Universidad Carlos III de Madrid

Ingeniería en Informática

Proyecto Fin de Carrera

Desarrollo de un videojuego educativo para el aprendizaje de inglés mediante teléfonos móviles

Autor: Eugenio Pérez Martínez

Tutor: Telmo Zarraonandia Ayo

Septiembre de 2011

Agradecimientos

A mis padres y mi hermana por todo el apoyo y facilidades que me han dado durante todos estos años.

A mi prima Débora por su ayuda prestada con el diseño gráfico.

En especial a Lara, que en los momentos difíciles siempre ha estado a mi lado, dándome ánimos y apoyándome para que un día como hoy pueda presentar mí Proyecto Fin de Carrera.

Abstract

The aim of this project is to study the possibilities offered by the so-called m-learning, learning supported by mobile devices, the Android platform, and game engines compatible with this platform. An analysis of the characteristic and the scope of Android Platform, in the mobile learning environment, has been carried out to develop this project. "Maz- E-nglish" is a game which allows you the learning of different skills associated with language learning. The goal of the game is to explore various mazes where the player must overcome several challenges both didactic and logical. The game has been developed for the Android platform and goes with an application that gives the possibility to the teacher to create new mazes associated with new sets of exercises.

Índice de contenido

Índice de ilustraciones

Índice de tablas

Índice de códigos

Capítulo I: Introducción

A lo largo de este capítulo se pretende dar al lector una perspectiva del panorama actual en el mercado de los teléfonos móviles, ofrecerle una breve introducción y presentación de los objetivos del proyecto y las motivaciones que me han llevado a la realización del mismo, para concluir se presentará un breve resumen del contenido de esta memoria.

1.1 Panorama actual

Los nuevos teléfonos móviles y smartphones ocupan actualmente un 75% de las ventas de dispositivos de telefonía móvil en el mercado. Esto ha provocado, entre otros fenómenos, que los ingresos de las compañías telefónicas se estén invirtiendo y pasen de provenir mayoritariamente del tráfico de voz y de los mensajes cortos al tráfico de datos (*ilustración 1*). Este tipo de dispositivos va mucho más allá de las prestaciones típicas de un terminal con servicios solo de telefonía, proporcionando multitud de servicios y funciones que podemos explotar de diferentes maneras a través de múltiples aplicaciones. Uno de los ámbitos donde estas posibilidades son especialmente interesantes es el de la educación ya que el uso de dispositivos móviles como soporte o complemento del proceso educativo permite continuar aprendiendo en cualquier momento y en cualquier lugar. A esta modalidad de aprendizaje se ha venido a denominar *mobile-learning* o *m-learning* y existen ya en el mercado multitud de aplicaciones de este tipo como por ejemplo *Bon dèpart: Beginners´ French [31]* para la enseñanza del francés, *ADL mLearning Guide [32]* cuyo objetivo es proporcionar un recurso universal sobre todos los temas pertinentes para el aprendizaje móvil o *Brain Training [7]* para ejercitar la memoria.

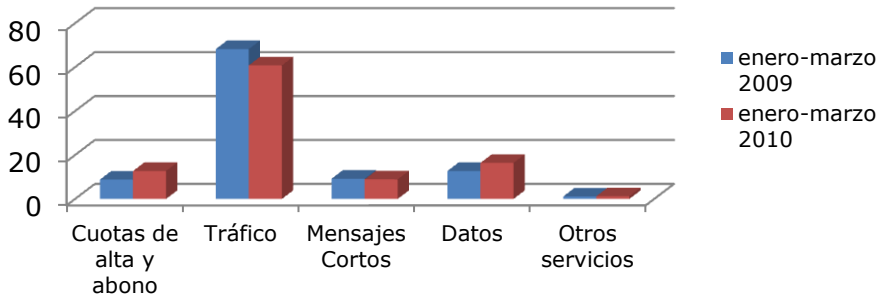


Ilustración 1: Porcentaje de ingresos obtenidos por las compañías de telefonía móvil

Por otro lado, en los últimos años el uso de videojuegos como soporte de procesos educativos ha ido incrementando paulatinamente su popularidad. El objetivo de estos videojuegos es aumentar la motivación del alumno, pieza clave para el aprendizaje autónomo. Los educadores han ido descubriendo los beneficios derivados de su uso en términos de motivación y nuevas capacidades de educación. Por ejemplo, el videojuego *Nicoland* es un juego destinado para estudiantes de 6 a 12 años que permite reforzar entre otros los conceptos sobre ortografía o matemáticas y donde el estudiante se siente identificado con el protagonista del juego lo que fomenta su motivación.

El propósito de este Proyecto de Fin de Carrera ha sido el de aunar y explorar los beneficios derivados del uso de tecnologías móviles y de videojuegos como soporte de procesos de aprendizaje. Con este fin se ha desarrollado un videojuego para teléfonos móviles con sistema operativo *Android* que trata de facilitar el aprendizaje y la práctica de la lengua inglesa.

1.2 Descripción del proyecto

La realización de este proyecto surge gracias a las capacidades que nos ofrecen los nuevos teléfonos móviles y *smartphones.* Las propiedades de estos dispositivos brindan al usuario la posibilidad de utilizarlos en cualquier parte y en cualquier momento, es por esto que surgen las aplicaciones de *m-learning.* El desarrollo de un videojuego, orientado al aprendizaje de idiomas, busca el objetivo de aumentar la motivación del alumno y convertir el uso de las aplicaciones de *m-learning* en entretenido y dinámico.

El objetivo de este proyecto fin de carrera es el desarrollo de un videojuego, "*Maz-E-nglish*", que consiste en una **aplicación *m-learning*** cuyo objetivo es la enseñanza de la lengua inglesa. El videojuego ha sido desarrollado para teléfonos móviles con sistema operativo ***Android*,** utilizando el motor de juego ***AndEngine***, cuya distribución es libre y de código abierto.

El videojuego consiste en ir superando una serie de **laberintos** y **desafíos** que se le presentan al jugador. Para superar estos laberintos el jugador deberá ir recogiendo las monedas que encuentre por los diferentes laberintos, pero para ello necesitará superar una serie de desafíos. Estos desafíos consisten en ejercicios donde se evaluarán los conocimientos que el usuario posee sobre vocabulario, gramática y compresión oral, todo ello sobre la lengua inglesa. El jugador puede consultar los resultados y objetivos que ha cumplido y contrastarlos con los de otros jugadores, de esta manera se fomenta un mayor interés por parte de los jugadores. Los diferentes laberintos que dispone el videojuego podrán ser **temáticos**, es decir, el diseñador de los laberintos puede otorgar una temática como tiempos verbales, uso del futuro, uso del pasado, a cada uno de los escenarios que se encuentran en el videojuego.

Esta aplicación también dispone de una funcionalidad que permite al usuario repasar la teoría que se encuentra almacenada en el momento que el desee, consiguiendo así reforzar la parte teórica del idioma.

Como complemento a este videojuego también se ha desarrollado una **aplicación Java**, que permite de una manera sencilla y visual la **creación de nuevos laberintos**. Esta aplicación permite a personas con un conocimiento tecnológico bajo, crear los escenarios que posteriormente se podrán aplicar en el juego a través de una conexión a internet, mediante la descarga de los mismos.

Este proyecto está relacionado con otros proyectos diferentes que dirige mi tutor, *Telmo Zarraonandia,* los cuales se ocupan del contenido didáctico que se utiliza en esta aplicación.

1.3 Objetivos

A continuación se presenta los principales objetivos que se han perseguido con la elaboración de este proyecto final de carrera.

El primer objetivo de este proyecto ha sido por una parte el estudio de las posibilidades ofrecidas por el llamado m-*learning,* aprendizaje soportado por dispositivos móviles*,* y por otro el estudio de la plataforma *Android* y de los diferentes motores de juego compatibles con esta plataforma y disponibles en el mercado, que pudiesen servir para la elaboración de una aplicación destinada a la educación de las características antes mencionadas.

El objetivo fundamental del este proyecto se corresponde con la realización de un videojuego para la enseñanza de la lengua inglesa que sea soportado mediante tecnología móvil. La idea fundamental es que el videojuego sirviese como una herramienta para aumentar la motivación del alumno a la hora de realizar ejercicios que permitiesen la práctica de las distintas habilidades necesarias para el aprendizaje del inglés. Dentro de este objetivo se pueden definir una serie de distintos sub-objetivos relacionados bien que pueden ser clasificados según afecten al propio diseño del videojuego en si o a las características que el desarrollo debe satisfacer.

A continuación se detallan los objetivos relacionados con el videojuego en sí:

- La aplicación desarrollada tendrá un **fin lúdico** y un **fin didáctico** que perseguirá reforzar los conocimientos sobre gramática, ortografía y compresión oral de la lengua inglesa. Lo que se pretende conseguir es que los usuarios de la aplicación se diviertan jugando, consiguiendo así mejorar su nivel de inglés de una manera que no les suponga un gran esfuerzo. Se buscará la superación del usuario mostrándole las estadísticas conseguidas, para que pueda compararlas con las de otros jugadores y aumentar así su afán de superación.

- Uno de los objetivos más ambiciosos que se plantearon al crear esta aplicación, era que este sistema también dispusiese de un apartado únicamente teórico, donde el usuario pudiese consultar y repasar la teoría que se encuentre disponible. Con esto se consigue que la aplicación no sea una simple herramienta de evaluación, sino también de **formación y adquisición de nuevos conceptos**.

- Otro de los objetivos del videojuego es, ofrecer al usuario pequeños **ejercicios mentales** que deberá superar para desbloquear nuevos

caminos, así como **fortalecer el uso de la memoria** del jugador, ya que deberá recordar el camino que debe seguir hasta llegar a la meta.

La segunda categoría está relacionada con los diferentes usuarios que pueden trabajar con la aplicación, así como la reusabilidad de la aplicación para futuras mejoras y ampliaciones.

- Otro de los objetivo del proyecto es la creación de un **aplicación** que permita dar soporte a los profesores o autores en la creación de escenarios para el videojuego. Esta aplicación consistirá en una interfaz donde el usuario por medio de una serie de desplegables podrá diseñar el laberinto de una manera fácil e intuitiva. Una vez diseñado el laberinto el usuario podrá exportarlo al fichero *.xml* con su estructura correspondiente para poder ser ejecutado en el videojuego.

- Un objetivo importante es conseguir realizar un videojuego que sea **fácilmente actualizable**, donde un público con conocimientos tecnológicos bajos, pueda actualizar el contenido didáctico o los escenarios del videojuego únicamente disponiendo de una conexión a internet.

- El último objetivo que se detalla y no por ello menos importante es conseguir un videojuego que sea **reutilizable,** a la vez que rehusar contenidos de otros proyectos relacionados**.** Un aspecto importante dentro de las aplicaciones de *m-learning* basadas en objetos de aprendizaje es su reutilización. La reutilización de los conceptos es una pieza básica para la creación de herramientas destinadas a la formación de los estudiantes. Por este motivo se han utilizado los recursos de aprendizaje en un formato *xml*, fácilmente portable, reutilizable y con gran interoperabilidad. Esta cualidad nos ofrece la posibilidad de transformar muy fácilmente esta aplicación *m-learning* destinada a la lengua inglesa, a convertirla en una herramienta para la enseñanza de cualquier idioma.

1.4 Estructura de la memoria

En este apartado se ofrece un breve resumen del contenido de cada uno de los capítulos y anexos que contiene este documento.

En el capítulo I, "Introducción" se intenta dar una visión global de los mercados en la telefonía móvil. En este capítulo también se realiza una descripción general del proyecto y se mencionan cuales son los principales objetivos del proyecto. Por último se define la estructura de este documento.

Durante el capítulo II se presenta el estado del arte, así como un estudio sobre el *m-learning* y las plataformas utilizadas.

El capítulo III presenta las tecnologías utilizadas en especial el motor de juego utilizado, *AndEngine*, y las herramientas que se han utilizado para el desarrollo de este proyecto fin de carrera.

Los capítulos IV, V, VI y VII comprenden el análisis, diseño, implementación y pruebas de las aplicaciones desarrolladas. En ellos podrá encontrar un estudio detallado sobre los casos de uso, requisitos, arquitectura utilizada, y detalles de implementación y pruebas.

El capítulo VIII trata lo concerniente a la gestión del proyecto, incluyendo la planificación y los costes del mismo.

El último capítulo, capítulo IX, presenta las conclusiones y líneas futuras extraídas por el autor de este documento, sobre el proyecto realizado.

Al final de este documento también podrá encontrar una serie de anexos que incluyen:

- Ejemplo de un parseador *xml.*
- Ejemplo de un fichero de interfaz en *xml.*
- Manual de usuario del videojuego.
- Manual de la aplicación de edición de escenarios.
- Manual del videojuego para el desarrollador para futuras actualizaciones y mejoras.

Capítulo II: Estado de la cuestión

El objetivo de este capítulo es situar la investigación dentro de un conjunto más amplio de información y situar su marco tecnológico con la pretensión de ayudar al lector a valorar el trabajo realizado y apreciar con mayor claridad las principales aportaciones y novedades del proyecto.

2.1 Los videojuegos

La historia de los videojuegos comenzó hace más de 40 años, pero fue en la década de los ochenta con él lanzamiento de videojuegos como *Galaxian*, *Pac-Man* o *SpaceWars* cuando empezó a suponer un potencial cultural y comercial. El mercado de los videojuegos explotó en esta década con el lanzamiento de los ordenadores de 8 bits, como el ZX *Spectrum o el Amstrad*, y posteriormente se disparó durante los noventa con el lanzamiento de las videoconsolas como *Sega Megadrive, GameBoy, GameGear o Nintendo* que consiguieron llevar los videojuegos a las casas de todos sus usuarios, sin necesidad de disponer de una máquina recreativa para su uso. En los últimos años los videojuegos se han ido convirtiendo en una de las industrias de entretenimiento más importantes, facturando en la actualidad por encima de otras industrias más tradicionales en el ámbito audiovisual como el cine o la música. En la actualidad en el 65% de los hogares de Estados Unidos se juega a los videojuegos. *[28].*

Inicialmente los videojuegos fueron catalogados como un entretenimiento dirigido principalmente al mundo infantil, pero poco a poco y debido a la aparición de nuevos géneros y nuevas formas de jugar se han expandido a un público mucho más general.

Hoy en día los videojuegos se pueden encontrar para diferentes plataformas ya sean videoconsolas de última generación como *Play Station 3 o Xbox 360*, videoconsolas portátiles, ordenadores o dispositivos móviles. Esto hace que haya que diferenciar entre varios mercados dentro del mundo de los videojuegos, ya que cada mercado posee un público diferente con unas necesidades y unas expectativas diferentes.

En los siguientes apartados, se va a describir cual es el estado actual de los videojuegos en el entorno de teléfonos móviles de última generación y *smartphones*, así como los proyectos que se han desarrollado con un fin educativo. *[4].*

2.1.1 Videojuegos en dispositivos móviles

La historia de los videojuegos para dispositivos móviles comenzó a finales de los años noventa. Una de las principales compañías en lanzar un dispositivo con un videojuego fue *Nokia* con su popular juego *Snake*. Pero en la actualidad este negocio ha evolucionado enormemente y el mercado de los videojuegos para teléfonos móviles ha sabido adaptarse a las nuevas funcionalidades que ofrecen los dispositivos y satisfaciendo las necesidades de los actuales usuarios.

Los videojuegos que se realizan para dispositivos móviles ocupan un mercado distinto que los destinados a las videoconsolas. Por ello es por lo que hay que diferenciar a la hora de realizar un videojuego sobre que plataforma se va a utilizar, ya que los usuarios y las formas de jugar serán diferentes. Algunas de las

características principales que hay que tener en cuenta a la hora de desarrollar un videojuego para un dispositivo móvil son:

- El jugador puede disponer del videojuego en cualquier momento y en cualquier parte. Esto puede suponer que el usuario sufra de un mayor déficit de atención al estar involucrado en otras tareas y puede, por tanto, requerir implementar un menor nivel de dificultad. Igualmente parar o reanudar el videojuego no debe suponer un problema para el jugador.

- Las prestaciones que ofrece un teléfono móvil son muy inferiores a las prestaciones de una videoconsola moderna. Menos exigencias gráficas por parte del jugador.

- La conexión a internet bien mediante WiFi o 3Ge, potencia los videojuegos *on-line*, donde el usuario puede competir con otros jugadores a través de la red.

- La usabilidad de los teléfonos móviles es baja y muy diferente a las videoconsolas tradicionales. Los videojuegos para los teléfonos móviles no deben por tanto adaptar los mecanismos heredados de las consolas sino crear nuevas formas de jugar utilizando los nuevos recursos que nos ofrecen los dispositivos móviles como son los acelerómetros, sensores, etc.

- Evolución muy rápida del *hardware,* lo que impide desarrollar proyectos de muy larga duración.

Actualmente la gama de videojuegos que se pueden encontrar en el mercado es muy amplia tratando géneros como: Acción, aventuras, cartas, conducción, cooperación, deportivos, disparos, educativos, estrategia, lucha, plataformas, puzle… lo que permite al usuario elegir dentro del género deseado.

2.1.2 Videojuegos en la educación

Diferentes estudios, [5, 9] sugieren que los videojuegos pueden ser una herramienta efectiva a la hora de aumentar la motivación de los usuarios para el aprendizaje de diversas materias como las matemáticas, las ciencias, idiomas… Incluso pueden ser utilizados como entrenamiento eficaz en programas de tipo visio-motor, desarrollo del pensamiento reflexivo, reducir el número de errores de razonamiento, conseguir un mayor control de los tiempos de reacción y servir de entrenamiento ante situaciones vitales que pueden ser simuladas.

El experto, Félix Etxeberría catedrático de pedagogía de la universidad del País Vasco de San Sebastián, diferencia dos efectos distintos como consecuencia de los videojuegos [*5*]: Efectos de carácter negativo, como la violencia y el sexismo y efectos de carácter positivo que pueden ser interesantes de cara al aprendizaje, como por ejemplo:

- Carácter lúdico y de entretenimiento del alumno durante el estudio y el aprendizaje. El estudiante se siente más cómodo y más predispuesto en estos entornos.

- Estimulación visual y auditiva, mediante fotos, video, texto, juegos, etc. Lo que permite un mayor grado de interés y menor déficit de atención del alumno.

- Identificación con el personaje del juego, lo que fomenta que el alumno desee volver a jugar.

- Reconocimiento de los logros adquiridos mediante puntuaciones o niveles superados.

- Plan adaptado al ritmo de cada alumno. La dificultad ofrecida al alumno va creciendo según van ampliando los conocimientos adquiridos.

Tal y como dice McFarlane:

"L*os videojuegos facilitan la adquisición y el desarrollo de ciertas estrategias fundamentales para el aprendizaje: la resolución de problemas, el aprendizaje de secuencias, el razonamiento deductivo y la memorización. También, simplifican la realización de trabajos engrupo de tipo cooperativo o en colaboración y el aprendizaje basado en la resolución de tareas*". McFarlane (2002) *[5].*

2.1.2.1 Videojuegos desarrollados

En España existen varios videojuegos que están destinados a la educación, a continuación se muestra un análisis de tres videojuegos que se encuentran actualmente en el mercado y que abarcan diferentes públicos y materias.

2.1.2.1.1 Nicoland

Nicoland es un videojuego especialmente diseñado para los niños de 6 a 12 años. Los jugadores a través de diferentes aventuras podrán mejorar sus conocimientos en ortografía, cálculo, ciencias, historia y geografía. Este videojuego está disponible únicamente de forma *on-line*. [*6*]

Los usuarios podrán participar en aquellas aventuras que deseen o que necesiten mejorar según la materia elegida, lo que permite al jugador reforzar los conocimientos más débiles.

Este tipo de videojuegos favorece la participación del niño a la hora del estudio, ya que se identifica con los protagonistas del juego, y supone que las tareas de aprendizaje sean menos monótonas y más entretenidas. *[3]*

Ilustración 2: Videojuego Nicoland

2.1.2.1.2 Brain Training

Brain Training es uno de los juegos más populares que fomentan la capacidad mental de las personas. Este videojuego está formado por una serie de ejercicios de diversa dinámica con el fin de ejercitar la mente. Las diferentes temáticas que se pueden encontrar en el juego son varían desde ejercicios de cálculo y retención de datos hasta ejercicios de lectura y ortografía, pasando por muchos otros aspectos.

Este videojuego no está dedicado en exclusiva a niños, como es *Nicoland*, sino que se intenta dirigir a un público mucho mayor con la etiqueta de ser un videojuego que potencia el uso de la mente y refuerza la memoria. Lo que puede ser enormemente beneficioso en personas, por ejemplo de la tercera edad.

El videojuego está organizado para que el usuario deba ir superando una serie de ejercicios para poder realizar desafíos nuevos, no obstante, poco a poco se van desbloqueando nuevos ejercicios de forma paulatina según se va desarrollando la partida, para que al usuario no le resulte tan monótono.

También ofrece la posibilidad de que el usuario consulte una serie de gráficos donde es posible consultar los datos de los resultados de los distintos ejercicios y de los exámenes realizados, pudiendo el jugador apreciar así su evolución en cada uno de los distintos ejercicios y exámenes.

2.1.2.1.3 Second Life

Second Life se diferencia de los dos videojuegos anteriores en que no se considera un videojuego destinado a la educación sino más una plataforma que facilita la educación *e-learning*. Se ha decido añadirla en este párrafo ya que se considera la mayor plataforma para la educación que simula un entorno real. Utilizada en diferentes ambientes educativos ya sean universitarios o de primaria, Second Life permite realizar clases a distancia donde los usuarios pueden participar activamente.

Second Life es un entorno virtual donde cualquier persona puede participar creando su propio personaje, e interaccionar con otros usuarios situados en cualquier parte del mundo. Esta capacidad y la posibilidad de crear escenarios para la docencia a distancia hacen de *Second Life* una herramienta muy poderosa para el *e-learning*. Sus principales características que le diferencian de otros videojuegos parecidos y que le convierten en una poderosa herramienta para la educación son:

- Ofrece la Posibilidad de incrustar contenido de diferentes formatos.(texto, video, presentaciones)

- Exportar los contenidos audiovisuales a otras plataformas: repositorios, wiki, Televisión virtual, blogs…

Ilustración 3: Videojuego Second Life

2.1.2.2 Problemas asociados al desarrollo

El principal problema que existe para desarrollar videojuegos para la educación es el elevado coste de desarrollo. Los videojuegos actuales son el resultado de proyectos de gran tamaño y duración lo que produce que el coste de los mismos sea muy elevado y cuya rentabilidad sea difícil de conseguir si no consigue un gran número de ventas.

Desarrollar un videojuego desde cero, es decir sin un motor de juego ya desarrollado, puede resultar muy costoso y complejo, por ello se utilizan dichas herramientas para la creación de videojuegos como *Unreal Engine* o *Unity 3D.* En los capítulos siguientes se describirán los motores de juego gratuitos que facilitan la creación del mismo pero la capacidad y tecnología que ofrecen dichos motores se aleja mucho de los motores de juego de pago, con lo que la calidad final del videojuego desarrollado estará por debajo de la del mercado.

Otro problema es dotar al videojuego de unos gráficos potentes donde el movimiento de los personajes, la interacción entre ellos o los sonidos del videojuego sean realistas y que se asemejen lo máximo posible a la realidad algo realmente difícil de conseguir y que requiere un gran esfuerzo y trabajo.

2.2 M-learning

El continuo desarrollo de las *TIC´s* está produciendo un constante impacto en la sociedad y en las diferentes formas de educación. Las nuevas tecnologías están revolucionando los procesos de enseñanza-aprendizaje, con el objetivo de formar estudiantes autocríticos y autodidactas capaces de llevar a cabo un proceso de aprendizaje usando un dispositivo móvil en cualquier momento y en cualquier parte del mundo.

Los viejos procesos estáticos de aprendizaje han sido sustituidos por otros donde los estudiantes necesitan una mayor diversidad de espacios y recursos. Los estudiantes solicitan una completa disponibilidad y contar con una gran movilidad (*Álvarez y Edwards, 2006*), y los teléfonos móviles cumplen con creces están expectativas.

Hoy en día algunas instituciones educativas se han visto en la necesidad de producir contenidos específicamente dirigidos a los dispositivos móviles, debido a que se trata de un mercado con millones de usuarios y en constante crecimiento.

Se considera *m-learning* el aprendizaje a través de dispositivos móviles de comunicación. Este concepto se fundamenta en tres elementos principales:

- El dispositivo móvil.
- La infraestructura de comunicación.
- El modelo de aprendizaje.

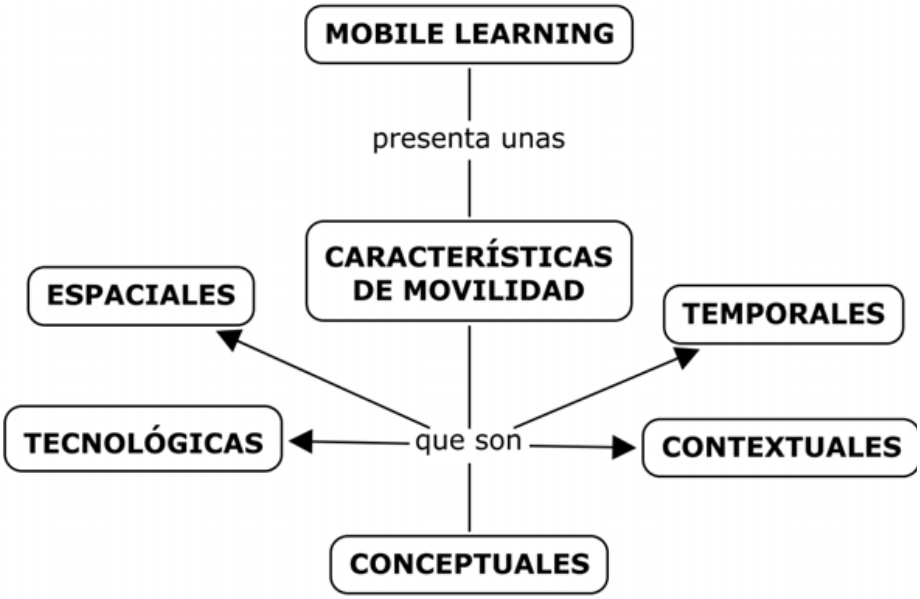


Ilustración 4: Características de movilidad del m-learning

Ilustración obtenida de *[10].*

Las aplicaciones educativas de *m-learning* están formadas por objetos de aprendizaje (OA). Un objeto de aprendizaje se considera cualquier entidad digital

que puede ser usado y referenciado durante el aprendizaje haciendo uso de las tecnologías. Los objetos de aprendizaje pueden encontrarse en cualquier formato ya sean imágenes, audio, video, texto… únicamente deben tener un contenido didáctico. Este concepto de OA no es específico del m-learning ya que también se utiliza en otros entornos educativos como e-learning.

Para la elaboración de un OA se deben considerar varios aspectos tecnológicos y pedagógicos que deberán cooperar entre sí.

Ilustración 5: Elementos para el diseño de OA

Ilustración obtenida de *[9].*

En la figura anterior se puede ver como el objeto de aprendizaje está representado en el centro de la figura y posee dos capas más. Una primera capa con cinco niveles, que constituyen aspectos tecnológicos y una capa más exterior formada por aspectos pedagógicos.

Dentro de los aspectos tecnológicos consideramos los siguientes términos:

- *Reusabilidad*: El objeto debe estar diseñado para su reutilización, con el objetivo de que pueda servir como componente a diferentes recursos de *m-learning*.

- *Accesibilidad*: El objeto debe tener la capacidad de indexarse fácilmente para una localización y recuperación más eficiente utilizando estándares de metadatos.

- *Interoperabilidad*: El objeto debe ser capaz de operar en diferentes plataformas de *hardware* y *software*.

- *Portabilidad*: Debe tener la capacidad de poder ser alojado en cualquier plataforma sin sufrir cambios.

- *Durabilidad*: No deben sufrir modificaciones ante actualizaciones de software.

Si nos referimos a los aspectos pedagógicos que se aprecian en el gráfico, deben cumplir con las siguientes características:

- *Objetivos*: Expresan de manera sencilla y clara el contenido del objeto de aprendizaje.

- *Contenidos*: EL contenido didáctico puede estar almacenado en multitud de formatos como definiciones, explicaciones, artículos, videos, entrevistas, lecturas, opiniones, incluyendo enlaces a otros objetos, fuentes, referencias, etcétera.

- *Actividades de aprendizaje*: Se consideran aquellas actividades que ayudan a los estudiantes a cumplir sus metas y objetivos.

- *Elementos de contextualización*: Permiten reutilizar el objeto en otros escenarios mediante la simulación de los mismos.

- *Evaluación*: Instrumento a través del cual se pretende verificar el aprendizaje logrado por el estudiante.*[9]*

2.2.1 Ejemplos de aplicaciones m-learning

Hoy en día existen multitud de aplicaciones y sistemas destinados al *m-learning* a continuación se presentan algunos de ellos:

- *Biology and Physical Science Nursing School:* Aplicación para dispositivos con sistema operativo iOS, que permite a sus usuarios formarse para el examen de enfermería y donde puede realizar exámenes, test y comprobar los resultados obtenidos.

- *Matemáticas para primaria:* Aplicación destinada a estudiantes de primaria que permite aprender matemáticas jugando, consiguiendo así que el usuario aprenda de una manera divertida.

- *Bon dèpart: Beginners´ French:* Aplicación destinada a la enseñanza de la lengua francesa que permite aprender los conceptos básicos. La aplicación dispone de audios con conversaciones para ayudar a la comprensión oral y ayuda para la pronunciación.

2.3 Android

Android es el sistema operativo creado por *Google* para dispositivos móviles, lanzado al mercado en octubre de 2008 y basado en una versión modificada del *kernel* de *Linux* 2.6, el cual es el encargado de por ejemplo gestionar la seguridad, la memoria o los procesos. Actualmente este sistema operativo se puede encontrar en teléfonos móviles, *Tablets*, *Netbooks* y *PDAs*. Más tarde este proyecto empezó a ser desarrollado por la *Open Handset Alliance [11]*.

Es una plataforma de código abierto distribuida bajo la licencia *Apache 2.0* por lo que su distribución es libre y posibilita el acceso y modificación de su código fuente, esto permite que multitud de desarrolladores independientes tengan la posibilidad de crear y desarrollar sus propios productos de manera gratuita lo que potencia la creación de software para esta plataforma.

2.3.1 Historia

En Octubre de 2003 se fundó *Android Inc*. en Palo Alto, California, sus creadores fueron *Andy Rubin, Rich Miner, Nick Sears* y *Chris White*. Casi dos años más tarde en Agosto de 2005 la compañía fue adquirida por *Google*.

El 5 de noviembre de 2007 se creó la *Open Handset Alliance*, un consorcio de varias compañías entre las que destacan *Broadcom Corporation, Nvidia, Samsung Electronics, Intel, LG, Motorola, y T-Mobile* entre otras; Su propósito era la creación de estándares abiertos para dispositivos móviles.

El 23 de Septiembre de 2008 sale la primera versión de *Android*, *Android 1.0* en el teléfono móvil *HTC Dream*. Las primeras novedades más importantes fueron la integración con los servicios de *Google*, un navegador para mostrar al completo y con zoom las páginas web y un mercado de aplicaciones para *Android*. Donde todos los desarrolladores podían subir sus aplicaciones y ponerlas a la venta por un bajo precio.

Ilustración 6: Logo de Android 1.0

Un año y medio más tarde el 30 de Abril de 2009 sale la versión de *Android 1.5*, también conocida como *CupCake* con mejoras en la cámara, en el servicio del GPS y pantalla táctil, así como integración con Youtube y Picassa.

Ilustración 7: Logo de Android 1.5

El 15 de Septiembre de 2009 sale la versión de *Android 1.6*, conocida como *Donut*, y el 26 de Octubre de 2009 da el salto a su versión *Android 2.0, Eclair*. Como novedades incluía velocidad de hardware optimizada, soporte para más tamaños de pantalla y resoluciones, interfaz de usuario renovada, nuevas listas de contactos y zoom digital entre otros.

Ilustración 8: Logo de Android 2.0

Las versiones posteriores a la 2.0 fueron la versión 2.2 (*Froyo*), que se lanzó el 20 de mayo de 2010. Y la 2.3 (*Gingerbread*) lanzada el 6 de diciembre de 2010. Pero la versión con mayor capacidad la encontramos en *Android 3.0, Honeycomb*, especialmente diseñada para *Tablets*, entre sus características más importantes destacan diseño más llamativo con escritorio en 3D, sistema multitarea perfeccionado y mejor soporte para redes WI-FI.

Ilustración 9: Logo de Android 3.0

2.3.2 Arquitectura Android

Android es un sistema operativo diseñado por capas. Utiliza el *kernel* de *Linux 2.6* que le proporciona el acceso a la parte *hardware* de los dispositivos. A continuación se muestra un diagrama donde se puede observar la arquitectura que desarrolla, posteriormente en los siguientes apartados se detallará y explicará en qué consiste cada una de las capas.

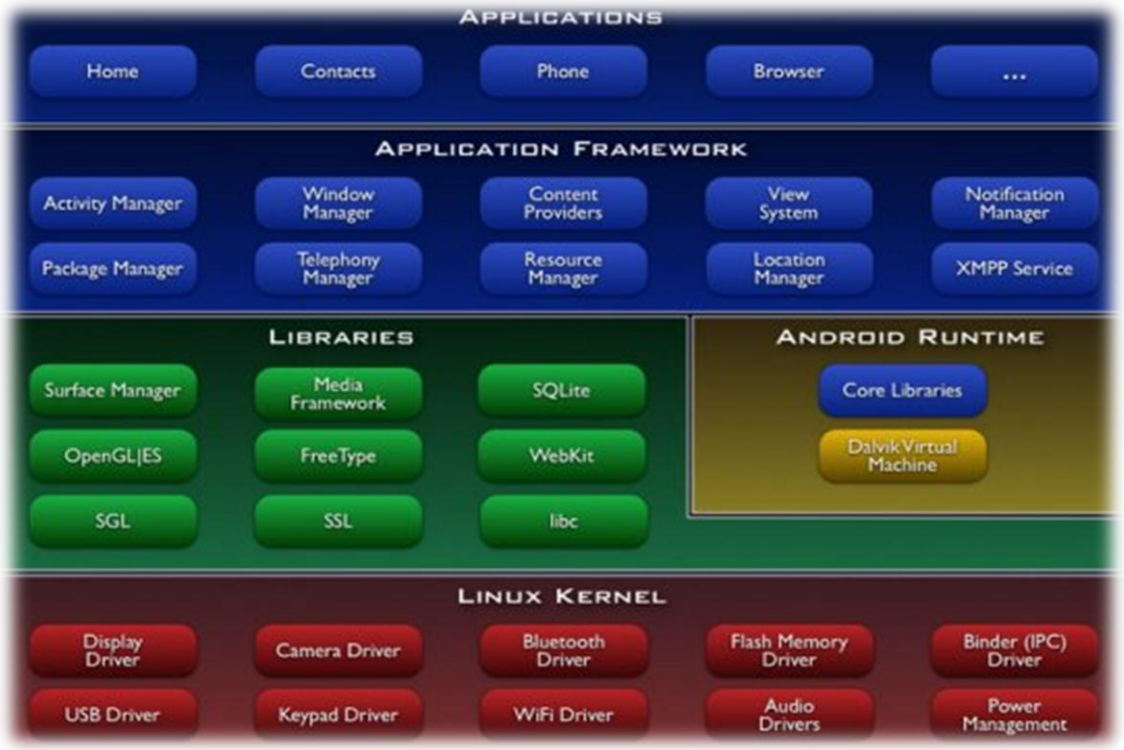


Ilustración 10: Arquitectura Android

Ilustración obtenida de *[15].*

A continuación, se detallan cada una de las capas que constituyen el sistema operativo *Android*:

- *Aplicaciones*: Forman la capa superior, esta capa la controla plenamente el usuario. Está formada por las aplicaciones que vienen instaladas de base y por aquellas aplicaciones que el usuario desee instalare. Todas las aplicaciones están escritas en el lenguaje de programación *Java*.

- *Framework de aplicaciones*: El marco de aplicaciones da acceso completo a los programadores a las mismas *APIs* utilizadas por las aplicaciones básicas. La arquitectura está diseñada para facilitar el rehúso de componentes; cualquier aplicación puede publicar sus capacidades con el objetivo de que otra aplicación pueda más tarde hacer uso de ellas. Esto se permite siempre y cuando la seguridad impuesta por el *framework* no se vea comprometida. En él gráfico anterior se pueden observar las librerías más importantes de esta capa.

- *Bibliotecas*: El sistema incluye un set de librerías en *C* y *C++* que proporcionan la mayor parte de las funcionalidades y que son utilizadas por varios componentes del sistema. Estas capacidades se exponen a los desarrolladores a través del *framework* de aplicaciones de *Android. Entre las librerías más importantes destacan:*
  - *System C library*: Librería estándar de C, optimizada para dispositivos móviles.
  - *SGL*: Motor gráfico 2D *[16].*
  - *3D Libraries*: Esta librería utiliza hardware para la aceleración 3D.
  - *SQLite*: Sistema gestor de base de datos. *[13]*
  - *FreeType*: Librería para fuentes de texto. *[14]*

- *AndroidRunTime*: Constituye el entorno de ejecución, compuesto por las librerías y la máquina virtual. Cada vez que se ejecuta una aplicación crea su propia proceso con su propia instancia de la maquina virtual *Dalvik [12]*, que ha sido desarrollada para poder albergar varios procesos a la vez.

- *Kernel de Linux*: Constituye la capa más próxima al *hardware* del dispositivo. Basado en el *kernel* de *Linux* 2.6, utiliza este *kernel* para abstraerse del *hardware* de cada terminal, conteniendo los *driver* necesarios para poder comunicarse con el dispositivo. El sistema operativo es el encargado de gestionar los servicios del sistema como seguridad, gestión de memoria, de procesos, la pila de red, etcétera. a través de este módulo.*[15]*

2.3.3 Componentes

Durante este apartado se van a describir los diferentes componentes que puede tener una aplicación *Android*. Dichos componentes pueden colaborar entre ellos de cualquier manera, no siendo necesaria la implementación obligatoria de alguno de ellos para el correcto funcionamiento de la aplicación.

2.3.3.1 Activity

El componente *Activity* es considerado el componente principal de la interfaz gráfica de una aplicación *Android*. Los *Activity* suelen ir asociados con al menos un *layout* y entre ambos crean la interfaz visual de la aplicación. Para implementar una clase *Activity* debe heredar de la clase base *Activity.* Este componente lleva asociada la interfaz de usuario, representada por la clase *View* y sus derivados.
La mayoría de las aplicaciones disponen de varias pantallas esto se suele conseguir usando varias *Activity,* cuando el usuario pasa de una pantalla a otra la primera actividad queda en estado de pausa dentro de la pila. Estos aspectos serán detallados más adelante en el apartado "*Ciclo de vida de una aplicación Android*".

Los ***Intent*** *están* íntimamente relacionados con las *Activity,* se consideran un objeto mensaje que describe que quiere hacer la aplicación. Un Intent tiene dos partes básicas:

- La Acción que se desea realizar en ese momento.
- Los parámetros que se le otorgan al objeto por medio de la clase *Bundle*.

Las principales capacidades que desarrollan los objetos *Intent* son: Solicitar información de un contacto, escribir un mensaje de texto, realizar una llamada o abrir una aplicación.

Otra clase relacionada con los *Intent* son los ***Listeners***, estos objetos son utilizados para reaccionar a eventos externos (por ejemplo, tocar la pantalla, una llamada, recibir un mensaje, etcétera).

2.3.3.2 Service

Un Servicio está formado por un código que se ejecuta durante largo tiempo en segundo plano y sin necesidad de interfaz gráfica. Los servicios pueden realizar cualquier tipo de acción que requiera la aplicación como lanzar notificaciones, mostrar eventos, actualizar la base de datos.

2.3.3.3 Content provider

Las aplicaciones desarrolladas para *Android* pueden compartir información con otras aplicaciones utilizando el componente *Content Provider*. Consiste en una clase que implementa un conjunto estándar de métodos que permite a otras aplicaciones guardar y obtener la información que maneja dicho *Content Provider.* Dicho de otro modo este servicio nos permite compartir datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno o su estructura.

2.3.3.4 Broadcast reciever

Es el componente encargado en detectar llamadas o mensajes de otros servicios generados por el sistema. Dichos servicios pueden ser recibir una llamada, un e-mail, sincronización de datos, actualización disponible, etc.

Este componente a diferencia de las *Activity* no tiene una interfaz gráfica asociada. Pero pueden utilizar el *API Notification Manager* para comunicar al usuario que se ha producido una notificación en el sistema. *[29]*

2.3.4 Ciclo de vida

Como se ha comentado anteriormente cada aplicación *Android* ejecuta su propio proceso, el cual se crea en el momento de ejecutar la aplicación y permanece hasta que ya no sea requerido o el sistema reclame su memoria para otras aplicaciones.

Uno de los aspectos más importantes de *Android* es que el tiempo de vida de un proceso no está controlado por la misma aplicación que lo creó, sino que lo determina el sistema a partir de la información que dispone. Este utiliza parámetros como, que prioridad tienen para el usuario y cuanta memoria queda disponible en él. *Android* crea una jerarquía de importancia que se detalla a continuación:

1. Se considera un *proceso de primer plano* si:
    i. El proceso que se ejecuta pertenece a una actividad con la que el usuario está interactuando.
    ii. Está ejecutando un *Broadcast Receiver.*
    iii. Esta ejecutándose un servicio.

2. Los *procesos visibles* son aquellos que pueden aparecer en la pantalla del dispositivo pero no en primer plano, ya que su actividad se encuentra pausada. Estos procesos solo serán eliminados en caso de que el sistema necesite sus recursos para mantener en ejecución los procesos de primer plano.

3. Un *proceso de servicio* es aquel que contiene un servicio del sistema. Se mantendrán en ejecución a no ser que él sistema no pueda mantener a los dos anteriores.

4. Un *proceso en background* no es visible por el usuario. Es un proceso que aloja una *activity* que no es actualmente visible para el usuario (su método *onStop*() ha sido llamado). El sistema puede eliminarlos para dar memoria a cualquiera de los 3 servicios anteriores.

5. *Proceso nulo*, es un proceso que no alberga nada, lo usa *Android* como cache para cuando se crea un proceso nuevo.

Android controla las actividades como una pila, las nuevas se sitúan en lo más alto, relegando a la actividad que ocupaba la primera posición a la segunda y así consecutivamente con todas las actividades.

A continuación se puede observar el ciclo de vida de una aplicación, así como los métodos que se utilizan para cambiar el estado de los procesos *[17].*

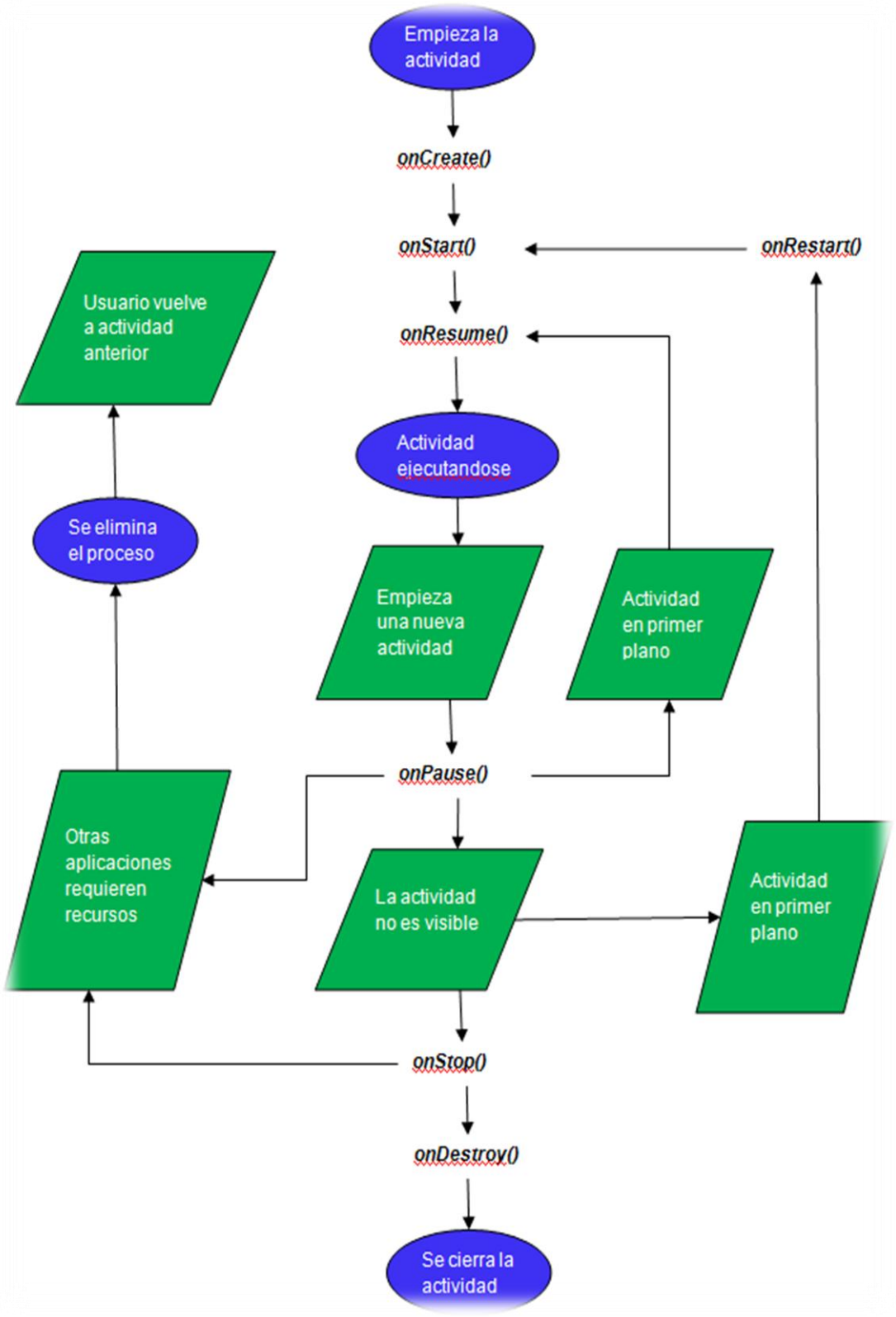


Ilustración 11: Ciclo de vida Aplicación Android

2.3.5 Motores de juego para Android

Todos los juegos que se presentan en forma gráfica necesitan contar con librerías específicas. Actualmente, las *APIs* gráficas que más se utilizan por los desarrolladores son *OpenGL [16]* y *DirectX [19]*, los cuales son los elementos encargados de comunicarse con las tarjetas gráficas. Cuando la aplicación a desarrollar es un videojuego, es necesario usar las herramientas llamadas motores de juego o *game engine*.

Estas herramientas consisten en un **conjunto de componentes reutilizables** que se desarrollan para posteriormente poder aplicarlos en los juegos o proyectos y facilitar en mayor medida la creación e implementación de nuevos videojuegos. Las principales capacidades que tiene un motor de juego son: Renderizado para gráficos 2D y/o 3D, un motor de la física o de detección de colisiones, componentes para poder dotar y controlar el sonido, componentes para la inteligencia artificial, la creación de redes, componentes para la creación de animaciones, gestión de memoria o implementar efectos gravitatorios entre otras cosas. En conclusión la funcionalidad de un motor de juegos es facilitar el diseño, la creación y la representación gráfica del juego.

Las arquitecturas utilizadas en los motores de juego permiten la portabilidad a varias plataformas. La ventaja de un motor de juego es que con un solo comando se utilizan rutinas que normalmente constarían de varias líneas de código y llamadas a la librería gráfica. A continuación se van a explicar las principales partes que componen un motor de juego.

- *Motor gráfico*: Realiza funciones de renderizado 2D/3D o de s*prite*. Se encarga de la visibilidad, el mapeo de texturas, *antialiasing* y la gestión de mallas en 3D entre otras muchas cosas.

- *Grafo de escena*: Ordena la representación lógica de una escena. Las escenas se representan en forma de árbol donde el nodo raíz es el escenario principal y los nodos hijos son los objetos que se encuentran en el escenario, que a su vez pueden tener otros nodos hijos y así sucesivamente.

- *Motor de física*: Es el encargado de simular los modelos del mundo, utilizando variables como la velocidad, gravedad, masa, fuerza, densidad, fricción, etc. Son los encargados de hacer más real las acciones que se producen en el videojuego.

- *Detector de colisiones*: Su función es la de calcular la cercanía de los objetos para avisar el momento exacto en el que pueden colisionar.

- *Motor de inteligencia artificial*: Es el encargado de dotar de cierta "inteligencia" a algunos elementos del juego. Los motores de juego más básicos carecen de este motor o su desarrollo es muy simple.

- *Motor de sonido*: Reproduce los sonidos, músicas y efectos necesarios para el desarrollo del videojuego. *[18]*

- *Componente de redes*: Componente que se encarga de sincronizar los elementos y jugadores en activo en juegos en línea.

A continuación se muestra un gráfico donde se puede observar cómo se relacionan unos componentes con otros.

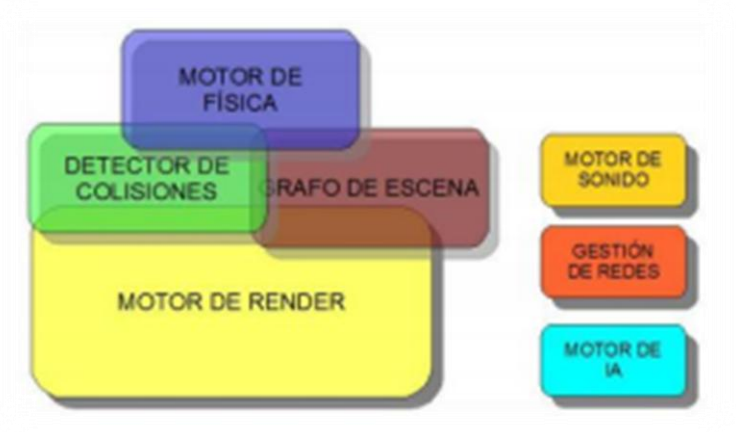


Ilustración 12: Diagrama de un motor de juego

Grafico extraído de *[18]*

Hoy en día existen varios motores de juego para *Android*, a continuación se van a detallar varios motores de juego con diferentes capacidades según las necesidades del desarrollador.

2.3.5.1 Unity3D

La herramienta que se detalla en los siguientes párrafos es una herramienta muy potente para la creación de juegos 3D de vídeo o de otros contenidos interactivos.

Unity3D es una de las herramientas más utilizadas por empresas de creación de videojuegos y animaciones ya que ofrece unas prestaciones muy elevadas frente algunas de las herramientas que se van a documentar en este proyecto. *Unity3D* es no es una herramienta gratuita, ya que para disponer de ella hay que contratar la licencia. Existen dos licencias que se adaptan al tipo de cliente que desee utilizarla. Una versión básica, y una versión pro que tiene funciones adicionales. A continuación se muestra una tabla con los precios de los diferentes productos que ofrece para programar en *Android* con *Unity3D*.

Unity3D Android	400$
Unity3D Pro Android	1900$
Unity3D Pro Android Pro	3000$

Tabla 1: Precios Unity3D Android

Este entorno de desarrollo se puede ejecutar en *Microsoft Windows y Mac OS X*, los juegos que produce se puede ejecutar en *Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone*, así como la plataforma *Android*.

Unity3D posee un editor para el desarrollo y diseño de contenidos y un motor de juego capaz de desarrollar el renderizado, juego de luces, terrenos, interiores, físicas, audio, gestión de redes así como todos los aspectos que requiera un videojuego de última generación.

Unity3D soporta la integración con *3ds Max, Maya, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Photoshop* y *Sustancias Allegorithmic* para el modelado de animaciones. Entre los motores gráficos que utiliza destacan *Direct3D (Windows), OpenGL (Mac, Windows), OpenGL ES (iPhone OS).*

*Unity3D*es una herramienta consagrada en el desarrollo de juegos y ganó el premio *Wall Street Journal* 2010, a la Innovación Tecnológica en la categoría de software *[21].*

Ilustración 13: Juego Samurai II Vengeance desarrollado Unity3D Android

2.3.5.2 Shiva3D

Shiva3D es una herramienta desarrollada por la compañía Europea *Stone trip* fundada hace 7 años en *Sophia Antipolis, Antibes Francia*.

Shiva3D consiste en otro interesante motor de juego multiplataforma que funciona en los diferentes sistemas operativos *Linux, Windows y MacOS* y que permite la compilación de juegos para *Android, iOS, Palm OS, Nintendo Wii, iPad* e incluso juegos web. Al igual que *Unity3D* permite la creación de videojuegos en 2D y 3D, y contiene herramientas para facilitar el desarrollo del videojuego.

Shiva3D usa OpenGL, para el diseño gráfico, y NVIDIA PhysX, para la física de los objetos, con el objetivo de dotar de un mayor realismo al videojuego. Esta herramienta también implementa una gran cantidad de funciones para el desarrollo de aplicaciones en red y multiusuario, además se puede encontrar documentación en el portal oficial.

Las características principales de esta herramienta son:

- *Capacidad de crear cualquier género de juego de un tiempo razonable.*
- *Fácil desarrollo para el renderizado, animación y efectos especiales*.
- Motor gráfico optimizado con iluminación dinámica, sombras y reflexión.
- Herramienta de creación unificada que compila código fuente generado por el editor para todas las plataformas soportadas.
- Alto nivel de desarrollo utilizando el lenguaje Lua..
- Capacidad de desarrollar los juegos en C+ +/ Cocoa/ Objective-C.

A continuación se muestra una tabla de precios para el desarrollo de aplicaciones A*ndroid* con *Shiva3D [22].*

Shiva Editor Basic	169 €
Shiva Editor Advanced	1499 €

Tabla 2: Precios Shiva 3D

2.3.5.3 Unreal Engine 3

El siguiente motor de juego es el primero en ser lanzado al mercado de los tres que se analizan en este documento, pero no fue hasta su tercera versión, cuando se empezó a utilizar en juegos desarrollados para *Android.*

Unreal Engine es un motor de juego de PC y consolas creados por la compañía *Epic Games*. Este motor de juego es uno de los más populares del mercado y es utilizado por empresas como *EA, Microsoft Game Studios, Sony Online o Namco*. Esta tercera entrega aparece en 2006, pero no será hasta el año 2010 cuando empiecen a aparecer los primeros videojuegos para *Android* diseñados con esta herramienta.

Actualmente es una de las herramientas más utilizadas para la creación de videojuegos para consolas y en los últimos años, se está utilizando para desarrollar videojuegos para los dispositivos móviles. *Epic Games,* la compañía que ha desarrollado *Unreal Engine,* ha sacado al mercado unas nuevas licencias gratuitas, con el objetivo de que pueda ser utilizado por todos de manera no comercial. En caso de querer sacar al mercado el producto no es necesario abonar el millón de dólares que cuesta la licencia, ya que la compañía ha propuesto nuevos modelos

comerciales: *Epic Games* después de los primeros $5,000 de ingresos obtenidos con el videojuego, se llevará un 25% de las ganancias *[24].*

A continuación se ofrecen una muestra de las principales características técnicas de *Unreal Engine 3.*

- Set de edición de desarrollo de videojuegos integrado.
- Sistema de renderizado multihilo *UnrealGemini*, con soporte para todas las técnicas modernas de renderizado y sombreado, iluminación avanzada, oclusión de ambiente, y un gestor para crear sombras complejas de forma dinámica. Iluminación global mediante *UnrealLightmass.*
- Herramienta visual de modelado físico *UnrealPhAT*, que incluye el sistema físico *NVIDIA PhysX*.
- *UnrealKismet*, para crear prototipos de juegos y mecánicas mediante scripts visuales.
- *AnimSetViewer y AnimTree Editor*, permiten a los animadores un control preciso sobre movimientos de músculos y huesos.
- *UnrealMatinee*, editor de vídeos profesional para crear vídeos, escenas de corte y cinemáticas.
- *UnrealScript*, lenguaje de programación de alto nivel orientado a objetos, totalmente integrado.
- *UnrealCascade*, sistema de efectos para implementar explosiones, niebla, humo y fuego.
- Herramienta para diseño de terrenos, configuración de vegetación, estructuras, etcétera.
- Mejora y aumento de la gestión de escenarios LOD (Level of Detail). Consiste en una técnica de programación que se utiliza para disminuir el número de polígonos de objetos lejanos. Con esta técnica se consigue acelerar el proceso de renderizado sin apenas notar pérdida de detalle *[23].*

Ilustración 14: Primer videojuego creado con Unreal Engine 3 Android

2.3.5.4 AndEngine

En los siguientes párrafos vamos a detallar otro motor de juego pero con un público muy diferente a los motores anteriores. *AndEngine*, es un motor de videojuegos 2D únicamente para la plataforma *Android*. Su principal diferencia con los otros motores es que *AndEngine* es de código abierto, se trata de una implementación 2D de *OpenGL* para *Android* por lo que utilizaremos puramente el lenguaje *Java*, a diferencia de los motores *Unity3D* y *Shiva3D* con los que podíamos trabajar con lenguajes *como C#, Python*, o incluso *Java script*.

Uno de los puntos en contra de *AndEngine* es que la documentación oficial es aún muy pobre y escasa, apenas un foro y unos ejercicios de ejemplo, es toda la información que se puede obtener de su web oficial.

Ilustración 15: Logo de AndEngine

Esta herramienta utiliza el motor de físicas *Box2D [26],* escrito en *C++* por *Erin Cattoque*, consiste en una biblioteca libre que implementa el motor de físicas en 2 dimensiones. *Box2d* también soporta *Flash, Java, C # o Python*. A continuación se nombran las características más importantes:

- Detección de colisiones con posibilidad de acción antes y después del contacto entre diferentes polígonos
- Control de las físicas en tiempo real
- Control de la fricción, densidad y elasticidad.
- Reacción a fuerzas/impulsos

Existen otros motores de juego similares también gratuitos como *Angle* o *Rokon2D* pero para la realización de la aplicación que se ha desarrollado en este proyecto fin de carrera se decidió utilizar *AndEngine* al considerarse más potente que el resto.

En el Capítulo III "*Entorno de desarrollo*" podrá encontrar más información sobre las características técnicas de *AndEngine*.

Capítulo III: Entorno de desarrollo

Durante los siguientes apartados el lector podrá encontrar una descripción de las tecnologías y herramientas que se han utilizado para el desarrollo de la aplicación de *m-learning*.

3.1 Tecnologías utilizadas

Una vez finalizado el estudio de la cuestión se procedió a realizar la selección de las tecnologías asociadas al proyecto. En la ilustración 16 se puede observar cómo se relacionan las diferentes tecnologías que se han utilizado para el desarrollo del proyecto, así como las funcionalidades que nos ofrece cada tecnología. En las siguientes secciones se describirá brevemente los motivos por los cuales se ha seleccionado cada una de ellas.

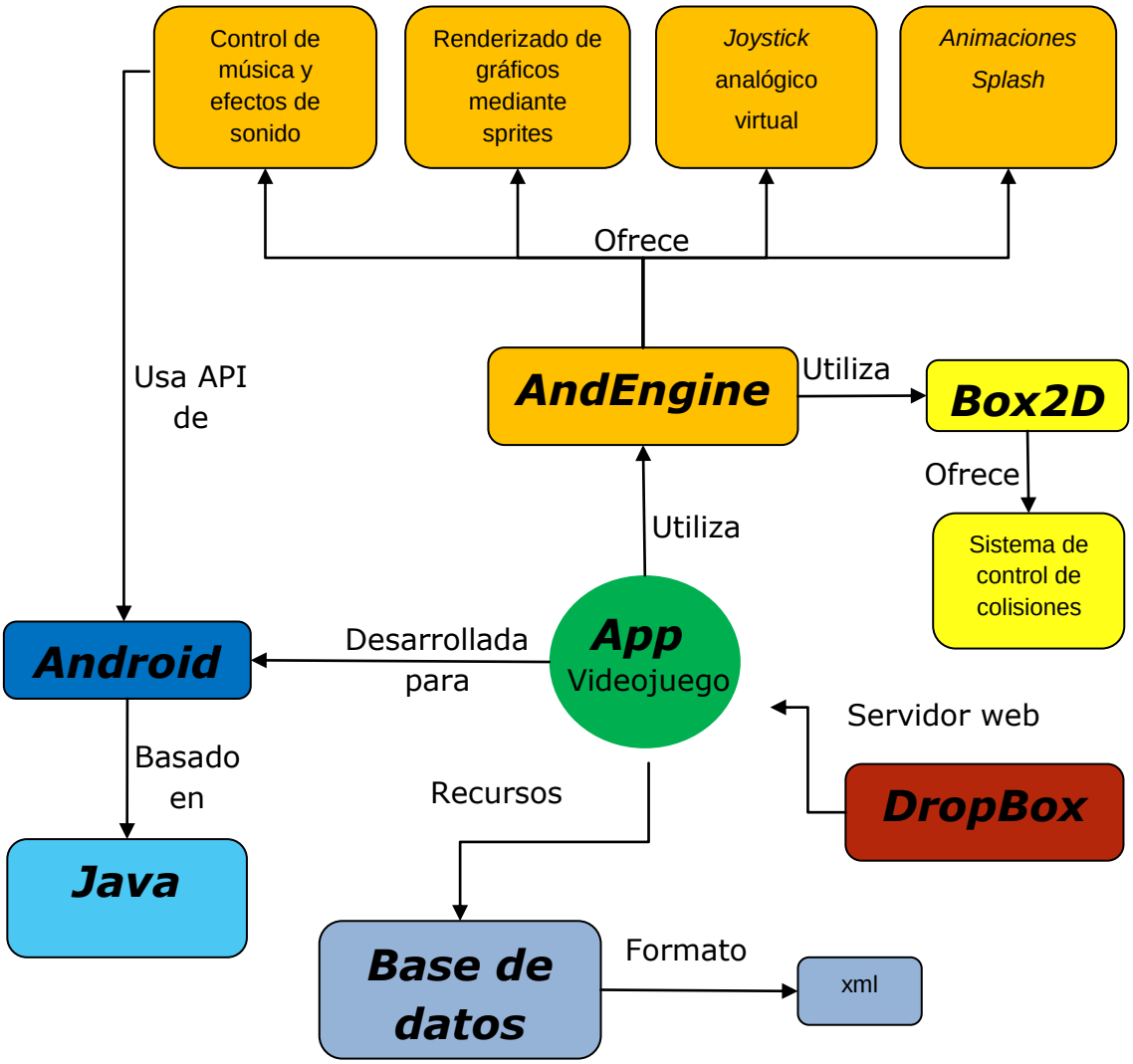


Ilustración 16: Diagrama de la tecnología utilizada

3.1.1 Sistema operativo

Para el desarrollo de la aplicación *m-learning* la principal tecnología que se ha utilizado y sobre la que se sustentan el resto de tecnologías utilizadas, es ***Android*** desarrollado en lenguaje *Java*. Se decidió utilizar el sistema operativo ofrecido por *Google* y no su más directa competencia, *iOS*, debido a que el coste de desarrollo de aplicaciones para el sistema operativo de *Google* es gratuito ya que ofrecen a coste cero el *SDK de Android,* mientras que *iOS,* no dispone de versión gratuita de su *SDK* y hace necesario disponer de un ordenador *MAC* para el desarrollo de proyectos en esta plataforma. Por otro lado los teléfonos móviles *Android* pueden instalar aplicaciones que no necesariamente provengan de la tienda oficial, mientras que los teléfonos con sistema operativo *iOS* no lo permiten y sus aplicaciones deben ser descargadas de la "*App Store"*. Por último, tal y como se muestra en la *ilustración 17* es uno de los sistema operativos más populares y que mejor se está adaptando a las nuevas necesidades de los usuarios.

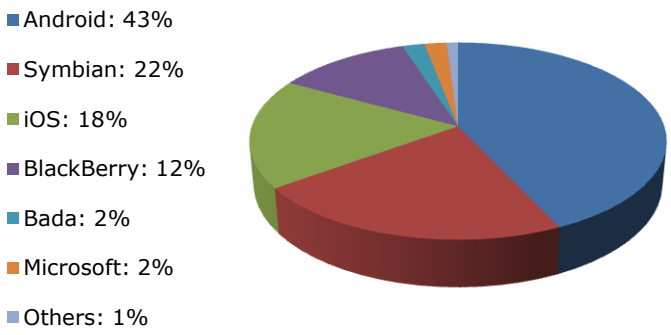


Ilustración 17: Ventas de teléfonos móviles según S.O

3.1.2 Motor de juego

Una de las tecnologías más importantes que se utilizan a la hora de desarrollar un videojuego, son los **motores de juego**. En la actualidad existen una gran multitud de motores de juego, gratuitos y de pago, que permiten la creación de videojuegos para *Android.* A continuación se nombran unos cuantos motores de juego gratuitos: *Age, Angle, Libgdx, Android-2D-Engine, CandroidEngine, Rokon, AndEngine, jPCT-AE , Dwarf-fw y Mages*. Evidentemente descartados los motores de juego de pago se intentó elegir el motor de juego gratuito que más se adaptaba a este proyecto, se realizandose una selección en función de la actividad del proyecto asociada al motor y de su madurez. A continuación se muestra una comparativa de dos de los motores de juego gratuitos más potentes [*30*].

	Rokon	**AndEngine**
FPS (un sprite)	52	58
FPS (50 físicas)	46	60
FPS (100 físicas)	42	36
Motor de físicas	Box2D	Box2D
Sistema de partículas	No	Sí
Licencia	BSD	GPL
Ejemplos	Si	Si
Juegos implementados	Tetronimo	Snake
	Drop block	AlienStars

Tabla 3: Comparativa Rokon vs AndEngine

Si observamos el rendimiento de ambos motores de juego se puede observar que *AndEngine* puede sufrir una caída de rendimiento en función de los objetos del juego algo que *Rokon* parece que es capaz de mantener constante, aunque para el objetivo del videojuego que se va a realizar, ambos serían válidos.

Ambos motores de juego utilizan el motor de físicas *box2d*, algo necesario para el desarrollo del proyecto, ya que se utilizará para la detección de colisiones entre objetos.

En conclusión, los dos motores de juego serían válidos pero me he decantado por *AndEngine*, por la cantidad de código de ejemplos, actividad en sus foros y modularidad ya que existen extensiones para multitud de funcionalidades y servicios como *multijugador*, *mutiTouch*, etcétera.

AndEngine, consiste en un motor de juego de código abierto que utiliza, el motor de físicas ***Box2D*** también de distribución libre. Este motor de juego facilita enormemente la creación de gráficos, la simulación de las físicas de los objetos, ofrece clases propias para trabajar directamente con los archivos de sonido y dispone de clases que simulan un *joystick* analógicos virtual que el desarrollador puede utilizar en cualquier momento. En el siguiente apartado podrá encontrar más información sobre este motor de juego de código abierto.

3.1.3 Otras tecnologías

Otra de las tecnologías que se han utilizado en el desarrollo de la aplicación ha sido los ficheros con formato ***xml***. Esta tecnología ha sido utilizada para almacenar el contenido de los escenarios, del contenido docente y de las partidas guardadas. Se ha utilizado esta tecnología ya que permite extraer fácilmente la información de una manera limpia y ordenada mediante *SAX*. *SAX* es un API escrito en *Java* e incluida dentro del *JRE* que nos permite crear nuestro propio *parser* de *xml*. Las ventajas que nos ofrece *xml* son:

- Separar el contenido de la presentación.
- Es extensible mediante la adición de nuevas etiquetas.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles.
- La estructura es sencilla y mejora la compatibilidad entre aplicaciones

En el ANEXO I, se describe un ejemplo del funcionamiento de un *parseador* de *xml* mediante *SAX*.

También se ha utilizado ***DropBox*** como servidor web, para albergar los ficheros que formaran parte de las actualizaciones.

3.2 Herramientas de desarrollo

A continuación se describen cuáles han sido las herramientas utilizadas para el desarrollo de la aplicación *de m-learning*, el diseño y edición de imágenes y la documentación.

- *Android Eclipse Views*: *Plugin* de *Android* que se integra con *Eclipse*. Este *plugin* nos permite depurar la aplicación. Añade una serie de vistas como *LogCat*, visor de *logs* del dispositivo que pueden ser filtrados por el desarrollador.
- *Emulador de Android*: Permite probar las aplicaciones en el ordenador así como depurarlas mediante *Eclipse* sin la necesidad de pasar la aplicación a un teléfono móvil con sistema operativo *Android*.
- *Sony Ericcson X10 mini*: Se ha utilizado este teléfono móvil el cual dispone del S.O *Android* para realizar las pruebas finales, con el objetivo de ajustar los gráficos y los textos del videojuego.

- Para el desarrollo de la aplicación de edición de escenarios, se ha utilizado la herramienta *NetBeans,* que permite la creación y compilación de código *Java,* y a su vez permite la creación de interfaces en este mismo formato.

- También se han utilizado diferentes *software* como: *IDE Eclipse 3.6, o Android SDK r12 para Linux.*

En cuanto a la elaboración de imágenes y gráficos del juego se han utilizado:

- *GIMP*: Programa de edición de imágenes digitales en forma de mapa de *bits*, tanto dibujos como fotografías. Está disponible bajo la Licencia pública general de *GNU*. Esta aplicación ha sido utilizada para la creación y diseño de imágenes.

- A pesar de no considerarlo una herramienta *software*, destacar también que se han utilizado fuentes *ttf*, para el formato de los diferentes textos que aparecen en el videojuego.

Referente a la documentación desarrollado para este proyecto se ha utilizado las siguientes herramientas:

- *Microsoft Word 2007*

- *Visual Paradigm 8.2 ComunityEdition,* para el desarrollo y creación de los diagramas del nivel detallado.

- *GanntProject:* Herramienta escrita en código *Java* para la elaboración de planificación de proyectos.

Capítulo IV: Análisis del sistema

La fase de análisis permite identificar las necesidades del producto a desarrollar mediante la obtención de los requisitos funcionales y no funcionales, ayudando al desarrollador a comprender la naturaleza del producto a construir, así como detectar las funcionalidades requeridas. El desarrollo de este capítulo comenzará con la extracción y análisis detallado de los casos de uso y concluirá con la extracción de los requisitos software, funcionales y no funcionales que se han extraído del análisis del sistema.

4.1 Explicación del videojuego

Una vez seleccionadas las tecnologías se llevaron a cabo una serie de reuniones con el tutor del proyecto a lo largo de las cuales se estableció la mecánica del juego que se implementaría con el fin de alcanzar los objetivos descritos en el primer capítulo de la memoria.

La idea principal de este proyecto es desarrollar un **videojuego *m-learning*** para la plataforma *Android.* Este videojuego deberá permitir a los usuarios la práctica de distintas habilidades relacionadas con el aprendizaje de la lengua inglesa, concretamente la gramática, vocabulario y comprensión oral. El objetivo del videojuego consistiría en recorrer varios laberintos que se le presentan al usuario. El jugador partirá de una posición inicial y superando los desafíos, tanto didácticos como lógicos, deberá llegar al final del laberinto señalizado por una bandera, siempre antes recogiendo todas las monedas que aparecen en el laberinto.

Ilustración 18: Imagen de una pantalla del Videojuego "Maz-E-nglish" desarrollado

Los desafíos estarían formados por **ejercicios de inglés**, podrán ser del tipo *Fill the gap*, *Matchin*g y *Listening*. Una vez el usuario vaya superando los laberintos se le presentaran nuevos escenarios con mayor nivel de dificultad.

Como complemento al desarrollo de este videojuego también se desarrollaría una **aplicación en Java** que permite a los profesores la **creación de nuevos escenarios**. Los cuales se componen de:

- Un laberinto, que puede estar compuesto por los siguientes objetos:
  - Muros.
  - Cajas.
  - Un protagonista.
  - Una bandera.
  - Escudos de pregunta de tipo *Fill the gap.*
  - Escudos de pregunta de tipo *Matching.*

  - Escudos de pregunta de tipo *Listening.*
  - Monedas.

- Nombres de los archivos de ejercicios asociados.

En las siguientes secciones pasamos a detallar más en profundidad el análisis del sistema llevado a cabo y las distintas funcionalidades que se proporcionará la aplicación.

4.2 Casos de uso

En esta sección se van a detallar los diferentes casos de uso que se han definido durante la fase de análisis del proyecto. Los casos de uso cubren las distintas funcionalidades que podrán realizar los distintos usuarios que interactúen con el producto.

La funcionalidad principal de la aplicación *m-learning* desarrollada es el aprendizaje de inglés mediante un videojuego, consiguiendo así el entretenimiento y la formación del usuario. En el siguiente apartado se van a definir los actores que pueden interactuar con la aplicación.

4.2.1 Actores

El actor inicialmente detectado en el videojuego es evidentemente un jugador, pero como también se ha desarrollado una aplicación *Java* para la elaboración de escenarios, también se detecta como actor del sistema el desarrollador del contenido del juego, en este caso el profesor.

Jugador: Se trata del rol más común a la hora de utilizar un videojuego. Cualquier persona puede desempeñar este rol ya que no se requiere de ninguna capacidad ni conocimiento concreto. El juego está dirigido a estudiantes del examen de *first certifícate,* con lo que se asume que los jugadores poseen un nivel alto de inglés, y no suelen ser jugadores habituales de videojuegos

Profesor: Este rol se deriva del uso de la aplicación de edición de escenarios, será el encargado de la creación de los escenarios y el contenido docente, como la teoría o los ejercicios. Este actor puede no tener conocimientos tecnológicos altos, con lo que el uso de la aplicación debe ser sencillo e intuitivo.

4.2.2 Especificación de Casos de uso

En este apartado se va a detallar el formato de las tablas utilizadas para definir los diferentes casos de uso que se extraen de las dos aplicaciones desarrolladas. Para ello se hará uso de la siguiente tabla:

Identificador	
Nombre	
Descripción	
Actor	
Precondiciones	
Pos condiciones	
Escenario	

Tabla 4: Tabla ejemplo de casos de uso

Donde cada campo significa lo siguiente:

- *Identificador***:** Código que identifica de forma unívoca cada caso de uso.
- *Nombre***:** Nombre descriptivo del caso de uso.
- *Descripción***:** Explicación clara y concisa del caso de uso que se está especificando.
- *Actor***:** tipo de usuario de la aplicación.
- *Precondiciones***:** Condiciones que se deben cumplir previamente para poder realizar una determinada operación.
- *Postcondiciones***:** Estado que presenta el sistema tras la ejecución de una determinada operación.
- *Escenario***:** Ejecución del caso de uso paso a paso con un orden determinado.

4.2.2.1 Casos de uso del jugador

En la *ilustración 19* se puede observar los casos de uso que puede realizar el actor *jugador* cuando se encuentra en la pantalla de menú principal del videojuego.

Las diferentes acciones que puede realizar el jugador son jugar al videojuego, ya sea iniciando una partida nueva o continuando una partida guardada, consultar la teoría disponible que tiene almacenada la aplicación para reforzar los conocimientos del inglés, comprobar las estadísticas que ha conseguido durante el desarrollo de una partida, actualizar el contenido didáctico y de escenarios del videojuego y abandonar la aplicación.

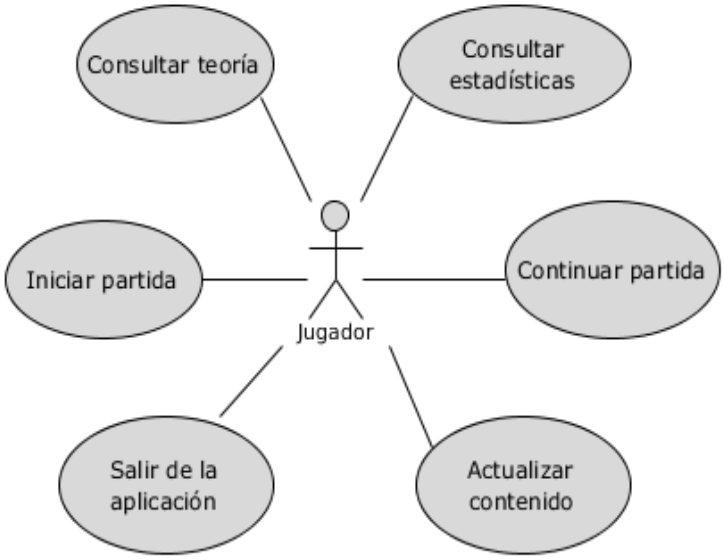


Ilustración 19: Casos de uso desde menú principal

Identificador: CU01	
Nombre	Iniciar partida.
Descripción	El jugador podrá iniciar una partida en cualquier momento, inclusive si tiene una partida guardada.
Actor	Jugador.
Precondiciones	Ninguna.
Pos condiciones	En caso de haber una partida guardada esta será sustituida por la partida nueva.
Escenario	El usuario inicia la aplicación y tras pulsar en los botones de nuevo juego introduce el nombre del personaje y selecciona un avatar

Tabla 5: CU01 - Iniciar partida

Identificador: CU02	
Nombre	Continuar partida.
Descripción	El jugador podrá continuar una partida guardada en caso de que esta exista.
Actor	Jugador.
Precondiciones	Se debe haber iniciado una partida o tener una partida guardada.
Pos condiciones	Se mostrará una pantalla con las estadísticas obtenidas.
Escenario	El usuario inicia la aplicación y tras pulsar en el botón estadísticas se mostrará una pantalla nueva con las estadísticas correspondientes

Tabla 6: CU02 - Continuar partida

Identificador: CU03	
Nombre	Consultar estadísticas.
Descripción	El jugador tendrá la posibilidad de consultar las estadísticas que ha obtenido durante el desarrollo de una partida anterior.
Actor	Jugador.
Precondiciones	Debe haber una partida guardada en el sistema.
Pos condiciones	Se cargará el laberinto con el último estado que dejó el usuario.
Escenario	El usuario inicia la aplicación y tras pulsar en el botón continuar se cargará el laberinto correspondiente con el último estado.

Tabla 7: CU03 - Consultar estadísticas

Identificador: CU04	
Nombre	Consultar teoría.
Descripción	El jugador dispondrá de un apartado de teoría desde el menú de la aplicación, donde podrá consultar las páginas de teoría que disponga el videojuego.
Actor	Jugador.
Precondiciones	Ninguna.
Pos condiciones	Ninguna.
Escenario	El usuario inicia la aplicación y tras pulsar en el botón teoría se cargará una nueva pantalla con la teoría disponible
	Ninguno

Tabla 8: CU04 - Consultar teoría

Identificador: CU05	
Nombre	Salir de la aplicación.
Descripción	El jugador podrá abandonar el juego en cualquier momento pulsando en el botón de salir.
Actor	Jugador.
Precondiciones	La aplicación debe estar iniciada
Pos condiciones	Se guardará la partida y se saldrá de la aplicación.
Escenario	El usuario inicia la aplicación y tras pulsar en el botón salir desde el menú principal o desde el menú del laberinto se cerrará la aplicación.

Tabla 9: CU05 - Salir de la aplicación

Identificador: CU06	
Nombre	Actualizar el contenido.
Descripción	El usuario podrá actualizar el contenido didáctico y escenarios del videojuego, mediante una conexión a internet.
Actor	Jugador.
Precondiciones	La aplicación debe estar iniciada y disponer de conexión a internet.
Pos condiciones	Se actualizarán los escenarios y el contenido didáctico que esté disponible en el servidor.
Escenario	El usuario inicia la aplicación y tras pulsar en el botón de actualizar comenzará la descarga de los nuevos ficheros.

Tabla 10: CU06 - Actualizar contenido

La *ilustración 20* se corresponde con las acciones que puede realizar el jugador cuando se encuentra en la pantalla del laberinto es decir, cuando está jugando al videojuego. El videojuego que se ha acordado desarrollar consiste en un laberinto que deberá ser resuelto por el jugador y para superarlo deberá realizar una serie de desafíos que estarán formados por ejercicios de inglés que intentarán agrupar una serie de habilidades como gramática, ortografía o comprensión. Se buscará combinar la práctica de las habilidades de inglés con la "diversión" en la tarea de resolver el laberinto.

Todas las acciones que puede realizar el jugador menos realizar *scroll* por la pantalla o salir de la aplicación. Después de la ilustración el lector podrá observar un análisis más detallado mediante tablas de los casos de uso extraídos.

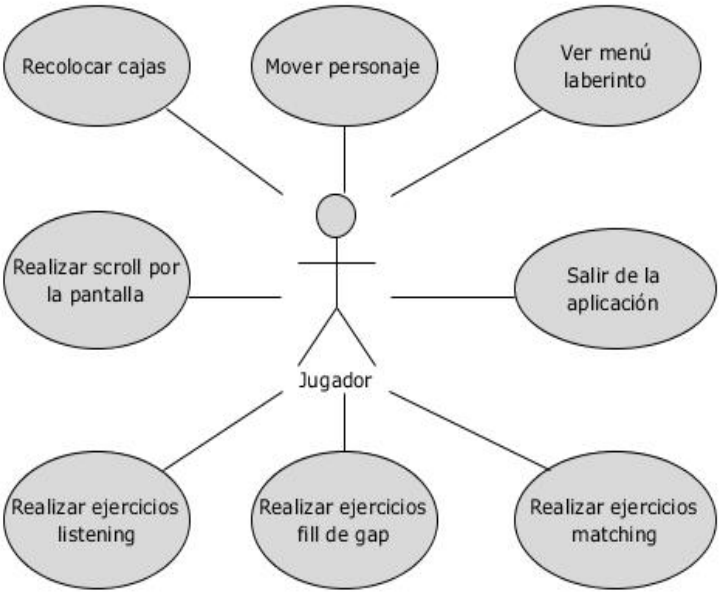


Ilustración 20: Casos de uso desde pantalla laberinto

Identificador: CU07	
Nombre	Ejercicios de *Matching*
Descripción	El jugador dispondrá de ejercicios del tipo *Matching* en los diferentes laberintos y niveles del juego.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y chocar con el escudo correspondiente que lanza los ejercicios de esta categoría.
Pos condiciones	Se mostrará una nueva pantalla con el ejercicio.
Escenario	El usuario desde el laberinto deberá chocar con el escudo correspondiente a esta categoría de ejercicios.

Tabla 11: CU07 - Ejercicios Matching

Identificador: CU08	
Nombre	Ejercicios de *Fill the gap*
Descripción	El jugador dispondrá de ejercicios del tipo *Fill the gap* en los diferentes laberintos y niveles del juego.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y chocar con el escudo correspondiente que lanza los ejercicios de esta categoría.
Pos condiciones	Se mostrará una nueva pantalla con el ejercicio.
Escenario	El usuario desde el laberinto deberá chocar con el escudo correspondiente a esta categoría de ejercicios.

Tabla 12: CU08 - Ejercicios Fill the gap

Identificador: CU09	
Nombre	Ejercicios de *Listening*
Descripción	El jugador dispondrá de ejercicios del tipo *Listening* en los diferentes laberintos y niveles del juego.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y chocar con el escudo correspondiente que lanza los ejercicios de esta categoría.
Pos condiciones	Se mostrará una nueva pantalla con el ejercicio.
Escenario	El usuario desde el laberinto deberá chocar con el escudo correspondiente a esta categoría de ejercicios.

Tabla 13: CU09 - Ejercicios Listening

Identificador: CU10	
Nombre	Ver menú laberinto
Descripción	El usuario tiene la posibilidad de pulsar el botón menú del teléfono móvil para acceder al menú del juego.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y pulsar el botón menú del dispositivo.
Pos condiciones	Se expondrá un menú con las opciones: Recolocar cajas, Menú y Salir.
Escenario	El usuario desde el laberinto deberá pulsar el botón de menú

Tabla 14: CU10 - Ver menú laberinto

Identificador: CU11	
Nombre	Mover personaje
Descripción	El usuario utilizará un *joystick* analógico que aparecerá en la pantalla del dispositivo con el cual podrá mover el personaje por la pantalla.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto.
Pos condiciones	El personaje se desplazará por la pantalla.
Escenario	El usuario desde el laberinto deberá pulsar los botones del *joystick* analógico

Tabla 15: CU11 - Mover personaje

Identificador: CU12	
Nombre	Realizar *scroll* por la pantalla
Descripción	El usuario dispondrá de la posibilidad de realizar scroll por la pantalla desde la pantalla del laberinto.
Actor	Jugador.
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto.
Pos condiciones	La cámara de la pantalla se desplazará según el movimiento del usuario.
Escenario	El usuario desde el laberinto deberá tocar la pantalla y realizar un movimiento con el dedo sobre ella en la dirección que desee mover la pantalla

Tabla 16: CU12 - Realizar scroll por la pantalla

Identificador: CU13	
Nombre	Mover cajas
Descripción	El personaje podrá mover cajas chocándose con ellas.
Actor	Jugador.
Precondiciones	El jugador debe chocarse con una caja.
Pos condiciones	La caja se desplazará en el sentido que haya sido empujada.
Escenario	El usuario desde el laberinto deberá mover el personaje hasta chocarse con una caja.

Tabla 17: CU13 - Mover cajas

4.2.2.2 Casos de uso del profesor

Por último se muestran los casos de uso que se corresponden con el uso por parte de un profesor de la aplicación para edición de escenarios. Esta aplicación permite al usuario la creación de nuevos escenarios para el videojuego así como generar el fichero correspondiente.

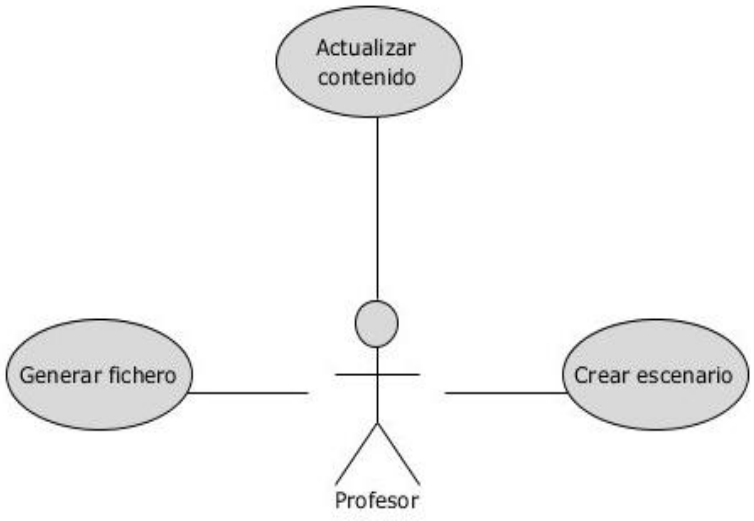


Ilustración 21: Casos de uso aplicación "Editor de escenarios"

Identificador: CU14	
Nombre	Crear laberintos
Descripción	El profesor podrá crear nuevos laberintos y asociarle los ejercicios según corresponda con la aplicación destinada a la creación de escenarios.
Actor	Profesor.
Precondiciones	Ninguno.
Pos condiciones	Se creara un nuevo laberinto asociado con sus ejercicios correspondientes.
Escenario	El profesor utilizará la aplicación destinada a la creación de laberintos, que le guiará y ayudará para la creación del fichero con los nuevos escenarios.

Tabla 18: CU14 - Crear laberintos

Identificador: CU15	
Nombre	Actualizar contenido
Descripción	El profesor podrá actualizar el contenido didáctico del videojuego.
Actor	Profesor.
Precondiciones	Debe disponer de los nuevos ficheros que desee actualizar.
Pos condiciones	La aplicación dispondrá de la actualización docente correspondiente.
Escenario	El profesor añadirá los nuevos ficheros, al directorio donde se albergan los *xml*.

Tabla 19: CU15 - Actualizar contenido

Identificador: CU16	
Nombre	Generar fichero
Descripción	El sistema permitirá la generación del fichero *xml,* con los nuevos escenarios creados.
Actor	Profesor.
Precondiciones	El profesor deber haber creado al menos un escenario
Pos condiciones	La aplicación generará el fichero correspondiente.
Escenario	El sistema creará un nuevo fichero llamado escenario.xml donde se albergaran los nuevos escenarios.

Tabla 20: CU16 - Generar fichero

4.3 Requisitos software

A continuación se describen los requisitos software del videojuego y de la aplicación desarrollada de edición de laberintos, estos requisitos se han concretado en las diferentes reuniones que se han tenido con el tutor del proyecto. Con el fin de facilitar la tarea de compresión por parte del lector, en primer lugar vamos a explicar la estructura de cada uno de los requisitos así como los campos por los que está compuesta su definición, y posteriormente mostrar la manera en la que están agrupados.

Cada requisito estará representado por una tabla de la siguiente forma:

Identificación			
Título			
Descripción			
Prioridad	☐ Alta ☐Media ☐ Baja	**Estabilidad**	☐Si☐ No
Claridad	☐ Alta ☐Media ☐ Baja	**Verificabilidad**	☐ Alta ☐ Media ☐ Baja
Necesidad	☐ Esencial ☐ Deseable ☐ Opcional		
Fuente			

Tabla 21: Tabla ejemplo requisitos

Donde cada campo significa lo siguiente:

- *Identificación***:** Código que identifica de forma unívoca cada uno de los requisitos.
- *Título***:** Título descriptivo del requisito.
- *Descripción***:** Explicación clara y concisa del requisito que se está especificando.
- *Prioridad***:** Orden de cumplimiento de un requisito. A mayor prioridad, mayor urgencia para realizarlo.

- *Estabilidad***:** Probabilidad de cambio de un requisito. A mayor estabilidad, menor posibilidades de que dicho requisito se vea modificado.

- *Claridad***:** Cataloga la no ambigüedad de un requisito. Cuanta mayor claridad tenga, menor ambigüedad contendrá.

- *Verificabilidad***:** Facilidad para comprobar que el sistema cumple un requisito.

- *Necesidad***:** Esencialidad. Interés del cliente en la realización de un requisito.

- *Fuente***:** Origen del requisito ya sea una persona o documentación.

Por otro lado, con ánimo de ofrecer una estructura que facilite la realización y comprensión de los requisitos, se van a organizar en dos categorías:

- *Requisitos software funcionales***:** especifican qué tiene que hacer el software. Definen el propósito del software.

- *Requisitos software no funcionales***:** especifican como el software debe hacer las cosas y que debe cumplir

4.3.1 Requisitos funcionales

Las siguientes tablas corresponden con los requisitos funcionales del videojuego y la aplicación de edición de laberintos. Aparecerán en apartados separados al considerarse dos aplicaciones independientes.

4.3.1.1 Requisitos funcionales del videojuego

Las siguientes tablas se corresponden con los requisitos funcionales que se han extraído del videojuego a desarrollar:

Identificación: RSF-V-01			
Título	Iniciar nueva partida.		
Descripción	El sistema deberá permitir iniciar una nueva partida siempre que el usuario lo desee. En caso de haber una partida guardada esta será sustituida		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 22: RSF-V-01 – Iniciar nueva partida

Identificación: RSF-V-02			
Título	Continuar partida guardada.		
Descripción	La aplicación deberá ofrecer la posibilidad al usuario de continuar con una partida guardada, los objetos muros, cajas, monedas, escudos, banderas y el protagonista deberán permanecer en la misma posición que se encontraban por última vez.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 23: RSF-V-02 - Continuar partida guardada

Identificación: RSF-V-03			
Título	Mostrar estadísticas.		
Descripción	La aplicación deberá disponer de un apartado donde se muestren las estadísticas que ha acumulado el jugador. Las estadísticas que deben mostrarse son: - Nivel en el que se encuentra el usuario. - Monedas conseguidas por el jugador. - Preguntas acertadas. - Preguntas falladas. - Tiempo total. - Número de veces que se han recolocado las cajas.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 24: RSF-V-03 - Mostrar estadísticas

Identificación: RSF-V-04			
Título	Guardar partida.		
Descripción	El sistema almacenará la situación de la partida actual así como sus estadísticas cada vez que el usuario salga de la aplicación. La aplicación deberá almacenar: - Posición de las cajas. - Posición de las monedas desbloqueadas. - Posición de los escudos desbloqueados. - Posición del protagonista. - Nivel en el que se encuentra el usuario. - Monedas conseguidas por el jugador. - Preguntas acertadas. - Preguntas falladas. - Tiempo total. - Número de veces que se han recolocado las cajas.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 25: RSF-V-04 - Guardar partida

Identificación: RSF-V-05			
Título	Consultar teoría.		
Descripción	La aplicación dispondrá de un apartado donde el usuario podrá consultar páginas de teoría que se encuentren disponibles en la aplicación.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 26: RSF-V-05 - Consultar teoría

Identificación: RSF-V-06			
Título	Páginas de teoría.		
Descripción	Las páginas de teoría estarán formadas por *usos*, *formulas* o ambas. Estas páginas deberán cumplir con un formato preestablecido.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 27: RSF-V-06 - Páginas de teoría

Identificación: RSF-V-07			
Título	Pintar laberinto.		
Descripción	La aplicación deberá pintar el laberinto correspondiente al nivel del jugador que extraerá del fichero escenarios.xml. Los diferentes objetos que puede pintar el laberinto son: - Cajas. - Muros. - Monedas. - Escudos de tipo 1. - Escudos de tipo 2. - Escudos de tipo 3. - El protagonista. - Fondo del laberinto.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 28: RSF-V-07 - Pintar laberinto

Identificación: RSF-V-08			
Título	Parsear *xml*.		
Descripción	El sistema deberá disponer de un *parseador xml* para extraer el contenido de los ficheros *xml* que albergan: - Los escenarios. - Los ejercicios de los diferentes tipos. - La teoría disponible. - Los datos necesarios al protagonista y al laberinto, para cargar una partida guardada.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 29: RSF-V-08 - Parseador de xml

Identificación: RSF-V-09			
Título	Mostrar ejercicios.		
Descripción	El sistema mostrará al usuario los diferentes ejercicios de inglés, que se encuentran en la aplicación cuando este, choque con uno de los escudos. Los tipos de ejercicios que deberá mostrar la aplicación son: - Ejercicios de tipo Matching. - Ejercicios de tipo Fill de gap. - Ejercicios de tipo Listening.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 30: RSF-V-09 - Mostrar ejercicios

Identificación: RSF-V-10			
Título	Comprobar respuestas.		
Descripción	La aplicación deberá obtener y comprobar la respuesta obtenida en cada uno de los ejercicios que realice el usuario y posteriormente deberá eliminar el escudo o mantener su posición según el acierto o no del jugador.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 31: RSF-V-10 - Comprobar respuestas

Identificación: RSF-V-11			
Título	Eliminar escudo.		
Descripción	El sistema deberá eliminar un escudo de la pantalla y de las físicas del juego cuando el jugador acierte el ejercicio correspondiente. También deberá almacenar cual ha sido la moneda desbloqueada.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 32: RSF-V-11 - Eliminar escudo

Identificación: RSF-V-12			
Título	Eliminar moneda.		
Descripción	El sistema deberá eliminar una moneda de la pantalla y de las físicas del juego cuando el personaje del jugador choque contra una de ellas. También deberá almacenar cual ha sido la moneda desbloqueada.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 33: RSF-V-12 - Eliminar moneda

Identificación: RSF-V-13			
Título	Salir de la aplicación.		
Descripción	La aplicación ofrecerá la capacidad al usuario de cerrar y salir de la aplicación cuando él lo desee. El sistema deberá liberar los recursos que estuviese utilizando para que el sistema operativo se los pueda asignar a otros procesos.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 34: RSF-V-13 - Salir de la aplicación

Identificación: RSF-V-14			
Título	Desplazar personaje.		
Descripción	El usuario podrá desplazar el personaje que controlará mediante un *joystick* analógico virtual que aparecerá en la pantalla táctil del dispositivo. El *joystick* analógico será implementado mediante unas clases que ofrece el motor de juego.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 35: RSF-V-14 - Desplazar personaje

Identificación: RSF-V-15			
Título	Desplazar cajas.		
Descripción	El usuario podrá desplazar las cajas que aparecen en la pantalla chocándose con ellas. Estas deberán moverse simulando un choque real entre dos objetos.		
Prioridad	☐ Alta ☑Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☐Alta ☑Media ☐Baja
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 36: RSF-V-15 - Desplazar cajas

Identificación: RSF-V-16			
Título	Menú principal.		
Descripción	La aplicación dispondrá de un menú principal desde donde el usuario podrá elegir entre las siguientes opciones: Nueva partida, continuar, estadísticas, teoría o salir.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 37: RSF-V-16 - Menú principal

Identificación: RSF-V-17			
Título	Menú laberinto.		
Descripción	La aplicación permitirá al usuario acceder a un menú desde la pantalla de laberinto, tras pulsar el botón de menú del teléfono móvil. Este menú dispondrá de las siguientes opciones: - Volver al menú principal. - Recolocar las cajas del laberinto. - Salir de la aplicación.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 38: RSF-V-17 - menú laberinto

Identificación: RSF-V-18			
Título	Recolocar cajas.		
Descripción	El menú del laberinto dispondrá de una opción para que el usuario pueda recolocar las cajas en su posición inicial. Una vez pulsado este botón el sistema se encargará de situar las cajas en las posiciones iniciales.		
Prioridad	☐ Alta ☐Media ☑ Baja	**Estabilidad**	☐Si☑ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 39: RSF-V-18 - Recolocar cajas

Identificación: RSF-V-19			
Título	Permitir *scroll.*		
Descripción	El sistema deberá implementar la funcionalidad de realizar *scroll* por la pantalla. Esta funcionalidad deberá ser ejecutada cuando el usuario toque la pantalla táctil en la dirección que desee mover la cámara.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☐Si☑ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 40: RSF-V-19 - Permitir scroll

Identificación: RSF-V-20			
Título	Mostar ejercicios *Matching.*		
Descripción	La aplicación deberá mostrar un ejercicio de tipo *Matching* cuando el personaje choque con los escudos de tipo 1, estos ejercicios estarán compuestos de: - Un enunciado que se corresponderá con un objeto de tipo *TextView.* - Tres posibles respuestas que se corresponderán con tres botones que se mostrarán en pantalla.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 41: RSF-V-14 – Mostrar ejercicios Matching

Identificación: RSF-V-21			
Título	Mostar ejercicios *Fill de gap.*		
Descripción	La aplicación deberá mostrar un ejercicio de tipo *Fill de gap* cuando el personaje choque con los escudos de tipo 2, estos ejercicios estarán compuestos de: - Un enunciado que se corresponderá con un objeto de tipo *TextView.* - Tres botones que se corresponden con las posibles respuestas del usuario. - Una línea de texto donde se podrá observar la respuesta completa, una vez pulsado un botón de respuesta. - Un botón para realizar la comprobación de la respuesta.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto		

Tabla 42: RSF-V-21 – Mostrar ejercicios Fill de gap

Identificación: RSF-V-22			
Título	Mostar ejercicios *Listening.*		
Descripción	La aplicación deberá mostrar un ejercicio de tipo *Listening* cuando el personaje choque con los escudos de tipo 3, estos ejercicios estarán compuestos de: - Un enunciado que se corresponderá con un objeto de tipo *TextView.* - Un archivo mp3 que contendrá la pronunciación de varias palabras que el jugador deberá recordar. - Varios mini-ejercicios correspondientes al contenido del audio. Compuestos de dos botones con las posibles respuestas.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 43: RSF-V-22 – Mostrar ejercicios Listening

Identificación: RSF-V-23			
Título	Reproducir música y efectos de sonido.		
Descripción	El sistema deberá utilizar las clases que ofrece el motor de juego para poder reproducir música y efectos de sonido.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 44: RSF-V-23 - Reproducir música y efectos de sonido

Identificación: RSF-V-24			
Título	Recorrer teoría.		
Descripción	Permitirá recorrer la teoría bidireccionalmente, pudiendo el usuario moverse hacia atrás y hacia delante entre las diferentes páginas de teoría.		
Prioridad	☐ Alta ☑Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 45: RSF-V-24 - Recorrer teoría

Identificación: RSF-V-25			
Título	Guardar estadísticas.		
Descripción	El sistema guardará las estadísticas de la última partida realizada cuando se cierre la aplicación. Los datos que deberá almacenar son: - Nivel en el que se encuentra el usuario. - Monedas conseguidas por el jugador. - Preguntas acertadas. - Preguntas falladas. - Tiempo total. - Número de veces que se han recolocado las cajas.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 46: RSF-V-25 - Guardar estadísticas

Identificación: RSF-V-26			
Título	Personificar jugador.		
Descripción	La aplicación permitirá personificar el personaje del juego al usuario mediante un nombre y la selección de un avatar que se mostrarán por la pantalla del dispositivo.		
Prioridad	☐ Alta ☑Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 47: RSF-V-26 - Personificar jugador

Identificación: RSF-V-27			
Título	Cambiar de nivel.		
Descripción	El videojuego dispondrá de varios niveles que aumentarán en dificultad según los vaya superando el usuario. Un nivel será superado cuando el usuario recoja todas las monedas del escenario y choque contra la bandera.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 48: RSF-V-27 - Cambiar de nivel

Identificación: RSF-V-28			
Título	Mostrar animaciones.		
Descripción	El videojuego mostrará las siguientes animaciones e imágenes en movimiento: - Animación al iniciar el videojuego. - Animación al pasar de nivel. - Animación al finalizar todos los niveles. - Imágenes en movimiento representando las monedas. - Imágenes en movimiento representado la bandera.		
Prioridad	☐ Alta ☐Media ☑ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 49: RSF-V-28 - Mostrar animaciones

Identificación: RSF-V-29			
Título	Realizar actualizaciones		
Descripción	El sistema deberá permitir al usuario realizar las actualizaciones del contenido didáctico y los escenarios. Para ello descargará el archivo *escenarios.xml* del servidor, y posteriormente se descargará los ficheros que estén asociados con ese fichero.		
Prioridad	☐ Alta ☐Media ☑ Baja	**Estabilidad**	☑Si☐ No
Claridad	☐ Alta ☑Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☐ Esencial ☑ Deseable ☐ Opcional		
Fuente	Tutor proyecto.		

Tabla 50: RSF-V-29 - Realizar actualizaciones

4.3.1.2 Requisitos funcionales de la aplicación, editor de escenarios

Las siguientes tablas se corresponden con los requisitos funcionales que se han extraído de la aplicación para la edición de escenarios:

Identificación: RSF-A-01			
Título	Seleccionar fila.		
Descripción	La aplicación permitirá al usuario seleccionar la fila que desea pintar del laberinto. Permitirá la navegabilidad entre filas en ambos sentidos, pudiendo así el usuario modificar una fila que ya había dibujado.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 51: RSF-A-01 - Seleccionar fila

Identificación: RSF-A-02			
Título	Seleccionar objeto por posición.		
Descripción	La aplicación permitirá al usuario seleccionar el objeto que desea colocar en la posición específica por número de fila y número de columna. Los diferentes objetos que puede seleccionar son: - Protagonista - Muro - Caja - Moneda - Escudo de tipo 1 - Escudo de tipo 2 - Escudo de tipo 3 - Bandera		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 52: RSF-A-02 - Seleccionar objeto por posición

Identificación: RSF-A-03			
Título	Generar fichero.		
Descripción	El sistema generará un fichero con los escenarios creados bajo el formato *xml,* el cual será compatible con el videojuego desarrollado.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 53: RSF-A-03 - Generar fichero

Identificación: RSF-A-04			
Título	Introducir nombres fichero.		
Descripción	El sistema permitirá al usuario introducir los nombres de los ficheros de ejercicios, que irán relacionados con el escenario creado. Los gráficos de los escenarios estarán controlados por la aplicación no pudiendo ser modificados por el desarrollador de los escenarios.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 54: RSF-A-04 - Introducir nombres fichero

4.3.2 Requisitos no funcionales

A continuación se mostrarán los requisitos no funcionales tanto del videojuego como de la aplicación desarrollada.

4.3.2.1 Requisitos no funcionales del videojuego

Las siguientes tablas se corresponden con los requisitos no funcionales del videojuego:

Identificación: RSNF-V-01			
Título	Compatibilidad con dispositivos.		
Descripción	El sistema deberá ser compatible con diferentes teléfonos móviles con sistema operativo *Android* 2.1		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☐Alta ☑Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 55: RSNF-V-01 - Compatibilidad con dispositivos

Identificación: RSNF-V-02			
Título	Compatibilidad con resoluciones.		
Descripción	La aplicación será compatible con diferentes resoluciones de pantalla.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☐Alta ☑Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 56: RSNF-V-02 - Compatibilidad con resoluciones

Identificación: RSNF-V-03			
Título	Compatibilidad con formatos de audio.		
Descripción	El sistema deberá reproducir distintos formatos de sonido como mp3, AAC, Ogg, MIDI y WAV.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 57: RSNF-V-03 - Reproducir sonido

Identificación: RSNF-V-04			
Título	Mostrar textos legibles.		
Descripción	El sistema se adaptará a la resolución de las pantallas y mostrará los textos legibles para el usuario.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador		

Tabla 58: RSNF04 - Mostrar textos legibles

Identificación: RSNF-V-05			
Título	Descarga de ficheros.		
Descripción	La descarga de ficheros deberá ser fiable y rápida con el objetivo de no bloquear el videojuego.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador.		

Tabla 59: RSNF05 - Descarga de ficheros

4.3.2.2 Requisitos no funcionales de la aplicación de editor de escenarios.

Las siguientes tablas se corresponden con los requisitos no funcionales de la aplicación de edición de escenarios:

Identificación: RSNF-A-01			
Título	Interfaz sencilla e intuitiva		
Descripción	La aplicación deberá ser sencilla e intuitiva, para que usuarios con bajos conocimientos tecnológicos puedan usarla sin que les suponga un esfuerzo adicional.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador		

Tabla 60: RSNF-A-01 - Interfaz sencilla e intuitiva

Identificación: RSNF-A-02			
Título	Control de errores		
Descripción	La aplicación controlará los posibles errores que se pudiesen cometer en la creación de un laberinto. No generando el fichero hasta estar el laberinto bien formado.		
Prioridad	☑ Alta ☐Media ☐ Baja	**Estabilidad**	☑Si☐ No
Claridad	☑ Alta ☐Media ☐ Baja	**Verificabilidad**	☑Alta ☐Media ☐Baja
Necesidad	☑ Esencial ☐ Deseable ☐ Opcional		
Fuente	Desarrollador		

Tabla 61: RSNF-A-02 - Control de errores

Capítulo V: Diseño conceptual

En el siguiente capítulo se le ofrece al lector una visión general del problema, describiendo el diseño del sistema a implementar, e incluyendo tanto la definición de los **componentes** que forman el *videojuego y* la *aplicación desarrollada,* como la **arquitectura del propio sistema**. En el último punto del capítulo se presenta además el diseño detallado de la aplicación.

5.1 Visión general

A continuación se presenta una visión general del problema y el contexto donde se desarrolla el videojuego y la aplicación de edición de escenarios desarrollados. En líneas generales, el funcionamiento del sistema es el siguiente:

Principalmente existirán dos roles, el profesor y el jugador. Las diferentes tareas que puede realizar el profesor son:

- Un profesor podrá crear nuevos escenarios utilizando la aplicación "*Edición de escenarios*", donde podrá asociar los escenarios con los ejercicios que el desee.

- Mediante una aplicación que se está desarrollando en otro Proyecto Fin de Carrera podrá crear las páginas de teoría y de ejercicios que se utilizarán en el videojuego.

- Una vez creado el contenido didáctico y los escenarios, el profesor podrá subir todo el contenido a un servidor, para que el usuario pueda descargarlo desde su teléfono móvil y actualizar así el videojuego.

El juego está dirigido a estudiantes del *first certifícate,* con lo que se asume que los jugadores poseen un nivel alto de inglés. El usuario puede realizar diferentes acciones:

- El jugador podrá jugar al videojuego, realizando así ejercicios de inglés, y consultar las páginas de teoría que dispone, aumentando así su dominio de la lengua inglesa

- El usuario del videojuego podrá actualizar el contenido del mismo, siempre y cuando haya una actualización disponible.

En la ilustración que aparece en la siguiente página *ilustración 22* se muestra de manera simplificada la interacción entre los diferentes usuarios y aplicaciones desarrolladas.

Ilustración 22: Visión general

5.2 Arquitectura de la aplicación

La arquitectura de la aplicación permite obtener un diseño a alto nivel del sistema, identificando y definiendo los módulos por los que está formado y las relaciones que existen entre ellos. Para la elaboración del siguiente diseño del videojuego me he basado en modelo de arquitectura en tres capas, a la que se ha añadido una capa intermedia entre la capa "Núcleo" y la capa de "Presentación". Las prioridades a la hora de construir el diseño han sido maximizar la cohesión entre los componentes de cada uno de los módulos y minimizar el grado de dependencia entre ellos.

Para la aplicación de diseño de escenarios no ha sido necesario realizar un diseño previo, ya que consiste en una sencilla aplicación con tan solo dos clases.

A continuación se muestra el diagrama de componentes que se ha desarrollado para el videojuego y se detallará brevemente el rol que juega cada uno.

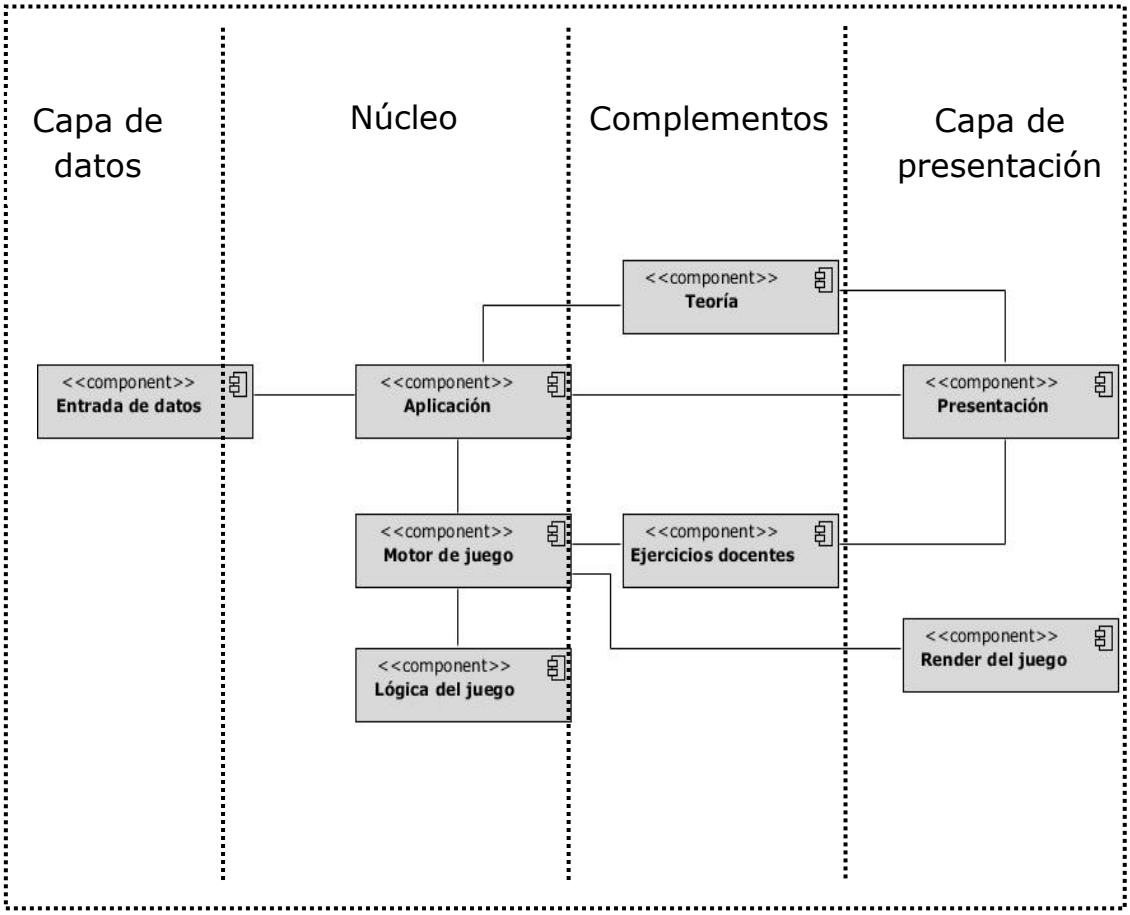


Ilustración 23: Diagrama de componentes

- *Módulo de Entrada de datos*: Este módulo es el encargado de aportar a la aplicación los ficheros auxiliares necesarios para la aplicación. Estos ficheros serán los encargados de obtener todo el material docente así como los diferentes escenarios que se pueden encontrar en el juego y la partida guardada. Este componente será el encargado de realizar el *parseado* a los diferentes ficheros y se comunicará con el componente "*Aplicación*" para ofrecerle dichos datos. Si desea más información acerca del contenido de los ficheros *xml* puede consultar el ANEXO IV en los apartados "Actualizar el contenido de la teoría" y "Actualizar el contenido de los ejercicios".

- *Módulo de aplicación*: Es el módulo central encargado de llevar a cabo la ejecución del juego. Establece el control de las distintas pantallas que componen al juego y la transición entre las mismas según las acciones que realice el usuario. Este módulo es el encargado de acceder al módulo de *entrada de datos* para obtener por ejemplo la partida guardada. Se encarga de comunicarse con el componente de presentación para la salida por pantalla, otra de sus funciones es recoger la entrada de datos del usuario mientras este permanece en las pantallas que controla (todas menos la pantalla del laberinto), que se realiza mediante eventos en la pantalla y es el encargado de ejecutar el *motor del juego*. También es el encargado de comunicarse con el servidor web para la descarga de

actualizaciones. Este componente se relaciona con el componente "Teoría" encargado de mostrar la teoría al usuario.

- *Módulo lógica del juego*: Está formado por los diferentes conceptos que forman el juego. Realiza el control de los escenarios del juego y del protagonista, es la base que sustenta el motor del juego.

- *Módulo motor del juego***:** Su elemento principal es el laberinto y está encargado de establecer su comportamiento y su correcto funcionamiento, controla todo lo que ocurre cuando se inicia un laberinto. Controla el movimiento del personaje mediante un *joystick* analógico en que aparece en la pantalla táctil del teléfono móvil, los movimientos del personaje así como la de otros objetos dinámicos que aparecen en el juego. Es además el encargado de generar el sonido y los efectos de sonido, así como el control de colisiones y físicas del juego. Por último realiza el control de las estadísticas y es el encargado de realizar la salida por pantalla a través del componente "Render del juego". Este componente está relacionado con el componente "Lógica del juego" que es quien dota de significado a los objetos que controla. Por último también se relaciona con el componente "Ejercicios docentes" que controla la actividad del usuario cuando este se encuentra en las pantallas de ejercicios.

- *Módulo teoría:* Se encarga del control de la teoría disponible del videojuego, gestiona la lista con las páginas de teoría y permite recorrerla en ambas direcciones.

- *Módulo ejercicios docentes:* Controla los posibles eventos y acciones que se producen cuando el usuario se encuentra en una pantalla de ejercicios, se comunica con el componente "Motor de juego" para comunicarle los resultados obtenidos del jugador.

- *Módulo de presentación***:** Es el encargado de la salida por pantalla, su función es la de presentar al usuario al usuario las diferentes interfaces y pantallas de la aplicación.

5.2.1 Capas de la arquitectura

El diseño de la arquitectura se ha realizado, como ya se ha comentado, intentando maximizar la cohesión entre los componentes de cada uno de los módulos y minimizar el grado de dependencia entre ellos con el fin de favorecer la reusabilidad. Se ha utilizado un modelo en tres capas, el cual permite la separación de la lógica de negocios, de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

En la ilustración anterior, *ilustración 23*, se puede observar cuál ha sido el modelo de arquitectura seguido. La **capa de datos** está formada por el

componente entrada de datos, será la encargada de obtener el contenido de los ficheros *xml* externos a la lógica del juego, el **núcleo** constituye la parte fundamental del videojuego desarrollado, está formada por una agrupación de tres componentes: La lógica del juego, el componente aplicación y el motor del juego. Las modificaciones en los elementos de esta capa actúan directamente en el sistema de juego, alterando las características del mismo y su funcionalidad. La capa intermedia entre el núcleo y la capa de presentación es la **capa de complementos,** cuya funcionalidad es controlar y gestionar las pantallas de teoría y ejercicios del videojuego. La última capa que forma esta arquitectura es la **capa de presentación** que está formada únicamente por el componente de presentación, el cual es el encargado de las salidas por pantalla.

Las dos capas de los extremos, capa de datos y capa de presentación, afectan al juego en su presentación y contenido pero nunca una modificación en estas capas afectaría al desarrollo del juego y a su funcionalidad.

5.3 Diseño detallado

El diseño detallado que se ha realizado pretende establecer una definición más específica de cada uno de los componentes anteriormente descritos, estableciendo para cada uno de ellos las **clases que los componen**, y definiendo a la vez para cada uno de ellos sus **principales atributos y métodos.** Para mejorar la presentación de este diseño es posible que algunas clases no se ajuste al cien por cien al de la realidad con el propósito de presentar de una forma más clara la solución propuesta.

Los siguientes diagramas que se encuentra en este apartado corresponden al videojuego desarrollado, posteriormente se incluirá también un diseño de la aplicación para la edición de laberintos. Los diagramas correspondientes al videojuego comenzarán con la capa de datos, para posteriormente explicar el núcleo, los complementos y por último la capa de presentación.

5.3.1 Diseño detallado del videojuego

En los siguientes puntos se encuentra el diseño detallado que se ha realizado sobre el videojuego de *m-learning.* Junto con cada uno de los diagramas que definen las capas de la arquitectura aparecerá también una descripción detallando el funcionamiento y la solución propuesta a cada uno de los componentes.

5.3.1.1 Capa de datos

A continuación se muestra el diagrama detallado correspondiente a la capa de datos del videojuego desarrollado para este proyecto.

5.3.1.1.1 Componente Entrada de datos

La capa de datos está formada únicamente por el componente de entrada de datos, cuya función es la obtener la información contendida en los ficheros externos a la aplicación, cuyo contenido escrito en *xml* puede incluir escenarios para crear las partidas, partidas guardadas o todo el material docente referente la lengua inglesa.

Para disminuir el tamaño del diagrama y conseguir así una mayor claridad se han omitido las clases, *RssListening* y *RssMatching* al ser muy similares a la clase que aparece en el diagrama *RssFill.*

La clase *RssParserSax* es la encargada de crear los objetos que posteriormente realizarán el "*parser"* al archivo *xml* correspondiente, ya sea del tipo, *RssProtagonista, RssListening, RssMatching, RssFill o RssLaberinto.* Estas clases devolverán un objeto correspondiente al tipo de fichero que haya sido parseado y la clase *RssParserSax* será la encargada de comunicarse con el componente aplicación de la capa Lógica de negocio para pasarle el objeto creado. En la siguiente página se muestra el diagrama detallado de este componente.

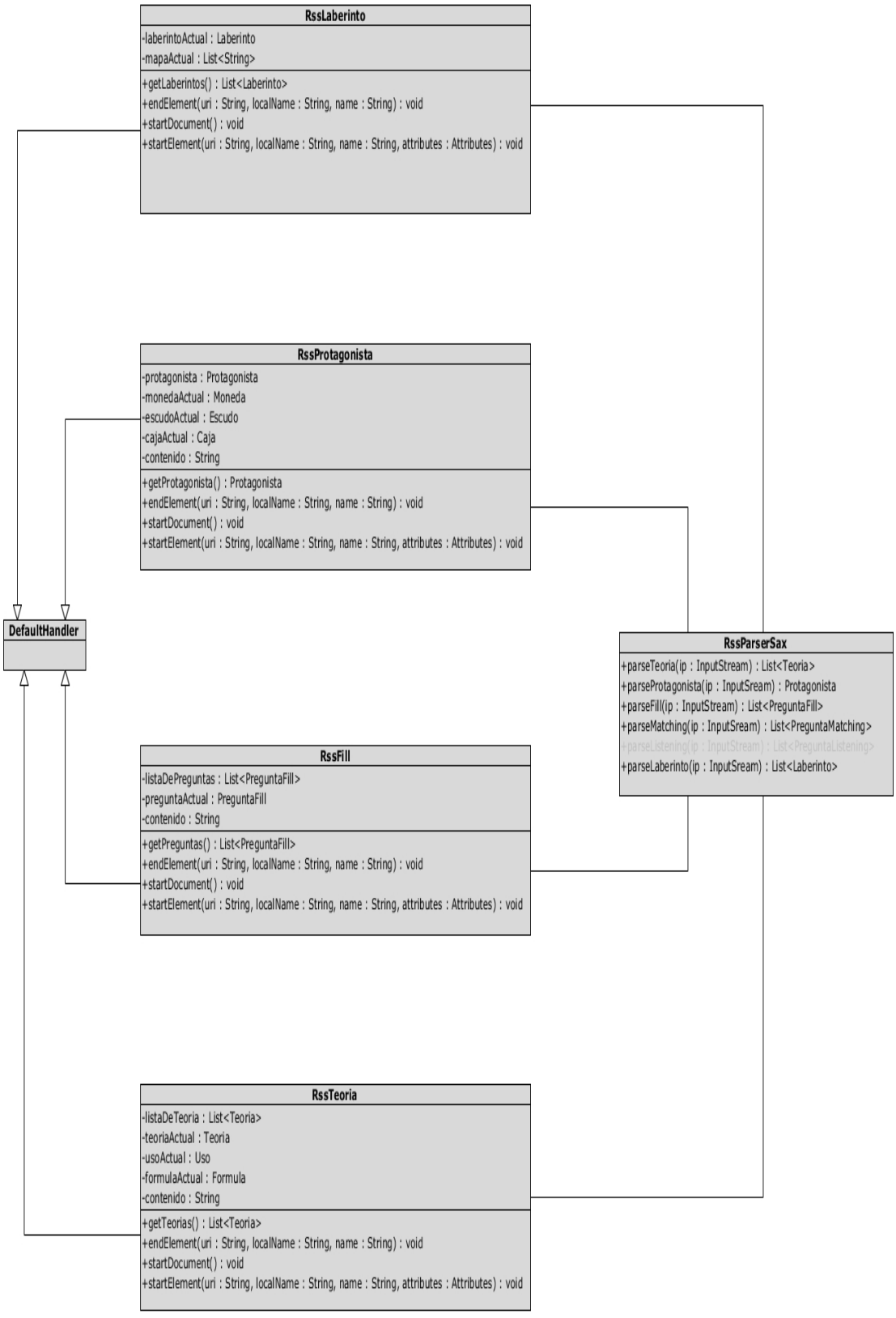


Ilustración 24: Diseño detallado componente Entrada de datos

5.3.1.2 Núcleo y complementos

A continuación se muestra el diagrama detallado correspondiente al núcleo del videojuego y la capa de complementos. El núcleo está formado por tres componentes, "Lógica del juego", "Motor del juego" y "Aplicación", mientras que la capa de complementos está formada por el componente "Teoría" y "Ejercicios docentes"

5.3.1.2.1 Componente Lógica del juego y Motor del juego

El siguiente diagrama que se va a mostrar está compuesto por los dos componentes "*Lógica del juego*" y "*Motor del juego*". Se ha decidido unir ambos componentes en un solo diagrama para reflejar de una forma más clara como se comunican y relacional.

El componente "*Lógica del juego*" está constituido por aquellas clases que componen la lógica del juego y el material docente de la aplicación. Este componente es la base que utiliza el motor del juego, para desarrollar sus actividades.

El componente "*Motor del juego*" está formado únicamente por la clase *ActivityLaberinto,* mientras que el componente "*Lógica del juego*" está formado por el resto de clases.

Como se pude observar en la *ilustración 25*, las clases correspondientes al componente "*Lógica del juego*" únicamente poseen atributos ya que las únicas operaciones que disponen son métodos de accesos y modificación a sus respectivos atributos y estas no han sido representadas. Por el contrario la clase *ActivityLaberinto* dispone una multitud de métodos que controlan todo lo que ocurre en el juego como: el control del protagonista, pintar en la pantalla los objetos, el *scroll* de la pantalla, las colisiones, etc. También se ha incluido la relación entre el motor del juego y los ejercicios docentes.

En la *ilustración 26* puede observar el flujo de ejecución que se produce entre ambos componentes.

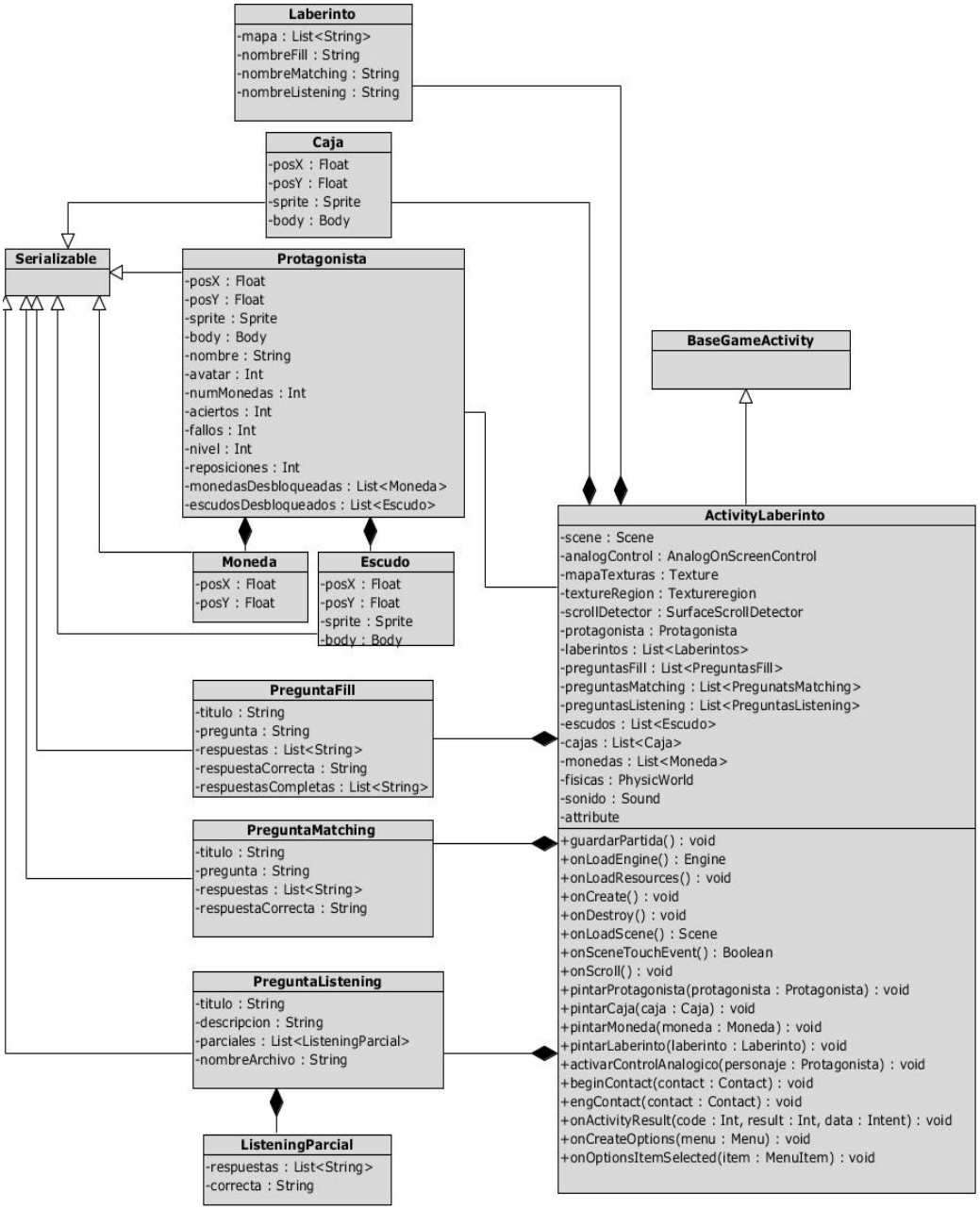


Ilustración 25: Diseño detallado componente Lógica del juego y Motor del juego

Para mostrar el flujo de ejecución y las transiciones que se producen entre el componente "Ejercicios docentes" y "Motor del juego" se presenta el siguiente diagrama de ejecución. En el siguiente diagrama se puede comprobar cómo es la clase *ActivityLaberinto* la encargada de crear el resto de clases, y como estas una vez creadas y realizada su actividad devuelven el control a la clase *ActivityLaberinto*. La clase *ActivityLaberinto* creará el resto de *Activity* cuando el jugador choque con un objeto escudo en el videojuego, lo que producirá el lanzamiento de las actividades correspondientes a los ejercicios.

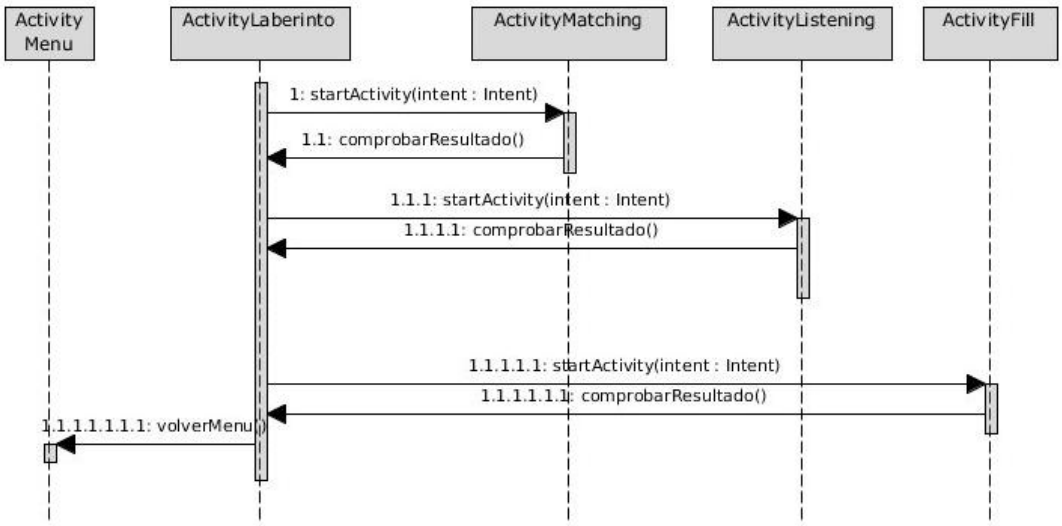


Ilustración 26: Flujo de ejecución del componente Lógica del juego y Motor del juego

5.3.1.2.2 Componente Aplicación

El siguiente componente controla toda la ejecución de las diferentes pantallas del juego. Es el encargado de las diferentes transiciones que se realizan sobre las pantallas con las que interactúa el usuario.

Este componente está relacionado con el componente "*Presentación*", ya que es él quien debe comunicar a este componente que interfaces son las que deben mostrarse. El componente "*Aplicación*" también es el encargado de iniciar el componente "*Motor del juego*" y el componente "Teoría". Una de las funcionalidades más importantes de este componente, consiste en la actualización del contenido didáctico y los escenarios del videojuego. Para ello el sistema se conectará con un servidor web desde donde realizará la descarga de los ficheros.

Cada una de estas clases no tiene relación unas con otras, por eso para mostrar la transición que se puede dar entre clases se han incluido diagramas de secuencia en la *ilustración 28.*

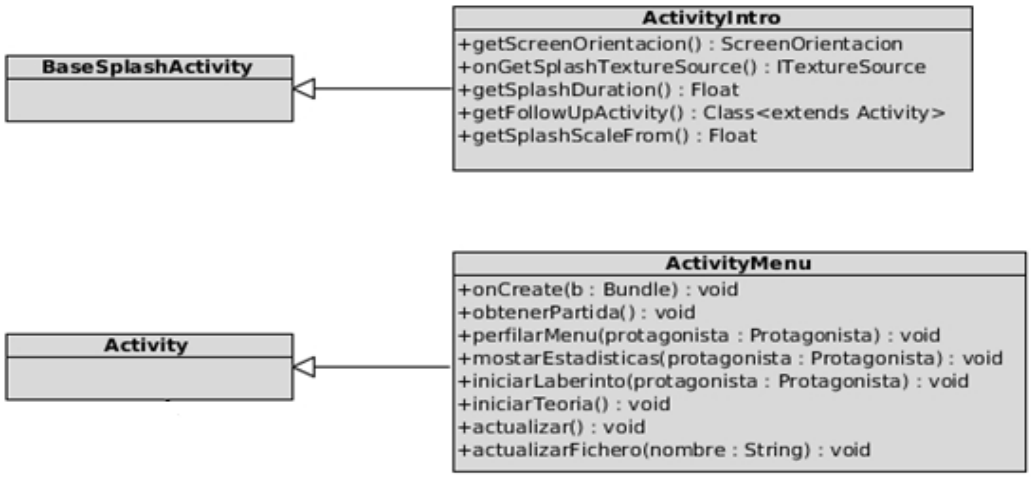


Ilustración 27: Diseño detallado componente Aplicación

A continuación se muestra el diagrama correspondiente al flujo de ejecución de este componente y el componente de "Teoría". El videojuego siempre comenzará por la clase *ActivityIntro*, que después de mostrar una animación pasará el control a la clase *ActivityMenu,* una vez la aplicación se encuentre en este punto, el flujo de ejecución variará según las actividades que desee realizar el usuario. Ya sea pasar a ver la teoría disponible en la aplicación o iniciar/continuar la partida del laberinto.

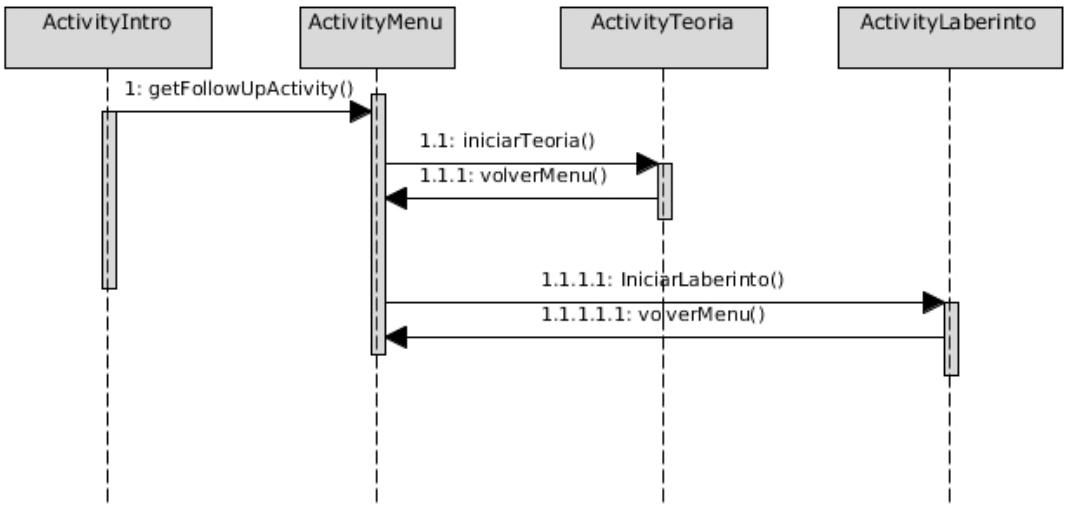


Ilustración 28: Flujo de ejecución componente Aplicación y Teoría

5.3.1.3 Capa de Complementos

En los siguientes aparatados se van a detallar los componentes pertenecientes a la capa de complementos, la funcionalidad de estas complementos es la de dotar de significado a algunas partes del juego como la teoría y los ejercicios.

5.3.1.4.1 Componente de Teoría

Este componente es el encargado crear la estructura de la lista de teoría y almacenar el contenido de la misma, para poder dotar al videojuego del contenido didáctico teórico. Este componente está relacionado con la clase *ActivityMenu* que pertenece al componente "Aplicación" ya que será esta clase la encargada de realizar la llamada para ejecutar la clase *ActivityTeoria* encargada de la gestión de la teoría del videojuego.

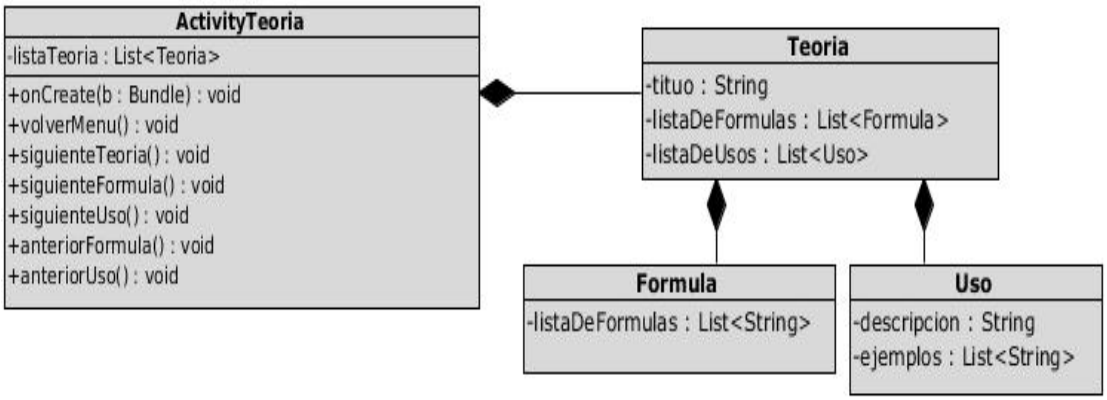


Ilustración 29: Diseño detallado componente Teoría

5.3.1.4.1 Componente de Ejercicios docentes

Es siguiente componente es el encargado de controlar los eventos que se producen en las pantallas de ejercicios y crear la estructura para almacenar los diferentes ejercicios que dispone el videojuego. Este componente será siempre ejecutado desde el motor del juego a quien deberá devolver el resultado obtenido por el jugador y el control de la partida. En la siguiente página se muestra el diagrama detallado de este componente.

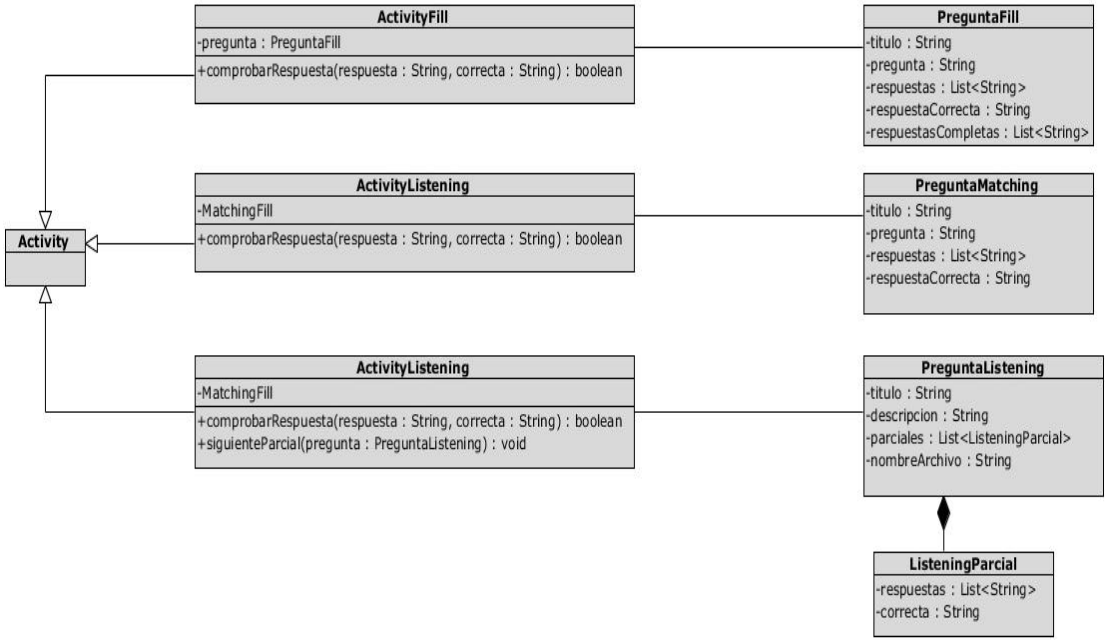


Ilustración 30: Diseño detallado componente Ejercicios docentes

5.3.1.4 Capa de Presentación

El siguiente apartado se corresponde con la capa de presentación que está formada únicamente por el componente de Presentación.

5.3.1.4.1 Componente de Presentación

El componente de presentación no es un componente como los dos anteriores ya que no está formado por clases java, este componente está formado por archivos *xml* que representan los *layout* de cada una de las pantallas. Es decir son los ficheros que utiliza el componente "*Aplicación*" para mostrar las diferentes interfaces que dispone la aplicación desarrollada.
Este componente está formado por los siguientes ficheros *xml*:

- *menu.xml*: Contiene la interfaz gráfica del menú principal del juego. Desde el se puede acceder a crear nuevo jugador, continuar, ver estadísticas, teoría y salir.

- *teoría.xml*: Es la interfaz correspondiente a pantalla de mostrar la teoría que se encuentra disponible en la aplicación de m-learning.

- *estadísticas.xml*: Corresponde a la pantalla de visualizar las estadísticas del usuario.

- *crearJuador.xml*: Esta interfaz corresponde al menú de crear un jugador cuando se inicia una partida nueva. Está compuesta por varios *TexView*, un *EditText* para que el usuario introduzca su nombre, varios

ImageButton para que el usuario seleccione el avatar que desea y dos botones, uno para comenzar la partida y otro para volver al menú.

- *listening.xml, matching.xml* y *fill.xml:* Estos ficheros lo utiliza la clase *ActivityListening, ActivityMatching o ActivityFill* según corresponda para mostrar la interfaz correspondiente a cada tipo de ejercicios. La interfaz correspondiente a los ejercicios de tipo *Listening,* se diferencia del resto, en que a lo largo del transcurso del ejercicio se va modificando al ser estar este tipo de ejercicio compuesto por varios mini-ejercicios.

- *menuLaberinto.xml*: Este fichero es diferente al resto ya que este es un archivo para configurar el menú que aparece cuando el usuario pulsa el botón menú del teléfono móvil. Este fichero únicamente contiene los botones que aparecerán en ese menú.

Para más información sobre el contenido de los componentes de presentación, ver ANEXO II.

5.3.2 Diseño detallado de la aplicación de edición de escenarios

El diseño detallado que se muestra a continuación se corresponde con la aplicación de edición de escenarios que ha sido desarrollada como complemento del videojuego. La principal funcionalidad de esta aplicación, es la de permitir al usuario la creación de laberintos para el videojuego, de una forma sencilla y gráfica, y la posterior creación del fichero *xml,* con el formato que trabaja el videojuego.

Esta aplicación está formada únicamente por dos clases, la clase *App* que hereda de la clase *JFrame* y la clase escenario, que representa cada laberinto. A continuación se muestra el diseño detallado de esta aplicación.

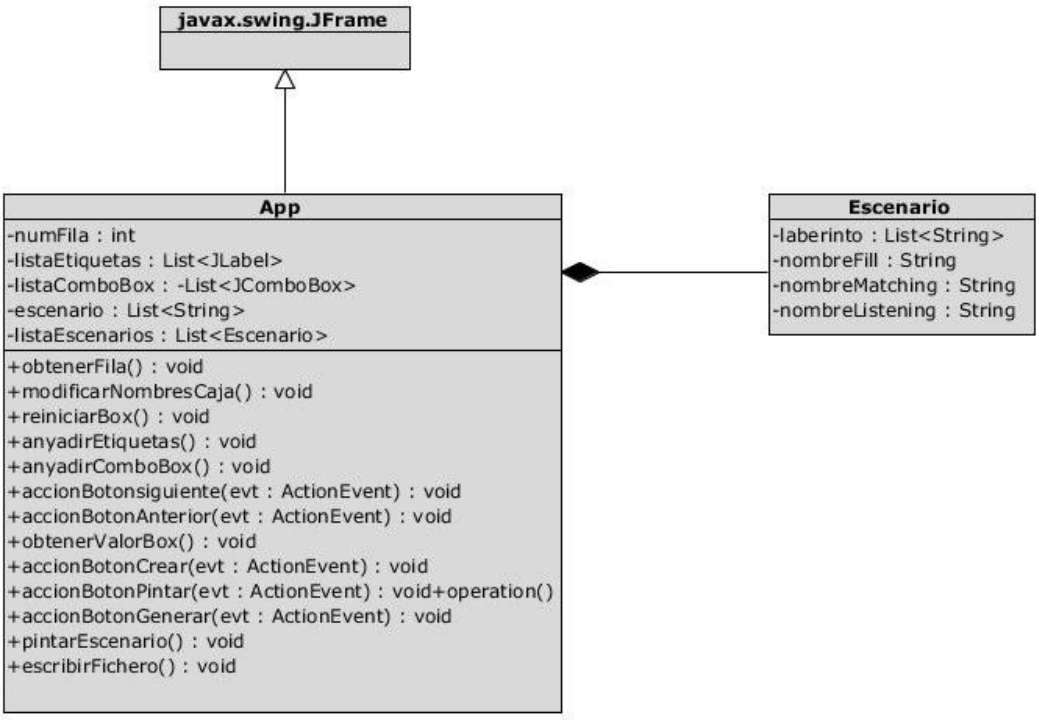


Ilustración 31: Diseño detallado de la aplicación "Editor de escenarios"

Capítulo VI: Implementación

El propósito de este capítulo es describir al lector cómo se ha implementado la arquitectura y el diseño detallado de los capítulos anteriores. Se realizará un estudio en profundidad de las funcionalidades implementadas dando al lector una visión más detallada sobre el funcionamiento interno.

6.1 Implementación del videojuego

En el siguiente apartado se van a detallar las decisiones que se han tomado para el desarrollo del proyecto en los aspectos de gestión de datos, gráficos, menús, control del usuario y motor del juego. En ocasiones se mostrarán fragmentos de código para facilitar la comprensión del lector, y poder así explicar conceptos de nivel de programación.

6.1.1 Gestión de datos

A continuación se va a detallar cual ha sido la solución tomada para la gestión de los datos externos que utiliza la aplicación. Se explicará cual es el formato de fichero para cada tipo de archivo y la estructura del mismo.

El proyecto que se ha desarrollado utiliza varios tipos de ficheros externos, estos están almacenados en la carpeta "*assets*" del proyecto y está compuesta por la siguiente estructura.

- Carpeta *imágenes*: Se almacenan las imágenes que utiliza el motor del juego. Únicamente las que utiliza el motor del juego, ya que el resto de imágenes se guardan en la estructura interna del proyecto y no se consideran archivos externos a la aplicación.

- Carpeta *audio*: Lugar donde se albergan los ficheros de audio correspondientes a los ejercicios de *Listening* y los efectos sonoros del videojuego.

- Carpeta *ttf*: Carpeta donde se almacenan los ficheros *ttf* que se corresponden con los ficheros de las fuentes de texto que se utilizan en la aplicación.

- Carpeta *xml*: Directorio que contiene los ficheros con formato *xml,* este tipo de fichero es usado para almacenar información relativa a: el contenido de los escenarios, la teoría y los ejercicios.

Al igual que ocurre con la carpeta externa al proyecto, el videojuego también utiliza la memoria interna del dispositivo para almacenar aquellos ficheros que se actualizan o modifican a lo largo de la ejecución del videojuego.

A continuación se van a describir cuales han sido los tipos de ficheros que se han utilizado para el desarrollo del proyecto. En la carpeta interna del proyecto solo se podrán encontrar archivos *xml* o archivos de audio, ya que son los tipos de ficheros que se utilizan en las actualizaciones del contenido del videojuego.

- *Imágenes y animaciones*: Para las imágenes y *animaciones* se ha utilizado el formato *PNG (Portable Network Graphics)*, Se ha escogido este formato principalmente debido a que evita la pérdida de calidad de imagen, y utiliza un algoritmo de compresión que no está sujeto a patentes.

- Sonidos: El formato utilizado para los archivos de sonido ha sido *mp3 (MPEG-1 Audio Layer III* o *MPEG-2 Audio Layer III)*, que consiste en un formato de compresión de audio digital patentado, que utiliza un pérdida de información para conseguir un menor tamaño de fichero.

- *Fuentes de texto*: Para la elaboración de los textos que aparecen en el videojuego se han utilizado fuentes ttf, que consiste en un formato estándar de tipos de letra escalables.

- *Escenarios*: La estructura de los laberintos que se utilizan en el videojuego vendrá definida por el contenido del correspondiente fichero de tipo *escenario.xml.* Se ha utilizado *xml* para definir estas estructuras debido a que permite definir los elementos según las necesidades requeridas, y facilita la compatibilidad entre sistemas. Además como se ha mencionado en el capítulo 3, ofrece la capacidad de analizar el contenido de una forma sencilla y rápida.

- *Ejercicios*: Los diferentes ejercicios se almacenar y proporcionarán en un fichero *xml. Cada* tipo de ejercicio es almacenado en un fichero distinto, es decir, los ejercicios *de Fill the gap, Listening y Matching* se almacenan en ficheros diferentes.

- *Teoría*: El fichero que contiene la teoría ha sido almacenado al igual que los escenarios y los ejercicios en un formato *xml.*

6.1.2 Menús

Android ofrece la posibilidad de crear las propias interfaces en lenguaje *Java* pero esto resulta más tedioso que crearlas en formato *xml.* De esta manera además conseguimos separar la representación gráfica del código de la aplicación desarrollada.

Las librerías de *Android* definen un conjunto de elementos personalizados, que heredan de la clase *View*, para el diseño de las interfaces. Los desarrolladores deben anidar los diferentes objetos creados y posteriormente se guardan en el directorio "*layout"*, que se encuentra a su vez en el directorio "*res"* de la aplicación. Cada fichero puede estar formado por objetos simples o objetos compuestos, tantos como el desarrollador crea necesario.

Dichos objetos son de la clase base *android.view.View*. Es básicamente una estructura de datos que almacena la presentación de una pantalla de la aplicación. A continuación se van a detallar los usados en este proyecto:

- *Text:* Permite mostrar texto por la pantalla.
- *EditText:* Permite que un objeto del tipo *Text* sea editado.
- *Button*: Representa un botón, el cual realiza una acción cuando es pulsado.
- ImageButton: Se comporta igual que el objeto de tipo *Button,* pero se le puede añadir una imagen al botón.
- *Toast:* Contiene un pequeño mensaje que se muestra durante un corto periodo de tiempo al usuario. Nunca recibe el foco de acción, con el objetivo de no molestar al usuario, se utilizan para advertir de algún evento que se va a producir.
- *Menú:* Interfaz para gestionar los elementos de un menú. Cada *Activity* de la aplicación puede albergar un menú que se activa al pulsar la tecla menú del teléfono móvil. A continuación se muestra el menú que se ha creado en el videojuego desarrollado.

Ilustración 32: Menú del videojuego

- *ViewGroup: Este elemento es un elemento* especial de la clase *View, su* función es contener y controlar un conjunto de *Views* y/o *ViewGroups*. Cada *ViewGroup* controla el espacio disponible para cada elemento que contiene. Existen varios objetos de tipo *ViewGroup* a continuación se van a describir los más importantes.
    - *FrameLayout: Es el más sencillo, los objetos se añaden a la pantalla a medida que son escritos en el fichero.*

- *LinearLayout:* Su función es la de alinear todos los elementos en una única dirección (vertical u horizontal) y los posiciona uno detrás del otro.
- *AbsoluteLayout:* Permite a los elementos especificar exactamente las coordenadas (x, y) donde van a ser mostrados.
- *RelativeLayout:* Permite a los nodos hijos especificar su posición relativa respecto al nodo padre u otro nodo indicando el ID.

Los elementos son dibujados en la interfaz en el orden en el que aparecen en el fichero *xml*. Cada fichero se compila a partir del árbol de elementos que lo forman, y como consecuencia de esto, solo puede tener un elemento *root*. En la siguiente imagen se muestra el menú de crear una partida nueva. En este menú aparecen distintos elementos como *TextView, EditText, ImageButton, Button,* y elementos compuestos como son el *LinearLayout*.

Ilustración 33: Pantalla crear nuevo jugador

En la siguiente página se muestra el código que se ha utilizado para la creación del menú de iniciar una nueva partida.

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/androidandroid:layout_width="fill_parent"androi
d:layout_height="fill_parent" android:orientation="vertical" android:gravity="center|top">

        <TextView android:text="New player"  android:id="@+id/textCrearJugador"
        android:minHeight="50sp" android:textScaleX="2"></TextView>

         <TextView android:text="Name" android:id="@+id/textIntroduzcaNombre"
        android:minHeight="20sp"></TextView>

        <EditText  android:id="@+id/nombreIntroducido" android:inputType="text" a
        android:focusableInTouchMode="true" android:minHeight="40sp"></EditText>

         <TextView android:text="Choose an avatar "  android:id="@+id/textSeleccionAvatar"
        android:gravity="center" ></TextView>

        <LinearLayout android:id="@+id/linearLayout1" android:layout_height="wrap_content"
        android:layout_width="fill_parent" android:gravity="center">

                <ImageButton android:layout_height="wrap_content"
                android:src="@drawable/avatar1"
                android:id="@+id/botonAvatar1"></ImageButton>

                <ImageButton android:layout_height="wrap_content"
                android:src="@drawable/avatar2"
                android:id="@+id/botonAvatar2"></ImageButton>

                <ImageButton android:layout_height="wrap_content"
                android:src="@drawable/avatar3"
                android:id="@+id/botonAvatar3"></ImageButton>

                <ImageButton android:layout_height="wrap_content"
                android:src="@drawable/avatar4"
                android:id="@+id/botonAvatar4"></ImageButton>

        </LinearLayout>

                <Button android:id="@+id/botonComenzar" android:text="Play"
        android:minHeight="40sp"></Button>

        <ImageButton android:src="@drawable/menu"
        android:id="@+id/botonVolverCrearJugador"
        android:background="@null"></ImageButton>

</LinearLayout>
```

Código 1: Crear nueva partida

6.1.3 Motor del juego, AndEngine

AndEngine es un motor de juego para videojuegos en dos dimensiones que utilicen la plataforma *Android*. Para la visualización de los gráficos utiliza *Opengl* y ha sido desarrollado en código abierto, con el objetivo de que la comunidad de usuarios pueda desarrollar y ampliar sus componentes. *AndEngine* posee una terminología propia, que se va a detallar a continuación:

- *BaseGameActivity*: Consiste en la raíz del juego, contiene el motor y crea la vista donde se va a dibujar todo lo que aparezca en la pantalla del dispositivo. Por cada actividad únicamente puede haber un objeto *Engine*.

- *Engine*: Es el motor interno del juego, se encarga de pintar y actualizar en pantalla todos los objetos que hay en la escena.

- *IResolutionPolicy*: Es una de las opciones del *Engine,* se encarga de la resolución del videojuego. Esta interfaz se ocupa de gestionar las resoluciones del videojuego y adaptar el tamaño de los gráficos a la resolución de la pantalla automáticamente, con el objetivo de que el desarrollador solo tenga que trabajar con una sola resolución.
- *Camera*: Define el rectángulo visible de la escena actual, no tiene porqué ser la escena completa. Existen subclases específicas que el desarrollador puede utilizar para hacer zoom o mover la cámara.

- *Scene*: Se comporta como un contenedor para todos los objetos que se van a dibujar en la pantalla. Una escena puede tener *Layers*, que son capas, donde se sitúan los objetos con la intención de tenerlos de una forma organizada y poder aplicar reglas a todos los objetos que forman una capa.

- *Entity*: Una entidad es un objeto que puede ser dibujado, como imágenes, rectángulos, texto, líneas. Una entidad tiene posición, rotación, zoom, color, etcétera.

- *Texture*: Una textura es una reserva de memoria para almacenar imágenes, deben ser potencia de 2 pero pueden almacenar tantas imágenes como su capacidad le permita.

- *TextureRegion*: Define el rectángulo que ocupa una imagen y se almacena en un objeto *Texture*. Cada *TextureRegion* puede almacenar un *sprite*.

- *PhysicsConnector*: Motor de físicas integrado en el *Engine*.

Durante los siguientes apartados se van a detallar los aspectos concernientes a los gráficos, música, animaciones y control del videojuego a través del motor de juego utilizado para el desarrollo.

6.1.3.1 Gráficos

Como se ha comentado anteriormente *AndEngine* se ocupa de adaptar los gráficos a las diferentes resoluciones que pueden disponer los teléfonos móviles. El encargado de esto será *OpenGL*, motor gráfico con el que trabaja *AndEngine*, el desarrollador puede definir el alto y ancho que el desee para cada objeto y *OpenGL* redefinirá el tamaño para que se visualice correctamente en la pantalla de cada teléfono móvil, manteniendo las proporciones.

Para la elaboración de los gráficos del videojuego se ha utilizado el motor de juego *AndEngine*, este motor de juego es capaz de trabajar con gráficos en 2D mediante la utilización de la clase ***Sprite***. Un *Sprite*s un objeto que puede poseer diferentes atributos y al que se le asigna una imagen. Para utilizar una imagen debemos antes cargarla en un objeto *Texture*. Dicho objeto *Texture* se utiliza para albergar la imagen en memoria, por ello hay que inicializarlo e indicar cuanto espacio de memoria queremos reservar. A continuación aparecen la líneas necesarias para realizar esta operación.

```
//Damos un tamaño de 1024x512,  tiene que ser potencia de 2
this.mTexture= new Texture(1024,512,
TextureOptions.BILINEAR_PREMULTIPLYALPHA);

//Cargamos la imagen correspondiente
this.mFondoTextureRegion=TextureRegionFactory.createFromAsset(this.mTexture, this,
"fondo.png",0,0);
```

Código 2: Carga de una imagen en una Textura

En la primera línea inicializamos la textura y le damos un tamaño de 1024*512, con el segundo parámetro le indicamos qué método utilizar para redimensionar las imágenes que tenemos dentro de la textura. Los diferentes modos que se pueden utilizar son:

- *NEAREST:* La imagen posee menor calidad pero se carga más rápido.
- *BILINER:* La imagen aparece más difuminada, aunque no aparecen pixeles.
- *REPEATING*: La imagen ocupa todo el tamaño de la textura, es decir se repite tantas veces, hasta ocupar toda la textura.

Para cargar las imágenes en memoria se puede realizar de dos formas diferentes:

- El programador es el encargado de colocar la imagen dentro de la textura sin que se superpongan unas sobre otras.

- Dejar que *AndEngine* coloque las imágenes automáticamente utilizando el algoritmo, *Packing Lightmaps*. Este algoritmo consiste en colocar primero la imagen más grande en la primera posición de la textura, posteriormente se colocaría la segunda imagen más grande en la primera posición donde entrase la imagen, así sucesivamente hasta haber colocado todas las imágenes.[1]

Como se ha comentado las imágenes se guardan en objetos de tipo *Texture*, cada una de las imágenes ocupa un espacio en memoria y estas no se pueden sobreponer. Para la realización del videojuego se han utilizado dos objetos de tipo Texture, uno de ellos para las imágenes del joystick analógico y otro para todas las imágenes que componen el videojuego.

A continuación se muestra una imagen aproximada donde se puede observar cómo se han almacenado las diferentes imágenes del videojuego en la textura, la textura creada es de tamaño 1024*512.

Ilustración 34: Posicionamiento de las imágenes en el objeto Texture

Los objetos *Texture* deben ser potencia de dos, por este motivo como se puede observar en la ilustración anterior se desperdicia mucho espacio de memoria, ya que la imagen del fondo del juego (la imagen grande) es mayor de 512 de ancho, lo que obliga a pasar a un tamaño de textura de 1024 de ancho. Como se observa en la imagen no están cargados todos los fondos, todos los avatares o todas las banderas, ya que esto se realiza dinámicamente según el nivel en que se encuentre el usuario y el avatar que haya elegido el usuario. Gracias a esto conseguiremos ahorrar un gran espacio de memoria en el dispositivo.

Una vez tenemos cargadas las imágenes en la textura, el siguiente paso será mostrar los *sprite en la pantalla*. Un sprite es una imagen que posee los siguientes

[1] www.blackpawn.com/texts/lightmaps/default.html

atributos, rotación, dimensión y posición. Para la posición los Sprite utilizan un eje de coordenadas que simula la pantalla que aparece a continuación.

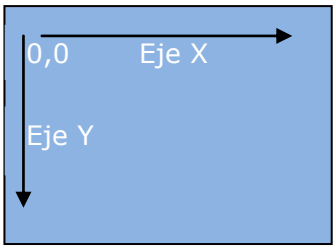


Ilustración 35: Eje de coordenadas Android

Para el desarrollo del videojuego se han utilizado dos tipos de *Sprites:*

- *Sprite estático:* Consisten en imágenes fijas que se representan en la escena del videojuego. Para utilizar un *sprite* estático son necesarias las siguientes líneas de código.

```
//creamos el sprite
sprite = new Sprite (0,200, mCajaTextureRegion);

//añadimos el sprite a la escena
this.scene.getLastChild().attachChild(sprite);
```

Código 3: Carga de un sprite

La primera línea crea el objeto de tipo *sprite,* le asigna una posición X e Y, que corresponde con el eje de coordenadas de la pantalla del dispositivo, explicado anteriormente, el último parámetro le indica la textura que guarda la imagen que se desea cargar con este Sprite.

- *AnimatedSprite (animados):* El motor de juego que hemos utilizado hace uso de las animaciones mediante *frames*. Un *frame* es la unidad mínima de una animación, es decir, será la imagen que se verá en un determinado tiempo. Cada imagen puede tener tantos *frames* como se desee. En la siguiente imagen se muestra, los diferentes *frames* que se han utilizado en el videojuego para realizar la animación de una de las banderas.

Ilustración 36: Frames del sprite animado de la bandera

El orden en el que aparecen los distintos *frames* es importante, ya que *AndEngine* recorrerá de izquierda a derecha y de arriba abajo la imagen. Siendo el primero el de la esquina izquierda superior. A continuación se muestra el código correspondiente con la creación de un *sprite* animado del videojuego.

```
//Definimos la variable
private TiledTextureRegion monedaTextureRegion;

//Inicializamos la texture
this.banderaTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mTexture,
                            this, "bandera.png", 839, 0, 2, 2);
//Creamos el sprite animado
AnimatedSprite bandera = new AnimatedSprite(posX, posY, texture);

//especificamos el tiempo de la animación
bandera.animate(new long[]{300L, 100L, 200L, 50L, 300L, 100L, 200L, 50L }, 0, 7, true);
```

Código 4: Código de creación de un sprite animado

Las dos primeras líneas corresponden con la declaración y la inicialización del objeto de tipo *TiledTextureRegion,* variable donde se almacena la imagen y se indica el número de *frames* que dispone dicha imagen. En la siguiente línea de código, se crea el *sprite* animado y en la última línea se crea la animación. El primer parámetro representa el tiempo que se mostrará en pantalla cada uno de los *frames*, (el primer tiempo es el correspondiente al primer *frame* de la imagen), el segundo y tercer parámetro indica desde que frame hasta que *frame* se quiere crear la animación y el último parámetro indica si se quiere que la animación se repita constantemente una vez acabe.

Con lo explicado en los párrafos anteriores ya tendríamos nuestras imágenes cargadas en memoria y mostradas por la pantalla del teléfono móvil. Pero con esto no es suficiente para poder crear las colisiones entre los diferentes objetos que aparecen en pantalla, para ello necesitamos hacer uso de la clase *Body*. Esta clase representa un objeto físico en la escena del juego, cada objeto *Body* debe estar asociado con un objeto de tipo *sprite*. Gracias a la clase *Body* podremos hacer uso de la clase *ContactListener* que mediante los métodos *beginContact*() y *endContact*() podremos controlar las colisiones.

Para el control de las colisiones es necesario que nos creemos e inicialicemos un objeto del tipo *PhysicsWorld,* que se encarga de registrar todos los objetos de tipo *Body* que aparecen en la escenas, y será el encargado de detectar las colisiones. Este objeto se registrará también en el objeto *Scene* del videojuego al igual que hacíamos con los objetos de tipo *Sprite.* Para dotar de mayor grado al comportamiento de los objetos que se muestran en pantalla, cada objeto *Body* tiene asociado una textura (objeto de tipo *FixtureDef*), la cual permite definir la elasticidad, densidad y fricción del objeto. A continuación se muestra el código correspondiente con la creación de un objeto *Body.*

```
//Creamos las texturas del sprite
final FixtureDef texturasObjeto = PhysicsFactory.createFixtureDef(100, 0, 100);

Body b= PhysicsFactory.createBoxBody(this.mPhysicsWorld, sprite BodyType.DynamicBody,
texturasObjeto))

//registramos el protagonista en las físicas de la escena
this.mPhysicsWorld.registerPhysicsConnector(new PhysicsConnector(sprite,body,true, false));
```

Código 5: Crear un objeto tipo Body

Durante la primera línea de código se crea la textura del objeto. En la segunda línea se crea el objeto de tipo Body y se le pasan los siguientes parámetros. El primero parámetro indica el objeto de tipo *PhysicsWorld* al que está asociado, el segundo, el *sprite* con el que se corresponde el objeto *Body,* el tercer parámetro indica si el objeto puede ser estático o dinámico y con el último parámetro se le asocia las texturas creadas*.* En la última línea de código se registra el objeto *Body* en las físicas del juego.

6.1.3.2 Fuente de las imágenes

Las diferentes imágenes que se han utilizado durante el desarrollo del proyecto tienen diferentes orígenes:

- Algunas imágenes provienen de los códigos de ejemplo del motor de juego, *AndEngine*, como pueden ser los avatares, la imagen de las cajas que aparecen en el videojuego, etcétera.

- Otras imágenes han sido extraídas de internet y modificadas según las necesidades del proyecto. Estas imágenes principalmente son las que aparecen en los menús, como pueden ser las imágenes de las flechas, o las que aparecen en el menú de la pantalla del laberinto.

- Por último hay un grupo de imágenes que son de creación propia como la que aparece en la animación de introducción del videojuego. Para la creación de estas imágenes se ha utilizado el software libre GIMP, como se ha comentado en el capítulo "Entorno de desarrollo".

6.1.3.3 Música y sonido

En este punto se va a detallar cual ha sido la manera utilizada para dotar al videojuego de música, efectos de sonido y archivos de audio correspondientes a los ejercicios de comprensión oral. Para ello se ha utilizado algunas clases que nos proporciona el motor de juego utilizado.

AndEngine ofrece al programador una capa que se encuentra por encima de del API de *Android* que se ocupa del sonido. Esta capa nos ofrece la posibilidad de reproducir archivos con los siguientes formatos: *AAC, MP3, Ogg, MIDI y WAV*.

Este motor de juego divide la capacidad sonora en dos tipos de objetos:

- *Music (música)*: *AndEngine* normalmente no carga por completo los archivos de música en memoria, esto se debe a que suelen ser archivos de gran tamaño. Por este motivo, y con el objetivo de no ocupar muchos recursos del sistema, los archivos de audio los va reproduciendo en flujo.

- *Sound(efectos de sonido)*: Son cargados en memoria, ya que suelen tener un tamaño más pequeño, y así podemos hacer uso de ellos rápidamente.

Para hacer uso de un archivo de sonido es necesario realizar los siguiente pasos:

1. Activar el sonido en AndEngine.
2. Cargar los ficheros que contienen la música o los efectos de sonido.
3. Configurar el sonido: continuidad, repeticiones, volumen...
4. Operaciones sobre el sonido: reproducir, detener, pausar, continuar la reproducción...

A continuación aparece el código necesario para poder hacer uso de un efecto de sonido.

```
//Creamos las opciones del engine
EngineOptions engineOptions = new EngineOptions([...]);

//Activamos el sonido
engineOptions.setNeedsSound(true);
//Creamos el sonido
Sound
sonido=SoundFactory.createSoundFromAsset(mEngine.getSoundManager(),this,"sfx/sonido.
mp3")
```

Código 6: Crear efecto de sonido

Las dos primeras líneas crean las opciones del *Engine* y habilitan los efectos de sonido. Mientras que en la última línea se índica cual es el controlador de sonido, y cuál es el archivo del efecto de sonido deseado. Posteriormente para hacer uso de este efecto de sonido basta con utilizar los métodos *play(), pause() o stop().*

6.1.3.4 Control de usuario

Para el **control del personaje** que el usuario mueve por la pantalla se ha utilizado la clase *analogOnScreenControl* que ofrece *AndEngine*. Esta clase nos permite crear un *joystick* analógico en la pantalla del dispositivo y modificar algunas de sus propiedades como transparencia, tamaño, posición, etcétera. Para poder hacer uso de este objeto también es necesario implementar los métodos *onControlChange(final baseOnScreenCotrol pBaseOnScreenControl, final float pValueX, final float pValueY) y onControlClick(final AnalogOnScreenControl pAnalogOnScreenControl),* los cuales podremos redefinir para que el objeto se comporte como nosotros deseemos.

Un objeto del tipo *analogOnScreenControl* hereda de la clase *BaseOnScreenControl* posee, entre otros, dos atributos de tipo Sprite que almacenan las imágenes correspondientes al control del *joystick* y a la base del *joystick*, como se muestran en las siguientes figuras.

Ilustración 37: Base del joystick

Ilustración 38: Control del joystick

Cuando se crea un objeto del tipo *analogOnScreenControl,* es necesario pasarle la posición donde aparecerán las imágenes y la textura donde se albergan dichas imágenes y un objeto del tipo *IAnalogOnScreenControlListener*, el cual será el encargado de detectar cuando se ha pulsado sobre el joystick mediante los métodos anteriores.

El videojuego que se ha desarrollado también permite hacer ***scroll*** por la pantalla del videojuego. Para ello ha sido necesario implementar las siguientes clases:

- *IScrollDetectorListener*: Esta clase es ofrecida por *AndEngine* para poder controlar el *scroll* de la pantalla. Es necesario implementar el método *onScroll(ScrollDetector pScollDetector, TouchEvent pTouchEvent, floatpDistanceX, floatpDistanceY).*

- *IOnSceneTouchListener*: Esta clase nos da la posibilidad de controlar la pantalla táctil. Es necesario implementar el método *onSceneTouchEvent(ScrollDetector pScollDetector, TouchEvent pTouchEvent, floatpDistanceX, floatpDistanceY*).

6.1.3.5 Animaciones Splash

El videojuego que se ha desarrollado muestra durante la ejecución del mismo varios *Splash*. Estos consisten en una animación que muestra una imagen durante un periodo de tiempo que se va escalando de forma progresiva.

Estos elementos son habitualmente utilizados para realizar introducciones y para dotar al juego de una mayor calidad gráfica. En la *ilustración 39*, que aparece a continuación, se puede observar la animación correspondiente de pasar de nivel.

Ilustración 39: Animación pasar de nivel

En el videojuego desarrollado aparecen animaciones en los siguientes momentos de la partida:

- Al iniciar el juego.
- Tras conseguir pasar de nivel.
- En la finalización del juego.

En la siguiente página se muestra el código correspondiente a la creación de la animación de pasar de nivel.

```
public class ActivitySiguienteNivel extends BaseSplashActivity {

	/**Constante para la duraccion de la animacion*/
	private static final int SPLASH_DURATION = 3;
	/**Constante para el escalado de la animacion*/
	private static final float SPLASH_SCALE_FROM = 0.3f;
	@Override
	/**
	 * Metodo que indica la horientacion de la pantalla
	 */
	protected ScreenOrientation getScreenOrientation() {
		//devolvemos la orientacion de la pantalla
		return ScreenOrientation.LANDSCAPE;
	}
	@Override
	/**
	 * Metodo que carga la textura de la imagen
	*/
	protected ITextureSource onGetSplashTextureSource() {
		//Obtemos la imagen de la intro
		return new AssetTextureSource(this, "imagenes/siguientenivel.png");
	}
	@Override
	/**
	 * Metodo que carga la duraccion de la imagen
	 */
	protected float getSplashDuration() {
		//devolvemos la duracion de la intro
		return SPLASH_DURATION;
	}
	@Override
	/**
	 * Metodo que carga la siguiente activity cuando acaba la animacion
	*/
	protected Class<? extends Activity> getFollowUpActivity() {
		return (Class<? extends Activity>) ActivityPasarDeNivel.class;
	}
		/**
	 * Metodo que carga desde donde se escala la imagen
	*/
	protected float getSplashScaleFrom() {

		return SPLASH_SCALE_FROM;

	}
}
```

Ilustración 40: Código para crear una animación Splash

Las animaciones de tipo *Splash*, suelen utilizarse al iniciar el videojuego, consisten en una imagen fija que suele aparecer con un tamaño pequeño y a medida que transcurren los segundos esta se va ampliando hasta ocupar toda la pantalla.

Este tipo de animaciones gracias al motor de juego, *AndEngine*, son fáciles de implementar. Basta con crear una clase que herede de la clase *BaseSplashActivity* e implementar todos los métodos de los que dispone esta clase. A continuación se detallan brevemente los cinco métodos que son necesarios implementar:

- *getScreenOrientation():* Este método se encarga de establecer la posición en la que se debe mostrar la animación, ya sea vertical u horizontal.

- *onGetSplashTextureSource*(): Es el encargado de devolver la ruta de la imagen que se va a utilizar para la animación.

- *getSplashDuration*(): Devuelve el tiempo que debe durar la animación.

- *getFollowUpActivity*(): Es el método que se ejecuta al finalizar la animación es el encargado de devolver cual es la siguiente clase *Activity* que se tiene que ejecutar.

- *getSplashScaleFrom*(): Indica el tamaño desde donde se empieza a escalar la imagen.

6.1.4 Actualización del contenido

Para las actualizaciones del contenido didáctico y de escenarios del videojuego, se ha utilizado una cuenta en el servidor *DropBox*, que posee una carpeta pública donde se puede alojar ficheros que pueden ser accesibles a cualquier persona. De esta manera nos permite almacenar en dicha cuenta los ficheros para la descarga, que podrán ser descargados mediante el botón *actualizar* del menú principal.

Los profesores o personas encargadas de actualizar el contenido del videojuego, deberán alojar todos los ficheros correspondientes a la actualización en el servidor. Únicamente el fichero *escenarios.xml,* fichero que almacena los escenarios del juego y las relaciones de los distintos ficheros de ejercicios con los laberintos, deberá llamarse siempre con el nombre *escenarios.xml*. Con el objetivo de que el primer fichero en descargarse sea este, el cual pose la referencia al resto de ficheros de ejercicios. Posteriormente una vez descargados los ficheros de ejercicios se analizarán los ficheros correspondientes a los ejercicios de *Listening* para descargar los ficheros de audio correspondientes. En la siguiente ilustración se puede observar el flujo que sigue una descarga.

Para actualizar el contenido de la teoría únicamente se debe subir al servidor el nuevo fichero con el nombre *teoría.xml*

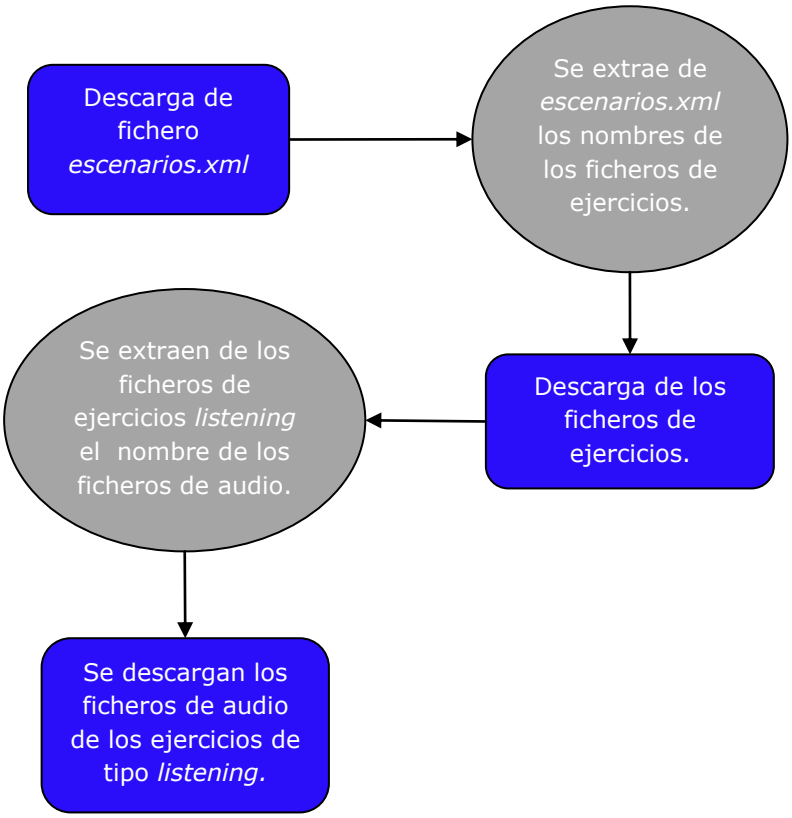


Ilustración 41: Flujo de actualización de contenido

6.2 Resultado final

En los siguientes apartados se va a presentar el resultado obtenido en el videojuego, "*Maz-E-nglish*", para teléfonos móviles *Android* y el resultado obtenido en la aplicación de edición de escenarios para el videojuego desarrollado.

6.2.1 Resultado final del videojuego

En el siguiente apartado se va a detallar y a ilustrar cual ha sido el resultado final del videojuego así como cuáles son las acciones posibles que permite. También se mostrará cual ha sido el resultado final de la aplicación desarrollada para la edición de escenarios.

El videojuego comienza con una animación de introducción y una vez concluida, se muestra la pantalla de menú principal, donde el usuario podrá elegir entre las siguientes opciones:

- Iniciar una nueva partida.
- Continuar una partida guardada.

- Ver las estadísticas.
- Ver las páginas de teoría
- Salir del videojuego

En la siguiente figura se puede observar la imagen que muestra la primera animación del videojuego.

Ilustración 42: Primera animación

Como se ha comentado a lo largo de esta memoria, el videojuego desarrollado está destinado al *m-learning,* para la enseñanza de idiomas. Dicho videojuego permite consultar **páginas de teoría** que pueden estar compuestas de usos y de fórmulas. El usuario podrá navegar entre las diferentes páginas de teoría con los botones que aparecen en la parte inferior de la pantalla. Mostrándose en negrita aquellas partes de la frase que el profesor quiera resaltar. A continuación se muestran dos imágenes que corresponden a diferentes páginas de teoría.

Ilustración 43: Página de teoría, fórmula.

Ilustración 44: Página de teoría, uso.

Una vez el usuario este decidido a **iniciar una nueva partida**, tras pulsar en el botón de nueva partida de la pantalla de menú, se le mostrará la pantalla de crear un nuevo jugador, donde podrá introducir su nombre y elegir el avatar con el que desee jugar. El usuario siempre deberá introducir un nombre de usuario y seleccionar un jugador, de lo contrario el sistema le mostrará un mensaje de error y no le permitirá iniciar la partida. Cuando un jugador ha iniciado una partida y abandona el juego, esta se **guarda automáticamente**, con el objetivo de poder reanudar la partida en otro momento, para ello el usuario deberá pulsar sobre el botón continuar del menú principal. La *ilustración 45* corresponde con la pantalla de iniciar una nueva partida. El usuario podrá seleccionar uno de los cuatro avatares que hay disponibles en el videojuego, los cuales se muestran en la imagen.

Ilustración 45: Pantalla crear jugador

El videojuego está diseñado para que el profesor pueda **añadir tantos laberintos como el desee y actualizar el contenido didáctico,** sin tener que realizar ninguna modificación en el código del juego, pero tan solo los cinco primeros estarán diseñados con diferentes fondos y diferentes muros. El resto de laberintos obtendrán sus imágenes por defecto. Para actualizar el contenido del videojuego basta con subir los nuevos archivos al servidor de *Dropbox* proporcionado con este proyecto, y pulsar sobre el botón actualizar del menú

principal del videojuego. A continuación se muestran los diferentes escenarios que se pueden encontrar en el videojuego.

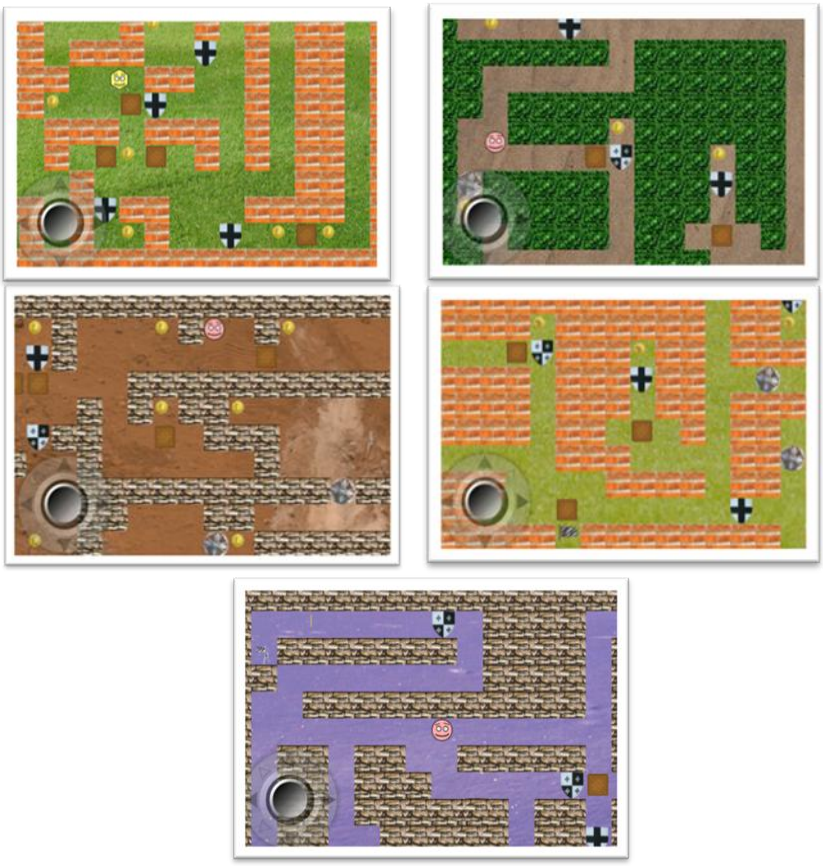


Ilustración 46: Escenarios del juego

Cada uno de los escenarios puede tener asociados uno o varios tipos de ejercicios. Los diferentes ejercicios que se pueden encontrar en el videojuego son ejercicios de *Fill the gap*, ejercicios de *Matching* y ejercicios de *Listening*.

Los **ejercicios** de **Fill the gap**, consisten en un enunciado incompleto y en tres respuestas que pueden ir dentro de ese espacio. El usuario una vez pulse alguna de las respuestas, podrá visualizar en pantalla cual es el resultado final de la frase, y si decide que esa respuesta es la correcta deberá pulsar en el botón de comprobar. Acto seguido se le mostrará con un *tic* verde o una cruz roja si la respuesta es correcta o no.

Ilustración 47: Pantalla ejercicios Fill the gap

Los **ejercicios** de tipo **Matching**, disponen de un enunciado, y tres botones posibles. Cada uno de los botones se corresponde con una posible respuesta al enunciado. En este tipo de ejercicios una vez se pulse el botón de la respuesta que se cree correcta, se mostrará el tic verde o cruz roja según corresponda.

Ilustración 48: Pantalla ejercicios Matching

El último tipo de ejercicios que se puede encontrar en el videojuego son los **ejercicios** de **Listening**, estos ejercicios están compuestos por dos pantallas. Primero una pantalla donde se ofrece un título, una descripción del ejercicio y un botón para empezar a reproducir el archivo de audio. Y una segunda pantalla que se va modificando a la vez que va respondiendo el jugador las preguntas, en esta pantalla el jugador también podrá observar el número de fallos permitidos que le quedan para superar el desafío.

La imagen siguiente muestra la primera pantalla que se mostraría en un ejercicio de tipo *listening* una vez el jugador ha chocado con el escudo correspondiente.

Ilustración 49: Pantalla 1 ejercicios Listening

La siguiente imagen se corresponde con la segunda parte del ejercicio, donde el jugador debe seleccionar la palabra que cree que ha escuchado, automáticamente se pasará a la siguiente palabra hasta la finalización del ejercicio.

Ilustración 50: Pantalla 2 ejercicios Listening

Una vez el usuario a comenzado una partida si lo desea puede **consultar las estadísticas** que ha obtenido, para ello deberá acceder al menú del juego y pulsar el botón de estadísticas.

En este apartado podrá observar en qué nivel del juego encuentra, cuantas respuestas ha acertado, cuantas ha fallado, el número de monedas conseguidas, el tiempo que ha tardado y el número de veces que hemos necesitado recolocar las cajas.

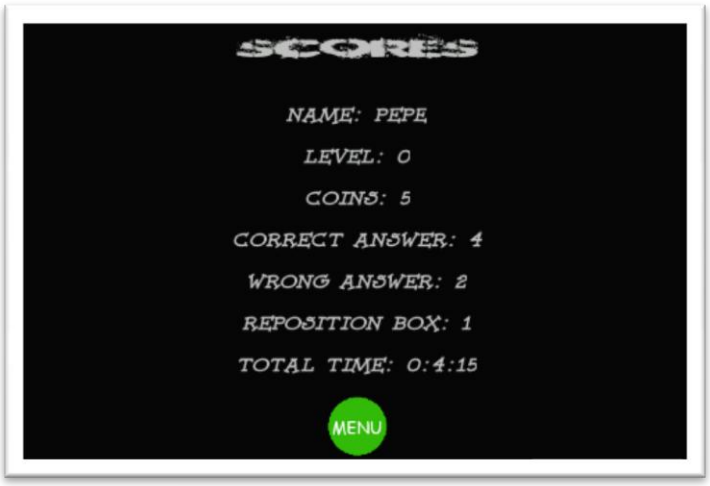


Ilustración 51: Pantalla de estadísticas

El videojuego también permite **recoger las monedas** que aparecen en el escenario y **empujar las cajas**. Debido a que el juego a veces también ofrece desafíos mentales para llegar a obtener las monedas, se ha creado una opción en el menú del juego, que permite **recolocar las cajas**. De esta manera si por error hemos colocado una caja en algún sitio donde no nos permite superar el nivel, con esta opción podremos volver a colocar la caja en su sitio inicial, y conseguir así la moneda que nos faltaba.

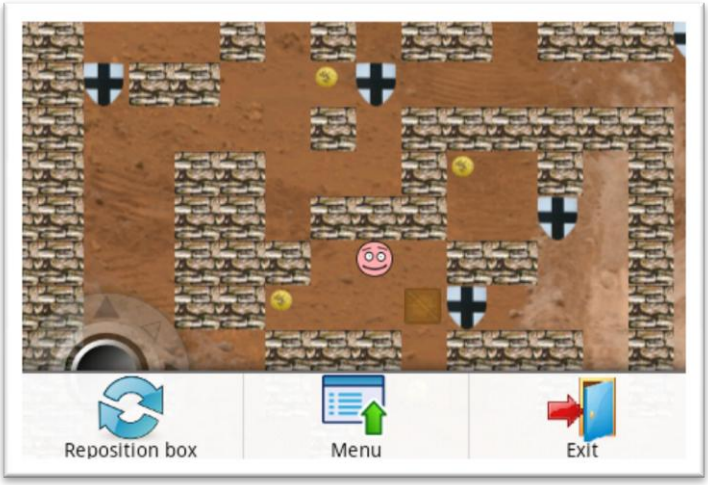


Ilustración 52: Menú pantalla laberinto

Si desea conocer más información sobre el videojuego, puede consultar el ANEXO III "Manual de juego".

6.2.2 Resultado final de la aplicación de edición de escenarios.

Como complemento al videojuego también se ha desarrollado una aplicación escrita en lenguaje *Java,* que permite la creación de laberintos para el videojuego "*Maz-E-nglish*". Esta aplicación facilitará a aquellas personas que deseen crear un nuevo escenario y asociarle los diferentes ejercicios que se corresponden con el nuevo laberinto.

Permitirá crear de una manera sencilla laberintos de tamaño 15*15 donde cada posición estará representada por un objeto. A continuación se nombran los objetos que puede contener un laberinto:

- Un muro.
- Una caja.
- Un protagonista.
- Una bandera.
- Una moneda.
- Un escudo de tipo uno asociado a los ejercicios de tipo *Matching.*
- Un escudo de tipo dos asociado a los ejercicios de tipo *Fill the gap.*
- Un escudo de tipo tres asociado a los ejercicios de tipo *Listening.*

La aplicación creada es una aplicación sencilla e intuitiva de usar, que permitirá a usuarios con pocos conocimientos de tecnología crearse sus propios laberintos para el videojuego "*Maz-e-nglish*". Esta aplicación está compuesta de una sola pantalla donde aparecen dieciocho desplegables que se corresponde cada uno con una posición del laberinto. Dicho desplegable siempre se corresponde con la posición de una columna determinada, pero a medida que se va pintando el laberinto la posición de la fila que le corresponde va aumentando.

Con esta aplicación el usuario tiene la posibilidad de pintar el laberinto fila a fila, así como la de repintar una fila anterior en caso de que se haya producido algún error. Una vez el usuario haya terminado de pintar el laberinto deberá introducir los nombres correspondientes de los ficheros de ejercicios que van asociados con el laberinto, y posteriormente podrá bien generar el fichero con el escenario o seguir creando escenarios y generar el fichero más adelante.

A continuación se muestra una imagen de la aplicación con un laberinto creado.

Ilustración 53: Aplicación editor de escenarios

La aplicación controlará que el laberinto pintado por el usuario sea de tamaño 15*15 así como que introduzca los nombres de los ejercicios asociados con el laberinto. Permite la creación de varios escenarios en un mismo fichero, que generará automáticamente una vez se pulse el botón de generar fichero.

Capítulo VII: Evaluación y pruebas

La necesidad de comprobar el correcto funcionamiento del producto, hace que sea imprescindible un plan de pruebas, que permita diseñar un conjunto de ensayos mediante los que se analice el proceso de ejecución y se garantice que el conjunto de requisitos previamente especificados durante la fase de diseño están cubiertos.

7.1 Objetivos

Durante este apartado se va a describir y determinar las pruebas necesarias que nos permitan comprobar que el sistema cumple todas las especificaciones requeridas. Con el objetivo de no crear un documento demasiado extenso, únicamente se va a incluir la descripción de las pruebas de integración realizadas. El objetivo de este tipo de pruebas identificar y encontrar los posibles errores que se hayan producido en la fase de implementación para que puedan ser rectificados, de tal modo que se asegure la satisfacción de los requisitos.

Las pruebas realizadas durante el desarrollo del proyecto han servido para validar y verificar el correcto funcionamiento y acoplamiento de los componentes del sistema, el adecuado funcionamiento de las interfaces, así como el rendimiento del sistema, y comprobar que los cambios en un componente no afecten al comportamiento no deseado de otros componentes no modificados. En el siguiente apartado se describen las pruebas de integración realizadas.

7.2 Pruebas de integración

El objetivo de estas pruebas es verificar que los subsistemas funcionan correctamente y son capaces de interactuar unos con otros a través de las interfaces definidas.

Para la descripción de las pruebas se utilizará la siguiente tabla:

Identificador	
Objetivo	
Subsistemas	
Precondiciones	
Postcondiciones	
Descripción	

Tabla 62: Plantilla para las pruebas de integración

Los campos que componen la tabla son los siguientes:

- *Identificador***:** distintivo unívoco de la prueba.
- *Objetivo***:** funcionalidad a verificar con la prueba.
- *Subsistemas***:** subsistemas que se ven afectados por la prueba.
- *Precondiciones***:** condiciones necesarias para poder realizar la prueba.
- *Pos condiciones***:** resultados posibles de la prueba
- *Descripción***:** pasos a seguir para realizar la prueba

PR-IN-01	
Objetivo	Iniciar nueva partida.
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *ActivityMenu* de la capa aplicación
Precondiciones	El usuario debe encontrarse en la pantalla de iniciar una nueva partida e introducir un nombre y seleccionar un avatar.
Postcondiciones	Se inicia una nueva partida con el jugador creado.
Descripción	El usuario debe encontrarse la pantalla de iniciar una nueva partida e introducir un nombre y seleccionar un avatar. Si el jugador no selecciona un avatar o no introduce un nombre la aplicación mostrará un mensaje de error, expresando que faltan datos por introducir. La aplicación mostrará un mensaje de alerta si hubiese una partida guardada, avisando de que se va a borrar la partida anterior.

Tabla 63: PR-IN-01 Iniciar nueva partida

PR-IN-02	
Objetivo	Continuar partida guardada
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *Protagonista* de la capa lógica del juego
Precondiciones	El usuario debe encontrarse la pantalla de menú y pulsar sobre el botón continuar.
Postcondiciones	Se inicia la partida guardada anteriormente.
Descripción	La aplicación ofrece al usuario la posibilidad de continuar con una partida guardada. Se debe comprobar que todos los objetos y las estadísticas han sido guardados correctamente.

Tabla 64: PR-IN-02 Continuar partida guardada

PR-IN-03	
Objetivo	Guardar partida.
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *Protagonista* de la capa lógica del juego
Precondiciones	El usuario debe encontrarse en la pantalla del laberinto y salir de la aplicación.
Postcondiciones	Se guardará la partida en la memoria interna del teléfono móvil.
Descripción	La aplicación guardará automáticamente la partida siempre que el usuario cierre la aplicación. El sistema deberá almacenar el escenario, las estadísticas del jugador y los punteros que almacenan la lista de preguntas.

Tabla 65: PR-IN-03 Guardar partida

PR-IN-04	
Objetivo	Ver estadísticas
Subsistemas	- Clase *ActivityMenu* de la capa aplicación - Clase *Protagonista* de la capa lógica del juego
Precondiciones	El usuario debe encontrarse en el menú principal y pulsar sobre el botón estadísticas
Postcondiciones	Se mostrarán las estadísticas del usuario.
Descripción	Se debe comprobar que las estadísticas mostradas son correctas.

Tabla 66: PR-IN-04 Ver estadísticas

PR-IN-05	
Objetivo	Iniciar ejercicio *Listening.*
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *ActivityListening* de la capa aplicación.
Precondiciones	El usuario debe chocar con el escudo correspondiente a los ejercicios *Listening.*
Postcondiciones	Se lanzará una nueva pantalla correspondiente a los ejercicios *Listening.*
Descripción	Siempre que el jugador choque con los escudos correspondientes a los ejercicios *Listening*, se iniciará un ejercicio de dicho tipo.

Tabla 67: PR-IN-05 Iniciar ejercicio Listening

PR-IN-06	
Objetivo	Chocar con objetos.
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *Escudo* de la capa lógica del juego - Clase *Moneda* de la capa lógica del juego - Clase *Muro* de la capa lógica del juego - Clase *Caja* de la capa lógica del juego
Precondiciones	El usuario debe encontrarse en la pantalla del laberinto y chocar contra los diferentes tipos de objetos que se encuentran en el laberinto.
Postcondiciones	Los objetos dinámicos deberán ser arrastrados, mientras que los objetos estáticos guardarán su posición. Los objetos nunca podrán atravesarse entre ellos.
Descripción	El jugador podrá chocarse con los diferentes objetos que se encuentran por el laberinto. Pudiendo desplazar aquellos objetos dinámicos.

Tabla 68: PR-IN-06 Chocar con objetos

PR-IN-07	
Objetivo	Iniciar ejercicio *Fill the gap*.
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *ActivityFill* de la capa aplicación.
Precondiciones	El usuario debe chocar con el escudo correspondiente a los ejercicios *Fill the gap.*
Postcondiciones	Se lanzará una nueva pantalla correspondiente a los ejercicios *Fill the gap.*
Descripción	Siempre que el jugador choque con los escudos correspondientes a los ejercicios *Fill the gap*, se iniciará un ejercicio de dicho tipo.

Tabla 69: PR-IN-07 Iniciar ejercicio Fill the gap

PR-IN-08	
Objetivo	Iniciar ejercicio *Matching.*
Subsistemas	- Clase *ActivityLaberinto* de la capa motor de juego - Clase *ActivityMatching* de la capa aplicación.
Precondiciones	El usuario debe chocar con el escudo correspondiente a los ejercicios *Matching.*
Postcondiciones	Se lanzará una nueva pantalla correspondiente a los ejercicios *Matching.*
Descripción	Siempre que el jugador choque con los escudos correspondientes a los ejercicios *Matching*, se iniciará un ejercicio de dicho tipo.

Tabla 70: PR-IN-08 Iniciar ejercicio Matching

PR-IN-09	
Objetivo	Recorrer teoría
Subsistemas	- Clase *ActivityMenu* de la capa aplicación - Clase *Teoría* de la capa lógica del juego - Clase *Uso* de la capa lógica del juego - Clase *Formula* de la capa lógica del juego
Precondiciones	El usuario podrá recorrer las páginas de teoría hacia delante y hacia atrás.
Postcondiciones	Se mostrará la página de teoría correspondiente.
Descripción	El usuario podrá recorrer las páginas de teoría como si fuesen una lista, podrá recorrerla en ambas direcciones.

Tabla 71: PR-IN-09 Recorrer teoría

PR-IN-10	
Objetivo	Ver menú laberinto
Subsistemas	- Clase *ActivityLaberinto* de la capa motor del juego - Clase *ActivityMenu* de la capa aplicación
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y pulsar sobre la tecla de menú del teléfono móvil.
Postcondiciones	Se mostrará un menú en la parte inferior de la pantalla con las siguientes opciones: - Recolocar cajas - Volver a menú - Salir
Descripción	El usuario podrá acceder al menú del laberinto siempre que se encuentre en la pantalla del mismo.

Tabla 72: PR-IN-10 Ver menú laberinto

PR-IN-11	
Objetivo	Recolocar cajas
Subsistemas	- Clase *ActivityLaberinto* de la capa motor del juego - Clase *Cajas* de la capa lógica del juego
Precondiciones	El jugador debe encontrarse en la pantalla del laberinto y pulsar en la opción del menú de recolocar cajas.
Postcondiciones	Todas las cajas del laberinto deberán volver a su posición inicial.
Descripción	El jugador siempre que lo desee podrá colocar las cajas en su posición inicial pulsando sobre el botón de recolocar cajas.

Tabla 73: PR-IN-11 Recolocar cajas

7.3 Matriz de trazabilidad

En la siguiente página se puede observar la matriz de trazabilidad que relaciona las distintas pruebas realizadas con los requisitos software.

Como se puede observar en la tabla que aparece en la página siguiente, todos los requisitos han sido cumplidos y verificados por al menos una prueba de integración, con lo que podemos afirmar que el **sistema cumple al cien por cien con la funcionalidad requerida**.

	PR-IN-01	**PR-IN-02**	**PR-IN-03**	**PR-IN-04**	**PR-IN-05**	**PR-IN-06**	**PR-IN-07**	**PR-IN-08**	**PR-IN-09**	**PR-IN-10**	**PR-IN-11**
RSF-V-01	■										
RSF-V-02		■									
RSF-V-03								■			
RSF-V-04				■							
RSF-V-05									■		
RSF-V-06									■		
RSF-V-07	■	■									■
RSF-V-08	■	■			■	■	■	■	■		
RSF-V-09					■	■	■				
RSF-V-10					■	■	■				
RSF-V-11		■	■								
RSF-V-12		■	■								
RSF-V-13		■									
RSF-V-14				■							
RSF-V-15				■							
RSF-V-16	■	■						■			
RSF-V-17										■	
RSF-V-18								■			■
RSF-V-19	■	■									
RSF-V-20						■		■			
RSF-V-21					■			■			
RSF-V-22							■	■			
RSF-V-23	■	■									
RSF-V-24									■		
RSF-V-25			■					■			
RSF-V-26	■										
RSF-V-27	■	■									
RSF-V-28	■	■									

Tabla 74: Matriz de trazabilidad de requisititos software y pruebas de integración

Capítulo VIII: Gestión del proyecto

A lo largo del siguiente capítulo se presenta de forma gráfica la planificación realizada para el desarrollo del Proyecto Fin de Carrera y se detallan todos los aspectos relacionados con el presupuesto del proyecto.

8.1 Ciclo de vida del proyecto

Para la elaboración del proyecto se ha seguido un ciclo de vida en **cascada** y se han identificado una serie de fases que abarcan al completo el desarrollo del proyecto. Desde el estudio y desarrollo de la aplicación, hasta la documentación necesaria del proyecto. Se ha elegido este ciclo de vida ya que se considera necesaria la elaboración de cada una de las fases de una manera ordenada para poder continuar de una manera adecuada con el desarrollo del proyecto. A continuación se presentan las fases que se han detectado:

- *Análisis del problema*: Esta fase permite obtener una idea más profunda del problema que se debe resolver y cuáles son los caminos que se deben seguir.

- *Estudio inicial*: Durante esta etapa debe realizarse un estudio sobre la plataforma que se va a desarrollar la aplicación y las herramientas que se van a utilizar para el desarrollo del proyecto.

- *Análisis de la aplicación*: Definir los casos de uso y requisitos software del sistema.

- *Diseño de la aplicación*: Establece la arquitectura del sistema y la definición de los distintos componentes de los que se forma.

- *Implementación de la aplicación*: Fase en la que se lleva a cabo todo el desarrollo del código de la aplicación construida.

- *Fase de pruebas y validación*: Durante esta fase se analizará y validará la aplicación desarrollada con el objetivo de eliminar pequeños fallos de programación y de visualización del videojuego.

- *Redacción de la memoria*: Redacción de toda la documentación necesaria para el desarrollo y la entrega del PFC.

- *Generación de la presentación*: Generación de la presentación utilizada para la lectura del PFC.

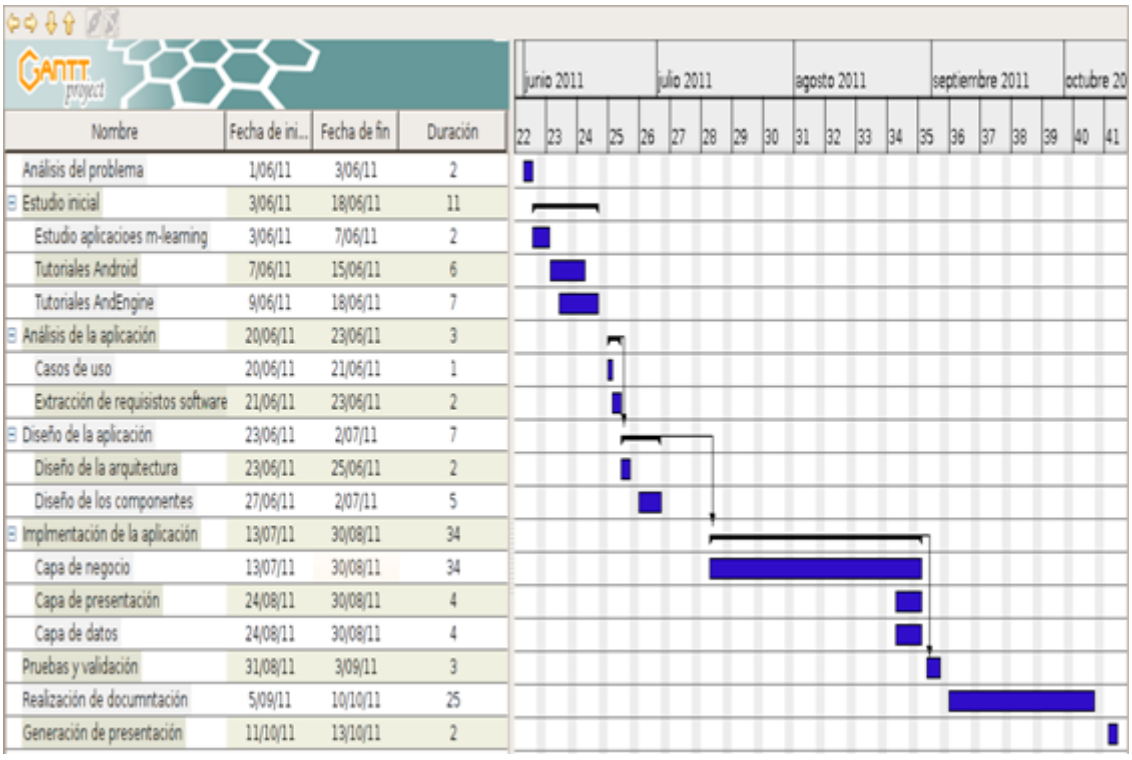
Una vez definidas las fases que constituyen el proyecto se muestra la planificación de las mismas, con la finalidad de realizar una estimación del tiempo necesario para la realización del Proyecto Fin de Carrera.

8.2 Planificación

En este apartado se describe por una parte cuál fue la planificación inicial llevada a cabo al inicio del proyecto, con el fin de disponer de una estimación del tiempo y recursos que serían necesarios para la realización del Proyecto Fin de Carrera. Por otra parte, y con objeto de poder analizar la desviación entre lo estimado y lo finalmente realizado, se incluye además una revisión final de la estimación en la que se refleja el tiempo finalmente dedicado a cada fase del proyecto.

8.2.1 Planificación inicial

A continuación se muestra el gráfico de la planificación realizada en el mes de Junio de 2011.



Nombre	Fecha de ini...	Fecha de fin	Duración
Análisis del problema	1/06/11	3/06/11	2
Estudio inicial	3/06/11	18/06/11	11
Estudio aplicacioes m-learning	3/06/11	7/06/11	2
Tutoriales Android	7/06/11	15/06/11	6
Tutoriales AndEngine	9/06/11	18/06/11	7
Análisis de la aplicación	20/06/11	23/06/11	3
Casos de uso	20/06/11	21/06/11	1
Extracción de requisistos software	21/06/11	23/06/11	2
Diseño de la aplicación	23/06/11	2/07/11	7
Diseño de la arquitectura	23/06/11	25/06/11	2
Diseño de los componentes	27/06/11	2/07/11	5
Implmentación de la aplicación	13/07/11	30/08/11	34
Capa de negocio	13/07/11	30/08/11	34
Capa de presentación	24/08/11	30/08/11	4
Capa de datos	24/08/11	30/08/11	4
Pruebas y validación	31/08/11	3/09/11	3
Realización de documntación	5/09/11	10/10/11	25
Generación de presentación	11/10/11	13/10/11	2

Ilustración 54: Planificación inicial

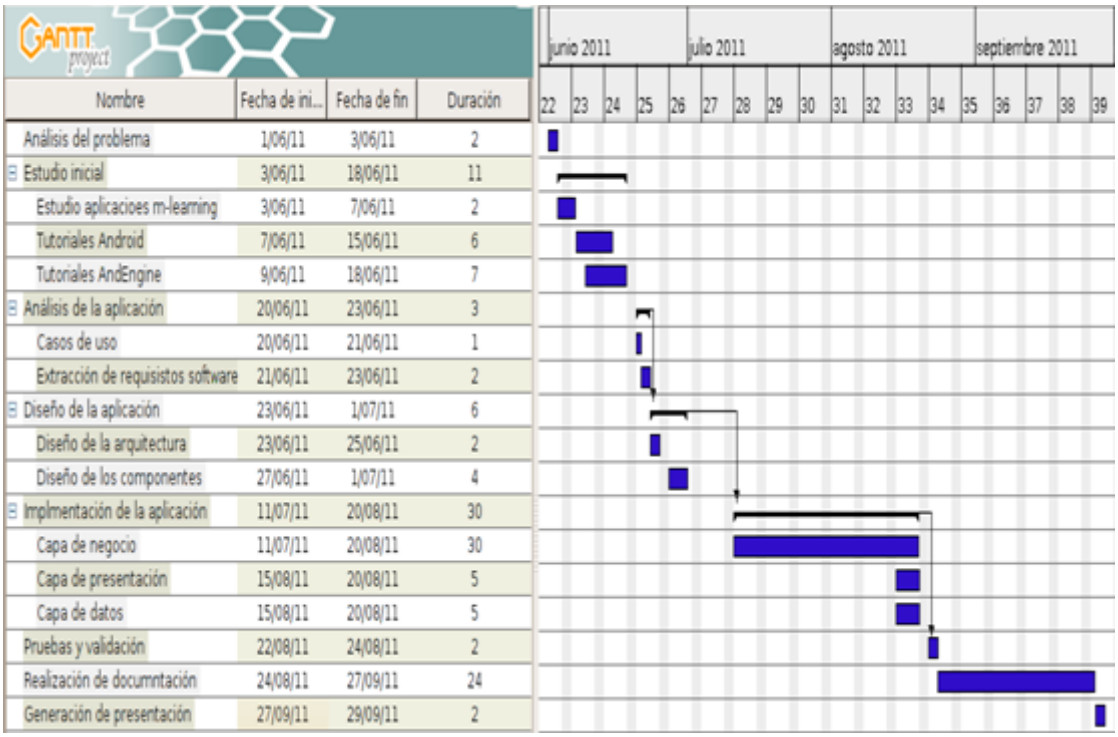
En la ilustración anterior se puede observar la duración y los plazos para cada una de las tareas que lo comprenden. Las fases más complejas están divididas a su vez en tareas más simples. En algunas ocasiones las sub tareas se solapan en el tiempo, esto se debe a que ambas tareas están relacionadas unas con otras a pesar de ser fases independientes.

Por último, existen dependencias entre algunas de las tareas, ya que la realización de alguna de ellas no se puede comenzar sin haber finalizado por completo la tarea de la que depende.

La duración total estimada en la planificación para este proyecto es de casi cuatro meses, con una dedicación de **8 horas al día** aproximadamente.

Como se puede observar en la ilustración anterior, se ha llevado una labor importante de documentación y estudio sobre las herramientas que se iban a utilizar, para adquirir los conocimientos sobre *Android* y *AndEngine*. Esta labor a pesar de no estar recogida en la ilustración anterior se ha llevado a cabo prácticamente durante el desarrollo de casi todo el proyecto, debido a problemas que surgían y al desconocimiento de cómo resolverlos.

8.2.2 Revisión final



Nombre	Fecha de ini...	Fecha de fin	Duración
Análisis del problema	1/06/11	3/06/11	2
Estudio inicial	3/06/11	18/06/11	11
Estudio aplicacioes m-learning	3/06/11	7/06/11	2
Tutoriales Android	7/06/11	15/06/11	6
Tutoriales AndEngine	9/06/11	18/06/11	7
Análisis de la aplicación	20/06/11	23/06/11	3
Casos de uso	20/06/11	21/06/11	1
Extracción de requisistos software	21/06/11	23/06/11	2
Diseño de la aplicación	23/06/11	1/07/11	6
Diseño de la arquitectura	23/06/11	25/06/11	2
Diseño de los componentes	27/06/11	1/07/11	4
Implmentación de la aplicación	11/07/11	20/08/11	30
Capa de negocio	11/07/11	20/08/11	30
Capa de presentación	15/08/11	20/08/11	5
Capa de datos	15/08/11	20/08/11	5
Pruebas y validación	22/08/11	24/08/11	2
Realización de documntación	24/08/11	27/09/11	24
Generación de presentación	27/09/11	29/09/11	2

Ilustración 55: Revisión final

Como se puede observar la planificación final se asemeja mucho a la planificación inicial realizada en el mes de Junio. Resaltar que se ha conseguido realizar en menos tiempo de lo esperado, ya que por motivos de entrada al mundo laboral ha sido necesario realizar un esfuerzo mayor, con el objetivo de presentar el proyecto, antes de empezar a trabajar.

En ambas planificaciones se puede observar un periodo de 10 días correspondientes a un viaje realizado, y por ese motivo no existen tareas planificadas para ese periodo de tiempo.

En ambas planificaciones los fines de semana no se cuentan en el gráfico como días trabajados a pesar que en algunas ocasiones esto no fue así, y hubo que hacer un esfuerzo extra para poder realizar el proyecto fin de carrera en las fechas deseadas.

8.3 Presupuesto

En el siguiente apartado se va a detallar el presupuesto del proyecto. En él se detallaran los gastos de personal, de software y de hardware generado durante el desarrollo del presente proyecto.

Las horas invertidas al total del proyecto son 576 horas aproximadamente, que provienen de los 72 días que se muestran en la planificación final, realizando una jornada de 8 horas al día. Que se descomponen de la siguiente manera:

- *Análisis del problema*: 2 días * 8 horas= 16 horas
- *Estudio inicial*: 11 días * 8 horas= 88 horas
- *Análisis de la aplicación*: 3días * 8 horas =24 horas
- *Diseño de la aplicación*: 6 días *8 horas = 48 horas
- *Implementación de la aplicación*: 30 días * 8 horas = 240 horas
- *Fase de pruebas y validación*: 2 días * 8 horas=16 horas
- *Redacción de la memoria*: 24 días * 8 horas = 192 horas
- *Generación de la presentación*: 2 días * 8 horas = 16 horas

En el siguiente párrafo se puede observar el presupuesto preparado, teniendo en cuenta la plantilla proporcionada por la Universidad Carlos III de Madrid. Los siguientes apartados son el desglose de la plantilla proporcionada con el objetivo de facilitar la lectura. Según la planificación estimada en jornadas de ocho horas, el valor de un hombre/mes es igual a 160 horas de trabajo.

8.3.1 Coste de personal imputable al proyecto

Puesto	Número de horas	Coste hora	Total (€)
Analista	128	30 €	3840
Diseñador	48	32 €	1536
Programador	256	25 €	6400
Responsable de documentación	208	18 €	3744
			15520

Tabla 75: Coste de personal

8.3.2 Coste de software

Para la elaboración del proyecto ha sido necesario disponer de diferentes *software* y en algunos casos de sus licencias correspondientes. A pesar de haber utilizado varias herramientas de código abierto que son gratuitas, para ciertas actividades se han elegido herramientas de pago, ya que ofrecen mejores prestaciones.

Nombre producto	Coste imputable (€)
Microsoft Windows Vista Home basic	232
Microsoft Office Profesional	283
Eclipse Galileo	0
SDK Android 1.2	0
GIMP 2.6	0
GanttProject 2.0	0
	515

Tabla 76: Coste de software

8.3.3 Coste de hardware

Indudablemente ha sido necesario disponer de cierto hardware para la realización del proyecto. Como ocurría con el software ya disponía de este aunque se incluyen los costes que supondría adquirirlo.

Descripción	Coste (€)	% de uso dedicado	Dedicación meses	Periodo de depreciación	Coste imputable (€)
Ordenador portátil compaq presario c 300	699	100	3,8	60	44,27
Sony Ericsson Xperia X10 Mini	141	100	3,8	60	9,12
Pendrive cruzer 8 Gb	18	100	3,8	60	1,14
					54,53

Tabla 77: Coste de hardware

8.3.4 Coste de material fungible

El único coste que se desprende de este apartado, es consecuente de la impresión de la memoria.

Descripción	Coste (€)
Impresión memoria	80
	80

Tabla 78: Coste material fungible

8.3.5 Resumen de costes

Para terminar este apartado, se muestra un resumen de todos los costes, incluyendo una tasa del 10 % de costes indirectos. Por último se muestra una tabla con el coste total con I.V.A y sin I.V.A

Descripción	Costes (€)
Personal	14.200
Software	515
Amortización del hardware	54
Costes de material fungible	80
Costes indirectos (10%)	1.484
	16.333

Tabla 79: Coste total

Descripción	Coste (€)
Total sin I.V.A	16.333
I.V.A 18 %	2.940
TOTAL	**19.273**

Tabla 80: Coste total con I.V.A

Capítulo IX: Conclusiones

En este capítulo se detallan las **conclusiones propias** a las que se ha llegado tras la realización de este Proyecto Fin de Carrera, y **posibles líneas futuras de trabajo y de ampliación del sistema.**

.

9.1 Valoraciones personales

Gracias al desarrollo de este proyecto se ha conseguido obtener un amplio conocimiento de la plataforma *Android*, plataforma que no había estudiado ni trabajado en la carrera, aprendiendo tanto sobre el funcionamiento interno de la plataforma, como sobre las distintas técnicas para desarrollar aplicaciones compatibles con la misma. Esta plataforma se encuentra actualmente entre las más demandadas del mercado, lo que me permitirá competir mejor en el mercado laboral. De igual manera el estudio que he llevado a cabo sobre el área de *m-learning,* ha supuesto que aumente mi interés personal por esta materia, descubriendo un mercado emergente en el cual se están realizando cada vez más investigaciones y trabajos, y que muy probablemente se expanda en los próximos años debido la gran demanda de este tipo de comunicación existente en la sociedad actual.

Uno de los principales contratiempos que he encontrado durante la realización de este proyecto, es el tener que trabajar con un motor de juego existente, algo que nunca había hecho y no habíamos visto en la carrera, con el inconveniente añadido de poseer una terminología propia y contar con una escasa documentación. Esto ha supuesto el tener que realizar un gran esfuerzo por mi parte, para poder conocer el funcionamiento del mismo de forma que pudiese hacer uso de los servicios que ofrecía.

En cuanto al resultado obtenido aunque hay aspectos que se pueden mejorar y ampliar, el balance final es muy positivo, ya que se ha desarrollado un proyecto muy completo en donde he conseguido aprender sobre tecnología móvil, motores de juego y tecnologías para la educación además de haber realizado una aplicación complementaría al videojuego, escrita en lenguaje *Java,* que permite la creación de nuevos escenarios para el videojuego.

Con el resultado obtenido se han cumplido los objetivos iniciales de desarrollar un videojuego para la plataforma *Android* con un fin lúdico y educacional y, a la vez, realizar un estudio sobre *Android*, los motores de juego y el *m-learning*.

Finalmente, añadir como conclusión que el proyecto ha seguido bastante la planificación inicial, y se ha conseguido incluso reducir los plazos que se plantearon al inicio del proyecto, gracias a una mayor dedicación diaria.

9.2 Líneas futuras

Debido al gran avance de los nuevos teléfonos móviles o *smartphones*, la popularidad de los mismos hacen que las posibilidades de *m-learning* sean cada vez más interesantes, lo que permite hacer aplicaciones más potentes y desarrollar procesos educativos en los que se haga uso de la ventajas de estos dispositivos, como ubicuidad y movilidad, con el consiguiente aprovechamiento de tiempos muertos. Por tal motivo este proyecto puede suponer una buena base para el estudio de los videojuegos con fines educativos.

Con la elaboración de este proyecto se abren varias líneas de investigación y desarrollo, desde la mejora de la aplicación presentada como el desarrollo e investigación de nuevos videojuegos destinados a la educación. Entre las mejoras posibles que se pueden realizar al videojuego desarrollado destacan:

- Ampliar el número de tipos de ejercicios. Actualmente el videojuego consta de tres tipos de ejercicios, ampliar el número de ejercicios conseguiría dotar a la aplicación de un mayor atractivo para los usuarios y un mayor poder docente.

- Desarrollo de un editor de ejercicios y material teórico. Para la creación de nuevas páginas de teoría y ejercicios se está desarrollando otro Proyecto Fin de Carrera, dirigido también por Telmo Zarraonandia, que permitirá la creación de este tipo de material a través de una interfaz gráfica fácil y sencilla de usar, de forma que pueda ser posteriormente cargado en el videojuego.

- Mayor personificación de los avatares por parte de los usuarios. Con el objetivo de que los jugadores se sientan más identificados con sus personajes, se podría proporcionar una mejora para que los usuarios obtuviesen sus avatares de fuera del sistema.

- Ofrecer un mayor nivel de detalle en las estadísticas, como por ejemplo mostrar el número de respuestas acertadas y falladas por cada tipo de ejercicio, así como mostrar una evolución de los aciertos del jugador, con el objetivo de que pueda comprobar su nivel de progreso y aumentar con ello el grado de implicación del usuario.

- Permitir a los usuarios mostrar sus estadísticas a los demás jugadores y/o tutores, mediante la publicación de sus resultados en una página web.

- Permitir la descarga de escenarios, teoría y ejercicios desde la aplicación para poder actualizar el contenido docente y lúdico.

Referencias

1. http://www.cmt.es Comisión del Mercado de las Telecomunicaciones.
2. http://www.gartner.com Proyecto de investigación de tecnologías de la información.
3. http://www.isfe.eu/ Federación Europea de Software Interactivo.
4. José María Benito García 'El mercado del videojuego: unas cifras', ICONO14 Nº 7 2005
5. Félix Etxeberria, Teoría de la Educación en la Sociedad de la Información, 2001, Vol. 2
6. http://www.nicoland.es Página web del videojuego Nicoland
7. http://www.nintendo.es/NOE/es_ES/games/nds/brain_training_del_dr_kawashima_cuntos_aos_tiene_tu_cerebro_3234.html Página web del videojuego *Brain training*
8. www.secondlife.com Página web de la plataforma *Second Life*
9. Elizabeth Acosta Gonzaga, Aldo Ramírez Arellano, José Antonio Rodríguez Mancera, "Objetos de Aprendizaje para m-learning como herramientas para generar ventajas en el proceso de aprendizaje". http://148.204.103.95/somece2010memorias/documentos/AcostaGonzagaElizabeth.doc
10. Alex Ibáñez Etxeberria y José Miguel Correa Gorospe, Universidad del País Vasco, y Mikel Asensio Brouard, Universidad Autónoma de Madrid. "M-learning: Aprendiendo historias con mi teléfono, mi GPS y mi PDA". http://www.uam.es/proyectosinv/idlla/docs/01-04.pdf
11. http://www.openhandsetalliance.com/ Página oficial de la *Open Handset Allicance.*
12. http://www.dalvikvm.com/ Página oficial de la máquina virtual de *Android.*
13. http://www.sqlite.org/ Página oficial de la SqLite
14. http://www.freetype.org/ Página oficial del organismo *Free Type.*
15. http://developer.android.com/guide/basics/what-is-android.html Guía oficial de desarrolladores *Android.*
16. http://www.opengl.org Página oficial de *OpenGl.*
17. http://droideando.blogspot.com/2011/02/ciclo-de-vida-de-una-aplicacion-android.html Guía del motor de juego AndEngine
18. Francisco A. Madera Ramírez "Herramientas de Programación Gráfica para Desarrollo de Videojuegos" Universidad Autónoma de Yucatán, Facultad de Matemáticas.
19. www.directx.es/ Página oficial de Microsoft directx.
20. http://www.unityspain.com/ Página oficial del motor de juego Unity3D.
21. The WSJ Technology Innovation Award Winners Artículo del diario digital Wall Street Journal.
22. http://www.stonetrip.com/ Página oficial de la empresa creadora de Shiva3D.
23. http://www.genbeta.com/imagen-digital/unreal-engine-developer-kit-en-descarga-gratuita Artículo sobre Unreal Engine 3.
24. https://www.udk.com/licensing Página oficial de Unreal Engine 3.
25. http://droideando.blogspot.com/2011/03/introduccion-andengine-parte-i.html Guía sobre AndEngine.

26. http://box2d.org/ Página oficial del motor de físicas box2D.
27. http://www.andengine.org/ Página oficial del motor de juego AndEngine.
28. http://mundogeek.net/archivos/2010/01/18/las-estadisticas-de-los-videojuegos/ Artículo de la página web mundogeek.
29. Frank Ableson, Charlie Collins, Robi Sen, 'Android, Guía para desarrolladores' (Ediciones Anaya Multimedia, 2010)
30. http://www.timemachineapps.com/blog/2011/02/eligiendo-un-frameworks-de-desarrollo-para-android/ Estudio sobre *Andegine* vs *Rokon*
31. http://itunes.apple.com/us/itunes-u/bon-depart-beginners-french/id380227515 Web de la aplicación *m-learning* para aprendizaje de francés.
32. http://itunes.apple.com/us/app/adl-mlearning-guide/id444482253?mt=8 Web de la aplicación *m-learning* para la formación de de contenidos de aprendizaje móvil.

Tutoriales AndEngine

A continuación se detalla la referencia de algunos tutoriales de AndEngine utilizados para la realización del proyecto.

Guía tutorial para programar juegos 2D en Android utilizando AndEngine.
http://droideando.blogspot.com/
http://www.andengine.org/forums/tutorials/

Glosario

Término	Descripción
CMT	Comisión del Mercado de las Telecomunicaciones.
ISFE	Federación Europea de Software Interactivo.
TIC	Tecnologías de la información y la comunicación.
OA	Objeto de aprendizaje.
Antialiasing	Procesos que permiten minimizar el *aliasing* cuando se desea representar una señal de alta resolución en un sustrato de más
Sprite	Mapa de bits, similar a una imagen.
LOD	Level of Detail.
APP	Aplicación de dispositivo móvil.
XML	*Extensible Markup Language.*
SAX	*Simple API for XML.*
Ttf	TrueType es un formato estándar de tipos de letra escalables.
RSF-V-XX	Requisito software funcional del videojuego.
RSF-A-XX	Requisito software funcional de la aplicación de edición de escenarios.
RSNF-V-XX	Requisito software no funcional del videojuego.
RSNF-A-XX	Requisito software no funcional de la aplicación de edición de escenarios.
CU-XX	Casos de uso.
PR-IN-XX	Prueba de integración.
OA	Objeto de aprendizaje.
PNG	Portable Network Graphics.
OHA	Open Handset Alliance.

Tabla 81: Tabla del glosario

ANEXO I Ejemplo de xml-parser

SAX (*Simple API for XML*) es una *API* de analizador sintáctico en serie para *XML*. SAX proporciona un mecanismo para leer datos de un documento *XML*.
El funcionamiento de SAX, se produce por eventos, el usuario define unos métodos que son llamados cuando ocurren determinados eventos como iniciar un documento, leer el inicio de un elemento, leer el final de un elemento…

Los atributos de los elementos se pasan como argumentos a los métodos cuando son lanzados los eventos.

A continuación se muestra un ejemplo del funcionamiento de un analizador mediante SAX.

Dado el siguiente código en XML:

<?XMLversion="1.0" encoding="UTF-8"?>
<PrincipalElemento="1">
<PrimerElemento> casa </PrimerElemento>
<SegundoElemento color="verde">coche </SegundoElemento></Principal>
</PrincipalElemento>

Los pasos que realizaría el analizador serían los siguientes:

1. El primer evento en ejecutarse sería el método StartDocument, donde el desarrollador debe implementarlo y puede añadir las funciones que el desee.

2. Posteriormente se ejecutaría el método startElement recibiendo como atributos "atributo" con valor igual a "1" y con el contenido de "casa".

3. Una vez finalizado el método startElement se ejecuta el método endElement con el PrimerElemento.

4. Continuaría repitiéndose los puntos 2 y 3 hasta llegar al final del documento donde se ejecutaría el método endDocument.

El análisis mediante SAX es unidireccional, es decir, que los datos no pueden ser releídos a no ser que se inicie el análisis otra vez. Existen otros analizadores como DOM que si permiten lectura en ambas direcciones pero que necesitan disponer del documento en memoria lo que produce un mayor consumo de recursos. Otro de los inconvenientes que tiene SAX es que no se puede acceder a un elemento directamente sino que debe recorrerse todo el documento.

ANEXO II Contenido de un fichero xml de presentación

A continuación se muestra la estructura de un fichero xml correspondiente al componente de presentación. Este fichero se encarga de la visualización en pantalla de los diferentes objetos de la clase View que implementa la interfaz.

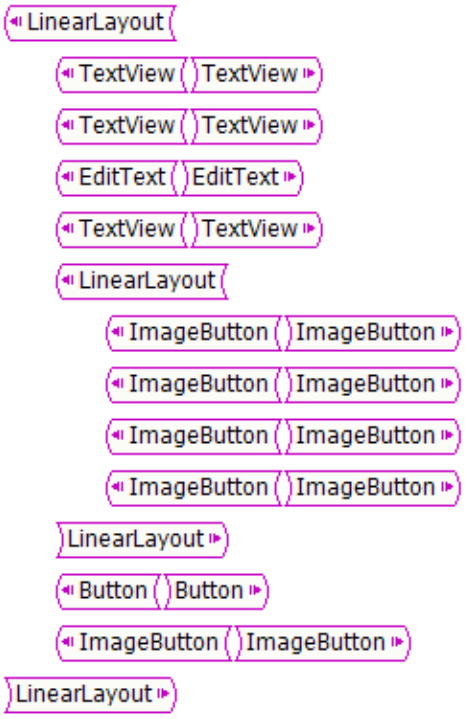


Ilustración 56: Contenido de un fichero xml

Como se puede observar el fichero contiene varios *LinearLayout* ya que son necesarios para albergar los objetos que heredan de la clase *View*, como los que se muestran en la ilustración. El ejemplo que se muestra está compuesto por líneas de texto (objetos *textView*), botones con imágenes (objetos *ImageButton*), botones (objeto *Button*) y los mencionados contenedores (objeto *LinearLayout*).

A pesar de que no se muestra en la ilustración cada uno de los elementos xml puede tener a su vez innumerables atributos que pueden indicar el tamaño, tipo de letra, color, transparencia, colocación en la pantalla, etc.

ANEXO III Manual del videojuego

En las siguientes páginas se presenta el manual de juego para el videojuego "Maz-E-nglish".

Instalación del videojuego

Para la instalación del videojuego "*Maz-E-nglish*" es necesario disponer del fichero con extensión .*apk* que contiene el dicho videojuego y configurar el teléfono móvil para que permita aplicaciones de origen desconocido. Una vez tenga el fichero en la tarjeta de memoria o lo descargue por internet tan solo tiene que ejecutar el fichero mencionado y se realizará la instalación automáticamente.

Instrucciones del videojuego

En los siguientes apartados se va a describir las posibles acciones que puede realizar el usuario y como llevarlas a cabo.

Menú principal

Una vez terminada la animación de introducción del videojuego, este ofrece su menú principal que consta de una serie botones que se corresponden con las posibles acciones que puede realizar el usuario, una vez elegida la opción pulse la pantalla sobre el texto que se corresponda con la opción elegida. Si es la primera vez que inicia una partida el videojuego le mostrará desactivadas las opciones de continuar y estadísticas al no tener una partida guardada, de lo contrario él menú le ofrecerá una serie de acciones que se detallarán a continuación. En la *ilustración 57* se muestra la pantalla de menú principal con las dos opciones mencionadas.

Ilustración 57: Pantalla de menú principal

Nueva partida

Esta opción le permite al usuario iniciar una nueva partida en el juego "Maz-E-nglish" Si ha seleccionado esta opción se le mostrará la pantalla que aparece a continuación.

Ilustración 58: Pantalla nueva partida

Antes de comenzar una partida nueva, debe saber que si disponía de alguna partida guardada esta se perderá y no podrá recuperarla. Esto se le mostrará al usuario mediante un mensaje como el que se observa en la siguiente ilustración.

Ilustración 59: Pantalla nueva partida con mensaje

Una vez este seguro que quiere iniciar una partida debe introducir un nombre para el personaje y seleccionar un avatar de los cuatro que se muestran en pantalla. Para introducir el nombre presione sobre el cuadro blanco que aparece en la pantalla y escriba el nombre elegido. Para la selección del avatar presione sobre el botón que corresponda con el avatar elegido. En caso de que no haya introducido

un nombre o no haya seleccionado un avatar se le mostrarán los siguientes mensajes de error.

Ilustración 60: Pantallas nueva parida con errores

Una vez desee comenzar la partida pulse sobre el botón *Play*, si por el contrario desea volver al menú principal pulse sobre el botón de Menú.

Continuar una partida guardada

Para continuar una partida guardada tan solo debe pulsar sobre la opción continuar del menú principal. Recuerde que si esta opción le sale desactivada es porque no dispone de ninguna partida guardada en su teléfono móvil. Una vez pulsado el botón le aparecerá automáticamente la partida tal y como la dejo por última vez.

Ver estadísticas

Si desea conocer las estadísticas conseguidas durante su partida pulse sobre el botón estadísticas del menú principal y se lo mostrará una pantalla como la que aparece a continuación.

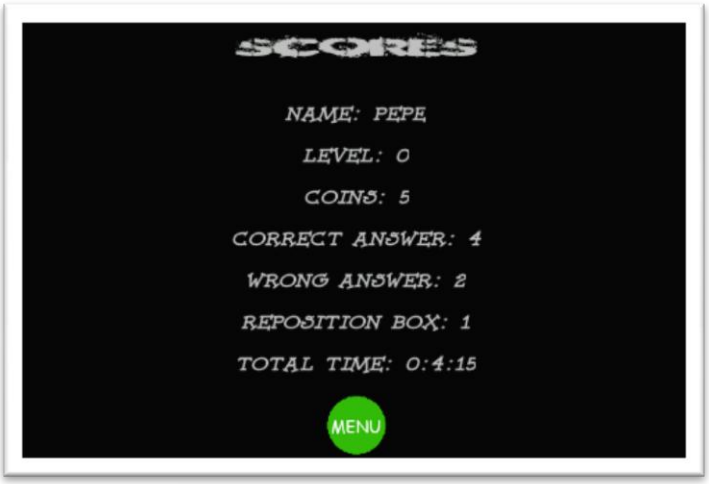


Ilustración 61: Pantalla de estadísticas

Para volver al menú principal pulse sobre el botón menú.

Ver teoría

Si la opción desea es ver y repasar la teoría disponible pulse sobre el botón de teoría del menú principal. La teoría está formada por usos y formulas que podrá consultar cuando usted lo desee. Para navegar entre las diferentes páginas de teoría que dispone la aplicación pulse sobre el botón de ir hacia delante o ir hacia atrás según lo desee. Una vez pulsado se le mostrará la pantalla correspondiente a la primera teoría que tenga disponible la aplicación. En las siguientes ilustraciones se muestran una página de teoría correspondiente a un uso y a una formula. En las formulas podrán aparecer resaltadas en negrita la palabras más importantes.

Ilustración 62: Pantalla de teoría de un uso

Ilustración 63: Pantalla de teoría de una fórmula

Una vez haya recorrido toda la teoría en alguna de las direcciones se le mostrará un mensaje advirtiéndole de que no hay mas teoría disponible. Si desea volver al menú principal pulse sobre el botón menú.

Salir del videojuego

Si desea salir y abandonar el videojuego pulse sobre el botón salir del menú principal o sobre el botón salir del menú del laberinto. No debe preocuparse por guardar la partida ya que esta se guarda automáticamente.

Jugar al laberinto

Una vez haya iniciado o continuado una partida se le mostrará el laberinto del juego. En la *ilustración 64* puede observar como es el laberinto de una partida.

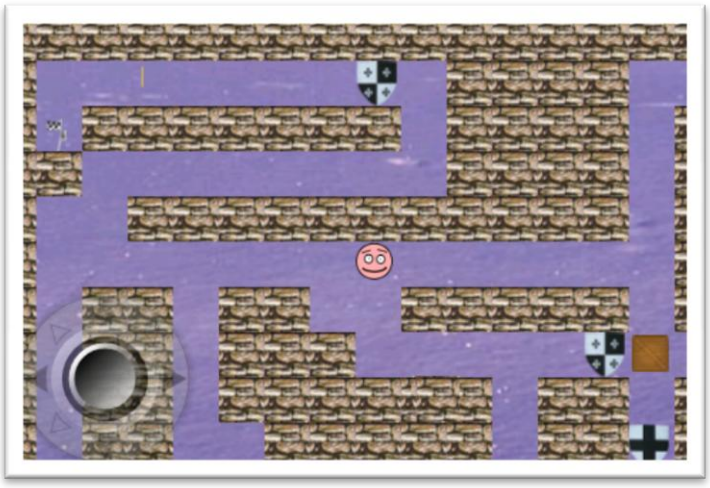


Ilustración 64: Laberinto del juego

Objetivo del juego

El objetivo del juego es conseguir todas las monedas de oro que aparecen en la pantalla y llegar hasta la bandera. Para ello deberá superar una serie de desafíos que se detallan más adelante. Una vez haya superado el nivel, pasará a jugar en el siguiente nivel donde le esperan nuevos desafíos, retos y laberintos.

Mover jugador y mover la pantalla

El usuario podrá mover el personaje mediante el joystick analógico que le aparecerá en la parte inferior izquierda de la pantalla. Para ello arrastre el dedo desde el joystick hacia la dirección deseada.

Si usted desea mover la pantalla para poder observar el laberinto arrastre el dedo por la pantalla en la dirección deseada. Atención: Esta función no está disponible si arrastra el dedo por la zona del joystick.

Conseguir monedas y entrar en desafíos

Para conseguir las diferentes monedas que aparecen en la pantalla tan solo debe chocar con ellas, estas desaparecerán automáticamente de la partida y se le sumarán a sus estadísticas.

Para entrar en los diferentes desafíos debe chocarse con los escudos que aparecen en la pantalla. Existen tres tipos de desafíos, no dude en participar en todos ellos. En la *ilustración 65* puede observar las monedas de oro, así como los diferentes escudos que hay para cada uno de los desafíos.

Ilustración 65: Pantalla con monedas y escudos

Mover y recolocar cajas

A lo largo del juego en ocasiones el jugador tendrá la necesidad de mover las cajas que obstaculizan su paso, para ello debe chocar con el personaje contra una de las cajas. Si debido a esto resulta imposible acceder a algún objetivo del juego puede volver a colocar la caja en su sitio original pulsando sobre la tecla menú de su teléfono móvil y posteriormente sobre el botón recolocar cajas. En el apartado de menú del laberinto podrá encontrar más información sobre este menú.

Menú del laberinto

Si el usuario se encuentra en la pantalla del laberinto y pulsa sobre la tecla menú de su teléfono móvil le aparecerá de forma automática un menú con las siguientes opciones: Recolocar cajas, menú y salir.

Pulse salir si desea salir del videojuego, pulse menú si lo que desea es volver al menú principal o pulsé recolocar cajas si lo que desea es que las cajas vuelvan a su posición inicial.

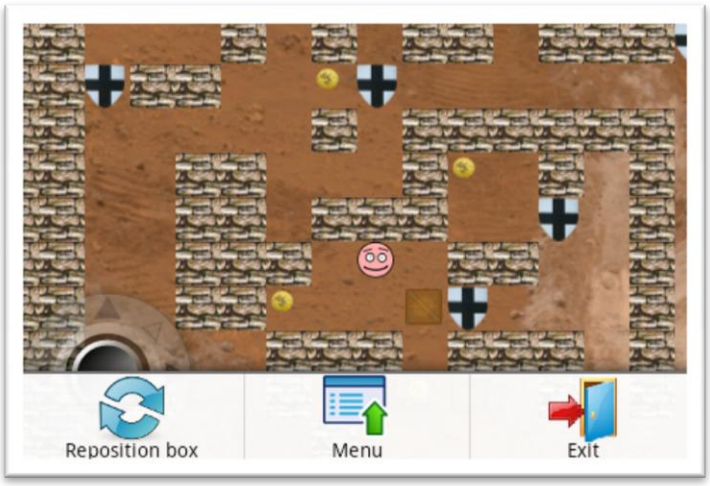


Ilustración 66: Menú laberinto

Desafíos

En los siguientes apartados se van a describir cuales son las posibilidades en cada uno de los desafíos.

Desafío Fill the gap

Si desea realizar un desafío *Fill the gap* debe chocar con su personaje con el escudo blanco y negro con forma de cruz. Automáticamente se iniciará el desafío que estará formado por una pantalla como la que se muestra a continuación.

Ilustración 67: Desafío Fill the gap

En ella aparecerá un título y un enunciado del ejercicio al que le falta una parte, que usted deberá completar pulsando en la respuesta que crea correcta. Una vez haya pulsado en un botón podrá comprobar cuál es el resultado final en la parte inferior de la pantalla, y cuando desee comprobar su respuesta deberá pulsar sobre el botón que aparece en la parte inferior con el texto "comprobar". Una vez haya

pulsado este botón se le mostrará con un tic verde o una cruz roja si su respuesta ha sido correcta o incorrecta y se le sumará a sus estadísticas.

Desafío Matching

Si desea realizar un desafío *Matching* debe chocar con su personaje con el escudo marrón y plata. Automáticamente se iniciará el desafío que estará formado por una pantalla como la que se muestra a continuación.

Ilustración 68: Desafío Matching

En ella aparecerá un título y un enunciado usted deberá pulsar sobre el botón que cree que tiene la respuesta correcta. Una vez haya pulsado el botón se le mostrará con un tic verde o una cruz roja si su respuesta ha sido correcta o incorrecta y se le sumará a sus estadísticas.

Desafío Listening

Si desea realizar un desafío *Listening* debe chocar con su personaje con el escudo formado por dos partes blancas y dos partes negras. Automáticamente se iniciará el desafío que estará formado por una pantalla como la que se muestra a continuación.

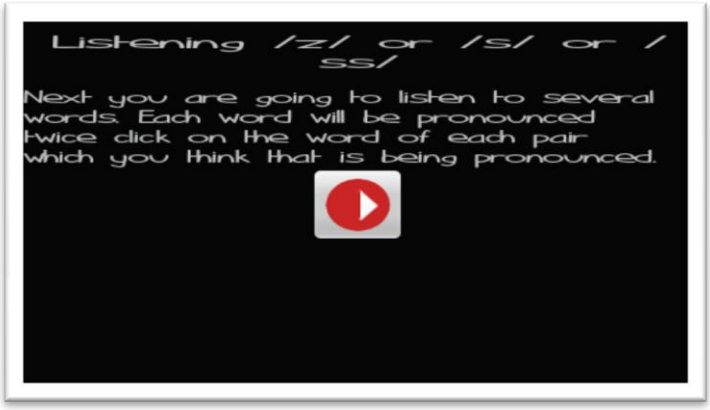


Ilustración 69: Desafío Listening pantalla inicio

En ella aparecerá un título, un enunciado, donde se le informará del objetivo del ejercicio, y un botón para iniciar la reproducción del sonido correspondiente. Una vez haya pulsado este botón se le mostrará una pantalla como la que aparece a continuación.

Ilustración 70: Desafío Listening pantalla sub ejercicio

En ella podrá observar en la parte superior los fallos que puede cometer para no perder el desafío, y se le mostrarán dos botones con la posible respuesta. Pulse el botón que crea que contiene la respuesta correcta y espere hasta que aparezcan las nuevas respuestas. Una vez haya terminado el ejercicio se le mostrará con un tic verde o una cruz roja si sus respuestas han sido correctas o incorrectas y por consiguiente supera el desafío.

Actualización del contenido

Si desea actualizar el contenido didáctico y los escenarios del videojuego únicamente debe pulsar sobre el botón actualizar que aparece en el menú principal. En caso de haber disponible una descarga, esta se iniciará automáticamente y se descargaran los nuevos ficheros que estén alojados en el servidor. Una vez terminada la descarga dispondrá del contenido descargado, tanto en las pantallas de teoría como en los laberintos y ejercicios.

ANEXO IV Manual del videojuego para el profesor

Este manual está dedicado a aquellas personas que deseen actualizar o modificar el contenido didáctico del videojuego.

Actualizar la teoría

Para actualizar la teoría contenida en el videojuego, una vez creado el fichero correspondiente deberá almacenarlo en la carpeta pública de la cuenta que se proporciona a continuación del servidor ***Dropbox***.

- Dirección de correo: **pfcmaze@hotmail.com**
- Contraseña: **pfcme2011**

Para crear un nuevo fichero de Teoría debe saber que las páginas de teoría están estructuradas de la siguiente manera:

- Un fichero de teoría puede contener tantas páginas como sea necesario. Cada página de teoría puede tener tantos usos y formulas como desee pero deber tener obligatoriamente un título.
- Una formula está formada por una o varias *formula-line*. Cada *formula-line* puede contener entre corchetes, [], las palabras que el desarrollador quiera y estas palabras aparecerán en formato negrita cuando el usuario visualice esta página de teoría en el teléfono móvil. Igual que ocurre con las páginas, una página de teoría puede tener tantas formulas como se desee.
- Un uso debe tener al menos una descripción y puede contener tantos ejemplos como se quiera. Una página de teoría puede estar formada por cualquier número de usos.

En el código que aparece en la siguiente página se puede observar un ejemplo de la estructura de una página de teoría.

```
<question>
        <page type = "theory">
         <title> Present continuous is used to talk about the future... </title>
          <formula>
                  <formula-line order="1">[If] + past perfect tense, + [would have] + past
                participle</formula-line>
                 <formula-line order="1">[or]</formula-line>
                 <formula-line order="1">would have + past participle (no comma) + [if] + past
                perfect tense</formula-line>
        </formula>
        <uses>
                <use order="1">
                <use-description> Soy la descripcion del uso 1</use-description>
                        <example order="1">I would have visited you if I had known you were
                        in hospital.</example>
                        <example order="2">I had left home earlier, I wouldn't have missed the
                        bus.</example>
                </use>
                <use order="2">
                        <use-description> Soy la descripcion del uso 2</use-description>
                        <example order="1">I I had slept well last night, I wouldn't be so tired
                        now.</example>
                        <example order="2">If you hadn't quit university, you would have a
                        good job now.</example>
                </use>
        </uses>
 </page>
```

Código 7: Código de ejemplo de una página de teoría

Actualizar los escenarios

Para actualizar los escenarios del videojuego, una vez creado el fichero correspondiente deberá almacenarlo en la carpeta pública de la cuenta que se proporciona a continuación del servidor ***Dropbox***.

- Dirección de correo: **pfcmaze@hotmail.com**
- Contraseña: **pfcme2011**

No debe preocuparse de cuantos escenarios hay en el fichero, ya que el videojuego se encarga de controlar que se cree un nivel por cada uno de ellos y detectar el final del juego en el último nivel.

Para la creación de los escenarios puede realizarlo de dos maneras:

1. Mediante un editor de texto, observe como está formado el fichero antiguo y modifique los nuevos escenarios a su antojo. Los símbolos que representan el laberinto son:

   - # representa un muro en esa posición.
   - 1 representa un escudo de tipo Matching en esa posición.
   - 2 representa un escudo de tipo Fill the gap en esa posición.
   - 3 representa un escudo de tipo Listening en esa posición.
   - - representa que no hay nada en esa posición
   - % representa la posición inicial del protagonista
   - B representa la posición de la bandera
   - C representa la posición inicial de una caja
   - M representa la posición de una moneda

   Esta forma de editar laberintos solo se recomienda para personas con conocimientos técnicos.

2. Utilizando la aplicación desarrollada para creación de escenarios. Para más información sobre el uso de esta aplicación consulte el ANEXO V

En la siguiente página se muestra el formato de un escenario, que está compuesto de un mapa y los nombres de los ficheros de preguntas de ejercicios que están asociados con ese laberinto.

```
<?xml version="1.0" encoding="utf-8"?>
<laberintos>
	<escenario>
		<mapa>
			<fila>####################</fila>
			<fila>#-M#---M#%-#M----1M#</fila>
			<fila>#-2#-------C-----###</fila>
			<fila>#CC---############-#</fila>
			<fila>#---#-#M##M-------M#</fila>
			<fila>#-3##--C-###---B--##</fila>
			<fila>##--#------#-----#-#</fila>
			<fila>#---##########1##--#</fila>
			<fila>#---##---##--------#</fila>
			<fila>#-M-#----1M#######C#</fila>
			<fila>#---#-#--##--------#</fila>
			<fila>#----C-#####-------#</fila>
			<fila>#---#---1--C----####</fila>
			<fila>#-----###M##----1-M#</fila>
			<fila>####################</fila>
		</mapa>
		<nombreFicheroMatching>matching.xml</nombreFicheroMatching>
		<nombreFicheroFill>fill.xml</nombreFicheroFill>
		<nombreFicheroListening>listening.xml</nombreFicheroListening>
	</escenario>
</laberintos>
```

Código 8: Código de ejemplo de un escenario

Actualizar los ejercicios de Matching

Si desea añadir nuevos ejercicios de esta categoría, únicamente debe alojar el fichero correspondiente en la carpeta pública de la cuenta que se proporciona a continuación del servidor ***Dropbox***.

- Dirección de correo: **pfcmaze@hotmail.com**
- Contraseña: **pfcme2011**

Recuerde que este ejercicio estará relacionado únicamente con aquel laberinto que almacene su nombre en el ejercicio asociado de dicha categoría.

El nuevo fichero de ejercicios puede contener tantos ejercicios de *Matching* como desee pero cada página de ejercicios debe contener obligatoriamente un título, un enunciado, tres respuestas y marcar cual es la respuesta correcta. A continuación se muestra, mediante un ejemplo, la estructura de los ficheros de ejercicios de tipo *Matching*.

```
<question>
   <page type = "matching">
            <Titulo>Matching</Titulo>
            <sentence>Mary's diamon ring is enormous!</sentence>
            <choices>
                        <choice rightanswer="Y"> It must have cost a fortune
</choice>
                        <choice> It should have cost a fortune </choice>
                        <choice> It could have cost a fortune </choice>
            </choices>
 </page>
</question>
```

Código 9: Código de ejemplo ejercicio Matching

No debe preocuparse por cuantos ejercicios contiene el fichero ya que la aplicación está desarrollada para que no se produzca ninguna situación de error. Aunque en el caso de que no tenga suficientes ejercicios de este tipo como escudos haya, se repetirán las preguntas.

Actualizar los ejercicios de Fill the gap

Para actualizar los ejercicios de esta categoría, debe alojar el fichero correspondiente en la carpeta pública de la cuenta que se proporciona a continuación del servidor ***Dropbox***.

- Dirección de correo: **pfcmaze@hotmail.com**
- Contraseña: **pfcme2011**

Recuerde que este ejercicio estará relacionado únicamente con aquel laberinto que almacene su nombre en el ejercicio asociado de dicha categoría.

El nuevo fichero de ejercicios puede contener tantos ejercicios de Fill the gap como desee pero cada página de ejercicios debe contener obligatoriamente un título, un enunciado, tres respuesta, tres respuestas completas y marcar cual es la correcta. A continuación se muestra, mediante un ejemplo, la estructura de los ficheros de ejercicios de tipo *Fill the gap*.

```
<question>
   <page type = "fill">
          <Titulo>Fill the gap</Titulo>
          <sentence>Mary's diamon ring is enormous! It_____ cost a fortune</sentence>
          <choices>
                 <choice rightanswer="Y">
                        <option>must have</option>
                        <full_sentence> Mary's diamon ring is enormous! It
                        must have cost a fortune </full_sentence>
                 </choice>
                 <choice>
                        <option>can't had</option>
                        <full_sentence>  Mary's diamon ring is enormous! It
                        can't had cost a fortune </full_sentence>
                 </choice>
                 <choice>
                        <option>might has</option>
                        <full_sentence>  Mary's diamon ring is enormous! It
                        might has cost a fortune </full_sentence>
                 </choice>
          </choices>
   </page>
</question>
```

Código 10: Código de ejemplo ejercicio Fill the gap

No debe preocuparse por cuantos ejercicios contiene el fichero ya que la aplicación está desarrollada para que no se produzca ninguna situación de error. Aunque en el caso de que no tenga suficientes ejercicios de este tipo como escudos haya, se repetirán las preguntas.

Actualizar los ejercicios de Listening

Si desea añadir nuevos ejercicios de esta categoría, debe alojar el fichero correspondiente a los ejercicios y los archivos de audio asociados con el nuevo fichero, en la carpeta pública de la cuenta que se proporciona a continuación del servidor ***Dropbox***.

- Dirección de correo: **pfcmaze@hotmail.com**
- Contraseña: **pfcme2011**

El nuevo fichero de ejercicios puede contener tantos ejercicios de Listening como desee pero cada página de ejercicios debe contener obligatoriamente un título, una introducción, el nombre del archivo de sonido con el que está asociado y la respuesta a cada uno de los sub ejercicios. A continuación se muestra, mediante un ejemplo, la estructura de los ficheros de ejercicios de tipo *Listening*.

```
<question>
  <page type="listening">
        <Titulo>Listening /z/ or /s/ or /ss/</Titulo>
        <Introduccion>Next you are going to listen to several words. Each word will be
pronounced twice click on the word of each pair which you think that is being
pronounced.</Introduccion>

        <NombreArchivo>audio.mp3</NombreArchivo> // el archivo contiene todas las palabras
(es asÃ como se hace en el listening)
          <audio-question>
                <choice>Buzz</choice>
                <choice rightanswer="Y">Bus</choice>
          </audio-question>

          <audio-question>
                <choice rightanswer="Y">Prize</choice>
                <choice>Price</choice>
          </audio-question>
  </page>
</question>
```

Código 11: Código de ejemplo ejercicio Listening

No debe preocuparse por cuantos ejercicios contiene el fichero ya que la aplicación está desarrollada para que no se produzca ninguna situación de error. Así mismo tampoco debe preocuparse de cuantos subejercicios está compuesto cada archivo de audio, el sistema está desarrollado para admitir cualquier número de subejercicios y mostrárselos al usuario.

ANEXO V Manual de uso de la aplicación de edición de escenarios

Este manual está dedicado a aquellas personas que deseen crear nuevos escenarios para el videojuego "*Maz-E-nglish*", mediante la aplicación Java desarrollada, para facilitar la tarea al profesor.

Para ejecutar esta aplicación es necesario disponer de una máquina virtual Java instalada en el ordenador, y únicamente deberá realizar doble clic sobre el icono de la aplicación. Inmediatamente se le mostrará una aplicación como la que se observa en la siguiente figura.

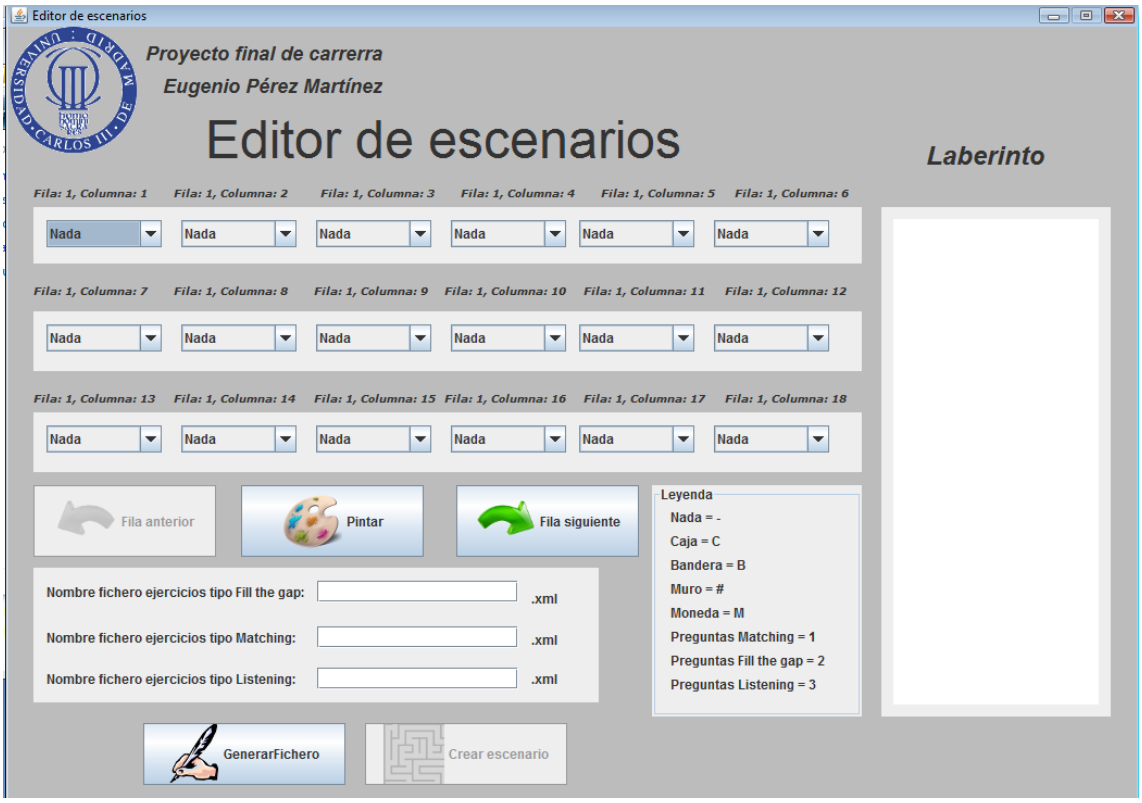


Ilustración 71: Editor de escenarios

La aplicación está compuesta por una serie de desplegables donde se indica la posición de fila y columna que corresponde al objeto seleccionado.

En la parte derecha se mostrará el escenario que se va creando cada vez que se pulsa en el botón de pintar. Justo al lado puede observar la leyenda del gráfico.

Con esta aplicación puede seleccionar mediante los desplegables el objeto que desea colocar en la posición indicad. Las opciones que puede elegir son:

- Nada
- Protagonista

- Bandera
- Moneda
- Caja
- Muro
- Pregunta de tipo *Fill the gap*
- Pregunta de tipo *Listening*
- Pregunta de tipo *Matching*

Una vez haya seleccionado todas las posiciones de una fila pulse en el botón pulsar para pintar la fila en el laberinto que le sale en la parte derecha de la pantalla. **Atención**: Automáticamente se le rellenarán los extremos del laberinto, no debe preocuparse por ellos. Una vez haya pintado la fila a su gusto pulse en el botón fila siguiente y repita la misma operación. Deberá realizar tantas veces esta operación hasta que termine de dibujar el laberinto, momento en el que se desbloqueará el botón de crear escenario.

Si desea modificar una fila que ya ha pintado, vuelva a la fila errónea pulsando el botón de fila anterior, realice las modificaciones oportunas y pulse en el botón pintar.

Una vez haya generado todo el laberinto, introduzca los nombres de los archivos correspondientes a los ejercicios que irán asociados con este laberinto y pulse el botón crear escenario. Si no introduce alguno de los nombres, la aplicación le mostrará un mensaje de error y no le permitirá crear el escenario. En la ilustración que aparece a continuación se puede observar un laberinto que se encuentra ya creado.

Ilustración 72: Editor de escenarios con laberinto

Después de crear el escenario, podrá crear tantos escenarios como usted desee, una vez haya realizado todos los escenarios pulse sobre el botón generar fichero. La aplicación automáticamente le generará el fichero correspondiente con todos los escenarios y le advertirá mediante un mensaje que los escenarios han sido creados. El nuevo fichero se almacenará en la misma carpeta donde esté ubicado el ejecutable de la aplicación.

En caso de intentar pintar alguna fila sin haber pintado las filas anteriores, la aplicación le mostrará un mensaje de error advirtiéndole que debe pintar las filas precedentes para poder pintar esa fila.

En la *ilustración 73* se puede observar el mensaje que lanza la aplicación cuando se genera el fichero.

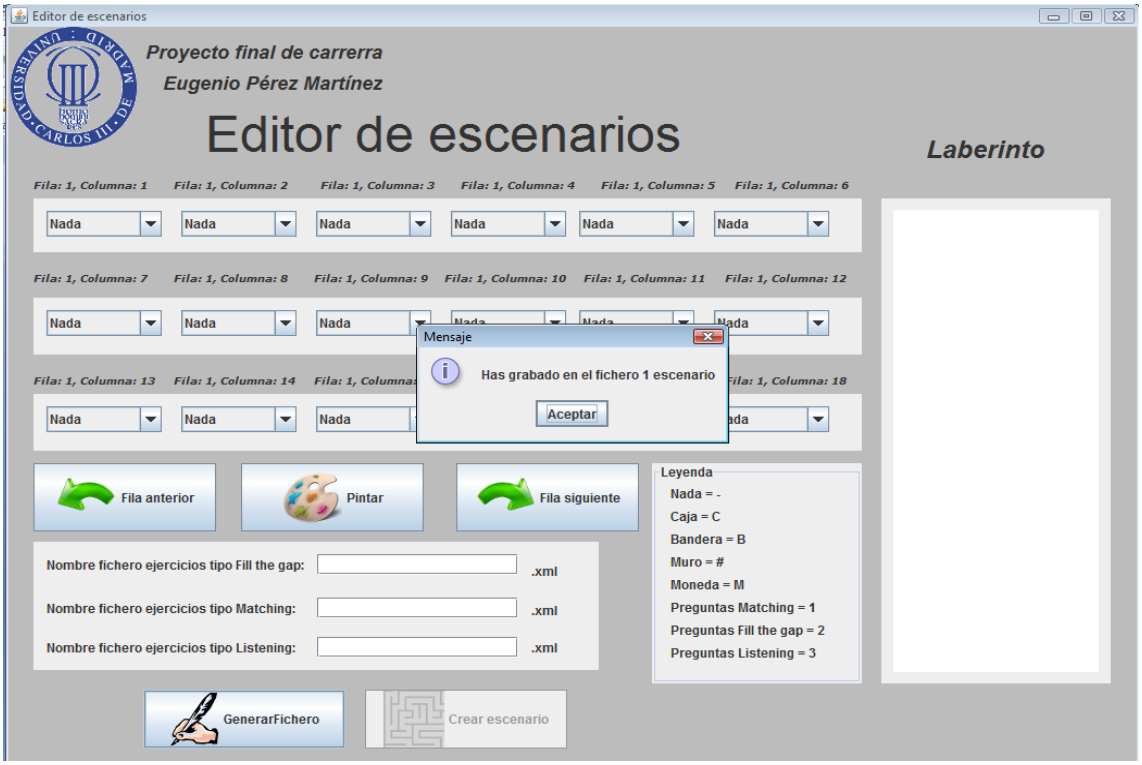


Ilustración 73: Aplicación editor de escenarios con aviso