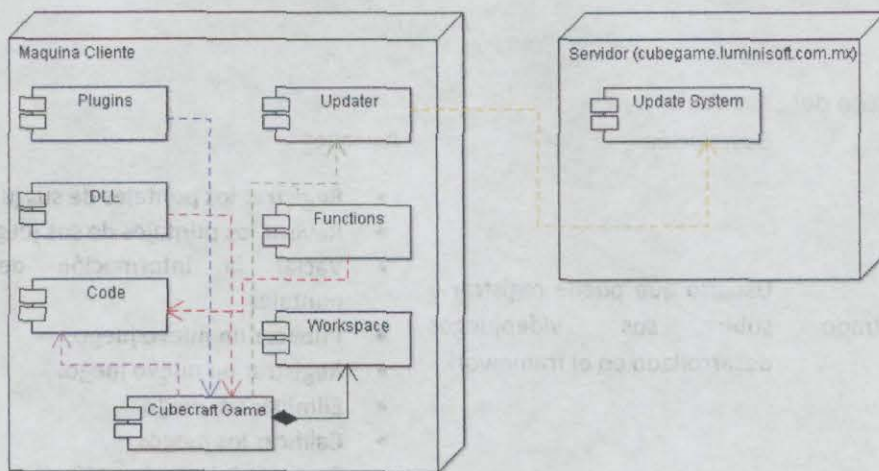
Cada una de las capas tiene la siguiente funcionalidad:

- Capa 1: AVD Plugin: Este Plugin es el otorgado por Oracle y Google, contiene simuladores de equipos y los generadores de código para cada tipo de dispositivo. El objetivo de esta capa es permitir que se generen los archivos APK o JAR a partir de las clases generadas por las capas superiores.
- Capa 2: Android: En esta capa se tiene todo el código de la API generada para Android. Toda la funcionalidad se encuentra distribuida en paquetes según su funcionalidad. Para que el framework pueda acceder a ella, se encuentran en archivos separados dentro de la carpeta "Functions" que la capa 4 (CodeConverter) tomará para construir las clases necesarias.
- Capa 3: J2ME: Esta capa se encuentra al mismo nivel de la capa 2 (Android) debido a que también contiene la librería API con funcional parecida a la de Android, utilizada por el CodeConverter para generar las clases necesarias por el AVD Plugin y generar el .jar firmado.
- Capa 4: CodeConverter: Este módulo se encargará de generar las clases y archivos XML necesarios por la capa AVD Plugin y generar los archivos apk o jar correctamente. Este módulo comenzará a funcionar una vez que el usuario del sistema ejecute la acción de compilar. Las clases generadas se encontrarán en una carpeta específica dentro del Workspace del framework.
- Capa 5: CubecraftGame: Este módulo contiene la interfaz gráfica del framework, su función principal es permitir al usuario generar su juego mediante el arrastre de cubos al

área de trabajo. Su tarea principal con las demás capas consiste en indicar el momento en que cada capa debe iniciar e informar al usuario sobre el resultado obtenido.

A.1.4 Vista Física

El framework está diseñado para ser una aplicación stand-alone, con conexión a internet solo cuando necesite actualizarse. La distribución dentro del disco duro se encuentra por carpetas según el modulo al que corresponda.



Cada una de las carpetas mostradas en la imagen, se describen:

- **Plugins:** En esta carpeta se encuentran los compiladores y simuladores de las tecnologías Android y J2ME, estos paquetes serán contenidos junto con el instalador del sistema.
- **Updater:** Aquí se encuentran los archivos que el sistema carga cuando se realiza una búsqueda de actualizaciones. El sistema no podrá ser actualizado manualmente, solo se podrá hacer desde el sistema. Dentro de la carpeta se encontrarán los archivos DLL necesarios para llevar a cabo esta acción.
- **DLLs:** Contiene los archivos de funcionalidad del framework como el Drag&Drop, el cambio de estados, entre otros. Estos archivos deben bloquearse para que los usuarios no puedan modificarlo y causar un error en el sistema.
- **Functions:** Dentro de esta carpeta se encuentran las funcionalidades cargadas al sistema, se encuentran separadas según el sistema operativo al que correspondan (Android, J2ME) junto toda aquella función generada por los usuarios finales.

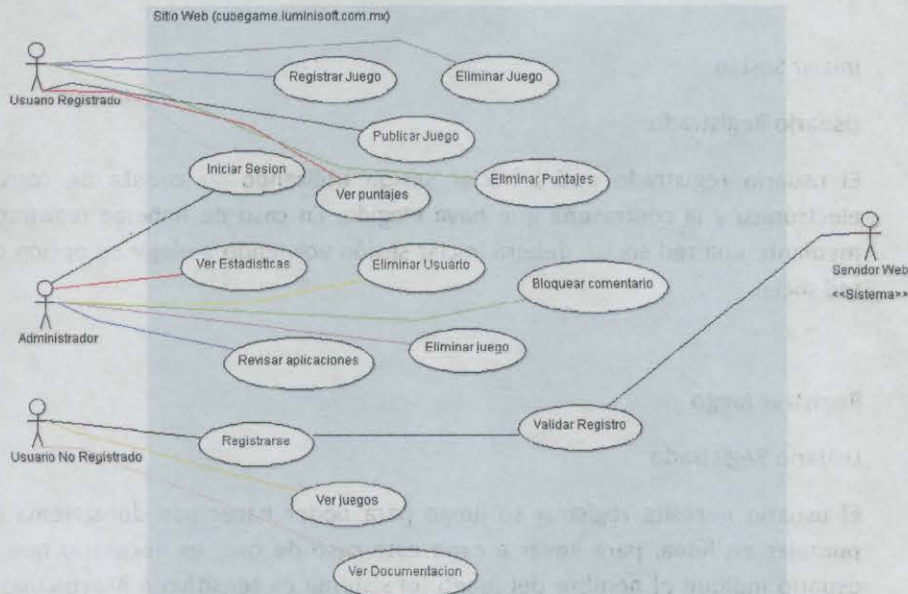
- **Code:** Esta carpeta corresponde a la capa de CodeConverter, aquí se encuentran las funciones (en formato DLL) para convertir los cubos en código fuente del sistema operativo móvil seleccionado.
- **Workspace:** Esta carpeta puede encontrarse dentro de la carpeta del framework o en cualquier lugar que el usuario elija como su espacio de trabajo. Esta carpeta contiene todos los proyectos que el usuario inicie.
- **UpdateService:** Este servicio se encuentra en el servidor web, su función es enviar una lista de las versiones actuales al framework, e indicarle los cambios que sufrirá. Si el usuario elige aceptar la actualización, el servicio empezará la descarga de los archivos.

A.2 Arquitectura del Sitio Web

Stakeholder	Descripción	Acciones
Usuario Registrado	Usuario que puede registrar y subir sus videojuegos desarrollado en el framework.	• Registrar los puntajes de sus juegos • Revisar los puntajes de sus juegos • Vaciar la información de sus puntajes • Publicar un nuevo juego. • Registrar un nuevo juego. • Eliminar un juego. • Calificar los juegos • Dar su opinión sobre el juego
Usuario Registrado	No Son los visitantes del sitio web, usualmente entran a conocer sobre el sistema, los juegos desarrollados y la documentación del mismo.	• Visitar los juegos publicados • Conocer la documentación del framework
Administrador	Personas encargadas de eliminar proyectos que nunca fueron utilizados y mantener la información del sitio actualizada.	• Mantener las estadísticas de los usuarios • Revisar la integridad de las aplicaciones • Eliminar usuarios • Eliminar aplicaciones • Bloquear comentarios

A.2.1 Vista de Casos de Uso

Los casos de uso de las acciones que se pueden llevar a cabo en el sitio web son las siguientes:



Nombre	Registrarse
Actores	Usuario No Registrado, Servidor Web
Descripción	Este caso de uso solo está disponible cuando el usuario no ha iniciado sesión, y se inicia cuando el usuario da clic sobre el botón de Registrarse. Puede registrarse indicando sus datos personales o utilizando sus cuentas de Facebook, Twitter o Google+. Una vez que el usuario se registró, debe validar su inscripción revisando el correo electrónico automático enviado.

Nombre	Ver Juegos
Actores	Usuario No Registrado
Descripción	El usuario lee las descripciones sobre los juegos publicados, podrá leer los comentarios y las calificaciones que se le dio. Se le indicará que para poder descargar los juegos o hacer un comentario deberá iniciar sesión.

Nombre	Ver Documentación
Actores	Usuario No Registrado
Descripción	Este caso de uso se dispara cuando el usuario entra a la sección de Documentación, en él, podrá leer toda la documentación pertinente del framework (casos de uso, documento de especificación, publicaciones, API de desarrollo, etc.).
Nombre	Iniciar Sesión
Actores	Usuario Registrado
Descripción	El usuario registrado podrá iniciar sesión utilizando su cuenta de correo electrónico y la contraseña que haya elegido. En caso de haberse registrado mediante una red social, deberá iniciar sesión volviendo a elegir su opción de red social.
Nombre	Registrar Juego
Actores	Usuario Registrado
Descripción	El usuario necesita registrar su juego para poder hacer uso del sistema de puntajes en línea, para llevar a cabo este caso de uso, es necesario que el usuario indique el nombre del juego (el sistema es sensitivo a Mayúsculas y Minúsculas) y que no se encuentre registrado por algún otro usuario.
Nombre	Publicar Juego
Actores	Usuario Registrado
Descripción	Para poder publicar su juego, el usuario debe escribir una reseña de su juego, incluyendo mínimo dos imágenes de su aplicación e indicando la edad mínima recomendada. Una vez publicado, el juego se pondrá en un estado BETA (indicado en el nombre del juego) hasta que un Administrador verifique no infringe ningún derecho de autor. En caso de que se infrinja alguna ley, será notificado al usuario y el juego será removido.
Nombre	Eliminar Juego
Actores	Usuario Registrado
Descripción	En este caso de uso, el usuario puede eliminar su juego seleccionando la opción correspondiente. El sitio le indicará que la eliminación de la aplicación

será definitiva y se perderán los puntajes (si está utilizando el sistema de puntajes en línea) quedando inservible ésta última opción.

Nombre	Ver puntajes
Actores	Usuario Registrado
Descripción	El usuario podrá revisar sus puntajes de todos los juegos que tenga descargados, en caso de que sea el desarrollador de uno de ellos, podrá revisar todos los puntajes de los usuarios y ordenarlos de la forma él crea conveniente para su análisis.

Nombre	Eliminar Puntajes
Actores	Usuario Registrado
Descripción	El caso de uso inicia cuando el usuario presiona la opción de Eliminar, estando en el apartado de Puntajes. La acción es irreversible y eliminará todos los puntajes de los usuarios.

Nombre	Iniciar Sesión
Actores	Administrador
Descripción	El Administrador podrá iniciar sesión solo usando el usuario y contraseña asignados. En caso de querer agregar un nuevo administrador, se debe ocupar la cuenta de súper-usuario.

Nombre	Ver estadísticas
Actores	Administrador
Descripción	El caso de uso se dispara cuando el usuario genera un reporte sobre el número de usuarios que visitan el sitio, el número de juegos publicados o descargados y las calificaciones de ciertos juegos.

Nombre	Revisar Aplicaciones
Actores	Administrador
Descripción	Cuando un juego es publicado y puesto en modo BETA, el administrador puede revisarlos en éste caso de uso. Una vez liberado el juego, no puede volver ser llevado a revisión. Si el juego no pasa la validación, entonces será removido del

sitio y se dará de baja del sistema de puntajes en línea (Si la aplicación lo ocupa).

Nombre Eliminar Usuario

Actores Administrador

Descripción El caso de uso se dispara cuando el administrador detecta que se está haciendo un mal uso de la cuenta o está infringiendo varias normas de ética o derechos de autor. El sistema eliminará al usuario y su cuenta no podrá ser reactivada o volverse a registrar con la misma cuenta. Las aplicaciones generadas y que hayan pasado la prueba de validación pasarán a control del súper-usuario.

Nombre Eliminar Juego

Actores Administrador

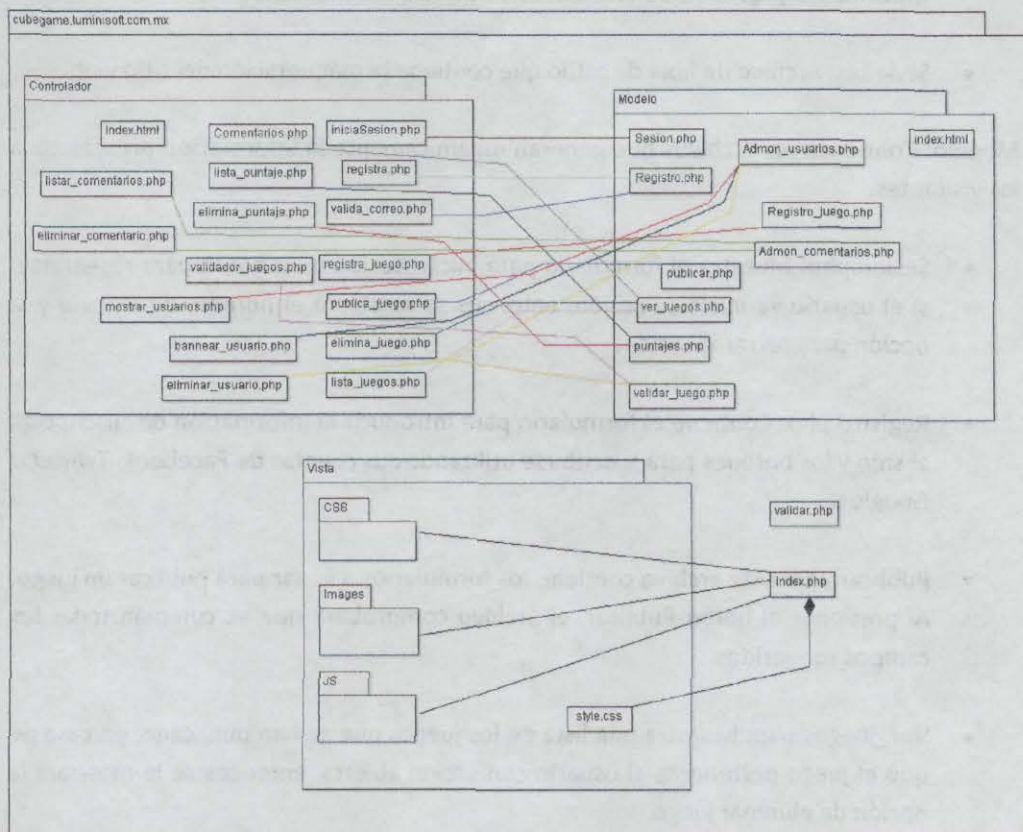
Descripción Este caso de uso puede ser incluido como continuación del caso de uso "Revisar Aplicaciones" o en el caso de uso "Bloquear Comentario", y consiste en la eliminación total del juego del sistema, enviando a los usuarios que hayan descargado la aplicación un correo indicando que el sitio no se hace responsable por la infracción de derechos de autor e invitándolos a eliminarlo de inmediato.

Nombre Bloquear Comentario

Actores Administrador

Descripción Cuando un comentario infringe alguna regla de ética o usa malas palabras, el administrador puede llevar a cabo éste caso de uso. El comentario es eliminado y al usuario que hizo el comentario se le marca una falta en el sistema, si el usuario supera dos faltas, entonces será eliminado del sistema y se envía un correo electrónico automático indicando las razones por las cuales se le dio de baja del sitio.

A.2.2 Vista Lógica



El sitio web se encuentra dentro de la carpeta raíz del subdominio, dentro se encuentran 3 paquetes cuyos archivos están divididos según la arquitectura de Modelo Vista Controlador (MVC). Por seguridad, en cada una de las carpetas, se genera un archivo Index que redirija al visitante al inicio de la página y así evitar conozca la maquetación de la página mediante un protocolo FTP.

- Index.php: Este archivo se encuentra fuera del modelo debido a su función dentro de un sitio web. Es el encargado de mostrar toda la información a los usuarios sin la necesidad de recargar la página.
- Validar.php: Este archivo es utilizado únicamente para la validación de usuarios, activando sus cuentas. El archivo recibe un parámetro cifrado en Base 64 para su seguridad.
- Vista: Contiene la interfaz gráfica del sistema, es decir, contiene la maquetación de la página web y espera que la Vista genere dinámicamente el contenido a mostrar.
 - CSS: Esta carpeta contiene todos las hojas de estilo a utilizar en el sitio web, indicando con los colores, formas y tamaños de cada división y letras.

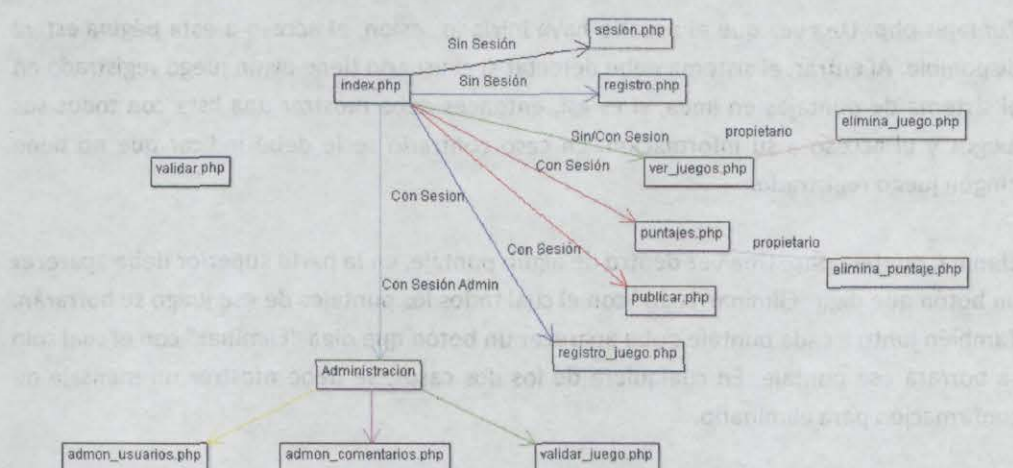
- Images: Contiene todas las imágenes que se utilizan en el sistema.
 - JS: Contiene los archivos JQuery necesarios para realizar la animación de movimiento y las API's de comunicación con las redes sociales.
 - Style.css: Archivo de hoja de estilo que contiene la maquetación del sitio web.
- **Modelo:** Contienen los archivos que generan dinámicamente la información presentada a los visitantes.
- Sesion.php: Muestra el formulario para iniciar sesión o un botón para registrarse, si el usuario ya inició su sesión, entonces se mostrará el nombre de usuario y la opción para cerrar la sesión.
 - Registro.php: Contiene el formulario para introducir la información de suscripción al sitio y los botones para suscribirse utilizando sus cuentas de Facebook, Twitter o Google+.
 - Publicar.php: Este archivo contiene los formularios a llenar para publicar un juego. Al presionar el botón Publicar, el archivo comprobará que se cumplan todas los campos requeridos.
 - Ver_juegos.php: Muestra una lista de los juegos que se han publicado, en caso de que el juego pertenezca al usuario con sesión abierta, entonces se le mostrará la opción de eliminar juego.
 - Puntajes.php: Muestra la información sobre los puntajes de un juego en particular. Para poder acceder a este archivo, se debe ser dueño del juego.
 - Validar_juego.php: En esta página se muestran todos los juegos que están en versión BETA, permitiendo al administrador liberar o eliminar el juego dependiendo de su contenido.
 - Admon_usuarios.php: El administrador de usuarios, permite al administrador eliminar o amonestar a usuarios que se encuentren incumpliendo las reglas éticas o de derechos de autor.
 - Registro_juego.php: Este archivo contiene el formulario para dar de alta un nuevo juego en el sistema de puntajes en línea. Se pide el nombre del juego y el sistema debe verificar si no se encuentra usado ese nombre.

- Admon_comentarios.php: El administrador de de comentarios muestra una lista con todos los juegos que se encuentran activos en el sistema, al seleccionar alguno, se muestran todos los comentarios de ese juego, y si el administrador encuentra alguno que no cumpla con las normas de ética, se elimina.
- Controlador: Esta carpeta contiene todos los archivos que realizan los cambios en la base de datos o en la información que es mostrada a los usuarios.
- iniciaSesion.php: Contiene la funcionalidad para buscar el usuario en la base de datos del sitio, y dependiendo de los permisos que tenga, serán las opciones que tendrá disponibles.
 - Registra.php: La información del formulario se envía a este archivo, el cual registra en la base de datos los datos personales del usuario. Una vez terminado, ejecuta el código del archivo "Valida_correo.php".
 - Valida_correo.php: Este archivo contiene la función de enviar al usuario la liga para activar su registro.
 - Registra_juego.php: Una vez que se envió la información del formulario, éste archivo lo recibe y lo registra en el sistema, para poder ser utilizado.
 - Publica_juego.php: El archivo inserta en la base de datos toda la información en el juego y lo pone en modo BETA, una vez terminado, muestra la información en el sitio.
 - Elimina_juego.php: Contiene el código para eliminar un juego (en estado BETA o RELEASE) del sistema.
 - Lista_juegos.php: Obtiene de la base de datos la lista de juegos con toda la información y genera un orden para ser mostrado en la Vista.
 - Comentarios.php: Obtiene de la base de datos todos los comentarios de un juego en específico. Se manda la información a la Vista llevando el formato visible.
 - Lista_puntaje.php: Genera la lista de puntajes con formato a partir de la información obtenida de la base de datos (como nombre del jugador, puntaje, día de inserción, etc).

- **Elimina_puntaje.php:** El archivo contiene la funcionalidad para eliminar un puntaje de un juego en específico. Una vez eliminado el puntaje del sistema, no puede ser recuperado.
- **Validador_juegos.php:** Esta función se ejecuta cuando ya se validó el juego, y cambia el estado del juego de BETA a RELEASE.
- **Mostrar_Usuarios.php:** Genera la información de la vista ya con formato a partir de la información obtenida de la base de datos. La información mostrada es: Nombre del Usuario, Nombre real, Numero de juegos publicados y Numero de amonestaciones.
- **Banear_usuarios.php:** El código del archivo pone una amonestación al usuario seleccionado, dependiendo del número de amonestaciones que tenga, será la acción que se lleva a cabo. Si es su primera amonestación, se le envía un correo electrónico indicándole que fue amonestado por alguna falla a la ética o por infringir derechos de autor; en caso de ser su segunda amonestación se ejecuta el código de "Eliminar_usuario.php" y se le manda un correo al usuario indicándole que fue eliminado del sistema por reincidir en alguna falta.
- **Eliminar_usuario.php:** Elimina un usuario del sistema, todos los juegos que haya publicado se vuelven propiedad del administrador del sistema. Una vez que fue eliminado el usuario, no puede volver a utilizar el mismo correo electrónico para registrarse.
- **Listar_comentarios.php:** Genera la vista con formato de los comentarios sobre un juego, obteniendo la información desde la Base de Datos.
- **Eliminar_comentario.php:** El comentario que se haya seleccionado es eliminado de la base de datos. Una vez terminada la acción, se ejecuta el código de "Banear_usuarios.php".

La iteración entre la Vista y el Controlador se llevará a cabo a través de Ajax, para que el manejo de la información sea transparente para el usuario final.

A.2.3 Vista de Desarrollo



En ésta vista se tiene el mapa de navegabilidad del sitio. La definición de cada uno de los componentes se lista:

- **Validar.php:** Este archivo no se encuentra conectado con ningún otro debido a que solo puede ser accedido mediante el correo electrónico enviado al usuario al registrarse.
- **Index.php:** Es la página principal, cuando se ingresa al sitio, debe cargar todas las demás paginas para mostrar la información a los usuarios. La página debe estar recargando la información sobre juegos nuevos, puntajes y comentarios cada 5 minutos (si se mantiene en inactividad).
- **Sesion.php:** Esta página podrá ser accedida desde el index solamente cuando no se tenga una sesión abierta. En ella se introduce el nombre de usuario o correo electrónico con el que se registró (en caso de haberse registrado usando una cuenta de red social deberá usar su correo electrónico) y la contraseña que asignó.
- **Registro.php:** Al igual que la página de inicio de sesión, el registro solo estará disponible mientras no haya una sesión abierta. El registro debe ser mostrado en Pop-ups (ventanas emergentes). La información debe ser revisada por el sistema en tiempo real e ir indicando al usuario si todos los campos obligatorios están llenos y si el usuario no existe en la Base de Datos.
- **Ver_juegos.php:** Esta página estará siempre visible para los usuarios (con o sin sesión iniciada). La única limitante que tiene el archivo es que solo los usuarios registrados pueden descargar y comentar los juegos.
- **Elimina_juego.php:** La opción para poder acceder a esta página aparecerá solo cuando el juego seleccionado es propiedad del usuario con sesión activa. Al presionar el botón, debe

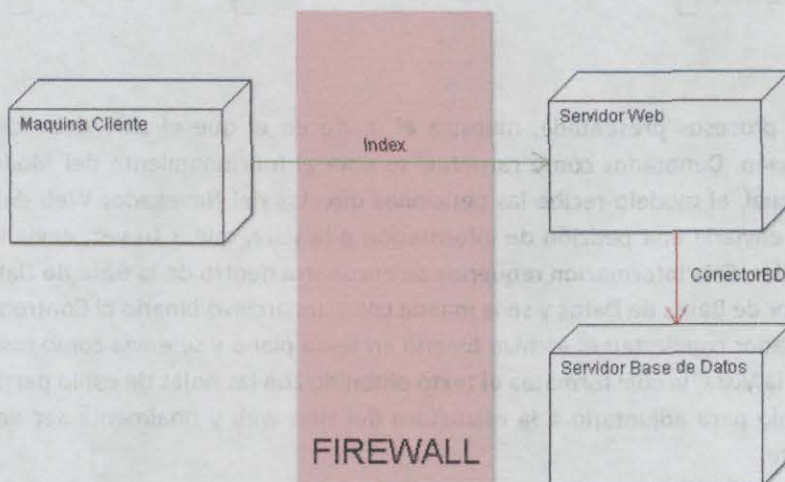
salir una ventana emergente indicando que una vez eliminado no podrá ser restaurado. Si el usuario acepta, entonces será regresado a la lista de juegos y el juego será eliminado junto con su registro al sistema de puntajes en línea (si lo tiene activo).

- **Puntajes.php:** Una vez que el usuario haya iniciado sesión, el acceso a esta página estará disponible. Al entrar, el sistema debe detectar si el usuario tiene algún juego registrado en el sistema de puntajes en línea, si es así, entonces debe mostrar una lista con todos sus juegos y el acceso a su información. En caso contrario se le debe indicar que no tiene ningún juego registrado.
- **Elimina_puntaje.php:** Una vez dentro de algún puntaje, en la parte superior debe aparecer un botón que diga “Eliminar Todo” con el cual todos los puntajes de ese juego se borrarán. También junto a cada puntaje debe aparecer un botón que diga “Eliminar” con el cual solo se borrará ese puntaje. En cualquiera de los dos casos, se debe mostrar un mensaje de confirmación para eliminarlo.
- **Publicar.php:** Al tener una sesión activa el usuario, ésta opción le permitirá publicar su juego en la página. Al aparecer los campos, el sistema debe ir verificando que todos los obligatorios tengan información y que el nombre del juego no esté ya tomado por alguien más. Cuando termine, el sistema deberá mandarlo a la página de su publicación.
- **Registro_juego.php:** Solo pueden acceder las personas registradas. En ella se muestra un campo que pide el nombre del juego a registrar en el sistema de puntajes. Mientras el usuario escribe, el sistema debe verificar si el nombre no se encuentra seleccionado por otro usuario. Al enviar la información, el sitio debe mostrar una ventana emergente indicando que el nombre que acaba de asignar debe ser usado en el framework cuando lo pida, cuando desaparezca, el usuario debe ser regresado a la página principal.
- **Administración:** Esta opción solo estaría disponible cuando se inicie sesión como Administrador. En ella se muestran tres opciones que serán detalladas más adelante, teniendo una distribución uniforme horizontal.
 - **Validar_juego.php:** En esta opción, deben aparecer todos aquellos juegos que se encuentren en una versión BETA. Cuando sea seleccionado, el sistema deberá mostrar la página del juego en una ventana emergente. Si el administrador considera que no se viola ninguna regla, entonces debe presionar el botón de “Aceptado”, en caso contrario debe presionar la opción “Rechazado”; una vez seleccionada una opción debe cerrar la ventana emergente y el juego seleccionado desaparecer de la lista.
 - **Admon_comentarios.php:** Al igual que en la opción de “Validar_juego”, el sistema debe mostrar una lista con todos los juegos que al menos tengan un comentario,

si el administrador cree que un comentario está fuera de lugar, debe presionar el botón de "Eliminar Comentario". Una vez terminada la eliminación, el sistema debe actualizar los comentarios.

- Admon_usuarios.php: La administración de usuarios es una herramienta de reporte solo para el Administrador. En él encontrará la información por usuario sobre su última conexión, el número de juegos publicados, el número de juegos registrados en el sistema de puntajes en línea, el número de comentarios que ha hecho y el número de amonestaciones que tiene cada uno. Esta función debe permitir al administrador ordenar la información como crea conveniente.

A.2.4 Vista Física

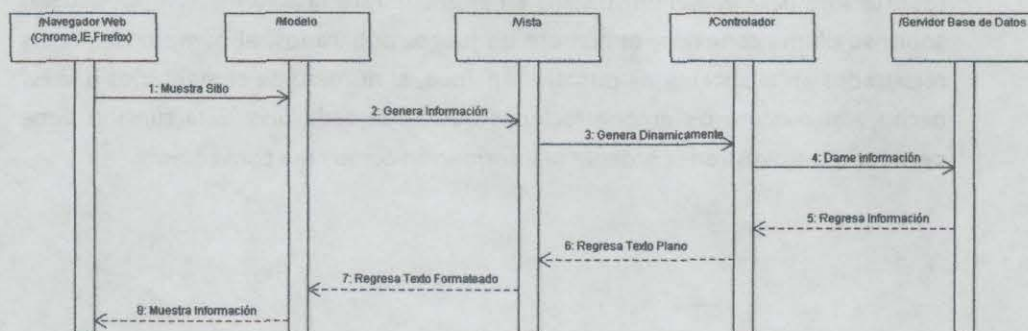


En ésta vista, se muestran los componentes del sitio web, dos de ellos se encuentran físicamente en un solo equipo (Servidor Web y Servidor de Base de Datos), el Firewall indicado es el otorgado por el distribuidor de hosting para evitar ataques al sitio. Se explica a continuación cada uno de los componentes de forma detallada:

- **Maquina Cliente:** Este equipo es la terminal del sitio, contiene el navegador web soportado (Firefox, Chrome, Internet Explorer) que muestra la información generada al usuario.
- **Servidor Web:** Este servidor está contenido dentro del servidor del hosting. Contiene todo el modelo MVC requerido en los requerimientos funcionales, permitiendo al sitio mantenerse activo en todo momento.

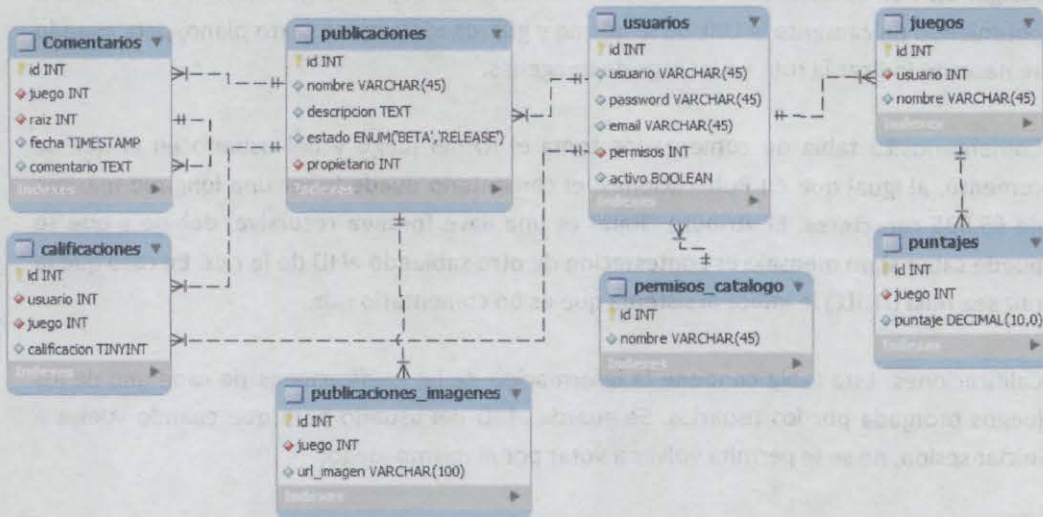
- **Servidor Base de Datos:** Este servidor se encuentra hospedado en el servidor del hosting, su función es contener la base de datos integra, manteniendo sus datos seguros. Se comunica con el servidor web por medio de un ODBC especializado para el servidor MySQL localizado en el archivo conectorBD.

A.2.5 Vista de Procesos



La vista de procesos presentada, muestra el modo en el que el sitio debe cargarse y generar su información. Denotados como carpetas, se nota el funcionamiento del Modelo Vista Controlador, en el cual, el modelo recibe las peticiones directas del Navegador Web del usuario final, para después enviarle una petición de información a la vista, que a su vez, envía la misma petición al controlador. Si la información requerida se encuentra dentro de la Base de Datos, hace la petición al servidor de Bases de Datos y se la manda como un archivo binario al Controlador. Los archivos del controlador convierten el archivo binario en texto plano y se envía como respuesta a la petición que hizo la Vista, la cual formatea el texto obtenido con las hojas de estilo pertinentes y es enviada al Modelo para adjuntarlo a la estructura del sitio web y finalmente ser enviado al navegador del cliente.

A.3 Diseño de la Base de Datos.



En el diseño de la Base de Datos, se puede observar el funcionamiento del sistema basándose en la información que contiene. La base de Datos se encuentra normalizada hasta 3ra Forma Normal (3FN). La información que guarda cada tabla son:

- **Usuarios:** En esta tabla se mantiene la información de los inicio de sesión de los usuarios. Mientras el valor del atributo Activo se mantenga en 0, el sistema no permitirá que inicie sesión, debido a que no ha confirmado su dirección de correo electrónico. La contraseña, que se guarda en el atributo Password, se almacenará en cifrada a 128 bits utilizando la función MD5 proporcionada por el Gestor de Base de Datos.
- **Permisos_catalogo:** Esta tabla funciona como un catalogo de permisos escalable, las opciones que el usuario podrá visualizar en el sitio cuando inicie sesión será gestionado desde el código del mismo.
- **Juegos:** Contiene todos los juegos registrados para el uso del Sistema de Puntajes En Línea, para que el sistema detecte cuando mostrar las opciones de visualizar los puntajes y eliminar algunos, se guarda junto con la información el ID del usuario que lo registró.
- **Puntajes:** Esta tabla es la encargada de guardar todos los puntajes de todos los juegos que se tengan registrados, junto con un nombre de jugador (máximo 45 caracteres) que serán enviados a los dispositivos cuando lo requieran.
- **Publicaciones:** Contiene la información sobre los juegos publicados. Es distinto a la tabla de Juegos debido a que un juego puede estar registrado en el sistema de puntajes en línea, pero el usuario no quiere publicarlo en el sitio. La descripción del juego puede tener un máximo de 65,535 caracteres; cuando se publica el juego, automáticamente la base de datos lo guarda en modo BETA.

- **Publicaciones_imagenes:** Al publicarse un juego, se piden al menos dos imágenes de la funcionalidad del mismo. Es por ello que esta tabla está preparada para recibir de 2 a n imágenes. Por optimización de la base de datos, la imagen es guardada en el servidor, obteniendo únicamente la URL de la misma y guardándola como texto plano, para cuando se necesite indicar la ruta en los tags de imágenes.
- **Comentarios:** La tabla de comentarios toma el ID del juego y del usuario en el que se comentó, al igual que en Publicaciones, el comentario puede tener una longitud máxima de 65,535 caracteres. El atributo "Raíz" es una llave foránea recursiva, debido a que se puede saber si un mensaje es contestación de otro sabiendo el ID de la raíz. En caso que la raíz sea nula (NULL) le indica al sistema que es un comentario raíz.
- **Calificaciones:** Esta tabla contiene la información de las calificaciones de cada uno de los juegos otorgada por los usuarios. Se guarda el ID del usuario para que cuando vuelva a iniciar sesión, no se le permita volver a votar por el mismo juego.

Se puede tomar de compañera a la fantasía,
Pero se debe tener como guía a la razón.
-- Samuel Johnson

Apéndice B. Estándar de Codificación C#

Para el desarrollo de la aplicación, se siguió los estándares de C#, esto se hizo para desarrollar aplicaciones rentables y mantenibles. Los patrones de nombres de variables, estándares de codificación y buenas prácticas es una compilación de la experiencia obtenida en el trabajo basándose en algunas guías de buenas prácticas de Microsoft y otras empresas.

Para el nombrado de las variables se utiliza el estilo Pascal, el cual consiste en escribir el primer carácter de cada una de las palabras que conforman el nombre con mayúsculas y las demás con minúsculas, y el estilo Camello, el cual consiste en que el primer carácter de cada palabra, excepto la primera palabra son con mayúsculas y las demás minúsculas.

B.1 Estándar de Escritura de Nombre de Métodos y Variables

1. Se debe usar la escritura Pascal para los nombres de las clases
2. Se debe usar la escritura Camello para el nombre de los métodos
3. Se utiliza la escritura Pascal para el nombre de las variables y los parámetros de los métodos.
4. Se debe usar el prefijo "I" seguido en estilo Pascal el nombre de la interfaz
5. El nombre de las variables debe ser con sentido y descriptivo al valor a guardar, evite utilizar abreviaciones.
6. No se deben utilizar caracteres simples como nombres de variables, a excepción de aquellas variables utilizadas para ciclos.
7. No se debe utilizar el guion bajo (_) para nombrar variables locales
8. Todas las variables de entorno, deben iniciar con guion bajo (_) para que puedan ser identificadas de otras variables locales
9. No utilice palabras clave como nombre de variables
10. Cuando el tipo de dato es Boolean, debe utilizarse el prefijo "is" antes de escribir el nombre de la variable
11. Se debe emplear un prefijo apropiado para cada control de interfaz que pueda ser identificado de las demás variables. La siguiente lista indica los controles de usuario más comunes y el prefijo a utilizar:

Control	Prefijo
Label	Lbl
TextBox	Txt

DataGrid	Dtg
Button	Btn
ImageButton	Imb
Hyperlink	Hlk
DropDown List	Ddl
ListBox	Lst
DataList	Dtl
Repeater	Rep
CheckBox	Chk
CheckBoxList	Cbl
RadioButton	Rdo
RadioButtonList	Rbl
Image	Img
Panel	Pnl
Placeholder	Phd
Table	Tbl
Validators	Val

12. Se debe utilizar el estilo Pascal para el nombre de los archivos.

B.2 Identación y Espacios

1. Se debe utilizar el tabulador para la Identación, nunca se debe utilizar espaciado. El tamaño del tabulador debe ser 4
2. Los comentarios deben encontrarse al mismo nivel que el código, es decir, debe llevar la Identación al mismo nivel que el código que comenta.
3. Las llaves deben encontrarse al mismo nivel donde se abre y se cierran, la llave que abre el bloque debe encontrarse al mismo nivel, sobre una nueva línea de la condición, método o clase.
4. Se debe ocupar una línea en blanco para separar el código en operaciones lógicas.
5. Debe existir solo un espacio en blanco entre cada uno de los métodos dentro de las clases
6. Usar un espacio en blanco antes y después de cada operador o paréntesis
7. Utilizar las regiones (#region) para mantener la funcionalidad de código junto, las regiones a utilizar son:

- a. Variables
 - b. Constructor
 - c. Events
 - d. Methods
 - e. Properties
8. Mantener las variables, propiedades y métodos privados antes que los miembros públicos.

B.3 Buenas Prácticas de Programación

1. Para evitar que los métodos escritos sean largos. Un método debe tener de 1 a 25 líneas de código, si un código contiene más de este límite de líneas, se debe considerar realizar una refactorización para separar en más métodos.
2. El nombre de los métodos deben indicar la acción que llevan a cabo. Si el nombre del método es obvio, no es necesario documentar sobre la acción que se lleva a cabo.
3. Los métodos deben realizar solo una acción. No se debe combinarse más de una acción en un solo método, aunque las acciones sean pequeñas.
4. Se deben utilizar los tipos de datos específicos para C#, en lugar de los tipos de datos definidos en el espacio de trabajo System.
5. Se deben esperar en todo momento valores inesperados. No se debe asumir que siempre se obtendrán los valores deseados.
6. Para el uso de números como opciones o estados, se deben indicar constantes. La declaración de las constantes se encuentran en la parte superior del archivo y deben utilizarse incluso dentro de la misma clase.
7. Convertir los tipos de datos string a minúsculas o mayúsculas antes de comparar para asegurar que la cadena de caracteres coincidirá incluso cuando son escritos de forma distinta.
8. Usar la propiedad String.Empty en lugar de una cadena vacía
9. Utilizar variables tipo ENUM cuando se requiera, no se deben indicar valores numéricos o cadenas de caracteres como valores discretos.
10. Las variables de clase no deben ser públicas ni protegidas, siempre deben ser privadas y permitir su acceso por medio de propiedades públicas o privadas
11. Los métodos controladores de eventos no deben contener código, sino más bien, deben llamar a otros métodos.
12. No indicar las rutas de un archivo o la unidad de disco a nivel de código. La aplicación debe obtener la dirección o utiliza direcciones relativas.
13. En el inicio de la aplicación, se debe hacer un chequeo y asegurarse que todos los archivos y dependencias necesarios para el sistema se encuentran disponibles.
14. Muestre al usuario mensajes cortos y cordiales, pero si es necesario, escriba en un log de sistema el error actual y toda la posible información que pueda ayudar a diagnosticar el problema más adelante.
15. No se debe tener más de una clase por archivo.

16. Evite pasar muchos parámetros a un método, si se tienen más de 5 parámetros, es candidato a ser definido en una clase o estructura.
17. Si el retorno de un método es una colección, regrese una colección vacía en lugar de nula, si no se tiene información que regresar.
18. Organice lógicamente todos los archivos del proyecto en folders apropiados, se puede tener un máximo de 10 folders a nivel de root y 5 sub folder en cada uno de ellos.
19. Si en el sistema se utiliza una conexión a base de datos, sockets, fileStreams, etc. Siempre se deben cerrar en el bloque Finally. Esto asegura que si una excepción ocurre después de abrir la conexión, será cerrado al finalizar el bloque finally.
20. Declare las variables tan cerca de donde es usada por primera vez. Use la declaración de variable por línea.

B.4 Comentarios

Los comentarios buenos y con un significado concreto hacen que el código sea más mantenible, sin embargo:

1. No escriba un comentario por cada línea de código o cada variable declarada
2. Use comentarios de una sola línea (`//` o `///`) para los comentarios, evite utilizar los comentarios multilinea (`/* */`)
3. Escriba comentarios cuando sean requeridos. Pero un buen código leíble requiere menos comentarios. Si todos los nombres de variables y métodos son significativos, eso hace el código más entendible.
4. No escriba comentarios si el código es fácil de entender sin comentarios. Si se dejan muchos comentarios y se cambia el código sin cambiar el comentario, esto puede causar confusión.
5. Pocas líneas de comentarios hace el código más elegante. Pero si el código no es limpio o legible y hay pocos comentarios, lo hace peor
6. Si es necesario utilizar una lógica compleja o rara por cualquier razón, se debe documentar con suficientes comentarios.
7. Si se inicializa una variable con un número especial o cadena de caracteres, se debe documentar la razón por la que se eligió ese valor.
8. Verifique faltas de ortografía en los comentarios y asegúrese que la gramática y puntuación es usada.

B.5 Manejo de Excepciones

1. No se debe hacer obtener una excepción y no hacer nada, si se esconden las excepciones no se conocerá si la excepción ocurre o no. En cualquier caso que el desarrollador no quiera mostrar el error al usuario, puede guardarlo en el log del sistema y continuar ejecutando.

2. En caso de una excepción se le debe dar un mensaje amigable al usuario, pero guardar en el log el error actual con toda la información posible detallada, incluyendo la hora a la que ocurrió, el método, la clase, etc.
3. Ocupa en la cláusula catch excepciones específicas, no excepciones genéricas.
4. Cuando se arroja una excepción, se debe usar la sentencia throw sin especificar la excepción original. De esta forma, el error original prevalecerá.
5. No escriba bloques try-catch muy largos. Si es requerido, escriba bloques try-catch separados por cada tarea que se realice.

EJEMPLOS

Nombre de Clase	public partial class ClassName
Nombre de Métodos	Public void method Name(Object paramName)
Nombre de Variables Locales	Type variableName;
Nombre de Variables de Entorno	Private Type _variableName:
Declaración de variables Boolean	Private Bool _isOnline;
Declaración de regiones	#regionregionName #endregion
Generación de Comentarios	// Esto es un comentario.
Bloques Try/Catch/Finally	Try {...} Catch { Throw }

No hay hombre de nación alguna que,
Habiendo tomado a la naturaleza por guía,
No pueda llegar a la verdad.

— Marco Tulio Cicerón

Apéndice C. Estándar de Codificación Java

Este modelo de programación está basado en los estándares recomendados por Sun Microsystems, que han sido difundidos y aceptados ampliamente por toda la comunidad de desarrolladores Java, y que han terminado por consolidarse como un modelo estándar de programación de facto.

Estas normas son muy útiles por muchas razones, entre las que destacan:

- Facilitan el mantenimiento de una aplicación.
- Permite que cualquier programador entienda y pueda mantener la aplicación

Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su comprensión rápida.

C.1 Organización de Archivos

1. Las clases se agrupan en paquetes. Estos paquetes se deben organizar de manera jerárquica, de forma que todo el código desarrollado se encuentre dentro de los paquetes.
2. Un fichero consta de secciones que deben estar separados por líneas en blanco y comentarios opcionales que identifiquen cada sección.
3. Deben evitarse que los archivos de gran tamaño, es decir que tengan más de 1000 líneas de código.
4. Cada fichero Java debe contener una única clase o interfaz pública. El nombre del fichero debe coincidir con el nombre de la clase.
5. Todo fichero debe de tener las siguientes secciones:
 - a. Comentarios de inicio: Todo fichero fuente debe comenzar con un comentario que incluya el nombre de la clase, información sobre la versión del código, la fecha y nombre del desarrollador.
 - b. Sentencia de Paquete: La primera línea no comentada de un fichero debe ser la sentencia de paquete, que indica el paquete al que pertenece la clase incluida.
 - c. Sentencia de importación: Tras la declaración del paquete, se incluirán las sentencias de importación de los paquetes necesarios. Estas importaciones seguirán el siguiente orden:
 - i. Paquetes del JDK de Java.
 - ii. Paquetes de utilidades no pertenecientes al JDK de Java.
 - iii. Paquetes desarrollados.

iv. Paquetes de la Aplicación.

Se recomienda minimizar en la medida de lo posible el uso de importaciones del tipo "package.*", pues dificultan la comprensión de las dependencias existentes entre las clases utilizadas por la aplicación.

- d. Declaración de clases e interfaces: La siguiente tabla describe los elementos que componen la declaración de una clase o interfaz, así como el orden en el que deben estar situados:

Elemento de declaración de una clase o interfaz	Descripción
Comentario de documentación	Permite describir la clase o interfaz desarrollada. Necesario para generar la documentación de la api mediante Javadoc.
Sentencia class o interface	
Comentario de implementación de la clase o interfaz (opcional)	Este comentario incluye cualquier información que no pueda incluirse en el comentario de documentación de la clase o interfaz.
Variables de estado Variables de instancia	En primer lugar se ponen las variables de clase pública, después las protegidas, posteriormente las de nivel de paquete sin modificador y por ultimo las variables privadas.
Constructores	
Métodos	Deben agruparse por funcionalidad en lugar de agruparse por ámbito o accesibilidad. El objetivo es desarrollar código fácil de leer y comprender.

C.2 Identación

1. Como norma general se establecen 4 caracteres como unidad de tabulación. La mayoría de los entornos de desarrollo más populares, incluyen facilidades para formatear código.

C.3 Longitud y División de Líneas

1. La longitud de línea no debe superar los 80 caracteres por motivos de visualización e impresión.
2. Cuando una expresión ocupe más de una línea, esta se podrá romper o dividir dependiendo de los siguientes criterios:
 - a. Tras una coma.

- b. Antes de un operador.
- 3. Cuando se divida una línea, la nueva línea debe estar alineada con el inicio de la expresión al mismo nivel que la línea anterior.

C.4 Declaraciones

1. Se recomienda el uso de una declaración por línea, promoviendo así el uso de comentarios.
2. Toda variable local tendrá que ser inicializada en el momento de su declaración, salvo que su valor inicial dependa de algún valor que tenga que ser calculado previamente.
3. Las variables deben situarse al principio de cada bloque principal en el que se utilicen y nunca en el momento de su uso, exceptuando las variables de ciclos.
4. Se debe evitar el uso de declaraciones que oculten a otras declaraciones de ámbito superior.

C.5 Declaración de Clases e Interfaces

1. No se debe incluir ningún espacio entre el nombre del método y el paréntesis inicial del listado de parámetros.
2. El carácter de inicio de bloque debe aparecer al final de la línea que contiene la sentencia de declaración.
3. El carácter de final de bloque se sitúa en una nueva línea tabulada al mismo nivel que su correspondiente sentencia de inicio de bloque, excepto cuando la sentencia sea nula.
4. Los métodos se separarán entre sí mediante una línea en blanco.

C.6 Sentencias

1. Cada línea debe contener como máximo una sentencia.
2. Las sentencias pertenecientes a un bloque de código estarán tabuladas un nivel más a la derecha con respecto a la sentencia que las contiene.
3. El carácter inicio de bloque debe situarse al final de la línea que inicia el bloque. El carácter de final de bloque debe situarse en una nueva línea tras la última línea del bloque y alinearla con respecto al primer carácter de dicho bloque.
4. Todas las sentencias de un bloque deben encerrarse entre llaves, aunque el bloque conste de una sola sentencia.
5. En una sentencia "Try/Catch" la cláusula Catch debe iniciar después del cierre del bloque Try.
6. En el bloque "catch" siempre se imprimirá una ruta de error, indicando el tipo de excepción generada y posteriormente se elevará dicha excepción al código invocante, salvo que la lógica de ejecución de la aplicación no lo requiera.

7. Siempre se utilizará el bloque "finally" para liberar recursos y para imprimir rutas de monitorización de fin de ejecución.

C.7 Espacios en Blanco

1. Se utiliza un espacio en blanco entre una palabra clave y un paréntesis. Esto permite que se distingan las llamadas a métodos de las palabras claves.
2. Se debe dejar un espacio en blanco después de cada coma en un listado de argumentos.
3. Para separar un operador binario de sus operandos, excepto en el caso de puntos decimal.
4. Para cuando se realiza un casteo de un tipo de dato.

C.8 Nomenclatura de Identificadores

1. Los paquetes se escribirán con letras minúsculas para evitar que entren en conflicto con los nombres de clases o interfaces. El prefijo del paquete siempre corresponderá a un nombre de dominio de primer nivel.
2. El resto de los componentes del paquete se nombrarán de acuerdo a las normas internas de la organización.
3. Los nombres de clases deben ser sustantivos y deben tener la primera letra en mayúsculas. Si el nombre es compuesto, cada palabra componente deberá comenzar con mayúsculas.
4. Los nombres de las clases deben ser simples y descriptivos. Debe evitarse el uso de acrónimos o abreviaturas, salvo en aquellos casos en los que dicha abreviatura sea más utilizada que la palabra que representa.
5. Las interfaces se nombrarán siguiendo los mismos criterios que los indicados para las clases. Como norma general, toda interfaz se nombrará con el prefijo "I" para diferenciarla de las clases que la implementan.
6. Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras, cada una de ellas tendrá la primera letra en mayúsculas.
7. Las variables se escribirán siempre en minúsculas. Las variables compuestas tendrán la primera letra de cada componente en mayúsculas.
8. Las variables nunca podrán comenzar con el carácter "_" o "\$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código.
9. Debe evitarse el uso de nombres de variables con un solo carácter excepto para variables temporales.
10. Todos los nombres de constantes tendrán que escribirse en mayúsculas. Cuando los nombres de constantes sean compuestos, las palabras se separarán entre sí mediante el carácter de guion bajo "_".

C.9 Prácticas de Programación

1. Los tributos de instancia y clase serán siempre privados, excepto cuando tengan que ser visibles en subclases heredadas, en tales casos serán declarados como protegidos.
2. El acceso a los atributos de una clase se realizarán por medio de los métodos get y set correspondientes, incluso cuando el acceso a dichos atributos se realicen en los métodos miembros de la clase.
3. Evitar el uso de objetos para acceder a los miembros estáticos de una clase. Se utilizará en su lugar el nombre de la clase.
4. Los valores de las variables constantes nunca aparecerán directamente en el código. Para designar dichos valores se utilizarán constantes escritas en mayúsculas y se declararán, según su ámbito de uso, o bien en una clase de constantes creada para tal efecto, o bien en la clase donde sean utilizadas.
5. Se deben evitar las asignaciones de un mismo valor sobre múltiples variables en una misma sentencia, ya que dichas sentencias suelen ser difíciles de leer.
6. No utilizar el operador de asignación en aquellos lugares donde sea susceptible de confusión con el operador de igualdad.
7. No utilizar asignaciones de variable embebidas o anidadas.
8. Se debe usar paréntesis en expresiones que incluyan distintos tipos de operadores para evitar problemas de precedencia de operadores.
9. Los valores de retorno tendrán que ser simples y comprensibles, de acuerdo al propósito y comportamiento del objeto en el que se utilicen.
10. Toda expresión compuesta por uno o más operadores binarios, situada en la parte condicional del operador ternario deberá ir entre paréntesis.

C.10 Documentación

1. Se debe incluir en la entrega de la aplicación la documentación de los archivos fuente de todas las clases. Dicha documentación será generada por la herramienta "javadoc".
2. Para generar el Javadoc se deberán agregar las siguientes etiquetas:
 - a. **@author:** Añade información sobre el autor o autores del código.
 - b. **@version:** Puede incluir información sobre la versión y fecha del código.
 - c. **@param:** Inserta el parámetro especificado y su descripción en la sección de "Parameters" de la documentación del método en el se incluya. Estas etiquetas deben aparecer en el mismo orden en el que aparezcan los parámetros especificados del método.
 - d. **@return:** Inserta la descripción indicada en la sección "returns" de la documentación del método.
 - e. **@throws:** Añade el bloque de comentarios "Throws" incluyendo el nombre y la descripción de la excepción especificada.
 - f. **@see:** Permite incluir en la documentación la sección de comentario "See Also", conteniendo la referencia indicada.

- g. **@deprecated**: Esta etiqueta indica que la clase, interfaz, método o campo está obsoleto y que no debe utilizarse, y que dicho elemento posiblemente desaparecerá en futuras versiones.
- h. **@since**: Se utiliza para especificar cuándo se ha añadido a la API la clase, interfaz, método o campo. Debería incluirse el número de versión u otro tipo de información.

La administración efectiva consiste en empezar por lo primero
 Mientras que el liderazgo decide que es lo "primero",
 La administración le va asignando el primer lugar día tras día.
 La administración es disciplina, puesta en práctica.
 -- Stephen Covey

Apéndice D. Administración del Proyecto

D.1 User Stories

No.	User Storie	Tareas	Pruebas	Tipo
1	Como desarrollado quiero utilizar herramientas gráficas para reducir el tiempo de desarrollo.	➤ [J2ME-0] Descarga, instalación y configuración del ambiente de trabajo para dispositivos J2ME (Symbian). ➤ [A2.1-0] Descarga, instalación y configuración del ambiente de trabajo para dispositivos Android 2.1 Eclair ➤ [c#-0] Descarga, instalación y configuración de Visual Studio 2012 para desarrollar el framework ➤ [PHP-0] Descarga, instalación y configuración del servidor apache y gestor de base de datos MySQL.	N/A	Técnica
2	Como desarrollador quiero tener las clases organizadas por funcionalidad para que se pueda dar mantenimiento a la librería	➤ [J2ME-1] Generar los paquetes interfaces, library e util dentro del paquete com.upaep.thesis ➤ [A2.1-1] Crear los paquetes framework, classes y library dentro del paquete com.upaep.thesis ➤ [c#-2] Generar las carpetas Clases, Controls y Forms	N/A	Técnica
3	Como usuario quiero que el juego ocupe toda la pantalla para que aproveche la capacidad de mi teléfono.	➤ [J2ME-2] Generar una clase para el control de las dimensiones y resolución de pantalla llamado "Limit". ➤ [J2ME-23] Generar barras de desplazamiento cuando el texto sea mayor al tamaño de la ventana. ➤ [J2ME-24] Permitir que el texto se ajuste al ancho de la ventana sin cortar palabras. ➤ [A2.1-2] Cambiar la configuración global del manifest para aceptar Internet y Control de Pantalla	Comprobar en pantallas de diferentes tamaños y resoluciones que se haga el escalamiento	Cliente

4	Como desarrollador quiero optimizar los procesos de la librería para que sea más fluida al cliente	➤ [J2ME-3] Generar la clase "Background" como patrón de diseño para pantallas ➤ [A2.1-3] Crear interfaces para el control de los dispositivos periféricos (teclado, touch, etc.) ➤ [A2.1-5] Crear una interfaz para el control de gráficos en pantalla, permitiendo generar imágenes y dibujar figuras geométricas ➤ [A2.1-8] Generar una interfaz que sirva como patrón de diseño para todas las pantallas que se muestran al usuario ➤ [A2.1-31] Generar una clase que permita actualizar el estado de la aplicación ➤ [A2.1-32] Permitir que la clase utilice doble buffer para actualizar los gráficos ➤ [A2.1-33] Generar una clase que permita ejecutar cualquier pantalla de juego	N/A	Técnica
5	Como usuario, quiero que los juegos no consuman mucha pila para aumentar el tiempo de vida de mi teléfono	➤ [J2ME-4] Generar un "Garbage Collector" llamado "Destroyer" para eliminar y ahorrar pila del dispositivo. ➤ [A2.1-7] Generar una clase que funcione como contenedor y generador de objetos siguiendo el patrón Singleton	Hacer funcionar un juego con el recolector y otro sin el recolector y revisar el tiempo que dura la batería del dispositivo.	Cliente
6	Como usuario, quiero poder mantener mis puntajes en línea	➤ [J2ME-5] Generar socket de conexión HTTP para el Web Service ➤ [A2.1-34] Generar un conector HTTP ➤ [PHP-1] Análisis de funcionalidad para sistema de puntajes en línea ➤ [PHP-2] Diseño de la base de datos en MySQL Workbench ➤ [PHP-3] Diseño del Web Service para PHP ➤ [PHP-4] Implementación del diseño de base de datos en MySQL 5.5 ➤ [PHP-5] Implementación de la base de datos en PHPMyAdmin del servidor ➤ [PHP-6] Desarrollo del conector a la base de datos MySQL desde PHP ➤ [PHP-7] Generar un método para obtener los nombres y puntajes	Sobre emuladores y dispositivos que cuenten con conexión a internet se debe acceder a los puntajes que están en el sitio.	Cliente

		de usuarios ➤ [PHP-8] Generar un método para agregar los puntajes de un usuario ➤ [PHP-9] Generar un método para agregar nuevos usuarios al sistema ➤ [PHP-10] Generar un método para obtener el estado del servicio ➤ [PHP-11] Generar variables ocultas para los tipos de métodos ➤ [PHP-12] Generar el Web Service para acceder a los métodos		
7	Como desarrollador quiero tener un compendio de fuentes, para poder emplearlas en cualquier parte de la librería	➤ [J2ME-6] Generar una lista de fuentes utilizables en todas las clases	N/A	Técnica
8	Como usuario, quiero poder utilizar todas las funciones del teléfono para poder hacer los juegos más entretenidos.	➤ [J2ME-9] Generar una clase para controlar el envío de mensajes de texto ➤ [J2ME-10] Permitir que se pueda guardar datos en la memoria persistente del dispositivo ➤ [J2ME-12] Controlador de Base de Datos temporal para dispositivos que lo soporten ➤ [A2.1-10] Generar una interfaz para abrir archivos localizados en el paquete o en la memoria interna o externa del dispositivo ➤ [A2.1-11] Generar una clase para el control del acelerómetro ➤ [A2.1-12] Permitir que el acelerómetro se pueda calibrar y configurar para videojuegos ➤ [A2.1-13] Permitir que el usuario elija el acelerómetro que se va a utilizar cuando se tienen más de uno en el dispositivo ➤ [A2.1-14] Generar una clase que permita la lectura y escritura de archivos en texto plano ➤ [A2.1-15] Permitir que la clase de lectura y escritura busque en el dispositivo las memorias de almacenamiento interno y externo conectadas ➤ [A2.1-21] Crear un controlador del teclado del dispositivo detectando el número de teclas con los que cuenta ➤ [A2.1-22] Permitir que el controlador de teclado devuelva una lista con todas las teclas y el	Generar un juego que utilice diferentes componentes y reaccione correctamente.	Cliente

		- status de cada una de ellas ➤ [A2.1-25] Generar una clase que permita detectar las características de pantalla touch ➤ [A2.1-26] La clase debe permitir guardar los movimientos realizados en la pantalla touch ➤ [A2.1-27] La clase permitir hacer multitouch a la pantalla si lo permite el dispositivo ➤ [A2.1-28] Generar una clase que permita configurar automáticamente las aptitudes del dispositivo ➤ [A2.1-29] La clase debe configurar la pantalla touch a partir de la versión de Android		
9	Como usuario quiero poder poner canciones en mis juegos para que cuando lo jueguen no sea aburrido	➤ [J2ME-7] Clase para manejar sonidos del sistema operativo y canciones en formato MP3 y WAV ➤ [A2.1-4] Crear interfaces para reproducir sonidos de sistema y archivos en formato MP3 ➤ [A2.1-9] Generar una interfaz que permita al usuario abrir imágenes con formato PNG y JPG ➤ [A2.1-16] Generar una clase basándose en la interfaz de sonido que permita localizar y reproducir un archivo de audio ➤ [A2.1-17] Permitir que el usuario pueda configurar el volumen inicial del archivo de audio ➤ [A2.1-18] Permitir que el usuario cambie el volumen del archivo en tiempo de ejecución ➤ [A2.1-19] Permitir al usuario pausar y continuar con la reproducción del sonido en tiempo real ➤ [A2.1-20] Crear una clase que permita reproducir sonidos predefinidos en el sistema operativo basados en la interfaz creada ➤ [A2.1-30] Crear una clase que englobe la funcionalidad de reproducir audio desde archivo o desde sistema tomando en cuenta las características del dispositivo	Cargar archivos de música en diferentes formatos y generar sonidos MIDI con bytes, reproducirlos en cualquier dispositivo.	Cliente
10	Como usuario, quiero que los gráficos sean buenos para que se vean los juegos como si lo hubiera	➤ [J2ME-8] Hacer una alerta independiente que se pueda quitar al presionar un botón o al	Hacer revisión de la calidad de dibujo con comparación de otros motores gráficos	Cliente

	hecho personas profesionales.	terminarse un tiempo ➤ [J2ME-11] Crear una clase que permite generar efectos de profundidad en texto y bordes de las alertas ➤ [A2.1-6] Permitir que el usuario elija la calidad de los gráficos a mostrar en la pantalla (ARGB8888, ARGB4444 o RGB565) ➤ [A2.1-23] Generar una clase que permita configurar la calidad de las imágenes cargadas al sistema ➤ [A2.1-24] Permitir que se obtenga el ancho, largo y calidad de una imagen cargada	2D.	
11	Como usuario, quiero que mis juegos puedan tener una pantalla de inicio, para mostrar el nombre del juego y quien lo desarrolló	➤ [J2ME-17] Desarrollo de una pantalla de inicio, donde se permita poner el logo del juego y el desarrollador.	Al generar un juego debe mostrar esta pantalla antes de que muestre el juego.	Cliente
12	Como usuario, quiero que mis juegos tengan un menú para que el jugador pueda seleccionar diferentes opciones.	➤ [J2ME-13] Generar un patrón de opciones para cambiar configuración, cambiar de pantalla y salir. ➤ [J2ME-14] Crear clase para menú, debe permitir agregar un título, y las opciones se deben adaptar a la pantalla. ➤ [J2ME-15] Permitir cambiar el color de fondo y del texto de la opción seleccionada y no seleccionada ➤ [J2ME-16] Permitir poner imágenes como fondo en la pantalla Menú ➤ [J2ME-18] Permitir que al navegar en las opciones del menú pueda reproducirse un sonido del sistema operativo o de archivo	Mostrar un menú que cambie de pantalla por cada opción que seleccione.	Cliente
13	Como usuario quiero poder hacer una pantalla de instrucciones para decirle al jugador como se juega.	➤ [J2ME-19] Crear una clase que permita mostrar texto plano en la pantalla del dispositivo ➤ [J2ME-20] Permitir que se pueda agregar solo una opción a la pantalla de instrucciones ➤ [J2ME-21] Permitir cambiar el color de fondo y del texto mostrado. ➤ [J2ME-22] Permitir poner el texto de la pantalla instrucciones a partir de un archivo de texto plano	El usuario puede mostrar esta pantalla con cualquier texto desde archivo o predefinido respetando el tamaño de la pantalla	Cliente
14	Como desarrollador, quiero que cualquier equipo con Windows pueda hacer funcionar el	➤ [c#-1] Generar el proyecto bajo el framework 4.0	N/A	Técnica

	framework para que no se limite solo a computadoras recientes.			
15	Como usuario, quiero que el programa utilice toda la pantalla de la computadora para que lo utilicen en computadoras de escritorio o laptops	➤ [c#-3]Hacer que la ventana principal inicie maximizada ➤ [c#-7]Permitir que los controles se readapten al tamaño de la ventana	La ventana debe maximizarse y restaurarse en pantallas de diferentes resoluciones	Cliente
16	Como usuario quiero que el programa se pueda identificar por áreas independientes para facilitar su uso al cliente.	➤ [c#-4] Insertar un árbol de opciones en el lado izquierdo de la ventana ➤ [c#-5] Insertar dos controles de vista en el lado derecho para las funciones agregadas y los errores ➤ [c#-6] Insertar un panel en la parte central que permita posicionar los cubos	Los componentes deben actuar independiente uno de otro y mantener su posición en la pantalla.	Cliente
17	Como usuario quiero que la ventana principal tenga accesos rápidos para facilitar el uso de la herramienta al usuario.	➤ [c#-8] Agregar una barra de menús con las opciones Inicio, Proyecto, Cubos y Ayuda ➤ [c#-9]En el menú Inicio agregar las opciones Nuevo, Abrir, Guardar, Exportar como Imagen y Salir. ➤ [c#-10]En el menú Proyecto agregar las opciones Verificar y Compilar ➤ [c#-11]En el menú Cubos agregar las opciones Crear cubo, Modificar cubo y Eliminar Cubo ➤ [c#-12]En el menú Ayuda agregar las opciones Ayuda, Documentación y Acerca De. ➤ [c#-13] Agregar una barra de accesos rápidos con las funciones Nuevo Proyecto, Abrir, Guardar, Compilar, Nuevo Cubo, Modificar Cubo, Eliminar Cubo y Ayuda ➤ [c#-51] Se deben agregar iconos a cada una de las funciones	Las barras de acceso deben mostrar ayudas de la función que hacen y estar siempre visibles en la pantalla principal.	Cliente
18	Como usuario quiero que las funciones que contiene el sistema los lea de unos archivos para que pueda actualizar las funciones en cualquier momento.	➤ [c#-14] Generar un patrón de diseño para las funciones que agregará el framework ➤ [c#-21] Escribir funciones para el framework de prueba en formato DLL ➤ [c#-22] Desarrollar algoritmo para análisis y agrupación de funcionalidades según su tipo ➤ [c#-30] Generar tipos de datos para los atributos de las funciones ➤ [c#-34] Se deberá catalogar cada atributo con los valores 1 para crítico y 0 para no crítico	Crear archivos de prueba y revisar que se haga la catalogación de las funciones correctamente, tomando en cuenta el sistema operativo seleccionado	Cliente

19	Como usuario, quiero que ningún proyecto se repita para que no se sobrescriban	➤ [c#-15] Generar una clase que permita leer un archivo que se encuentre en una ruta específica ➤ [c#-16] Permitir que la clase lectora guarde las líneas del archivo en un vector ➤ [c#-17] Generar una clase que permita crear la carpeta workspace en la carpeta de Mis Documentos ➤ [c#-18] Crear una clase que permita obtener el nombre de todos los proyectos creados en la carpeta workspace ➤ [c#-19] Crear un método para buscar el nombre de un proyecto en el directorio workspace	Generar proyectos con diferentes nombres y estar al pendiente de la verificación de nombres. Revisar que se inhabilite el botón de Aceptar al repetirse un nombre	Cliente
20	Como usuario, quiero que pueda guardar proyectos existentes para que pueda seguir trabajándolos más tarde.	➤ [c#-20] Crear una clase que permita generar archivos en el disco duro ➤ [c#-23] Generar método que permita guardar la información de un proyecto en disco ➤ [c#-49] El sistema deberá guardar los cubos empleados y la posición en pantalla que ocupa	Guardar y revisar la integridad del archivo generado. El archivo debe tener la posición y configuración de todos los cubos agregados.	Cliente
21	Como usuario, quiero que el sistema permita abrir proyectos para terminar un trabajo inconcluso.	➤ [c#-24] Permitir que el sistema revise los archivos de un proyecto y el tipo de sistema operativo empleado ➤ [c#-25] El sistema debe permitir abrir proyectos ya generados ➤ [c#-26] El sistema debe permitir mostrar un icono diferente cuando se arrastre un cubo ➤ [c#-27] El sistema deberá indicar con un cursor distinto cuando no sea un lugar válido para soltar un cubo	Abrir uno de los proyectos generados, se debe mostrar en el área de trabajo los cubos en las posiciones que se guardaron.	Cliente
22	Como usuario, quiero que pueda arrastrar las funciones para generar los cubos	➤ [c#-28] El sistema deberá agregar un cubo en la posición del cursor donde se soltó la función ➤ [c#-29] Se deberá agregar el nombre de la función y la clase a la que pertenece a la lista de funcionalidad ➤ [c#-37] Se debe permitir mostrar un menú al presionar clic derecho a un cubo	Realizar el arrastrado de las funciones y verificar que el cubo corresponda a la función deseada y que la posición sea la misma que la del puntero.	Cliente
23	Como usuario, quiero que pueda configurar los valores de la función y me indique si hay algún error.	➤ [c#-31] Generar control de usuario para cada uno de los nuevos tipos de datos ➤ [c#-32] Escribir algoritmo que	Configurar cubos con diferentes valores y revisar la forma en la que se comporta el color del cubo	Cliente

		genere ventana de configuración a partir de los atributos necesarios ➤ [c#-33] Permitir que al cambiar el valor de un componente se actualice la clase. ➤ [c#-35] Al presionar el botón cerrar se debe actualizar la lista de errores ➤ [c#-36] Se debe deshabilitar el botón de compilar cuando haya errores críticos en el sistema ➤ [c#-38] Los cubos deben cambiar de color según si existe o no error. Verde si está correcto, Amarillo por error no crítico y rojo por error crítico. ➤ [c#-39] Al presionar doble clic sobre un elemento de la lista de funcionalidad, se debe abrir la ventana de configuración ➤ [c#-40] Al presionar doble clic sobre un elemento de la lista de errores, se debe abrir la ventana de configuración	y la consola de errores.	
24	Como usuario, quiero que el programa genere el código del juego para que cualquier persona pueda programar.	➤ [c#-61] El sistema deberá eliminar el contenido de la carpeta SRC cada vez que se compile un juego ➤ [c#-62] El sistema deberá generar las carpetas bin, obj y assets en la carpeta SRC ➤ [c#-63] El archivo batch que firma el juego deberá agregar los recursos de la carpeta assets ➤ [c#-64] Generar un método que catalogue los cubos en maestros y de funcionalidad ➤ [c#-65] El sistema deberá guardar el código generado en un vector ➤ [c#-66] El sistema debe crear los objetos con el nombre que le dio el usuario ➤ [c#-67] El sistema deberá agregar la funcionalidad al código generado ➤ [c#-68] Si el usuario utilizó imágenes, el sistema deberá copiarlas a la carpeta "Assets" ➤ [c#-69] El sistema deberá permitir sobrescribir imágenes que tengan el mismo nombre ➤ [c#-70] El sistema debe revisar los cubos maestros que sean los de inicio	Hacer que la aplicación genere el código fuente de un juego y revisar la clase creada de errores de sintaxis.	Cliente

		➤ [c#-71] El sistema debe invalidar otros cubos maestros si son marcados como pantalla inicial ➤ [c#-72] El sistema deberá mostrar una ventana con el avance de la compilación ➤ [c#-73] El sistema deberá copiar los archivos de la librería a la carpeta SRC ➤ [c#-77] El sistema deberá cerrar la ventana de compilación cuando termine el proceso		
25	Como usuario, quiero que el programa me muestre el juego que cree en la computadora para observar la forma en la que funciona el juego.	➤ [c#-41] Investigar uso de comandos de consola para Android ➤ [c#-42] Crear un batch estático para crear un proyecto ➤ [c#-43] Crear un batch estático para compilar el proyecto ➤ [c#-44] Crear un batch estático para firmar el juego ➤ [c#-45] Generar llave de Java para el framework ➤ [c#-46] Crear un batch estático para ejecutar un emulador ➤ [c#-47] Crear un batch estático para instalar el juego ➤ [c#-48] Crear método batch para limpiar parámetros ➤ [c#-60] Modificar los archivos batch para que reciba parámetros ➤ [c#-74] El sistema deberá ejecutar los archivos batch de forma independiente ➤ [c#-75] El sistema deberá esperar a que termine la ejecución de un archivo batch para comenzar con el siguiente ➤ [c#-76] La ejecución de los archivos batch debe ser transparente al usuario ➤ [c#-78] Por cada vez que el usuario compile, el sistema deberá cargar un emulador independiente ➤ [c#-79] El sistema deberá instalar el juego en todos los emuladores y dispositivos Android conectados	Generar diferentes juegos y revisar que se ejecuten en los simuladores o en cualquier dispositivo Android que se encuentre conectado al equipo.	Cliente
26	Como usuario quiero poder indicarle al programa la ruta donde se guardaron, para que pueda instalar en cualquier carpeta las herramientas	➤ [c#-52] Generar un archivo config.ini donde se guarde la ruta a Android y la ruta a Java ➤ [c#-53] El sistema debe leer el archivo config.ini y configurar las	Instalar Android SDK y Java 1.6.0_26 en carpetas distintas a las que trae por default, hacer la modificación en el	Cliente

		➤ rutas como variables del sistema ➤ [c#-54] Crear una ventana donde se pidan las rutas para el config.ini ➤ [c#-55] La ventana de configuración del config.ini debe mostrar un selector de carpetas ➤ [c#-56] El sistema debe generar un archivo config.ini con los datos de la ventana "Configurar"	programa y verificar que realice la compilación e instalación del juego en el emulador o dispositivo.	
27	Como usuario quiero ver la versión del sistema para que pueda revisar si está actualizado.	➤ [c#-50] El sistema deberá mostrar una ventana con la información de "Acerca De"	Abrir la ventana de "Acerca De" y revisar se encuentra la información deseada.	Cliente
28	Como usuario, quiero poder ingresar mis propios cubos para aumentar la funcionalidad del sistema	➤ [c#-57] Permitir que los usuarios generen sus propios cubos desde la aplicación en una ventana ➤ [c#-58] Guardar el cubo del usuario en la carpeta "Custom" dentro de la carpeta de funciones ➤ [c#-59] Al generar un cubo de usuario, se debe actualizar el árbol de cubos	Ingresar cubos de funcionalidad por medio de la herramienta y revisar en el sistema de archivos que se genere correctamente. Comprobar si se encuentra la función dentro del árbol de opciones.	Cliente
29	Como usuario, quiero que el sistema tenga una ventana de inicio que revise los archivos de configuración para que no falle el sistema en ningún momento.	➤ [c#-80] Generar una ventana de inicio que sea transparente ➤ [c#-81] Generar imagen de fondo con esquinas bordeadas ➤ [c#-82] El sistema deberá revisar la integridad de los archivos al cargar la pantalla de inicio ➤ [c#-83] El sistema debe notificar cualquier error que suceda ➤ [c#-84] El sistema no deberá tardar más de 4 segundos en hacer las verificaciones	Realizar pruebas quitando algunos archivos o carpetas y comprobar el comportamiento del sistema.	Cliente
30	Como desarrollador, quiero generar un instalador en un clic para que la distribución del sistema sea rápida.	➤ [c#-85] El sistema deberá instalar la aplicación sobre la carpeta "Archivos de Programa" ➤ [c#-86] El sistema deberá generar todos los archivos de configuración ➤ [c#-87] El sistema deberá generar un acceso directo en el escritorio ➤ [c#-88] El sistema deberá crear una carpeta en el menú inicio ➤ [c#-89] El sistema deberá contener los instaladores de las herramientas ➤ [c#-90] El sistema deberá otorgar los permisos al sistema para escribir sobre la unidad del Sistema Operativo	N/A	Técnica

D.2 Product Backlog

ID	Módulo	Descripción de la característica	Sprint Backlog Asociado	Prioridad
J2ME-0	Librería J2ME	Descarga, instalación y configuración del ambiente de trabajo para dispositivos J2ME (Symbian)	Sprint 1	100
J2ME-1	Librería J2ME	Generar los paquetes interfaces, library e util dentro del paquete com.upaep.thesis	Sprint 1	90
J2ME-2	Librería J2ME	Generar una clase para el control de las dimensiones y resolución de pantalla llamado "Limit"	Sprint 1	90
J2ME-3	Librería J2ME	Generar la clase "Background" como patrón de diseño para pantallas	Sprint 1	90
J2ME-4	Librería J2ME	Generar un "Garbage Collector" llamado "Destroyer" para eliminar y ahorrar pila del dispositivo.	Sprint 1	90
J2ME-5	Librería J2ME	Generar socket de conexión HTTP para el Web Service	Sprint 1	80
J2ME-6	Librería J2ME	Generar una lista de fuentes utilizables en todas las clases	Sprint 1	80
J2ME-7	Librería J2ME	Clase para manejar sonidos del sistema operativo y canciones en formato MP3 y WAV	Sprint 1	80
J2ME-8	Librería J2ME	Hacer una alerta independiente que se pueda quitar al presionar un botón o al terminarse un tiempo	Sprint 1	90
J2ME-9	Librería J2ME	Generar una clase para controlar el envío de mensajes de texto	Sprint 1	90
J2ME-10	Librería J2ME	Permitir que se pueda guardar datos en la memoria persistente del dispositivo	Sprint 1	90
J2ME-11	Librería J2ME	Crear una clase que permite generar efectos de profundidad en texto y bordes de las alertas	Sprint 1	90
J2ME-12	Librería J2ME	Controlador de Base de Datos temporal para dispositivos que lo soporten	Sprint 1	90
J2ME-13	Librería J2ME	Generar un patrón de opciones para cambiar configuración, cambiar de pantalla y salir.	Sprint 1	80
J2ME-14	Librería J2ME	Crear clase para menú, debe permitir agregar un título, y las opciones se deben adaptar a la pantalla.	Sprint 1	80
J2ME-15	Librería J2ME	Permitir cambiar el color de fondo y del texto de la opción seleccionada y no seleccionada	Sprint 1	80
J2ME-16	Librería J2ME	Permitir poner imágenes como fondo en la pantalla Menú	Sprint 1	80

J2ME-17	Librería J2ME	Desarrollo de una pantalla de inicio, donde se permita poner el logo del juego y el desarrollador.	Sprint 1	80
J2ME-18	Librería J2ME	Permitir que al navegar en las opciones del menú pueda reproducirse un sonido del sistema operativo o de archivo	Sprint 1	80
J2ME-19	Librería J2ME	Crear una clase que permita mostrar texto plano en la pantalla del dispositivo	Sprint 1	80
J2ME-20	Librería J2ME	Permitir que se pueda agregar solo una opción a la pantalla de instrucciones	Sprint 1	80
J2ME-21	Librería J2ME	Permitir cambiar el color de fondo y del texto mostrado.	Sprint 1	80
J2ME-22	Librería J2ME	Permitir poner el texto de la pantalla instrucciones a partir de un archivo de texto plano	Sprint 1	80
J2ME-23	Librería J2ME	Generar barras de desplazamiento cuando el texto sea mayor al tamaño de la ventana	Sprint 1	80
J2ME-24	Librería J2ME	Permitir que el texto se ajuste al ancho de la ventana sin cortar palabras.	Sprint 1	80
A2.1-0	Librería Android	Descarga, instalación y configuración del ambiente de trabajo para dispositivos Android 2.1 Eclair	Sprint 1	100
A2.1-1	Librería Android	Crear los paquetes framework, classes y library dentro del paquete com.upaep.thesis	Sprint 1	100
A2.1-2	Librería Android	Cambiar la configuración global del manifest para aceptar Internet y Control de Pantalla	Sprint 1	100
A2.1-3	Librería Android	Crear interfaces para el control de los dispositivos periféricos (teclado, touch, etc.)	Sprint 1	90
A2.1-4	Librería Android	Crear interfaces para reproducir sonidos de sistema y archivos en formato MP3	Sprint 1	90
A2.1-5	Librería Android	Crear una interfaz para el control de gráficos en pantalla, permitiendo generar imágenes y dibujar figuras geométricas	Sprint 1	90
A2.1-6	Librería Android	Permitir que el usuario elija la calidad de los gráficos a mostrar en la pantalla (ARGB8888, ARGB4444 o RGB565)	Sprint 1	90
A2.1-7	Librería Android	Generar una clase que funcione como contenedor y generador de objetos siguiendo el patrón Singleton	Sprint 1	90
A2.1-8	Librería Android	Generar una interfaz que sirva como patrón de diseño para todas las pantallas que se muestran al usuario	Sprint 1	90
A2.1-9	Librería Android	Generar una interfaz que permita al usuario abrir imágenes con formato PNG y JPG	Sprint 1	90

A2.1-10	Librería Android	Generar una interfaz para abrir archivos localizados en el paquete o en la memoria interna o externa del dispositivo	Sprint 1	90
A2.1-11	Librería Android	Generar una clase para el control del acelerómetro	Sprint 1	90
A2.1-12	Librería Android	Permitir que el acelerómetro se pueda calibrar y configurar para videojuegos	Sprint 1	90
A2.1-13	Librería Android	Permitir que el usuario elija el acelerómetro que se va a utilizar cuando se tienen más de uno en el dispositivo	Sprint 1	90
A2.1-14	Librería Android	Generar una clase que permita la lectura y escritura de archivos en texto plano	Sprint 1	90
A2.1-15	Librería Android	Permitir que la clase de lectura y escritura busque en el dispositivo las memorias de almacenamiento interno y externo conectadas	Sprint 1	90
A2.1-16	Librería Android	Generar una clase basándose en la interfaz de sonido que permita localizar y reproducir un archivo de audio	Sprint 1	90
A2.1-17	Librería Android	Permitir que el usuario pueda configurar el volumen inicial del archivo de audio	Sprint 1	80
A2.1-18	Librería Android	Permitir que el usuario cambie el volumen del archivo en tiempo de ejecución	Sprint 1	80
A2.1-19	Librería Android	Permitir al usuario pausar y continuar con la reproducción del sonido en tiempo real	Sprint 1	80
A2.1-20	Librería Android	Crear una clase que permita reproducir sonidos predefinidos en el sistema operativo basados en la interfaz creada	Sprint 1	80
A2.1-21	Librería Android	Crear un controlador del teclado del dispositivo detectando el número de teclas con los que cuenta	Sprint 1	90
A2.1-22	Librería Android	Permitir que el controlador de teclado devuelva una lista con todas las teclas y el status de cada una de ellas	Sprint 1	80
A2.1-23	Librería Android	Generar una clase que permita configurar la calidad de las imágenes cargadas al sistema	Sprint 1	90
A2.1-24	Librería Android	Permitir que se obtenga el ancho, largo y calidad de una imagen cargada	Sprint 1	80
A2.1-25	Librería Android	Generar una clase que permita detectar las características de pantalla touch	Sprint 1	90
A2.1-26	Librería Android	La clase debe permitir guardar los movimientos realizados en la pantalla touch	Sprint 1	80
A2.1-27	Librería Android	La clase permitir hacer multitouch a la pantalla si lo permite el dispositivo	Sprint 1	90

A2.1-28	Librería Android	Generar una clase que permita configurar automáticamente las aptitudes del dispositivo	Sprint 1	80
A2.1-29	Librería Android	La clase debe configurar la pantalla touch a partir de la versión de Android	Sprint 1	80
A2.1-30	Librería Android	Crear una clase que englobe la funcionalidad de reproducir audio desde archivo o desde sistema tomando en cuenta las características del dispositivo	Sprint 1	80
A2.1-31	Librería Android	Generar una clase que permita actualizar el estado de la aplicación	Sprint 1	90
A2.1-32	Librería Android	Permitir que la clase utilice doble buffer para actualizar los gráficos	Sprint 1	90
A2.1-33	Librería Android	Generar una clase que permita ejecutar cualquier pantalla de juego	Sprint 1	90
A2.1-34	Librería Android	Generar un conector HTTP	Sprint 1	80
c#-0	Framework	Descarga, instalación y configuración de Visual Studio 2012 para desarrollar el framework	Sprint 1	100
c#-1	Framework	Generar el proyecto bajo el framework 4.0	Sprint 2	70
c#-2	Framework	Generar las carpetas Classes, Controls y Forms	Sprint 2	60
c#-3	Framework	Hacer que la ventana principal inicie maximizada	Sprint 2	60
c#-4	Framework	Insertar un árbol de opciones en el lado izquierdo de la ventana	Sprint 2	60
c#-5	Framework	Insertar dos controles de vista en el lado derecho para las funciones agregadas y los errores	Sprint 2	60
c#-6	Framework	Insertar un panel en la parte central que permita posicionar los cubos	Sprint 2	60
c#-7	Framework	Permitir que los controles se readapten al tamaño de la ventana	Sprint 2	60
c#-8	Framework	Agregar una barra de menús con las opciones Inicio, Proyecto, Cubos y Ayuda	Sprint 2	60
c#-9	Framework	En el menú Inicio agregar las opciones Nuevo, Abrir, Guardar, Exportar como Imagen y Salir	Sprint 2	60
c#-10	Framework	En el menú Proyecto agregar las opciones Verificar y Compilar	Sprint 2	60
c#-11	Framework	En el menú Cubos agregar las opciones Crear cubo, Modificar cubo y Eliminar Cubo	Sprint 2	60
c#-12	Framework	En el menú Ayuda agregar las opciones Ayuda, Documentación y Acerca De	Sprint 2	60
c#-13	Framework	Agregar una barra de accesos rápidos con las funciones Nuevo Proyecto, Abrir, Guardar, Compilar, Nuevo Cubo, Modificar Cubo, Eliminar Cubo y Ayuda	Sprint 2	60

c#-14	Framework	Generar un patrón de diseño para las funciones que agregará el framework	Sprint 2	60
c#-15	Framework	Generar una clase que permita leer un archivo que se encuentre en una ruta específica	Sprint 2	60
c#-16	Framework	Permitir que la clase lectora guarde las líneas del archivo en un vector	Sprint 2	50
c#-17	Framework	Generar una clase que permita crear la carpeta workspace en la carpeta de Mis Documentos	Sprint 2	50
c#-18	Framework	Crear una clase que permita obtener el nombre de todos los proyectos creados en la carpeta workspace	Sprint 2	50
c#-19	Framework	Crear un método para buscar el nombre de un proyecto en el directorio workspace	Sprint 2	50
c#-20	Framework	Crear una clase que permita generar archivos en el disco duro	Sprint 2	60
c#-21	Framework	Escribir funciones para el framework de prueba en formato DLL	Sprint 2	50
c#-22	Framework	Desarrollar algoritmo para análisis y agrupación de funcionalidades según su tipo	Sprint 2	60
c#-23	Framework	Generar método que permita guardar la información de un proyecto en disco	Sprint 2	50
c#-24	Framework	Permitir que el sistema revise los archivos de un proyecto y el tipo de sistema operativo empleado	Sprint 2	60
c#-25	Framework	El sistema debe permitir abrir proyectos ya generados	Sprint 2	50
c#-26	Framework	El sistema debe permitir mostrar un icono diferente cuando se arrastre un cubo	Sprint 2	60
c#-27	Framework	El sistema deberá indicar con un cursor distinto cuando no sea un lugar válido para soltar un cubo	Sprint 2	50
c#-28	Framework	El sistema deberá agregar un cubo en la posición del cursor donde se soltó la función	Sprint 2	50
c#-29	Framework	Se deberá agregar el nombre de la función y la clase a la que pertenece a la lista de funcionalidad	Sprint 2	50
c#-30	Framework	Generar tipos de datos para los atributos de las funciones	Sprint 2	60
c#-31	Framework	Generar control de usuario para cada uno de los nuevos tipos de datos	Sprint 2	60
c#-32	Framework	Escribir algoritmo que genere ventana de configuración a partir de los atributos necesarios	Sprint 2	60
c#-33	Framework	Permitir que al cambiar el valor de un componente se actualice la clase.	Sprint 2	50
c#-34	Framework	Se deberá catalogar cada atributo con los valores 1 para crítico y 0 para no crítico	Sprint 2	70

c#-35	Framework	Al presionar el botón cerrar se debe actualizar la lista de errores	Sprint 2	70
c#-36	Framework	Se debe deshabilitar el botón de compilar cuando haya errores críticos en el sistema	Sprint 2	50
c#-37	Framework	Se debe permitir mostrar un menú al presionar clic derecho a un cubo	Sprint 2	50
c#-38	Framework	Los cubos deben cambiar de color según si existe o no error. Verde si está correcto, Amarillo por error no crítico y rojo por error crítico.	Sprint 2	50
c#-39	Framework	Al presionar doble clic sobre un elemento de la lista de funcionalidad, se debe abrir la ventana de configuración	Sprint 2	50
c#-40	Framework	Al presionar doble clic sobre un elemento de la lista de errores, se debe abrir la ventana de configuración	Sprint 2	50
c#-41	Framework	Investigar uso de comandos de consola para Android	Sprint 3	40
c#-42	Framework	Crear un batch estático para crear un proyecto	Sprint 3	30
c#-43	Framework	Crear un batch estático para compilar el proyecto	Sprint 3	30
c#-44	Framework	Crear un batch estático para firmar el juego	Sprint 3	30
c#-45	Framework	Generar llave de Java para el framework	Sprint 3	30
c#-46	Framework	Crear un batch estático para ejecutar un emulador	Sprint 3	20
c#-47	Framework	Crear un batch estático para instalar el juego	Sprint 3	20
c#-48	Framework	Crear método batch para limpiar parámetros	Sprint 3	20
c#-49	Framework	El sistema deberá guardar los cubos empleados y la posición en pantalla que ocupa	Sprint 3	30
c#-50	Framework	El sistema deberá mostrar una ventana con la información de "Acerca De"	Sprint 3	30
c#-51	Framework	Se deben agregar iconos a cada una de las funciones	Sprint 3	30
c#-52	Framework	Generar un archivo config.ini donde se guarde la ruta a Android y la ruta a Java	Sprint 3	30
c#-53	Framework	El sistema debe leer el archivo config.ini y configurar las rutas como variables del sistema	Sprint 3	30
c#-54	Framework	Crear una ventana donde se pidan las rutas para el config.ini	Sprint 3	20
c#-55	Framework	La ventana de configuración del config.ini debe mostrar un selector de carpetas	Sprint 3	20
c#-56	Framework	El sistema debe generar un archivo config.ini con los datos de la ventana "Configurar"	Sprint 3	20
c#-57	Framework	Permitir que los usuarios generen sus propios cubos desde la aplicación en una ventana	Sprint 3	20

c#-58	Framework	Guardar el cubo del usuario en la carpeta "Custom" dentro de la carpeta de funciones	Sprint 3	20
c#-59	Framework	Al generar un cubo de usuario, se debe actualizar el árbol de cubos	Sprint 3	20
c#-60	Framework	Modificar los archivos batch para que reciba parámetros	Sprint 3	20
c#-61	Framework	El sistema deberá eliminar el contenido de la carpeta SRC cada vez que se compile un juego	Sprint 3	20
c#-62	Framework	El sistema deberá generar las carpetas bin, obj y assets en la carpeta SRC	Sprint 3	20
c#-63	Framework	El archivo batch que firma el juego deberá agregar los recursos de la carpeta assets	Sprint 3	20
c#-64	Framework	Generar un método que catalogue los cubos en maestros y de funcionalidad	Sprint 3	20
c#-65	Framework	El sistema deberá guardar el código generado en un vector	Sprint 3	20
c#-66	Framework	El sistema debe crear los objetos con el nombre que le dio el usuario	Sprint 3	40
c#-67	Framework	El sistema deberá agregar la funcionalidad al código generado	Sprint 3	40
c#-68	Framework	Si el usuario utilizó imágenes, el sistema deberá copiarlas a la carpeta "Assets"	Sprint 3	40
c#-69	Framework	El sistema deberá permitir sobrescribir imágenes que tengan el mismo nombre	Sprint 3	30
c#-70	Framework	El sistema debe revisar los cubos maestros que sean los de inicio	Sprint 3	30
c#-71	Framework	El sistema debe invalidar otros cubos maestros si son marcados como pantalla inicial	Sprint 3	30
c#-72	Framework	El sistema deberá mostrar una ventana con el avance de la compilación	Sprint 3	30
c#-73	Framework	El sistema deberá copiar los archivos de la librería a la carpeta SRC	Sprint 3	30
c#-74	Framework	El sistema deberá ejecutar los archivos batch de forma independiente	Sprint 3	30
c#-75	Framework	El sistema deberá esperar a que termine la ejecución de un archivo batch para comenzar con el siguiente	Sprint 3	30
c#-76	Framework	La ejecución de los archivos batch debe ser transparente al usuario	Sprint 3	30
c#-77	Framework	El sistema deberá cerrar la ventana de compilación cuando termine el proceso	Sprint 3	30

c#-78	Framework	Por cada vez que el usuario compile, el sistema deberá cargar un emulador independiente	Sprint 3	30
c#-79	Framework	El sistema deberá instalar el juego en todos los emuladores y dispositivos Android conectados	Sprint 3	30
c#-80	Framework	Generar una ventana de inicio que sea transparente	Sprint 3	30
c#-81	Framework	Generar imagen de fondo con esquinas bordeadas	Sprint 3	30
c#-82	Framework	El sistema deberá revisar la integridad de los archivos al cargar la pantalla de inicio	Sprint 3	30
c#-83	Framework	El sistema debe notificar cualquier error que suceda	Sprint 3	20
c#-84	Framework	El sistema no deberá tardar más de 4 segundos en hacer las verificaciones	Sprint 3	20
c#-85	Framework	El sistema deberá instalar la aplicación sobre la carpeta "Archivos de Programa"	Sprint 3	20
c#-86	Framework	El sistema deberá generar todos los archivos de configuración	Sprint 3	20
c#-87	Framework	El sistema deberá generar un acceso directo en el escritorio	Sprint 3	30
c#-88	Framework	El sistema deberá crear una carpeta en el menú inicio	Sprint 3	20
c#-89	Framework	El sistema deberá contener los instaladores de las herramientas	Sprint 3	30
c#-90	Framework	El sistema deberá otorgar los permisos al sistema para escribir sobre la unidad del Sistema Operativo	Sprint 3	20
PHP-0	Sitio Web y Web Service	Descarga, instalación y configuración del servidor apache y gestor de base de datos MySQL	Sprint 1	100
PHP-1	Sitio Web y Web Service	Análisis de funcionalidad para sistema de puntajes en línea	Sprint 1	80
PHP-2	Sitio Web y Web Service	Diseño de la base de datos en MySQL Workbench	Sprint 1	80
PHP-3	Sitio Web y Web Service	Diseño del Web Service para PHP	Sprint 1	80
PHP-4	Sitio Web y Web Service	Implementación del diseño de base de datos en MySQL 5.5	Sprint 1	80
PHP-5	Sitio Web y Web Service	Implementación de la base de datos en PHPMyAdmin del servidor	Sprint 1	80
PHP-6	Sitio Web y Web Service	Desarrollo del conector a la base de datos MySQL desde PHP	Sprint 2	50
PHP-7	Sitio Web y Web Service	Generar un método para obtener los nombres y puntajes de usuarios	Sprint 2	50
PHP-8	Sitio Web y Web Service	Generar un método para agregar los puntajes de un usuario	Sprint 2	50

PHP-9	Sitio Web y Web Service	Generar un método para agregar nuevos usuarios al sistema	Sprint 2	50
PHP-10	Sitio Web y Web Service	Generar un método para obtener el estado del servicio	Sprint 2	50
PHP-11	Sitio Web y Web Service	Generar variables ocultas para los tipos de métodos	Sprint 2	70
PHP-12	Sitio Web y Web Service	Generar el web service para acceder a los métodos	Sprint 2	70

D.3 Product Backlog Primer Sprint

ID	Tarea	Tiempo (Minutos)		Estatus	Valor Ganado	Prioridad
		Estimado	Real			
J2ME-0	Descarga, instalación y configuración del ambiente de trabajo para dispositivos J2ME (Symbian)	250	620	L	5.59%	Alta
J2ME-1	Generar los paquetes interfaces, library e util dentro del paquete com.upaep.thesis	10	5	L	0.22%	Media
J2ME-2	Generar una clase para el control de las dimensiones y resolución de pantalla llamado "Limit"	10	15	L	0.22%	Media
J2ME-3	Generar la clase "Background" como patrón de diseño para pantallas	55	40	L	1.23%	Media
J2ME-4	Generar un "Garbage Collector" llamado "Destroyer" para eliminar y ahorrar pila del dispositivo.	15	10	L	0.34%	Media
J2ME-5	Generar socket de conexión HTTP para el Web Service	120	135	L	2.68%	Baja
J2ME-6	Generar una lista de fuentes utilizables en todas las clases	60	75	L	1.34%	Baja
J2ME-7	Clase para manejar sonidos del sistema operativo y canciones en formato MP3 y WAV	165	145	L	3.69%	Baja
J2ME-8	Hacer una alerta independiente que se pueda quitar al presionar un botón o al terminarse un tiempo	90	100	L	2.01%	Media
J2ME-9	Generar una clase para controlar el envío de mensajes de texto	50	60	L	1.12%	Media
J2ME-10	Permitir que se pueda guardar datos en la memoria persistente del dispositivo	70	90	L	1.56%	Media
J2ME-11	Crear una clase que permite generar efectos de profundidad en texto y bordes de las alertas	110	95	L	2.46%	Media
J2ME-12	Controlador de Base de Datos temporal para dispositivos que lo soporten	130	120	L	2.91%	Media

J2ME-13	Generar un patrón de opciones para cambiar configuración, cambiar de pantalla y salir.	110	83	L	2.46%	Baja
J2ME-14	Crear clase para menú, debe permitir agregar un título, y las opciones se deben adaptar a la pantalla.	50	50	L	1.12%	Baja
J2ME-15	Permitir cambiar el color de fondo y del texto de la opción seleccionada y no seleccionada	60	65	L	1.34%	Baja
J2ME-16	Permitir poner imágenes como fondo en la pantalla Menú	60	50	L	1.34%	Baja
J2ME-17	Desarrollo de una pantalla de inicio, donde se permita poner el logo del juego y el desarrollador.	80	90	L	1.79%	Baja
J2ME-18	Permitir que al navegar en las opciones del menú pueda reproducirse un sonido del sistema operativo o de archivo	50	40	L	1.12%	Baja
J2ME-19	Crear una clase que permita mostrar texto plano en la pantalla del dispositivo	90	105	L	2.01%	Baja
J2ME-20	Permitir que se pueda agregar solo una opción a la pantalla de instrucciones	45	40	L	1.01%	Baja
J2ME-21	Permitir cambiar el color de fondo y del texto mostrado.	40	39	L	0.89%	Baja
J2ME-22	Permitir poner el texto de la pantalla instrucciones a partir de un archivo de texto plano	90	110	L	2.01%	Baja
J2ME-23	Generar barras de desplazamiento cuando el texto sea mayor al tamaño de la ventana	60	75	L	1.34%	Baja
J2ME-24	Permitir que el texto se ajuste al ancho de la ventana sin cortar palabras.	90	85	L	2.01%	Baja
A2.1-0	Descarga, instalación y configuración del ambiente de trabajo para dispositivos Android 2.1 Eclair	500	330	L	11.17%	Alta
A2.1-1	Crear los paquetes framework, clases y library dentro del paquete com.upaep.thesis	10	5	L	0.22%	Alta
A2.1-2	Cambiar la configuración global del manifest para aceptar Internet y Control de Pantalla	20	35	L	0.45%	Alta
A2.1-3	Crear interfaces para el control de los dispositivos periféricos (teclado, touch, etc.)	30	20	L	0.67%	Media
A2.1-4	Crear interfaces para reproducir sonidos de sistema y archivos en formato MP3	20	18	L	0.45%	Media
A2.1-5	Crear una interfaz para el control de gráficos en pantalla, permitiendo generar imágenes y dibujar figuras geométricas	30	40	L	0.67%	Media
A2.1-6	Permitir que el usuario elija la calidad de los gráficos a mostrar en la pantalla (ARGB8888, ARGB4444 o RGB565)	40	20	L	0.89%	Media

A2.1-7	Generar una clase que funcione como contenedor y generador de objetos siguiendo el patrón Singleton	55	80	L	1.23%	Media
A2.1-8	Generar una interfaz que sirva como patrón de diseño para todas las pantallas que se muestran al usuario	20	40	L	0.45%	Media
A2.1-9	Generar una interfaz que permita al usuario abrir imágenes con formato PNG y JPG	25	15	L	0.56%	Media
A2.1-10	Generar una interfaz para abrir archivos localizados en el paquete o en la memoria interna o externa del dispositivo	30	60	L	0.67%	Media
A2.1-11	Generar una clase para el control del acelerómetro	100	90	L	2.23%	Media
A2.1-12	Permitir que el acelerómetro se pueda calibrar y configurar para videojuegos	20	25	L	0.45%	Media
A2.1-13	Permitir que el usuario elija el acelerómetro que se va a utilizar cuando se tienen más de uno en el dispositivo	40	20	L	0.89%	Media
A2.1-14	Generar una clase que permita la lectura y escritura de archivos en texto plano	30	40	L	0.67%	Media
A2.1-15	Permitir que la clase de lectura y escritura busque en el dispositivo las memorias de almacenamiento interno y externo conectadas	15	5	L	0.34%	Media
A2.1-16	Generar una clase basándose en la interfaz de sonido que permita localizar y reproducir un archivo de audio	60	80	L	1.34%	Media
A2.1-17	Permitir que el usuario pueda configurar el volumen inicial del archivo de audio	30	10	L	0.67%	Baja
A2.1-18	Permitir que el usuario cambie el volumen del archivo en tiempo de ejecución	20	15	L	0.45%	Baja
A2.1-19	Permitir al usuario pausar y continuar con la reproducción del sonido en tiempo real	30	20	L	0.67%	Baja
A2.1-20	Crear una clase que permita reproducir sonidos predefinidos en el sistema operativo basados en la interfaz creada	60	40	L	1.34%	Baja
A2.1-21	Crear un controlador del teclado del dispositivo detectando el número de teclas con los que cuenta	30	60	L	0.67%	Media
A2.1-22	Permitir que el controlador de teclado devuelva una lista con todas las teclas y el status de cada una de ellas	30	25	L	0.67%	Baja
A2.1-23	Generar una clase que permita configurar la calidad de las imágenes cargadas al sistema	15	20	L	0.34%	Media
A2.1-24	Permitir que se obtenga el ancho, largo y calidad de una imagen cargada	20	10	L	0.45%	Baja

A2.1-25	Generar una clase que permita detectar las características de pantalla touch	25	60	L	0.56%	Media
A2.1-26	La clase debe permitir guardar los movimientos realizados en la pantalla touch	30	20	L	0.67%	Baja
A2.1-27	La clase permitir hacer multitouch a la pantalla si lo permite el dispositivo	30	25	L	0.67%	Media
A2.1-28	Generar una clase que permita configurar automáticamente las aptitudes del dispositivo	80	100	L	1.79%	Baja
A2.1-29	La clase debe configurar la pantalla touch a partir de la versión de Android	30	10	L	0.67%	Baja
A2.1-30	Crear una clase que englobe la funcionalidad de reproducir audio desde archivo o desde sistema tomando en cuenta las características del dispositivo	30	55	L	0.67%	Baja
A2.1-31	Generar una clase que permita actualizar el estado de la aplicación	60	100	L	1.34%	Media
A2.1-32	Permitir que la clase utilice doble buffer para actualizar los gráficos	40	30	L	0.89%	Media
A2.1-33	Generar una clase que permita ejecutar cualquier pantalla de juego	130	190	L	2.91%	Media
A2.1-34	Generar un conector HTTP	100	50	L	2.23%	Baja
c#-0	Descarga, instalación y configuración de Visual Studio 2012 para desarrollar el framework	150	120	L	3.35%	Alta
PHP-0	Descarga, instalación y configuración del servidor apache y gestor de base de datos MySQL	200	210	L	4.47%	Alta
PHP-1	Análisis de funcionalidad para sistema de puntajes en línea	100	80	L	2.23%	Baja
PHP-2	Diseño de la base de datos en MySQL Workbench	60	80	L	1.34%	Baja
PHP-3	Diseño del Web Service para PHP	60	50	L	1.34%	Baja
PHP-4	Implementación del diseño de base de datos en MySQL 5.5	50	40	L	1.12%	Baja
PHP-5	Implementación de la base de datos en PHPMyAdmin del servidor	60	75	L	1.34%	Baja
		4475	4760		100%	

Porcentaje de Desviación: 6.36%

D.4 Product Backlog Segundo Sprint

ID	Tarea	Tiempo (Minutos)		Estatus	Valor Ganado	Prioridad
		Estimado	Real			
PHP-6	Desarrollo del conector a la base de datos MySQL desde PHP	20	10	L	0.57%	Baja
PHP-7	Generar un método para obtener los nombres y puntajes de usuarios	60	50	L	1.70%	Baja
PHP-8	Generar un método para agregar los puntajes de un usuario	60	45	L	1.70%	Baja
PHP-9	Generar un método para agregar nuevos usuarios al sistema	60	50	L	1.70%	Baja
PHP-10	Generar un método para obtener el estado del servicio	60	45	L	1.70%	Baja
PHP-11	Generar variables ocultas para los tipos de métodos	20	30	L	0.57%	Alta
PHP-12	Generar el Web Service para acceder a los métodos	100	120	L	2.84%	Alta
C#-1	Generar el proyecto bajo el framework 4.0	10	5	L	0.28%	Alta
C#-2	Generar las carpetas Classes, Controls y Forms	10	5	L	0.28%	Media
C#-3	Hacer que la ventana principal inicie maximizada	5	10	L	0.14%	Media
C#-4	Insertar un árbol de opciones en el lado izquierdo de la ventana	25	15	L	0.71%	Media
C#-5	Insertar dos controles de vista en el lado derecho para las funciones agregadas y los errores	30	25	L	0.85%	Media
C#-6	Insertar un panel en la parte central que permita posicionar los cubos	20	15	L	0.57%	Media
C#-7	Permitir que los controles se readapten al tamaño de la ventana	45	60	L	1.28%	Media
C#-8	Agregar una barra de menús con las opciones Inicio, Proyecto, Cubos y Ayuda	30	40	L	0.85%	Media
C#-9	En el menú Inicio agregar las opciones Nuevo, Abrir, Guardar, Exportar como Imagen y Salir	15	10	L	0.43%	Media
C#-10	En el menú Proyecto agregar las opciones Verificar y Compilar	15	10	L	0.43%	Media
C#-11	En el menú Cubos agregar las opciones Crear cubo, Modificar cubo y Eliminar Cubo	15	10	L	0.43%	Media
C#-12	En el menú Ayuda agregar las opciones Ayuda, Documentación y Acerca De	15	10	L	0.43%	Media
C#-13	Agregar una barra de accesos rápidos con las funciones Nuevo Proyecto, Abrir, Guardar, Compilar, Nuevo Cubo, Modificar Cubo, Eliminar Cubo y Ayuda	30	45	L	0.85%	Media

C#-14	Generar un patrón de diseño para las funciones que agregará el framework	60	50	L	1.70%	Media
C#-15	Generar una clase que permita leer un archivo que se encuentre en una ruta específica	40	25	L	1.14%	Media
C#-16	Permitir que la clase lectora guarde las líneas del archivo en un vector	20	30	L	0.57%	Baja
C#-17	Generar una clase que permita crear la carpeta workspace en la carpeta de Mis Documentos	40	35	L	1.14%	Baja
C#-18	Crear una clase que permita obtener el nombre de todos los proyectos creados en la carpeta workspace	100	80	L	2.84%	Baja
C#-19	Crear un método para buscar el nombre de un proyecto en el directorio workspace	40	60	L	1.14%	Baja
C#-20	Crear una clase que permita generar archivos en el disco duro	150	125	L	4.26%	Media
C#-21	Escribir funciones para el framework de prueba en formato DLL	200	320	L	5.68%	Baja
C#-22	Desarrollar algoritmo para análisis y agrupación de funcionalidades según su tipo	250	300	L	7.10%	Media
C#-23	Generar método que permita guardar la información de un proyecto en disco	150	100	L	4.26%	Baja
C#-24	Permitir que el sistema revise los archivos de un proyecto y el tipo de sistema operativo empleado	60	120	L	1.70%	Media
C#-25	El sistema debe permitir abrir proyectos ya generados	80	95	L	2.27%	Baja
C#-26	El sistema debe permitir mostrar un icono diferente cuando se arrastre un cubo	35	20	L	0.99%	Media
C#-27	El sistema deberá indicar con un cursor distinto cuando no sea un lugar válido para soltar un cubo	35	15	L	0.99%	Baja
C#-28	El sistema deberá agregar un cubo en la posición del cursor donde se soltó la función	200	280	L	5.68%	Baja
C#-29	Se deberá agregar el nombre de la función y la clase a la que pertenece a la lista de funcionalidad	100	45	L	2.84%	Baja
C#-30	Generar tipos de datos para los atributos de las funciones	480	600	L	13.64%	Media
C#-31	Generar control de usuario para cada uno de los nuevos tipos de datos	200	150	L	5.68%	Media
C#-32	Escribir algoritmo que genere ventana de configuración a partir de los atributos necesarios	150	107	L	4.26%	Media
C#-33	Permitir que al cambiar el valor de un componente se actualice la clase.	100	60	L	2.84%	Baja
C#-34	Se deberá catalogar cada atributo con los valores 1 para crítico y 0 para no crítico	150	210	L	4.26%	Alta

C#-35	Al presionar el botón cerrar se debe actualizar la lista de errores	100	92	L	2.84%	Alta
C#-36	Se debe deshabilitar el botón de compilar cuando haya errores críticos en el sistema	15	40	L	0.43%	Baja
C#-37	Se debe permitir mostrar un menú al presionar clic derecho a un cubo	10	25	L	0.28%	Baja
C#-38	Los cubos deben cambiar de color según si existe o no error. Verde si está correcto, Amarillo por error no crítico y rojo por error crítico.	50	62	L	1.42%	Baja
C#-39	Al presionar doble clic sobre un elemento de la lista de funcionalidad, se debe abrir la ventana de configuración	30	40	L	0.85%	Baja
C#-40	Al presionar doble clic sobre un elemento de la lista de errores, se debe abrir la ventana de configuración	30	35	L	0.85%	Baja
		3520	3731		100%	

Porcentaje de Desviación: 5.99%

D.5 Product Backlog Tercer Sprint

ID	Tarea	Tiempo (Minutos)		Estatus	Valor Ganado	Prioridad
		Estimado	Real			
c#-41	Investigar uso de comandos de consola para Android	700	850	L	19.53%	Alta
c#-42	Crear un batch estático para crear un proyecto	50	90	L	1.39%	Media
c#-43	Crear un batch estático para compilar el proyecto	50	60	L	1.39%	Media
c#-44	Crear un batch estático para firmar el juego	50	75	L	1.39%	Media
c#-45	Generar llave de Java para el framework	30	20	L	0.84%	Media
c#-46	Crear un batch estático para ejecutar un emulador	50	35	L	1.39%	Baja
c#-47	Crear un batch estático para instalar el juego	50	60	L	1.39%	Baja
c#-48	Crear método batch para limpiar parametros	60	120	L	1.67%	Baja
c#-49	El sistema deberá guardar los cubos empleados y la posición en pantalla que ocupa	80	120	L	2.23%	Media
c#-50	El sistema deberá mostrar una ventana con la información de "Acerca De"	50	30	L	1.39%	Media
c#-51	Se deben agregar iconos a cada una de las funciones	90	70	L	2.51%	Media
c#-52	Generar un archivo config.ini donde se guarde la ruta a Android y la ruta a Java	45	60	L	1.26%	Media
c#-53	El sistema debe leer el archivo config.ini y configurar las rutas como variables del sistema	30	45	L	0.84%	Media

c#-54	Crear una ventana donde se pidan las rutas para el config.ini	80	50	L	2.23%	Baja
c#-55	La ventana de configuración del config.ini debe mostrar un selector de carpetas	30	25	L	0.84%	Baja
c#-56	El sistema debe generar un archivo config.ini con los datos de la ventana "Configurar"	150	190	L	4.18%	Baja
c#-57	Permitir que los usuarios generen sus propios cubos desde la aplicación en una ventana	80	100	L	2.23%	Baja
c#-58	Guardar el cubo del usuario en la carpeta "Custom" dentro de la carpeta de funciones	60	50	L	1.67%	Baja
c#-59	Al generar un cubo de usuario, se debe actualizar el árbol de cubos	30	45	L	0.84%	Baja
c#-60	Modificar los archivos batch para que reciba parámetros	50	130	L	1.39%	Baja
c#-61	El sistema deberá eliminar el contenido de la carpeta SRC cada vez que se compile un juego	40	20	L	1.12%	Baja
c#-62	El sistema deberá generar las carpetas bin, obj y assets en la carpeta SRC	45	30	L	1.26%	Baja
c#-63	El archivo batch que firma el juego deberá agregar los recursos de la carpeta assets	50	60	L	1.39%	Baja
c#-64	Generar un método que catalogue los cubos en maestros y de funcionalidad	50	80	L	1.39%	Baja
c#-65	El sistema deberá guardar el código generado en un vector	35	15	L	0.98%	Baja
c#-66	El sistema debe crear los objetos con el nombre que le dio el usuario	35	40	L	0.98%	Alta
c#-67	El sistema deberá agregar la funcionalidad al código generado	40	60	L	1.12%	Alta
c#-68	Si el usuario utilizó imágenes, el sistema deberá copiarlas a la carpeta "Assets"	30	80	L	0.84%	Alta
c#-69	El sistema deberá permitir sobrecribir imágenes que tengan el mismo nombre	30	10	L	0.84%	Media
c#-70	El sistema debe revisar los cubos maestros que sean los de inicio	50	80	L	1.39%	Media
c#-71	El sistema debe invalidar otros cubos maestros si son marcados como pantalla inicial	40	60	L	1.12%	Media
c#-72	El sistema deberá mostrar una ventana con el avance de la compilación	20	40	L	0.56%	Media
c#-73	El sistema deberá copiar los archivos de la librería a la carpeta SRC	30	15	L	0.84%	Media
c#-74	El sistema deberá ejecutar los archivos batch de forma independiente	60	50	L	1.67%	Media

c#-75	El sistema deberá esperar a que termine la ejecución de un archivo batch para comenzar con el siguiente	120	70	L	3.35%	Media
c#-76	La ejecución de los archivos batch debe ser transparente al usuario	50	20	L	1.39%	Media
c#-77	El sistema deberá cerrar la ventana de compilación cuando termine el proceso	25	10	L	0.70%	Media
c#-78	Por cada vez que el usuario compile, el sistema deberá cargar un emulador independiente	40	20	L	1.12%	Media
c#-79	El sistema deberá instalar el juego en todos los emuladores y dispositivos Android conectados	50	60	L	1.39%	Media
c#-80	Generar una ventana de inicio que sea transparente	100	120	L	2.79%	Media
c#-81	Generar imagen de fondo con esquinas bordeadas	130	80	L	3.63%	Media
c#-82	El sistema deberá revisar la integridad de los archivos al cargar la pantalla de inicio	80	90	L	2.23%	Media
c#-83	El sistema debe notificar cualquier error que suceda	80	55	L	2.23%	Baja
c#-84	El sistema no deberá tardar mas de 4 segundos en hacer las verificaciones	100	80	L	2.79%	Baja
c#-85	El sistema deberá instalar la aplicación sobre la carpeta "Archivos de Programa"	60	40	L	1.67%	Baja
c#-86	El sistema deberá generar todos los archivos de configuración	20	25	L	0.56%	Baja
c#-87	El sistema deberá generar un acceso directo en el escritorio	40	15	L	1.12%	Media
c#-88	El sistema deberá crear una carpeta en el menú inicio	60	45	L	1.67%	Baja
c#-89	El sistema deberá contener los instaladores de las herramientas	60	45	L	1.67%	Media
c#-90	El sistema deberá otorgar los permisos al sistema para escribir sobre la unidad del Sistema Operativo	200	120	L	5.58%	Baja
		3585	3760		100%	

Porcentaje de Desviación: 4.88%

La técnica es el esfuerzo
Para ahorrar esfuerzo
-- José Ortega y Gasset

Apéndice E. Manual Técnico

E.1 Requerimientos Mínimos

Para el funcionamiento del framework es necesario tener los siguientes requerimientos mínimos en el equipo:

- Windows Vista, Windows 7 o Windows 8
- Microsoft .NET 4.0 Framework
- Tarjeta Gráfica 500Mb o superior
- Memoria RAM 1 Gb o superior
- Procesador Dual Core o superior
- Android SDK con Android 2.1 Eclair
- Java SDK versión 1.6.0 revisión 26

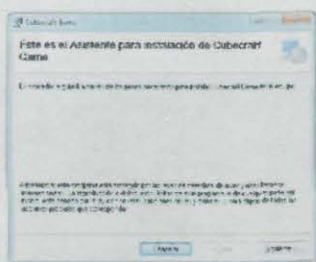
E.2 Instalación del Sistema

E.2.1 Instalación de Cubecraft Game

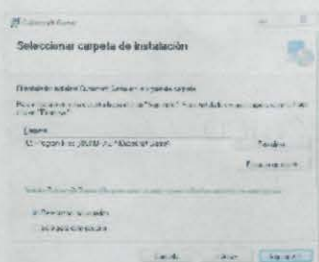
Para instalar correctamente el sistema con todas sus características se deben realizar los siguientes pasos:

1. En la primera pantalla se indica al usuario que el instalador lo guiará por toda la instalación del sistema, a esta ventana debe presionar el botón siguiente.
2. Aquí se le pedirá al usuario indique la ruta donde se desea instalar la aplicación. Por comodidad del usuario, el instalador ya trae una ruta hacia "Archivos de Programas" por default. En esta misma ventana el usuario debe indicar si desea instalar el sistema solo para su usuario o para todos los usuarios. Una vez terminada esta configuración, debe presionar "siguiente".
3. Durante esta tercera ventana el instalador indica al usuario que comenzará la instalación del sistema, el usuario solo debe presionar el botón "siguiente" para comenzar la instalación.
4. Durante la instalación se pedirá permiso para escribir sobre disco. Al otorgar el permiso comenzará la instalación del sistema.

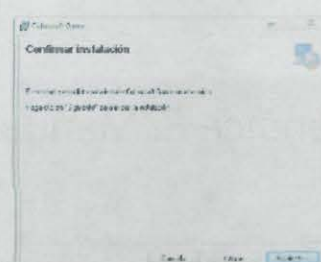
5. Una vez terminada la instalación, se mostrará una pantalla donde se indica si la instalación fue correcta o incorrecta y el usuario debe presionar el botón "Cerrar" para finalizar la instalación.



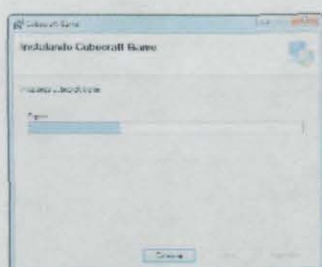
Paso 1



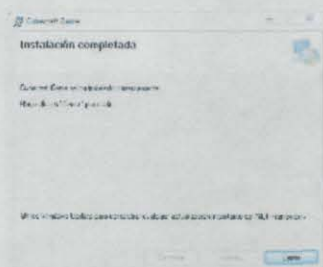
Paso 2



Paso 3



Paso 4



Paso 5

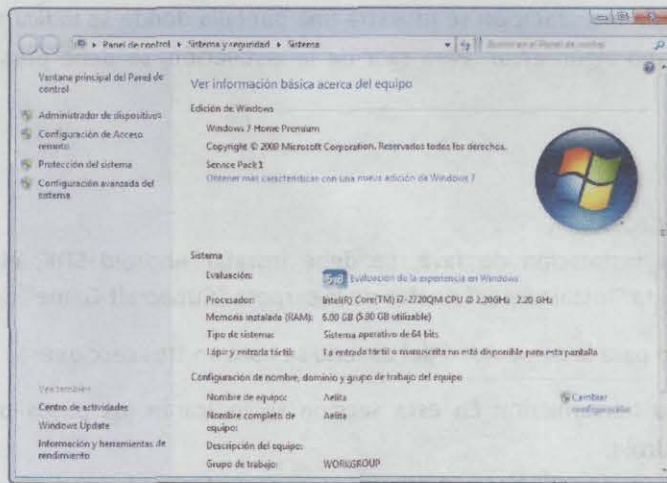
Una vez instalado Cubecraft Game, el sistema generará de forma automática un acceso directo a la aplicación en el escritorio y otro en el menú inicio.

E.2.2 Instalación de Java SDK

Dentro de la carpeta generada en el menú inicio, se encuentra una subcarpeta con el nombre de Instaladores, en ellos se encuentran las herramientas nombradas en los requerimientos mínimos del framework.

Para instalar Java primero debe revisar la arquitectura de su equipo, para hacer esto debe seguir los siguientes pasos:

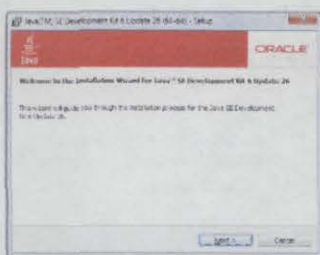
1. Seleccione el icono de Mi PC, de clic derecho sobre él y seleccione la opción Propiedades
2. Se mostrará una ventana que contiene información sobre su Sistema Operativo y datos físicos de su computadora.
3. En el apartado de Sistema hay una sección llamada "Tipo de Sistema", en él se indica la arquitectura que tiene su procesador. Estas opciones pueden ser "Sistema Operativo de 64 bits" o "Sistema Operativo de 32 bits"



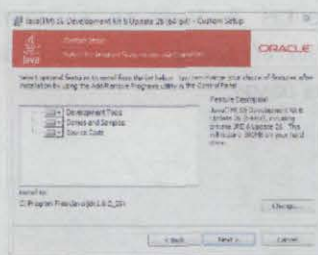
Una vez revisado su arquitectura de procesador, debe ir a la subcarpeta de "Instaladores" dentro de la carpeta de "Cubecraft Game" en su menú de inicio y seleccionar la opción de Java dependiendo de su procesador:

- Java x32 para procesadores con arquitectura de 32 bits
- Java x64 para procesadores con arquitectura de 64 bits

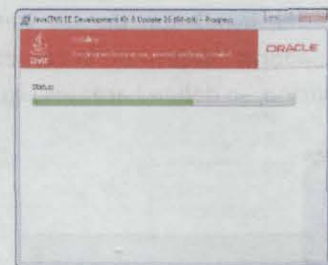
Para instalar cualquiera de las dos versiones de Java se deben seguir los siguientes pasos:



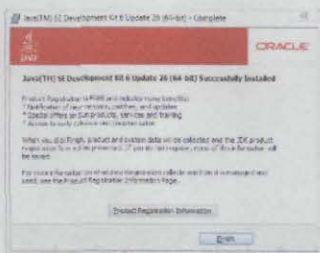
Paso 1



Paso 2



Paso 3



Paso 4

1. La primera pantalla da la bienvenida al usuario a la instalación de Java. En esta pantalla debe dar clic en el botón Siguiente.
2. En la segunda pantalla, se muestra la ruta donde se quiere instalar Java y las herramientas que se quieren que se instalen, para un óptimo funcionamiento de Cubecraft Game se deben instalar todas las características de Java.
3. Ya una vez el usuario haya presionado el botón "Siguiente" de la segunda pantalla se inicia la instalación de la herramienta.

- Una vez terminada la instalación se muestra una pantalla donde se indica si la instalación fue exitosa o hubo algún error. Para salir de la instalación, se debe presionar el botón "Finalizar".

E.2.3 Instalación de Android SDK

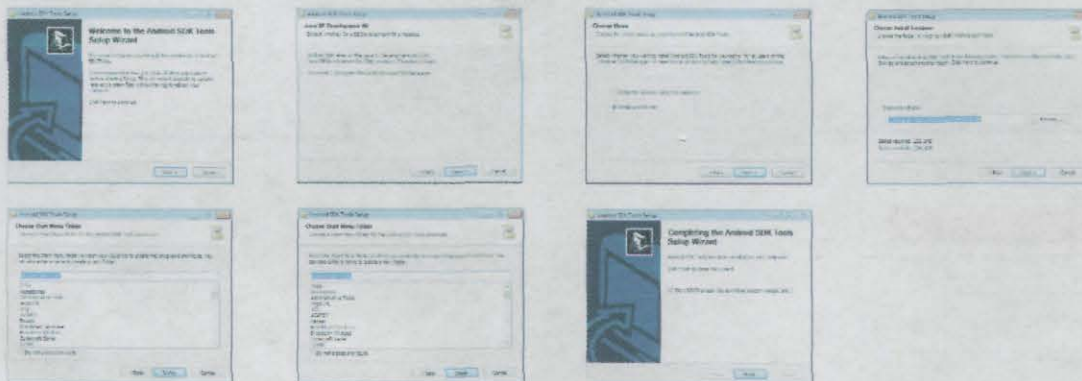
Ya terminada la instalación de Java, se debe instalar Android SDK, el instalador se encuentra en la subcarpeta "Instaladores" dentro de la carpeta "Cubecraft Game" del menú Inicio.

La instalación de Android para la comodidad del usuario se hará en tres secciones:

- Instalación de la herramienta: En esta sección se indicarán los pasos para instalar la herramienta Android.
- Instalación de la versión de Android: Una vez instalado Android, es necesario descargar e instalar la versión de Android con la que trabaja Cubecraft Game, el tiempo que dura esta instalación dependerá de la velocidad de internet con la que cuente el usuario.
- Creación y Primer arranque de emulador: En esta sección se explica la forma de poder crear el emulador que ocupa Cubecraft Game y el primer arranque para su configuración.

E.2.3.1 Instalación de la herramienta

El instalador de las herramientas Android se encuentra localizado en la carpeta "Instaladores" dentro de la carpeta Cubecraft Game del menú inicio. En esta primera parte de la instalación, se deben hacer los siguientes pasos:

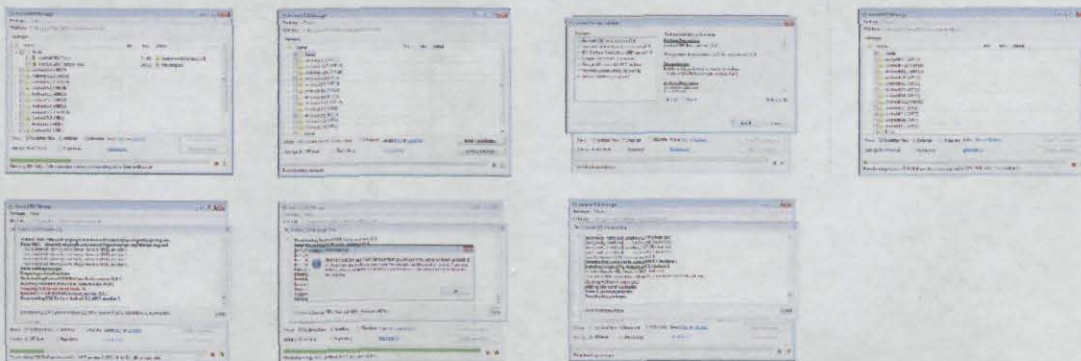


- La primera pantalla da la bienvenida al usuario a la instalación de Android y recomienda que se cierren todos los programas antes de iniciar la instalación, para que no haya algún problema de conflicto con otro programa.
- La siguiente pantalla hace la búsqueda de la versión de Java SDK que se tiene en el equipo, en este punto, no hay ningún problema que no aparezca la versión de Java que se instaló anteriormente.

3. En la tercera pantalla se pide al usuario indique si Android se instalará para todos los usuarios o solo para el usuario actual. La recomendación es que se instale para todos los usuarios del equipo.
4. Una vez que el usuario presiona el botón "Siguiente" de la tercera pantalla, se pedirá que el usuario indique la dirección donde se va a instalar Android. Por default, es instalador indica la ruta en la carpeta Archivos de Programa.
5. Al presionar "Siguiente" pide al usuario si desea que se cree un acceso directo en el menú inicio, al terminar, el usuario debe presionar la opción "Instalar para comenzar la instalación".
6. Una vez terminada la instalación, se muestra una pantalla donde se pide al usuario si desea iniciar el "SDK Manager". En esta pantalla el usuario debe seleccionar la opción y presionar el botón de Finalizar.

E.2.3.2 Instalación de la Versión de Android

Si el usuario seleccionó la opción de Iniciar el "SDK Manager", se mostrará una pantalla donde se listan todas las herramientas Android y las versiones de Android que se encuentran en el mercado. Es necesario que para llevar a cabo esta parte de la instalación, el usuario cuente con una conexión a internet, el tiempo que puede llegar a llevar la instalación puede variar dependiendo de la velocidad de la conexión.



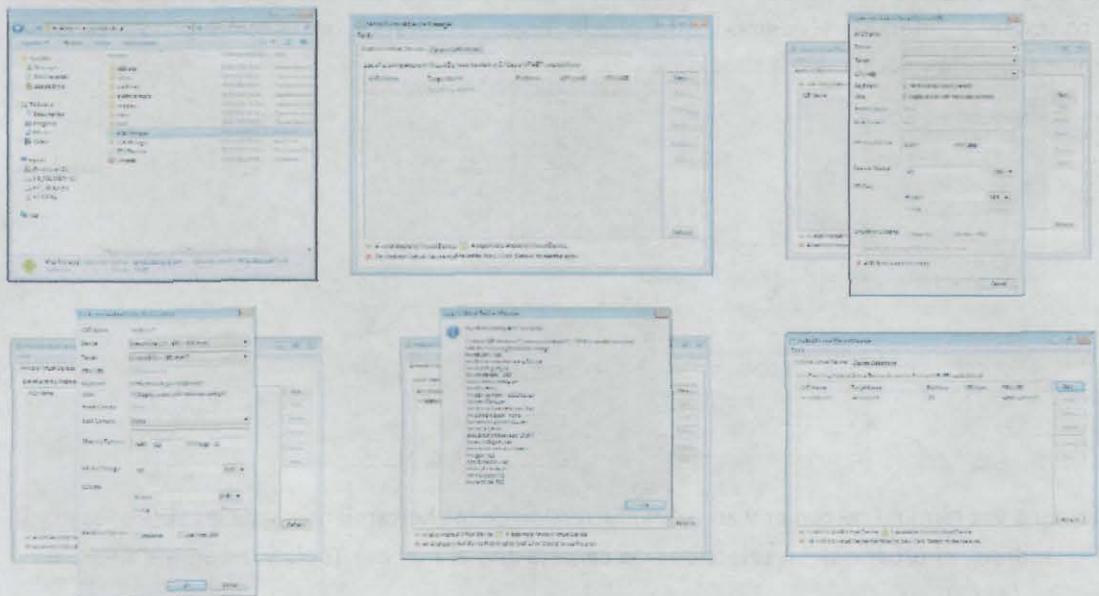
1. Una vez termine de cargar y actualizar la versión de las herramientas y de las versiones de Android, el usuario debe seleccionar la carpeta con el nombre Tools y la carpeta Android 2.1 (API 7)
2. El usuario debe presionar el botón "Install 7 Packages..." y mostrará una ventana donde se pide se acepten los contratos de las versiones de Android. En esta ventana el usuario solo debe presionar la opción "Accept All" y después seleccionar "Install"
3. Al cerrarse esta ventana, comenzará la descarga de las herramientas seleccionadas, se puede seguir el avance de la descarga e instalación en la barra de progreso localizada en la parte inferior de la pantalla.

4. En un punto de la instalación se mostrará un cuadro de dialogo donde se listan las acciones que se están llevando a cabo, aparecerá un error llamado "Stopping ADB Server Failed (code -1)". Este error es común en la primera instalación y debe ignorarse.
5. Una vez terminada la instalación se muestra un cuadro de dialogo donde se indica que se debe reiniciar las herramientas que utilicen Android, este cuadro de dialogo está pensado para otro tipo de herramientas, por lo cual solo debe presionarse "Ok" y continuar con la instalación.
6. Una vez cerrado el cuadro de dialogo, aparecerá en la ventana con el nombre "Android SDK Manager Log" que la instalación fue completa. Se debe verificar que el número de paquetes que se instalaron coincida con el número que decía al inicio de la instalación.

Si todos los datos coinciden entonces el usuario debe cerrar esa ventana y comenzar la configuración del emulador.

E.2.3.3 Creación y Primer arranque del Emulador

El usuario debe abrir un explorador de Windows y dirigirse a la carpeta donde se tiene instalado Android, dentro de esta carpeta se debe encontrar una subcarpeta llamada "Android-sdk", y dentro de esta se encuentran todas las herramientas de Android. El usuario debe abrir la aplicación con nombre AVD Manager y seguir los siguientes pasos:



1. En la pestaña "Android Virtual Devices" se encuentran listados todos los emuladores que se crean. Si es la primera vez que se instala Android, esta lista deberá estar vacía, por lo que el usuario solo debe presionar el botón "New..."
2. Al presionar el botón se muestra un formulario donde el usuario debe indicar los siguientes valores:

- a. AVD Name: telefono2
- b. Device: Nexus One (3.7", 480x800:hdpi)
- c. Target: Android 2.1 – API Level 7
- d. CPU/ABI: ARM (armeabi) [Valor por default]
- e. Keyboard: Hardware keyboard present
- f. Skin: Display a skin with hardware controls
- g. Back Camera: None
- h. Memory Options: RAM: 512 VM Heap: 32
- i. Internal Storage 200 MiB

Y después debe seleccionar Ok.

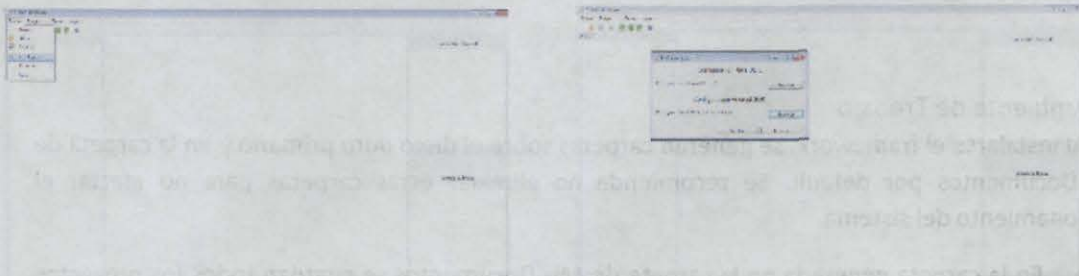
3. Se muestra una ventana donde se indica el resultado de la creación del emulador, si toda la información es correcta, el usuario debe presionar el botón "Ok"
4. En el listado de dispositivos debe aparecer el emulador que se acaba de crear y debe aparecer con un icono verde antes del nombre.
5. El usuario debe seleccionar el emulador y presionar el botón "Start" para iniciar el emulador, si carga correctamente y permite utilizarlo entonces la configuración es completa y pueden cerrarse todas las ventanas.

E.2.4 Configuración del config.ini

Existen dos formas de configurar las rutas hacia la versión de Java y Android instalados. Estos métodos varían debido a los permisos que tenga el usuario en el equipo donde se instaló el sistema.

E.2.4.1 Usuarios con permisos de Administrador

Los usuarios con permisos de administrador son aquellos que pueden escribir directamente sobre cualquier carpeta del sistema operativo, principalmente en Archivos de Programas. Para configurar las rutas de las herramientas se hace lo siguiente:

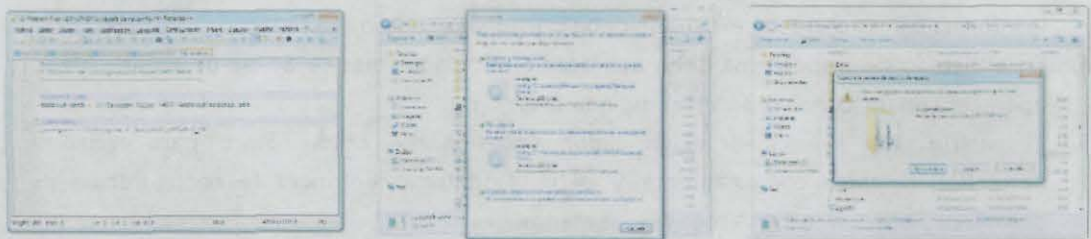


1. Abrir la aplicación Cubecraft Game
2. Seleccionar el menú Inicio y después la opción de Configurar
3. Dentro de la ventana de configuración se indica en cada campo la ruta que necesita cargar.

4. En el campo de Java SDK al presionar el botón de "Examinar" se mostrarán las carpetas que se tienen en el disco duro. Se debe seleccionar la carpeta de Java 1.6.0_26
5. En el campo de Android SDK al igual que el campo anterior, se mostrarán las carpetas que se tienen y se tiene que seleccionar la carpeta donde se instaló Android hasta la carpeta con el nombre de Android-SDK.
6. Una vez que se tengan las rutas en los cuadros de texto, se debe seleccionar la opción de "Aceptar" y se guardarán las rutas de acceso.

E.2.4.2 Usuarios sin permisos de Administrador

Cuando se encuentra el usuario en una maquina compartida o sin todos los permisos, se deben realizar los siguientes pasos para la configuración:



1. Ingrese a la carpeta de instalación de Cubecraft Game. Por default es C:/Archivos de Programa/UPAEP/Cubecraft Game.
2. Copie el archivo con el nombre config.ini a una carpeta a la que tenga acceso, como por ejemplo Mis Documentos.
3. Abra un bloc de notas, seleccione la opción de Abrir y abra el archivo que acaba de copiar.
4. Copie la ruta donde se instaló Java SDK hasta la carpeta con nombre Java 1.6.0_26 y péguela después del signo de igual en la línea que dice java-path.
5. Copie la ruta donde se instaló Android SDK hasta la carpeta Android-SDK y péguela después del signo de igual en la línea Android-path.
6. Guarde el archivo y copie de nuevo el archivo config.ini a la carpeta de Cubecraft Game. Cuando pida la autorización para copiar el archivo y sobrescribirlo seleccione la opción de Continuar y espere.

E.3 Ambiente de Trabajo

Al instalarse el framework, se generan carpetas sobre el disco duro primario y en la carpeta de Mis Documentos por default. Se recomienda no eliminar éstas carpetas para no afectar el funcionamiento del sistema.

- En la carpeta generada en la carpeta de Mis Documentos se guardan todos los proyectos realizados en el sistema, dentro de esta carpeta, el archivo .cub contiene toda la

información sobre el proyecto, incluyendo información del sistema operativo empleado, los cubos con sus respectivas configuraciones, etc.

En la carpeta "src" se encuentran los archivos fuente de la aplicación creada, estos archivos son necesarios para que el compilador permita generar la aplicación. Si uno de estos archivos se encuentra corrupto o modificado fuera del sistema, es probable que la compilación falle.

La carpeta "Entregas" contiene el archivo que instalará la aplicación en el dispositivo móvil. En este caso, los archivos pueden ser .APK para dispositivos Android o .JAR para dispositivos que funcionen bajo J2ME.

En esta primera versión del sistema, la configuración del sistema operativo no puede ser modificada, y por default corresponde a la versión 2.1 Eclair en dispositivos Android y la configuración MIDP 2.0 y CLDC 1.1 para los dispositivos Symbian.

E.4. Configuración de dispositivo móvil para ejecutar la aplicación desde el sistema

Para poder configurar algún dispositivo Android para realizar pruebas directamente desde Cubecraft Game se deben tener las siguientes características:

1. No tener configurado el emulador que utiliza Cubecraft Game: Esto es necesario para que el sistema no intente instalar la aplicación sobre el emulador que utiliza y se mantenga a la espera de un dispositivo para instalar.
2. Tener instalado el software de comunicación del dispositivo: Este software es el que trae por default el dispositivo, normalmente distribuido dentro del dispositivo o en un disco aparte. Este programa de comunicación es utilizado comúnmente para la transferencia de archivos entre la computadora y el dispositivo.
3. Permitir que el dispositivo tenga conexión como Desarrollador: Para configurar el dispositivo como Desarrollador se deben hacer los siguientes pasos:
 - a. Ir al menú de Aplicaciones
 - b. Seleccionar la opción de Ajustes
 - c. Seleccionar la opción Aplicaciones
 - d. Buscar y seleccionar la opción Desarrollador
 - e. Marcar la opción de "Depuración USB" y la opción de "Permanecer activo"

Si la configuración del dispositivo se hizo correctamente, al conectar el dispositivo a la computadora se mostrará en el dispositivo el ícono de un bicho, el cual indica que se encuentra listo para recibir aplicaciones.

Si se desea comprobar que el dispositivo se encuentra listo para pruebas, se puede ejecutar un comando de revisión. Para hacerlo se debe hacer lo siguiente:

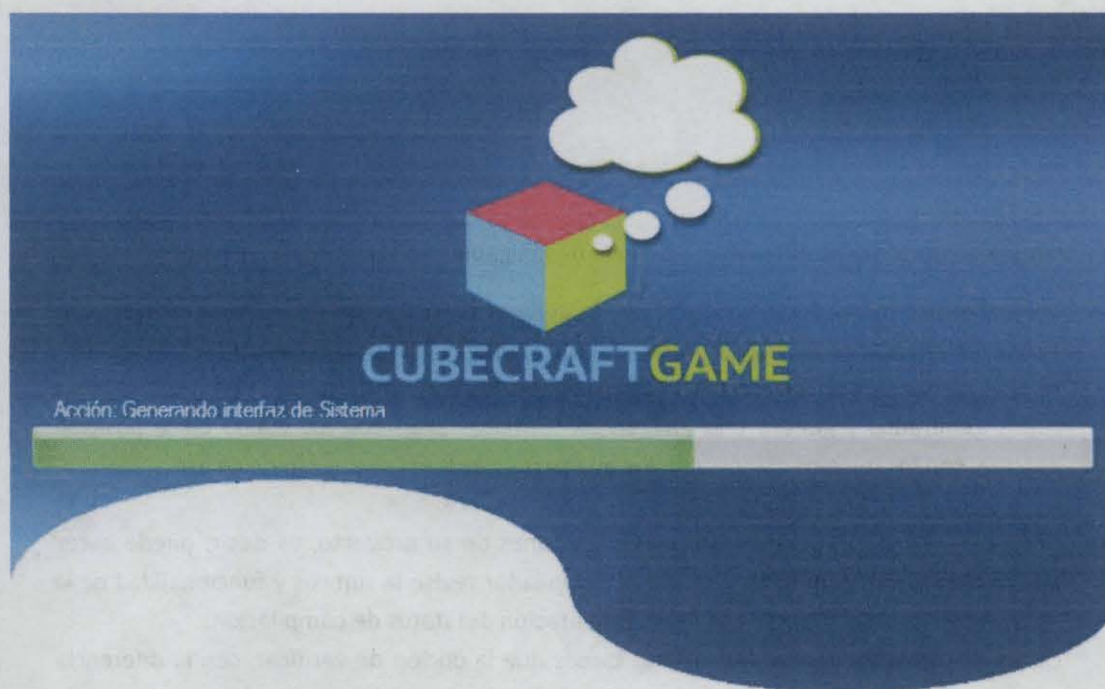
1. Abrir una línea de comandos: Para abrir una línea de comandos, en Windows XP el usuario debe presionar el menú inicio y después la opción de "Ejecutar", y en el cuadro de texto escribir **CMD** y presionar enter; mientras que para la versión Windows Vista o superior, se debe dar clic en el menú inicio y en la barra de búsqueda escribir **CMD** y presionar enter.

- [illegible]

No se puede enseñar nada a un hombre,
Solo se le puede ayudar a encontrar la respuesta
Dentro de sí mismo
-- Galileo Galilei

Apéndice F. Manual de Usuario

F.1. Pantalla de Inicio



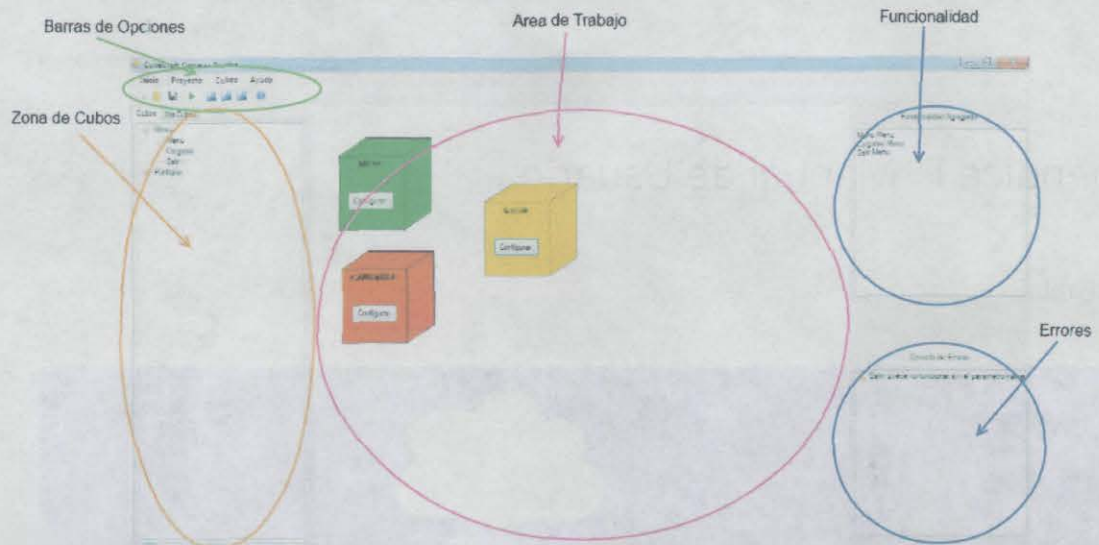
En la pantalla de inicio se hace la configuración del ambiente de trabajo y revisa que ningún elemento del sistema se encuentre lastimado. El tiempo promedio que tarda en hacer la revisión y cargar la interfaz de usuario es de aproximadamente cuatro segundos.

Si alguna de las verificaciones falla, entonces se le informa al usuario y se detiene la ejecución del sistema. Los posibles errores que pueden suceder son:

- Error en archivos: Los archivos que ocupa Cubecraft Game para la funcionalidad o para ejecutar los juegos no se encuentran o están corruptos. La revisión de archivos que buscan son con terminación .bat y .dll.
- No se encuentran las carpetas: Revisa que se encuentren todas las carpetas que ocupa el sistema y que ninguna de ellas se encuentra vacía.
- Error al cargar archivos de interfaz: Los archivos y manifiestos se encuentran corruptos o ausentes, por lo que es necesario reinstalar la aplicación.

F.2. Pantalla Principal

La pantalla principal consta en una ventana dividida en 4 partes que facilita al usuario el uso intuitivo del sistema.



En la barra de opciones se encuentra un menú desplegable con las opciones:

- **Inicio:** En este menú están las opciones de generar un nuevo proyecto, abrir un proyecto existente, guardar el proyecto, importar como imagen y salir del framework. Al seleccionar la opción importar como imagen, el sistema toma la información de los cubos generados y genera una imagen en formato "png" con los cubos en la posición exacta y con los mismos colores con los que se encuentra al momento de su uso.
- **Proyecto:** El usuario puede realizar verificaciones de su proyecto, es decir, puede hacer que el framework genere las clases y el compilador revise la sintaxis y funcionalidad de la aplicación, dándole al usuario la retroalimentación del status de compilación. La opción Compilar realiza las mismas tareas que la opción de verificar, con la diferencia que ejecuta el simulador del dispositivo elegido, instala la aplicación y ejecuta la aplicación.
- **Cubos:** Esta opción está pensada para todos aquellos desarrolladores que quieran utilizar el framework. Al seleccionar la opción de "Crear Cubo" se muestra una pantalla que permite escribir en el lenguaje el código fuente del cubo. La opción "Modificar Cubo" le permite al usuario acceder al código fuente de cierto cubo y modificarlo a su gusto. Mientras que la opción "Eliminar Cubo" como su nombre lo indica, elimina del sistema el cubo, ésta acción no puede ser deshecha.
- **Ayuda:** En este menú se encuentra información sobre el framework y el desarrollador.

En la parte inferior, se encuentra una barra de acceso rápido con las herramientas más utilizadas en el sistema que son:

- Nuevo Proyecto: Esta opción elimina el proyecto actual y genera un nuevo proyecto en blanco.
- Abrir Proyecto: Abre un proyecto válido en el framework, si hay un proyecto abierto, el sistema elimina la información no guardada.
- Guardar Proyecto: Guarda el proyecto actual. Esta opción solo está disponible cuando un cubo se encuentra activo. Para saber si el sistema guardó satisfactoriamente el proyecto, en la barra de título aparece un asterisco (*) junto al nombre del proyecto cuando hay cambios sin guardar.
- Compilar: Este acceso rápido permite a los usuarios compilar y correr su aplicación en el simulador del sistema operativo.
- Generar Cubo: Acceso rápido a la opción "Crear Cubo" del menú Cubos
- Modificar Cubo: Acceso rápido a la opción "Modificar Cubo" del menú Cubos
- Eliminar Cubo: Acceso rápido a la opción "Eliminar Cubo" del menú Cubos
- Ayuda: Accede a la ventana de "Acerca De" del framework.

En la zona de cubos aparecen las funcionalidades que actualmente están disponibles para el framework y el sistema operativo seleccionado. En esta sección, existen dos pestañas indicadas como "Cubos" y "Mis cubos" en las cuales se muestran los cubos por default y los cubos generados por el usuario respectivamente.

La zona de Funcionalidad permite a los usuarios revisar el nombre y tipo de los cubos que ha ingresado al sistema. Esta herramienta está pensada para cuando los proyectos en los que se trabajan alcancen gran tamaño y sea difícil navegar entre todos los cubos.

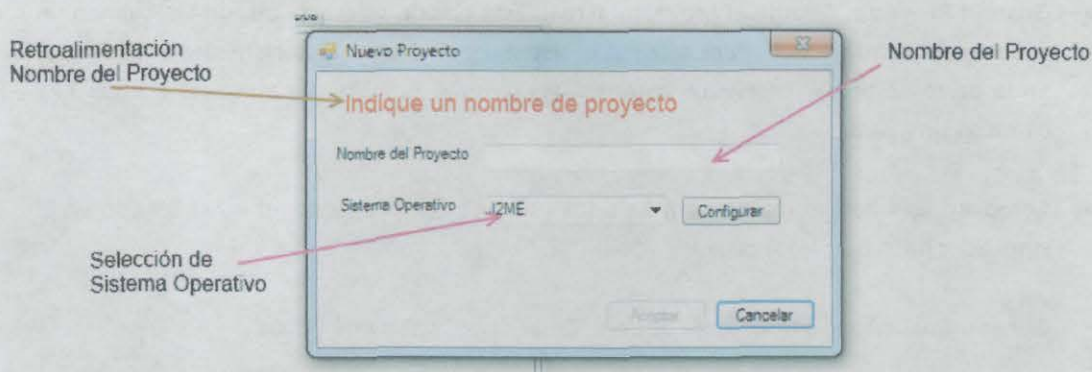
La consola de Errores, muestra los errores de configuración de cada cubo, indicando el tipo de problema que lo causa. Estos errores están divididos en Error y Advertencia, si se marca un error, el framework inhabilita la opción de "Compilar" hasta que se encuentre eliminado, mientras que la advertencia solo indica al usuario que omitir el campo podría causar un malfuncionamiento de la aplicación. Estos errores son calculados y mostrados hasta que el usuario acepte el terminado de modificación de configuración del cubo.

El área de trabajo es la parte con mayor tamaño visual, y es el área donde los cubos son ubicados para ser agregados al proyecto. El área permite que una vez alcanzado el límite de la pantalla, aparezca una barra de desplazamiento para ubicar más cubos en la parte inferior del

área. Los cubos son mostrados en tres diferentes colores: Rojo, Amarillo y Verde. El cubo en color rojo, indica que existe un problema en el cubo o que no se encuentra configurado, los cubos al ingresarse se pone en este color; el cubo en color amarillo indica que las características críticas están cubiertas, pero que existe alguna advertencia en el cubo, mientras que el color verde es el indicador de que el cubo se encuentra completamente configurado y no causará ningún problema dentro de la aplicación desarrollada.

F.3. Generar un Nuevo Proyecto

Cuando se genera un nuevo proyecto, se presenta al usuario la siguiente pantalla:

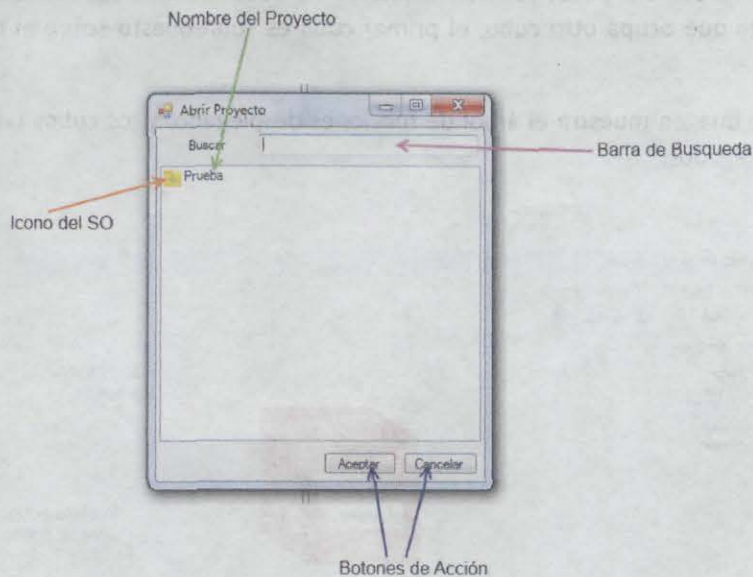


La retroalimentación del nombre de proyecto consiste en dos mensajes posibles dependiendo de la condición si existe o no el nombre del proyecto en la carpeta de espacio de trabajo. En caso de que el nombre exista, se indicará el mensaje "El proyecto ya existe" en letras rojas, mientras que si el nombre del proyecto no existe, se mostrará el mensaje "Todo parece ir bien" en letras verdes, y el botón de aceptar se habilitará.

La lista desplegable indicada como Sistema Operativo contiene el nombre de los sistemas operativos móviles que pueden ser utilizados en este sistema. Al seleccionar uno, permitirá al sistema identificar las funciones que debe cargar en el framework.

En la versión BETA del framework, se encuentra marcada la funcionalidad de "Configurar" la cual permite modificar la información del simulador a utilizar. Esta funcionalidad no será liberada hasta el momento en el que el framework sea liberado como una versión RC (Release Candidate).

F.4. Abrir un proyecto existente



Al presionar el botón de Abrir un Proyecto, se muestra la pantalla mostrada en la parte superior. Esta ventana muestra la información necesaria para abrir un proyecto. El framework revisa la integridad de archivos de cada uno de los proyectos guardados en la carpeta localizada en Mis Documentos; si uno de los proyectos se encuentra corrupto o no cumple con las características del sistema, no se mostrará en la lista. El ícono del SO le da la retroalimentación al usuario del Sistema Operativo que está empleado en el proyecto.

En la parte superior de la pantalla, se encuentra una barra de búsqueda, útil cuando se tienen muchos proyectos guardados y no se recuerda el nombre del proyecto. Esta barra de búsqueda busca los proyectos que contengan en cualquier posición de su nombre la palabra escrita.

Para abrir el proyecto, se puede hacer un doble clic sobre el nombre del proyecto a abrir o seleccionando el proyecto de la lista y presionando el botón "Aceptar".

F.5. Agregar funcionalidad al área de trabajo

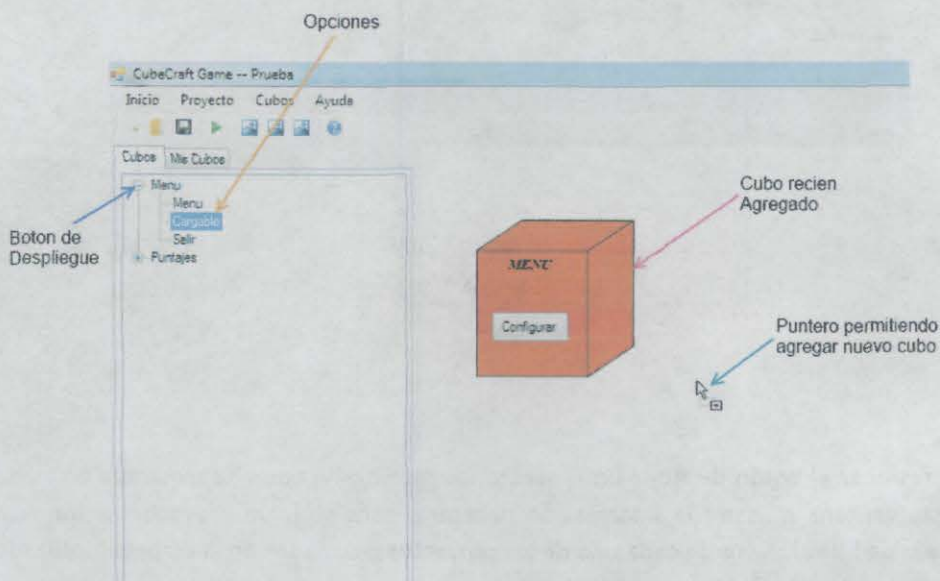
Una vez que se ha creado o abierto un nuevo proyecto, el framework carga según el sistema operativo seleccionado las opciones disponibles de cubos. Los cubos se encuentran organizados según su tipo y permiten mantener un orden de las funciones. Para abrir y cerrar los árboles de funciones, es necesario presionar el ícono '+' o '-' respectivamente.

Al acercarse a una de las opciones, en la parte inferior del puntero de mouse, se muestra una descripción sobre las opciones que están disponibles en esa opción (en caso de ser una opción que despliega otras opciones) y la funcionalidad que representa (cuando es un cubo arrastrable).

En caso de que se arrastre una opción desplegable, el cubo de funcionalidad será tomado como el cubo principal de la funcionalidad.

La posición del cubo dentro del área de trabajo es la posición del mouse en el momento de soltar el clic, y se toma como el punto superior-izquierdo del cubo. Cuando dos cubos son puestos sobre la misma área que ocupa otro cubo, el primer cubo es sobrepuesto sobre el nuevo cubo agregado.

La siguiente imagen muestra el árbol de funciones desplegado, y los cubos tal y como se muestran recién agregados.



F.6. Configurando cubos

Una vez agregado un cubo, se puede configurar dependiendo del tipo de dato que pida en los archivos de configuración. Cada vez que el usuario presiona el botón "Aceptar" se actualizan los datos de los cubos y cambian el color dependiendo si se comete un error crítico o no crítico.

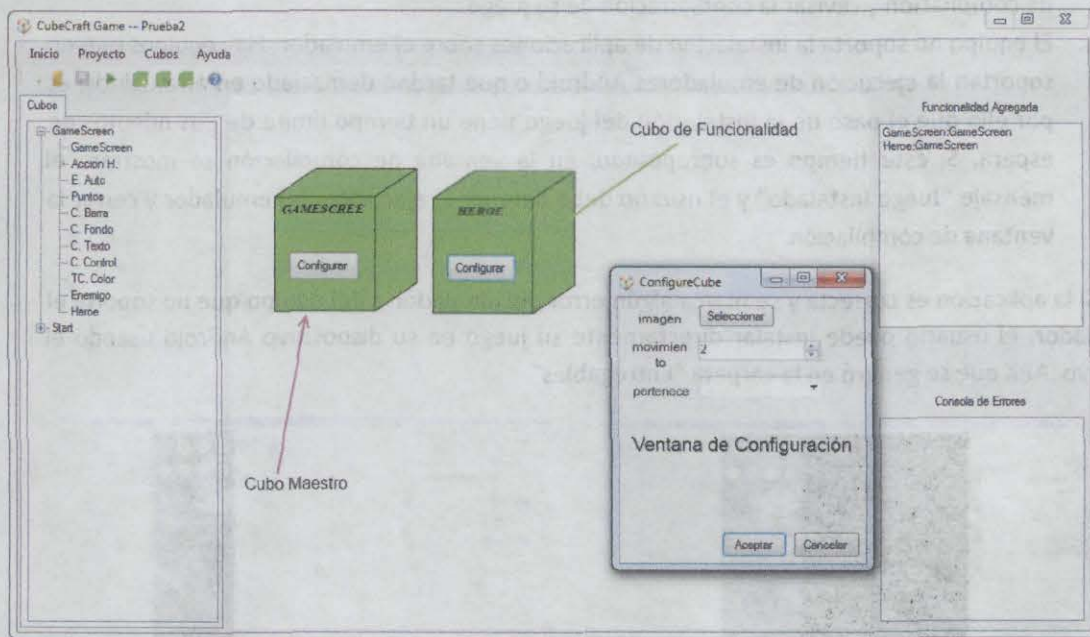
Un error crítico es aquel cuando falta un valor que permite ejecutar el código o que se realice una acción en específico, mientras que los errores no críticos son aquellos errores menores que pueden causar algún conflicto en la ejecución del juego.

Existen dos tipos de cubos dentro del sistema, los cubos maestros y los cubos de funcionalidad. Los cubos maestros son aquellos que comparten nombre con el árbol que los contiene, otra forma de identificarlos es porque solo pide como configuración el nombre que se le asignará.

Los cubos de funcionalidad son aquellos que permiten configurar la forma de actuar de un cubo maestro. Este tipo de cubos tiene un campo en común llamado "Pertenece" el cual permite al usuario seleccionar el cubo maestro al que se le hará la modificación.

Para abrir la ventana de configuración, existen dos métodos: El primero consiste en dar clic sobre el botón con el nombre de "Configurar" de cada uno de los cubos, y el segundo método es dando clic derecho sobre uno de los cubos y seleccionar la opción "Configurar". Existen estos dos métodos debido a que los cubos pueden sobreponerse uno sobre otro y no se pueda acceder al botón de configuración por lo que el segundo método puede ser la solución.

Al dar clic derecho sobre un cubo maestro, se muestra una opción extra en el menú, el cual permite que los usuarios indiquen la pantalla con la que se va a iniciar. El usuario puede indicar al sistema que se quiere que se inicie la aplicación en todos los cubos, por lo cual, se tomará en cuenta el primer cubo maestro que se haya agregado y se volverá el primario.



F.7. Compilando la aplicación

Una vez que el usuario ha terminado de agregar toda la funcionalidad que quiere a los cubos, debe iniciar la compilación y ejecución del juego. Para ello el usuario debe presionar el icono marcado como "Compilar", y el programa genera el ejecutable necesario.

En toda ejecución, el sistema abre un emulador para instalar el juego y mostrárselo al usuario, por lo cual el usuario debe cerrar en todo momento el emulador cuando lo termine de utilizar para evitar un conflicto de simuladores.

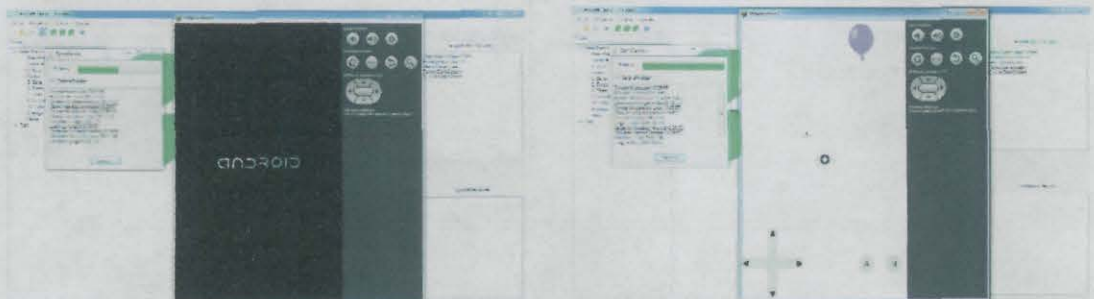
Si la compilación del juego es exitosa, en el simulador se muestra el juego con la funcionalidad que agregó el usuario. Existen varias razones por la cual el juego no puede ser mostrado:

1. El emulador no está instalado: Cuando el emulador no está instalado, entonces el sistema omitirá el paso de iniciar el emulador e instalar el juego. Este error puede eliminarse de diferentes maneras. La primera forma es esperando a que se termine el tiempo de

ejecución; la segunda es conectando algún dispositivo Android que tenga su software de conexión instalado, lo cual hará que el juego sea instalado directamente en el dispositivo y se muestre en él; y la tercera forma es terminando los procesos abd.exe que se generen en el administrador de tareas.

2. El emulador no fue configurado correctamente: Cuando el emulador no está configurado con las versiones que se pidió o con otro nombre diferente al que busca el sistema, el sistema se mantendrá en espera del dispositivo, por lo que el usuario debe finalizar su ejecución de una de las maneras nombradas anteriormente.
3. Existe un error en el código: Cuando existe un error en el código, el sistema intentará instalar la aplicación, pero si se muestra el mensaje de "Juego instalado" en la ventana de compilación y no fue mostrado, entonces el usuario debe cerrar el emulador y la ventana de compilación y revisar la configuración de su juego.
4. El equipo no soporta la instalación de aplicaciones sobre el emulador: Hay equipos que no soportan la ejecución de emuladores Android o que tardan demasiado en arrancarlos, es por ello que el paso de la instalación del juego tiene un tiempo límite de dos minutos de espera. Si este tiempo es sobrepasado, en la ventana de compilación se mostrará el mensaje "Juego instalado" y el usuario debe detener la ejecución del emulador y cerrar la ventana de compilación.

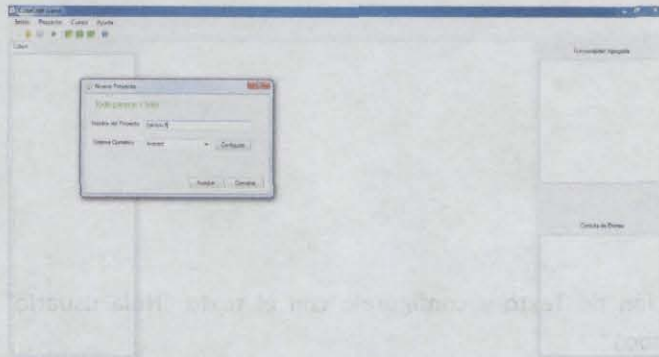
Si la aplicación es correcta y se marca algún error del emulador o del equipo que no soporte el emulador, el usuario puede instalar directamente su juego en su dispositivo Android usando el archivo .APK que se generó en la carpeta "Entregables".



F.8 Ejemplo de Aplicación en Cubecraft Game

Como un ejercicio de adaptación entre el usuario y el sistema, se puede llevar a cabo la siguiente aplicación, donde se espera que al final se pueda obtener un juego donde se tenga que reventar un globo usando el control en pantalla.

1. Generar un nuevo proyecto llamado "Ejemplo1" y seleccionar el Sistema Operativo Android



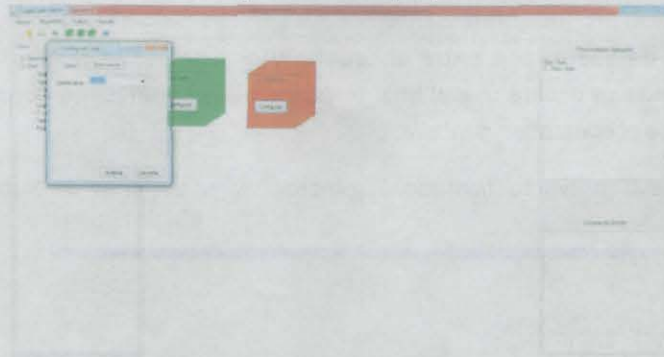
2. Cuando se hayan cargado los cubos de funcionalidad, desplegar la lista de "Start" y se debe seleccionar y arrastrar al área de trabajo el cubo con el nombre "Start".



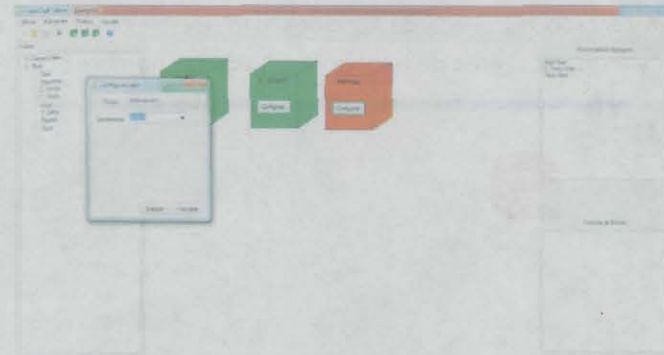
3. Cuando el cubo sea mostrado en el área de trabajo presione "Configurar" y asigne el nombre "Cubo1" al cubo.



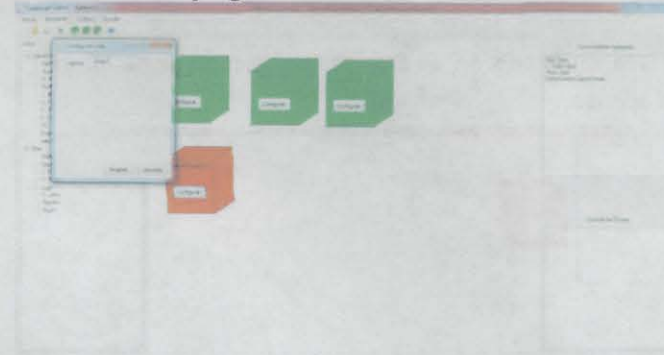
4. Agregue la funcionalidad de "C. Texto" y presione el botón configurar. En la configuración seleccionaremos el color blanco y le diremos que pertenece al cubo "Cubo1".



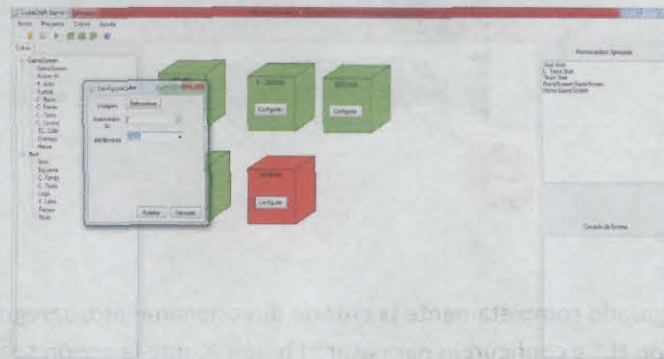
5. Agregue la función de Texto y configúrelo con el texto "Hola usuario" e indique que pertenece al "Cubo1".



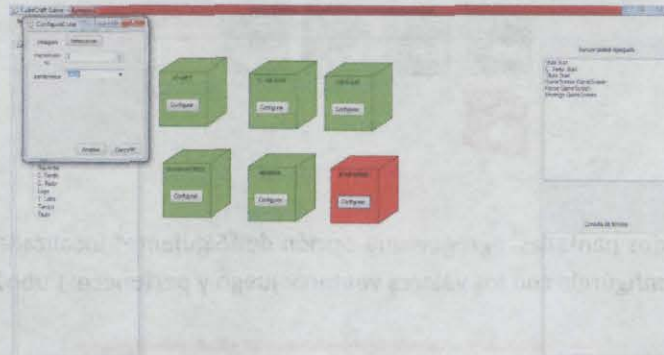
6. Abra las opciones de "GameScreen" y agregue su cubo maestro al área de trabajo y configúrelo con el nombre "juego".



- 5.2.7.2 Agregue la funcionalidad "Heroe" y en la configuración seleccione la imagen que quiera y que tenga formato PNG y ponga el movimiento en 2 e indique que le pertenece al cubo "juego".



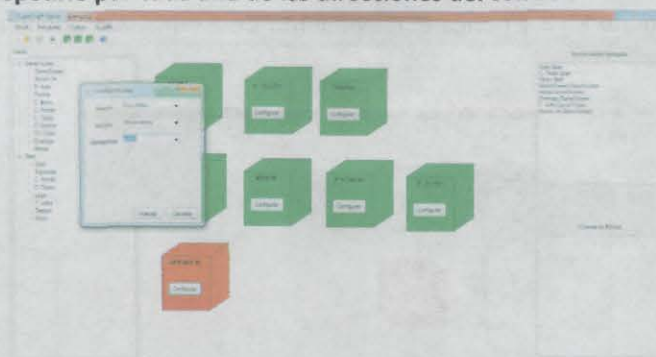
8. Agregue la funcionalidad "Enemigo" y configúrelo con una imagen de su agrado y en movimiento ponga 2 e indique que le pertenece al cubo "juego".



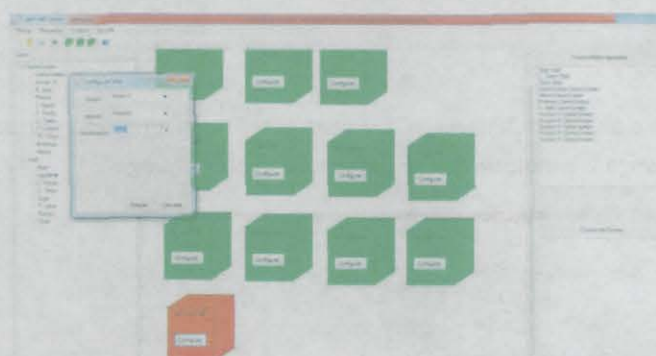
9. Para hacer que el enemigo se mueva solito, arrastre el cubo "E. Auto" y marque la opción "Automatico", seleccione el movimiento "Izquierda a derecha" y diga que pertenece al cubo "juego".



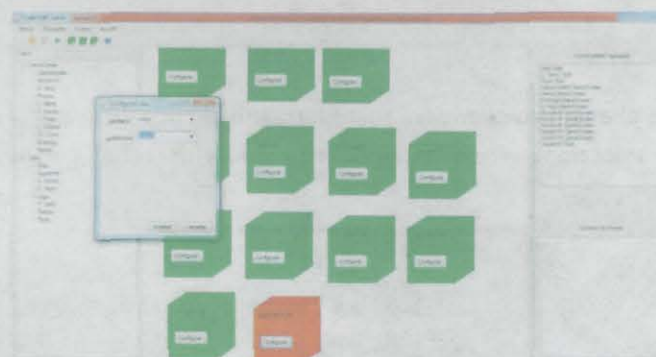
10. Arrastre el cubo que dice "Accion H." y configúrela para que se utilice la flecha hacia arriba, que el movimiento sea "Mover hacia arriba" y que le pertenezca al cubo "juego". Esto debe repetirlo por cada una de las direcciones del control.



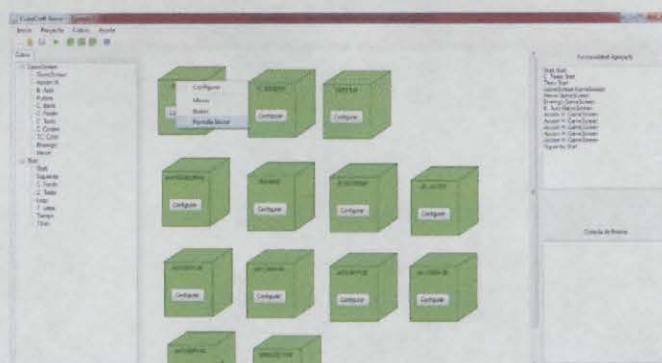
11. Una vez configurado completamente la cruz de direccionamiento, agregue una vez más un cubo de "Acción H." y configúrelo para usar el botón A, que la acción sea "Disparar" y que pertenezca al cubo "juego"



12. Para ligar las dos pantallas, agregue una opción de "Siguiente" localizada en las funciones de "Start" y configúrela con los valores ventana: juego y pertenece: Cubo1



13. De clic derecho sobre el cubo maestro llamado Cubo1 y seleccione la opción de "Pantalla inicial"



14. Compile el juego y compruebe que haga todos los movimientos que se marcaron.



El saber es la única propiedad
Que no puede perderse
-- Bías de Priene

Apéndice G. Derechos de Autor

Todas las herramientas utilizadas para el desarrollo del sistema se encuentran por licencia GLP por lo que el usuario tiene la obligación moral de nombrar la herramienta y la empresa desarrolladora.

- Visual Studio 2012 Ultimate de Microsoft. Licencia pagada vigente hasta 2014
- Java 1.6.0 revisión 26 de Oracle Inc.
- Android SDK con todas sus imágenes de Sistema Operativo de Android Open Source Project, propiedad de Google Inc.

El logo de Cubecraft Game es propiedad intelectual de LuminiSoft, al igual que el framework desarrollado.

Se prohíbe toda reproducción parcial o total sin conocimiento por escrito de la empresa desarrolladora.

www.luminisoft.com.mx

Lumin1Soft



LuminiSoft 2013

Referencias

- Acetenza, N., Coppes, A., Mesa, G. V., Fernandez, E., Lorenzo, T., & Vallespir, D. (2009). *Una metodología para desarrollo de Videojuegos: Versión Extendida*. Facultad de Ingeniería de la Universidad de la Republica de Montevideo.
- AFP. (13 de 09 de 2012). México casi duplica celulares y usuarios con internet en 6 años. *Zócalo Saltillo*.
- Albornoz, R. (2 de Octubre de 2006). *¿Cómo empezar en el Desarrollo de Videojuegos?* Obtenido de LosersJuegos: http://www.losersjuegos.com.ar/_media/referencia/articulos/como_empezar/como_empezar.pdf
- Aponte, S., & Dávila, C. (2011). *Sistemas Operativos Móviles: Funcionalidades, Efectividad y Aplicaciones útiles en Colombia*. Universidad EAN.
- Belli, S., & Lopez, C. (2008). *Breve Historia de los Videojuegos*. Athenea Digital.
- Bondon, P. A., & Felmer, L. R. (Diciembre de 2008). Usuarios habituales de videojuegos: Una aproximación inicial. *SciELO*.
- Carnegie Mellon. (2012). *Software Engineering Institute*.
- Conway, M. (1997). *Alice: Easy-to-learn 3D Scripting for Novices*. Virginia: University of Virginia Publications.
- Fakhreddine, K., Alemzadeth, M., Saleh, J., & Arab, M. (s.f.). *Human-Computer Interaction: Overview on State of the Art*. Department of Electrical and Computer Engineering.
- Felix, J. (2007). *Dispositivos Móviles*. Obtenido de Universidad Nacional del Nordeste Argentina: <http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/tfbasterretche.pdf>
- Ferré, X. (2010). *Principios Básicos de la Usabilidad para Ingenieros de Software*. Facultad de Informatica de la Universidad Politécnica de Madrid.
- García, F. (2005). *Videojuegos: Un análisis desde el punto de vista educativo*.
- Gil, A., & Vida, T. (2007). *Los Videojuegos*. Barcelona: UOC de Barcelona.
- González, J. A. (2007). *El lenguaje de programación C#*. Programacion en Castellano.
- Gutierrez, J. (2006). *¿Qué es un framework web?*
- Herrera, E., & Valencia, L. E. (2007). Del manifiesto ágil: Sus valores y principios. *Scientia et Technica*(34).
- Knowledge Bank: Next Generation Research. (2010). *The impact of Web 2.0 Technologies in the Classroom*. Sydney, Australia: Research and Innovation Department of Education and Early Childhood Development.

- Kolling, M. (2010). *The Greenfoot Programming Environment*. Canterbury, Kent: University of Kent Publications.
- Larman, C. (2004). *Agile and Iterative Development: A Manager's guide*. Addison-Wesley Professional.
- Lavilla, F. (2005). *Los videojuegos y los niños*.
- Maloney, J., Resnick, M., Rusk, B., & Eastmond, E. (2010). *The Scratch Programming Language and Environment*. Massachusetts: Massachusetts Institute of Technology.
- Maxxes System. (2010). *What is a Software Framework?*
- Molina, E., Maldonado, C., & Leidy, V. (2010). *La Telefonía Celular*. Corporación Instituto de Administración y Finanzas.
- Morales, G., Nava, C., Fernández, L., & Rey, M. (2010). *Proceso de Desarrollo de Videojuegos*. Instituto de Ingeniería y Tecnología de la Universidad Autónoma de Ciudad Juárez.
- Otter. (2008). *The Movie Industry Vs. The Gaming Industry*. Obtenido de Yahoo Voices: http://www.associatedcontent.com/article/1015720/the_movie_industry_vs_the_gaming_industin.html
- Palacios, J. (2004). *EL modelo Scrum*. Obtenido de Navegapolis: http://www.navegapolis.net/files/s/NST-010_01.pdf
- Pertegas, S., & Pita, S. (2001). *La Distribución Normal*. Unidad de Epidemiología Clínica y Bioestadística.
- Pleguezuelos, P. (2007). Creación de Juegos con LUA. *Solo Programadores*(154).
- Rodríguez, E. (2005). *Jóvenes y videojuegos: Espacio, significacion y conflictos*. Fundación de ayuda contra la drogadicción.
- Rogers, R. (2011). *Learning Android Game Programming: A hands-on Guide to Building Your First Android Game*. Addison-Wesley Professional.
- Sacristán, P. (2010). *Cuentos para dormir*.
- Taráguila, C. (2009). *Dispositivos móviles y multimedia*. Mosaic: Tecnologías y Comunicación Multimedia.
- Tzancogg, C., Rodríguez, A., & Valeria, S. (2008). *Usando Jmeter para pruebas de rendimiento*. La Plata, Argentina.
- Valencia, R., Riascos, V., & Niño, M. (2011). *Método para la creación de micromundos inmersivos*. Departamento de Sistemas de la Universidad de Cauca.