Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE

Juan Daniel Pineda Cabrera
Asesorado por la Inga. Mirna Ivonne Aldana Larrazabal

Guatemala, octubre de 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JUAN DANIEL PINEDA CABRERA
ASESORADO POR LA INGA. MIRNA IVONNE ALDANA LARRAZABAL

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2015

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Raúl Eduardo Ticún Córdova
VOCAL V	Br. Henry Fernando Duarte García
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

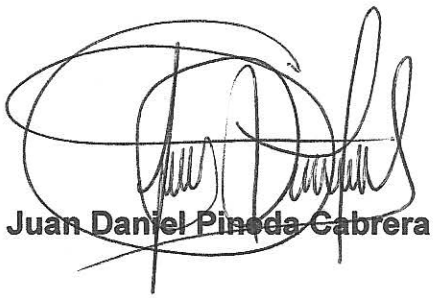
DECANO	Ing. Angel Roberto Sic García
EXAMINADORA	Inga. Floriza Felipa Ávila Pesquera de Medinilla
EXAMINADOR	Ing. Marlon Antonio Pérez Türk
EXAMINADORA	Inga. Susan Verónica Gudiel Herrera
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha agosto de 2014.



Juan Daniel Pineda Cabrera

Guatemala, 09 de mayo de 2015

Silvio José Rodríguez Serrano
Director EPS
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero Silvio José Rodríguez Serrano:

Por este medio atentamente le informó que el estudiante universitario **Juan Daniel Pineda Cabrera** de la Carrera de Ingeniería en Ciencias y Sistemas, con **carné No. 200611402,** ha finalizado y presentado el informe final del proyecto de EPS, cuyo título es **"PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUATIVOS PARA LA ASOCIACIÓN CIVIL *EDULIBRE*"**, *el cual he tenido la oportunidad de revisar, por lo cual doy por aprobado el mismo.*

Agradeciéndole la atención a la presente y quedo a sus órdenes para cualquier información adicional.

Mirna Ivonne Aldana Larrazabal
INGENIERA EN CIENCIAS Y SISTEMAS
Colegiada No. 9567

Inga. Mirna Ivonne Aldana Larrazábal de Ramírez
Ingeniera en Ciencias y Sistemas
Colegiado No. 9567
Asesora

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

UNIDAD DE EPS

Guatemala, 14 de mayo de 2015.
REF.EPS.DOC.367.05.2015.

Ing. Silvio José Rodríguez Serrano
Director Unidad de EPS
Facultad de Ingeniería
Presente

Estimado Ingeniero Rodríguez Serrano .

Por este medio atentamente le informo que como Supervisora de la Práctica del Ejercicio Profesional Supervisado, (E.P.S) del estudiante universitario de la Carrera de Ingeniería en Ciencias y Sistemas, **Juan Daniel Pineda Cabrera** carné No. **200611402** procedí a revisar el informe final, cuyo título es **PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE.**

En tal virtud, **LO DOY POR APROBADO**, solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,

"Id y Enseñad a Todos"

Universidad de San Carlos de Guatemala
COORDINADOR(A) EPS
ÁREA TECNOLOGÍA Y ENERGÍA
Unidad de Prácticas de Ingeniería y EPS

Inga. Floriza Felipa Avila Pesquera de Medinilla
Supervisora de EPS
Área de Ingeniería en Ciencias y Sistemas

FFAPdM/RA

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

UNIDAD DE EPS

Guatemala, 14 de mayo de 2015.
REF.EPS.D.233.05.2015.

Ing. Marlon Antonio Pérez Turk
Director Escuela de Ingeniería Ciencias y Sistemas
Facultad de Ingeniería
Presente

Estimado Ingeniero Perez Turk.

Por este medio atentamente le envío el informe final correspondiente a la práctica del Ejercicio Profesional Supervisado, (E.P.S) titulado **PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE,** que fue desarrollado por el estudiante universitario **Juan Daniel Pineda Cabrera carné No. 200611402,** quien fue debidamente asesorado por la Inga. Mirna Ivonne Aldana Larrazabal y supervisado por la Inga. Floriza Felipa Ávila Pesquera de Medinilla.

Por lo que habiendo cumplido con los objetivos y requisitos de ley del referido trabajo y existiendo la aprobación del mismo por parte de la Asesora y la Supervisora de EPS, en mi calidad de Director apruebo su contenido solicitándole darle el trámite respectivo.

Sin otro particular, me es grato suscribirme.

Atentamente,
"Id y Enseñad a Todos"

Ing. Silvio José Rodríguez Serrano
Director Unidad de EPS

Universidad de San Carlos de Guatemala
DIRECCIÓN
Unidad de Prácticas de Ingeniería y EPS
Facultad de Ingeniería

SJRS/ra

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 10 de Junio de 2015

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación-EPS del estudiante **JUAN DANIEL PINEDA CABRERA,** carné **200611402,** titulado: **"PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE",*** *realizado por el estudiante JUAN DANIEL PINEDA CABRERA, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pérez Türk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 09 de Octubre de 2015

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.530.2015

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **PROPUESTA DE UN MARCO DE TRABAJO PARA EL DESARROLLO DE VIDEOJUEGOS EDUCATIVOS PARA LA ASOCIACIÓN CIVIL EDULIBRE,** presentado por el estudiante universitario: **Juan Daniel Pineda Cabrera,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

USAC SAN CARLOS DE GUATEMALA DECANO FACULTAD DE INGENIERIA

Guatemala, octubre de 2015

/cc

ACTO QUE DEDICO A:

Mis padres	Rigoberto Pineda Godoy y Guadalupe Noemi Cabrera Sánchez de Pineda, por su apoyo incondicional en el transcurso de mi carrera.
Mis amigos	Por acompañarme y compartir conmigo sus conocimientos.
Mi asesora	Inga. Ivonne Aldana, por creer en este trabajo y por brindarme su tiempo y apoyo.

AGRADECIMIENTOS A:

Dios	Por su guía y todas las bendiciones recibidas hasta alcanzar este triunfo.
Mi familia	Por haberme dado todo su apoyo y los ánimos para seguir y alcanzar mi meta.
Asesora	Inga. Ivonne Aldana, por haberme dedicado parte de su tiempo, brindarme su apoyo y asesoría duran el transcurso del EPS.
Edulibre	Por haberme dado la oportunidad de realizar mi trabajo de EPS en la institución y aportar algo en apoyo a la educación del país.
Universidad de San Carlos de Guatemala	Por permitirme ser parte de ella y brindarme la oportunidad de superarme y alcanzar mi meta.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
GB	Gigabyte
GHz	Gigahertz
GUI	Interfaz gráfica
HTML	Lenguaje de marcas de hipertexto
MB	Megabyte
MHz	Megaherz

GLOSARIO

Assets	Se le llama así al conjunto de elementos que se utilizan para la realización de un videojuego como imagenes, sonido, texturas, entre otros.
Branches	*Branches* o ramas, son herramientas proporcionadas por los sistemas de control de versiones para poder trabajar paralelamente en distintas versiones del código sin interferir entre sí.
Builds	En el contexto de programación se le llama si a una versión del software. Como regla general suele ser una versión prelanzamiento.
Commits	Son puntos de guardado que se van realizando sobre las ramas en los que fijan los cambios realizados en el código.
Framework	En informática es a una estructura conceptual de soporte definido para módulos de software.
Game Engine	Es software especializado para la creación de videojuegos, conteniendo todas las herramientas necesarias para su elaboración.

Game	En la metodología Scrum se refiere a la fase de desarrollo del software.
Gameplay	En el ámbito de los videojuegos se refiere al modo en que el usuario interactuará con el videojuego.
Java	Es un lenguaje de programación independiente del sistema operativo que pertenece a Oracle.
La nube	Es un término muy empleado para referirse a algo que está en internet, comúnmente servicios.
Merge	Se le dice así a la acción de juntar una rama sobre otro y unificar cambios. Por ejemplo para añadir funcionalidad al proyecto principal.
Metodología	Hace referencia a una serie de procedimientos racionales utilizados para alcanzar una gama de objetivos.
Plataforma	Se le llama así a un sistema que sirve como base para hacer funcionar determinados módulos de hardware o software.
Post-game	En la metodología Scrum se refiere a la fase final del desarrollo del software.
Postmortem	Hace referencia al después de la finalización de un proyecto.

Pre-game	En la metodología Scrum se refiere a la planificación en general del software desarrollar.
Pull/Push	Son los procesos para sincronizar el estado de los repositorios locales con el servidor de código principal. Tanto para subir cambios (Push) como para descargar las últimas actualizaciones (Pull).
Release	En el contexto de programación se le llama así a una versión estable del software.
Repositorio	En el área de informática puede verse como un almacén donde se guardan ciertas cosas, especialmente códigos de programas.
Sprints	En la metodología Scrum se refiere a los períodos de trabajo establecidos para el desarrollo del software.
Sprite Sheet	Son imágenes que contienen otras imágenes, comúnmente contienen una secuencia de movimientos de algún elemento del videojuego.
Tag	Son etiquetas que se utilizan para definir un número de versión al software en el repositorio.

RESUMEN

El presente informe de graduación describe el proyecto de creación de una propuesta de un marco de trabajo para el desarrollo de videojuegos educativos para la Asociación Civil Edulibre.

Edulibre es una institución sin ánimo de lucro que busca llevar una mejor educación a niños y niñas de primaria a través del uso de tecnología en el aula, realizando para este propósito la puesta en marcha de laboratorios de computación en diferentes escuelas públicas del país utilizando únicamente software libre.

La propuesta del marco de trabajo realizada para la institución busca ayudar a Edulibre a agilizar el proceso de desarrollo de videojuegos educativos desde su concepción hasta su lanzamiento. Esto permitirá a Edulibre ampliar la biblioteca de programas de su sistema operativo insignia EdulibreOS.

El marco de trabajo abarca.

- La propuesta de una metodología enfocada a al desarrollo de videojuegos educativos.
- El estudio de varias herramientas de desarrollo de videojuegos y la elección de la más idónea para la institución.
- La definición de la administración de la configuración.
- El estudio de varias herramientas de control de versiones en la nube que utilicen *git* y la elección de la más idónea para la institución.

OBJETIVOS

General

Proveer a la Asociación Civil Edulibre de un marco de trabajo para estandarizar el proceso de desarrollo de videojuegos educativos, de manera que sirva de guía para estudiantes que busquen realizar su EPS desarrollando videojuegos para esta institución.

Específicos

1. Definir una metodología de desarrollo de videojuegos educativos, a partir del estudio de las principales metodologías agiles.

2. Definir los roles principales de equipo de desarrollo que intervendrán en la creación del videojuego educativo.

3. Proponer documentación de apoyo, útil para el desarrollo de los videojuegos educativos.

4. Evaluar al menos tres herramientas de desarrollo de videojuegos en dos dimensiones que sean de código abierto y con soporte para PC con GNU/Linux, Web y dispositivos móviles con Android.

5. Establecer la administración de la configuración que incluya la evaluación de tres diferentes opciones de software de control de versiones en línea, para el mantenimiento del código de los videojuegos educativos que

involucren una comunidad colaborativa y que soporten el código de las herramientas.

INTRODUCCIÓN

La Asociación Civil Edulibre es una institución sin fines de lucro que posee varios laboratorios de computación en diversas escuelas públicas del país. Lleva la educación a muchos niños mediante el uso de su sistema operativo EdulibreOS basada la popular distribución GNU/Linux Ubuntu, el cual cuenta con muchos programas y videojuegos educativos que permiten a los niños aumentar sus conocimientos. Es importante mencionar que esta distribución es meramente guatemalteca, y que al igual que el sistema operativo en que está basado, cuenta con una licencia general pública, para que cualquier persona pueda tomarlo y modificarlo si así lo desea.

El presente trabajo consiste en la propuesta de un marco de trabajo por el cual Edulibre pueda desarrollar videojuegos educativos, que estarán enfocados principalmente a niños de educación primaria. El marco de trabajo constará de una metodología de desarrollo propuesta específicamente para videojuegos educativos, además de sugerir herramientas para facilitar su desarrollo, y control de versiones. También contará con un documento de administración de la configuración para facilitar el manejo y propuesta de versiones.

1. MARCO TEÓRICO

1.1. Videojuego educativo

Es software que tiene como objetivo proporcionar al usuario algún tipo de conocimiento de uno o varios temas. Son videojuegos que enseñan mientras divierten y entretienen a los jugadores. El conocimiento en este tipo de software se adquiere de forma implícita, es decir que las personas inmersas en el videojuego, en primera instancia no se percatan de estar recibiendo una serie de conocimientos, sino que se van apropiando de ellos en el transcurso natural del videojuego.

1.2. Metodología de desarrollo de software

Consiste en un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de sistemas de software.

1.3. Game Engine

También conocido como motor de videojuego. Cuando se habla de un Game Engine se hace referencia a un software que contiene una interfaz gráfica, junto con varias herramientas que sirven para programar el diseño, la creación y representación de un videojuego con todas sus características, ya sea bidimensional o tridimensional.

1.4. Framework

En desarrollo de software se refiere a una librería o conjunto de ellas que contiene una serie de rutinas útiles que facilitan una o más tareas de desarrollo específicas. Están hechas para diferentes lenguajes de programación.

1.5. Administración de la configuración

Consiste en identificar, organizar y controlar los cambios que sufre el software durante su desarrollo y actualizaciones. Ayuda a aumentar la productividad y disminuir errores. Al controlar los cambios, el testeo que hay que realizar es menor y el software es menos costoso.

1.6. Software de control de versiones

Son aplicaciones ideadas para administrar ágilmente los cambios en el código fuente de programas y revertirlos. Comúnmente estos funcionan a través de un cliente que baja de un servidor el código fuente del programa. A través de este cliente se suben los cambios al servidor. Ejemplos de este tipo de software son subversion, git, entre otros. Aunque ahora está de moda el usar repositorios en la nube los cuales funcionan bastante bien para albergar de manera más segura un proyecto, un ejemplo de este tipo de repositorios sería GitHub, donde de hecho se encuentran muchos proyectos de software libre.

2. FASE DE INVESTIGACIÓN

2.1. Antecedentes de la empresa

A continuación se presentan las características más relevantes de la institución donde fue realizado el EPS.

2.1.1. Reseña histórica

Edulibre es una institución sin fines de lucro que nació como una iniciativa por parte de estudiantes de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala, que viendo la problemática de la educación del país, decidieron aportar para su mejora el uso de tecnología. Básicamente la idea fue tomar varias soluciones de tecnologías y transformarlas en laboratorios de computación de bajo costo y que utilizarán software libre. Su fin es ayudar a complementar la educación de los niños y niñas de educación primaria del país.

El proyecto se fundamenta en la distribución de un sistema operativo GNU/Linux basado en Ubuntu llamado EdulibreOS, el cual está orientado a ofrecerse como una solución para educar con tecnología actualizada a los niños, niñas y jóvenes que no tienen acceso a una computadora. El sistema operativo es desarrollado por Edulibre y apoyado por la empresa Xumak.

2.1.2. Misión

"Brindarles la oportunidad a las niñas y niños de Latino América de tener acceso a una Educación de calidad a través de tecnologías de información, guidados por los principios de código abierto"[1].

2.1.3. Visión

"Desarrollamos soluciones de tecnología de información de código abierto y asesoramos en su uso en las escuelas de nivel primario, integrándolas a su práctica pedagógica"[2].

2.2. Descripción de las necesidades

Para tener más claras las necesidades y el alcance del proyecto se tuvo varias reuniones con el director general de la Asociación Civil Edulibre quien planteó las siguientes necesidades.

- Una metodología de desarrollo de videojuegos educativos que contemple todos los procesos necesarios para llevarlo a cabo.
- La evaluación de herramientas que faciliten el desarrollo de los videojuegos educativos.
- Proponer el manejo del versionamiento de los videojuegos educativos por medio de una propuesta de administración de la configuración.
- La evaluación de herramientas que permitan el control de versiones que sean en la nube y soporten las herramientas de desarrollo de videojuegos.

[1] Edulibre. www.edulibre.net. Consulta: 5 de septiembre de 2014.
[2] Ibíd.

2.2.1. Necesidades identificadas

- La metodología debe tomar en cuenta que lo que se quiere crear son videojuegos educativos.
- La metodología debe ser fácil de adoptar, seguir y mejorar.
- Se debe tomar en cuenta los roles que deben intervenir en el equipo que diseñará y creará el videojuego.
- Las herramientas para el desarrollo de videojuegos a evaluar deben exportar el videojuego a diferentes plataformas, especialmente PCs con GNU/Linux, Web y dispositivos móviles con Android.
- Las herramientas a proponer deben ser de código abierto, y con una licencia que permita su modificación y adaptación.
- Las herramientas que se propongan para el control de versiones deben ser gratuitas y permitir repositorios privados.
- Un adecuado manejo del control de versiones de los videojuegos educativos por medio de una propuesta de administración de la configuración.

3. FASE TÉCNICO PROFESIONAL

3.1. Descripción del proyecto

El trabajo que se realizó consistió en una propuesta de un marco de trabajo para el desarrollo de videojuegos educativos, que incluye una metodología para desarrollo, un estudio de herramientas a utilizar, y una adecuada administración de la configuración para el manejo de versiones. Esto para que Edulibre tenga las herramientas necesarias para desarrollar los videojuegos educativos que planea realizar, de una manera ordenada para ser implantados en su plataforma de aprendizaje.

El marco de trabajo contiene una metodología de desarrollo propuesta con especial énfasis en el desarrollo de videojuegos educativos, proveyendo los pasos que serán necesarios llevar a cabo para que el proyecto que siga esta metodología sea exitoso. También, contiene una evaluación documentada de herramientas para el desarrollo de videojuegos en dos dimensiones que sean capaces de exportar el videojuego educativo a diferentes plataformas, especialmente con soporte para PC con GNU/Linux, Web y dispositivos móviles con Android.

Y por último un documento que especifica, cómo debe llevarse a cabo la administración de la configuración para el mantenimiento del código de los deferentes videojuegos educativos que se vayan a crear. Propone una política de versionamiento, administración de repositorios y la evaluación de herramientas que permitan interactuar con los repositorios en la nube, para el mantenimiento del código de los videojuegos.

3.2. Análisis Foda del proyecto

- Análisis interno
  - Fortalezas
    - Se cuenta con asesoría pedagógica para la crítica constructiva de lo que se debería tomar en cuenta en el diseño y concepto del videojuego.
    - Algunos videojuegos desarrollados por epesistas a los que se les puede cuestionar sobre las tecnologías y metodología empleada.
  - Debilidades
    - No hay en la institución alguien que esté ligado al desarrollo de videojuegos para Android.
    - Ninguna experiencia desarrollando videojuegos.
    - Información muy dispersa sobre el diseño de videojuegos educativos.
- Análisis externo
  - Oportunidades
    - Servir de base y guía para el desarrollo de videojuegos educativos, con tecnologías acorde a las necesidades de la institución que ayuden a futuros epesistas en su desarrollo.
    - Tratar de disuadir el tabú de que los videojuegos no son buenos para educar.
  - Amenazas
    - Hay muchas herramientas para el desarrollo de videojuegos que ofrecen despliegue multiplataforma, lo que puede dificultar la elección de las adecuadas a comparar.
    - Muy poca información sobre desarrollo de videojuegos orientados a la educación.

3.3. Investigación preliminar para la solución del proyecto

En Guatemala existe una industria de desarrollo videojuegos muy joven aún, y la mayoría son independientes, es decir que no están afiliados a alguna empresa distribuidora. Como paso preliminar para este trabajo se realizaron encuestas sobre el proceso de desarrollo de videojuegos a algunas de estas pequeñas empresas en agosto de 2014, especialmente buscando si estas utilizaban alguna metodología de desarrollo específica. El personal que necesitaban en el desarrollo, las herramientas de desarrollo que utilizan y para que plataformas desarrollaban. Todo para obtener información y orientar de mejor manera el trabajo realizado.

Es importante mencionar que la encuesta no es muy significativa pero sirvió de base para realizar el trabajo. Las empresas que ofrecieron su tiempo para responder a la encuesta fueron:

- Smirk Games
- CrystalBit
- 502 Studios

3.3.1. Resultados de la encuesta

A continuación se presenta un resumen de los resultados de la encuesta.

3.3.1.1. Metodologías y documentación

De las empresas entrevistadas solo Smirk Games y CrystalBit dijeron que utilizaban una metodología de desarrollo, pero solo CrystalBit seleccionó una de las respuestas propuestas, que consiste en codificar y corregir, como se puede

apreciar en la figura 1. Smirk Games, aunque mencionó otra metodología, al final resultaba en lo mismo en codificar y corregir como CrystalBit, y en medio de ello ir mejorando el videojuego hasta que sea considerado divertido y entretenido.

Figura 1. **Metodologías propuestas**

Fuente: elaboración propia.

Aun así, las empresas crean documentación, que está hecha con base en su propia experiencia entre las que destacan.

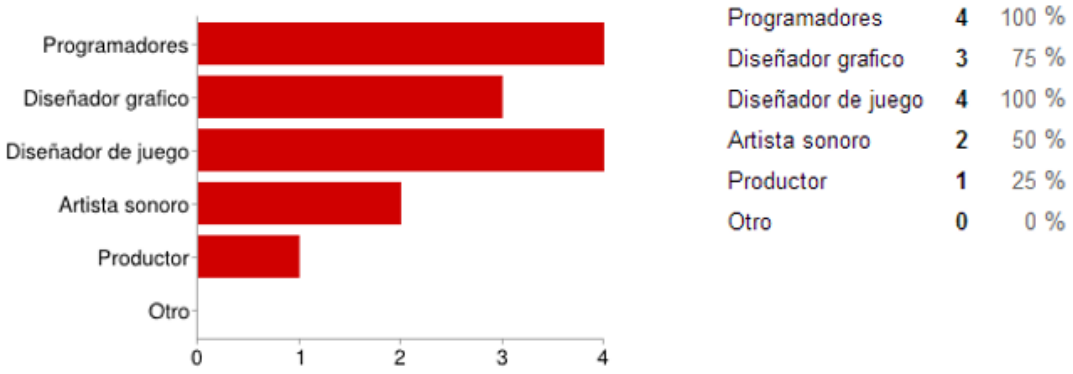
- Bocetos o arte conceptual: este contiene la parte gráfica que identificará al videojuego como interfaz, personajes, escenarios, entre otros.
- Documento de diseño del juego: este contiene la mecánica del juego con todas sus características.
- Cronograma de actividades: para controlar las tareas asignadas y su realización en el tiempo establecido.

3.3.1.2. Equipo de desarrollo

Como en el desarrollo de cualquier software, siempre hay presente un equipo multidisciplinario que se encarga de que se lleve a cabo. Y en los videojuegos no es una excepción, ya que al final es software para entretener.

Las empresas entrevistadas cuentan con equipos de dos a siete personas, que se encargan de diversas actividades. A continuación se muestra una gráfica con los roles propuestos y sus resultados.

Figura 2. **Roles en equipo de desarrollo**



Fuente: elaboración propia.

Por lo que se puede apreciar las empresas respondieron positivamente a varios de los roles involucrados en el desarrollo de videojuegos, en los que destacan los programadores, el diseñador gráfico y de juego. En cuanto al productor no aplicaba del todo, ya que ellos mismos costeaban todo lo relacionado a los videojuegos que desarrollan.

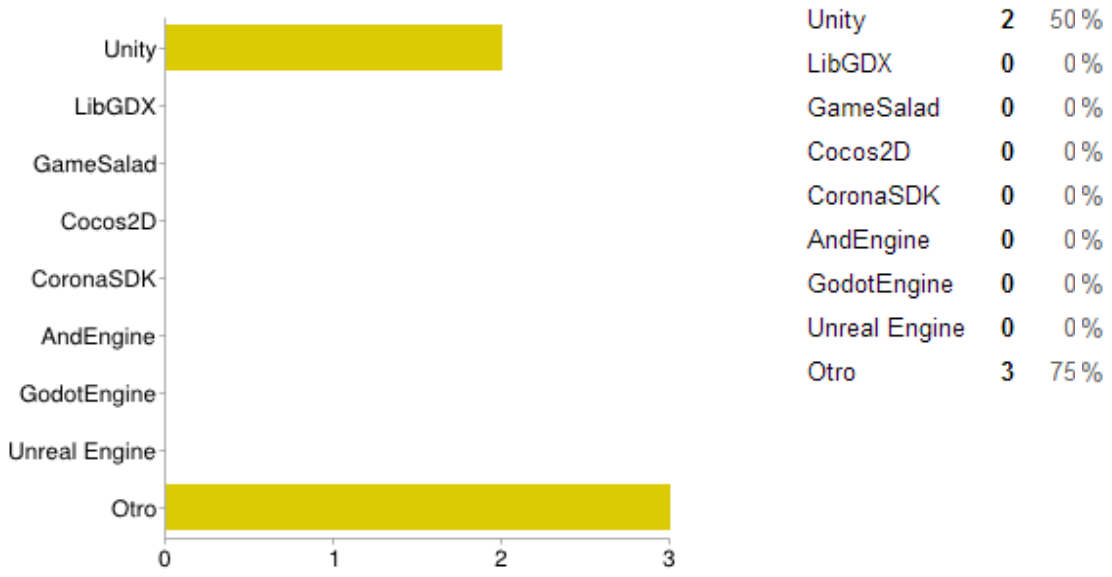
3.3.1.3. Game Engines o frameworks utilizados

Un videojuego puede ser desarrollado desde cero utilizando algún lenguaje de programación o bien al utilizar alguna herramienta que permita de alguna forma facilitar el proceso de desarrollo.

Estas herramientas abstraen algunas complicaciones que se tienen a la hora de desarrollar los videojuegos. Algunas están enfocadas a plataformas específicas, otras son multiplataforma, es decir que el juego se hace una sola vez y puede ser exportado a Android, Web, y otros por ejemplo, con solo unas cuantas modificaciones.

Para el caso de las empresas entrevistadas se les presentaron varias opciones entre las que podían escoger, que eran las más populares de la web. Pero la sorpresa fue que solo una escogió una de las propuestas y las demás utilizaban otras, lo cual es bueno porque aumenta el conocimiento de más opciones. La figura siguiente muestra ese hecho.

Figura 3. **Herramientas propuestas**



Fuente: elaboración propia.

Solo 502 Studios utiliza Unity en su versión gratuita y otras herramientas de código abierto. Las demás empresas utilizan otras herramientas. La siguiente tabla muestra los detalles.

Tabla I. **Empresas y herramientas**

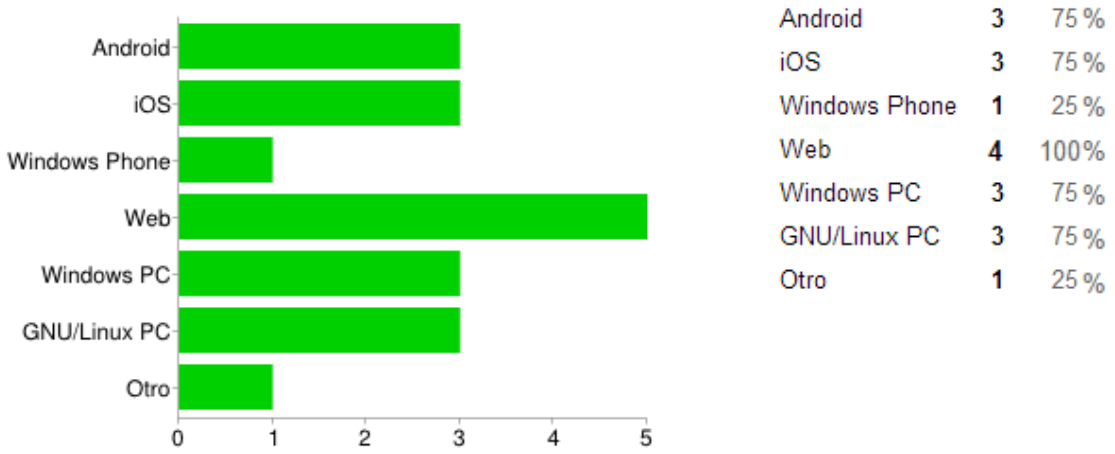
Empresa	Herramientas	Licencia
Smirk Games	Flash	Pago
CrystalBit	ImpactJS	Pago
502 Studios	Unity, PlayN, y Mono Game	Pago y gratuita, libre y libre.

Fuente: elaboración propia.

3.3.1.4. Plataformas destino de videojuegos

Cuando se habla de plataformas destino de los videojuegos se refiere a sobre qué sistema base de software se utilizará, para que el videojuego se ejecute. Hoy en día, está en auge el desarrollo de videojuegos para dispositivos móviles, como *tablets* y teléfonos inteligentes, que para muchos desarrolladores es una buena fuente de ingresos. En Guatemala las empresas entrevistadas, desarrollan para varias plataformas para que sus juegos tengan mayor alcance. La siguiente gráfica muestra las plataformas para las que desarrollan estas empresas.

Figura 4. **Plataformas utilizadas**



Fuente: elaboración propia.

La gráfica es bastante clara en cuanto a las plataformas seleccionadas. Internacionalmente hablando la mayoría de juegos desarrollados los acapara Android y iOS para dispositivos móviles y Windows para computadoras convencionales.

3.3.2. Conclusiones

Las metodologías que utilizan las empresas entrevistadas se basan en su propia experiencia, y no están formalmente definidas. Por los datos recaudados utilizan algunas prácticas de las metodologías agiles, realizando alguna documentación.

En resumen el proceso de desarrollo comienza por definir y acordar la idea del videojuego a realizar. Luego se especifican sus características y se planifican los plazos de entrega.

Los equipos de desarrollo son pequeños, y una persona puede desempeñar más de un rol a la vez. Los roles más cubiertos son el programador, el diseñador gráfico, diseñador de juego y el encargado del audio.

El conocimiento de nuevas herramientas además de las propuestas ayudarán a tener más opciones, para elegir el desarrollo de los videojuegos. Otra característica importante de algunas herramientas que estas empresas utilizan es que son multiplataforma.

3.3.3. Situación actual

La industria de videojuegos en Guatemala aún es muy joven, pero existen personas que se están aventurando a este mundo, donde la competencia y el mercado son a escala mundial. La industria no cuenta con una gran infraestructura y se emplean entre una a cinco personas por empresa. La mayoría de proyectos son de corta duración y se crean para plataformas como PC, moviles y web mayoritariamente. Lo interesante también es que algunas de las empresas entrevistadas se preocupan por la educación y crean videojuegos

con fines educativos, esto podría significar para Edulibre una oportunidad de recibir apoyo hacia su proyecto.

A continuación se presenta un análisis sobre las fortalezas, oportunidades, debilidades y amenazas (Foda), para la industria guatemalteca de videojuegos. Este análisis está basado en el artículo “Industria de Desarrollo de Videojuegos en Argentina”, como ejemplo de una realidad similar y el estudio hecho para el presente trabajo.

- Fortalezas
  - Capacidad de recursos humanos: el país cuenta con bastantes personas con conocimientos de tecnologías de la información. Además existen carreras a nivel de diversificado y educación superior orientadas a este sector.
  - Comunidad de desarrolladores: en Guatemala los miembros de las empresas entrevistadas se conocen entre sí, y han formado una pequeña comunidad de desarrolladores de videojuegos en Guatemala llamada GameDevGT, a la que cualquiera es libre de unírseles. La idea de esta comunidad es compartir conocimientos y experiencias en el área.
  - Costos competitivos: disponibilidad de personal calificado a costos competitivos internacionalmente.

- Debilidades
  - Escasez de personal: no hay mucho personal en el país con la capacitación adecuada, lo que puede resultar en costos mayores en su contratación.

- Poca experiencia de profesionales en el área: dado que es una industria muy joven, la experiencia adquirida radica en pocas personas.
- Falta de casos de éxito: aunque hay videojuegos desarrollados en el país, no ha habido alguno con un éxito significativo.
- No hay carreras especializadas: aunque hay algunas empresas que dan capacitación a personas en el uso de herramientas de desarrollo, no hay carreras formales en el área.

- Oportunidades
  - Mercado mundial: la aparición de tiendas online de se puede subir el software y darlo a conocer internacionalmente.
  - Mercado en crecimiento: el mercado mundial de videojuegos registra fuertes ingresos en los últimos años, superando a la industria cinematográfica en varias ocasiones.

- Amenazas
  - Creciente competencia: dado el creciente auge de las tecnologías, la competencia se incrementa cada vez más ya que hay otros países interesados en la industria, además de competir con empresas de desarrollo reconocidas mundialmente.

3.3.4. Estudio de metodologías de desarrollo de software

A continuación se presenta un estudio acerca de las metodologías ágiles para el desarrollo de software.

3.3.4.1. Metodologías ágiles

Hoy en día, hay muchas empresas de desarrollo de software que utilizan metodologías ágiles, por las cualidades que ofrecen. Los procesos y metodologías existentes se basan en el manifiesto ágil, que plantea.

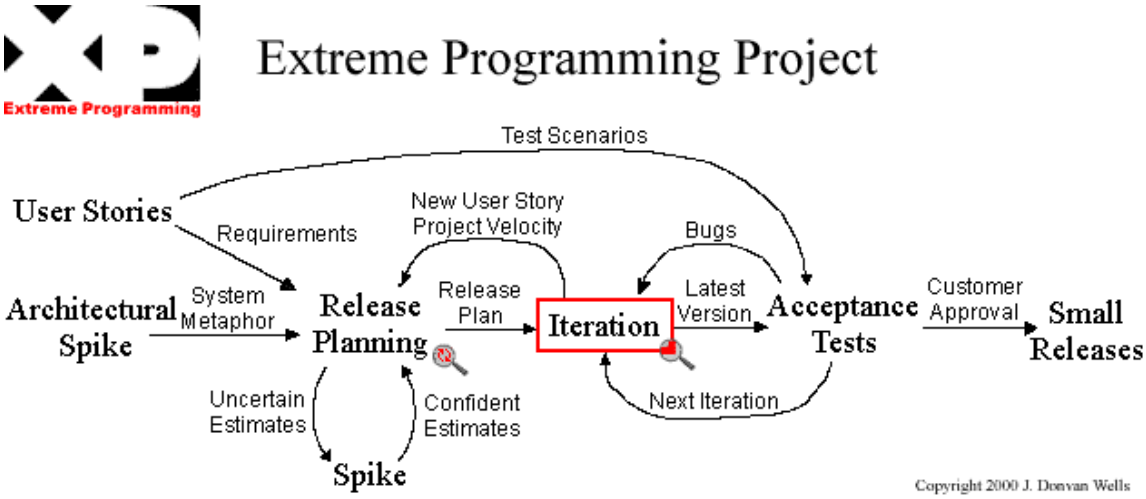
- Individuos y sus interacciones, frente a procesos y herramientas
- Software en funcionamiento, frente a documentación exhaustiva
- Colaboración del cliente, frente a una negociación de contrato
- Respuesta al cambio, frente a seguir un plan

Existen varias metodologías agiles, pero en particular se abordarán la metodología Scrum, XP y SUM. La razón de esta decisión es que tienen características que las hacen adaptables a cualquier proyecto de software.

3.3.4.1.1. Extreme Programming (XP)

Es una metodología ágil desarrollada a partir de un conjunto de buenas prácticas tomadas de la vida cotidiana. Fue diseñado para trabajarse en grupos de dos a diez personas. Permite desarrollar software de alta calidad con un presupuesto y en un tiempo previsible, y con una sobrecarga mínima de trabajo.

Figura 5. **Proceso de XP**



Fuente: *Extreme Programming*. http://www.extremeprogramming.org/map/images/project.gif. Consulta: 8 de septiembre de 2014.

La metodología agrupa doce prácticas fundamentales del desarrollo de software, haciendo que al usarse en conjunto se obtenga el mayor beneficio. Las doce prácticas se listan a continuación.

- Programación en parejas: todo el código es escrito por dos programadores en una computadora.
- Planificar el juego: implica determinar el alcance de la próxima liberación combinando las prioridades del negocio y las estimaciones técnicas. A medida que cambia la realidad hay que actualizar el plan.
- Desarrollo dirigido por las pruebas: se codifica a partir de las historias de usuario. Se trabaja en pequeños ciclos que son probados.
- Equipo completo: todos los colaboradores de un proyecto XP conforman un solo equipo.

- Integración continúa: XP busca mantener el sistema completamente integrado siempre, ya que se realizan varios *builds* por día.
- Mejoras de diseño: los programadores mejoran el diseño al hacer el sistema más flexible, sin afectar su comportamiento, lo que permite reducir tiempo y mejorar la calidad del producto.
- Pequeños entregables: cada *release* del producto es presentado al usuario al final de la iteración.
- Estándares de codificación: se siguen estándares de codificación para que el código sea fácilmente entendible por otros programadores. Parecerá que el código fue escrito por un solo programador.
- Código de propiedad colectiva: el código es propiedad de todos. En XP, una pareja de programadores puede mejorar la proporción del código en cualquier momento.
- Desarrollo simple: se comienza con un diseño sencillo, y que se mejora conforme avance el desarrollo del proyecto.
- Metáfora del sistema: constituye la visión común del trabajo de los programadores.
- Paso sostenible: los equipos de XP son de largo plazo. Los equipos trabajan 40 horas por semana para no estar cansados durante las jornadas de trabajo.

- Ventajas
    - Permite a los desarrolladores interactuar con los usuarios finales incrementando así la comunicación y mejorando el sistema mediante retroalimentación.
    - Agiliza el proceso de desarrollo mediante una documentación mínima.
    - Permite el empoderamiento de los desarrolladores para tomar decisiones.

- XP es fácil de mantener.
- El capital intelectual invertido en un desarrollador individual es distribuido a través de todo el equipo.

- Desventajas
  - La metodología es difícil de integrar en compañías grandes.
  - El usuario debe ser tomado en cuenta para las responsabilidades del día a día apoyando al equipo de desarrollo.

3.3.4.1.2. Scrum

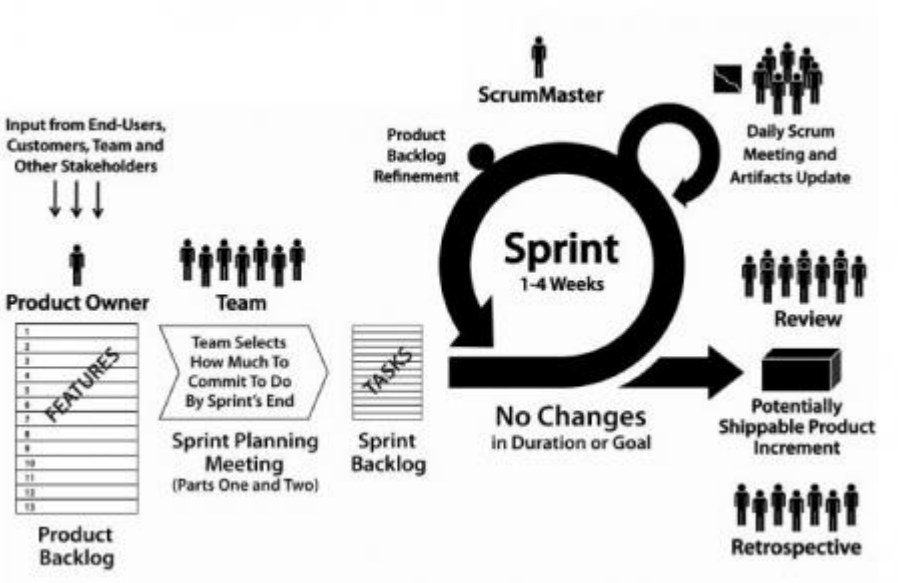
Es una metodología ágil para administrar y controlar el desarrollo de software de un producto en forma iterativa e incremental. También es una metodología de desarrollo adaptativa, rápida, con un proceso de desarrollo de software autoorganizable, que toma su nombre de la estrategia del juego de rugby.

Lo interesante de esta metodología es que no define técnicas de desarrollo de software específicas para la implementación, se concentra en cómo deben de funcionar los miembros del equipo para producir la flexibilidad del sistema en un ambiente cambiante. Esto brinda flexibilidad y permite ajustar el proceso a la realidad y forma de trabajo de cada proyecto.

Scrum se estructura en tres fases llamadas *pre-game*, *game* y *post-game*. Durante la fase de *pre-game* se define el producto basado en las características conocidas, estimando su tiempo y costo. También se analiza el sistema a construir se define la arquitectura y se realiza un diseño de alto nivel de la solución. La fase *game* o también conocida como *development*, consta de iteraciones llamadas *sprints* que duran de dos a seis semanas y donde se

desarrollan las características del producto. Durante esta fase se observan y controlan los diferentes ambientes y variables técnicas. La fase *post-game* contiene la finalización del *release*. Se completa cuando los requerimientos son satisfechos. Se verifican las versiones a entregar y se realiza la documentación final.

Figura 6. **Proceso Scrum**



Fuente: FRISCHE, Robert. *Scrum Process*.http://www.mentordigital.co.uk/media/115719/agile-scrum.png?width=460&height=292&crop=true. Consulta: 9 de septiembre de 2014.

La metodología define tres roles entre los cuales se dividen todas las responsabilidades del proyecto.

- Dueño del producto: está a cargo del proyecto y es quien maneja y prioriza las características a desarrollar.
- Scrum Master: es el responsable de que todos los miembros del equipo sigan el proceso como es debido y de remover los impedimentos que surjan en el transcurso de este.

- Equipo de Scrum: es un equipo multidisciplinario y autoorganizado, y su cometido principal es construir el producto que el dueño del producto precisa.

Además define un conjunto de artefactos útiles.

- *Backlog* de producto: representa el conjunto de todas las características que definen el producto.
- *Sprint Backlog*: representa el conjunto de todas las características y tareas a las cuales el equipo se compromete a realizar durante la iteración actual.
- *Sprint Burndown Chart*: es un gráfico que representa el trabajo que resta realizar para terminar un *sprint*.
- *Release Burndown Chart*: es un gráfico que representa el progreso del trabajo respecto al plan de entregas.
- *Task Board*: representa el estado de las tareas que el equipo está realizando durante el *sprint* actual.
- *Potentially Shippable Product*: representa el producto actual, el obtenido por todos los incrementos de cada *sprint*.

- Ventajas
  - Scrum está diseñado para ser flexible a los cambios de las variables del ambiente de desarrollo.
  - Provee un mecanismo de control para planificar el *release* de un producto y administrar las variables de progreso del proyecto.
  - Permite a los desarrolladores diseñar soluciones ingeniosas en cualquier momento a lo largo del proyecto, liberando el más apropiado *release*.
  - Es fácil de aprender.

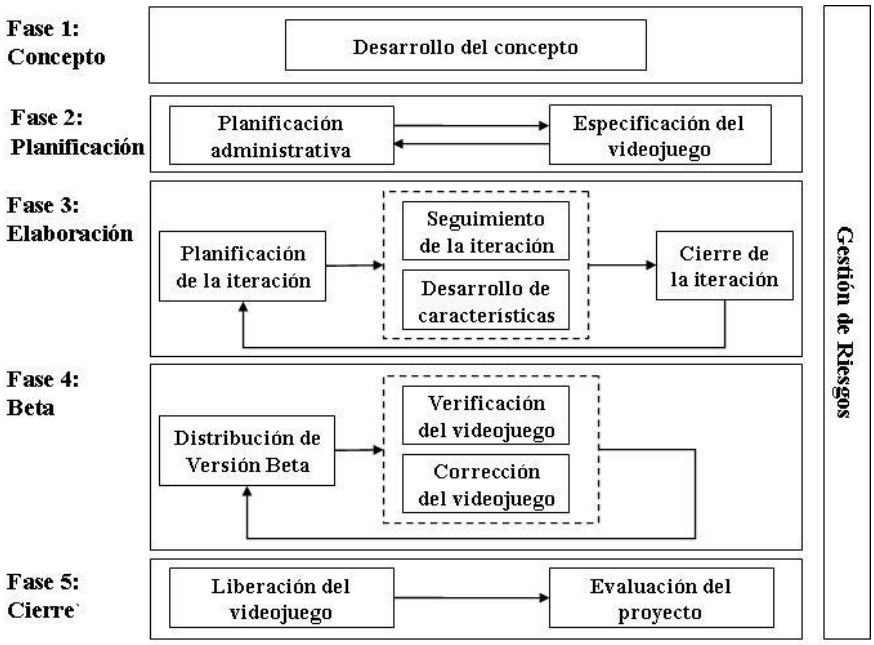
- Requiere muy poco esfuerzo para comenzar a utilizarse.
- Las reuniones se dedican a inconvenientes recientes, evitando el estancamiento.

- Desventajas
  - Requiere delegar responsabilidades al equipo, incluso permite fallar si es necesario.
  - Puede causar resistencia en su aplicación cuando las personas no están abiertas a cambios de una metodología rígida a una ágil.

3.3.4.2. Sum

Es una metodología propuesta para realizar videojuegos, basada principalmente en Scrum, de hecho al estudiarla bien, se puede apreciar que es ver un Scrum desglosado, como se puede apreciar en la figura.

Figura 7. **Proceso de Sum**



Fuente: *Proceso de Sum*.
http://www.gemserk.com/sum/Sum/guidances/supportingmaterials/resources/etapas2.jpg.
Consulta: 10 de octubre de 2014.

La estructura en sí, es una metodología Scrum orientada al desarrollo de videojuegos, aunque su definición es considerablemente larga, solo define los pasos a realizar, no como hacerlos. Lo interesante es que toma en cuenta varias cosas que realmente se necesitan en el desarrollo de videojuegos en general, como el concepto, la mecánica que tendrá este, las pruebas y otros. Todas estas englobadas en una metodología. Su estructura es comprensible, además parece ser idónea para su modificación, en la propuesta a realizar.

- Ventajas
    - Es iterativa e incremental
    - Adaptable a los cambios
    - Fácil de aprender
    - Orientada al desarrollo de videojuegos

- Desventajas
    - No es una metodología de uso general, solo una propuesta
    - Al igual que Scrum se delegan muchas tareas

3.3.4.3. Análisis de metodologías

Las metodologías agiles son iterativas e incrementales, por lo que según va avanzando el proyecto, se van obteniendo pequeñas versiones del producto en intervalos cortos y regulares de tiempo. Esto realmente facilita una visión temprana de lo que será el producto final, reduciendo los costos de cambios en los requerimientos en forma tardía y brinda una mejor retroalimentación con el cliente.

3.3.4.3.1. Conclusión

En conclusión las metodologías agiles dadas sus características son las idóneas para realizar el desarrollo de videojuegos, ya que se obtendrán los siguientes beneficios.

- Ver la idea del videojuego de forma temprana con el uso de iteraciones.
- Se involucra a todo el equipo de desarrollo para la toma de decisiones.
- Se podrá tener una versión jugable muy pronto para probarla y obtener retroalimentación.

- Proponer mejoras al diseño del videojuego, si no se ve como se esperaba.
- Agregar o quitar características más fácilmente.
- Se podrá validar el contenido educativo del videojuego conforme avance a fin de detectar problemas de forma temprana.

3.4. Presentación de la solución del proyecto

La solución al proyecto de Edulibre consiste en un marco de trabajo que les permita llevar de forma ordenada y eficiente el desarrollo de videojuegos educativos. Este marco de trabajo es un conjunto de subproductos, que solucionan diversos problemas.

- Perspectivas del producto:
    - Contar con una metodología de desarrollo propia, dirigida al desarrollo de videojuegos educativos.
    - Contar con una herramienta de desarrollo de videojuegos en dos dimensiones, que les permitiera tener disponible el videojuego para computadoras con GNU/Linux, en Web y para dispositivos con sistema operativo Android, a partir de la evaluación de diversas opciones existentes en el mercado.
    - Contar con una adecuada administración de la configuración para controlar, evaluar y verificar la historia del software durante su desarrollo, por medio del manejo de diferentes los diferentes ambientes juntamente con una política de versionamiento.
    - Contar con herramienta de software en la nube para el control de versiones de los videojuegos, con posibilidad de colaborar en equipo en la cual el alojamiento de los repositorios sea privado y gratuito.

El marco de trabajo propuesto a Edulibre es producto, compuesto por varios subproductos, que son documentos que contienen en detalle los procedimientos a llevar acabo para el desarrollo de los videojuegos. A continuación se presenta un resumen de cada uno.

3.4.1. Propuesta de metodología de desarrollo de videojuegos educativos

Lo siguiente es un resumen de la metodología propuesta a Edulibre, para mayor detalle sobre la misma, ir al apéndice A.

3.4.1.1. Justificación

Dado que la Asociación Civil Edulibre tiene como proyecto la elaboración de una plataforma que albergue software de juego educativo para niños y niñas de educación primaria, carece de una metodología de desarrollo de software para elaborar los videojuegos educativos que integrarán esta plataforma, por lo tanto surge la necesidad de elaborar una. La propuesta de la metodología está basada en las metodologías de desarrollo ágil XP, Scrum y Sum principalmente donde se toma su estructura, se modifica y adapta a lo que se está buscando.

3.4.1.2. Objetivos de la metodología

La propuesta de una metodología de desarrollo de videojuegos educativos, tiene como objetivo principal el facilitar el desarrollo de videojuegos educativos en tiempo y costos razonables, dando lugar a ser mejorado cada vez que se utilice, en función de la experiencia adquirida. La metodología es fácilmente manejable para equipos multidisciplinarios pequeños de no más de siete integrantes y proyectos con máximo de duración de seis meses.

3.4.1.3. Roles

La metodología define cinco roles: equipo de desarrollo, pedagogo, cliente, líder de proyecto y verificador de videojuego. El equipo de desarrollo es equivalente al Scrum Team, con la diferencia que se definen subroles dentro del equipo. El líder de proyecto y el cliente son equivalentes al Scrum Master y Product owner respectivamente.

Dado el tipo de proyecto, el equipo de desarrollo está conformado por varios especialistas en diferentes áreas, que juntos darán vida al videojuego. Estos se definen a continuación.

Tabla II. **Subroles equipo de desarrollo**

Subrol	Descripción
Diseñador del videojuego	Se encargará de la parte de la mecánica, el guión (si lo hubiera, depende del tipo de juego), los personajes, los niveles, y demás elementos que darán vida al videojuego.
Diseñador gráfico	Se encargará de crear todo el material artístico del videojuego, incluyendo el arte de concepto, modelos en 2D y animaciones.
Programador	Se encarga de darle vida al videojuego diseñado por el diseñador de videojuegos. Su tarea es diseñar, implementar y verificar el software que compondrá al videojuego.
Diseñador de efectos de sonido	Es el que está encargado de crear todos los efectos de sonido del videojuego que sean requeridos, así como también la música. Los efectos de sonido deben ser creados acorde, y que correspondan a lo que el jugador está viendo, y la acción que esté realizando.

Fuente: elaboración propia.

Tabla III. **Roles restantes**

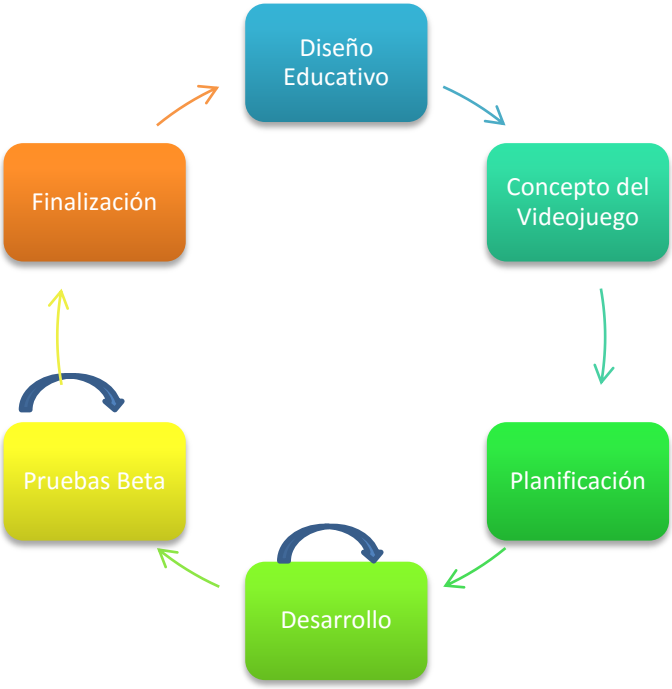
Pedagogo	Será la persona encargada de trabajar en equipo junto con el diseñador del videojuego, el diseñador gráfico y el programador para lograr que el videojuego cumpla su función de educar.
Cliente	Es el encargado de especificar que requiere el videojuego y mantener la perspectiva del mismo.
Líder de proyecto	Será el encargado de guiar el desarrollo del videojuego, promover buenas prácticas, interactuar con el cliente y resolver cualquier impedimento que pueda atrasar el proyecto.
Verificador de videojuego	La función principal de este rol es realizar la verificación funcional del videojuego. Entra en acción cuando ya se tiene una versión beta del videojuego, ya casi completa. Cuando realice la evaluación deberá elaborar un documento donde describa los errores encontrados.

Fuente: elaboración propia.

3.4.1.4. Ciclo de vida de la metodología propuesta

La propuesta de la metodología se divide en varias fases de las cuales algunas son iterativas e incrementales. Las seis fases secuenciales son: diseño educativo, concepto del videojuego, planificación, desarrollo, pruebas beta y finalización. La figura siguiente muestra adecuadamente las fases. La mayoría de fases tiene una sola iteración excepto la de desarrollo y pruebas beta que pueden tener más.

Figura 8. **Fases de la metodología propuesta**

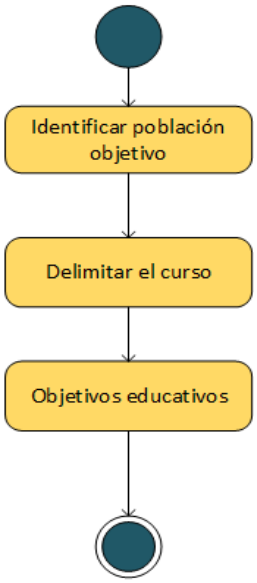


Fuente: elaboración propia.

3.4.1.4.1. Diseño educativo

La fase de diseño educativo consiste en organizar toda la estructura del contenido educativo a impartir. Se deben resolver interrogantes que se refieren al alcance, contenido y tratamiento que debe ser capaz de apoyar el videojuego. La fase diseño educativo está compuesta de varias actividades que se describen a continuación.

Figura 9. **Flujo de las actividades a realizar**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Identificación de la población objetivo**:** esta actividad consiste en conocer a los usuarios que harán uso del videojuego. No es lo mismo hacer un videojuego para un niño de cinco años que para uno de diez. Es importante ya que buena parte de la motivación y el esfuerzo así como la forma de comunicación dependen de quienes serán los futuros usuarios del material.

- Delimitación del curso**:** esta actividad consiste en la delimitación del contenido de área de conocimiento a impartir en el videojuego. Para esta parte se debe tomar en cuenta el contexto en que la ejemplificación y ejercitación se van a dar.

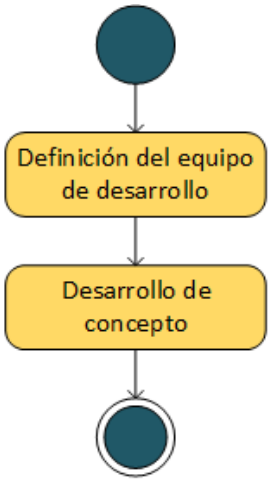
- Objetivos educativos**:** están enfocados a lo que se quiere lograr en el alumno, es decir la conducta que se espera de él, y donde se especifican los contenidos a considerar durante el proceso de enseñanza aprendizaje, propuesto por el videojuego. En el caso de Guatemala pueden tomarse como base los del Curriculum Nacional Base, propuestos por el Ministerio de Educación (Mineduc).

3.4.1.4.2. Concepto del videojuego

Para esta fase lo principal es tomar las decisiones técnicas y elementos de juego sobre el producto a desarrollar. Las decisiones técnicas implican la elección de las herramientas, tecnologías a utilizar y las plataformas para las que se pretende desarrollar. En cuanto a los elementos de juego se busca determinar las principales características que tendrá este, como la historia, los personajes, la ambientación y la mecánica del juego principalmente.

Es importante mencionar que el desarrollo del concepto se debe tomar en cuenta los aspectos definidos en la fase de diseño educativo, para que la propuesta pueda cumplir con los objetivos de enseñanza aprendizaje establecidos. Para completar esta fase deben de realizarse dos actividades que se ejecutan secuencialmente estas son: la definición del equipo de desarrollo y el desarrollo del concepto.

Figura 10. **Actividades – Concepto del videojuego**



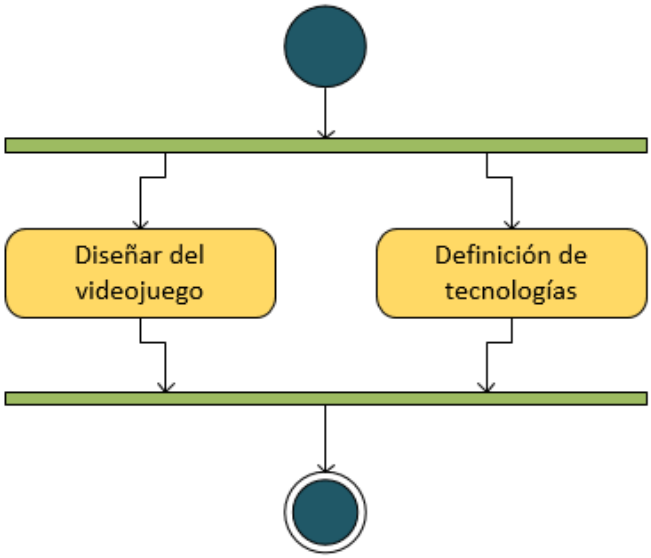
Fuente: elaboración propia, con programa Microsoft Office Visio.

- Definición del equipo de desarrollo: para esta parte se busca armar el equipo de desarrollo que creará el videojuego, y estará presente en el resto de las fases. Para esto se debe tener en cuenta el concepto del videojuego, donde se identifican las necesidades técnicas y artísticas requeridas por el proyecto. En función de estas necesidades se procede a seleccionar a las personas serán parte del equipo de desarrollo.

- Desarrollo del concepto: esta actividad implica la realización de dos tareas para definir los aspectos técnicos, y los elementos que compondrán el videojuego educativo. El concepto del videojuego se lleva a cabo a partir de ideas y propuestas por parte de cada rol sobre los aspectos a definir en función de lo planteado en el documento de diseño educativo. Las propuestas no necesariamente tienen que quedar definidas de una vez, estas pueden irse afinando a través de reuniones y analizar factibilidad. Las dos tareas propuestas se realizan en paralelo,

se puede empezar por cualquiera, ya que las decisiones en una pueden afectar a la otra.

Figura 11. **Tareas – Desarrollo de concepto**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Diseñar el videojuego: para esta tarea el fin principal es definir varios aspectos del videojuego como: perspectiva, género, mecánica de juego, y otros. Para esta parte se pueden realizar pruebas sobre el concepto del videojuego, verificando si este es divertido y entretenido. Se debe tomar en cuenta la fase de diseño educativo.

- Definición de tecnologías: para esta tarea se elige la plataforma o plataformas donde correrá el videojuego. Además las tecnologías y herramientas a utilizar en su desarrollo. Para probar las herramientas se puede hacer prototipos que prueben algunos

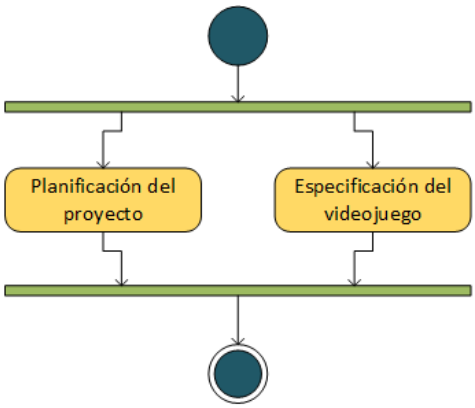
aspectos seleccionados del videojuego, y seleccionar la adecuada.

3.4.1.4.3. Planificación

Para esta fase se realiza el cronograma de actividades que tendrán el resto de las fases y además se especificarán las características que tendrá el videojuego. Esta fase implica la realización de dos actividades cuyos resultados compondrán el plan del proyecto en sí. Estas actividades son mutuamente dependientes ya que el cronograma de actividades debe ser coherente con el tiempo estimado y la realización de los requerimientos del videojuego.

Esta debería ser una fase corta. Se debe tener en cuenta que para que sea oficial, el cliente debe estar de acuerdo. El cronograma de actividades que se obtiene en esta fase, es el estimado y como tal también está sujeto a cambios conforme se vaya realizando el proyecto. Esto para reflejar el estado actual del desarrollo del videojuego, y tomar las medidas necesarias aprovechar de mejor manera el tiempo.

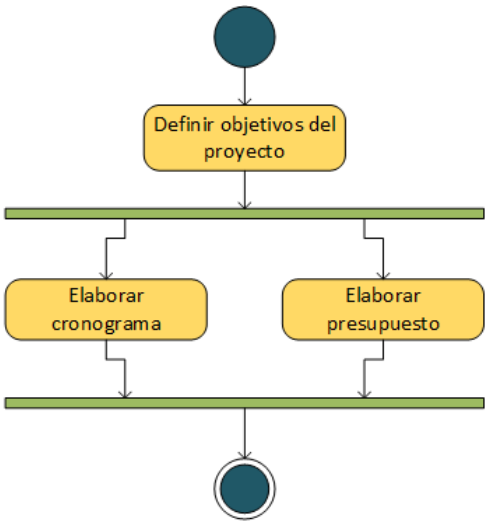
Figura 12. **Fase de planificación**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Planificación del proyecto: como cualquier proyecto, este debe tener una planificación que permita llevar a cabo las tareas en un orden adecuado. Para esta actividad se deben realizar tres tareas definiendo diversos elementos del plan de proyecto. La primera tarea “Definir objetivos del proyecto” es la primera que debe realizarse, luego de forma paralela se pueden realizar la de “Elaborar el cronograma” y “Elaborar el presupuesto”.

Figura 13. **Tareas – Planificación del proyecto**



Fuente: elaboración propia, con programa Microsoft Office Visio.

  - Definir objetivos del proyecto: para cualquier proyecto es importante definir objetivos claros que se requieren alcanzar para dar por terminado el proyecto. Para este apartado se recomienda que los objetivos a definir sean SMART, para medir su éxito.

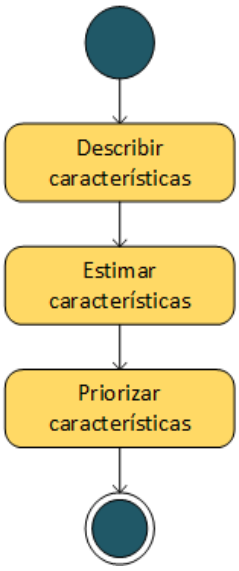
  - Elaborar el cronograma: crear un cronograma es parte de cualquier proyecto que se quiera implementar. Para la creación del cronograma que albergará las fases restantes del proyecto se

toma como base el concepto del videojuego. En el cronograma se establecen las fechas estimadas para dar comienzo a la fase de desarrollo, pruebas beta y finalización. Se debe incluir también la cantidad estimada de iteraciones a realizar durante la fase de desarrollo junto con sus duraciones y criterios de finalización, para empezar la fase de pruebas beta**.**

- Elaborar el presupuesto: hacer un presupuesto es muy útil para el proyecto, ya que se desea estimar los costos en los que se incurrirá para su elaboración. Para esta tarea se debe determinar el costo total que tendrá el proyecto y la forma en que se obtendrán los recursos económicos necesarios para realizarlo, en función del concepto del videojuego y el resto de los aspectos del plan de proyecto.

- Especificación del videojuego: el propósito de esta actividad es definir las características tanto funcionales como no funcionales del videojuego. Se entiende por característica funcional, como un requerimiento desde el punto de vista del usuario final, como una historia de usuario. Las características funcionales son una excelente herramienta que tiene el cliente para comunicar al equipo los requisitos del videojuego y medir el progreso del mismo. Las características no funcionales son cualidades inherentes que el videojuego debe presentar, como por ejemplo calidad, documentación, rendimiento, y otros.

Figura 14. **Tareas – Especificación del videojuego**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Describir características: para realizar esta tarea se deben determinar y describir cuáles serán las características funcionales según el concepto del videojuego. Cada requerimiento no necesariamente tiene que ser bien detallado, pero si lo suficientemente descriptivo para estimar un tiempo razonable para su realización. Se pueden utilizar historias de usuario para definir mejor estas características.

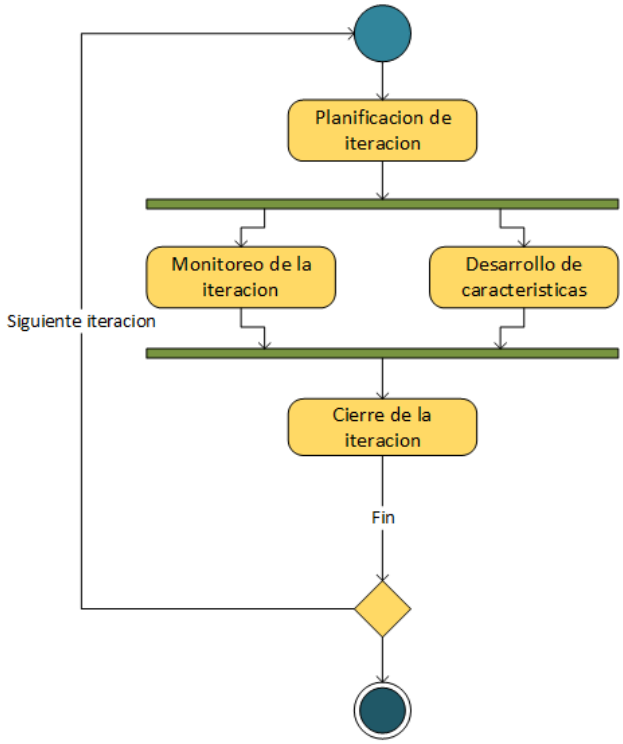
- Estimar características: se debe estimar el tiempo que tomará realizar todas las características descritas. Es importante que equipo de desarrollo las vea, y calcule cuanto tiempo les tomaría implementarlas, para tener una idea de lo que durará el proyecto. Pueden ser todas o bien una parte de ellas las estimadas, especialmente las consideradas urgentes.

- Priorizar características: para esta parte se clasifican en orden de las características definidas para el videojuego. Priorizar permitirá determinar de mejor manera en qué orden deben ser desarrolladas las características del videojuego, a modo de ver resultados rápido. Para esta tarea, es el cliente quien decide la importancia de las características de su punto de vista.

3.4.1.4.4. Desarrollo

Esta es la fase más importante, y cuyo objetivo principal es la realización del videojuego propuesto. Esta fase se trabaja de forma iterativa e incremental para que cuando cada iteración termine se obtenga un demo ejecutable del videojuego. Las actividades a desarrollar para esta fase son: planificación de la iteración, seguimiento de la interacción, desarrollo del videojuego y cierre de la iteración.

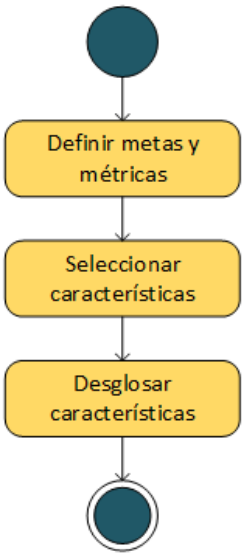
Figura 15. **Actividades – Fase de desarrollo**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Planificación de la iteración: la planificación se realiza una única vez por iteración y esta se hace para seleccionar las características a implementar. Para ello se realizan tres tareas secuenciales que son: definir metas y métricas, la selección de características y el desglose de las mismas.

Figura 16. **Tareas – Planificación de la iteración**



Fuente: elaboración propia, con programa Microsoft Office Visio.

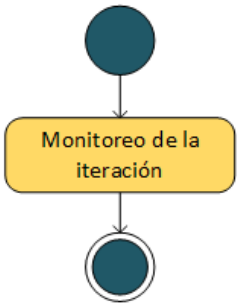
- Definir metas y métricas: es bueno definir una o más metas (o bien objetivos) en una iteración para tener una idea de hacia dónde vamos y lo que se quiere lograr. Es importante también definir de qué manera se va a medir el progreso de la iteración y si van logrando cada una de las metas propuestas. Esto permitirá que se puedan tomar mejores decisiones en el transcurso de la iteración y medir el éxito de la misma.

- Seleccionar características: para esta tarea se deben seleccionar las características que harán parte de la iteración. Se debe tomar en cuenta con base en su prioridad, el tiempo que tomará la realización de la iteración será la suma de los tiempos de cada requerimiento.

- Desglosar características: dada una característica, su realización puede conllevar la realización de varias tareas. El descomponer las características del videojuego en tareas de menor dificultad hará que sea más fácil su estimación, y asignarlas a un miembro del equipo, verificarlas y evaluarlas. Las tareas conviene dividirlas en grupos en función de las disciplinas que están involucradas. Por ejemplo tareas de lógica del juego, programación, visuales, audio, y otras.

- Monitoreo de la iteración: esta actividad en sí misma es una tarea a realizar. Su función es evaluar el avance de la iteración para detectar problemas e impedimentos que no permiten que el proyecto avance adecuadamente y tomar las decisiones necesarias en caso haya problemas.

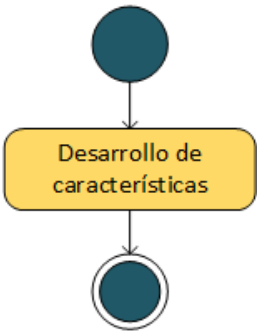
Figura 17. **Tareas – Monitoreo de la iteración**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Desarrollo del videojuego: comprende una sola tarea, la cual consiste simplemente en realizar las tareas que fueron programadas para la iteración.

Figura 18. **Tareas – Desarrollo del videojuego**

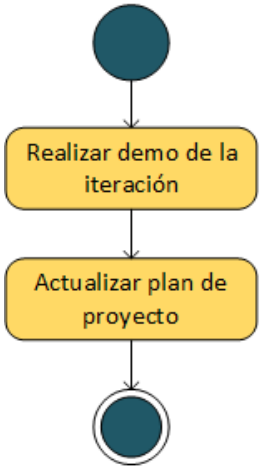


Fuente: elaboración propia, con programa Microsoft Office Visio.

- Desarrollar características: para esta tarea se realizan y validan las características programadas en la iteración, ejecutando cada una de las tareas que las componen. Cabe recalcar que la selección de las características a desarrollar queda a discreción del equipo, siendo responsables de llevarlas a cabo, tomando en cuenta que una característica se considera terminada, cuando se hayan realizado todas las tareas que componen la misma.

- Cierre de la iteración: se registra el estado actual del videojuego y lo ocurrido en el transcurso de la iteración, a modo de plasmar en el plan de proyecto la situación actual del mismo. Se realizan las siguientes tareas de forma secuencial.

Figura 19. **Tareas – Cierre iteración**



Fuente: elaboración propia, con programa Microsoft Office Visio.

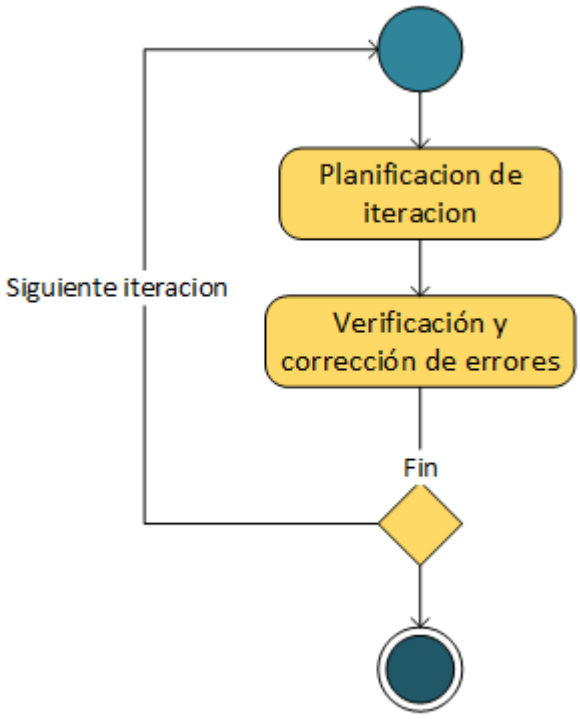
- Realizar demo de interacción: el principal objetivo de esta tarea es mostrar al cliente la versión del videojuego que se obtiene tras terminar la iteración. Y la intención de esta tarea es evaluar el cumplimiento de las características que se definieron para la iteración. Si no se alcanzaron a cumplir los requerimientos es bueno documentar las causas y evitar que se repitan.

- Actualizar plan de proyecto: la idea detrás de actualizar el plan de proyecto es registrar su situación actual. Además esto permitirá planificar de mejor manera la siguiente iteración, determinando cambios que sean necesarios para el propósito, en función del monitoreo realizado en la iteración.

3.4.1.4.5. Pruebas beta

Para esta fase el fin primordial es lanzar una versión beta del videojuego a fin de evaluar y ajustar distintos aspectos del mismo como diversión, contenido, menús, mecánica de juego, entre otros. Todo esto en función de los resultados que arrojen las pruebas del mismo, y así obtener un producto de mejor calidad. La fase prueba beta al igual que la fase de desarrollo también es iterativa, ya que al lanzarse una versión beta, es probada, si contiene errores entonces son reportados. Luego corregidos y se vuelve a lanzar una nueva versión beta, para volver a probar.

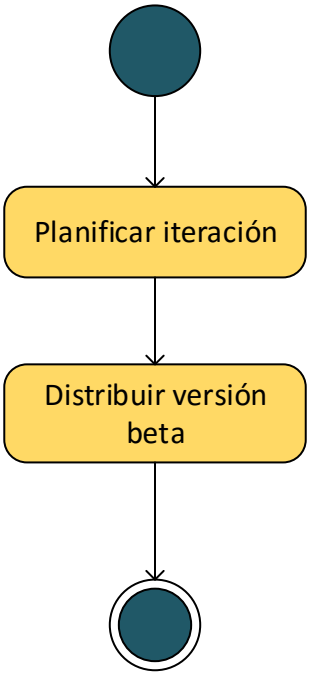
Figura 20. **Actividades – Fase de pruebas beta**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Planificación de iteración: llegados a esta actividad, ya se tiene una versión jugable del videojuego completa solo para su afinación. Al realizar esta planificación se deben tomar en cuenta los aspectos a evaluar del videojuego y cómo será la distribución beta para que pueda ser probada y verificada.

Figura 21. **Tareas – Planificación de iteración**



Fuente: elaboración propia, con programa Microsoft Office Visio.

  - Planificar la iteración: cuando se planifica la iteración se deben tomar en cuenta los aspectos funcionales y no funcionales del videojuego en los que se pondrá especial atención en las pruebas de ella. También se define a los verificadores del videojuego que evaluarán esos aspectos en el videojuego, la distribución del mismo y la forma en que los errores serán reportados para su

corrección. Esta planificación debe realizarse en el lanzamiento de cada versión beta para su verificación.

- Distribuir la versión beta del videojuego: distribuir la versión beta del videojuego implica los medios que serán utilizados para que esta llegue a las manos de los verificadores del videojuego, la manera en que reportaran los errores, y en qué aspectos del videojuego deben enfocarse.

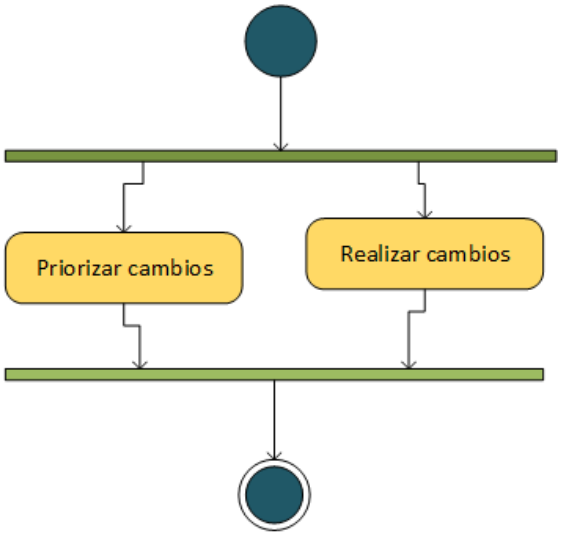
- Verificación y corrección de errores: consiste, en primer lugar, en verificar la versión beta del videojuego y reportar los errores encontrados, y por último realizar la corrección de los mismos, validando que al mismo tiempo estas correcciones no lleguen a producir nuevos errores. Los pasos a seguir son los siguientes, se puede avanzar más rápido si se realizan de forma paralela.

  - Verificación del videojuego: este paso solamente consiste en tomar los aspectos funcionales y no funcionales que se requieran sean verificados y reportar los resultados obtenidos. Estos resultados serán los errores encontrados, que pueden ser de varios tipos como por ejemplo en el código, la calidad de las imágenes, elementos poco atractivos, rendimiento, entre otros.

  - Corrección de errores: para este paso lo primordial es corregir los errores encontrados en el videojuego de acuerdo a los resultados obtenidos por parte de los verificadores del videojuego. De entre los resultados obtenidos puede que haya errores que sean críticos o urgentes los cuales deben de ser corregidos inmediatamente, es por ello que deben de priorizarse los ajustes y luego corregir en

función de su prioridad. Se deben de realizar las siguientes tareas de forma paralela.

Figura 22. **Tareas – Corrección de errores**



Fuente: elaboración propia, con programa Microsoft Office Visio.

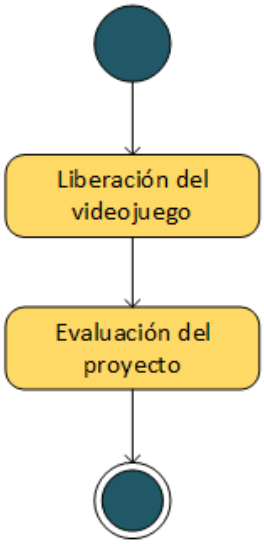
- Priorizar cambios: con base en los resultados arrojados por la evaluación y verificación del videojuego se deben definir los cambios que se van a realizar. Estos cambios tendrán que ser ordenados con base en el impacto e importancia que tienen para el videojuego.

- Realizar cambios: implica corregir los errores encontrados con base en la lista de cambios priorizados, antes de liberar el videojuego. A la hora de seleccionar el error hay que tomar en cuenta el impacto y el costo que tendrá corregirlo, además que esta elección debe ser informada al equipo.

3.4.1.4.6. Finalización

Llegada esta fase se tiene la versión final del videojuego, la cual queda a disposición del cliente, y además se hace una evaluación del desarrollo del proyecto. Para esta parte se realizan dos actividades, que se ejecutan secuencialmente, la primera consiste en la liberación del videojuego y la segunda en la evaluación del proyecto en sí. Es muy importante que para esta fase participen todas las personas que estuvieron involucradas en el proyecto.

Figura 23. **Actividades – Fase de finalización**



Fuente: elaboración propia, con programa Microsoft Office Visio.

- Liberación del videojuego: para la liberación en sí del videojuego fue seleccionada una versión beta que fue considerada estable, y se seleccionó como candidata a ser la versión final del videojuego. Para liberar apropiadamente la versión del videojuego se debe tomar en

cuenta la plataforma de distribución escogida, ya que puede que se requieran de diversas actividades para distribuir el producto.

- Evaluación del proyecto: para esta actividad se realiza lo conocido como evaluación *postmortem* en el cual se identifican aspectos relevantes que ocurrieron durante todo el proceso de desarrollo del proyecto, se registran las lecciones aprendidas y se plantean mejoras al proceso de desarrollo. Se realiza esto para hacer más eficiente y productiva la realización de futuros proyectos. La única tarea a realizar es la siguiente.

    - Evaluación *postmortem*: para este apartado el fin es evaluar el proyecto a partir de las decisiones que se tomaron durante el transcurso del proyecto, tomar en cuenta los puntos de vista de cada participante, y las evaluaciones que se hicieron cuando se terminó cada iteración. Una vez reunida toda esta información se identifican los problemas ocurridos, los éxitos conseguidos, las soluciones encontradas, el cumplimiento de los objetivos y la certeza de las estimaciones. A partir de todo esto se hacen varias conclusiones que dan lugar a las lecciones aprendidas y se buscan otras soluciones para mejorar el proceso de desarrollo.

3.4.2. Evaluación de tecnologías para desarrollo de videojuegos en 2D

A continuación se presenta la evaluación realizada para Edulibre. Los detalles de la tecnología sugerida y herramientas útiles se encuentran en el apéndice B.

3.4.2.1. Justificación

Para realizar el desarrollo de un videojuego, independientemente del tipo de este, es necesario utilizar herramientas que puedan facilitar su desarrollo. Para este caso se busca una herramienta de desarrollo de videojuegos ya sea un framework o un Game Engine que pueda facilitar el proceso de desarrollo.

La existencia de muchas opciones en el mercado tanto de pago como gratuitas y de código abierto, hacen necesaria la realización de una evaluación de las diversas opciones existentes. Para encontrar la herramienta idónea que se acople a las necesidades de Edulibre, para su proyecto de apoyo a la educación de los ñiños y ñinas de primaria de Guatemala a través del desarrollo videojuegos educativos.

3.4.2.2. Objetivos de la evaluación

- Establecer varios criterios de selección para las herramientas de desarrollo de videojuegos.
- Escoger al menos tres herramientas que cumplan con los criterios de selección y evaluarlas.
- Definir a partir de la evaluación una herramienta que sea útil y atienda a los requerimientos de desarrollo de videojuegos para Edulibre.
- Realizar la evaluación en base a la Norma ISO/IEC 9126 para calidad del software.

3.4.2.3. Método utilizado

Para evaluar la calidad del software a seleccionar se hizo uso de la Norma ISO/IEC 9126, que establece varios lineamientos que son de ayuda para la

evaluación de calidad de un software en general. El estándar provee una descomposición de las características en subcaracterísticas.

El modelo se basa en los atributos de calidad que se relacionan directamente con la arquitectura: funcionalidad, fiabilidad, usabilidad, eficiencia, matenibilidad. El estándar hace referencia a tres aproximaciones a la calidad del producto, que a continuación se listan.

- Calidad interna: se mide por las propiedades estáticas del código, utilizando técnicas de inspección.
- Calidad externa: se mide por las propiedades dinámicas del código cuando este se ejecuta.
- Calidad en uso: se mide por el grado por el cual el software esta realizado en función de las necesidades del usuario en el entorno de trabajo para el que fue construido.

3.4.2.3.1. Modelo de calidad externa e interna

A continuación se describen los diversos aspectos que se deben tomar en cuenta para hacer la evaluación del software, estos definen tanto la calidad interna como externa del producto.

- Funcionalidad

La Norma, cuando habla de funcionalidad, hace referencia al conjunto de funciones y sus propiedades, que hacen posible cumplir con su propósito, satisfaciendo las necesidades explicitas e implícitas cuando el software se utiliza bajo condiciones específicas.

Tabla IV. **Subcaracterísticas de la funcionalidad**

Adecuación	El software provee un adecuado conjunto de funciones para las tareas y objetivos especificados por el usuario.
Exactitud	El software provee los resultados o efectos acordados con un grado necesario de precisión.
Interoperabilidad	Capacidad del software de interactuar con uno o más sistemas especificados.
Seguridad	Capacidad del software para proteger la información y los datos de accesos no autorizados.

Fuente: elaboración propia.

- Fiabilidad

Hace referencia a la capacidad del software de mantener su nivel de funcionamiento y estabilidad, por un tiempo definido y bajo condiciones específicas. Ya que el software no se desgasta, su fiabilidad puede verse amenazada por fallas de diseño, implementación, hardware, y otras.

Tabla V. **Subcaracterísticas de fiabilidad**

Madurez	Capacidad del software de evitar fallas como resultado de errores en el software.
Tolerancia a errores	Capacidad del software para mantener un adecuado funcionamiento en caso de errores del software o incumplimiento de su interfaz especificada.

Continuación de la tabla V.

Recuperabilidad	Capacidad del software de restablecer un nivel especificado de funcionamiento y recuperar datos afectados directamente en el caso de una falla.

Fuente: elaboración propia.

- Eficiencia

Hace referencia a la capacidad del software de manejar adecuadamente los recursos requeridos por el mismo, a fin de proveer un desempeño adecuado, bajo condiciones planteadas. Entre los recursos pueden ser otros productos de software, algún tipo de hardware específico, y otros.

Tabla VI. **Subcaracterísticas de eficiencia**

Comportamiento de tiempos	El software provee tiempos adecuados de respuesta y procesamiento bajo ciertas condiciones.
Utilización de recursos	Capacidad del software de utilizar cantidades y tipos adecuados de recursos para su óptimo funcionamiento.

Fuente: elaboración propia.

- Mantenibilidad

Hace referencia a la capacidad del software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y especificaciones de requerimientos funcionales.

Tabla VII. **Subcaracterísticas de mantenibilidad**

Capacidad de ser analizado	Capacidad del software para atenerse a diagnósticos de deficiencias y causas de fallas.
Cambiabilidad	Capacidad del software de permitir que una determinada modificación sea implementada.
Estabilidad	Capacidad del software para evitar efectos inesperados debido a modificaciones de hardware.
Facilidad de prueba	Capacidad del software para permitir las modificaciones sean validadas.

Fuente: elaboración propia.

- Portabilidad

Es la capacidad del software para ser trasladado de un entorno a otro. El entorno puede incluir entornos organizacionales, de hardware, o de software.

Tabla VIII. **Subcaracterísticas de portabilidad**

Adaptabilidad	Capacidad del software de adaptarse a diferentes entornos especificados sin aplicar acciones o medios diferentes de los previstos para el propósito del software considerado.
Facilidad de instalación	Capacidad del software de ser instalado en un ambiente especifico.
Coexistencia	Capacidad del software de coexistir con otros productos de software independientes dentro del mismo entorno.
Reemplazabilidad	Capacidad del software para ser utilizado en lugar de otro producto de software, para el mismo propósito y el mismo entorno.

Fuente: elaboración propia.

- Usabilidad

Hace referencia a la capacidad del producto software para ser entendido, aprendido, usado y atractivo al usuario, cuando es utilizado bajo condiciones específicas. Puede que aspectos como funcionalidad, fiabilidad y eficiencia afecten la usabilidad, pero para propósitos de la ISO/IEC 9126 no son clasificados como usabilidad.

Tabla IX. **Subcaracterísticas usabilidad**

Entendimiento	El software permite al usuario entender si este es adecuado, y como puede ser utilizado para las tareas y las condiciones particulares de la aplicación.
Aprendizaje	El software permite al usuario aprender su aplicación sin mucho esfuerzo.
Operabilidad	El software permite al usuario operarlo y controlarlo.
Atracción	Capacidad del software de ser atractivo al usuario.

Fuente: elaboración propia.

- Calidad en uso

Es la capacidad del software de permitirles a usuarios específicos lograr sus metas con eficacia, productividad, seguridad y satisfacción.

Tabla X. **Aspectos de la calidad en uso**

Eficacia	El software permite a los usuarios lograr sus metas con exactitud e integridad.
Productividad	El software permite a los usuarios emplear cantidades razonables de recursos en relación a la eficacia lograda.
Seguridad	El software posee niveles aceptables de riesgo de daño a las personas, software, propiedad, y otros.
Satisfacción	El software es capaz de satisfacer las necesidades del usuario.

Fuente: elaboración propia.

3.4.2.4. Evaluación técnica

La evaluación estará basada en la ISO/IEC 9126 empleando los modelos de calidad descritos anteriormente. Para realizarlo se utiliza una matriz de evaluación en la que se asigna un punteo a cada aspecto, que a su vez que estará divido en sus subcaracterísticas, para dar un total de 100 puntos. El software que más punteo logre obtener será el elegido como ganador de la evaluación.

3.4.2.4.1. Criterios de selección de herramientas

Debido a la gran cantidad de software disponible para desarrollo de videojuegos, no es posible evaluarlos todos. Las herramientas a evaluar tuvieron que llenar los siguientes requerimientos.

- De código abierto y gratuita
- Soporte completo para el desarrollo de videojuegos en 2D
- Puede ser un framework o un Game Engine
- Capacidad de exportar a las plataformas requeridas por Edulibre

3.4.2.4.2. Herramientas seleccionadas

Tras una exhaustiva investigación sobre herramientas para el desarrollo de videojuegos en 2D, habiendo muchas en el mercado, las que cumplieron con los criterios de selección y presentaron más información fueron:

- LibGDX
- Cocos2D JS
- Starling

3.4.2.4.3. Evaluación

Como se mencionó anteriormente, la elección de la herramienta se basará en el punteo que esta obtenga de la evaluación. Es importante mencionar que la evaluación se hace un punto de vista de calidad externa. La tabla siguiente muestra la distribución de la puntuación.

Tabla XI. **Distribución del punteo**

TIPOS DE CALIDAD	CARACETERÍSTICA	SUBCARACTERÍSTICA	PONDERACIÓN
Calidad interna y externa	Funcionalidad	Adecuación	4
		Exactitud	4
		Interoperabilidad	4
		Seguridad	3
	Fiabilidad	Madurez	7
		Tolerancia a errores	4
		Recuperabilidad	4
	Eficiencia	Comportamiento de tiempos	4
		Utilización de recursos	6
	Mantenibilidad	Capacidad de ser analizado	3
		Cambiabilidad	3
		Estabilidad	5
		Facilidad de prueba	4
	Portabilidad	Adaptabilidad	5
		Facilidad de instalación	5
		Coexistencia	3
		Reemplazabilidad	2
	Usabilidad	Entendimiento	4
		Aprendizaje	5
		Operabilidad	3
		Atracción	3
Calidad en uso		Eficacia	4
		Productividad	4
		Seguridad	3
		Satisfacción	4
		Total	**100**

Fuente: elaboración propia.

- Descripción de las características y subcaracterísticas a evaluar
  - Funcionalidad
    - Adecuación: manejo de *assets* (imágenes, audio), físicas y colisiones, despliegue multiplataforma, eventos de usuario.
    - Exactitud: despliegue de imágenes, reproducción de audio, activación de eventos.

- Interoperabilidad: capaz de comunicarse a otro sistema para intercambio de información.
- Seguridad: al ser un framework y servir para construir una aplicación, la seguridad en sí misma también dependería de la aplicación.

- Fiabilidad
  - Madurez: el software tiene que tener historia, ser conocido tener versiones estables.
  - Tolerancia a errores: posee un adecuado manejo de errores, e información sobre ellos para su pronta solución.
  - Recuperabilidad: posee mecanismos para realizar algún tipo de copia de seguridad.
- Eficiencia
  - Comportamiento de tiempos: el procesamiento de imágenes y sonido de los *assets* es adecuado.
  - Utilización de recursos: utiliza una cantidad razonable de recursos del sistema (disco duro, memoria, procesamiento).
- Mantenibilidad
  - Capacidad de ser analizado: el software permite que se pueda analizar en busca de deficiencias.
  - Cambiabilidad: el software permite que sea modificado y luego implementado.
  - Estabilidad: el software trabaja en diferentes configuraciones de hardware, siempre y cuando estos llenen los requisitos.
  - Facilidad de pruebas: el software puede probarse fácilmente si se realizan modificaciones.

- Portabilidad
  - Adaptabilidad: facilidad para establecer el entorno y su utilización.
  - Facilidad de instalación: pocos pasos para su instalación.
  - Coexistencia: capaz de operar sin ningún problema con otros software instalados en el computador.
  - Reemplazabilidad: puede ser utilizado en lugar de otro software en el mismo entorno.
- Usabilidad
  - Entendimiento: es fácil saber si será útil para lo que se pretende realizar con él.
  - Aprendizaje: es fácil su aprendizaje hay mucha documentación, tutoriales, y herramientas extra útiles.
  - Operabilidad: permite hacer lo que se desee y no pone limitantes.
  - Atracción: el software es atractivo, llama la atención su uso.
- Calidad en uso
  - Eficacia: el software realiza lo prometido, y es capaz cumplir en la ayuda de facilitar el desarrollo de los videojuegos.
  - Productividad: facilita la creación de varios componentes del videojuego en un tiempo razonable.
  - Seguridad: el software esta constate actualización, en busca de resolución de fallas, y mejoras.
  - Satisfacción: el nivel de satisfacción.

3.4.2.5. Resultados

A continuación se presentan los resultados obtenidos durante la evaluación de los diferentes frameworks para la creación de videojuegos.

Tabla XII. **Tabla de resultados**

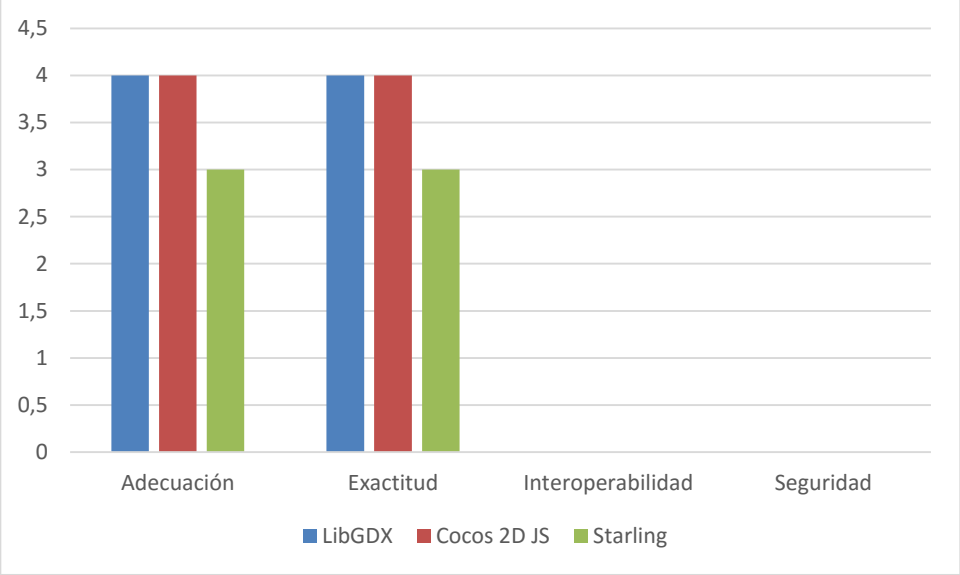
TIPOS DE CALIDAD	CARACTERÍSTICA	SUBCARACTERÍSTICA	PONDERACIÓN	Software a Evaluar					
				LibGDX		Cocos 2D JS		Starling	
				Evaluado	Puntaje	Evaluado	Puntaje	Evaluado	Puntaje
Calidad interna y externa	Funcionalidad	Adecuación	4	Si	4	Si	4	Si	3
		Exactitud	4	Si	4	Si	4	Si	3
		Interoperabilidad	4	No	0	No	0	No	0
		Seguridad	3	No	0	No	0	No	0
	Fiabilidad	Madurez	7	Si	7	Si	7	Si	6
		Tolerancia a errores	4	Si	4	Si	4	Si	4
		Recuperabilidad	4	Si	4	Si	4	Si	4
	Eficiencia	Comportamiento de tiempos	4	Si	3	Si	3	Si	2
		Utilización de recursos	6	Si	6	Si	6	Si	4
	Mantenibilidad	Capacidad de ser analizado	3	Si	3	Si	3	Si	3
		Cambiabilidad	3	Si	3	Si	3	Si	3
		Estabilidad	5	Si	5	Si	5	Si	4
		Facilidad de prueba	4	Si	4	Si	4	Si	2
	Portabilidad	Adaptabilidad	5	Si	5	Si	4	Si	5
		Facilidad de instalación	5	Si	5	Si	3	Si	3
		Coexistencia	3	Si	3	Si	3	Si	3
		Reemplazabilidad	2	No	0	No	0	No	0
	Usabilidad	Entendimiento	4	Si	4	Si	3	Si	2
		Aprendizaje	5	Si	5	Si	3	Si	3
		Operabilidad	3	Si	3	Si	3	Si	2
		Atracción	3	Si	3	Si	3	Si	3
Calidad en uso		Eficacia	4	Si	4	Si	3	Si	3
		Productividad	4	Si	4	Si	3	Si	3
		Seguridad	3	Si	3	No	3	No	3
		Satisfacción	4	Buena	3	Buena	3	Regular	3
		Total	100		89		81		71

Fuente: elaboración propia.

3.4.2.5.1. Gráficas

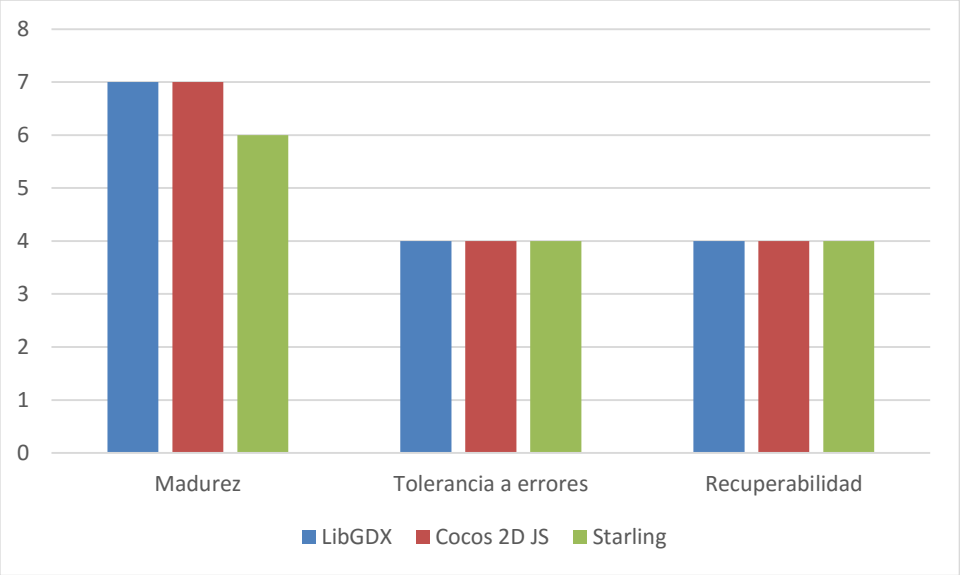
A continuación se presentan una serie de graficas que representan cada aspecto evaluado de las distintas herramientas seleccionadas, según la Norma ISO/IEC 9126.

Figura 24. **Funcionalidad**



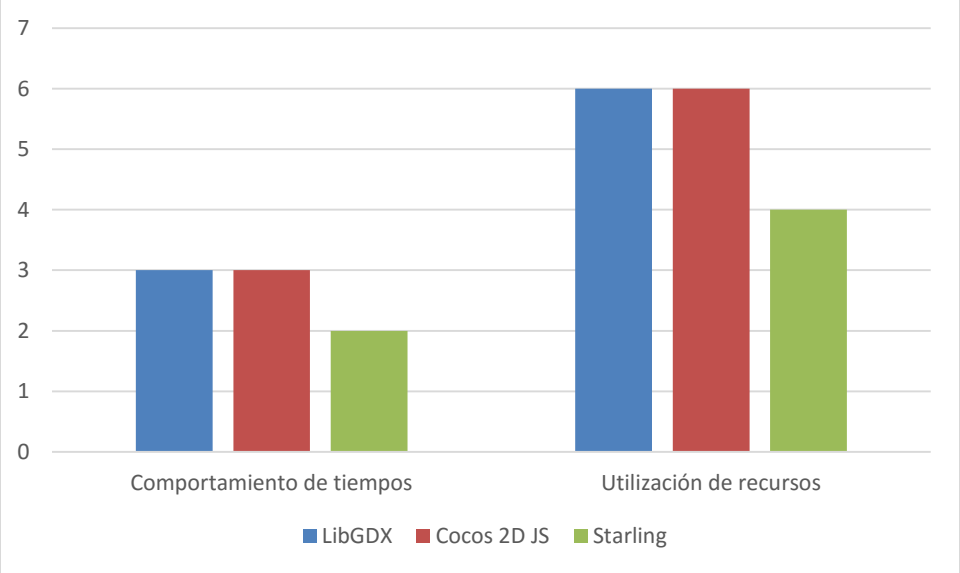
Fuente: elaboración propia.

Figura 25. **Fiabilidad**



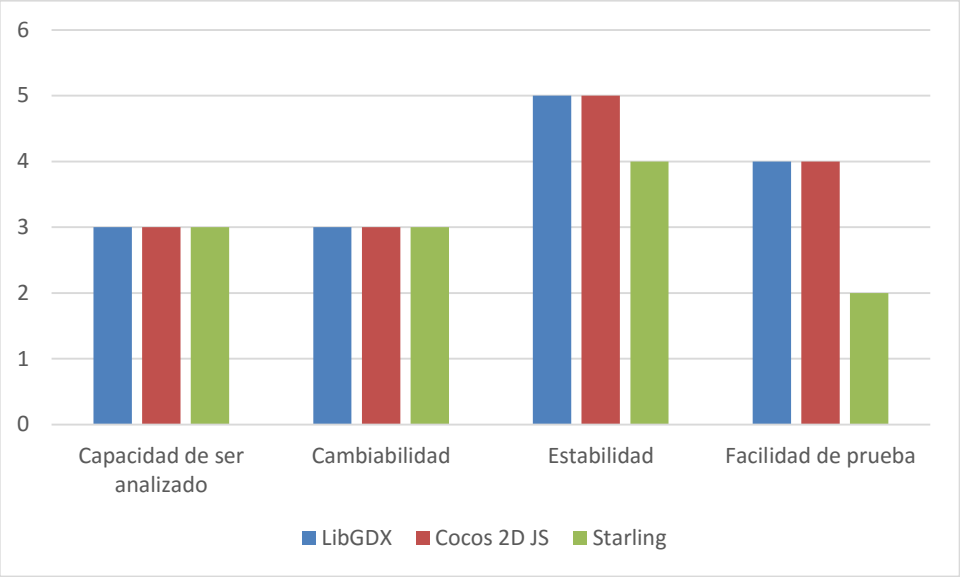
Fuente: elaboración propia.

Figura 26. **Eficiencia**



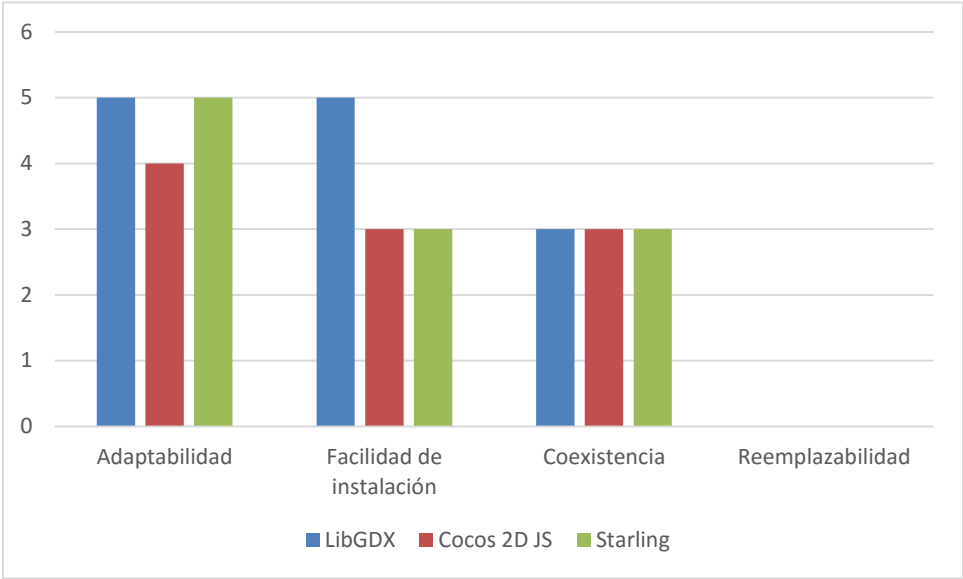
Fuente: elaboración propia.

Figura 27. **Mantenibilidad**



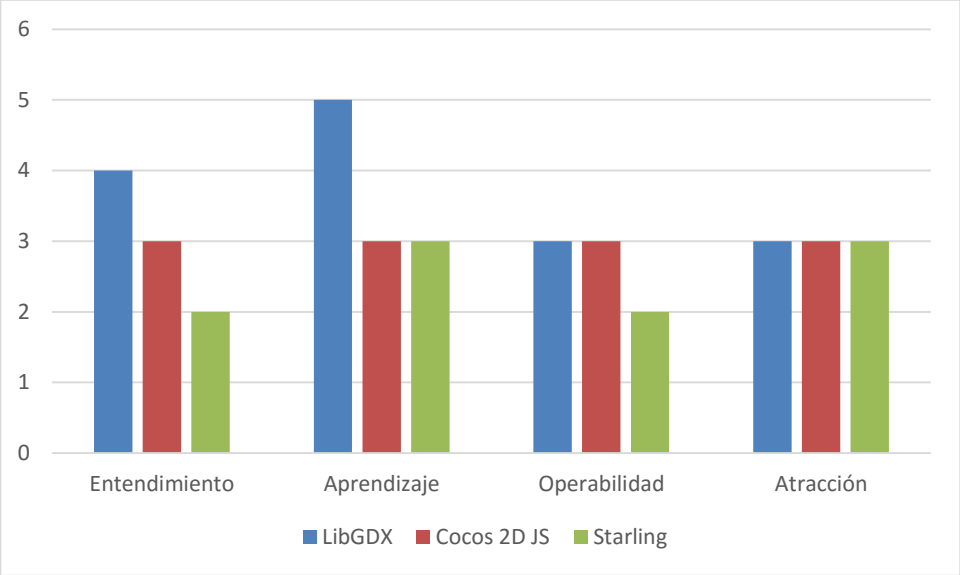
Fuente: elaboración propia.

Figura 28. **Portabilidad**



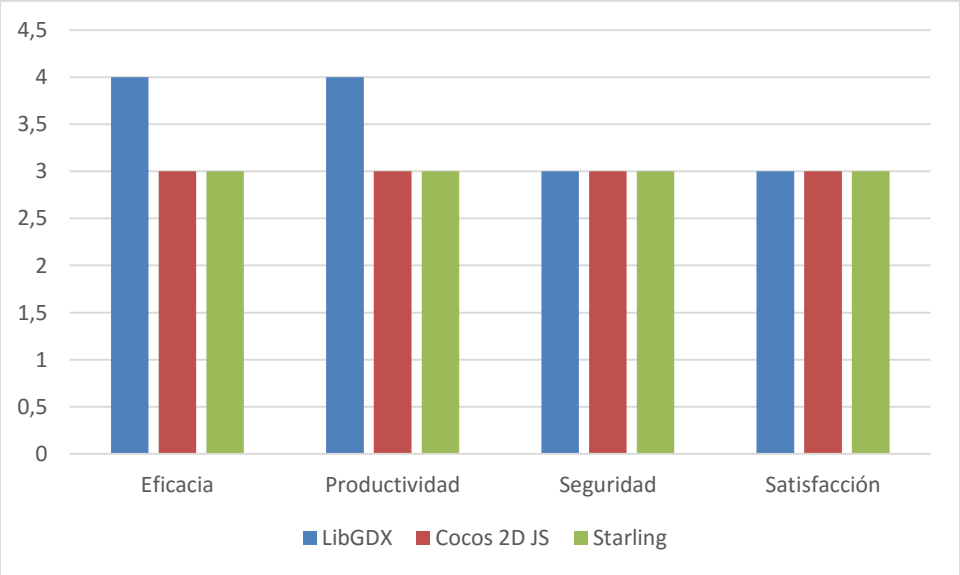
Fuente: elaboración propia.

Figura 29. **Usabilidad**



Fuente: elaboración propia.

Figura 30. **Calidad en uso**



Fuente: elaboración propia.

3.4.2.6. Conclusiones

Tras la evaluación de los tres frameworks para desarrollo de videojuegos especializados en 2D, cada uno tiene sus fortalezas y debilidades. LibGDX tuvo una mejor puntación seguido muy de cerca por Cocos2D JS y más lejos de *Starling*. Cabe aclarar antes de continuar, que estas conclusiones son con base en la propia experiencia del investigador con los frameworks, puede que para otro desarrollador esto no sea así.

LibGDX obtuvo mayor puntuación porque es bastante fácil su instalación y configuración en comparación con Cocos2D JS y Starling, donde resultó más complicado debido a la configuración de mucho más software adicional para lograr utilizar las librerías. LibGDX además obtuvo buena puntación en los diferentes aspectos de calidad evaluados, en comparación con los otros. También utiliza Java un lenguaje de programación muy conocido. Provee soporte, hay documentación actualizada y muchos ejemplos que pueden ser útiles.

Esto no quiere decir que las otras herramientas no pueden utilizarse. Al contrario son otras opciones que están disponibles para tomar en cuenta su uso, y en las cuales también se pueden hacer muchas cosas.

3.4.3. Administración de la configuración y evaluación de repositorios en la nube

Lo siguiente es un resumen de la solución planteada a Edulibre. La descripción completa se encuentra en el apéndice C.

3.4.3.1. Administración de la configuración

En el transcurso de la fase de desarrollo, y más si se trabaja en equipo debe haber una estructura que permita, de alguna manera manejar los diferentes ambientes, versiones del código y el software. El propósito de establecer una adecuada administración de la configuración es identificar el estado del videojuego educativo en distintos puntos en el tiempo, con el propósito de controlar de manera sistemática los cambios a la configuración y mantener la integridad de dicha configuración a lo largo del ciclo de vida del videojuego.

3.4.3.1.1. Política de versionado de los videojuegos educativos

Todos los videojuegos educativos desarrollados en Edulibre deberán seguir esta política, al momento de ser lanzados y distribuidos por los medios que Edulibre haya decidido, en la planificación del proyecto.

- Numeración de versiones

Existen diversas técnicas para la numeración de versiones de software, proponiendo para los proyectos de Edulibre la más extendida dentro de los proyectos de software libre. Es una notación numérica compuesta por tres números, con algunas variantes. Cabe aclarar que los números siempre van en orden creciente, y son enteros no negativos.

Figura 31. **Forma de la numeración**

Fuente: elaboración propia.

- vX: es un número entero que indica una versión principal del videojuego educativo, consistiendo en un conjunto de funcionalidades cubiertas en dicha versión, y que son visibles al usuario. La letra “v” se coloca por comodidad para indicar que se trata de una versión de software, por ejemplo v1.2.4. Si este número incrementa “Y” y “Z” se vuelven cero.
- Y: es un número entero que indica un conjunto nuevas funcionalidades agregadas al videojuego. Si este número incrementa “Z” se vuelve cero. En el desarrollo se podría tomar como referencia la iteración realizada.
- Z: es un número entero que se modifica cuando hay revisiones de código ante fallos encontrados en el videojuego, y no funcionalidad nueva. Este número se puede incrementar indefinidamente, siempre y cuando “Y” no haya cambiado, de lo contrario vuelve a cero.
- [beta]: esta etiqueta solo se utilizará para las versiones beta, que están sometidas a pruebas e indica que es una versión inestable. La versión de lanzamiento o estable no tendrá esta etiqueta.

3.4.3.1.2. Administración de repositorios

Es importante establecer un patrón de trabajo a nivel de la institución, a seguir por todos los involucrados en el desarrollo de los videojuegos, a modo realizar más fácilmente la tarea, y evitar complicaciones. El modelo de ramas es el propuesto por Vincent Driessen en su artículo titulado “A successful Git

branching model” con algunos agregados. Es un modelo muy utilizado para diversos proyectos, utilizando la tecnología Git para el versionamiento.

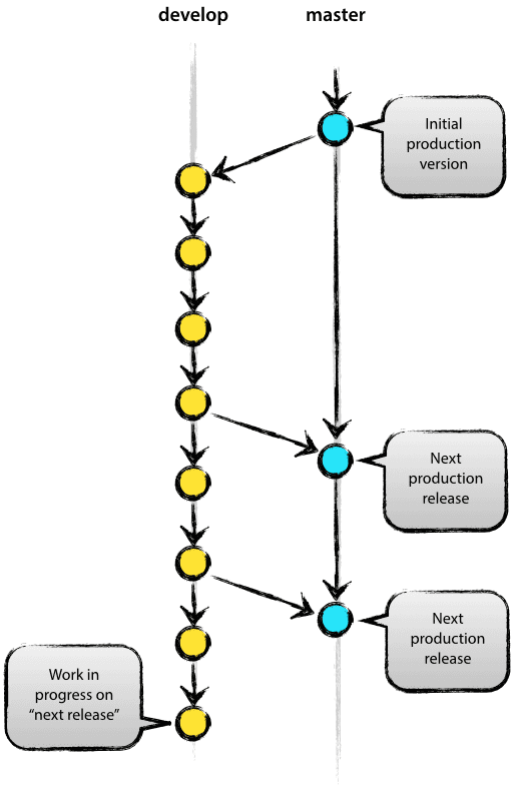
- Modelo de ramas

Todo videojuego educativo que se realice, deberá seguir el siguiente patrón de trabajo para los repositorios. La creación de estas ramas puede realizarse de forma manual o bien utilizando la herramienta *git-flow* disponible para los diferentes sistemas operativos.

Existirán dos ramas principales en el repositorio remoto (*Bitbucket*):

- Rama *master*: esta es la rama principal que contiene la aplicación lista producción. Esta rama contiene la última versión estable del videojuego educativo.
- Rama *develop*: en esta rama se encuentra todo el código que conformará la siguiente versión planificada del proyecto. Y del cual se desprende un *release* candidato para las pruebas beta.

Figura 32. **Ramas principales**



Fuente: *A successful Git branching model*. http://nvie.com/img/main-branches@2x.png. Consulta: 09 de enero de 2015.

- Ramas auxiliares

La idea principal del uso de ramas auxiliares es ayudar al desarrollo en paralelo entre los miembros del equipo. Esto es facilitar la implementación de requerimientos, preparar el software beta para producción y asistir de forma rápida a la reparación de fallos.

Cada una de estas ramas tiene un propósito en específico y están sujetas a estrictas normas de funcionamiento. Es importante mencionar que son

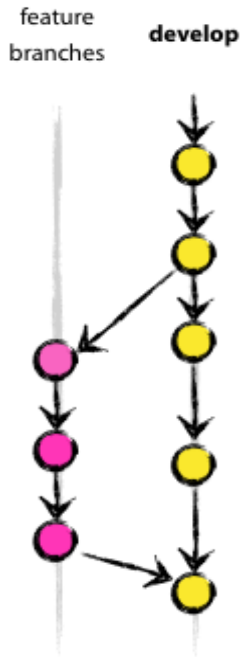
sugeridas y que se pueden agregar o quitar según se necesite, pero estas suelen ser las más usuales.

- Ramas de características (*feature*)

Estas ramas:

- Se originan a partir de la rama *develop*.
- Se incorporan siempre a la rama *develop*.
- Nombre: feature-*. Donde * es el nombre de la funcionalidad a implementar.

Figura 33. **Ramas de características**



Fuente: *A successful Git branching model*. http://nvie.com/img/fb@2x.png. Consulta: 09 de enero de 2015.

Estas ramas se utilizan para desarrollar las nuevas características serán parte del próximo *release*, que una vez terminadas, se incorporan a la rama *develop*. La esencia de una rama de característica es que existe, siempre y cuando la característica está en desarrollo, pero finalmente se fusiona de nuevo en *develop* o bien se descarta. Comúnmente cuando la rama se fusiona con *develop* esta es eliminada, ya que su utilidad terminó.

- Ramas de *release* (*release*)

Estas ramas:

- Se originan a partir de la rama *develop*
- Se incorporan o fusionan a *master* y *develop*
- Nombre: vX.Y.Zbeta (revisar política de versionamiento)

Estas ramas se utilizan para preparar el siguiente código que pasará a producción. En estas ramas se hacen los últimos ajustes y se corrigen los últimos pequeños *bugs* antes de pasar el código a producción incorporándolo a la rama *master* y luego a la rama *develop*. Realizando todo el trabajo sobre esta rama, la rama *develop* está lista para recibir nuevas características que darán paso a la próximo *release*.

El pasar a esta rama significa que la rama *develop* ya llegó a un punto en el cual se le considera casi estable, es decir que casi todas las características han sido implementadas, (el conjunto de características suelen pertenecer a una iteración del software).

- Ramas de revisiones (*hotfixes*)

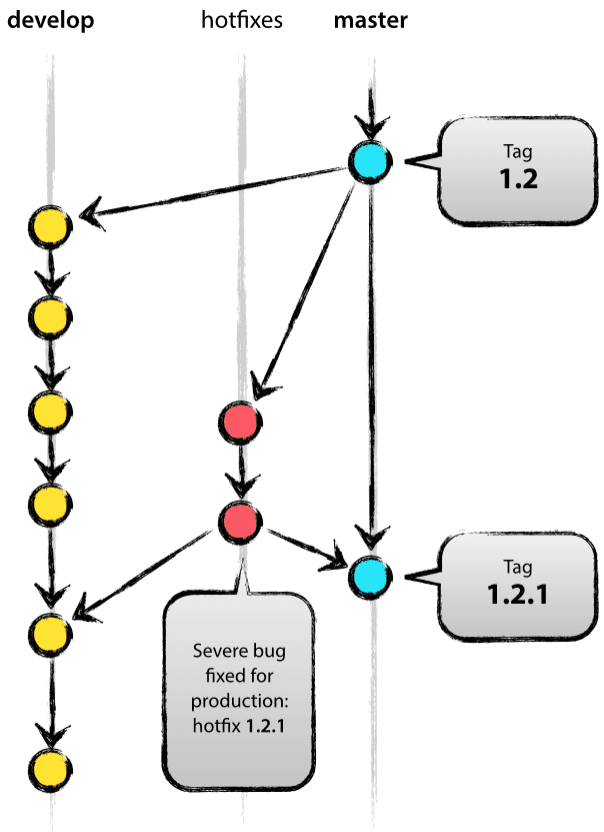
Estas ramas:

- Tienen su origen a partir de la rama *master*.
- Se fusionan o incorporan a la rama master y *develop*.
- Nombre: hotfix-[vX.Y.Z]. Donde lo que está entre corchetes es la versión a la que se le aplica el parche. Los corchetes no son parte del nombre.

Estas ramas son como las ramas de *release*, en las que también se prepara la aplicación para producción, con la diferencia que estas ramas no son planeadas. Surgen de la necesidad de resolver algún *bug* crítico del software, que está afectando el comportamiento de la versión actual en producción. Cuando un *bug* crítico debe ser resuelto, esta rama debe surgir de la versión actual correspondiente en la rama *master*.

En esencia el trabajo del equipo de desarrollo en la rama *develop*, no debe verse afectado, mientras otra persona resuelve un fallo en la versión de producción.

Figura 34. **Rama de revisiones**



Fuente: *A successful Git branching model*. http://nvie.com/img/hotfix-branches@2x.png. Consulta: 11 de enero de 2015.

No hay que olvidarse de modificar el número de versión después de la ramificación, a la rama *master*. Una vez la rama de revisión fue fusionada con *master* y *develop*, se puede eliminar ya que no será de utilidad.

Una observación importante, es que si cuando se creó la rama de revisión ya existía una rama de *release*, el arreglo debe incorporarse a esta, ya que de todos modos los arreglos hechos al *release*, y los incorporados por la revisión deben de fusionarse con la rama *develop*.

3.4.3.2. Evaluación de repositorios en la nube

A continuación se presenta la evaluación de los diferentes repositorios aptos para el almacenamiento y control de versiones de los videojuegos educativos.

3.4.3.2.1. Justificación

En proyectos de desarrollo de software independientemente del tipo que sean, cuando intervienen varias personas en su construcción se vuelve necesario llevar un control de los cambios que este sufre durante todo ese proceso. Para llevar ese control es necesario la utilización de alguna herramienta que pueda facilitar esa tarea.

En el ámbito de repositorios en la nube existen diversas opciones tanto con planes gratuitos como de pago; lo que hace necesaria la realización de una evaluación entre las diferentes opciones existentes, y así encontrar el repositorio idóneo para albergar de forma segura los proyectos de videojuegos de Edulibre.

3.4.3.2.2. Objetivos de la evaluación

- Establecer varios criterios de selección para las opciones de repositorios en la nube.
- Escoger al menos tres opciones que cumplan con los criterios de selección y evaluarlas.
- Definir a partir de la evaluación un repositorio que sea útil y atienda a los requerimientos de desarrollo de videojuegos para Edulibre.
- Realizar la evaluación con base en la Norma ISO/IEC 9126 para calidad del software.

3.4.3.2.3. Método utilizado

Para evaluar la calidad del software a seleccionar se hizo uso de la Norma ISO/IEC 9126, que establece varios lineamientos que son de ayuda para la evaluación de calidad de un software en general. El estándar provee una descomposición de las características en subcaracterísticas. Para mayor información del estándar este se encuentra desglosado en el apartado de Evaluación de Tecnologías para Desarrollo de Videojuegos en 2D.

3.4.3.2.4. Evaluación técnica

La evaluación estará basada en la ISO/IEC 9126 empleando los modelos de calidad descritos anteriormente, para realizarlo se utiliza una matriz de evaluación en la que se asigna un punteo a cada aspecto, que a su vez que estará divido en sus subcaracterísticas, para dar un total de 100 puntos. El software que más punteo logre obtener será el elegido como ganador de la evaluación. Similar al utilizado para la Evaluación de Tecnologías para Desarrollo de Videojuegos.

3.4.3.2.5. Criterios de selección de herramientas

Debido a que en el mercado hay varias opciones, no es posible evaluarlos todos. Las herramientas a evaluar tuvieron que llenar los siguientes requerimientos.

- Soporte de *git* como tecnología de versionamiento
- Reconocimiento del lenguaje de programación Java
- Manejo de repositorios privados y públicos gratuitos

- Herramientas seleccionadas

Después de una exhaustiva investigación sobre repositorios en la nube para albergar los proyectos de Edulibre, habiendo diversas opciones en el mercado, las que cumplieron con los criterios de selección y presentaron más información fueron.

- Bitbucket
- GitLab
- GitGo

3.4.3.2.6. Evaluación

Para la evaluación, la elección del gestor de repositorios en la nube se basará en el punteo que esta obtenga. Es importante mencionar que la evaluación se hace desde un punto de vista de calidad externa. La tabla de la siguiente página muestra la distribución de la puntuación.

- Observaciones

Para la matriz de evaluación en la columna de evaluación si en esta se encuentra la palabra “No aplica” o “No”, es porque esa subcaracterísticas no se puede evaluar o no es posible aplicarla. Un ejemplo de esto puede ser una aplicación web, que no se le puede evaluar la instalación, ya que esta solo para usarse, y no hay que instalar nada para usarla. Debido a que se encuentra alojada en un servidor, y es accesible a través de un explorador de internet.

Tabla XIII. **Distribución de punteo**

TIPOS DE CALIDAD	CARACETERÍSTICA	SUBCARACTERÍSTICA	PONDERACIÓN
Calidad interna y externa	Funcionalidad	Adecuación	4
		Exactitud	4
		Interoperabilidad	4
		Seguridad	3
	Fiabilidad	Madurez	7
		Tolerancia a errores	4
		Recuperabilidad	4
	Eficiencia	Comportamiento de tiempos	4
		Utilización de recursos	6
	Mantenibilidad	Capacidad de ser analizado	3
		Cambiabilidad	3
		Estabilidad	5
		Facilidad de prueba	4
	Portabilidad	Adaptabilidad	5
		Facilidad de instalación	5
		Coexistencia	3
		Reemplazabilidad	2
	Usabilidad	Entendimiento	4
		Aprendizaje	5
		Operabilidad	3
		Atracción	3
Calidad en uso		Eficacia	4
		Productividad	4
		Seguridad	3
		Satisfacción	4
		Total	**100**

Fuente: elaboración propia.

- Descripción de las características y subcaracterísticas a evaluar
  - Funcionalidad
    - Adecuación: manejo de repositorios privados y públicos, gestión de usuarios, almacenamiento en la nube.
    - Exactitud: despliegue de información de repositorios, accesibilidad desde internet.

- Interoperabilidad: capaz de comunicar se a otro sistema para intercambio de información.
- Seguridad: gestión de usuarios y permisos para la colaboración en los repositorios.

- Fiabilidad
  - Madurez: el software tiene que tener historia, ser conocido, actualización automática.
  - Tolerancia a errores: posee un adecuado manejo de errores, e información sobre ellos para su pronta solución.
  - Recuperabilidad: posee mecanismos para realizar algún tipo de copia de seguridad.
- Eficiencia
  - Comportamiento de tiempos: con una conexión adecuada y estable a internet.
  - Utilización de recursos: al ser una aplicación que se maneja en la nube a través de una página web, esta no aplica.
- Mantenibilidad
  - Capacidad de ser analizado: el software permite que se pueda analizar en busca de deficiencias.
  - Cambiabilidad: el software permite que se sea modificado y luego implementado.
  - Estabilidad: el software trabaja en diferentes configuraciones de hardware, siempre y cuando estos llenen los requisitos.
  - Facilidad de prueban: el software puede probarse fácilmente si se realizan modificaciones.

- Portabilidad
  - Adaptabilidad: facilidad para establecer el entorno y su utilización.
  - Facilidad de instalación: pocos pasos para su instalación.
  - Coexistencia: capaz de operar sin ningún problema con otros software instalados en el computador.
  - Reemplazabilidad: puede ser utilizado en lugar de otro software en el mismo entorno.
- Usabilidad
  - Entendimiento: es fácil saber si será útil para lo que se pretende realizar con él.
  - Aprendizaje: es fácil su aprendizaje, hay mucha documentación, tutoriales, y herramientas extra útiles.
  - Operabilidad: permite hacer lo que se desee y no pone limitantes.
  - Atracción: el software es atractivo, llama la atención su uso.
- Calidad en uso
  - Eficacia: el software realiza lo prometido, y es capaz cumplir en la ayuda de facilitar el control de versionamiento.
  - Productividad: facilita la creación de varios componentes de un repositorio en un tiempo razonable.
  - Seguridad: el software esta constate actualización, en busca de resolución de fallas, y mejoras.
  - Satisfacción: el nivel de satisfacción.

3.4.3.2.7. Resultados

A continuación se presentan los resultados obtenidos durante la evaluación de los diferentes repositorios en la nube, para el versionamiento de los videojuegos.

Tabla XIV. **Tabla de resultados**

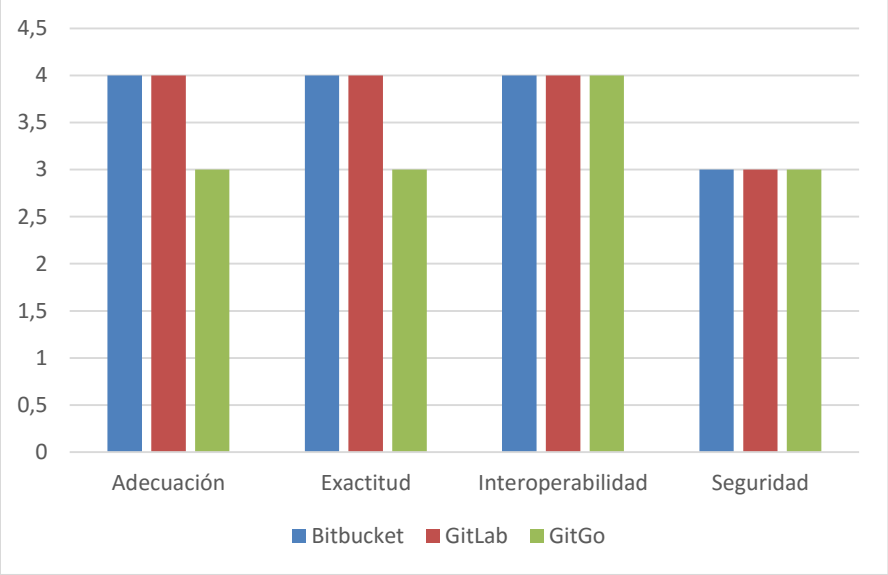
TIPOS DE CALIDAD	CARACETERÍSTICA	SUBCARACTERÍSTICA	PONDERACIÓN	Software a Evaluar					
				Bitbucket		GitLab		GitGo	
				Evaluado	Puntaje	Evaluado	Puntaje	Evaluado	Puntaje
Calidad interna y externa	Funcionalidad	Adecuación	4	Si	4	Si	4	Si	3
		Exactitud	4	Si	4	Si	4	Si	3
		Interoperabilidad	4	Si	4	Si	4	Si	4
		Seguridad	3	Si	3	Si	3	Si	3
	Fiabilidad	Madurez	7	Si	7	Si	7	Si	6
		Tolerancia a errores	4	Si	4	Si	4	Si	4
		Recuperabilidad	4	Si	4	Si	4	Si	4
	Eficiencia	Comportamiento de tiempos	4	Si	3	Si	3	Si	3
		Utilización de recursos	6	No aplica	0	No aplica	0	No aplica	0
	Mantenibilidad	Capacidad de ser analizado	3	No aplica	0	No aplica	0	No aplica	0
		Cambiabilidad	3	No aplica	0	No aplica	0	No aplica	0
		Estabilidad	5	Si	5	Si	5	Si	4
		Facilidad de prueba	4	Si	4	Si	4	Si	2
	Portabilidad	Adaptabilidad	5	Si	5	Si	4	Si	5
		Facilidad de instalación	5	No aplica	0	No aplica	0	No aplica	0
		Coexistencia	3	No aplica	0	No aplica	0	No aplica	0
		Reemplazabilidad	2	No	0	No	0	No	0
	Usabilidad	Entendimiento	4	Si	4	Si	3	Si	3
		Aprendizaje	5	Si	5	Si	4	Si	4
		Operabilidad	3	Si	3	Si	3	Si	3
		Atracción	3	Si	3	Si	2	Si	2
Calidad en uso		Eficacia	4	Si	4	Si	4	Si	4
		Productividad	4	Si	4	Si	4	Si	4
		Seguridad	3	Si	3	No	3	No	3
		Satisfacción	4	Buena	4	Regular	3	Regular	3
		Total	100		77		72		67

Fuente: elaboración propia.

3.4.3.2.8. Gráficas

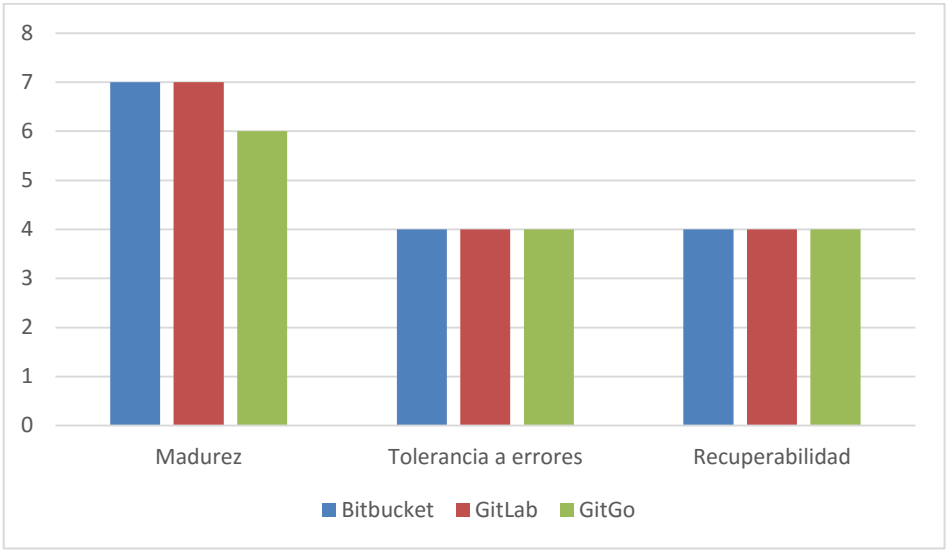
A continuación se presentan una serie de gráficas que representan cada aspecto evaluado de las distintas herramientas seleccionadas, según la Norma ISO/IEC 9126.

Figura 35. **Funcionalidad**



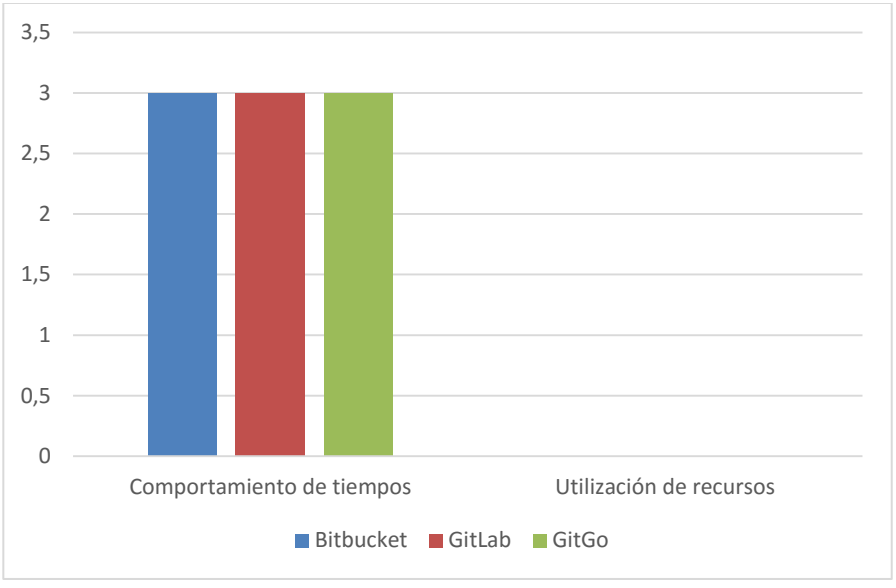
Fuente: elaboración propia.

Figura 36. **Fiabilidad**



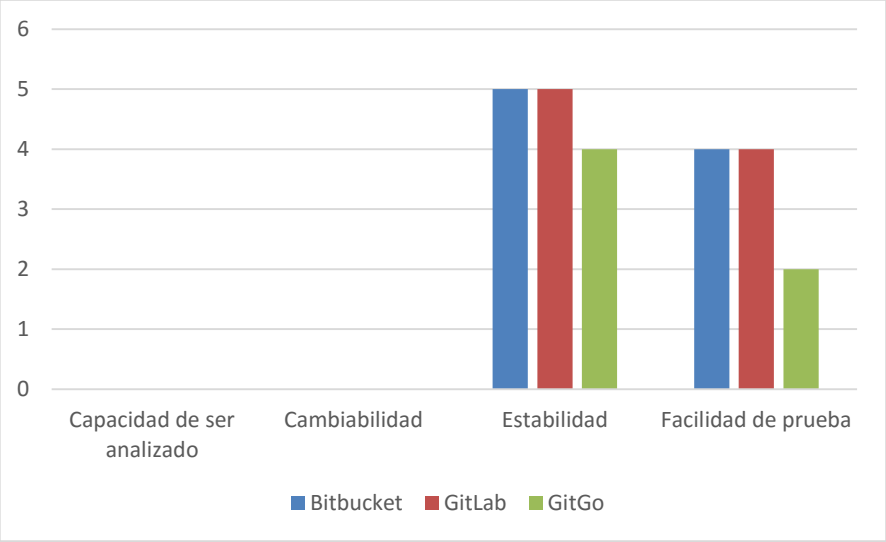
Fuente: elaboración propia.

Figura 37. **Eficiencia**



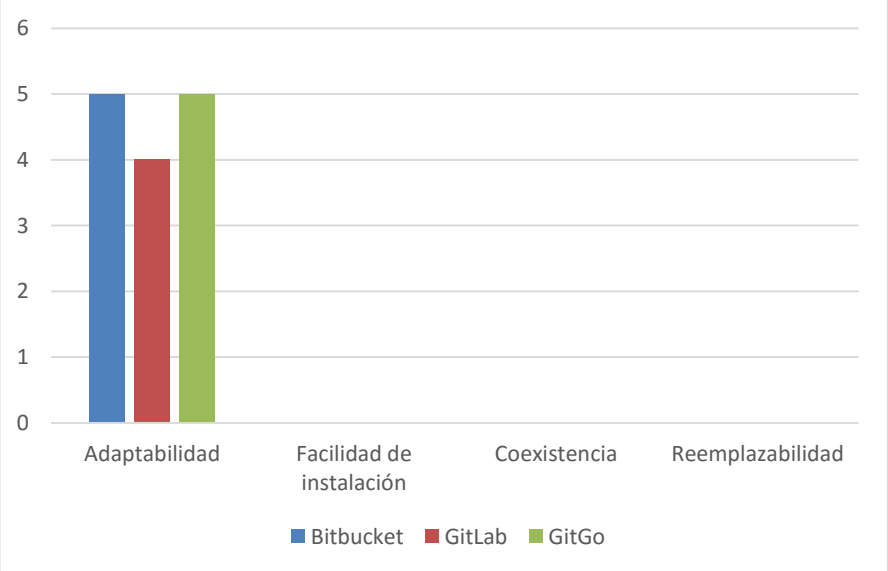
Fuente: elaboración propia.

Figura 38. **Mantenibilidad**



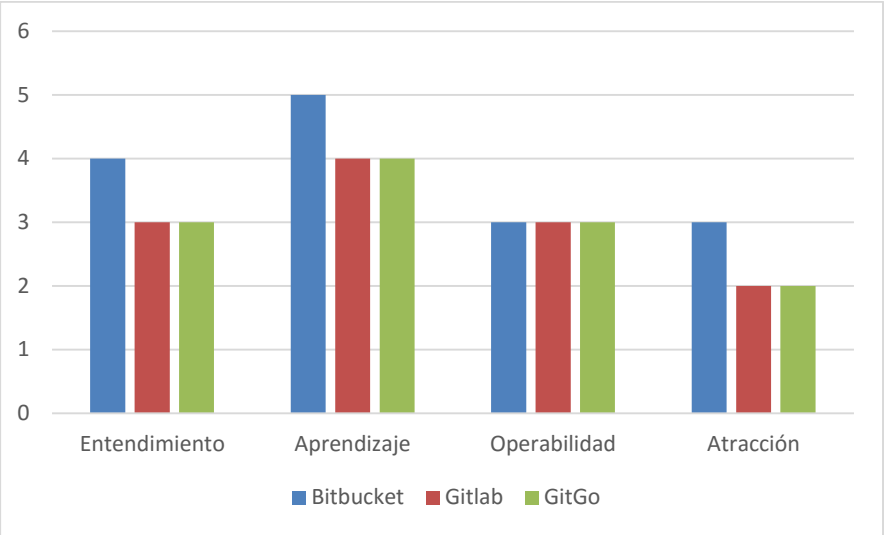
Fuente: elaboración propia.

Figura 39. **Portabilidad**



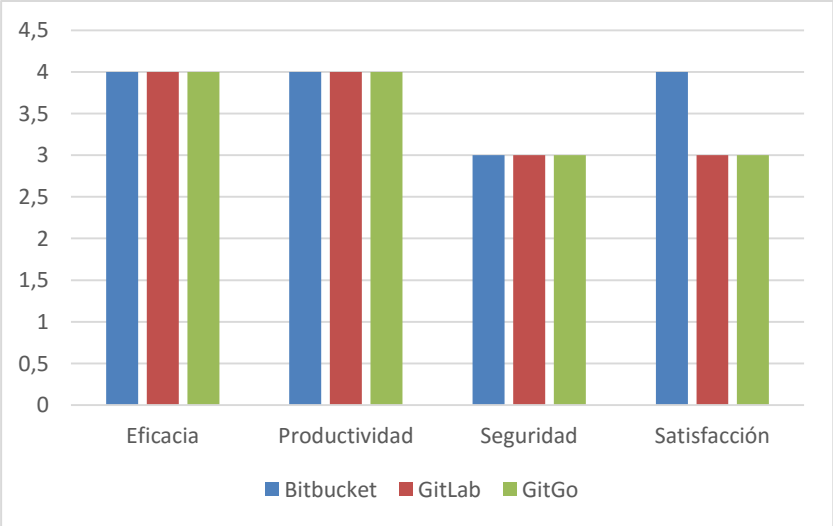
Fuente: elaboración propia.

Figura 40. **Usabilidad**



Fuente: elaboración propia.

Figura 41. **Calidad en uso**



Fuente: elaboración propia.

3.4.3.2.9. Conclusiones

Los resultados de la evaluación como se puede apreciar son muy cerrados, pero el ganador fue *Bitbucket*. Es importante mencionar los resultados de esta evaluación fueron con base en la experiencia del investigador.

Debido a que cada *hosting* de repositorios maneja un modelo de negocio distinto dificultó un poco la elección, ya que unos lo hacen con base en los usuarios, otros al número de repositorios, otros al espacio de almacenamiento disponible, entre otros. Una de las virtudes de *Bitbucket* en comparación con GitLab o GitGo es su amigabilidad al ser bastante instintivo para usarlo, aumentando considerablemente su calidad en uso.

Otra opción bastante viable es GitLab que aunque no es muy conocido también posee cualidades que lo hacen también elegible para ser usado como repositorio para los videojuegos en comparación con GitGo que pone una limitante de espacio de almacenamiento (100 Mb) que no permitiría albergar muchos proyectos.

Bitbucket ofrece espacio ilimitado, repositorios ilimitados y un máximo de 5 usuarios para una cuenta gratuita. Esta característica principalmente junto con el hecho de que es muy popular, y hay bastante información lo hicieron destacar mucho para su elección.

3.5. Costos del proyecto

A continuación se presenta un resumen de los recursos y los costos incurridos para realización del proyecto.

3.5.1. Recursos humanos

- Lic. Freddy Lara, director general de la Asociación Civil Edulibre y asesor de la institución.
- Inga. Ivonne Aldana, asesora del proyecto.
- Juan Daniel Pineda Cabrera, desarrollador del proyecto.

3.5.2. Recursos materiales

Para la realización del proyecto se utilizarón recursos propios, entre los cuales destacan.

- Computador portátil Toshiba Satellite L305 con procesador Intel Centrino Duo de 2Ghz con 4GB de RAM y 250GB de disco duro.
- Utilidades de oficina (lapiceros, lápiz, papel, etc).
- Impresora.
- Internet.
- Libros.

3.5.3. Costos

A continuación se presenta una tabla con el resumen de los costos incurridos en la realización del proyecto.

Tabla XV. **Costos del proyecto**

Recursos	**Cantidad**	**Costo Unitario**	**Subtotal**
Transporte público a institución	10 días	Q9,00	Q90,00
Transporte público a lugar de trabajo del asesor proyecto	10 días	Q16,00	Q160,00
Energía eléctrica	133 días	Q30,00	Q3 990,00
Teléfono	10 días	Q25,00	Q250,00
Internet	133 días	Q12,00	Q1 596,00
Materiales de oficina	-	Q350,00	Q350,00
Impresiones	300 impresiones	Q0,25	Q75,00
Asesor del proyecto	20 días	Q200,00	Q4 000,00
Asesor de la institución	10 días	Q150,00	Q1 500,00
Desarrollo del proyecto	133 días	Q200,00	Q26 600,00
		Costo total	Q38 661,00

Fuente: elaboración propia.

3.6. Beneficios del proyecto

- La obtención de una metodología de desarrollo que permitirá a Edulibre la creación de videojuegos educativos, en la cual con su aplicación podrán ver, evaluar y documentar su evolución.

- La recomendación de una herramienta de desarrollo de videojuegos, en función de sus necesidades, por medio de una evaluación de las diversas opciones en el mercado. Estas serán de código abierto y que cumplan con dar soporte a varias plataformas de destino.
- Una guía para la administración de la configuración para llevar el control de versiones de los videojuegos que se desarrollen, a través de varios lineamientos propuestos en el documento para facilitar la tarea.

4. FASE DE ENSEÑANZA APRENDIZAJE

4.1. Capacitación propuesta

En cada documento de avances del desarrollo del proyecto se les expuso a las personas de Edulibre, especialmente al director general de la institución en funciones lic. Freddy Lara, la investigación realizada y se explicó cada una de las propuestas hechas, las cuales duraban treinta minutos aproximadamente. En la última reunión con representantes de la institución, incluido el director general a la fecha el ing. Javier Hernández se realizó la última exposición de las propuestas la cual duro aproximadamente una hora.

Al ser trabajo de análisis e investigación y no de desarrollo (ya que no hay un software hecho propiamente), no había en sí un usuario que capacitar aún, ya que este EPS servirá de base a otro que realizará el prototipo piloto utilizando las propuestas realizadas en la investigación. Por lo cual se le ofreció a Edulibre la posibilidad contactar al investigador, para la resolución de dudas por parte de la persona que realice el prototipo.

4.2. Material elaborado

Los documentos en sí son los manuales a seguir para echar a andar el marco de trabajo, cada uno de ellos contiene instrucciones específicas de cómo proceder. En los apéndices A, B, y C se pueden ver a detalle.

CONCLUSIONES

1. Se desarrolló como parte del marco de trabajo un documento que contiene una propuesta de metodología de desarrollo de software orientada a la creación de videojuegos educativos, tomando como base las metodologías de desarrollo ágil con modificaciones para el propósito.

2. Se establecieron los roles necesarios que deben estar presente en el equipo de trabajo del proyecto para realizar el videojuego.

3. Como documentación de apoyo a la metodología se realizaron una serie de plantillas para ayudar a hacer mejor el trabajo con la metodología propuesta.

4. Se realizó un documento que contiene la evaluación de tres herramientas para el desarrollo de videojuegos en dos dimensiones de código abierto y con soporte para PC con GNU/Linux, Web y dispositivos móviles con Android, quedando como ganador de la evaluación LibGDX.

5. Se realizó un documento que contiene la administración de la configuración para el manejo del código de los videojuegos y además la evaluación de tres opciones de repositorios en la nube, para el control de versiones.

RECOMENDACIONES

1. Hacer una prueba piloto en la cual se use la metodología de desarrollo propuesta, la herramienta de desarrollo de videojuegos seleccionada y la utilización de la administración de la configuración con la herramienta seleccionada, con el fin de mejorar el marco de trabajo.

2. Cada documento realizado contiene una sección de recomendaciones, por lo que es preciso tomarlas en cuenta a la hora de realizar las diversas tareas propuestas en cada uno.

BIBLIOGRAFÍA

1. ACOSTA, J. *Manejo y administración de branches en GIT.* [en línea]. <http://www.solvetic.com/tutoriales/article/1371-manejo-y-administracion-de-branches-en-git/ >. [Consulta: 03 de febrero de 2015].

2. ALBA, Alfonso. *git-flow: la rama develop y uso de feature branches* [en línea]. < http://aprendegit.com/git-flow-la-rama-develop-y-uso-de-feature-branches/>. [Consulta: 03 de febrero de 2015].

3. *Análisis y Diseño de Sistemas I*. [en línea]. <https://www.dropbox.com/s/cp5vrwxfrt9070c/Libro_Analisis_Disenio1.pdf?dl=0>. [Consulta: 09 de octubre de 2014].

4. ARELLANO, Gerardo. *Tiled + Box2D Parte 1*. [en línea]. <http://tutoriales.tiarsoft.com/2014/10/tiled-box2d-parte-1.html>. [Consulta: 28 de diciembre de 2014].

5. BADLOGICGAMES. *LibGDX*. [en línea]. <http://libgdx.badlogicgames.com/documentation.html>. [Consulta: 08 de diciembre de 2014].

6. BARRIOS GONZALES, Carlos Andrés, "*Análisis de diferentes lenguajes y herramientas para el desarrollo de videojuegos*". [en línea]. <http://biblioteca.usac.edu.gt/tesis/08/08_0523_CS.pdf>. [Consulta: 16 de septiembre de 2014].

7. *BitBucket: Mi mejor alternativa frente a GitHub*. [en línea]. <http://blog.desdelinux.net/bitbucket-mi-mejor-alternativa-frente-a-github/>. [Consulta: 22 de enero de 2015].

8. CLINTON, Keith. *Agile Game Development with Scrum*. EEUU: Addison-Wesley Signature Series (Cohn), 2010. 312 p.

9. COCOS2DX. *Cocos2D JS*. [en línea]. <http://www.cocos2d-x.org/wiki/Getting_Started_Cocos2d-js>. [Consulta: 20 de diciembre de 2014].

10. CODEANDWEB. *Texturepacker*. [en línea]. <https://www.codeandweb.com/texturepacker>. [Consulta: 26 de diciembre de 2014].

11. COVEYOU, John. *Educational Game Design Topic #2*. <http://gotgeniusgames.com/educational-game-design-topic-2-streamline-summarize-overarching-concept/ >. [Consulta: 26 de octubre de 2014].

12. DELGADO EXPÓSITO, Erly. *Metodologías de desarrollo de software. ¿Cuál es el camino?.* [en línea]. < http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml>. [Consulta: 17 de noviembre de 2014].

13. DRIESSEN, Vincent. *A successful Git branching model*. [en línea]. <http://nvie.com/posts/a-successful-git-branching-model/#supporting-branches >. [Consulta: 06 de febrero de 2015].

14. *EHOWSPANOL. ¿Qué habilidades se necesitan para ser un diseñador de juegos?.* [en línea]. <http://www.ehowenespanol.com/habilidades-necesitan-disenador-juegos-lista_365995/>. [Consulta 09 de octubre de 2014].

15. GALVIS PAQUEVA, Alvaro Hernán. *Ingeniería de Software Educativo*. [en línea]. <http://www.slideshare.net/algalvis50/ingeniera-de-software-educativo-1992-parte-2-metodologia>. [Consulta: 15 de octubre de 2014].

16. GAMUA. *Starling*. [en línea]. <http://gamua.com/starling/help/>. [Consulta: 15 de diciembre de 2015].

17. GARCÍA FERNÁNDEZ, Fernando. *Videojuegos: un análisis desde el punto de vista educativo.* [en línea]. <http://www.irabia.org/departamentos/nntt/proyectos/futura/futura06/Analisis_educativo.pdf>. [Consulta: 07 de septiembre de 2014].

18. *Gestión de la configuración.* [en línea]. <http://isg2.pbworks.com/w/page/7624276/Gesti%C3%B3n%20de%20la%20Configuraci%C3%B3n>. [Consulta: 10 de septiembre de 2014].

19. GIT-SCM. *git –fast-version-control.* [en línea]. <http://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones>. [Consulta: 23 de enero de 2015].

20. *Guía Técnica para la Evaluación de Software en la Administración Publica*. [en línea]. <http://www.ongei.gob.pe/bancos/banco_normas/archivos/guia-evaluacion-sw.pdf>. [Consulta: 29 de noviembre de 2014].

21. HIPERTEXTUAL. *Github y Bitbucket: servicios de Git en la nube*. [en línea]. <http://hipertextual.com/archivo/2014/05/github-y-bitbucket/>. [Consulta: 22 de enero de 2015].

22. KNIBERG, Henrik. *Scrum y XP desde las trincheras.* [en línea]. <http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf>. [Consulta: 26 de septiembre de 2014].

23. LARGO GARCIA, Carlos Alberto; MARIN MAZO, Erledy. *Guía Técnica para la Evaluación de Software*. [en línea]. <http://es.slideshare.net/abetancur/guia-tecnica-para-evaluacin-de-software>. [Consulta: 28 de noviembre de 2014].

24. MAPEDITOR. *Tiled*. [en línea]. <https://github.com/bjorn/tiled/wiki>. [Consulta: 27 de diciembre de 2014].

25. *Matriz de Evaluación de Software*. [en línea]. <http://fdv.wikispaces.com/Matriz+de+Evaluacion+de+Software>. [Consulta: 03 de enero de 2015].

26. MESA, Gustavo; ACERENZA, Nicolás; COPPES, Ariel; VIERA, Alejandro. *Una Metodología Ágil para Desarrollo de videojuegos*. [en línea]. < http://www.ajedrezyleyendas.com/docs/informe-final.pdf >. [Consulta: 16 de octubre de 2014].

27. MICROEVOLUTION*. ¿Qué es una matriz de evaluación de software?* [en línea]. <http://microevolution.wikispaces.com/2.-+%C2%BFQu%C3%A9+es+una+Matriz+de+Evaluaci%C3%B3n+de+Software%3F>. [Consulta: 03 de enero de 2015].

28. PINEDA, Dazia. *Control de versiones de software.* [en línea]. <http://www.cristalab.com/blog/control-de-versiones-de-software-c82353l/>. [Consulta: 04 de febrero de 2015].

29. QUINTERO, Hugo; PORTILLO, Lisbeth; LUQUE, Rafael; GONZÁLEZ, Maricela. *Desarrollo de software educativo: una propuesta metodológica*. [en línea]. <http://www.redalyc.org/articulo.oa?id=99318837004>. [Consulta: 15 de octubre de 2014].

30. SOURCETREEAPP. *SourceTree*. [en línea]. <http://www.sourcetreeapp.com/ >. [Consulta: 28 de enero de 2015].

31. SYNTEVO. *Smartgit*. [en línea]. < http://www.syntevo.com/smartgit/>. [Consulta: 27 de enero de 2015].

32. TABOA, Nora. *¿Qué son los objetivos SMART?.* [en línea]. <http://www.norataboada.com/blog/qu-son-los-objetivos-smart>. [Consulta: 10 de septiembre de 2014].

33. UNICEF. *La educación en Guatemala*. [en línea]. http://www.unicef.org/guatemala/spanish/resources_2562.htm>. [Consulta: 16 de septiembre de 2014].

34. WIKIPEDIA. *Motor de videojuego*. [en línea]. <http://es.wikipedia.org/wiki/Motor_de_videojuego>. [Consulta: 10 de septiembre de 2014].

35. WIKIPEDIA. *Framework*. [en línea]. <http://es.wikipedia.org/wiki/Framework>. [Consulta: 10 de septiembre de 2014].

36. WIKIPEDIA. *ISO/IEC 9126.* [en línea]. <http://es.wikipedia.org/wiki/ISO/IEC_9126>. [Consulta: 02 de enero de 2015].

37. WIKIPEDIA. *Bitbucket*. [en línea]. <http://es.wikipedia.org/wiki/Bitbucket>. [Consulta: 22 de enero de 2015].

38. WORCESTER POLYTECHNIC INSTITUTE. *The Game Development Process Game Design*. [en línea]. <http://web.cs.wpi.edu/~id111x/c05/slides/GameDesign.pdf>. [Consulta: 17 de noviembre de 2014].

39. _____. *The Game Development Process Documentation.* [en línea]. <http://web.cs.wpi.edu/~id111x/c05/slides/Documentation.pdf>. [Consulta: 17 de noviembre de 2014].

40. _____. *The Game Development Process Postmortems*. [en línea]. <http://web.cs.wpi.edu/~id111x/c05/slides/PostMortem.pdf>. [Consulta: 17 de noviembre de 2014].

APÉNDICES

Apéndice A

1. PROPUESTA DE METODOLOGÍA DE DESARROLLO DE VIDEOJUEGOS EDUCATIVOS

1.1. Justificación

Dado que la Asociación Civil Edulibre tiene como proyecto la elaboración de una plataforma que albergue software de juego educativo para niños y niñas de educación primaria, carece de una metodología de desarrollo de software para elaborar los videojuegos educativos que integrarán esta plataforma, surge la necesidad de elaborar una. La propuesta de la metodología está basada en las metodologías de desarrollo ágil XP, Scrum y Sum principalmente de las cuales se toma su estructura, se modifica y adapta a lo que se está buscando.

1.2. Objetivos de la metodología

La propuesta de una metodología de desarrollo de videojuegos educativos, tiene como objetivo principal el facilitar el desarrollo de videojuegos educativos en tiempo y costos razonables, dando lugar a ser mejorado cada vez que se utilice, en función de la experiencia adquirida en su utilización. La metodología

es fácilmente manejable para equipos multidisciplinarios pequeños de no más de siete integrantes y proyectos con máximo de duración de seis meses.

1.3. Roles

La metodología define cinco roles: Equipo de desarrollo, Pedagogo, Cliente, Líder de proyecto y Verificador de videojuego. Es propicio mencionar que una persona puede desempeñar varios roles a la vez, si así se requiere.

1.3.1. Equipo de desarrollo

Dado el tipo de proyecto, el equipo de desarrollo está conformado por varios especialistas en diferentes áreas, que juntos darán vida al videojuego. Estos se definen a continuación.

1.3.1.1. Diseñador del videojuego

El diseñador del videojuego se encargara de la parte de la mecánica, el guion (si lo hubiera, depende del tipo de juego), los personajes, los niveles, y demás elementos que darán vida al videojuego. Es importante mencionar que para que el juego sea divertido la dificultad y el aprendizaje del videojuego deben estar equilibrados, ya que de esto dependerá que tan divertido sea el videojuego, que es realmente lo que se busca, que sea divertido.

Ahora bien también debe ser capaz de definir la plataforma, el tipo de videojuego y tener en mente el público al cual va dirigido el videojuego. Ya que la idea es hacer videojuegos educativos, el pedagogo también estará presente cuando el diseñador haga la propuesta del videojuego, para ayudarle en la

parte educativa sugiriendo cambios que crea necesarios a modo de elevar el aprendizaje del educando y preservar lo entretenido del videojuego.

La persona que desempeñe este rol debe tener varias habilidades como:

- Ser creativo y original
- Saber contar historias y escribirlas
- Conocimiento básico de programación
- Comunicación interpersonal
- Trabajo en equipo
- Capacidad para adaptarse al cambio
- Conocimiento básico de diseño y animación en 2D

1.3.1.2. Diseñador Gráfico

El diseñador gráfico se encargara de crear todo el material artístico del videojuego, incluyendo el arte de concepto, modelos en 2D y animaciones. El trabajo que este rol representa es muy importante ya que este hará la parte visible del videojuego. Se requiere de conocimientos sobre herramientas gráficas, mucha creatividad, y buena técnica de dibujo. El pedagogo ayudará el diseñador gráfico con la parte gráfica del videojuego para que esta sea adecuada a la edad y grado de los niños.

La persona que desempeñe este rol deberá contar con habilidades como:

- ✓ Conocimiento de paquetes de diseño y creación de imágenes
- ✓ Talento, creatividad y originalidad
- ✓ Conocimientos básicos de informática

- ✓ Trabajo en equipo
- ✓ Capacidad de comunicación

1.3.1.3. Programador

El programador se encarga de darle vida al videojuego diseñado por el diseñador de videojuegos. Su tarea es diseñar, implementar y verificar el software que compondrá al videojuego.

La persona que desempeñe este rol deberá contar con habilidades como:

- Habilidad para programar
- Disfrutar crear videojuegos
- Conocimientos de lenguajes de programación
- Trabajo en equipo
- Trabajar bajo presión
- Capacidad para la resolución de problemas

1.3.1.4. Diseñador de efectos de sonido.

Es el que está encargado de crear todos los efectos de sonido del videojuego que sean requeridos, así como también la música. Los efectos de sonido deben ser creados acorde, y que correspondan a lo que el jugador está viendo, y la acción que esté realizando. Es sonido es una parte muy importante del videojuego ya que este le da vida a la escena y enriquece la experiencia del jugador.

La persona que desempeñe este rol deberá contar con habilidades como:

- Composición, producción y grabación de música
- Manejo de herramientas
- Componer e interpretar música
- Sentido de la diversión
- Trabajo en equipo
- Conocimiento musical

1.3.2. Pedagogo

El pedagogo será la persona encargada de trabajar en equipo junto con el diseñador del videojuego, el diseñador gráfico y el programador para lograr que el videojuego cumpla su función de educar. Deberá estar desde el principio para poder ayudar especialmente al diseñador del videojuego para que juntos logren diseñar un videojuego que sea divertido, y que el contenido de este acorde a lo que se busca enseñar en los educandos, además de ponerse de acuerdo en cómo se va a medir el progreso del alumno.

Con el diseñador gráfico vera la parte estética del videojuego para que las imágenes que se vayan a mostrar a los niños en el videojuego sean las adecuadas para su edad. Estará junto con el programador en las pruebas, para ver el videojuego y verificar que cumpla con lo establecido.

La persona que desempeñe este rol deberá contar con habilidades como:

- Trabajo en equipo
- Buena comunicación
- Tener un título universitario de pedagogía
- Conocimiento en temas sobre aprendizaje y enseñanza

1.3.3. Cliente

El cliente es el encargado de especificar que requiere del videojuego y mantener la perspectiva del mismo.

Las responsabilidades del cliente durante el proyecto son:

- Define los requerimientos del videojuego.
- Decide la fecha y contenido del próximo realese.
- Define y valida el concepto del videojuego, aprueba los planes de proyecto y determina los hitos.
- Prioriza las características y tareas que dan más valor al videojuego en cada iteración como se necesite.
- Acepta o rechaza los resultados del trabajo o iteración.

1.3.4. Líder de proyecto

El líder del proyecto será el encargado de guiar el desarrollo del videojuego, promover buenas prácticas, interactuar con el cliente y resolver cualquier impedimento que pueda atrasar el proyecto.

Las responsabilidades del líder el proyecto son:

- Remover impedimentos
- Responsable de la correcta aplicación de la metodología
- Promover el trabajo en equipo entre los diferentes roles y sus funciones
- Proteger al equipo de interferencias externas

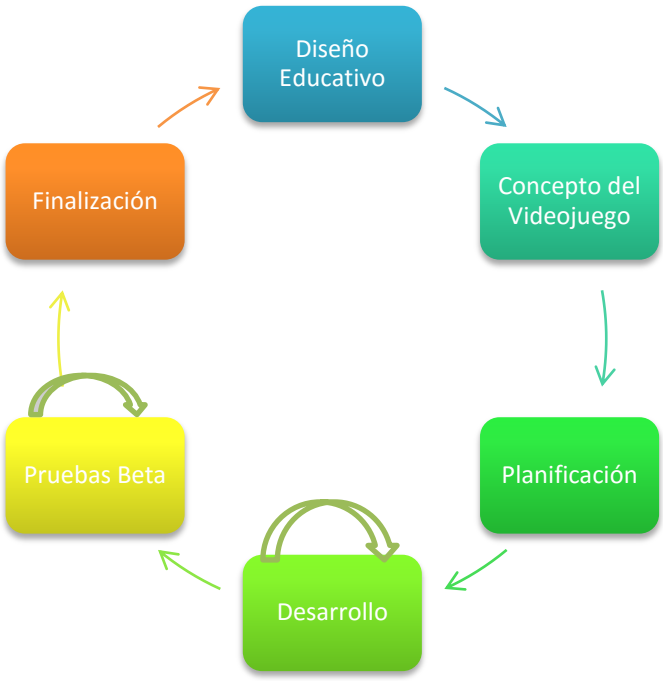
1.3.5. Verificador de videojuego

La función principal de este rol es realizar la verificación funcional del videojuego. Entra en acción cuando ya se tiene una versión beta del videojuego, ya casi completa. Cuando realice la evaluación deberá elaborar un documento donde describa los errores encontrados.

1.4. Ciclo de vida de la metodología propuesta

La propuesta de la metodología se divide en varias fases de las cuales algunas son iterativas e incrementales. Las seis fases secuenciales son: Diseño Educativo, Concepto del Videojuego, Planificación, Desarrollo, Pruebas Beta y Finalización. La figura siguiente muestra adecuadamente las fases. La mayoría de fases tiene una sola iteración excepto la de Desarrollo y Pruebas Beta que pueden tener más.

Figura 1. **Fases de la metodología propuesta.**

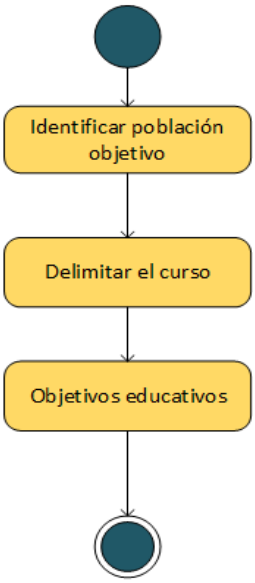


Fuente: elaboración propia.

1.4.1. Diseño educativo

La fase de diseño educativo consiste en organizar toda la estructura del contenido educativo a impartir. Se deben resolver interrogantes que se refieren al alcance, contenido y tratamiento que debe ser capaz de apoyar el videojuego. La fase diseño educativo está compuesta de varias actividades que se describen a continuación.

Figura 2. **Flujo de las actividades a realizar**



Fuente: elaboración propia.

1.4.1.1. Identificación de la población objetivo

Esta actividad consiste en conocer a los usuarios que harán uso del videojuego. No es lo mismo hacer un videojuego para un niño de cinco años que para uno de diez. Es importante esta actividad ya que buena parte de la motivación y el esfuerzo así como el sistema de comunicación dependen de quienes serán los futuros usuarios del material. Para establecer adecuadamente las características conviene resolver las siguientes preguntas.

- ¿A qué grupo de edad pertenecen los estudiantes?
- ¿Qué nivel de escolaridad tienen?
- ¿Qué experiencias previas deben tener los estudiantes, relevantes para el estudio del tema?

- ¿Deben tener alguna aptitud o habilidad especial que deba tomarse en cuenta?

1.4.1.2. Delimitación del curso

Esta actividad consiste en la delimitación del contenido de área de conocimiento a impartir en el videojuego. Para esta parte se debe tomar en cuenta el contexto en que la ejemplificación y ejercitación se van a dar.

Para tener más clara el área de contenido deben de resolverse, por lo menos las siguientes preguntas:

- ¿Qué grado de estudios se va atender? Por ejemplo primer grado.
- ¿Qué área de conocimiento se desea enseñar? Por ejemplo matemáticas.
- ¿Qué contenido del área de conocimiento se planea dar? Por ejemplo aritmética.
- ¿Deben de tener algún conocimiento previo, que les ayude a entender el de mejor manera el tema? Por ejemplo los números naturales.

1.4.1.3. Objetivos educativos

Estos objetivos están enfocados a lo que se quiere lograr en el alumno, es decir la conducta que se espera de él, y donde se especifican los contenidos a considerar durante el proceso de enseñanza aprendizaje, propuesto por el videojuego. En el caso de Guatemala pueden tomarse como base los del Curriculum Nacional Base, propuestos por el Ministerio de Educación.

Tabla I. **Roles, entradas y salidas de diseño educativo**

Roles involucrados	Entradas de la fase	Salidas de la fase
Cliente **Pedagogo** **Líder de proyecto**	Ninguna	Documento de diseño educativo

Fuente: elaboración propia.

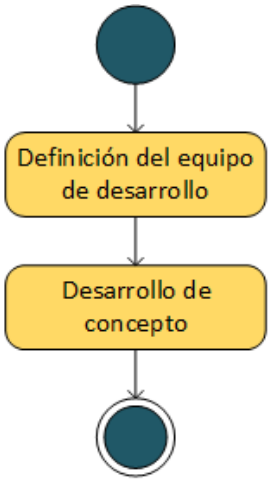
1.4.2. Concepto del videojuego

Para esta fase lo principal es tomar las decisiones técnicas y elementos de juego sobre el producto a desarrollar. Las decisiones técnicas implican la elección de las herramientas, tecnologías a utilizar y las plataformas para las que se pretende desarrollar. En cuanto a los elementos de juego se busca determinar las principales características que tendrá este, como la historia, los personajes, la ambientación y la mecánica del juego principalmente.

Para la realización del concepto todo el equipo debe estar presente eso incluye al cliente, al equipo de desarrollo y al líder de proyecto. La construcción del concepto se realizara a partir de una lluvia de ideas y propuestas de cada una de las partes sobre los aspectos a definir.

Es importante mencionar que el desarrollo del concepto se debe tomar en cuenta los aspectos definidos en la fase de diseño educativo, para que la propuesta pueda cumplir con los objetivos de enseñanza aprendizaje establecidos. Para completar esta fase deben de realizarse dos actividades que se ejecutan secuencialmente estas son: la definición del equipo de desarrollo y el desarrollo del concepto. Las propuestas son modificables, se pueden ir afinando en el camino, no tiene que estar del todo definida.

Figura 3. **Actividades – Concepto del videojuego.**



Fuente: elaboración propia.

1.4.2.1. Definición del equipo de desarrollo

Para esta parte se busca armar el equipo de desarrollo que creará el videojuego, y estará presente en el resto del proyecto. Para esto se debe tener en cuenta el concepto del videojuego, donde se identifican las necesidades técnicas y artísticas requeridas por proyecto. En función de estas necesidades se procede a seleccionar a las personas serán parte del equipo de desarrollo.

Es posible que existan necesidades en el proyecto que las personas que integran el equipo no pueden suplir, por lo que deben ser cubiertas de forma externa. Entonces se puede contratar a alguna persona o entidad externa que pueda cubrir la necesidad, tomando en cuenta el precio (si lo hay) y que esto no afecte de forma drástica el presupuesto. Para llevar a cabo de forma adecuada esta tarea se recomienda lo siguiente:

- Identificar subroles faltantes: Tomando el concepto del videojuego y su diseño, identificar los conocimientos y cuantos especialistas son necesarios para cumplir con los requerimientos del proyecto.

- Definir el equipo de desarrollo: Buscar y seleccionar a los miembros que conformaran el equipo de desarrollo en base a sus habilidades y los roles a cubrir.

- Ayuda externa: Esta solo se usa en caso de no contar con algún miembro que pueda cubrir cierta necesidad del proyecto. Se debe buscar ayuda externa a la institución, para ver la posibilidad de colaboración para el proyecto.

Tabla II. **Roles, entradas y salidas de definición del equipo**

Roles involucrados	**Entradas de la actividad**	**Salidas de la actividad**
Cliente **Líder de proyecto**	Ninguna	Definición del equipo de desarrollo

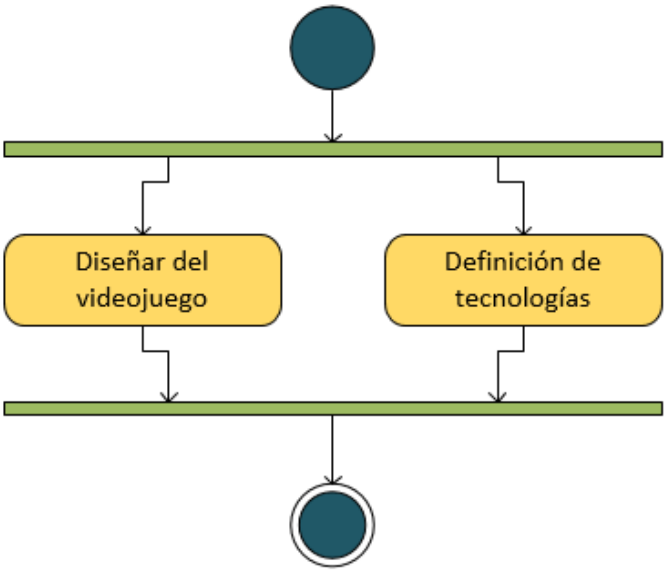
Fuente: elaboración propia.

1.4.2.2. Desarrollo del concepto

Para esta actividad se realizan dos tareas, una para definir los aspectos técnicos, y en la otra los elementos que compondrán el videojuego educativo. El concepto del videojuego se lleva a cabo por medio de una lluvia de ideas en función de lo planteado en el documento de diseño educativo. Las propuestas no necesariamente tienen que quedar definidas de una vez, estas pueden cambiar o mejorarse, y analizar su factibilidad. Las dos tareas propuestas se

realizan en paralelo, se puede empezar por cualquiera, ya que las decisiones en una pueden afectar a la otra.

Figura 4. **Tareas – Desarrollo de concepto**



Fuente: elaboración propia.

1.4.2.2.1. Diseñar el videojuego

Para esta tarea el fin principal es definir varios aspectos del videojuego como: perspectiva, género, mecánica de juego, etc. Para esta parte se pueden realizar pruebas sobre el concepto del videojuego, para verificar si este es divertido, educativo y entretenido.

Antes de empezar en primer lugar se debe hacer una reunión en la que todos puedan discutir y proponer ideas para definir tanto la visión como las características principales del videojuego**,** tomando en cuenta la fase de diseño educativo**,** ya que nos interesa proporcionar conocimiento a través del

videojuego. Como ayuda es importante la realización de bocetos o *storyboards* para tener una mejor idea de cómo será el videojuego.

A continuación se define de mejor manera los aspectos a tomar en cuenta.

- Perspectiva: básicamente consiste en describir la experiencia que se quiere crear con el videojuego, es decir lo divertido, entretenido, excitante, y que lo diferencia de otros. Aquí definimos los objetivos del videojuego, la forma de lograrlos, los retos que se llevaran a cabo, etc.

- Transmisión de conocimiento: es importante que acorde a la visión se tome en cuenta también los objetivos enseñanza aprendizaje que se quieren lograr.

- Género: definir el género del videojuego, no necesariamente tiene que ser uno específico, también puede ser la mezcla de varios géneros. En esta parte es bueno buscar juegos similares y hacer comparaciones.

- Mecánica del juego: para esta parte definiremos el modo en que el jugador interactuara con el ambiente del videojuego, es decir que acciones este podrá realizar durante el juego.

- Características: las características son los requerimientos que se planea implementar en el videojuego. Es importante detallar cada una de estas y porque son importantes, además de tener una idea de cómo se implementaran.

- Historia y ambiente: implica describir el mundo del videojuego en detalle y explicar a cada uno de sus personajes, su función, motivación, sus

objetivos y cómo piensan alcanzarlos o si lo harán. La historia no tiene que ser muy larga si el videojuego será algo repetitivo como Pacman por ejemplo. Para esta parte también debemos tomar en cuenta la forma en que se impartirá el conocimiento que se requiere obtenga el alumno en el transcurso de la historia.

- Arte: el arte definirá el estilo del videojuego, es decir la forma de sus gráficas, también si será en 2D o 3D. También el nivel de detalle que tendrán estas, se sugiere el uso de bocetos para esta parte.

- Música: la música es un elemento importante en el videojuego, ya que esta le da más realce a la historia, mientras se avanza en el videojuego. Es importante que esta case correctamente en cada parte del videojuego. No tiene que estar completa pero por lo menos tener una idea de qué manera se oirá.

- Pruebas sobre el concepto: es importante realizarlas para mejorar aún más la propuesta de concepto del videojuego, y tratar de minimizar los riesgos de que el videojuego no sea divertido y educativo. La forma de realizar estas pruebas puede ser a través de simulaciones en papel, investigar y jugar videojuegos identicos, la realización de un prototipo, o algún otro método que permita tener una mejor perspectiva de la idea.

Tabla III. **Roles, entradas y salidas de "Diseñar videojuego"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Pedagogo** **Líder de proyecto** **Equipo de desarrollo**	Documento de diseño educativo	Documento de concepto -- Diseño del videojuego

Fuente: elaboración propia.

1.4.2.2.2. Definición de tecnologías

Para esta tarea elegimos la plataforma o plataformas donde correrá el videojuego. Además las tecnologías y herramientas a utilizar en su desarrollo. Para probar las herramientas se puede hacer prototipos que prueben algunos aspectos seleccionados del videojuego, y seleccionar la adecuada.

Parea realizar correctamente esta tarea se debe definir lo siguiente.

- La plataforma o plataformas: se debe elegir la plataforma o plataformas donde podrá ser ejecutado el videojuego.

- Tecnologías y herramientas: se debe seleccionar las tecnologías y las herramientas que se utilizaran para el desarrollo, tomando en cuenta que soporten adecuadamente la plataforma o plataformas seleccionadas. En el documento de Evaluación de tecnologías se realiza el análisis de varias herramientas y se recomienda una.

- Prototipos técnicos: los prototipos técnicos suelen hacerse cuando se tienen varias opciones de herramientas que parecen idóneas para el

desarrollo. Se pueden realizar pequeños prototipos, ya que esto permitirá una elección mejor y más informada.

Tabla IV. **Roles, entradas y salidas de "Aspectos técnicos"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Documento de Concepto *– Diseño del videojuego*	Documento de Concepto *– Diseño del videojuego* *– Aspectos técnicos*

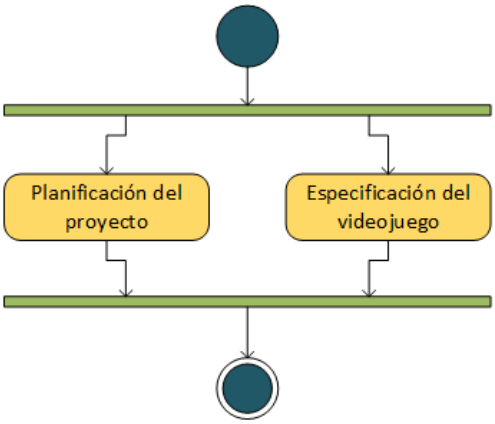
Fuente: elaboración propia.

1.4.3. Planificación

Para esta fase se realiza el cronograma de actividades que tendrán el resto de las fases y además se especificaran las características que tendrá el videojuego. Esta fase implica la realización de dos actividades cuyos resultados compondrán el plan del proyecto en sí. Estas actividades son mutuamente dependientes ya que el cronograma de actividades debe ser coherente con el tiempo estimado y la realización de los requerimientos del videojuego.

Esta debería ser una fase corta. Se debe tener en cuenta que para que sea oficial, el cliente debe estar de acuerdo. El cronograma de actividades que se obtiene en esta fase, es el estimado y como tal también está sujeto a cambios conforme se vaya realizando el proyecto, esto para reflejar el estado actual del desarrollo del videojuego, y tomar las medidas necesarias aprovechar de mejor manera el tiempo.

Figura 5. **Fase de planificación**

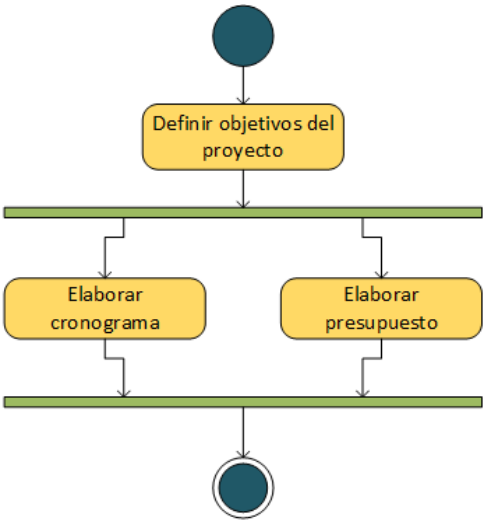


Fuente: elaboración propia.

1.4.3.1. Planificación del proyecto

Como cualquier proyecto, este debe tener una planificación que permita llevar a cabo las tareas en un orden adecuado. Para esta actividad se deben realizar tres tareas para definir diversos elementos del plan de proyecto. La primera tarea “Definir objetivos del proyecto” es la primera que debe realizarse, luego de forma paralela se pueden realizar la de “Elaborar el cronograma” y “Elaborar el presupuesto”.

Figura 6. **Tareas – Planificación del proyecto**



Fuente: elaboración propia.

1.4.3.1.1. Definir objetivos del proyecto

Para cualquier proyecto es importante definir objetivos claros que se requieren alcanzar para dar por terminado el proyecto. Para este apartado se recomienda que los objetivos a definir sean SMART, para poder medir su éxito.

Tabla V. **Roles, entradas y salidas de "Objetivos del proyecto"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto**	Ninguna	Plan de proyecto – Objetivos del proyecto

Fuente: elaboración propia.

1.4.3.1.2. Elaborar el cronograma

Crear un cronograma es parte de cualquier proyecto que se quiera implementar. Para la creación del cronograma que albergara las fases restantes del proyecto se toma como base el concepto del videojuego. En el cronograma se establecen las fechas estimadas para dar comienzo a la fase de desarrollo, pruebas beta y finalización. Se debe incluir también la cantidad estimada de iteraciones a realizar durante la fase de desarrollo junto con sus duraciones y criterios de finalización, para poder empezar la fase de pruebas beta.

Para que el cronograma sea elaborado de forma adecuada se recomienda hacer lo siguiente:

- Definir la cantidad de iteraciones y la duración que tendrán estas durante la fase de desarrollo.
- Definir la cantidad de iteraciones y los criterios que den por terminada la fase de pruebas.
- Definir una fecha tentativa de finalización del proyecto.
- Definir los hitos que servirán de guía para poder pasar de fase, en función de los objetivos establecidos.

Tabla VI. **Roles, entradas y salidas de "Elaborar cronograma"**

Roles involucrados	**Entradas de la tarea**	**Salidas de la tarea**
Cliente **Líder de proyecto**	Documento de Concepto	Plan de proyecto – Cronograma

Fuente: elaboración propia.

1.4.3.1.3. Elaborar el presupuesto

Hacer un presupuesto es muy útil para el proyecto, ya que se desea estimar los costos en los que se incurrirá para su elaboración. Para esta tarea se debe determinar el costo total que tendrá el proyecto y la forma en que se obtendrán los recursos económicos necesarios para poder realizarlo, en función del concepto del videojuego y el resto de los aspectos del plan de proyecto.

Algunos aspectos a tomar en cuenta para el presupuesto son:

- Salarios (Si aplica, lo ideal sería conseguir voluntarios)
- Licencias de software y herramientas (Si aplica)
- Hardware (Si Edulibre, proporcionara equipo de cómputo)
- Publicidad (Si se hará, o será proyecto interno)
- Costos fijos (luz, internet, teléfono, etc.)

Tabla VII. **Roles, entradas y salidas de "Elaborar presupuesto"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Líder de proyecto **Equipo de desarrollo**	Documento de Concepto	Plan de proyecto – Presupuesto

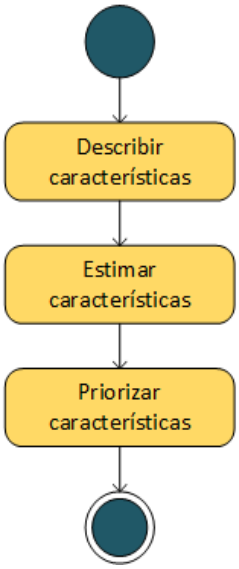
Fuente: elaboración propia.

1.4.3.2. Especificación del videojuego

El propósito de esta actividad es definir las características tanto funcionales como no funcionales del videojuego. Se entiende por un característica funcional, una funcionalidad desde el punto de vista del usuario final, como una

historia de usuario. Las características funcionales son una excelente herramienta que tiene el cliente para comunicar al equipo los requisitos del videojuego y poder medir el progreso del mismo. Las características no funcionales son cualidades inherentes que el videojuego debe presentar, como por ejemplo calidad, documentación, rendimiento, etc. Las tareas a realizar se harán de forma secuencial, estas son *Describir características*, *Estimar características* y *Priorizar características*.

Figura 7. **Tareas – Especificación del videojuego**



Fuente: elaboración propia.

1.4.3.2.1. Describir características

Para realizar esta tarea se deben determinar y describir cuáles serán las características funcionales según el concepto del videojuego. Cada requerimiento no necesariamente tiene que ser bien detallado, pero si lo suficientemente descriptivo para poder estimar un tiempo razonable para su

realización. Se pueden utilizar historias de usuario para definir mejor estas características.

Tabla VIII. **Roles, entradas y salidas de "Describir características"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Documento de Concepto	Plan de proyecto – Características del videojuego

Fuente: elaboración propia.

1.4.3.2.2. Estimar características

Se debe estimar el tiempo que tomara realizar todas las características descritas. Es importante que equipo de desarrollo las vea, y calcule cuanto tiempo les tomaría implementarlas, para tener una idea de lo que durara el proyecto. Pueden ser todas o bien una parte de ellas las estimadas, especialmente las consideradas urgentes.

Tabla IX. **Roles, entradas y salidas de "Estimar características"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Documento de Concepto Plan de proyecto - Características del videojuego	Plan de proyecto – Estimación de características

Fuente: elaboración propia.

1.4.3.2.3. Priorizar características

Para esta parte se clasifican en orden de las características definidas para el videojuego. Priorizar permitirá determinar de mejor manera en que orden deben ser desarrolladas las características del videojuego, a modo ver resultados rápido. Para esta tarea, es el cliente quien decide la importancia de las características de su punto de vista.

Tabla X. **Roles, entradas y salidas de "Priorizar características"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Documento de Concepto Plan de proyecto - Características del videojuego	Plan de proyecto – Características prioritarias.

Fuente: elaboración propia.

1.4.4. Desarrollo

Esta es la fase más importante, y cuyo objetivo principal es la realización del videojuego propuesto. Esta fase se trabaja de forma iterativa e incremental para que cuando cada iteración termine, se obtenga un demo ejecutable del videojuego.

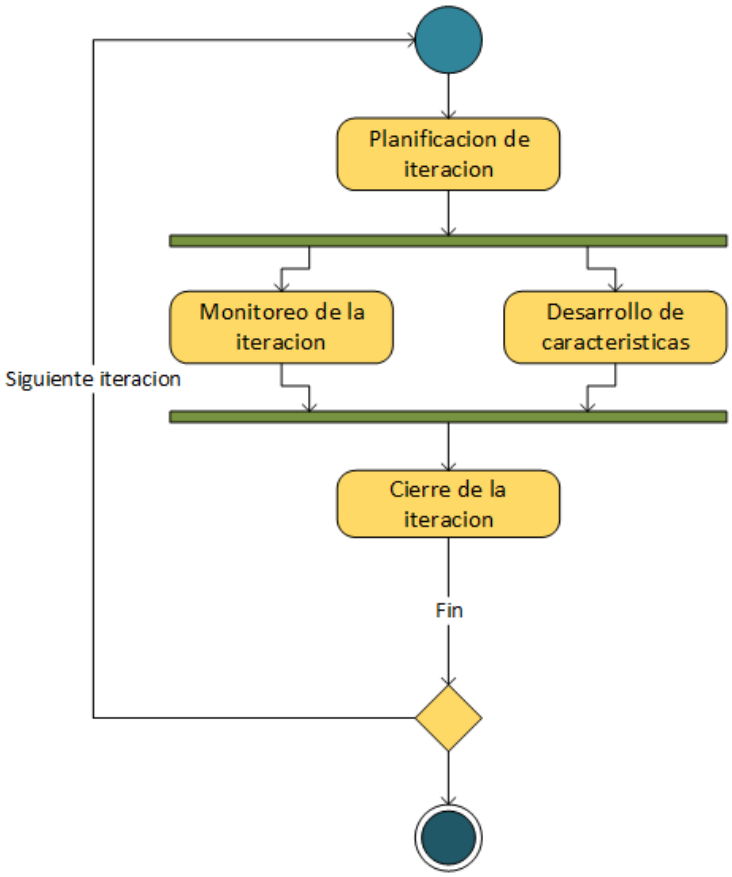
Al hacer iterativo e incremental la fase de desarrollo se puede obtener una versión temprana del videojuego lo cual permite ver su evolución; y así poder evaluar el avance del mismo, lo cual también permitirá hacer las modificaciones necesarias al cronograma de actividades con relación al tiempo y tomar las

decisiones que permitan cumplir con los plazos especificados. Esto aumentara la experiencia del equipo, mejorar su forma de trabajo y ser más productivos.

Se recomienda que las iteraciones no pasen de un mes, de hecho se recomienda tres semanas ya que es un tiempo lo suficientemente largo para no estresar al equipo de desarrollo y lo suficientemente corto para obtener un ejecutable demo del videojuego.

Las actividades a desarrollar para esta fase son: Planificación de la iteración, Seguimiento de la interacción, Desarrollo del videojuego y Cierre de la iteración.

Figura 8. **Actividades – Fase de desarrollo**

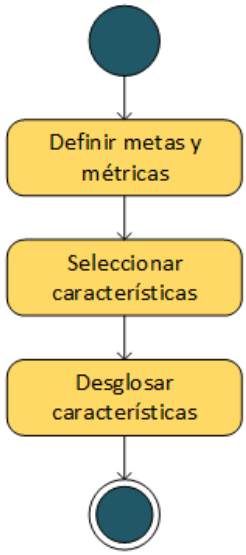


Fuente: elaboración propia.

1.4.4.1. Planificación de la iteración

La planificación se realiza una única vez por iteración y esta se hace para seleccionar las características a implementar. Para ello se realizan tres tareas secuenciales que son: Definir metas y métricas, la selección de características y el desglose de las mismas.

Figura 9. **Tareas – Planificación de la iteración**



Fuente: elaboración propia.

1.4.4.1.1. Definir metas y métricas

Es bueno definir una o más metas (o bien objetivos) en una iteración para tener una idea de hacia dónde vamos y lo que se quiere lograr. Es importante también definir de qué manera se va a medir el progreso de la iteración y si van logrando cada una de las metas propuestas. Esto permitirá que se puedan tomar mejores decisiones en el transcurso de la iteración y poder medir el éxito de la misma.

Tabla XI. **Roles, entradas y salidas de "Definir metas y métricas"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Plan de proyecto - Características del videojuego	Plan de iteración – Características del videojuego. – Métricas – Metas u objetivos de la iteración.

Fuente: elaboración propia.

1.4.4.1.2. Seleccionar características

Para esta tarea se deben seleccionar las características que harán parte de la iteración. Se debe tomar en cuenta en base a su prioridad, el tiempo que tomara la realización de la iteración será la suma de los tiempos de cada requerimiento.

Tabla XII. **Roles, entradas y salidas de "Seleccionar características"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Plan de proyecto - Características del videojuego	Plan de iteración – Características del videojuego seleccionadas.

Fuente: elaboración propia.

1.4.4.1.3. Desglosar características

Dada una característica, su realización puede conllevar la realización de varias tareas. El descomponer las características del videojuego en tareas de menor dificultad hará que sea más fácil su estimación, y asignarlas a un miembro del equipo, verificarlas y evaluarlas. Las tareas es conveniente dividirlas en grupos en función de las disciplinas que están involucradas. Por ejemplo tareas de lógica del juego, programación, visuales, audio, etc.

El equipo de desarrollo es quien decide las tareas necesarias para poder cumplir con la realización de las características, por lo cual se convierten en responsables del cumplimiento de estas.

Tabla XIII. **Roles, entradas y salidas de "Desglosar características"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Plan de iteración - Características del videojuego seleccionadas.	Plan de iteración – Tareas de la iteración.

Fuente: elaboración propia.

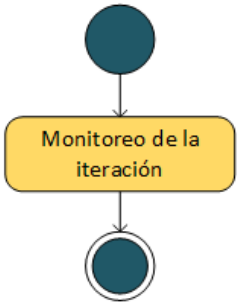
1.4.4.2. Monitoreo de la iteración

Esta actividad en sí misma es una tarea a realizar. Su función es evaluar el avance de la iteración para detectar problemas e impedimentos que no permiten que el proyecto avance adecuadamente y tomar las decisiones necesarias en caso haya problemas.

En base a las métricas establecidas se mide el progreso del proyecto acorde a las metas del mismo para verificar el estado de la iteración y medir la rapidez con que el equipo avanza en su cumplimiento. Los problemas si ocurrieran, se

deben tomar nota de ellos, tomando en cuenta la causa, las posibles soluciones y su impacto en la iteración. El líder del proyecto es el encargado de realizar el monitoreo y mantener informado al cliente y al equipo sobre el avance.

Figura 10. **Tareas – Monitoreo de la iteración**



Fuente: elaboración propia.

Tabla XIV. **Roles, entradas y salidas de "Monitoreo de la iteración"**

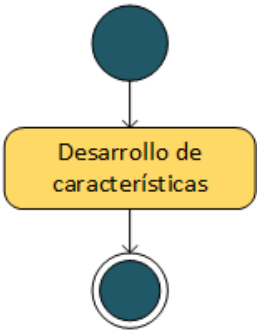
Roles involucrados	Entradas de la tarea	Salidas de la tarea
Líder de proyecto **Equipo de desarrollo**	Plan de iteración - Características del videojuego seleccionadas. - Tareas de la iteración. - Métricas - Metas de la iteración.	Resultados de medición.

Fuente: elaboración propia.

1.4.4.3. Desarrollo del videojuego

El desarrollo del videojuego comprende una sola tarea, la cual consiste simplemente en realizar las tareas que fueron programadas para la iteración.

Figura 11. **Tareas – Desarrollo del videojuego**



Fuente: elaboración propia.

1.4.4.3.1. Desarrollar características

Para esta tarea se realizan y validan las características programadas en la iteración, ejecutando cada una de las tareas que las componen. Cabe recalcar que la selección de las características a desarrollar queda a discreción del equipo, siendo responsables de llevarlas a cabo, tomando en cuenta que una característica se considera terminada, cuando se hayan realizado todas las tareas que componen la misma.

Tabla XV. **Roles, entradas y salidas de “Desarrollar características”**

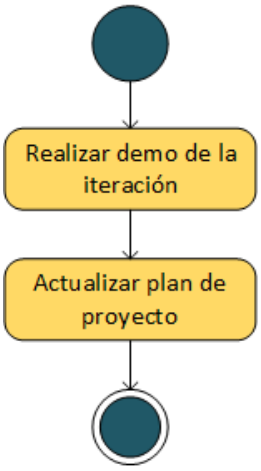
Roles involucrados	Entradas de la tarea	Salidas de la tarea
Líder de proyecto **Equipo de desarrollo**	Plan de iteración - Características del videojuego seleccionadas.	Videojuego ejecutable.

Fuente: elaboración propia.

1.4.4.4. Cierre de la iteración

El cierre de la iteración se registra el estado actual del videojuego y lo ocurrido en el transcurso de la iteración, a modo de poder plasmar en el plan de proyecto la situación actual del mismo. Se realizan las siguientes tareas de forma secuencial.

Figura 12. **Tareas – Cierre iteración**



Fuente: elaboración propia.

1.4.4.4.1. Realizar demo de interacción

El principal objetivo de esta tarea es mostrar al cliente la versión del videojuego que se obtiene tras terminar la iteración. Y la intención de esta tarea es evaluar el cumplimiento de las características que se definieron para la iteración. Si no se alcanzaron a cumplir los requerimientos es bueno documentar las causas y evitar que se repitan.

Tabla XVI. **Roles, entradas y salidas de "Demo de iteración"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Videojuego Plan de iteración - Características del videojuego seleccionadas. - Metas de la iteración.	Documento de evaluación postmortem - Lecciones aprendidas

Fuente: elaboración propia.

1.4.4.4.2. Actualizar plan de proyecto

La idea de tras de actualizar el plan de proyecto, es registrar la situación actual de este, además esto permitirá planificar de mejor manera la siguiente iteración, determinando cambios que sean necesarios para el propósito, en función del monitoreo realizado en la iteración.

Tabla XVII. **Roles, entradas y salidas de "Actualizar plan de proyecto"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	Plan de proyecto	Plan de proyecto (Actualizado)

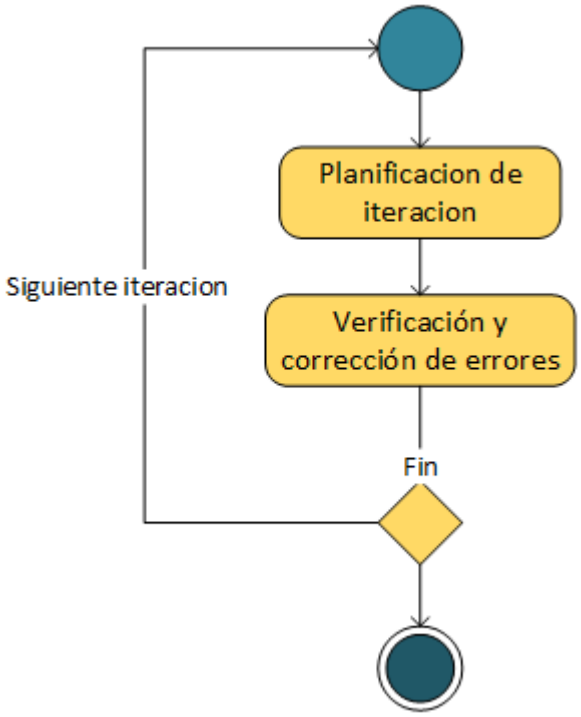
Fuente: elaboración propia.

1.4.5. Pruebas beta

Para esta fase el fin primordial es lanzar una versión beta del videojuego a fin de evaluar y ajustar distintos aspectos del mismo como diversión, contenido, menús, mecánica de juego, etc. Todo esto en función de los resultados que arrojen las pruebas del mismo, y así obtener un producto de mejor calidad. La fase pruebas beta al igual que la fase de desarrollo también es iterativa, ya que al lanzarse una versión beta, esta es probada, si contiene errores entonces son reportados, luego corregidos y se vuelve a lanzar una nueva versión beta, para volver a probar. Este ciclo termina cuando se considera que el videojuego alcanza los criterios de finalización establecidos en el proyecto lo que hará que la versión sea un release estable, listo para su lanzamiento al público.

Para poder realizar la fase de pruebas beta adecuadamente se definen las siguientes actividades.

Figura 13. **Actividades – Fase de pruebas beta**

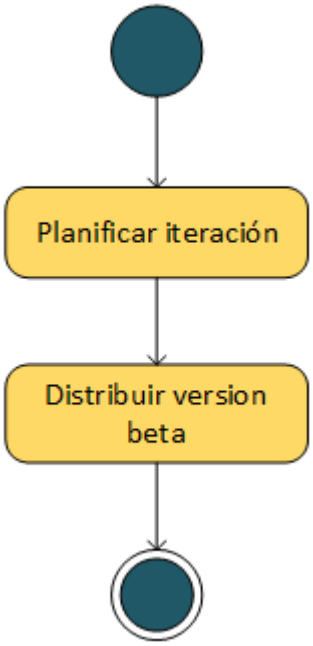


Fuente: elaboración propia.

1.4.5.1. Planificación de iteración

Llegados a esta actividad, ya se tiene una versión jugable del videojuego completa solo para su afinación. Al realizar esta planificación de deben tomar en cuenta los aspectos a evaluar del videojuego y cómo será la distribución beta para que pueda ser probada y verificada.

Figura 14. **Tareas – Planificación de iteración**



Fuente: elaboración propia.

1.4.5.1.1. Planificar la iteración

Cuando se planifica la iteración se deben tomar en cuenta los aspectos funcionales y no funcionales del videojuego en los que se pondrá especial atención en las pruebas de la iteración. También se define a los verificadores del videojuego que evaluaran esos aspectos en el videojuego, la distribución del mismo y la forma en que los errores serán reportados para su corrección. Esta planificación debe realizarse en el lanzamiento de cada versión beta para su verificación.

Tabla XVIII. **Roles, entradas y salidas de "Planificar la iteración"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Líder de proyecto **Equipo de desarrollo**	- Concepto del videojuego. - El videojuego.	Aspectos a verificar y validar.

Fuente: elaboración propia.

1.4.5.1.2. Distribuir la versión beta del videojuego

Distribuir la versión beta del videojuego, implica los medios que serán utilizados para que esta llegue a las manos de los verificadores del videojuego, la manera en que reportaran los errores, y en qué aspectos del videojuego deben enfocarse.

Tabla XIX. **Roles, entradas y salidas de "Planificar la iteración"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Líder de proyecto **Equipo de desarrollo**	- Concepto del videojuego. - El videojuego.	- Aspectos a verificar. - Forma de envío de errores.

Fuente: elaboración propia.

1.4.5.2. Verificación y corrección de errores

La verificación y corrección de errores consiste en primer lugar en verificar la versión beta del videojuego y reportar los errores encontrados, y por ultimo realizar la corrección de los mismos, validando que al mismo tiempo estas

correcciones no lleguen a producir nuevos errores. Los pasos a seguir son los siguientes, se puede avanzar más rápido si se realizan de forma paralela.

1.4.5.2.1. Verificación del videojuego

Este paso solamente cosiste en tomar los aspectos funcionales y no funcionales que se requieran sean verificados y reportar los resultados obtenidos. Estos resultados serán los errores encontrados, estos errores pueden ser de varios tipos como por ejemplo en el código, la calidad de las imágenes, elementos poco atractivos, rendimiento, etc.

Tabla XX. **Roles, entradas y salidas de "Verificación el videojuego"**

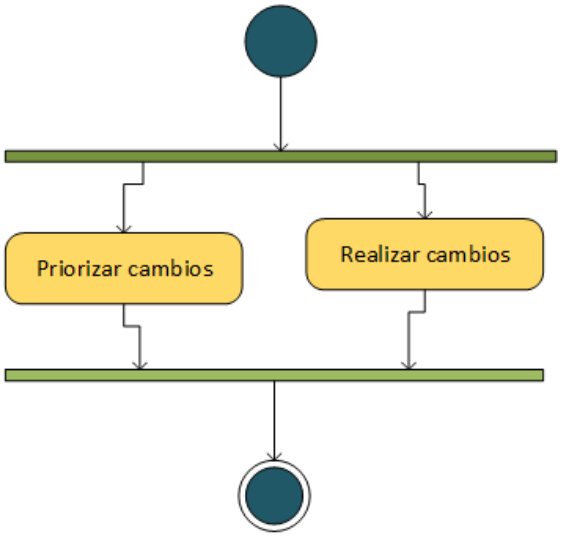
Roles involucrados	Entradas de la actividad	Salidas de la actividad
Verificador del videojuego **Equipo de desarrollo**	- Aspectos a evaluar. - El videojuego.	Aspectos a verificar.

Fuente: elaboración propia.

1.4.5.2.2. Corrección de errores

Para este paso lo primordial es corregir los errores encontrados en el videojuego de acuerdo a los resultados obtenidos por parte de los verificadores del videojuego. De entre los resultados obtenidos puede que haya errores que sean críticos o urgentes los cuales deben de ser corregidos inmediatamente, es por ello que deben de priorizarse los ajustes y luego corregir en función de su prioridad. Se deben de realizar las siguientes tareas de forma paralela.

Figura 15. **Tareas – Corrección de errores**



Fuente: elaboración propia.

1.4.5.2.3. Priorizar cambios

En base a los resultados arrojados por la evaluación y verificación del videojuego se deben definir los cambios que se van a realizar. Estos cambios tendrán que ser ordenados en base al impacto e importancia que tienen para el videojuego.

Tabla XXI. **Roles, entradas y salidas de "Priorizar la iteración"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto** **Equipo de desarrollo**	- Errores encontrados.	Aspectos a verificar y validar.

Fuente: elaboración propia.

1.4.5.2.4. Realizar cambios

Realizar los cambios implica corregir los errores encontrados en base a la lista de cambios priorizados, antes de liberar el videojuego. A la hora de seleccionar el error hay que tomar en cuenta el impacto y el costo que tendrá corregirlo, además que esta elección debe ser informada al equipo.

Una vez ya seleccionado y corregido el error, se debe verificar que haya sido introducido con éxito al videojuego y que este haya sido corregido correctamente, y que esta corrección no haya producido nuevos defectos.

Tabla XXII. **Roles, entradas y salidas de "Realizar cambios"**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Equipo de desarrollo	- Lista de cambios priorizados. - El videojuego	Videojuego corregido.

Fuente: elaboración propia.

1.4.6. Finalización

Llegada esta fase se tiene la versión final del videojuego, la cual queda a disposición del cliente, y además se hace una evaluación del desarrollo del proyecto. Para esta parte se realizan dos actividades, que se ejecutan secuencialmente, la primera consiste en la liberación del videojuego y la segunda en la evaluación del proyecto en sí. Es muy importante que para esta fase participen todas las personas que estuvieron involucradas en el proyecto.

Figura 16. **Actividades – Fase de finalización**

Fuente: elaboración propia.

1.4.6.1. Liberación del videojuego

Para la liberación en sí del videojuego fue seleccionada una versión beta del que fue considerada estable, y se seleccionó como candidata a ser la versión final del videojuego. Para liberar apropiadamente la versión del videojuego se debe tomar en cuenta la plataforma de distribución escogida, ya que puede que se requieran de diversas actividades para distribuir el producto.

El entregable final del producto consiste en el videojuego funcionando correctamente. También puede incluir documentación y otros productos exigidos por el cliente. El entregable final es validado por cliente para dar por terminada la tarea.

Tabla XXIII. **Roles, entradas y salidas de “Liberación del videojuego”**

Roles involucrados	Entradas de la tarea	Salidas de la tarea
Cliente **Líder de proyecto**	- El videojuego.	El videojuego.

Fuente: elaboración propia.

1.4.6.2. Evaluación del proyecto

Para esta actividad se realiza lo conocido como evaluación postmortem en el cual se identifican aspectos relevantes que ocurrieron durante todo el proceso de desarrollo del proyecto, se registran las lecciones aprendidas y se plantean mejoras al proceso de desarrollo. Se realiza esto para hacer más eficiente y productiva la realización de futuros proyectos.

1.4.6.2.1. Evaluación postmortem

Para este apartado el fin es evaluar el proyecto a partir de las decisiones que se tomaron durante el transcurso del proyecto, tomar en cuenta los puntos de vista de cada participante, y las evaluaciones que se hicieron cuando se terminó cada iteración.

Una vez reunida toda esta información se identifican los problemas ocurridos, los éxitos conseguidos, las soluciones encontradas, el cumplimiento de los objetivos y la certeza de las estimaciones. A partir de todo esto se hacen varias conclusiones que dan lugar a las lecciones aprendidas y se buscan otras soluciones para mejorar el proceso de desarrollo.

Tabla XXIV. **Roles, entradas y salidas de "Evaluación postmortem"**

Roles involucrados	Entradas de la actividad	Salidas de la actividad
Líder de proyecto **Equipo de desarrollo**	- Lecciones aprendidas. - Métricas - Mejoras al proceso	- Lecciones aprendidas. - Mejoras al proceso.

Fuente: elaboración propia.

.

Apéndice B.

1. LIBGDX

1.1. Descripción

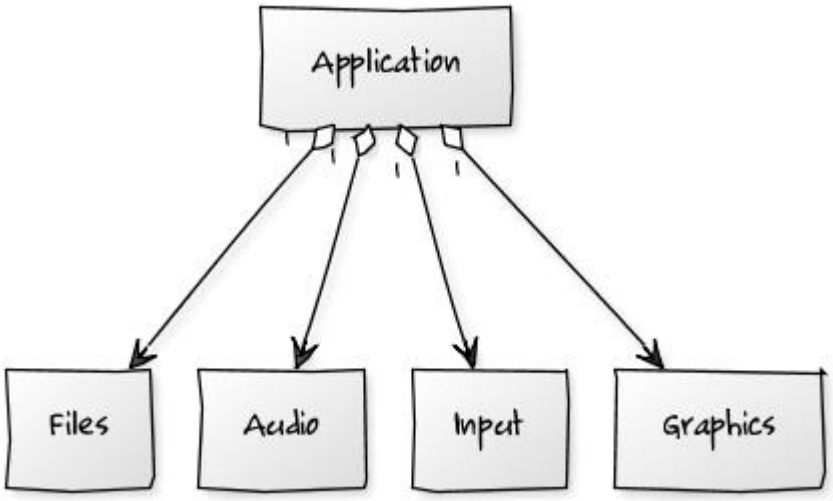
LibGDX es un framework multiplataforma escrito en Java de desarrollo de videojuegos para Windows, Linux, Android, iOS, y Web.

LibGDX permite generar una aplicación para las diferentes plataformas a partir de un código base, de tal manera que el proceso de pruebas y depuración se realiza de forma más rápida y cómoda si se realiza en un PC que directamente en Android por ejemplo. Con LibGDX nos aseguramos de que la misma aplicación puede funcionar correctamente en los diferentes dispositivos. Algo también muy positivo de este framework es que hay mucha documentación, como wikis, videos, etc.

1.1.1. Arquitectura

Cada aplicación desarrollada en el framework se compone de diferentes módulos, que juntos darán vida al videojuego. La siguiente imagen muestra los módulos más elementales.

Figura 1. **Módulos de LibGDX**



Fuente: http://libgdx.googlecode.com/svn/wiki/libgdx-overview.png.

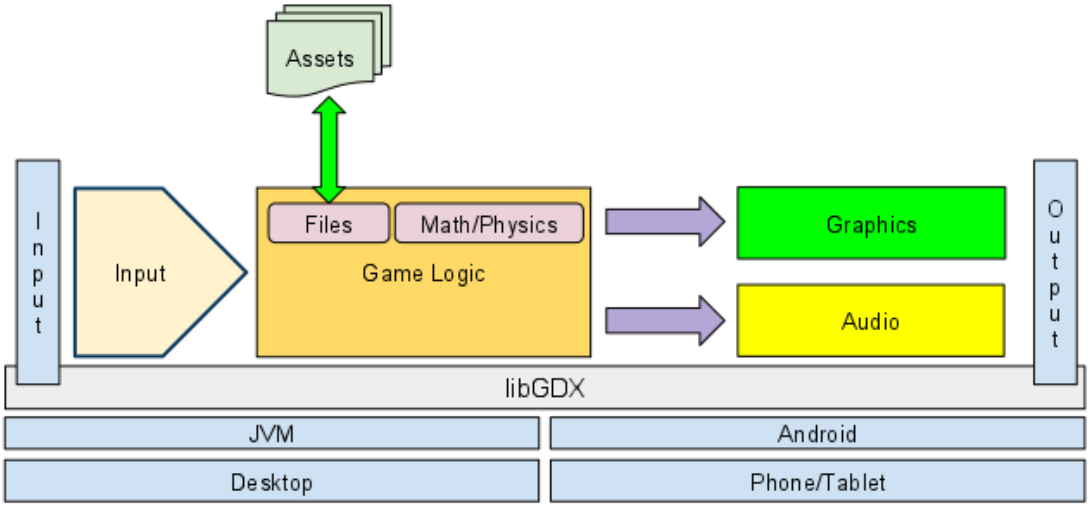
La aplicación será el centro de todo. Esta manejara ciclo de vida del videojuego es decir todos los eventos de creación, destrucción, pausa y resumen de la misma. El componente de Gráficos permite gestionar la representación de imágenes y objetos gráficos que se mostraran en la pantalla.

El componente de Audio ayuda a facilitar el acceso a los sonidos y música que contendrá la aplicación. El componente de Archivos ayuda a gestionar la lectura y escritura de los diferentes ficheros de datos que contendrá el videojuego como imágenes, sonidos, música, archivos de configuración, etc. Y por último el componente de entrada que ayuda a gestionar los eventos causados por el usuario a través del teclado, la pantalla táctil, o acelerómetro.

LibGDX tiene otros módulos útiles como Math que permite una gestión rápida de cálculos matemáticos orientados al desarrollo de videojuegos. Physics o físicas que ayuda a controlar las colisiones de los objetos dentro del videojuego

comúnmente por medio del framework Box2D. La siguiente figura muestra en mayor detalle la arquitectura del sistema.

Figura 2. **Arquitectura de LibGDX**



Fuente: http://libgdx.googlecode.com/svn/wiki/img/modules_overview_diagram.png.

1.1.2. Requisitos del sistema

Para poder desarrollar los videojuegos se necesitan los siguientes requisitos.

- Hardware
  - Una PC o Laptop con Windows 7 o superior o bien con alguna distribución Linux (Se recomienda Ubuntu LTS 14.04 o Linux Mint 16+).
  - 1GB de memoria RAM (2GB si piensa desarrollar para Android)
  - Un procesador de doble núcleo o superior.
  - Al menos 5GB de almacenamiento para albergar todas las herramientas a utilizar.

- Una tarjeta de video con soporte para OpenGL 2.0 (Procurar tener drivers actualizados).

- Software
  - Java Development Kit (JDK) versión 7.
  - Eclipse para desarrolladores Java (Recomendado).
  - Android Development Tools ADT (Plugin para Eclipse).
  - Gradle (Pluging para Eclipse).
  - Para exportar el proyecto a Android, se necesita tener instalado el SDK de Android se recomienda la API 20.

1.1.3. Ventajas y Desventajas

Como cualquier software presenta tanto ventajas como desventajas. A continuación se listan.

- Ventajas
  - Soporte completo para 2D.
  - Exportación a varias plataformas (Android, iOS, Desktop).
  - Bastante documentación, tutoriales, ejemplos de código, etc.
  - Es software de código abierto y gratuito.
  - Actualizada frecuentemente.
  - Se puede probar en una PC antes de subir al móvil.
  - Maneja audio, imágenes, entradas de usuario, archivos, etc.
  - Facilidad para adaptar otros videojuegos hechos en Java para que corra en móvil.
  - Cuenta con una comunidad que crean nuevas herramientas para elaborar proyectos rápidamente.

- Desventajas
    - Para desarrollar en iOS se necesita monotouch. Además de contar con un Mac para el efecto.
    - Pretende dar soporte a 3D pero aún necesita afinarse.

1.2. Funcionamiento

LibGDX puede utilizarse tanto de forma manual como usando un gestor de repositorios y librerías llamado Gradle llamado LibGDX Project Setup, que es de hecho la forma recomendada de utilizar el framework.

1.2.1. Usando Gradle

De forma predeterminada el generador de proyectos de LibGDX genera los proyectos para las diferentes plataformas teniendo uno como principal. Para los proyectos utiliza Gradle, para el manejo del framework mismo de otros que utiliza para funcionar. La interfaz es bastante intuitiva y fácil, solo nos pide unos cuantos datos como se muestra en la figura tres.

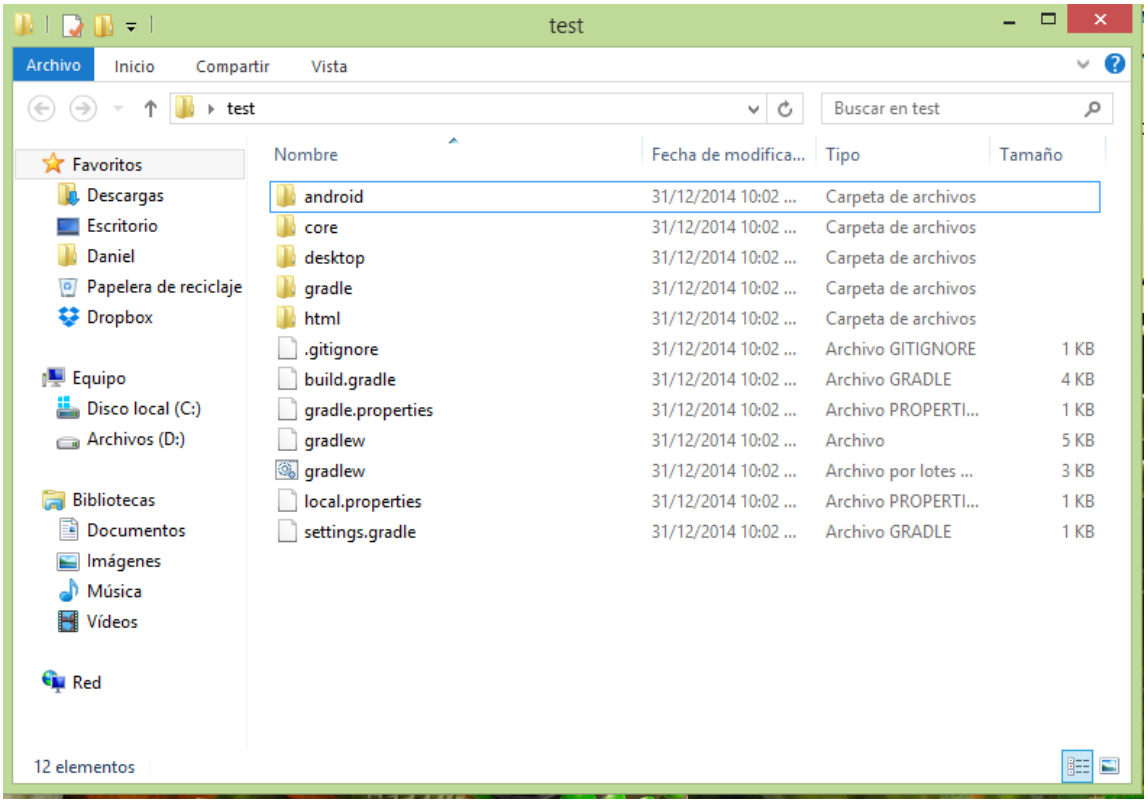
Figura 3. **Generador de proyectos**

Fuente: elaboración propia.

Para poder desarrollar en Android hay que tener instalado el SDK, ya que hace uso de este para construir la aplicación que va dirigida a la misma plataforma. Y para iOS se necesita tener una Mac con el SDK Tools de Apple para el desarrollo.

Al generar un proyecto por primera vez es posible que se tome su tiempo ya que empieza a bajar todas las herramientas que serán necesarias para su creación, esto solo sucede la primera vez. En el directorio elegido crea la siguiente estructura.

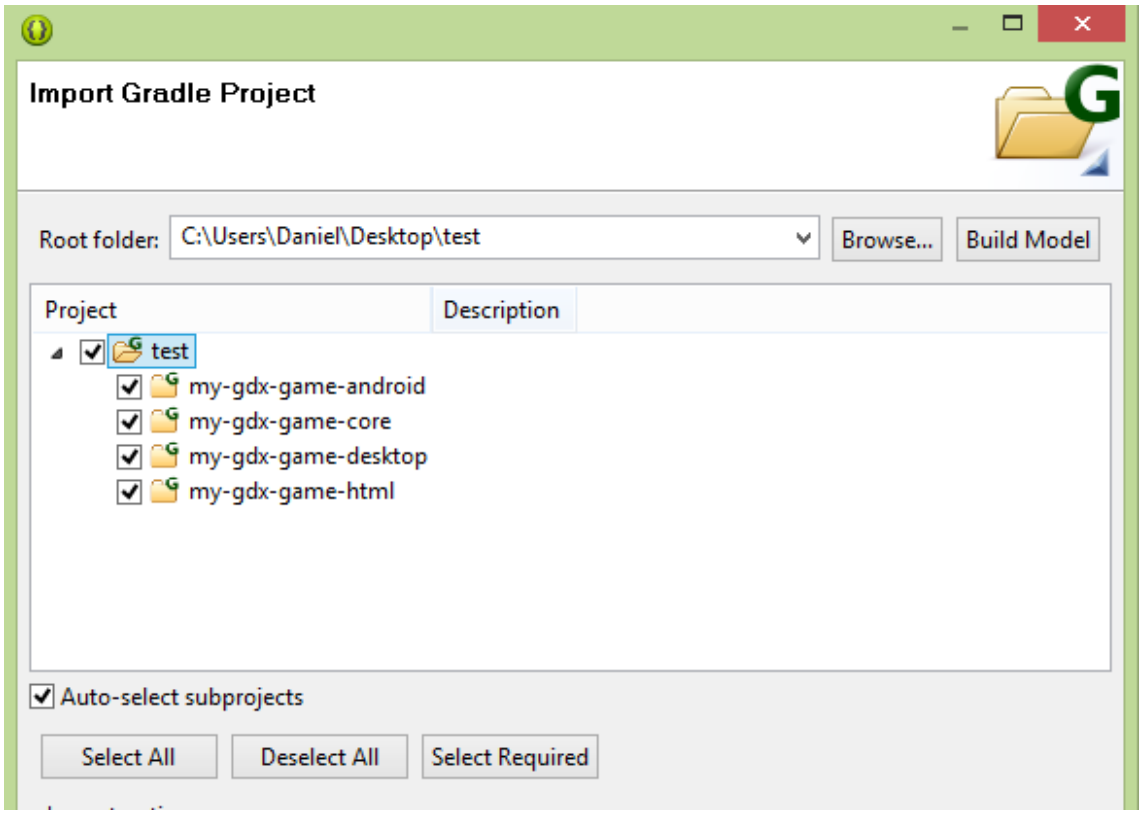
Figura 4. **Estructura de un proyecto en LibGDX**



Fuente: elaboración propia.

Ahora lo que queda es importar el proyecto con un IDE, se recomienda utilizar Eclipse en su versión para desarrolladores Java, con el plugin ADT instalado para desarrollar en Android, y el plugin de Gradle para poder importar el proyecto.

Figura 5. **Importando proyecto**

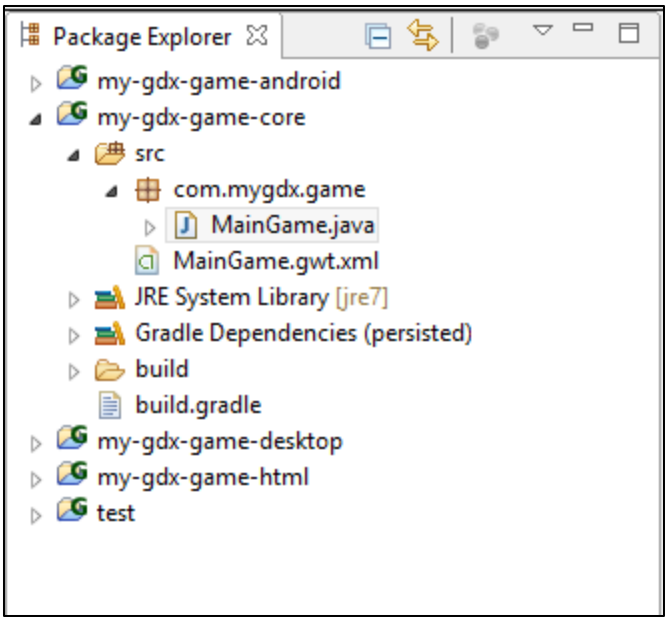


Fuente: elaboración propia.

Media vez se cierre la ventana de finalización, empezara a descargar todas las librerías necesarias para la realización del proyecto esto puede tardar dependiendo de la velocidad de la conexión a internet que se tenga.

Una vez todo termina ya se puede empezar a trabajar en el proyecto, la siguiente imagen muestra cómo quedan los directorios. El desarrollo propio del videojuego se lleva a cabo en el proyecto cuyo nombre termina con core**.**

Figura 6. **Directorios del proyecto**



Fuente: elaboración propia.

1.3. Tutoriales

Al ser LibGDX un framework bastante usado se dispone de bastante documentación para lograr un total dominio de la librería. A continuación se listan varios links útiles.

- Páginas Web
  - Documentación Oficial: contiene una selección de documentos ideales para empezar.
    - http://libgdx.badlogicgames.com/documentation.html
  - Guía para desarrolladores: es un tutorial completo sobre el funcionamiento de la librería. Va desde cero colocando nuestro ambiente de desarrollo hasta las opciones más avanzadas de la librería.

        - https://github.com/libgdx/libgdx/wiki/Introduction
    - Foro Oficial de LibGDX: Es un foro muy activo sobre el uso de la herramienta, puede ser de gran ayuda por si tiene algún problema. Es un foro bastante activo.
        - http://www.badlogicgames.com/forum/
    - Instalando Gradle en Eclipse: Es importante ya que con esto podremos importar el proyecto.
        - http://estiloasertivo.blogspot.com/2013/03/tutorial-howto-install-and-configure.html
    - Tutorial sobre LibGDX: Esta página ofrece un tutorial completo sobre el framework, se actualiza frecuentemente según los cambios que vaya sufriendo este.
        - http://www.gamefromscratch.com/page/LibGDX-Tutorial-series.aspx
    - Tutorial sobre LibGDX en español: Contiene muy buenos contenidos para la realización de videojuegos con esta librería.
        - http://tutoriales.tiarsoft.com/

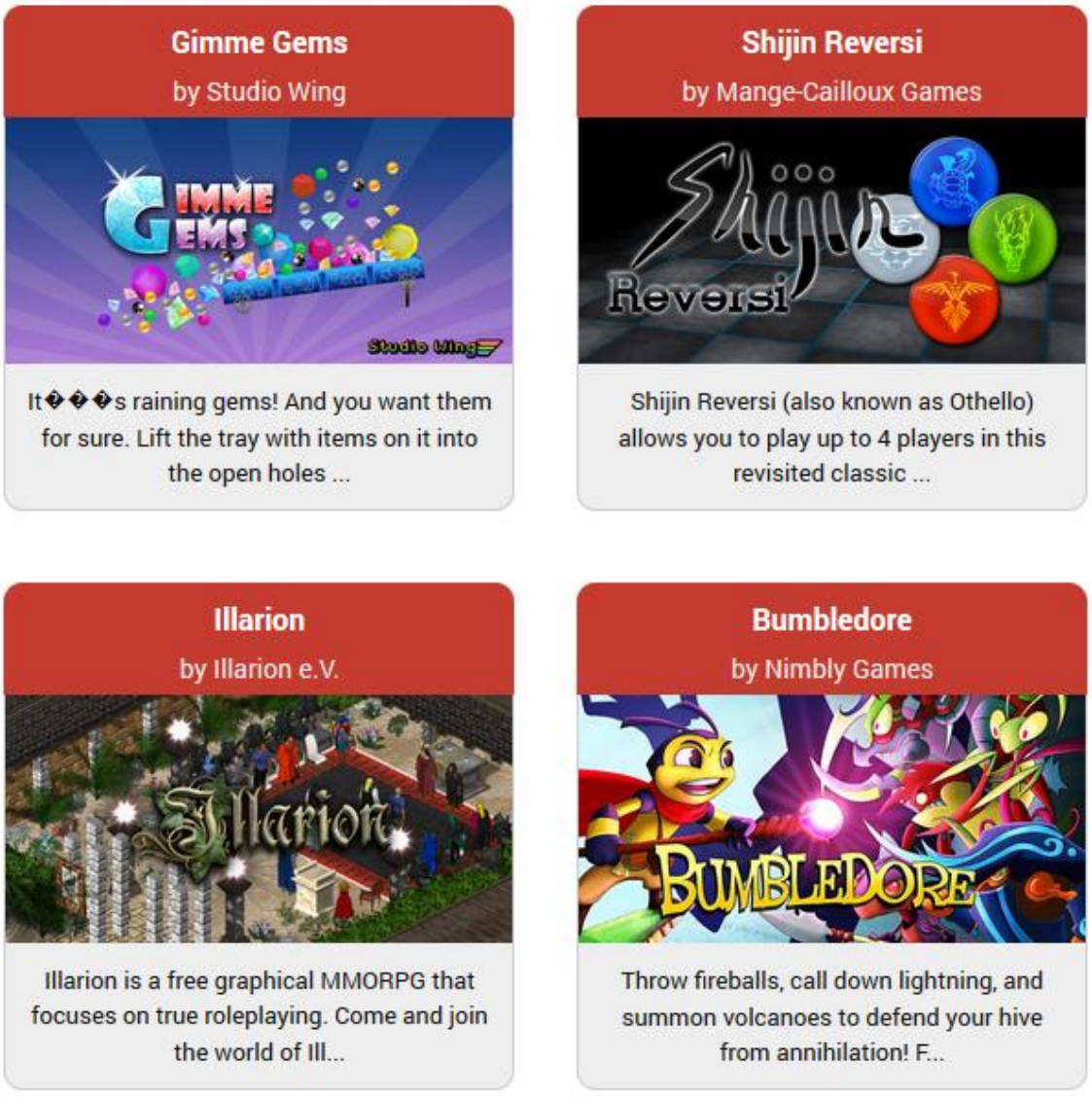
- **Video Tutorial**
    - Este canal tiene una buena lista de reproducción con video tutoriales en español del desarrollo con LibGDX.
        - https://www.youtube.com/watch?v=rdunnEnAxg0&list=PLTd5ehIj0goPFu6_VYCg8G2HSg3D8vWEk

1.3.1. Demos de videojuegos

Algo que es muy bueno, es que hay muchos ejemplos que pueden ser de utilidad para aprender mejor la herramienta. En el gestor de repositorios GitHub hay varios proyectos de código abierto para descargar.

En la dirección https://github.com/libgdx/libgdx/wiki/External-tutorials en el apartado de Open Source Projects hay varios videojuegos hechos con la librería, que pueden utilizarse de referencia para la creación del propio.

Figura 7. **Algunos Videojuegos hechos con LibGDX**



Fuente: *Badlogic Games*. https://libgdx.badlogicgames.com/gallery.html.

1.4. Descarga

A continuación se presenta una lista de enlaces útiles para empezar a trabajar.

- LibGDX
  - http://libgdx.badlogicgames.com/download.html
- Eclipse
  - http://www.eclipse.org/downloads/
- Instalación de ADT
  - http://developer.android.com/sdk/installing/installing-adt.html
- Instalación y configuración de Gradle
  - http://estiloasertivo.blogspot.com/2013/03/tutorial-howto-install-and-configure.html

2. APLICACIONES ÚTILES

A continuación se presentan unas herramientas que pueden ser de mucha ayuda para la realización de los videojuegos educativos, especialmente la parte gráfica de los mismos. Estas aplicaciones son especialmente para tratar las imágenes y no para crearlas.

2.1. TexturePacker

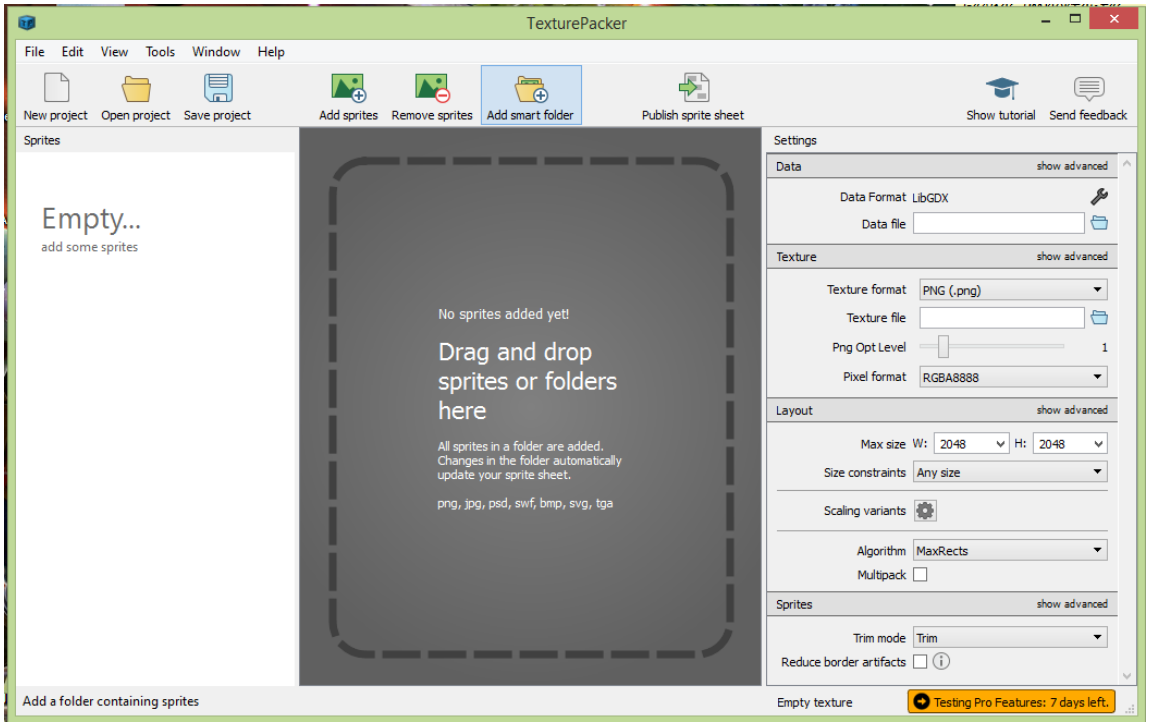
Es una de las herramientas para manejo de sprite sheets que la mayoría de desarrolladores de videojuegos en 2D utilizan, para reducir el consumo de memoria y agregar mejor rendimiento al videojuego. Es realmente una herramienta imprescindible. Esta herramienta permite simplemente arrastrando y soltando las imágenes en la aplicación generar los sprites (también llamados texturas) y sus propiedades pudiendo exportar las propiedades para utilizar diferentes frameworks entre ellos LibGDX.

La versión gratuita del software tiene todo lo necesario, aunque para desarrolladores más experimentados esta la versión Pro la cual es pagada.

Tiene varias características entre las cuales destacan.

- Disponible para Windows, Linux y Mac OS.
- Soporta múltiples resoluciones de pantalla, creando imágenes de alta calidad, y prescalado de imágenes.
- Organización jerárquica de los sprites para tener todo más ordenado.
- Si los sprites son demasiados es posible hacer pack de ellos.
- Reduce el consumo de memoria, sin sacrificar la calidad.

Figura 8. **TexturePacker**



Fuente: elaboración propia.

El software está disponible en https://www.codeandweb.com/texturepacker/download.

2.2. Tiled

Tiled es un editor que permite la creación de mapas a través del uso de tiles. Se pueden agregar más cosas como las posiciones de los enemigos, monedas, algunos ítems del videojuego, etc. La siguiente imagen muestra un videojuego realizado con la ayuda de Tiled.

Figura 9. **Mapa hecho con Tiled**

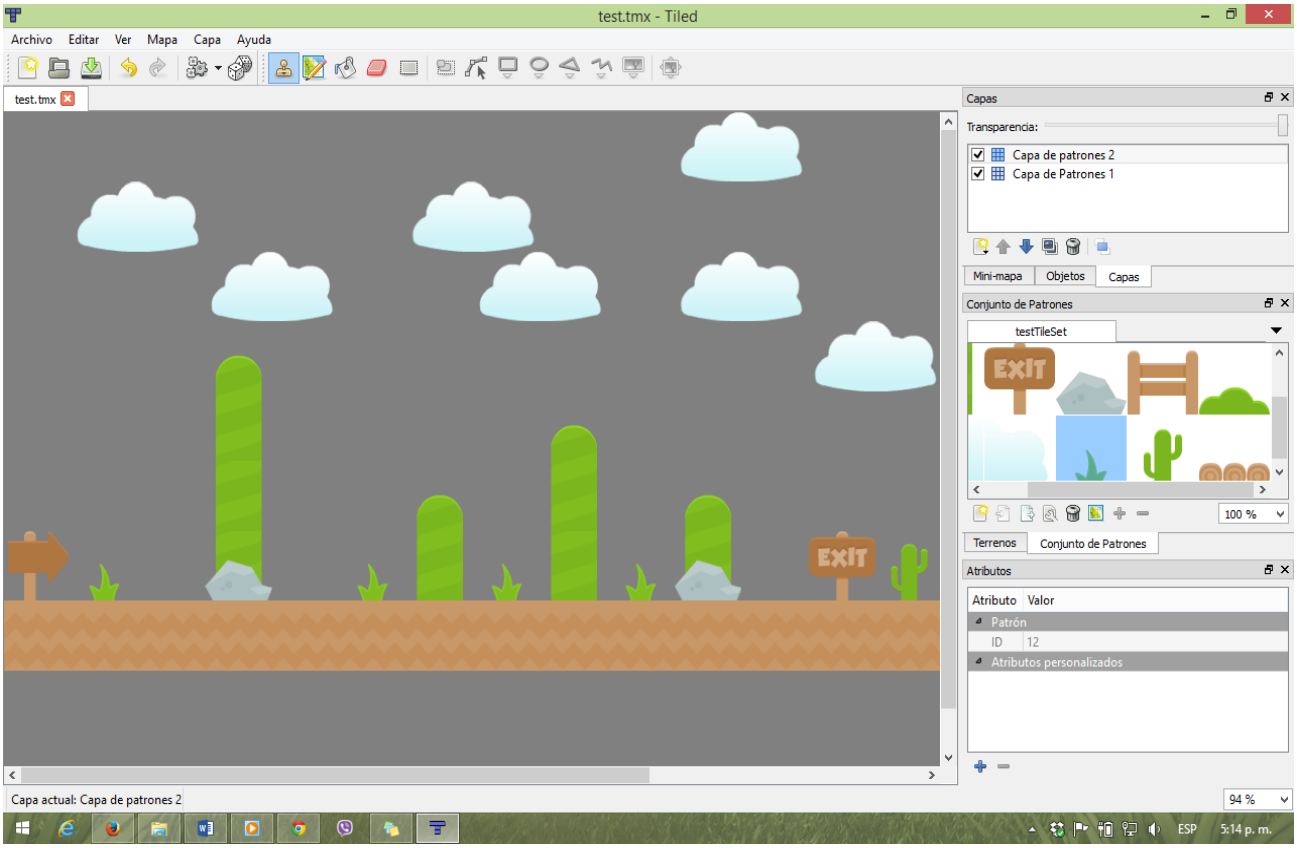
Fuente: Tiarsoft. http://2.bp.blogspot.com/-5OnGRBE4B7A/U4D3acbiF8I/AAAAAAAAAEs/EDAczh2rsN8/s400/3.png.

Tiene varias características entre las que destacan.

- Está disponible para Windows, Ubuntu, Fedora y Mac OS
- Es totalmente gratuito y de código abierto.
- Utiliza Sprite Sheets para la creación de los mapas o mundos.
- Editor de propósito general.
- Manejo de mapas tanto ortogonales como isométricos.
- Soporte de múltiples capas tanto de tiles como de objetos.

En resumen su función básica es crear mundos a través de la colocación de imágenes en diferentes posiciones para formar un tile.

Figura 10. **Como se ve el software y un ejemplo**



Fuente: elaboración propia.

El software se encuentra disponible en: http://www.mapeditor.org/download.html.

Apéndice C.

1. ADMINISTRACIÓN DE LA CONFIGURACIÓN

1.1. Descripción

En el transcurso de la fase de desarrollo, y más si se trabaja en equipo debe haber una estructura que permita, de alguna manera manejar los diferentes ambientes, versiones del código y el software. El propósito de establecer una adecuada administración de la configuración es identificar el estado del videojuego educativo en distintos puntos en el tiempo, con el propósito de controlar de manera sistemática los cambios a la configuración y mantener la integridad de dicha configuración a lo largo del ciclo de vida del videojuego. A continuación se explican los lineamientos recomendados a seguir para la administración de la configuración.

1.2. Política de versionado de los videojuegos educativos

Todos los videojuegos educativos desarrollados en Edulibre (también puede aplicarse a otros proyectos de software), deberán seguir esta política, al momento de ser lanzados y distribuidos por los medios que Edulibre haya decidido, en la planificación del proyecto.

1.2.1. Numeración de versiones

Existen diversas técnicas para la numeración de versiones de software, proponiendo para los proyectos de Edulibre la más extendida dentro de los proyectos de software libre. Es una notación numérica compuesta por tres

números, con algunas variantes. Cabe aclarar que los números siempre van en orden creciente, y son enteros no negativos.

Figura 11. **Forma de la Numeración**

vX.Y.Z[beta]

Fuente: elaboración propia.

- vX: es un número entero que indica una versión principal del videojuego educativo, consistiendo en un conjunto de funcionalidades cubiertas en dicha versión, y que son visibles al usuario. La letra “v” se coloca por comodidad para indicar que se trata de una versión de software, por ejemplo *v1.2.4*. Si este número incrementa “Y” y “Z” se vuelven cero.
- Y: es un número entero que indica un conjunto nuevas funcionalidades agregadas al videojuego. Si este número incrementa “Z” se vuelve cero. En el desarrollo, se podría tomar como referencia la iteración realizada.
- Z: es un número entero que se modifica cuando hay revisiones de código ante fallos encontrados en el videojuego, y no funcionalidad nueva. Este número se puede incrementar indefinidamente, siempre y cuando “Y” no haya cambiado, de lo contrario vuelve a cero.
- [beta]: esta etiqueta solo se utilizara para las versiones beta, que están sometidas a pruebas e indica que es una versión inestable. La versión de lanzamiento o estable no tendrá esta etiqueta.

1.2.2. Criterios para modificación de versiones

A continuación se listan los criterios para la modificación de la numeración de versiones. El criterio de cambiar o no cambiar los números de versión estará a cargo del administrador del repositorio.

- vX: Nuevas funcionalidades clave del videojuego respecto a la versión anterior debido a la inclusión de nuevas características para el mismo, como lo pueden ser:
  - Inclusión de nuevos módulos.
  - Una revisión completa de los ya existentes.
  - Reescritura de funcionalidades por cambio de versión de frameworks, utilizados para readaptar funcionalidades.
  - Readecuación de la idea inicial del videojuego, etc.
- Y: implica funcionalidades nuevas o que amplían las ya existentes.
  - Un nuevo nivel creado.
  - Agregado un nuevo modo de juego
  - Ampliación de contenidos del videojuego.
  - Cambios en la estética del videojuego, etc.
- Z: cualquier fallo encontrado en el videojuego ya sea durante las pruebas o después de su lanzamiento y uso por parte de los usuarios que reportaron el bug, como por ejemplo.
  - Problemas en dispositivos específicos
  - Problemas de rendimiento, etc.

1.3. Administración de repositorios

Es importante establecer un patrón de trabajo a nivel de la institución, a seguir por todos los involucrados en el desarrollo de los videojuegos, a modo realizar más fácilmente la tarea, y evitar complicaciones. El modelo de ramas es el propuesto por Vincent Driessen en su artículo titulado “A successful Git branching model” con algunos agregados. Es un modelo muy utilizado para diversos proyectos.

1.3.1. Git

Git es una herramienta de control de versiones distribuido, abierto y gratuito; muy utilizado por proyectos de software libre en todo el mundo. Los desarrolladores tendrán la obligación de usarlo correctamente en todos los proyectos en que estén trabajando. Una vez instalado debe configurarse el nombre y el correo del desarrollador para que sus *commits* sean identificados correctamente. En el capítulo dos de este documento se puede encontrar información al respecto.

1.3.2. Modelo de ramas

Todo videojuego educativo que se realice, (este modelo también se puede adoptar para cualquier otro proyecto de software), deberá seguir el siguiente patrón de trabajo para los repositorios. La creación de estas ramas puede realizarse de forma manual o bien utilizando la herramienta git-flow disponible para los diferentes sistemas operativos.

Existirán dos ramas principales en el repositorio remoto (Bitbucket):

- Rama master: esta es la rama principal que contiene la aplicación lista producción. Esta rama contiene la última versión estable del videojuego educativo. Para trabajar con esta rama se debe tomar en cuenta lo siguiente:
    - Cuando la rama develop es lo suficientemente estable, se puede crear una rama release, lo que lo hará una versión beta.
    - Es responsabilidad del administrador del repositorio decidir cuando la rama release pasa a master y a develop, ocasionando con ello un cambio de versión en la rama master.
    - La rama master tiene que tener los tags de versionado, es decir manejar la numeración de versiones, para mayor referencia ver la política de versionamiento en el punto 1.2 de este capítulo.
    - Tomar en cuenta que cada merge hecho en master es una nueva versión lista para producción.

- Rama develop: en esta rama se encuentra todo el código que conformara la siguiente versión planificada del proyecto. Y del cual se desprende un release candidato para las pruebas beta.

1.3.2.1. Ramas auxiliares

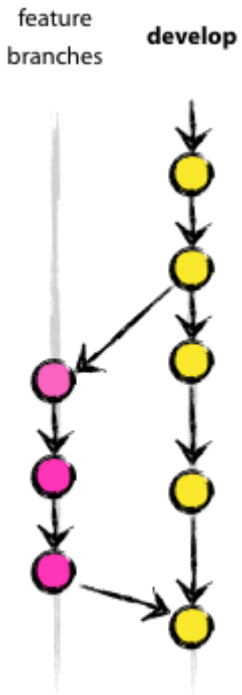
La idea principal del uso de ramas auxiliares es ayudar al desarrollo en paralelo entre los miembros del equipo, facilitando la implementación de requerimientos, preparar el software beta para producción y asistir de forma rápida a la reparación de fallos.

Cada una de estas ramas tiene un propósito en específico y están sujetas a estrictas normas de funcionamiento. Es importante mencionar que son

sugeridas y que se pueden agregar o quitar según se necesite, pero estas suelen ser las más usuales.

1.3.2.1.1. Ramas de características (feature)

Figura 12. **Ramas de características**



Fuente: *A Successful Git Branching model*. http://nvie.com/img/fb@2x.png.

Estas ramas:

- Se origina a partir de la rama develop.
- Se incorporan siempre a la rama develop.
- Nombre: feature-*. Donde * es el nombre de la funcionalidad a implementar.

Estas ramas se utilizan para desarrollar las nuevas características serán parte del próximo release, que una vez terminadas, se incorporan a la rama develop. La esencia de una rama de característica es que existe, siempre y cuando la característica está en desarrollo, pero finalmente se fusiona de nuevo en develop o bien se descarta. Comúnmente cuando la rama se fusiona con develop esta es eliminada, ya que su utilidad termino.

1.3.2.1.2. Ramas de release (release)

Estas ramas:

- Se originan a partir de la rama develop
- Se incorporan o fusionan a master y develop
- Nombre: vX.Y.Zbeta (revisar política de versionamiento)

Estas ramas se utilizan para preparar el siguiente código que pasara a producción. En estas ramas se hacen los últimos ajustes y se corrigen los últimos pequeños bugs antes de pasar el código a producción incorporándolo a la rama master y luego a la rama develop. Realizando todo el trabajo sobre esta rama, la rama develop está lista para recibir nuevas características que darán paso a la próximo release.

El pasar a esta rama significa que la rama develop ya llego a un punto en el cual se le considera casi estable, es decir que casi todas las características han sido implementadas, (El conjunto de características suelen pertenecer a una iteración del software).

1.3.2.1.3. Ramas de revisiones (hotfixes)

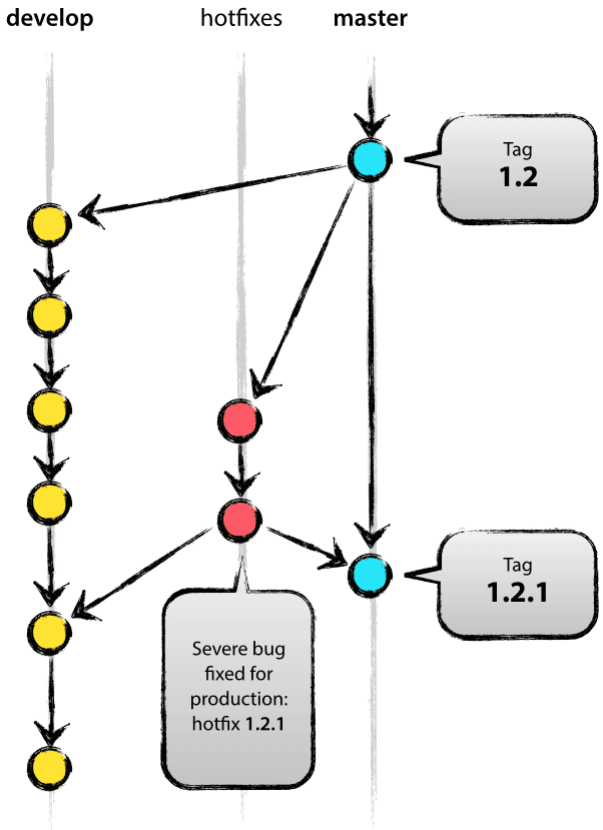
Estas ramas:

- Tienen su origen a partir de la rama master.
- Se fusionan o incorporan a la rama master y develop.
- Nombre: hotfix-[vX.Y.Z]. Donde lo que está entre corchetes es la versión. a la que se le aplica el parche. Los corchetes no son parte del nombre.

Estas ramas son como las ramas de release, en las que también se prepara la aplicación para producción, con la diferencia que estas ramas no son planeadas. Surgen de la necesidad de resolver algún bug crítico del software, que está afectando el comportamiento de la versión actual en producción. Cuando un bug crítico debe ser resuelto, esta rama debe surgir de la versión actual correspondiente en la rama master.

En esencia el trabajo del equipo de desarrollo en la rama develop, no debe verse afectado, mientras otra persona resuelve un fallo en la versión de producción.

Figura 13. **Rama de revisiones**



Fuente: *A Successful Git Branching model*. http://nvie.com/img/hotfix-branches@2x.png.

No hay que olvidarse de modificar el número de versión después de la ramificación, a la rama master. Una vez la rama de revisión fue fusionada con master y develop, se puede eliminar ya que no será de utilidad.

Una observación importante, es que si cuando se creó la rama de revisión ya existía una rama de release, el arreglo debe incorporarse a esta, ya que de todos modos los arreglos hechos al release, y los incorporados por la revisión deben de fusionarse con la rama develop.

1.3.3. Herramientas

1.3.3.1. Servidor

Se utilizara Bitbucket como plataforma para almacenar todos los repositorios de forma centralizada, por lo que los integrantes de equipo deben contar con una cuenta de usuario en dicha plataforma. En el capítulo dos de este documento se explica con mayor detalle la plataforma.

1.3.3.2. Clientes

Existen diversas herramientas de trabajo con Git. Git posee su propia línea de comandos con la cual se puede trabajar, pero se puede hacer más eficiente el trabajo, utilizando alguna herramienta GUI para la administración de los repositorios. El capítulo tres de este documento sugiere algunas herramientas para este cometido.

1.4. Buenas prácticas

Cuando se trabaja en un proyecto y más si este está a cargo de un equipo que estará manejando los diferentes archivos de este en el repositorio, puede resultar mal si no se hace de forma adecuada. Se sugieren un conjunto de buenas prácticas, que ayudaran a tener un mejor control sobre el repositorio, y evitar así inconvenientes.

Cuando se trabaja con más personas, algo muy importante es el uso de una guía de estilo para escribir código, y usar las mismas configuraciones del editor de texto, eliminación de espacios al final de las líneas, saltos de línea al final del fichero, anidaciones, etc. Se evitará modificaciones innecesarias.

1.4.1. Repositorio

Estas prácticas se pueden aplicar tanto al repositorio remoto como al local.

- No reescribir la historia ya publicada, es decir no usar rebase sobre *commits* que ya estén en el servidor remoto.
- Trabajar sobre una rama puntual, nunca sobre las de largo recorrido (develop y master).
- El registro de cambios debe ser granular, de tal forma que cada *commit* incluya en lo posible cambios a una sola funcionalidad.
- Hacer *commits* con mayor frecuencia.
- Hacer siempre un *pull* en master siempre antes de mezclar una rama sobre *master*.
- Revisar el código fuente y asegurarse que se hayan incluido todos los archivos requeridos.
- Evitar hacer *commits* de funcionalidades sin terminar.
- Utilizar el modelo de ramas propuesto, para el manejo de varios ambientes y evitar caos en el repositorio.
- Utilizar tags para definir versiones de la aplicación, según la política de versionado.
- Seguir el formato para los commits.

1.4.2. Nombres de ramas o branches

El colocar nombres a las ramas es algo totalmente arbitrario y queda al juicio del usuario, pero muchas veces se llega a confusiones dentro de un equipo, es por ello que se recomienda seguir algunas recomendaciones y mejores prácticas, para ser mejores usuarios dentro de la plataforma.

- Respetar los nombres por defecto del modelo de ramas.
- A pesar de que se puede utilizar el símbolo / en el nombre de las ramas, este no puede ser el carácter final de un nombre.
- No colocar punto (.) después de un slash (/).
- No colocar puntos suspensivos seguidos (…) dentro de un nombre.
- No utilizar caracteres especiales (~ ^ : ? * []) ya que estos caracteres tienen significado dentro la sintaxis de Git y se pueden prestar a errores.
- No se debe tener espacios en blanco o caracteres de control ASCII.

1.4.3. Formato de commits

Los commits deben ser lo más atómicos posibles y respetar las buenas prácticas para la confección los mensajes, las cuales se resumen a continuación:

- Una primera línea con una descripción general de menos de 50 caracteres.
- Si es necesario, dejar un alinea en blanco y escribir una descripción más detallada, máximo 72 caracteres.
- Todo esto ayuda a una mejor legibilidad y a posteriores tareas como por ejemplo escribir un log de cambios o notas de lanzamiento de versión.

2. BITBUCKET

2.1. Descripción

Es un servicio de alojamiento en la nube para proyectos de software en los que se puede utilizar el sistema de control de versiones Git o Mercurial. Fue lanzado en el año 2008 por la empresa Atlassian Software y está escrito en Python mediante el framework web Django.

El modelo de negocio de Bitbucket se basa en la cantidad de usuarios activos para trabajar en los diferentes proyectos. Si el equipo supera los cinco hay que empezar a pagar por cada miembro que se una, aunque se puede llegar a ocho si se invitan a tres a unirse al servicio.

2.2. Requisitos del sistema

Para poder utilizar e interactuar adecuadamente el repositorio en la nube se deben llenar los siguientes requerimientos, tomando en cuenta el uso de clientes para el repositorio y el uso del software Git.

- Hardware
  - Una PC o Laptop con Windows 7 o superior o bien con alguna distribución Linux (Se recomienda Ubuntu LTS 14.04 o Linux Mint 16+).
  - 2 GB de memoria RAM (Para condiciones óptimas 3GB o más)
  - Un procesador de doble núcleo o superior.
- Software
  - Tener Git instalado y configurado adecuadamente (Disponible para Windows, Mac y Linux).

- Cliente para interactuar con el repositorio en la nube (Opcional).
- Un explorador de internet compatible, se recomienda Firefox 29 o superior o bien Google Chrome.

2.3. Ventajas y Desventajas

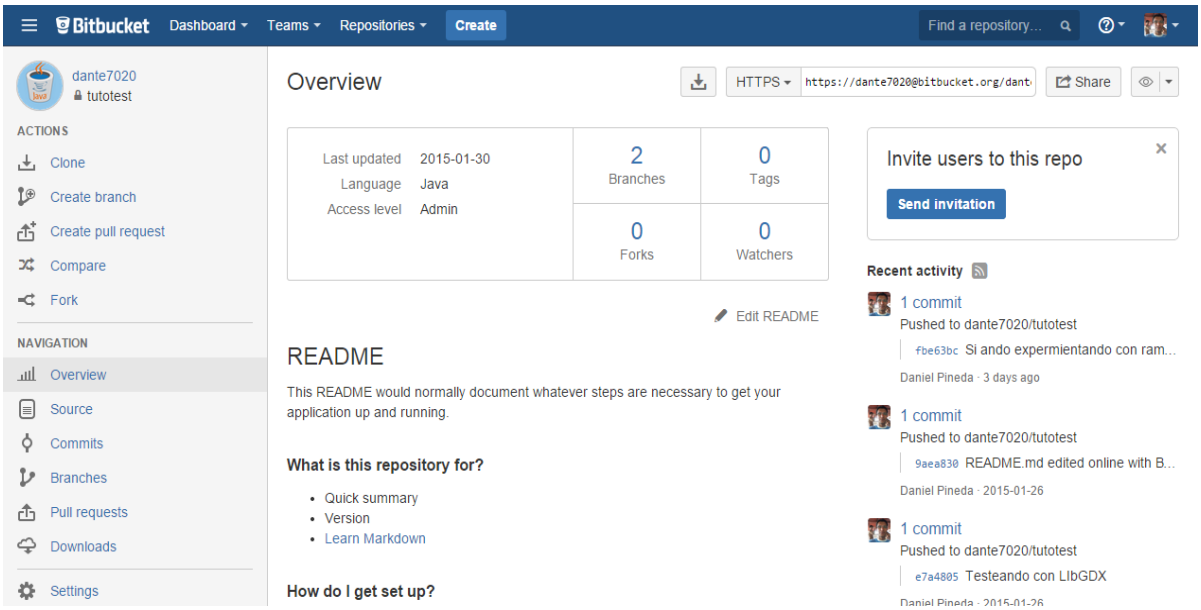
Como cualquier producto presenta tanto ventajas como desventajas las cuales se listan a continuación.

- Ventajas
  - Repositorios privados ilimitados.
  - Espacio de almacenamiento ilimitado.
  - Cinco usuarios gratuitos para crear un pequeño equipo de desarrollo.
  - Reconocimiento de varios lenguajes de programación.
  - LibGDX trabaja bien.
  - Posee un cliente gratuito llamado SourceTree para Windows y Mac (La versión para Linux no es prioritaria para ellos).
  - Interfaz web muy intuitiva para llevar el control y manejo del proyecto.
  - Los repositorios privados también pueden volverse públicos.
- Desventajas
  - Si el número de usuarios supera los cinco se tiene que empezar a pagar.

2.4. Funcionamiento

Con la interfaz web es posible hacer configuraciones al proyecto, crear archivos, ramas de desarrollo, forks, etc. Aunque su fin primordial es albergar el proyecto en la nube de forma segura.

Figura 14. **Interfaz gráfica**



Fuente: elaboración propia.

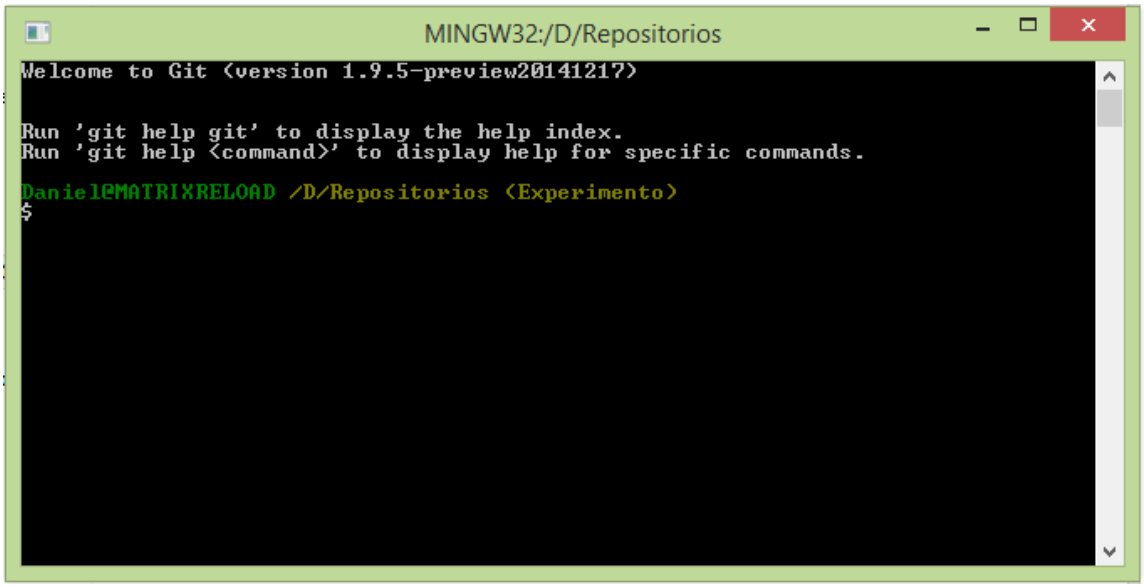
2.4.1. Git

Para poder interactuar con los repositorios albergados en Bitbucket es necesario tener instalado Git en nuestra computadora, ya que este software nos permitirá conectarnos al repositorio albergado en la nube para subir nuestros cambios, bajar otros cambios realizados, etc.

Al utilizar Git hacemos uso de un control de versiones distribuido, esto quiere decir que todas las computadoras conectadas albergan una copia exacta de lo

que hay en el servidor, lo que permite trabajar y avanzar bastante en el proyecto sin tener que estar todo el tiempo conectado al servidor para guardar los cambios realizados (como es el caso de Subversion). Además es necesario para comunicarnos con nuestro repositorio remoto albergado en Bitbucket.

Figura 15. **Git Bash Windows**



Fuente: elaboración propia.

Cada repositorio Git maneja su propia copia, y su propia historia de manera local a través de la línea de comandos.

2.5. Tutoriales

Git es un software ampliamente utilizado para el control de versiones, tanto para proyectos de código abierto, como cerrado. Y para un mejor dominio del repositorio en la nube, se necesita de su conocimiento. Algo bastante bueno es

que hay mucha información disponible. A continuación se listan varios links útiles.

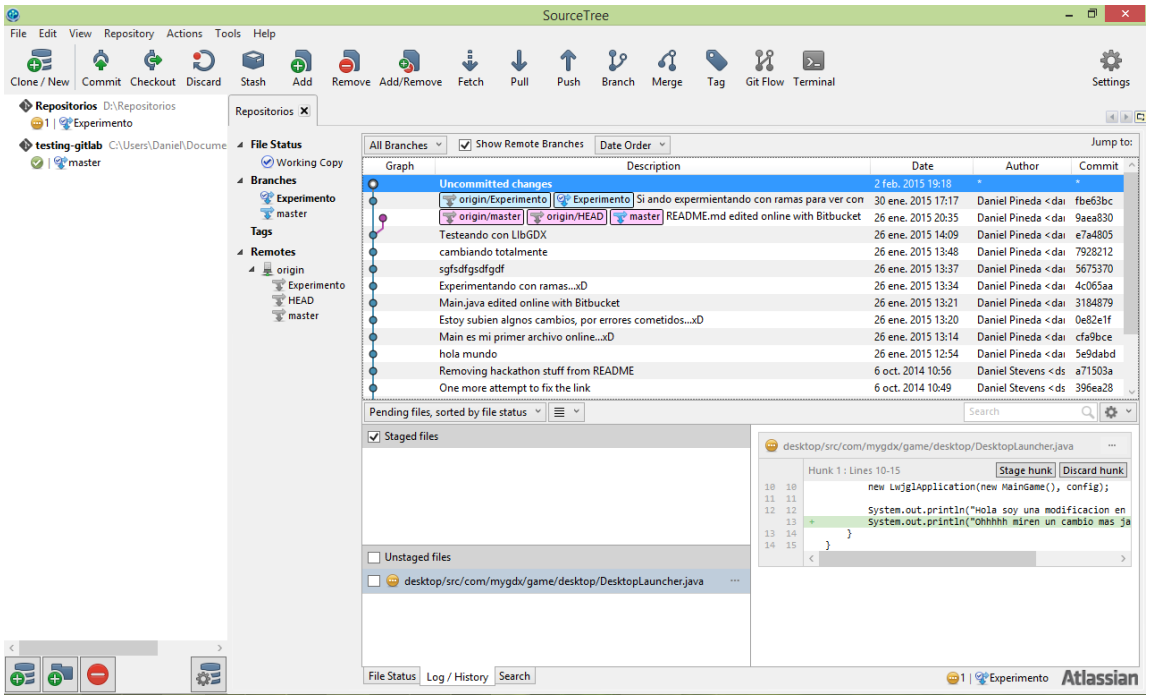
- Páginas Web
  - Documentación oficial Bitbucket: aquí se encuentra todo lo relacionado con la herramienta, su correcto uso.
    - https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+Documentation+Home
  - Pro Git: en esta página se encuentra todo lo que hay que saber sobre Git, está muy completa y aunque aún falta para su traducción al español, no le falta mucho por lo que puede empezar a usarse.
    - http://git-scm.com/book/es/v1
  - Instalación de Git: abarca su instalación tanto para Windows, Linux y Mac.
    - http://git-scm.com/book/es/v1/Empezando-Instalando-Git
  - Tutorial Git: es un tutorial bastante simplificado que abarca lo esencial que hay que saber sobre Git incluyendo su interacción con Bitbucket.
    - https://www.atlassian.com/git/
  - Bitbucket 101: es un tutorial para iniciar en Git y su uso a través de la interfaz web. Explica todo lo necesario para empezar a utilizar el repositorio.
    - https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+101

3. APLICACIONES ÚTILES

3.1. SourceTree

SourceTree es un cliente GUI para manejar repositorios Git y Mercurial. Se integra muy bien con Bitbucket. Solo está disponible para Windows y Mac, por ahora la empresa no planea lanzar una versión para Linux. Lo mejor de todo es que es completamente gratuita con todas sus funcionalidades, solamente hay que registrarse para poder usarla por completo.

Figura 16. **Interfaz gráfica**



Fuente: elaboración propia.

Tiene varias características entre las cuales destacan.

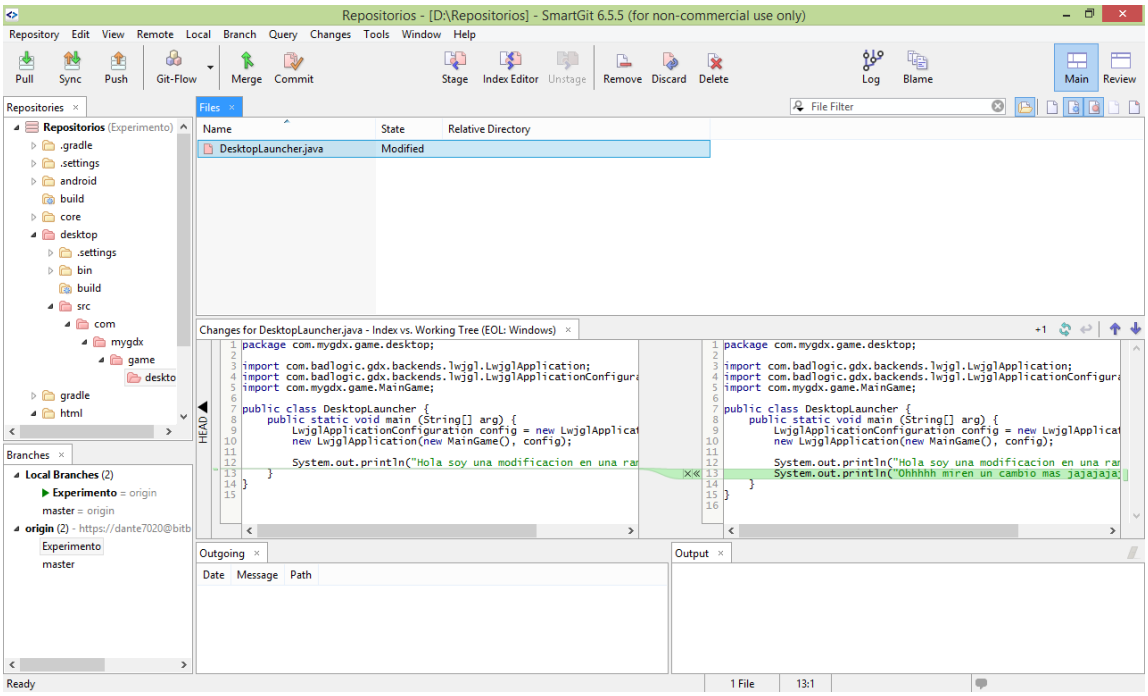
- Interfaz gráfica amigable para el manejo de varios repositorios.
- Todas las operaciones de Git.
- Creación y clonación de cualquier sitio, tanto Git como Mercurial.
- Detección y resolución de conflictos.
- Consulta de historial de cambios de los repositorios.

El software está disponible en http://www.sourcetreeapp.com/.

3.2. SmartGit

Es un cliente GUI para Git disponible para Windows, Mac y Linux. Es completamente gratuito para uso no comercial, provee acceso grafico a repositorios Git o Mercurial, lo que implica que se integra muy en Bitbucket. Para la versión no comercial es completamente funcional, se dispone de todas las opciones y no es necesario registrarse.

Figura 17. **Interfaz gráfica**



Fuente: elaboración propia.

Tiene varias características entre las cuales destacan.

- Una interfaz gráfica bastante intuitiva para manejar los repositorios.
- Realiza todas las operaciones de Git.
- Modificación de *commits* antes de hacer *pushing*, *commits* individuales de líneas, recuperación de *commits* anteriores y más.
- Compatible con varios repositorios populares en la nube.
- Trae herramientas SSH client, un comprador de archivos y una herramienta para hacer merge.
- Manejo de logs, etc.

El software está disponible en http://www.syntevo.com/smartgit/download.