Universidad de Jaén
Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera:

DESARROLLO DE UN JUEGO EDUCATIVO PARA DISPOSITIVOS MÓVILES

Alumno:
Luis Barragán Sánchez

Tutores:
Dr. D. Luis Martínez López
Dr. Dª. Macarena Espinilla Estévez

Departamento de Informática

Área de Conocimiento: Lenguajes y Sistemas Informáticos

Septiembre 2009

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dr. D. Luis Martínez López y Dra Dª. Macarena Espinilla Estévez, tutores del Proyecto Fin de Carrera titulado: Desarrollo de un juego educativo para dispositivos móviles, que presenta D. Luis Barragán Sánchez, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre 2009

El Alumno: Los Tutores:

D. Luis Barragán Sánchez

Dr. D. Luis Martínez López
Dra. Dª. Macarena Espinilla Estévez

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que han hecho posible que, de una manera u otra, este proyecto viera la luz, con especial dedicación a:

Mis padres, porque gracias a su sacrificio y esfuerzo me han permitido llegar hasta aquí y por apoyar siempre mis decisiones.
Mis hermanas, por sus consejos y creer en mí.
Mis amigos/as, por ser como son y en especial Ángel Lopéz, que es como un hermano, y los que me ha ayudado y animado durante su realización.
Mis tutores, Luís Martínez y Macarena Espinilla, por proponerme este proyecto, por su gran paciencia y profesionalidad y por tratarme siempre tan bien.

Gracias a todos

ÍNDICE GENERAL

CAPÍTULO 1

PREFACIO

1.1 Introducción

El auge y desarrollo de las Tecnologías de la Información y de las Comunicaciones (TIC), en especial de las tecnologías inalámbricas, representadas principalmente por las redes Wi-Fi, Bluetooth y telefonía móvil (celular) y el amplio despliegue que ha tenido en los últimos años los dispositivos móviles, ha mostrado la potencialidad que tiene este nuevo paradigma de computación, conocido como Computación Móvil, en todos los ámbitos de nuestra sociedad. La Computación Móvil permite independientemente del lugar y del tiempo, realizar diferentes actividades de procesamiento en los dispositivos móviles conectados a Internet o a redes corporativas a través de las diferentes opciones de comunicación [1].

Concretamente, el teléfono móvil cuya principal característica es su portabilidad y que permite comunicarse desde casi cualquier lugar, es el que ha experimentado una mayor evolución, debido a la gran aceptación que ha tenido. Prueba de ello son los más de 4.100.000.000 de móviles en uso en todo el mundo [2]. Su gran atractivo debido a que es un dispositivo de pequeñas dimensiones que se puede llevar a todas partes, ha propiciado su gran desarrollo, al que se le ha ido incorporando funciones, que hasta no hace mucho tiempo parecían futuristas, como son cámara fotográfica, vídeo, vídeo llamada, acceso a Internet, reproducción de música, mensajes cortos (SMS), mensajes multimedia (MMS), ejecución de juegos y otras aplicaciones típicas de oficina, como editor de texto, GPS, etc. [1,3].

Todo este desarrollo tecnológico ha generado grandes expectativas en el sistema educativo, el cual ha ido introduciendo y aplicando estas nuevas tecnologías en el proceso de aprendizaje, desde distintas perspectivas. Esto ha supuesto la creación de nuevos espacios o marcos educativos, con nuevas reglas, que exige nuevos roles, pero en definitiva, marcos en los que es posible aprender y que viene a complementar los anteriores sistemas y métodos educativos. La adopción generalizada de Internet como plataforma de distribución es la que ha facilitado la creación, difusión y puesta en marcha de estos nuevos marcos educativos. Así pues, este campo tan amplio y diverso del uso de Internet y las TIC en la educación es lo que se está conociendo bajo los términos de teleformación, formación en línea, formación virtual y en especial e-learning [4,5,7].

En nuestro caso, las metodologías de enseñanza y aprendizaje **e-Learning** [4,5,6,7,8,9] y **m-learning** [9,10,11] son las que establecen la base de este proyecto. Entendemos por e-learning aquella metodología que lleva a cabo la formación y aprendizaje del alumnado a través del uso de la tecnología de redes, Internet y las TIC en general, posibilitando el acceso a recursos y contenidos de manera inmediata y por m-learning a la metodología de aprendizaje que integra el uso de dispositivos móviles que dispongan de conectividad inalámbrica, como teléfonos móviles, PDAs, etc., con el fin de producir experiencias educativas en cualquier lugar o momento. Nuestro mayor interés se centra precisamente en m-learning, ya que nos hemos propuesto desarrollar un prototipo de software de autoentrenamiento y autoevaluación basado en m-learning, para dispositivos móviles, que permita al usuario ejercitarse en una serie de materias y autoevaluarse.

Por alcanzar dicho objetivo, primero profundizaremos en el conocimiento de estas metodologías y estudiaremos diferentes procesos de aprendizaje, que faciliten el autoentrenamiento y además la autoevaluación, ya que con la adopción de estas metodologías, surge la necesidad del alumno de evaluarse para así conocer el grado de adquisición de conocimientos conseguido [14]. También hay que destacar que el prototipo software de autoentrenamiento y autoevaluación se basará en pruebas objetivas o tipo test, que consisten en la resolución de un cuestionario de respuestas cerradas, con un número variable a elegir de preguntas, con múltiples respuestas. Las ventajas que nos aporta este método de evaluación es la amplitud de temas que puede cubrir, la objetividad de la calificación, la rapidez de la misma y la poca duración que puede suponer la realización del examen [13].

1.2 Propósito y objetivos

El propósito de este proyecto fin de carrera es diseñar y desarrollar un prototipo software que implemente un juego educativo basado en preguntas tipo test para dispositivos móviles utilizando el lenguaje de programación y la tecnología J2ME (Java 2 Micro Edition). Dicho software permitirá al usuario autoentrenarse en las temáticas registradas en el prototipo.

Ahora bien, para conseguir nuestro propósito, hay que alcanzar una serie de objetivos, que presentamos a continuación:

1. Búsqueda y revisión de bibliografía adecuada.

2. Estudio y repercusión de los nuevos métodos de aprendizaje e-learning y m-learning, basados en la utilización y aplicación de las nuevas tecnologías de la información y comunicación con el objetivo de mejorar el proceso de autoevaluación y autoentrenamiento.

3. Estudio de las tecnologías y herramientas disponibles para los dispositivos móviles.

4. Diseñar un prototipo software para dispositivos móviles que contemple un proceso de autoentrenamiento basado en un juego de preguntas tipo test.

5. Implementar dicho prototipo ajustándose a las especificaciones contempladas en el diseño.

Es importante que el prototipo que se va a desarrollar cumpla con unos requisitos mínimos, que se consideran cruciales para conseguir nuestro propósito, entre todos ellos podemos destacar:

- El prototipo debe presentar una interfaz amigable y fácil de usar.

- El prototipo presentará al usuario unos cuestionarios, basados en preguntas tipo test de materias elegidas por el usuario, de las disponibles en dicho prototipo.

- El prototipo debe de proporcionar al usuario capacidad de realizar procesos de autoentrenamiento y autoevalución durante la ejecución del mismo.

- El prototipo debe ser compatible con el mayor número posible de dispositivos móviles existentes.

1.3 Estructura de la memoria

A continuación, se hará una breve introducción de los diferentes capítulos en los que se estructura este proyecto y los contenidos planteados en los mismos.

En el segundo capítulo se realiza una revisión de los conceptos E-Learning y M-Learning, que establecen las bases y fundamentos de este proyecto, junto con los procesos de aprendizaje que propician la autoevaluación y autoentrenamiento y los diferentes mecanismos para realizar estos procesos.

El tercer capítulo se centra en el lenguaje de programación y la tecnología J2ME, describiendo a su origen, evolución, características, amplia integración y aplicación en varios tipos de dispositivos electrónicos, que finalmente se utilizará para implementar nuestro prototipo.

En el cuarto capítulo describiremos el proyecto en sí. Veremos que se trata de un proyecto de desarrollo software y mostraremos en detalle las distintas actividades de la Ingeniería del Software realizadas en su desarrollo.

Finalmente, en el capítulo quinto, se expondrán una serie de conclusiones generales derivadas de este proyecto y por último, se adjuntará un manual de usuario del prototipo desarrollado.

CAPÍTULO 2

ANTECEDENTES

En este capítulo vamos a revisar las metodologías básicas sobre las que se desarrolla nuestro proyecto, centrándonos en modelos de aprendizaje e-learning y m-learning, que incorporan las Tecnologías de la Información y Comunicación (TIC) en los procesos de enseñanza. Estas metodologías, como veremos, flexibilizan los procesos de enseñanza clásica y requieren del uso de herramientas y nuevos métodos de enseñanza.

Haremos una introducción de cada una de las metodologías, junto con sus respectivas definiciones y pasaremos a describir y detallar sus características, herramientas y aspectos que se estiman relevantes de su conocimiento.

2.1 E-LEARNING

La incorporación de las Tecnologías de la Información y de la Comunicación al área de la enseñanza ha producido la aparición de un nuevo modelo de concepción de la misma unida al uso de Internet, el e-Learning. Este nuevo concepto hace uso de las nuevas tecnologías, que están a nuestro alcance, para flexibilizar las metodologías tradicionales de la enseñanza y así poder atender a la creciente necesidad de formación que demanda la actual Sociedad [14].

El e-learning, desde su comienzo, ha contemplado varias acepciones, entre las que engloba el concepto de manera amplia:

- E-Learning es un medio electrónico para el aprendizaje a distancia o virtual, completamente autónomo, donde interactúan profesores y alumnos y donde el alumno pasa a ser el centro de la formación, al tener que autogestionar su aprendizaje, con ayuda de tutores y compañeros. Constituye una propuesta de formación que contempla su implementación predominantemente mediante plataformas, cuyo medio es Internet, haciendo uso de los servicios y herramientas que esta tecnología provee, junto también con Intranets, CD-ROM, producciones multimedia, entre otros dispositivos electrónicos [5].

- Entendemos por e-learning aquella formación y aprendizaje facilitado a través de la tecnología de redes, internet y las TIC en general, posibilitando el acceso a recursos y contenidos de manera inmediata. La metodología e-learning se presenta como una nueva estrategia formativa, compatible y complementaria con otros modelos formativos más tradicionales que deben ir evolucionando por los constantes cambios que se han producido y se siguen produciendo en los entornos sociales y tecnológicos [9].

El e-learning no es una metodología que venga a sustituir a modelos de formación ya existentes, si no que, como señalábamos anteriormente, constituye una metodología complementaria que ofrece la oportunidad de mejorar la eficacia global de la enseñanza y del aprendizaje. Este hecho se pone de manifiesto en la aplicación de las teorías del aprendizaje en la acción formativa del e-learning, donde se está poniendo en práctica una estrategia mixta entre las perspectivas del aprendizaje conductista y constructivista [9,25], siendo estas:

Teoría conductista: El aprendizaje se refiere a un cambio sensible de conducta que no hace énfasis en los procesos cognitivos, sino, en el resultado.

Teoría constructivista: El aprendizaje se concibe desde los procesos mentales que desarrolla el sujeto.

Estas teorías del aprendizaje han contribuido al diseño de materiales online y acciones formativas, donde el conductivismo permite elaborar contenidos estructurados que apuntan a objetivos específicos del conocimiento y el constructivismo basa el aprendizaje en las interacciones que desarrollen los alumnos entre ellos y con el personal docente.

2.1.1 Caracterización del e-learning

Entre las características distintivas que posicionan al e-learning como una metodología que mejora la calidad del aprendizaje, podemos destacar comparándola con la enseñanza presencial tradicional [4,14]:

- **Es dinámico.** Los contenidos presentes en el sistema se actualizan regularmente para atender a las necesidades educativas que en cada momento un tutor o grupo de expertos consideren necesarios para

atender a la constante evolución de la tecnología y/o las necesidades específicas de empresas. Con los sistemas tradicionales se necesita mucho tiempo y recursos para poder actualizar toda la documentación necesaria además de los costes de distribución que conllevan.

- **Opera en tiempo real.** Podemos encontrar en el sistema e-Learning lo que necesitamos en el momento de necesitarlo. Con los sistemas tradicionales no podemos atender a una necesidad puntual de forma rápida, es necesario un tiempo para poder recopilar la información y hacerla llegar al interesado.

- **Es individual.** El sistema e-Learning atiende a las necesidades particulares, de conocimientos o tiempo de aprendizaje, que pueda tener o seleccionar cada alumno en cada momento frente al sistema de enseñanza tradicional, que se basa en clases magistrales a una serie de alumnos, y no atienden a las necesidades puntuales de cada uno.

- **Proporciona distintos modos de obtener conocimiento.** En los sistemas e-Learning, al hacer uso de las TIC, disponemos de distintas tecnologías para poder presentar el conocimiento en el medio que resulte más apropiado o más cómodo para cada alumno, según sus necesidades. En los modelos de enseñanza tradicionales también podemos presentar el conocimiento utilizando las TIC pero se necesita hacer una inversión mayor que utilizando un sistema e-Learning.

Una vez expuestas las características que presenta el e-learning, se puede sintetizar los principales objetivos que desea conseguir esta metodología de enseñanza y aprendizaje, siendo los siguientes:

- Mejorar la calidad del aprendizaje, con la innovación en las herramientas y materiales didácticos, empleando un profesorado cualificado como guía del aprendizaje y aumentando la motivación del alumnado.

- Favorecer el acceso a la formación a todas las personas, cualquiera que sea su situación geográfica, ocupación, horarios, etc.

Aún así, el e-learning, como ocurre con otras metodologías, presenta una serie de ventajas e inconvenientes, que revisaremos a continuación. Entre las ventajas que presentan, resaltamos las siguientes:

- Pone a disposición de los alumnos un amplio volumen de información.

- Facilita la actualización de la información y de los contenidos.

- Flexibiliza la información, independientemente del espacio y el tiempo en el cual se encuentren docentes y estudiantes.

- Favorece la interactividad en diferentes ámbitos: con la información, con el profesorado y entre el alumnado.

A pesar de todas las ventajas que el e-learning puede aportar, es necesario tener en cuenta los inconvenientes que presenta, antes de adoptar esta metodología, entre los que se señalan [4]:

- Requiere más inversión de tiempo por parte del profesorado y más trabajo que la convencional.

- Se encuentra con la resistencia al cambio del sistema tradicional.

- Precisa unas mínimas competencias tecnológicas por parte del profesorado y de estudiantes.

- Requiere que los estudiantes tengan habilidades para el aprendizaje autónomo.

- Puede disminuir la calidad de la formación si no se da una ratio profesor-alumno adecuada y si no se cuidan los contenidos y objetivos a seguir.

- Requiere que los profesores e instructores tengan un adecuado entrenamiento y habilidades para dar soporte correctamente al alumnado.

Estos inconvenientes se irán supliendo a la vez que se adquiera una mayor experiencia en su utilización. Por ello, es fundamental la labor de formación y capacitación a sus docentes por parte de las instituciones que deciden incorporar este tipo de formación, así como explicar desde el principio al alumnado cual es su papel en este nuevo escenario.

2.1.2 Modelo pedagógico en e-learning

Todos los elementos que conforman un programa formativo en el e-learning, son importantes (alumnado, expertos, contenidos, TIC...), deben estar enmarcados dentro de un modelo pedagógico que los sustente, haciendo referencia a cómo se entiende el proceso de enseñanza-aprendizaje, métodos didácticos, estrategias de aprendizaje, herramientas, figuras en el proceso enseñanza–aprendizaje, evaluación, etc. Los modelos pedagógicos se sustentan en teorías y principios de aprendizaje, que son los que perfilan los aspectos organizativos y didácticos de un programa formativo.

Las estrategias y métodos pedagógicos vinculados a un modelo e-learning contextualizado en el ámbito del aprendizaje permanente deben contemplar los siguientes principios [9]:

- **Aprender a aprender**: Situando al alumno como el protagonista activo en el proceso de enseñanza-aprendizaje.

- **Aprendizaje colaborativo**. Fomentando la participación activa entre alumnos a través de distintas actividades dentro de las acciones formativas, así como con la creación de comunidades de aprendizaje.

- **Compatibilidad y conexión con la realidad profesional**. Ofreciendo programas de utilidad para alumnos (profesionales) que posibiliten una puesta en práctica inmediata de los aprendizajes en su realidad profesional.

- **Uso de las TIC para el aprendizaje**. Potenciando nuevas formas de desarrollo profesional y facilitando el acceso a las oportunidades de aprendizaje permanente.

Las estrategias y métodos pedagógicos vinculados a un modelo e-learning, son aplicables a un modelo m-learning, al igual que los principios anteriormente contemplados, pero teniendo en cuenta las peculiaridades que presenta este modelo, ya que está orientado al aprendizaje con el uso de dispositivos móviles.

Siguiendo los principios que deben seguir las estrategias y métodos pedagógicos para llevar a cabo un adecuado proceso de enseñanza y aprendizaje, al alumnado se le sitúa en el centro de este proceso, y se ponen en marcha estrategias que guíen y orienten su aprendizaje, como es la tutoría, donde ocupa un lugar muy importante en el e-learning, ya que el alumno, generalmente, realiza un proceso de aprendizaje autónomo, que facilita el autoentrenamiento y autoevalución.

De esta manera, teniendo como criterio el nivel de autonomía del alumnado en su proceso de aprendizaje, podemos distinguir entre aprendizaje autónomo y aprendizaje guiado [9]:

- **Aprendizaje autónomo**. Se caracteriza por dar la posibilidad al alumnado de aprender de manera autónoma en base a unas guías y criterios pedagógicos definidos para cada curso. Cobra un especial protagonismo el autoentrenamiento y la autoevaluación que desempeña el propio alumno. En este tipo de aprendizaje encontramos al dinamizador de programas como figura clave que orienta al alumnado y que se ocupa de garantizar que se cumpla la planificación prevista. La autoevaluación del proceso se irá realizando a través de cuestionarios de corrección automática (figura 2.1).

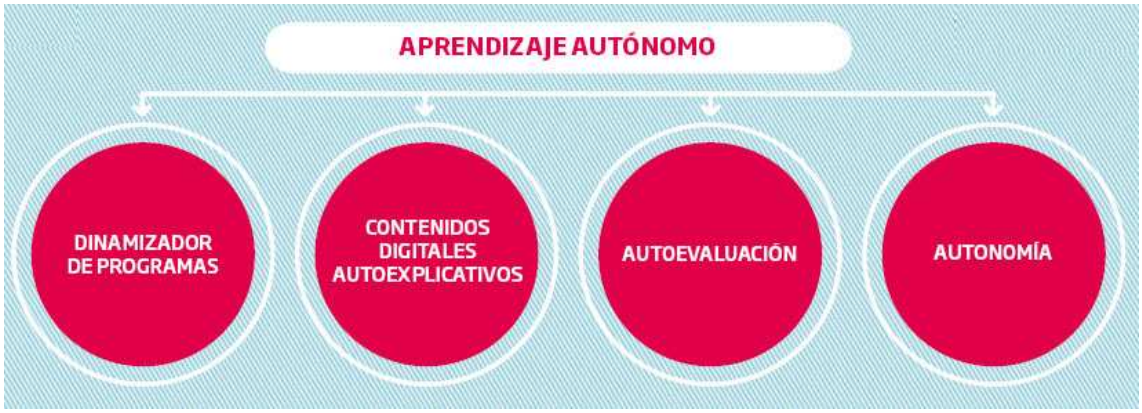


Figura 2.1 Proceso de Aprendizaje autónomo.

- **Aprendizaje guiado**. Vendrá determinado por un seguimiento tutorizado por un mediador, que será una figura de referencia para el alumnado y lo acompañará en su proceso de aprendizaje. El aprendizaje guiado vendrá apoyado por materiales didácticos que requieran de orientaciones didácticas aportadas por el mediador. La evaluación de los aprendizajes se realizará de forma continua a través de los criterios de aprovechamiento establecidos para cada curso de esta categoría (figura 2.2).

Figura 2.2 Proceso de Aprendizaje guiado.

Para la realización de nuestro proyecto, nos decantamos por el tipo de aprendizaje autónomo, ya que se ajusta a nuestro propósito de desarrollar un prototipo software que permita al usuario autoentrenarse y autoevaluarse en las temáticas registradas en el mismo.

2.1.3 Herramientas y plataformas para e-learning

En el e-learning, las herramientas que se pongan en práctica han de cumplir varios requisitos para otorgar calidad al proceso de enseñanza y aprendizaje, entre otras muchas características, hablamos de flexibilidad y capacidad para adaptarse al cambio.

A continuación vamos a ver a las herramientas más comunes en la formación e-learning, que la mayoría se utilizan dentro de plataformas de teleformación o a través de internet. De todos modos, estas herramientas pueden usarse en combinación con otras metodologías. Pueden clasificarse en tres grandes bloques [9]:

- **Herramientas didácticas**:
  - Foro.
  - Mensajería electrónica.
  - Wikis y Blogs.
  - Glosario interactivo.
  - Contenidos digitales.
  - Contenidos adicionales (multimedia, abiertos, enlaces, etc).
  - Zona de ficheros / zona de descarga.
  - Chat.
  - Clase virtual.
  - Videoconferencia.
  - Talleres de entrenamiento (presencial).

- **Herramientas de evaluación**:
  - Ejercicios y Tareas.
  - Exámenes.

- **Herramientas de gestión docente**:
  - Calendario.
  - Tablón de anuncios.
  - Envío SMS.

Inicialmente, los procesos e-learning se realizaban a través de herramientas aisladas, como hemos comentado. Posteriormente, se produjo una evolución, donde estas herramientas se integran en lo que se conoce como “plataformas”. Las plataformas e-Learning son aplicaciones informáticas que permiten gestionar acciones formativas a través de Internet (crear cursos, dar de alta usuarios, usar herramientas de comunicación, etc.). Mediante una clave el usuario accede a un espacio privado en el que se llevan a cabo los procesos de enseñanza-aprendizaje [9,25].

Las principales funcionalidades que presentan las plataformas son:

- Autenticación al sistema.
- Generación de contenidos.
- Visualización de contenidos.
- Diferentes medios de comunicación con el profesor/tutor.
- Realización de actividades como tareas, trabajos en grupo.
- Reporte de las actividades realizadas por el alumno, etc.

- Herramientas de evaluación.

En función a qué criterio atendamos, las plataformas se pueden clasificar de múltiples formas. En nuestro caso, contemplaremos según su licencia de distribución y su funcionalidad [26].

Las plataformas, en función de su licencia de distribución, es decir, si para su uso hace falta adquirir una licencia del producto o no:

Plataformas Comerciales, las que para su uso hay que abonar una cuota a una empresa, ya sea la que desarrolló el sistema o la que lo distribuye. Entre estas las más conocidas están: Blackboad, WebCT, QSMedia, Saba, etc. Son sistemas generalmente robustos, y bastante documentados con diversas funcionalidades que pueden expandirse de acuerdo a las necesidades y presupuesto del proyecto.

Plataformas de uso libre, surgidas como una alternativa para economizar un proyecto de formación en línea, las herramientas “Open Source” como también se les llaman son generalmente desarrolladas por instituciones educativas o por personas que están vinculadas al sector educativo. Es amplia la gama de funcionalidades que traen cada una de ellas, hay algunas que pueden equipararse a las comerciales mientras que otras solo cuentan con funcionalidades básicas. Entre las más usadas están: PhiTone.LCMS, Atutor, Dokeos, Claroline, dotLRN, Moodle, etc.

Por funcionalidad:

CMS:Content Management System ó Sistema Gestor de Contenidos es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos, principalmente en páginas web, por parte de los participantes. Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior (directorio) que permite que estos contenidos sean visibles a todo el público (los aprueba).

El sistema CMS es de uso más básico, empleado para proyectos pequeños en los que se necesite generar el contenido dentro del sistema.

Dentro de las herramientas de comunicación se pueden encontrar los foros, correo electrónico, chats. Como ejemplo podemos mencionar: PHPNuke, Drupal, Mambo, Content Management Server, CoreMedia CMS, etc.

LMS: Learning Management System o Sistema Gestor del Aprendizaje son aplicaciones software instaladas en un servidor, que se emplean para administrar, distribuir y controlar las actividades de formación no presencial de una institución u organización. Las principales funciones de un LMS son:

- Gestión de usuarios.
- Gestión de recursos, así como de materiales y actividades de formación.
- Administrar el acceso, control y seguimiento del proceso de aprendizaje.
- Realizar evaluaciones.
- Generar informes.
- Gestionar servicios de comunicación como foros de discusión, videoconferencias, entre otros.

Un LMS generalmente no incluye posibilidades de autoría (crear sus propios contenidos), pero se focaliza en gestionar contenidos creados por fuentes diferentes. La labor de crear los contenidos para los cursos se desarrolla mediante un LCMS (Learning Content Management Systems), sistema de gestión de contenidos (CMS) para el aprendizaje.

Cuenta con la mayoría de las herramientas de comunicación y seguimiento de actividades de los usuarios. Algún ejemplo de este tipo de plataformas son: WebCT, Moodle, ATutor.

LCMS: Learning Content Management System (Sistema de gestión de contenidos de aprendizaje) integra las herramientas y utilidades de los sistemas CMS y LMS, lo que le proporciona una mayor robustez y funcionalidad. El LCMS se utiliza para crear y manejar el contenido de una parte de un programa de educación, por ejemplo un curso. Normalmente se crean partes de contenido en forma de módulos que se pueden personalizar, manejar, y que se pueden usar en diferentes ocasiones (cursos). Generalmente es un LMS al que se le ha agregado el módulo de crear contenido dentro de él, como ocurre con los sistemas CMS. Marcas comerciales como: Blackboard, Saba, lo integran de esta manera. Dependiendo de la naturaleza del sistema e-Learning que se quiera implantar se pueden seleccionar varios tipos de

plataforma, aunque en muchos de los casos esta decisión se ve influida por los costos.

2.2 M-LEARNING

Actualmente, estamos inmersos en una sociedad en la que los desarrollos tecnológicos modifican nuestros hábitos y como es previsible, la forma de cómo aprendemos también se está viendo influenciada por todas estas innovaciones. Si las personas cambiamos, los métodos y formas de enseñanza-aprendizaje han de evolucionar de forma paralela. La aparición del e-learning y del m-learning responde a esta situación, aunque el m-learning lo hace desde el punto de vista de la máxima portabilidad, interactividad y conectividad que brindan los nuevos dispositivos móviles [27].

Al igual que ocurre con el e-learning, podemos encontrar diferentes definiciones de m-learning (mobile learning), entre las que destacamos:

M-learning es la metodología de enseñanza con la posibilidad de aprender a través del uso de dispositivos móviles. Se trata de la integración del e-learning con estos dispositivos, que proporcionan una vía de comunicación (teléfonos móviles, agendas electrónicas, reproductores mp3, etc.) con el fin de producir experiencias educativas en cualquier lugar o momento [9].

M-learning es un nuevo enfoque para llevar el e-learning a los dispositivos móviles comunes que usamos a diario [10].

M_learing es e-learning independiente del lugar del tiempo y del espacio[11].

Los contenidos que se transmiten bajo esta metodología, pueden ser consultados con independencia del lugar, al no precisarse conexión física. En cuanto al tiempo, no requiere ningún momento concreto para realizarse el aprendizaje. Esta metodología reduce aún más, si cabe, las pocas limitaciones que poseen los sistemas de aprendizaje a través de la red.

Esta metodología responde a unos procesos educativos novedosos que atienden a demandas urgentes de aprendizaje –formación just in time- además

de poder ubicarse en escenarios de aprendizaje móviles, lo que ofrece una gran interactividad.

Ejemplos muy concretos de este tipo de formación suelen ser los realizados a través de dispositivos PDA en entornos rurales o de difícil acceso (formación para las ONG), así como los utilizados por profesionales que tienen una alta movilidad en su trabajo (personal de urgencias sanitarias). Pero también puede usarse para el aprendizaje informal, en un museo, por ejemplo, cuando estamos delante de un cuadro y un dispositivo móvil nos explica la biografía del autor [9].

2.2.1 Caracterización del M-Learning

Las principales características que presenta el m-learning y que lo diferencia del e-learning, son [9,11]:

- Los alumnos tienen total flexibilidad.
- Independencia tecnológica de los contenidos: una lección no está hecha para un dispositivo concreto.
- *"Just in time, just for me":* lo que el alumno quiere, cuando el alumno lo quiere.
- Todas las actividades online del espacio de formación están disponibles para dispositivos móviles.
- Navegación sencilla y adaptación de contenidos teniendo en cuenta la navegabilidad, procesador y velocidad de conexión de estos dispositivos.

Ahora bien, los objetivos del m-learning son, en cierta medida, similares a los del e-learning, en el sentido de que buscan mejorar la calidad del aprendizaje y que pretenden favorecer el acceso a la formación a cualquier persona, pero se centran en:

- Producir experiencias educativas en cualquier lugar o momento con el uso de los dispositivos móviles.

- Responder a unas necesidades concretas y urgentes de aprendizaje.

Teniendo en cuenta las características y objetivos del m-learning, éste presenta una serie de ventajas y es realmente efectivo en:

- Aprendizaje basado en la resolución de problemas y mejora de determinadas habilidades.

- Aprendizaje al aire libre o para trabajos de campo.

- Aprendizaje en instituciones culturales. En estos entornos se ha demostrado que las tecnologías multimedia e inalámbricas son una eficaz herramienta, ya que proporcionan al usuario información de interés en función del lugar en el que se encuentre. Un ejemplo muy práctico sería estar en un parque y aprender exactamente qué plantas lo componen.

- Reciclaje profesional, en una sociedad tan dinámica y que nos exige estar cada vez más capacitados y cualificados para desempeñar un trabajo.

Pero otro hecho muy importante es que los dispositivos móviles, sobre todo el teléfono móvil, tiene un uso para el ocio incuestionable y es en las nuevas generaciones donde más está llegando a influenciar. El éxito de los juegos para móvil da una idea del potencial que puede tener el m-learning empleando juegos didácticos, entre los cuales se recopilan ejercicios tradicionales, como nuevos y son adaptados a los dispositivos móviles para estimular la memoria y las capacidades cognitivas, ya sea tanto para jóvenes como para personas adultas, como se está viendo últimamente en el mercado de las consolas portátiles (como la Nintendo DS, orientanda a todo el público con su amplia variedad de juegos , tanto de entretenimiento, como didácticos). Y es que está sobradamente demostrado que jugando se mejora la predisposición al aprendizaje y facilita dicho proceso al efectuarse de una manera agradable y estimulante.

También hay que ser conscientes de que el m-learning también presenta algunos inconvenientes que habría que tener en cuenta, y es que a pesar de sus ventajas, existen aún obstáculos que hay que salvar para desarrollar productos formativos de calidad que se adapten a las necesidades de los estudiantes, como serían:

- El ambiente en el que se desarrollará el estudio va a ser propenso a las interrupciones y no será óptimo para un estudio excesivamente profundo y extenso en tiempo.

- El alumno también puede sufrir una incomodidad en la velocidad reducida de descarga en el dispositivo móvil o las limitaciones de cobertura pueden hacer el proceso más lento.

- También se puede presentar dificultades debidas a que la pantalla de estos dispositivos suele ser pequeña y su resolución puede resultar en algunos casos insuficiente, junto también a una limitación de memoria y procesamiento, con respecto a los ordenadores de sobremesa o los portátiles.

- El mercado actual de los móviles es muy heterogéneo, la diversidad de dispositivos con sus respectivos modelos, resoluciones de pantallas, sistemas operativos, conectividad, compatibilidad, etc. presentan una limitación a la hora de establecer una media, para cualquier formato de contenidos a generar, con lo cual es necesario avanzar en etapas y brindar requerimientos claros para cada tipo de contenido.

- Una de las principales barreras a afrontar es la conectividad a Internet. Las nuevas generaciones de smartphone ya poseen acceso a redes wifi, permitiendo un acceso sin restricciones a los contenidos on-line y lograr niveles de interactividad de forma descentralizada. Pero los móviles que todavía no poseen esta funcionalidad deben conectarse a Internet con un alto costo (cada empresa establece un valor por cada kb descargado), siendo también mucho más lenta la carga de las páginas. Esto reduce notablemente los usuarios potenciales para acceder al m-learning.

Y es que como es lógico, dichos dispositivos aún presentan limitaciones técnicas, que se pueden ir salvando y mejorando conforme evolucione la tecnología, pero en el que también juega un papel imprescindible la calidad de las aplicaciones didácticas que se empleen junto con un contenido ideado para conseguir unos objetivos didácticos o de aprendizaje claros.

2.2.2 Herramientas y plataformas m-learning

Las herramientas que se ponen en práctica en el e-learning no pueden emplearse tal cuál en el m-learning, debido a las peculiaridades y limitaciones que presentan los dispositivos móviles. Así pues, algunas de estas herramientas son adaptadas para su aplicación en el m-learning, entre las que destacamos:

- e-Books: libros en formato electrónico, adaptados para ser leídos desde dispositivos móviles
- Videos: vídeos en formato MP4, el cual es soportado por una amplia gama de dispositivos.
- Animaciones: películas flash.
- Avisos por SMS.
- Foro: foros de debate especiales para este tipo de dispositivos.
- Chat.

Si bien una plataforma m-learning es un conjunto de herramientas y soluciones que se desarrollan para dar un soporte m-learning, en este caso, estas plataformas están siendo desarrolladas por empresas y también por universidades en colaboración con otros organismos de investigación y con empresas relacionadas con la fabricación de dispositivos móviles. Siendo el m-learning aún una metodología muy reciente y que además depende también del desarrollo y evolución de los dispositivos móviles, la mayor parte de estas plataformas m-learning están en una fase de experimentación y financiadas por proyectos entre universidades de varios países [11].

CAPÍTULO 3

PLATAFORMA DE DESARROLLO SOFTWARE PARA DISPOSITIVOS MÓVILES: J2ME

En este capítulo se introduce y expone los aspectos más importantes sobre la Plataforma de desarrollo J2ME (Java 2 Micro Edition), la cual proporciona los medios necesarios para diseñar y construir aplicaciones Java destinadas a ejecutarse en dispositivos móviles con pocos recursos, como pueden ser PDAs o teléfonos móviles.

El motivo de centrarnos en esta plataforma es consecuencia del alto nivel de aceptación que ha obtenido, habiéndose implantado en una amplia gama de dispositivos, entre los que hay que destacar que la mayoría de los fabricantes de teléfonos móviles la incorpora en sus productos. Por esa razón, nos decantamos por esta plataforma de desarrollo para la realización de nuestro proyecto, ya que nos garantiza que podrá ser llevada y ejecutada a una extensa variedad de dispositivos móviles, satisfaciéndose uno de los requisitos requeridos en la propuesta del proyecto

Además, se revisarán también las diferencias fundamentales entre J2ME y las Plataformas de desarrollo J2SE (Java 2 Standard Editions) y J2EE (Java 2 Enterprise Edition), consideradas como plataformas hermanas, puesto que han sido desarrolladas por la misma empresa y presentan semejanzas en su arquitectura, conteniendo elementos comunes. Seguidamente, pasaremos a profundizar sobre los conceptos teóricos que subyacen bajo la plataforma J2ME y comentaremos los aspectos básicos en la programación de dispositivos móviles de pequeña envergadura, como puede ser la arquitectura, configuraciones, perfiles, paquetes específicos, limitaciones, etc.

3.1 Introducción

La empresa Sun Microsystems lanzó a mediados de los años 90 el lenguaje de programación Java que, aunque en un principio fue diseñado para generar aplicaciones que controlaran electrodomésticos como lavadoras, frigoríficos, etc., debido a su gran robustez e independencia de la plataforma donde se ejecutase el código, desde sus comienzos se utilizó para la creación de componentes interactivos integrados en páginas Web y programación de aplicaciones independientes. Estos componentes se denominaron applets y casi todo el trabajo de los programadores se dedicó al desarrollo de éstos. Con los años, Java ha progresado enormemente en varios ámbitos como servicios HTTP (HyperText Transfer Protocol), servidores de aplicaciones, acceso a bases de datos (JDBC, Java DataBase Connectivity) y se ha ido adaptando a

las necesidades tanto de los usuarios como de las empresas ofreciendo soluciones y servicios tanto a unos como a otros [15].

Debido al crecimiento exponencial de la tecnológica en estos últimos años, Java ha desarrollado soluciones personalizadas para cada área tecnológica. Sun ha agrupado cada área en una edición distinta de su lenguaje Java. Estas ediciones son Java 2 Stantard Edition, orientada al desarrollo de aplicaciones independientes y de Applets, Java 2 Enterprise Edition, enfocada al entorno empresarial y Java 2 Micro Edition, orientada a la programación de aplicaciones para pequeños dispositivos. Esta última edición de Java es en la que nos vamos a centrar, ya que nuestro estudio va enfocado al uso de este tipo de dispositivos [15, 18].

3.2 Análisis comparativo de las soluciones Java de Sun

Sun Microsystems dividió su paquete Java en 3 ediciones distintas, como se ha comentado anteriormente: J2SE (Java Standard Edition) orientada al desarrollo de aplicaciones independientes de la plataforma, J2EE (Java Enterprise Edition) orientada al entorno empresarial y J2ME (Java Micro Edition) orientada a dispositivos con capacidades restringidas. A continuación, mostramos cuáles son las características de cada una de las versiones [15,16,18]:

3.2.1 Java 2 Platform, Standard Edition (J2SE)

Está orientada al desarrollo de aplicaciones independientes de la plataforma y es la edición de Java que en cierta forma recoge la iniciativa original del lenguaje Java. Tiene las siguientes características:

- Inspirado inicialmente en C++, pero con componentes de alto nivel, como soporte nativo de strings y recolector de basura.

- Código independiente de la plataforma, precompilado a bytecodes intermedio y ejecutado en el cliente por una JVM (Java Virtual Machine).

- Modelo de seguridad tipo *sandbox* proporcionado por la JVM.

- Abstracción del sistema operativo subyacente mediante un juego completo de APIs (Aplication Program Interface) de programación.

Esta versión de Java contiene el conjunto básico de herramientas usadas para desarrollar Java Applets, así como las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, etc. [15,16,18].

3.2.2 Java 2 Platform, Enterprise Edition (J2EE)

Esta plataforma está orientada al entorno empresarial ya que el software empresarial tiene unas características propias marcadas. Además está pensado no para ser ejecutado en un equipo, sino sobre una red de ordenadores de manera distribuida y remota mediante Ejes (Enterprise Java Beans). De hecho, el sistema se monta sobre varias unidades o aplicaciones. En muchos casos, además, el software empresarial requiere que sea capaz de integrar datos provenientes de entornos heterogéneos. Esta edición está orientada especialmente al desarrollo de servicios Web, servicios de nombres, persistencia de objetos, XML (Extensive Makup Language), autenticación, APIs para la gestión de transacciones, etc. El cometido de esta especificación es ampliar la J2SE para dar soporte a los requisitos de las aplicaciones de empresa.

3.2.3 Java 2 Platform, Micro Edition (J2ME)

La edición Java 2 Micro Edition fue presentada en 1999 por Sun Microsystems con el propósito de habilitar aplicaciones Java para pequeños dispositivos. En esta presentación, lo que realmente se enseñó fue una primera versión de una nueva Java Virtual Machine (JVM) que podía ejecutarse en dispositivos Palm. Java Micro Edition es la versión del lenguaje Java que está orientada al desarrollo de aplicaciones para dispositivos pequeños con capacidades restringidas tanto en pantalla gráfica, como de procesamiento y memoria (teléfonos móviles, PDA`s, Handhelds, Pagers, etc). La posterior aparición de esta tecnología, (hemos visto que la tecnología Java nació a mediados de los 90 y Java Micro Edition apareció a finales), es debido a que las necesidades de los usuarios ha cambiado mucho en estos últimos años,

especialmente en telefonía móvil y cada vez demandan más servicios y prestaciones por parte tanto de los terminales como de las compañías. Además el uso de esta tecnología depende del asentamiento en el mercado de otras, como GPRS, íntimamente asociada a J2ME y que no ha estado al alcance de los usuarios hasta hace relativamente poco. J2ME es la tecnología del futuro para la industria de los dispositivos móviles [15,18].

En la figura 3.1 viene representada la arquitectura que presenta la plataforma Java 2 de Sun comentada.

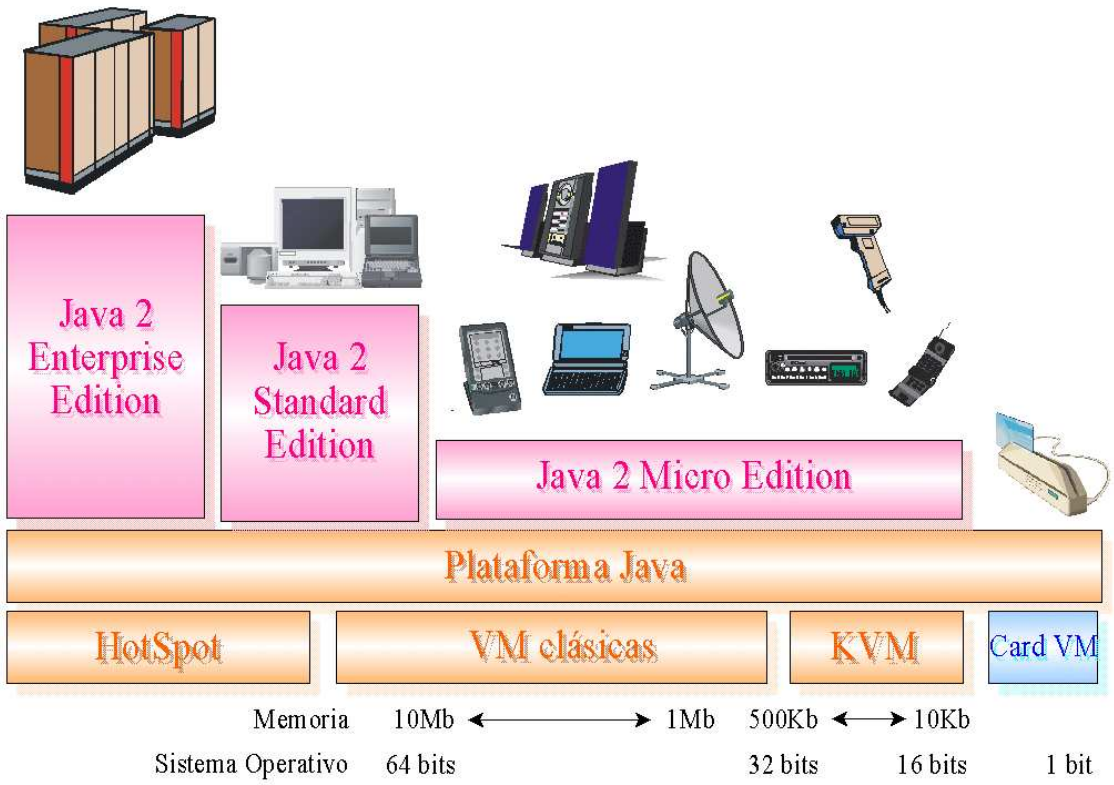


Figura 3.1: Arquitectura de la plataforma Java 2 de Sun.

3.2.4 Comparativa de las plataformas Java

En la actualidad no podemos ver Java como un simple lenguaje de programación, si no como un conjunto de tecnologías que abarca a todos los ámbitos de la computación con dos elementos en común:

- El código fuente en lenguaje Java es compilado a código intermedio interpretado por una Java Virtual Machine (JVM), por lo que el código ya compilado es independiente de la plataforma.

- Todas las tecnologías comparten un conjunto más o menos amplio de APIs básicas del lenguaje, agrupadas principalmente en los paquetes java.lang y java.io.

Un claro ejemplo de éste último punto es que J2ME contiene una mínima parte de las APIs de Java. Esto es debido a que la edición estándar de APIs de Java ocupa 20 MB y los dispositivos pequeños disponen de una cantidad de memoria mucho más reducida. En concreto, J2ME usa 37 clases de la plataforma J2SE provenientes de los paquetes java.lang, java.io, java.util. Esta parte de la API que se mantiene fija forma parte de lo que se denomina "Configuración" y se detallará más adelante. Otras diferencias con la plataforma J2SE vienen dadas por el uso de una máquina virtual distinta de la clásica JVM denominada KVM (Kilobyte Virtual Maquine). Esta KVM tiene unas restricciones que hacen que no posea todas las capacidades incluidas en la JVM. Estas diferencias las veremos más detenidamente cuándo analicemos las capacidades de la KVM en el siguiente apartado.

Como vemos en la figura 3.2, J2ME representa una versión simplificada de J2SE. Sun separó estas dos versiones ya que, J2ME estaba pensada para dispositivos con limitaciones de proceso y capacidad gráfica. También separó J2SE de J2EE porque este último exigía unas características muy pesadas o especializadas de E/S, trabajo en red, etc. Por tanto, separó ambos productos por razones de eficiencia. Hoy, J2EE es un superconjunto de J2SE pues contiene toda la funcionalidad de éste y más características, así como J2ME es un subconjunto de J2SE (excepto por el paquete javax.microedition) ya que, como se ha mencionado, contiene varias limitaciones y peculiaridades con respecto a J2SE.

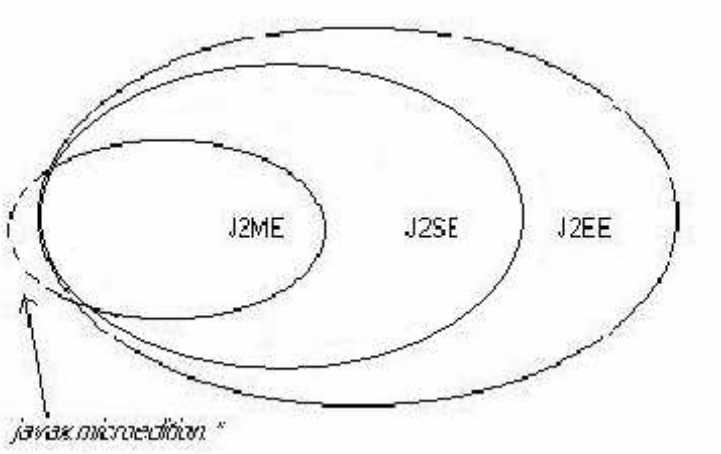


Figura 3.2: Relación entre las APIs de la plataforma Java.

De forma simple, se puede considerar a J2ME y J2EE como versiones reducidas y ampliadas respectivamente de J2SE (ver figura 3.2). En realidad cada una de las ediciones está enfocada a ámbitos de aplicación muy distintos. Las necesidades computacionales y APIs de programación requeridas para un juego ejecutándose en un móvil difieren en gran medida con las de un servidor distribuido de aplicaciones basado en EJB (Enterprise JavaBeans) [15,17,19].

3.3 Nociones básicas de J2ME

Dado que para el desarrollo de nuestro proyecto vamos a utilizar J2ME, en este apartado exponemos los componentes que forman parte de esta tecnología, describiendo su estructura y los elementos que la integran para tener una idea más detallada sobre J2ME [15,19]:

- Por un lado, tenemos una serie de *máquinas virtuales Java* con diferentes requisitos, cada una para diferentes tipos de pequeños dispositivos y que son las que median con el sistema operativo del dispositivo.

- Las *configuraciones*, que son un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones para dispositivos de características específicas y que teniendo en cuenta la arquitectura de ejecución, se establecen por encima de la máquina virtual java correspondiente. Existen 2 configuraciones definidas en J2ME:

- Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria

- Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

- *Perfiles*, que son unas bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos. Éstas se construyen sobre las configuraciones.

- *Paquetes opcionales*, son bibliotecas Java de clases específicas que se ha diseñado para responder a requisitos concretos de mercado y ofrecen un conjunto de APIs estándares para utilizar tanto tecnologías existentes como emergentes (ej.: Bluetooth).

Teniendo en cuenta estos componentes, un entorno de ejecución determinado de J2ME se compone entonces de una selección de:

a) Máquina virtual.
b) Configuración.
c) Perfil.
d) Paquetes opcionales.

La arquitectura de un entorno de ejecución para una plataforma móvil la podemos ver en la Figura 3.3. En ella se observa que sobre el sistema operativo del dispositivo se instala una máquina virtual java acorde a las características del dispositivo, sobre ésta, se establece una configuración y finalmente sobre la configuración se desarrollan unos perfiles.

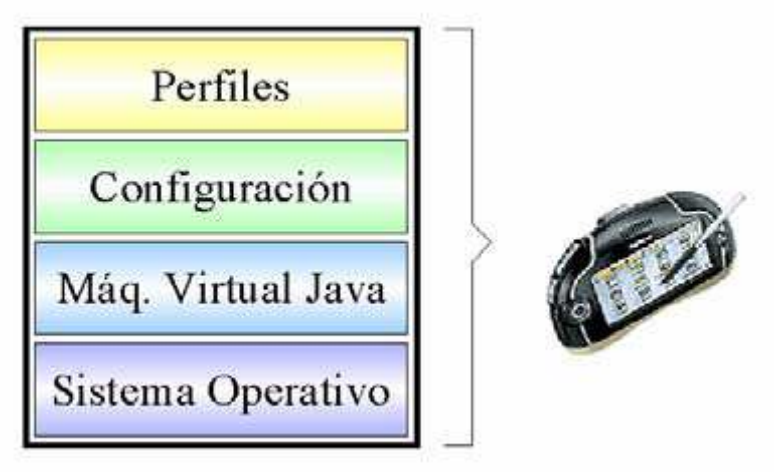


Figura 3.3: Entorno de ejecución.

A continuación se presenta en detalle cada uno de los componentes de un entorno de ejecución que forman parte de la plataforma J2ME:

3.3.1 Máquinas Virtuales J2ME

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVM de referencia, adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Anteriormente, se ha comentado que existen 2 configuraciones, CLDC y CDC, cada una con unas características propias que veremos en profundidad más adelante. Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM. Veremos, a continuación, las características principales de cada una de ellas.

- **KVM**

Se corresponde con la Máquina Virtual más pequeña. Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, (según Sun, que fue quien la desarrolló, se compone de aproximadamente unas 24000 líneas de código) y fue diseñada para ser:

- Pequeña, con una carga de memoria entre los 40Kb y los 80 Kb, dependiendo de la plataforma y las opciones de compilación.
- Alta portabilidad.
- Modulable.
- Lo más completa y rápida posible y sin sacrificar características para las que fue diseñada.

Sin embargo, esta baja ocupación de memoria hace que posea algunas limitaciones con respecto a la clásica *Java Virtual Machine* (JVM):

1- No hay soporte para tipos en coma flotante. No existen por tanto los tipos double ni float.

2- No existe soporte para JNI, *Java Native* Interface, debido a los recursos limitados de memoria.

3- No existen cargadores de clases (*class loaders*) definidos por el usuario. Sólo existen los predefinidos, es parte de las restricciones de seguridad del modelo *sandbox*.

4- No se permiten los grupos de hilos (threads) o hilos *daemon*, aunque es capaz de implementar múltiples hilos o multithreading.

5- No existe la finalización de instancias de clases ya que no existe el método Object.finalize().

6- No hay referencias débiles, es decir, un objeto que está siendo apuntado mediante una referencia débil es un candidato para la recolección de basura.

7- Limitada capacidad para el manejo de excepciones debido a que el manejo de éstas depende en gran parte de las APIs de cada dispositivo.

8- Carece del mecanismo de reflexión, por el cual los objetos pueden obtener información de otros objetos en tiempo de ejecución como, por ejemplo, los archivos de clases cargados o sus campos y métodos.

La verificación de clases merece un comentario aparte. El verificador de clases estándar de Java es demasiado grande para la KVM. Este verificador de clases es el encargado de rechazar las clases no válidas en tiempo de ejecución.

Los dispositivos que usen la configuración CLDC y KVM introducen un algoritmo de verificación de clases en dos pasos. Este proceso puede apreciarse gráficamente en la Figura 3.4.

La KVM puede ser compilada y probada en 3 plataformas distintas:

1- Solaris Operating Environment.

2- Windows.

3- PalmOs.

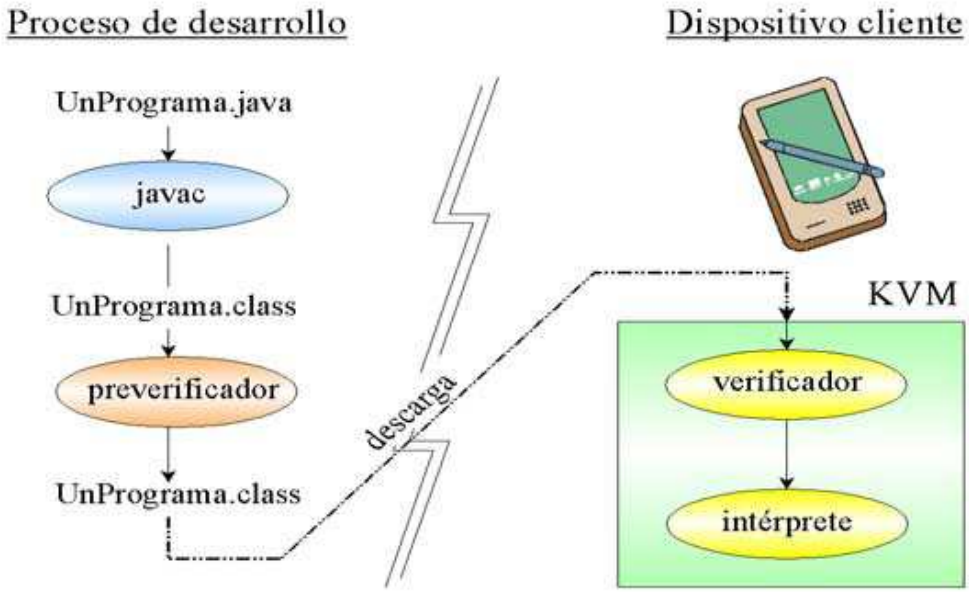


Figura 3.4 Preverificación de clases en CDLC/KVM.

- **CVM**

La CVM (Compact Virtual Machine) ha sido tomada como Máquina Virtual Java de referencia para la configuración CDC y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y entorno a 2Mb o más de memoria RAM. Las características que presenta esta Máquina Virtual son:

1- Sistema de memoria avanzado.

2- Tiempo de espera bajo para el recolector de basura.

3- Separación completa de la VM del sistema de memoria.

4- Recolector de basura modularizado.

5- Portabilidad.

6- Rápida sincronización.

7- Ejecución de las clases Java fuera de la memoria de sólo lectura (ROM).

8- Soporte nativo de hilos.

9- Baja ocupación en memoria de las clases.

10- Proporciona soporte e interfaces para servicios en Sistemas Operativos de Tiempo Real.

11- Conversión de hilos Java a hilos nativos.

12- Soporte para todas las características de Java2 v1.3 y librerías de seguridad, verificación de clases, referencias débiles, Interfaz Nativa de Java (JNI), invocación remota de métodos (RMI), Interfaz de depuración de la Máquina Virtual (JVMDI).

3.3.2 Configuraciones

Aunque ya se ha mencionado anteriormente de forma breve, para que quede de forma clara, una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Éstas APIs describen las características básicas, comunes a todos los dispositivos:

- Características soportadas del lenguaje de programación Java.
- Características soportadas por la Máquina Virtual Java.
- Bibliotecas básicas de Java y APIs soportadas.

Como también se ha indicado, existen dos configuraciones en J2ME, CDC y CDLC, que describiremos más en profundidad [15,17,19]:

- **Configuración de dispositivos con conexión, CDC (*Connected Device Configuration*)**

  Orientada a dispositivos con cierta capacidad computacional y de memoria. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC usa una Máquina Virtual Java similar en sus

características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo. Ésta Máquina Virtual es la que hemos visto como CVM (Compact Virtual Machine). La CDC está enfocada a dispositivos con las siguientes capacidades:

- Procesador de 32 bits.
- Disponer de 2 Mb o más de memoria total, incluyendo memoria RAM y ROM.
- Poseer la funcionalidad completa de la Máquina Virtual Java2.
- Conectividad a algún tipo de red.

La CDC está basada en J2SE v1.3 e incluye varios paquetes Java de la edición estándar. Las peculiaridades de la CDC están contenidas principalmente en el paquete javax.microedition.io, que incluye soporte para comunicaciones HTTP y basadas en datagramas. La tabla 3.1 nos muestra las librerías incluidas en la CDC.

Nombre de Paquete CDC	Descripción
java.io	Clases e interfaces estándar de E/S.
java.lang	Clases básicas del lenguaje.
java.lang.ref	Clases de referencia.
java.lang.reflect	Clases e interfaces de reflection.
java.math	Paquete de matemáticas.
java.net	Clases e interfaces de red.
java.security	Clases e interfaces de seguridad.
java.security.cert	Clases de certificados de seguridad.
java.text	Paquete de texto.
java.util	Clases de utilidades estándar.
java.util.jar	Clases y utilidades para archivos JAR.
java.util.zip	Clases y utilidades para archivos ZIP y comprimidos.
Javax.microedition.io	Clases e interfaces para conexión genérica CDC.

Tabla 3.1 Librerías de configuración CDC.

- **Configuración de dispositivos limitados con conexión, CLDC (*Connected Limited Device Configuration*).**

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de estos dispositivos son: teléfonos móviles, buscapersonas (pagers), PDAs, organizadores personales, etc. Debido a que nuestro proyecto va dirigido precisamente a este tipo de dispositivos, se va a prestar una mayor atención a esta configuración.

Como CLDC está orientado a dispositivos con ciertas restricciones, indicamos que algunas de estas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño. Los dispositivos que usan CLDC deben cumplir los siguientes requisitos:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb. de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución.

- Procesador de 16 o 32 bits con al menos 25 Mhz de velocidad.

- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.

- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

La CLDC aporta las siguientes funcionalidades a los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su Máquina Virtual (KVM).

- Un subconjunto de las bibliotecas Java del núcleo.

- Soporte para E/S básica.
- Soporte para acceso a redes.
- Seguridad.

La Tabla 3.2 nos muestra las librerías incluidas en la CLDC.

Nombre de paquete CLDC	Descripción
Java.io	Clases y paquetes estándar de E/S. Subconjunto de J2SE
Java.lang	Clases e interfaces de la Máquina Virtual. Subconjunto de J2SE.
Java.util	Clases, interfaces y utilidades estándar. Subconjunto de J2SE
Javax.microedition.io	Clases e interfaces de conexión genérica CLDC

Tabla 3.2 Librerías de configuración CLDC.

Un aspecto muy a tener en cuenta es la seguridad en CLDC. Hay que asegurar la integridad de los datos transmitidos y de las aplicaciones. Éste modelo de seguridad no es nuevo, ya que la ejecución de applets (programas Java que se ejecutan en un navegador web) se realiza en una zona de seguridad denominada *sandbox*. Los dispositivos englobados en CLDC se encuentran ante un modelo similar al de los applets. Este modelo establece que sólo se pueden ejecutar algunas acciones que se consideran seguras. Existen, entonces, algunas funcionalidades críticas que están fuera del alcance de las aplicaciones. De esta forma, las aplicaciones ejecutadas en estos dispositivos deben cumplir unas condiciones previas:

- Los ficheros de clases Java deben ser verificados como aplicaciones Java válidas.
- Sólo se permite el uso de APIs autorizadas por CLDC.
- No está permitido cargar clases definidas por el usuario.
- Sólo se puede acceder a características nativas que entren dentro del CLDC.

- Una aplicación ejecutada bajo KVM no debe ser capaz de dañar el dispositivo dónde se encuentra. De esto se encarga el verificador de clases que se asegura que no haya referencias a posiciones no válidas de memoria. También comprueba que las clases cargadas no se ejecuten de una manera no permitida por las especificaciones de la Máquina Virtual.

En cualquier caso, una determinada Configuración no se encarga del mantenimiento del ciclo de vida de la aplicación, interfaces de usuario o manejo de eventos, sino que estas responsabilidades caen en manos de los Perfiles.

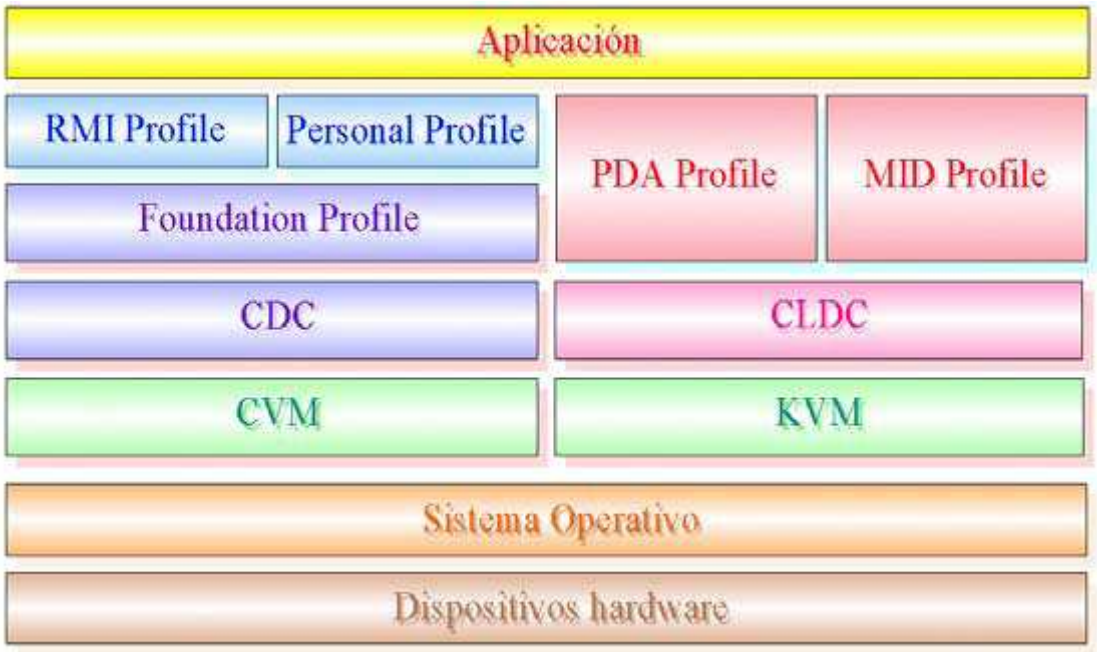
3.3.3 Perfiles

Un perfil define las APIs que controlan el ciclo de vida de la aplicación, interfaz de usuario, etc. Más concretamente, un perfil es un conjunto de APIs orientado a un ámbito de aplicación determinado. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutarán en ellos [15,17,18].

Las bibliotecas de la interfaz gráfica son un componente muy importante en la definición de un perfil. Aquí nos podemos encontrar grandes diferencias entre interfaces, desde el menú textual de los teléfonos móviles hasta los táctiles de los PDAs.

El perfil establece las APIs que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación se cuente tanto con las APIs del perfil como de la configuración. Un perfil siempre se construye sobre una configuración determinada. De este modo, podemos pensar en un perfil como un conjunto de APIs que dotan a una configuración de funcionalidad específica.

La figura 3.5, muestra en mayor detalle un entorno de ejecución J2ME.



nn

Figura 3.5 Arquitectura del entorno de ejecución de J2ME.

Como anteriormente se vió, para una configuración determinada se usa una Máquina Virtual Java específica. Con la configuración CDC se utiliza la CVM (Compact Virtual Machina) y con la CLDC la KVM (Kilobyte Virtual Machina). Con los perfiles ocurre lo mismo, para la configuración CDC tenemos los siguientes perfiles:

- Foundation Profile.
- Personal Profile.
- RMI Profile.

Y para la configuración CLDC:

- PDA Profile.
- Mobile Information Device Profile (MIDP).

Un perfil puede ser construido sobre cualquier otro, sin embargo, una plataforma J2ME sólo puede contener una configuración. A pesar de que

existen varios perfiles definidos para la plataforma J2ME, nos vamos a centrar exclusivamente en el MIDP, Mobile Information Device Profile, ya que es el utilizaremos para realizar nuestro proyecto:

- **Mobile Information Device Profile (MIDP):** Este perfil está construido sobre la configuración CLDC. MIDP fue el primer perfil definido para J2ME y está orientado para dispositivos con las siguientes características:

  - Reducida capacidad computacional y de memoria.
  - Conectividad limitada (en torno a 9600 bps).
  - Capacidad gráfica reducida (mínimo un display de 96x54 pixels monocromo).
  - Entrada de datos alfanumérica reducida.
  - 128 Kb de memoria no volátil para componentes MIDP.
  - 8 Kb de memoria no volátil para datos persistentes de aplicaciones.
  - 32 Kb de memoria volátil en tiempo de ejecución para la pila Java.

El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con:

- La aplicación (semántica y control de la aplicación MIDP)
- Interfaz de usuario.
- Almacenamiento persistente.
- Trabajo en red.
- Temporizadores.

En la Tabla 3.3 podemos ver cuáles son los paquetes que están incluidos en el perfil MIDP.

Paquetes del MIDP	Descripción
javax.microedition.lcdui	Clases e interfaces para GUIs
javax.microedition.rms	Record Management Sotorage. Soporte para el almacenamiento persistente del dispositivo
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases interfaces de conexión genérica
java.io	Clases e interfaces de E/S básica

java.lang	Clases e interfaces de la Máquina Virtual
java.util	Clases e interfaces de utilidades estándar

Tabla 3.3 Librerías del perfil MIDP.

Las aplicaciones que se desarrollan utilizando MIDP reciben el nombre de *MIDlets* (por simpatía con *APPlets*). Decimos así que un MIDlet es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC. Desde un punto de vista práctico MIDP es el único perfil actualmente disponible.

3.3.4 MIDLets

Los MIDlets son aplicaciones creadas usando la especificación MIDP. Están diseñados para ser ejecutados, en dispositivos con poca capacidad gráfica, de cómputo y de memoria. En estos dispositivos no disponemos de líneas de comandos donde poder ejecutar las aplicaciones que queramos, si no que reside en él un software denominado gestor de aplicaciones o AMS (Application Management System) que es el encargado de gestionar los MIDlets y los recursos que éstos ocupan. Este software nos permite ejecutar, pausar o destruir nuestras aplicaciones J2ME [15,17,18].

El gestor de aplicaciones realiza dos grandes funciones:

- Por un lado gestiona el ciclo de vida de los MIDlets.
- Por otro, es el encargado de controlar los estados por los que pasa el MIDlet mientras está en la memoria del dispositivo, es decir, en ejecución.

3.3.4.1 Ciclo de vida de un MIDlet

El ciclo de vida de un MIDlet pasa por 5 fases, que son localización, instalación, ejecución, actualización y borrado, como puede apreciarse en la figura 3.6.

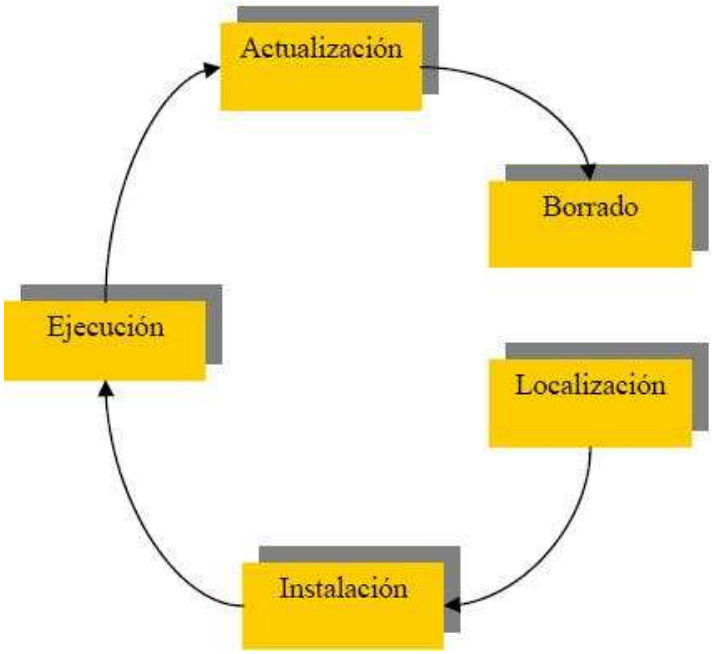


Figura 3.6: Ciclo de vida de un *MIDlet*.

El gestor de aplicaciones es el encargado de gestionar cada una de estas fases de la siguiente manera:

1. Localización: Esta fase es la etapa previa a la instalación del MIDlet y es dónde seleccionamos a través del gestor de aplicaciones la aplicación a descargar. Por tanto, el gestor de aplicaciones nos tiene que proporcionar los mecanismos necesarios para realizar la elección del MIDlet a descargar. El gestor de aplicaciones puede ser capaz de realizar la descarga de aplicaciones de diferentes maneras, dependiendo de las capacidades del dispositivo. Por ejemplo, mediante el puerto USB, o mediante una conexión inalámbrica, como WIFI, Bluetooth.

2. Instalación: Una vez descargado el MIDlet en el dispositivo, comienza el proceso de instalación. En esta fase el gestor de aplicaciones controla todo el proceso informando al usuario tanto de la evolución de la instalación como de si existe algún problema durante ésta. Cuando un MIDlet está instalado en el dispositivo, todas sus clases, archivos y almacenamiento persistente están preparados y listos para su uso.

3. Ejecución: Mediante el gestor de aplicaciones vamos a ser capaces de iniciar la ejecución de MIDlets. En esta fase, el gestor de

aplicaciones tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan durante esta ejecución, como veremos en el siguiente apartado.

4. Actualización: El gestor de aplicaciones tiene que ser capaz de detectar después de una descarga si el MIDlet descargado es una actualización de un MIDlet ya presente en el dispositivo. Si es así, nos tiene que informar de ello, además de darnos la oportunidad de decidir si queremos realizar la actualización pertinente o no.

5. Borrado: El gestor de aplicaciones borra el MIDlet seleccionado por el usuario, del dispositivo. El gestor de aplicaciones pedirá la confirmación antes de proceder a su borrado e informará de cualquier circunstancia que se produzca.

Hay que indicar que el MIDlet puede permanecer en el dispositivo todo el tiempo que queramos. Después de la fase de instalación, el MIDlet queda almacenado en una zona de memoria persistente del dispositivo MID. El usuario de éste dispositivo es el encargado de decidir en qué momento quiere eliminar la aplicación y así se lo hará saber al gestor de aplicaciones mediante alguna opción que éste nos suministre.

3.3.4.2. Estados de un MIDlet en fase de ejecución

El gestor de aplicaciones es el encargado de controlar los estados del MIDlet durante su ejecución. Durante ésta el MIDlet es cargado en la memoria del dispositivo y es aquí donde puede transitar entre 3 estados diferentes:

- **Activo**: El MIDlet está actualmente en ejecución.

- **Pausa**: El MIDlet no está actualmente en ejecución. En este estado el MIDlet no debe usar ningún recurso compartido. Para volver a pasar a ejecución tiene que cambiar su estado a Activo. Este estado se puede dar cuando por ejemplo un teléfono móvil recibe la interrupción de una llamada.

- **Destruido**: El MIDlet no está en ejecución ni puede transitar a otro estado. Además se liberan todos los recursos ocupados por el MIDlet.

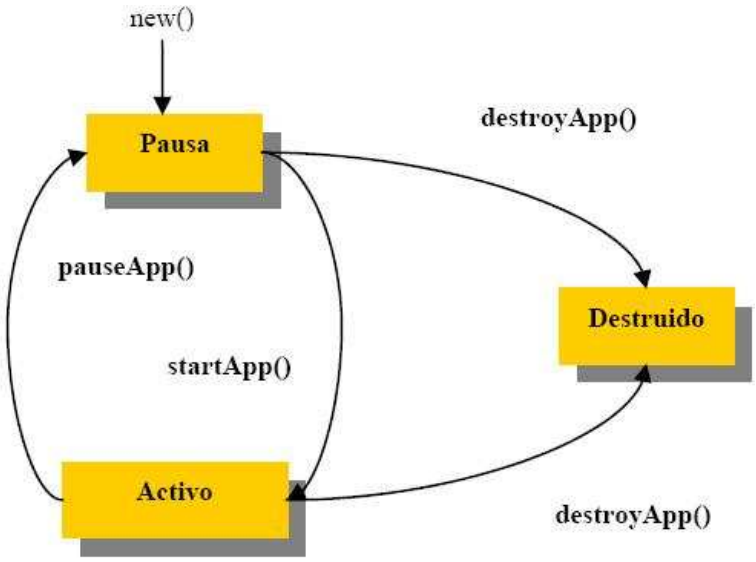


Figura 3.7: diagrama de estados de un *MIDlet* en ejecución

Como vemos en el diagrama de la figura 3.7, un MIDlet puede cambiar de estado mediante una llamada a los métodos MIDlet.startApp(), MIDlet.pauseApp() o MIDlet.destroyApp(). El gestor de aplicaciones cambia el estado de los MIDlets haciendo una llamada a cualquiera de los métodos anteriores. Un MIDlet también puede cambiar de estado por sí mismo, en función de las opciones que seleccione el usuario.

3.3.5 Interfaces de usuario con MIDP

Vamos a revisar brevemente un aspecto sobre las interfaces de usuario que se pueden definir con el perfil MIDP, ya que influirá a la hora de diseñar la interfaz gráfica de nuestro prototipo software. El perfil MIDP dispone de una serie de elementos que nos permiten crear dos tipos de interfaces de usuario, las de alto nivel y las de bajo nivel. De manera breve, vamos a explicar en que se diferencian [15,19]:

- **Interfaz de usuario de alto nivel**. Esta interfaz usa componentes tales como botones, cajas de texto, formularios, etc. Estos elementos son implementados por cada dispositivo y la finalidad de usar las APIs

de alto nivel es su portabilidad. Al usar estos elementos, perdemos el control del aspecto de nuestra aplicación ya que la estética de estos componentes depende exclusivamente del dispositivo donde se ejecute. En cambio, usando estas APIs de alto nivel ganaremos un alto grado de portabilidad de la misma aplicación entre distintos dispositivos. Fundamentalmente, se usan estas APIs cuando queremos construir aplicaciones de negocios.

- **Interfaz de usuario de bajo nivel**. Al crear una aplicación usando las APIs de bajo nivel, tendremos un control total de lo que aparecerá por pantalla. Estas APIs nos darán un control completo sobre los recursos del dispositivo y podremos controlar eventos de bajo nivel como, por ejemplo, el rastreo de pulsaciones de teclas. Generalmente, estas APIs se utilizan para la creación de juegos donde el control sobre lo que aparece por pantalla y las acciones del usuario juegan un papel fundamental.

En la figura 3.8, podemos ver la jerarquía de clases que pueden formar parte de una interfaz gráfica definida por el perfil MIDP. Hay que distinguir que la clase Screen y las que heredan de ella, junto con la clase ITEM y las que heredan de ITEM, forman parte de la API de alto nivel. La clase CANVAS y las de heredan de ella, forman parte de la API de bajo nivel. Aunque hay que puntualizar que no existe ningún impedimento que nos permita usar en el mismo MIDlet pantallas tanto derivadas de Canvas como de Screen.

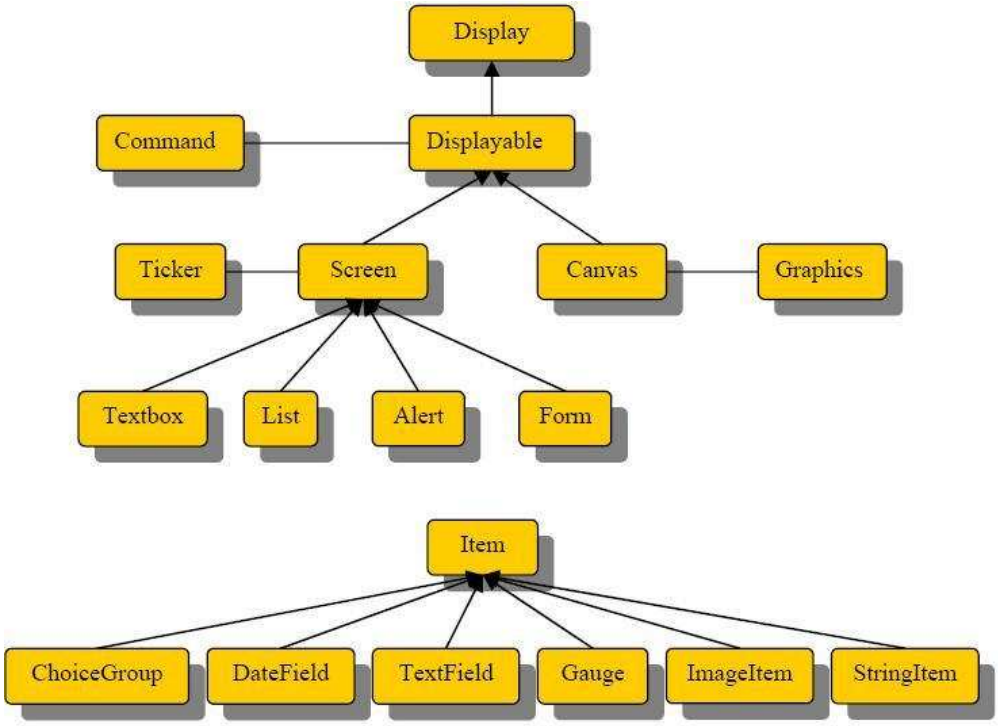


Figura 3.8: Jerarquía de clases de Display e Item.

3.3.6 Gestión de información en J2ME: RMS

En este apartado vamos a detallar cómo se gestiona y se lleva a cabo el almacenamiento de datos en memoria persistente de un dispositivo. Es importante que realicemos una especial mención a este aspecto, ya que nuestro prototipo software necesitará disponer de una serie de datos que deben permanecer en la memoria del dispositivo sobre el que se ejecute para su correcto funcionamiento.

El perfil MIDP proporciona precisamente un mecanismo a las aplicaciones que se ejecutan en dispositivos móviles, que les permite almacenar datos de forma persistente para su futura recuperación. Este mecanismo está implementado sobre una base de datos en registros que se denomina Record Management System o RMS (Sistema de gestión de registros) [15,17,19].

El sistema de gestión de registros o RMS, nos permite almacenar información en cada ejecución de nuestra aplicación o MIDlet. Esta información será guardada en el dispositivo en una zona de memoria dedicada para este propósito. La cantidad de memoria y la zona asignada para ello, dependerá de

cada dispositivo. Como ya se ha dicho, el RMS está implementado en una base de datos basada en registros, como ejemplo se puede ver la figura 3.9:

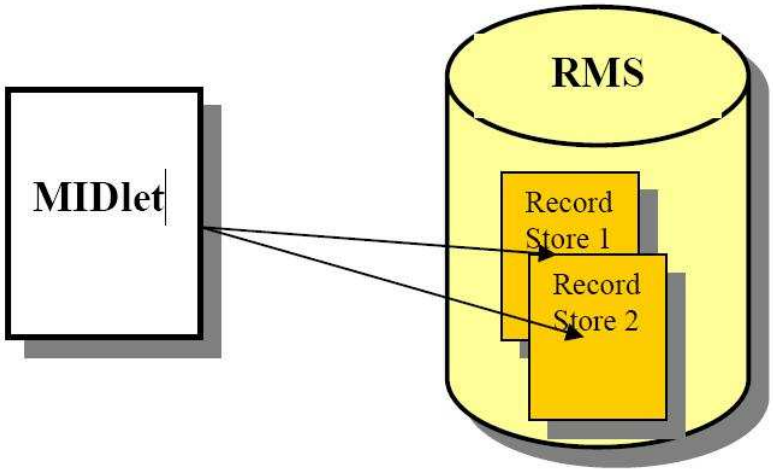


Figura 3.9: Comunicación entre un MIDlet y el RMS.

Los MIDlets son los encargados de crear Record Stores para comunicarse con ellos. Un Record Store queda almacenado en el dispositivo y puede ser accedido por cualquier MIDlet que pertenezca a la misma suite (ver figura 3.10).

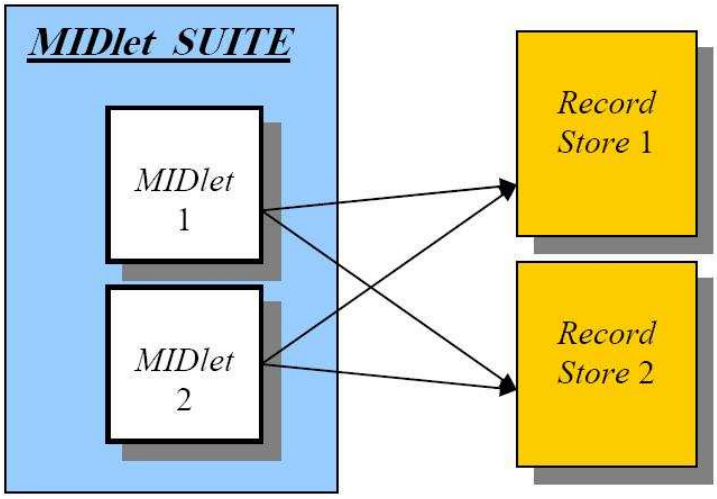


Figura 3.10 Acceso a un RMS a traves de una MIDlet suite.

Record Stores

Las propiedades de estos almacenes de registros son:

1- Cada Record Store está compuesto por cero o más registros.

2- Un nombre de Record Store es sensible a mayúsculas y minúsculas y está formado por un máximo de 32 caracteres UNICODE.

3- Dentro de una suite no pueden coexistir dos Record Stores con el mismo nombre.

4- Si una suite de MIDlets es borrada del dispositivo MID, todos los Record Stores pertenecientes a esa suite se borrarán.

5- Es posible que un MIDlet acceda a un Record Store creado por otra suite, siempre que ésta de permiso para ello.

Un Record Store tal como su nombre indica es un almacén de registros. Estos registros son la unidad básica de información que utiliza la clase RecordStore para almacenar datos. Cada uno de estos registros está formado por dos unidades:

- Un número identificador de registro (Record ID) que es un valor entero que realiza la función de clave primaria en la base de datos.

- Un array de bytes que es utilizado para almacenar la información deseada.

En la figura 3.11 se ilustra la estructura de un Record Store:

Record Store	
***Record* ID**	**Datos**
1	byte [] arrayDatos
2	byte [] arrayDatos
…	…

Figura 3.11: Estructura de un Record Store

Además de un nombre, cada Record Store también posee otros dos atributos:

- Número de versión: Es un valor entero que se actualiza conforme vayamos insertando, modificando o borrando registros en el Record Store. (Podemos consultar este valor invocando al método RecordStore.getVersion().)

- Marca temporal: Es un entero de tipo long que representa el número de milisegundos desde el 1 de enero de 1970 hasta el momento de realizar la última modificación en el Record Store. (Este valor lo podemos obtener invocando al método RecordStore.getLastModified()).

Así, la estructura de un Record Store se aproxima más a la figura 3.12:

Nombre: Record Store 1
Version: 1.0
TimeStamp: 2909884894049
Registros:

Record ID	Datos
1	byte [] arrayDatos
2	byte [] arrayDatos
…	…

Figura 3.12: Estructura Completa de un Record Store.

CAPÍTULO 4

DESARROLLO DEL PROYECTO

En los capítulos anteriores se ha expuesto el propósito y los objetivos del proyecto abordado y se ha tratado de plasmar de una manera clara y concisa, los conceptos y herramientas sobre los que se va a desarrollar este proyecto.

Tratándose de un prototipo software de un juego educativo basado en preguntas tipo test para dispositivos móviles, este proyecto se clasifica en un proyecto de desarrollo software y como tal, debe seguirse las actividades de la Ingeniería del Software para conseguir un buen diseño e implementación. Así pues, en este capítulo se detalla las distintas fases de desarrollo que hemos seguimos para la realización de este prototipo.

4.1 Fases de desarrollo

El desarrollo del proyecto se puede dividir en varias fases, que en este caso corresponderán al fin y al cabo a los pasos que se detallan mediante las técnicas de ingeniería del software, como serán el análisis del sistema, desarrollo del sistema, implementación y donde se tendrá en cuenta en cada una de estas fases, las peculiaridades y limitaciones que presentan los dispositivos móviles, ya que finalmente el prototipo desarrollado será implantado y ejecutado en un dispositivo de este tipo.

A día de hoy, no existe una definición precisa, única y estandarizada para la Ingeniería del Software. Sin embargo, las dos que se exponen seguidamente, abarcan ampliamente el concepto y resultan perfectamente válidas para su entendimiento:

- Ingeniería del Software es la construcción de software de calidad con un presupuesto limitado y un plazo de entrega en contextos de cambio continuo.

- Ingeniería del Software es el establecimiento y uso de principios y métodos firmes de ingeniería para obtener software económico que sea fiable y funcione de manera eficiente en maquinas reales.

La Ingeniería del Software requiere llevar a cabo numerosas actividades, las cuales se pueden agrupar en etapas, o también llamadas fases, que se detallan a continuación [14]:

- **Especificación de Requerimientos**: define el propósito del sistema, las propiedades y restricciones del mismo, es decir, se describe el comportamiento esperado en el software una vez desarrollado. Gran parte del éxito de un proyecto de software radicará en la identificación de las necesidades del negocio, así como la interacción con los usuarios funcionales para la recolección, clasificación, identificación, priorización y especificación de los requerimientos del software.

- **Análisis del Sistema**: se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.

- **Diseño del Sistema**: se definen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- **Implementación**: se traduce el modelo a código fuente, pudiendo ser la parte más obvia del trabajo de la ingeniería del software. La complejidad y la duración de esta etapa está íntimamente relacionada al o lenguajes de programación utilizados, al diseño previamente realizado y también, en nuestro caso, a las características y limitaciones propias de los dispositivos móviles.

- **Prueba**: verificar y validar el sistema. Básicamente consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación del problema.

En los puntos siguientes se profundizará en cada una de estas etapas y en cómo se han llevado a cabo en el ámbito de nuestro proyecto.

4.2 Especificación de requerimientos

El primer paso en la Ingeniería del Software debe ser determinar el propósito último del proyecto, las propiedades que debe satisfacer y las restricciones a las que está sometido [14].

Este es, sin duda, un paso de vital importancia dentro del desarrollo de cualquier proyecto software ya que, sin conocer el propósito del mismo y todas

las limitaciones a las que debe hacer frente, resultaría muy difícil poder realizar una aplicación software que cumpliera o se ajustara a dicho propósito.

En este caso, el propósito de nuestro proyecto es conocido desde el mismo momento de su propuesta y como ya se ha expuesto anteriormente, es el siguiente:

Diseño y desarrollo de un prototipo software que implemente un juego educativo basado en preguntas tipo test para dispositivos móviles utilizando el lenguaje de programación y la tecnología J2ME (Java 2 Micro Edition). Dicho software permitirá al usuario autoentrenarse y autoevaluar sus conocimientos en cada una de las temáticas registradas en el software.

Una vez determinado el propósito último del proyecto, el siguiente paso consiste en especificar los requerimientos del mismo. Los requerimientos de un proyecto software son el conjunto de propiedades o restricciones definidas con total precisión, que dicho proyecto software debe satisfacer. Existen dos tipos bien diferenciados de requerimientos:

- **Requerimientos funcionales**: son aquellos que se refieren específicamente al funcionamiento de la aplicación o sistema.

- **Requerimientos no funcionales**: aquellos no referidos al funcionamiento estricto sino a otros factores externos.

En los siguientes subapartados definiremos cuales son estos requerimientos, tanto funcionales como no funcionales, para este proyecto. Sin embargo, estas definiciones sólo serán previas ya que en la actividad de análisis del sistema se pueden descubrir nuevas necesidades.

4.2.1 Requerimientos funcionales

Las funcionalidades que se esperan para esta aplicación y que pueden reclamar los usuarios potenciales de esta aplicación, siendo un juego con carácter educativo, son las siguientes:

- **Identificación del jugador.**

El usuario antes de iniciar un test, se podrá identificar, introduciendo sus iniciales, para luego al término del test, mostrar su puntuación y puesto en el ranking correspondiente a la asignatura o materia elegida (siempre y cuando esté entre los 5 primeros).

- **Seleccionar tipo de test.**

Se le permitirá al usuario escoger, antes de iniciar un test, entre dos tipos de test, donde la diferencia entre uno y otro radica en el número de preguntas que presentará, optando pues por test corto, que constará de diez preguntas y test largo, de veinte preguntas.

- **Seleccionar asignatura.**

El usuario podrá elegir la asignatura sobre la que se basará el test de preguntas que se le planteará.

- **Mostrar pregunta actual y totales del test.**

Durante el test, se mostrará el número de la pregunta sobre la que se encontrará el usuario y el número de preguntas de las que se compone el test, para ayudar a situarse en el mismo y saber en todo momento la pregunta sobre la que está y las que le quedan.

- **Retroceder entre las preguntas.**

Durante el cuestionario del test, se permitirá al usuario retroceder a las preguntas ya contestadas y modificar su respuesta, para permitir así una mayor interacción, permitiendo también una mayor reflexión de las respuestas dadas. Cuando se retroceda, las respuestas dadas previamente permanecerán intactas hasta su modificación, al igual que en su avance.

- **Consultar resultado del test.**

Tras contestar a todas las preguntas del test, se deberá mostrar el resultado del mismo; el número de respuestas correctas e incorrectas y

la puntuación. Seguidamente mostrará un resumen del test donde se verá las preguntas que han sido bien contestadas y las que no, permitiendo además ver de cada pregunta su resultado. Si se ha contestado bien, ver cual es la respuesta contestada y en el caso de haber contestado mal, la respuesta incorrecta dada y la correcta. Esto permitirá al usuario autoevaluarse.

- **Aleatoriedad de las preguntas.**

Para cada test o cuestionario planteado, la aplicación deberá de seleccionar de forma aleatoria las preguntas que compondrán el test de entre la base de datos de la asignatura elegida, tantas en función del tipo de test escogido, largo o corto y además como requisito no se deberán repetir ninguna en el mismo test, lógicamente.

- **Consultar ranking de puntuaciones.**

El usuario podrá consultar cada uno de los rankings de puntuaciones correspondientes a las distintas asignaturas. Los rankings aparecerán de forma ordenada por puntuación, mostrando por un lado las iniciales del usuario o jugador y su puntuación, estableciendo un número máximo de cinco puestos, debido a que estos dispositivos no suelen tener una pantalla amplia y no tiene tampoco sentido mantener una larga lista de puntuaciones.

- **Guardar los rankings en memoria permanente**

Los rankings de puntuaciones de las distintas asignaturas se almacenaran en memoria permanente del dispositivo para que, siempre que se inicie el juego, se carguen tal cual quedaron tras la última ejecución. Si un usuario, tras un cuestionario de una asignatura, obtiene una puntuación que se encuentre entre los 5 primeros del ranking, el ranking se actualizará, quedando reflejado en la BD de dicho ranking.

- **Inicializar ranking.**

Se permitirá al usuario resetear o inicializar todos los rankings de puntuaciones.

- **Consultar ayuda.**

  El juego incluirá una breve ayuda para aclarar el objetivo del juego y los pasos a dar para iniciar una partida, en este caso, para comenzar un test (de alguna asignatura concreta).

- **Salir del juego**.

  En todo momento se permitirá al usuario salir del juego y se le pedirá siempre la confirmación de esta petición por seguridad, ya que esta opción puede haberla seleccionado por error o simplemente puede cambiar de parecer.

4.2.2 Requerimientos no funcionales

Los requerimientos no funcionales, como hemos visto, tienen que ver con características que de una u otra forma puedan limitar el sistema, como puede ser el equipo informático a utilizar, plataforma sobre la que se va ejecutar, el rendimiento, interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), etc. Son tan importantes como los propios requerimientos funcionales y pueden incluso a llegar a ser críticos para la aceptación del sistema.

En este caso, es importante prestar una especial atención a los requerimientos no funcionales, ya que nuestro prototipo de juego educativo va dirigido a dispositivos móviles, los cuales imponen un alto grado de restricciones, debido a sus limitadas capacidades tanto en hardware como en software, y por lo tanto serán críticas a la hora de diseñar nuestro prototipo.

Teniendo esto en cuenta, los requerimientos no funcionales deben obtenerse y analizarse a partir de las restricciones que presenten estos dispositivos. A continuación, definimos dichos requerimientos en detalle:

- Requerimientos no funcionales del dispositivo móvil

Nuestra aplicación va dirigida a una amplia gama de dispositivos móviles, muy diferentes entre ellos, por lo que los requerimientos no

funcionales propios del dispositivo móvil, los podemos clasificar o dividir en dos tipos, los requerimientos de hardware y los requerimientos software.

Hardware

Los dispositivos móviles deben cumplir los siguientes requisitos de hardware como mínimo para ejecutar la máquina virtual de Java (KVM) necesaria para soportar la aplicación:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb de memoria no volátil para la Máquina Virtual Java y 32 Kb de memoria volátil (RAM) para la Máquina Virtual en tiempo de ejecución. También será necesario disponer de 64Kb en memoria permanente para ubicar la aplicación con sus bases de datos.

- Procesador de 16 o 32 bits con al menos 25 Mhz de velocidad.

- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.

- Tener conexión a algún tipo de red, ya sea sin cable o con cable, por ejemplo WIFI, bluetooth, USB, para efectuar la descarga de la aplicación al dispositivo.

Pantalla gráfica. Hay que considerar que estos dispositivos móviles presentan unas pantallas con características limitadas y muy distintas entre sí, tanto en tamaño como en resolución y gama cromática, con respecto a un monitor de un PC. Para ello habrá que prestar una mayor atención a los mecanismos necesarios para que la información que se muestre por pantalla aparezca de forma legible y pueda verse en su totalidad, a pesar de sus reducidas dimensiones y gama cromática.

Software

El requisito no funcional software para que el prototipo o aplicación se ejecute en un dispositivo móvil, es que dicho dispositivo soporte y posea la máquina virtual de Java, concretamente la KVM, que es la plataforma que está orientada y pensada para estos dispositivos de reducidas capacidades

computacionales, de memoria y gráficas. La KVM actúa de intermediaria entre la aplicación y el sistema operativo del dispositivo móvil. En el capítulo 3 de esta memoria se ha especificado las características de esta plataforma software, junto con las configuraciones y perfiles desarrollados en función del tipo de dispositivo.

Hay que tener en cuenta que para el desarrollo de una aplicación para un dispositivo móvil también será necesario saber qué configuración y perfiles soporta, que va en función de las características del dispositivo. En este proyecto, al ir dirigido a un dispositivo móvil del tipo teléfono móvil, PDA, etc, la configuración a usar para la creación del prototipo será la CLDC (configuración de dispositivos limitados con conexión) versión 1.1, junto con el perfil MIDP (Mobile Information Device Profile) versión 2.0, que ofrece más prestaciones y presenta menos limitaciones que la versión MIDP 1.0, acaparando un mayor número de dispositivos. Estas APIs y bibliotecas, como se especificó en el tercer capítulo de la memoria, describen las características básicas y comunes a todos los dispositivos sobre los que se va a ejecutar la aplicación y controlaran el ciclo de vida de la misma, y ya están disponibles en la plataforma de desarrollo de Netbeans, por lo que según el dispositivo móvil sobre el que queremos ejecutar nuestro prototipo o aplicación, habrá que elegir previamente en la fase de desarrollo.

- Requerimientos no funcionales de la interfaz

Los requerimientos de la interfaz gráfica entre la aplicación y el usuario están íntimamente ligados a la usabilidad y sus principios. Primeramente introduciremos la definición de usabilidad, para tener una idea clara de lo que debe contemplarse a la hora de diseñar una interfaz gráfica y seguidamente comentaremos el criterio que se va a seguir para el diseño de la interfaz gráfica de nuestro prototipo.

El concepto de usabilidad se puede definir de varias formas [23]:

- Se define coloquialmente como facilidad de uso, ya sea de una página Web, una aplicación para una PDA o cualquier otro sistema que interactúe con un usuario.

- La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

- La usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico.

A partir de estas tres definiciones se pueden obtener los principios básicos de la usabilidad, los cuales se asociarán a los requerimientos no funcionales que deberá cumplir la interfaz gráfica:

→ Facilidad de aprendizaje: Este principio se refiere a aquellas características de la interfaz que permiten comprender cómo usarla inicialmente y conseguir una interacción efectiva y productiva con nuevos usuarios. Depende de los siguientes factores:

  - Predecibilidad: Una vez conocida la aplicación, se debe saber en cada momento a qué estado se pasará en función de la tarea que se realice.

  - Síntesis: Los cambios de estado tras una acción deben ser fácilmente captados.

  - Generalización: Las tareas semejantes se resuelven de modo parecido.

  - Familiaridad: El aspecto de la interfaz tiene que resultar conocido y familiar para el usuario.

  - Consistencia: Siempre se han de seguir una misma serie de pasos para realizar una tarea determinada.

→ Flexibilidad: Relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. Características que hacen a una interfaz flexible son:

  - La posibilidad de diálogo llamado también control del diálogo por parte del usuario, donde hay que proporcionar al usuario la

capacidad para decidir cuándo empezar o acabar las operaciones, siempre que sea posible.

- La migración de tareas, tanto el usuario como el sistema deben poder realizar una tarea en exclusiva, o pasarla al otro, o realizarla de forma conjunta.

- Capacidad de sustitución, que permite que valores equivalentes puedan ser sustituidos los unos por los otros, como por ejemplo mostrar una distancia en distintos sistemas de medición.

- Capacidad de adaptación, donde sería ideal que la interfaz pudiera adaptarse automáticamente a las necesidades del usuario actual.

→ Robustez: Es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos y, también, la capacidad del sistema para tolerar fallos. Está relacionada con los siguientes factores o características:

- Navegable: El usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.

- Recuperación de información: La aplicación debe permitir volver a un estado anterior.

- Tiempo de respuesta: Es el tiempo necesario para que el sistema pueda mostrar los cambios realizados por el usuario.

- Persistencia: Un sistema persistente es aquel en el que las notificaciones al usuario permanecen como objetos manipulables después de su presentación.

- Uso de valores por defecto: Ayudan al usuario mediante recuerdo pasivo. Mostrar un valor por defecto ayuda a que el usuario sepa qué tipo de valor debe introducir.

Para cumplir estos requerimientos, vamos a emplear los mecanismos apropiados que nos permitan utilizar la interfaz gráfica que ofrece cada dispositivo móvil. Es decir, la interfaz gráfica del prototipo o aplicación será la

que disponga el dispositivo móvil, para que sea desde un principio más amigable, fácil e intuitiva la navegación y uso, de nuestro prototipo o juego educativo.

4.3 Análisis del Sistema

Una vez conocido el propósito del proyecto software, las funciones que debe cumplir y las restricciones a las que debe someterse, llega el momento de analizar el sistema y crear un modelo del mismo que sea correcto, completo, consistente, claro y verificable. Para conseguir esto se estudiarán los perfiles de usuario, se crearán y definirán casos de uso en base a los requerimientos previamente obtenidos. Por último se describirán ciertos escenarios de acción de dichos casos de uso [20].

4.3.1 Perfil de Usuario

En esta fase el primer paso es determinar quienes son los usuarios potenciales de la aplicación, para a partir de esto, obtener las características generales que nos permitan caracterizar los requisitos de usabilidad que posteriormente habrá que tener en cuenta en el diseño de la aplicación y de su interfaz gráfica.

En nuestro caso, la aplicación va destinada a cualquier tipo de usuario que al menos tenga los conocimientos necesarios para manejar el dispositivo móvil donde se ubicará la aplicación.

4.3.2 Casos de uso

Un caso de uso representa una clase de funcionalidad dada por el sistema como un flujo de eventos. También se puede definir como la representación de una situación o tarea de interacción de un usuario con la aplicación.

Los casos de uso son tareas con significado, coherentes y relativamente independientes, que los actores realizan en su trabajo cotidiano. En un caso de uso concreto puede participar más de un actor [20].

Los casos de uso describen cómo se realiza una tarea de manera exacta y constan de los siguientes elementos:

- Nombre único e unívoco.
- Actores participantes.
- Condiciones de entrada.
- Flujo de eventos.
- Condiciones de salida.
- Requerimientos especiales.

Por lo tanto, es necesario determinar cuales son los actores que van a participar en cada no de los casos de uso. Un actor modela una entidad externa que se comunica con el sistema, es decir, es un tipo de usuario del sistema. Un actor, al igual que un caso de uso, debe tener un nombre único y puede tener una descripción asociada.

En nuestro sistema, vamos a contar sólo con un tipo de actor, que vamos a definir como **Jugador**, y corresponde con la persona que interactúe con el sistema.

Una vez que hemos definido los actores del sistema, pasamos a crear los distintos casos de uso. A la hora de realizar esta acción es importante que cada uno de los requerimientos funcionales ya definidos aparezca en al menos uno de los casos de uso aunque, por otra parte, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que estamos en una fase de refinamiento del sistema donde queremos construir un modelo detallado del mismo.

Un paso previo a la creación y descripción de los distintos casos de uso es la obtención de los diversos diagramas de casos de uso de nuestro sistema. El primero es el diagrama frontera que describe completamente la funcionalidad del sistema y se presenta en la figura 4.1:

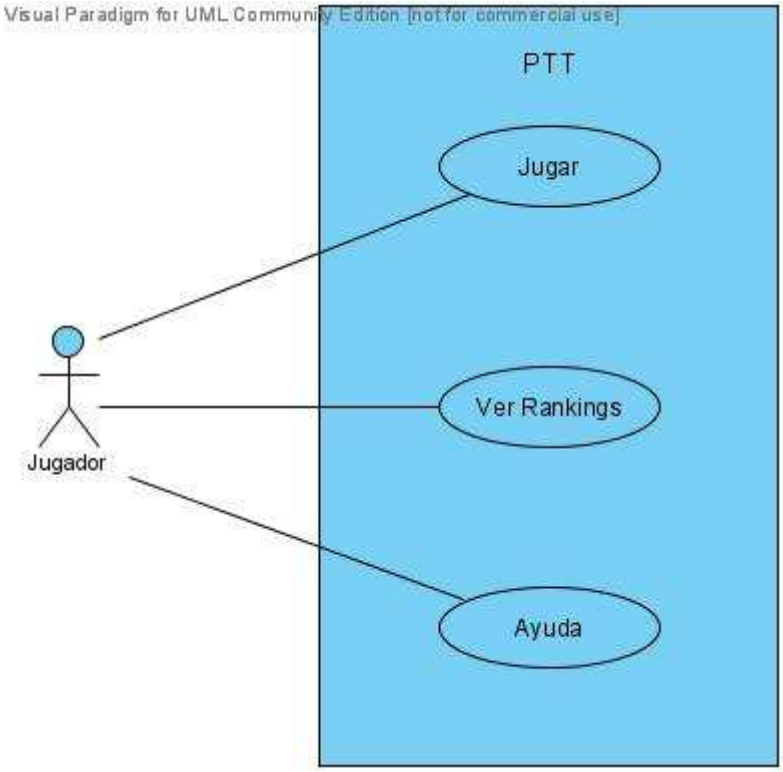


Figura 4.1: Diagrama Frontera del Sistema

Los casos de uso mostrados en un diagrama frontera pueden ser suficientemente precisos o necesitar ser explicados en mayor detalle. A la hora de detallar un caso de uso se pueden emplear dos tipos de relaciones:

- <<extend>>: Es una relación cuya dirección es hacia el caso de uso a detallar que representa comportamientos excepcionales del caso de uso.

- <<include>>: Es una relación cuya dirección es contraria a la de la relación <<extend>> que representa un comportamiento común del caso de uso.

En nuestro caso se da la circunstancia de que sólo un caso de uso debe ser descrito con mayor profundidad, porque presenta una mayor complejidad, tratándose del caso de uso “Jugar”. Los demás son tan simples que no lo requieren un mayor detalle, aunque también pasaremos a describirlos de manera breve.

A continuación, se expone en la figura 4.2 el diagrama que se identifica con el caso de uso Jugar:

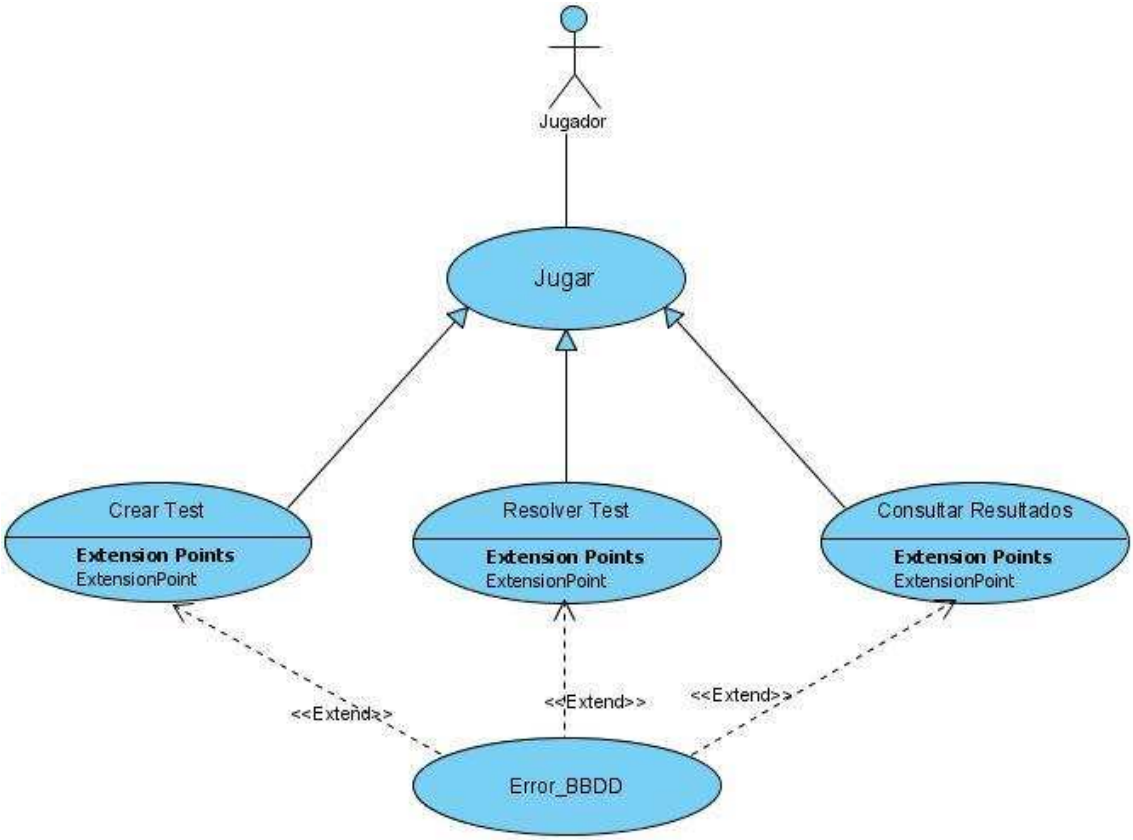


Figura 4.2: Diagrama Caso de uso Comenzar Juego

Caso de Uso 1: Crear Test

Actores participantes: Jugador.

Condiciones de entrada: El jugador haya iniciado la aplicación, haya elegido la opción de Jugar y la base de datos esté creada.

Flujo de eventos:

1. El sistema muestra por pantalla un campo de texto para solicitar al usuario sus iniciales.

2. El jugador introduce sus iniciales y pulsa ok.

3. El sistema muestra al usuario los tipos de test a elegir.

4. El jugador selecciona uno de los tipos de test y pulsa ok.

5. El sistema muestra al usuario las asignaturas existentes a elegir.

6. El jugador elige una de las asignaturas existentes y pulsa ok.

7. El sistema elabora un test según los datos introducidos, si no ocurren excepciones (E-1).

Condiciones de salida: Los datos introducidos quedan registrados. Se obtiene una lista con el número de las preguntas que forman el test y cumplen con las opciones elegidas.

Excepciones:

En los puntos 2, 4 y 6 el jugador puede elegir salir de la aplicación, finalizando el caso de uso, antes de su término, y la aplicación.

E-1: Ocurre un error en la consulta o lectura en la base de datos. Se informa al jugador de este hecho. Finaliza el caso de uso.

Caso de Uso 2: Resolver Test

Actores participantes: Jugador.

Condiciones de entrada: Esté creado el test.

Flujo de eventos:

1. El sistema muestra la primera pregunta del test, con sus posibles respuestas (E-1).

2. El jugador selecciona una respuesta y pulsa ok.

3. El sistema almacena la respuesta elegida y si no es la última, muestra otra pregunta (E-1) y pasa al punto 2, si no, muestra un resumen de las preguntas bien y mal contestadas y la puntuación obtenida.

4. El jugador pulsa ok para terminar.

5. El sistema actualiza la base de datos en el caso de que sea necesario modificarla por la puntuación obtenida, si no ocurre ninguna excepción con la base de datos (E-2).

Condiciones de salida: El jugador ha realizado el test, se ha actualizado la base de datos si lo ha requerido y se tiene la lista de preguntas con las respuestas dadas por el jugador.

Flujo de eventos alternativo:

En el paso 2, el jugador puede elegir volver a la pregunta anterior, si no se encuentra en la primera pregunta, y entonces el sistema mostraría la pregunta anterior en la que se encontraba el jugador.

Excepciones:

E-1: Ocurre un error en la lectura de la pregunta de la base de datos. Se informa al jugador de este hecho. Finaliza el caso de uso.

E-2: Ocurre un error en la lectura o escritura de la base de datos al actualizar las puntuaciones. Se informa del tipo de error al jugador. Finaliza el caso de uso.

En el paso 2 y 4, el jugador puede pulsar salir de la aplicación, finalizando el caso de uso y la aplicación.

Caso de Uso 3: Consultar resultados

Actores participantes: Jugador.

Condiciones de entrada: Lista de preguntas planteadas en el test con las respuestas dadas por el jugador.

Flujo de eventos:

1. El sistema muestra una lista ordenada numéricamente por la posición de las preguntas planteadas en el test y si ha sido contestada correctamente o no.

2. El jugador puede seleccionar de la lista la pregunta que quiera consultar.

3. El sistema muestra la pregunta y en función de si la respuesta fue la correcta o no (E-1):

   3.1. Correcta, el sistema muestra la respuesta correcta.
   3.2. Incorrecta, el sistema muestra la respuesta incorrecta dada por el jugador y la que sería la solución.

4. El jugador puede elegir volver al paso 1 (para seguir viendo más preguntas), o bien seguir.

5. El sistema muestra el ranking de puntuaciones correspondiente a la asignatura sobre la que se ha planteado el test, para ver cómo ha quedado tras el test (E-2).

6. El jugador pulsa ok para terminar de visualizar el ranking.

Condiciones de salida: El jugador ha podido consultar y ver su resultado del test y el ranking de puntos correspondiente a la asignatura elegida.

Excepciones:

E-1: Ocurre un error en la lectura de una pregunta en la base de dato. Se informa al jugador de este hecho. Finaliza el caso de uso.

E-2: Ocurre un error en la lectura de un ranking de la base de datos. Se informa del tipo de error al jugador. Finaliza el caso de uso.

En el paso 6 puede pulsar salir de la aplicación, terminando también el caso de uso.

Caso de Uso 4: Ver rankings

Actores participantes: Jugador.

Condiciones de entrada: La aplicación esté inicializada (estando en el menú principal) y la base de datos esté creada.

Flujo de eventos:

1. El jugador selecciona la opción Ver Rankings.

2. El sistema muestra la lista de los rankings de puntos de las asignaturas disponibles.

3. El jugador elige la asignatura que desea consultar.

4. El sistema lee de la base de datos el ranking elegido y lo muestra (E-1).

5. El usuario pulsa ok para terminar de ver el ranking.

Condiciones de salida: El jugador ha podido ver el ranking de puntos solicitado.

Excepciones:

E-1: Ocurre un error en la lectura de la base de datos. Se informa del tipo de error al jugador. Finaliza el caso de uso.

En el paso 5 puede pulsar salir de la aplicación, terminando también el caso de uso.

Caso de Uso 5: Ayuda

Actores participantes: Jugador.

Condiciones de entrada: La aplicación esté inicializada (estando en el menú principal).

Flujo de eventos:

1. El jugador selecciona la opción Ayuda.

2. El sistema muestra en pantalla un texto a modo explicativo del funcionamiento del juego.

3. El jugador pulsa ok para terminar de ver la ayuda.

Condiciones de salida: El jugador ha podido ver la ayuda del juego.

Excepciones: En el paso 3, el usuario también puede optar por salir de la aplicación, finalizando tanto el caso de uso como la aplicación.

4.3.3 Escenarios

Un caso de uso es una representación abstracta de una funcionalidad a realizar por el sistema. La representación concreta de un caso de uso se realiza mediante la creación de uno o más escenarios que muestren todas las interacciones posibles entre el sistema y sus usuarios [20].

Los escenarios son historias ficticias que describen posibles interacciones con una interfaz. Permiten a los diseñadores anticiparse a los problemas. Aunque son historias ficticias deben hacerse lo más detalladas posibles, así por ejemplo, los personajes deben tener nombres, motivaciones para usar la interfaz, deben encontrarse en entornos reales con las restricciones que ello conlleva, etc. De esta manera se facilita a los diseñadores la discusión sobre la interfaz ya que a las personas nos cuesta más trabajo discutir sobre una situación abstracta.

Esta forma de proceder fuerza a los diseñadores a considerar el rango de usuarios que va a usar el sistema y el rango de actividades por las que lo van a usar. Los escenarios permiten hacer diferentes combinaciones de usuarios y actividades de forma que se tengan en cuenta todas las posibilidades.

Un escenario esta formado por los siguientes elementos:
- Un nombre único y unívoco.
- Una descripción.
- Los actores participantes.
- El flujo de eventos.

Como se ha indicado, para cada caso de uso puede haber varios escenarios. Para nuestro caso vamos a definir un escenario para cada caso de uso principal, de tal modo que se tenga un ejemplo de las principales funcionalidades del sistema:

Escenario 1: Crear un test corto de Inglés.

Nombre: CrearTest_FranGL_Tcorto_Ingles.

Descripción: Al usuario Fran García López se le va pedir que introduzca sus iniciales y datos sobre el test que quiere realizar, y este va elegir el tipo de test corto, de la asignatura Inglés.

Actores: Fran García López.

Flujo de eventos:

1. El sistema muestra por pantalla un campo de texto para solicitar las iniciales.

2. Fran García López introduce las iniciales FGL y pulsa ok.

3. El sistema muestra los tipos de test a elegir, siendo Test corto y Test largo.

4. Fran García López selecciona el tipo de test “corto” y pulsa ok.

5. El sistema muestra las asignaturas existentes a elegir, siendo Informática e Inglés.

6. Fran García López selecciona la asignatura Inglés y pulsa ok.

7. El sistema elabora un test de 10 preguntas basado en preguntas de la asignatura de Inglés.

Escenario 2: Resolver test largo de la asignatura Informática.

Nombre: ResolverTest_ManuelRR_Largo_Informatic.

Descripción: La aplicación plantea un test del tipo “largo”, de la asignatura Informática, al usuario Manuel Ruiz Ruiz, según seleccionó con anterioridad.

Actores: Manuel Ruiz Ruiz.

Flujo de eventos:

1. El sistema muestra una pregunta de Informática con sus cuatro posibles respuestas A, B, C y D.

2. Manuel Ruiz Ruiz elige una respuesta y pulsa ok.

3. El sistema almacena la respuesta y si:

No es la última, muestra otra pregunta y pasa al punto 2.
Es la última, muestra un resumen de las preguntas bien y mal contestadas y la puntuación obtenida.

4. El sistema muestra que ha contestado bien a 8 preguntas y 12 mal y la puntuación obtenida es de 40 puntos.

5. Manuel Ruiz Ruiz pulsa ok para terminar.

6. El sistema no actualiza la base de datos con la puntuación porque no se ha obtenido la suficiente para ello.

Como exponer un test en su totalidad a modo de ejemplo resultaría excesivo por su volumen de datos, no se han dado más detalles en este escenario, aunque se refleja completamente la interacción que se produce entre el jugador y el sistema, que es lo que se busca con un escenario y para más o menos concretar un caso real que se puede dar.

Escenario 3: Consultar resultados de un test corto de la asignatura Ingles.

Nombre: Consultar_Resultados_FGL_Tcorto_Ingles.

Descripción: El usuario Fran García López consulta los resultados del test corto sobre la asignatura de Ingles que ha realizado.

Actores: Fran García López.

Flujo de eventos:

1. El sistema muestra la lista ordenada numéricamente (del 1 al 10) por la posición de las preguntas en el test y si ha sido contestada correctamente o no.

2. Fran García Lopéz selecciona de la lista la pregunta número 1.

3. El sistema lee la pregunta de la base de datos, correspondiente a la ubicada en la posición número 1 del test y la muestra y como fue mal contestada, muestra la respuesta incorrecta y la que es la solución.

4. Fran García López pulsa volver.

5. El sistema vuelve a mostrar la lista ordenada numéricamente de las preguntas en el test y si han sido o no contestadas correctamente.

6. Fran García López selecciona la pregunta número 2.

7. El sistema lee la pregunta de la base de datos, correspondiente a la ubicada en la posición número 2 del test y la muestra y como había sido bien contestada, sólo muestra la respuesta correcta.

8. Fran García López pulsa seguir.

9. El sistema lee de la base de datos el ranking correspondiente a la asignatura de Ingles y lo muestra.

10. Fran García López pulsa ok para terminar de visualizar el ranking.

Aclaración: Siguiendo la línea de lo que es un escenario, se ha hecho una invención sobre el tema de si las preguntas del test habían sido o no contestadas correctamente por el usuario Fran, pero que al fin y al cabo reflejan una situación que se puede dar en la interacción con la aplicación y más concretamente en el caso de uso que aborda este escenario.

Escenario 4: Ver ranking de la asignatura Informática.

Nombre: Ver_Ranking_MaríaDN_Informatica.

Descripción: La usuario María Díaz Navarro quiere consultar el ranking de puntos de la asignatura Informática para comprobar si aparece.

Actores: María Díaz Navarro.

Flujo de eventos:

1. La jugadora María Díaz Navarro selecciona la opción Ver Rankings.
2. El sistema muestra la lista de los rankings de puntos de las asignaturas disponibles, siendo Informática e Inglés.
3. María elige el ranking de puntos de Informática.
4. El sistema lee de la base de datos el ranking de puntos de Informática y lo muestra por pantalla.
5. María pulsa ok para terminar.

Escenario 5: Ver la ayuda del juego

Nombre: Ver_Ayuda_IsabelLF.

Descripción: La jugadora Isabel Lozano Ferrer desea consultar la ayuda del juego para ver si le aclara el funcionamiento del mismo.

Actores: Isabel Lozano Ferrer.

Flujo de eventos:

1. Isabel selecciona la opción Ayuda.

2. El sistema muestra en pantalla un texto a modo explicativo del funcionamiento del juego.

3. Isabel pulsa ok para terminar de ver la ayuda.

4.4 Diseño del Sistema

Realizar de manera adecuada cada una de las actividades que conlleva la Ingeniería del Software es, indudablemente, indispensable para el desarrollo de un proyecto software de calidad. Por lo tanto, no se puede decir que ninguna de estas actividades sea más importante que otra. Sin embargo, sí podemos decir que la actividad de diseño es la más delicada y la más laboriosa de llevar a cabo.

Esto es debido a que si no se lleva a cabo correctamente se hace imposible el codificar, de manera correcta, en la fase de implementación el modelo obtenido en el análisis del sistema, lo que puede repercutir en hacer inútil todo el esfuerzo realizado durante las primeras actividades de la Ingeniería del Software.

También se considera la más laboriosa porque las estrategias a seguir para conseguir que la traducción entre modelo y código se lleve a cabo correctamente son muy diversas y complejas.

Por tanto, se puede decir que el diseño del sistema es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema, se plantean las diversas estrategias para alcanzarlos en la actividad de implementación [19].

Sin embargo, el sistema no se suele diseñar de una sola vez sino que hay que diferenciar entre el diseño y estructura de los datos que se van a manejar y el diseño de la interfaz entre la aplicación y el usuario. Estas dos

fases del diseño no se realizan de forma consecutiva una detrás de la otra sino que lo normal es realizarlas de manera concurrente y finalizarlas a la vez.

4.4.1 Diseño de los datos

El objetivo de esta fase del diseño software es determinar la estructura que poseen cada uno de los elementos de información del sistema, es decir, la estructura de los datos sobre los que se va a trabajar.

En nuestro caso y teniendo en cuenta que se trata de un prototipo de juego educativo orientado a dispositivos móviles, donde dichos dispositivos poseen unas limitaciones tanto en memoria como en capacidad de computación reducidas, no debemos abordar una excesiva complejidad en los datos que se van a manejar, ya que por un lado no nos lo permitirían estos dispositivos (no soportan todos los tipos de datos que suelen soportar un PC) y además sufriríamos un detrimento de las prestaciones en cuanto a velocidad de reacción y refresco del dispositivo. Por lo tanto, lo que intentaremos es simplificar al máximo y quedarnos con los datos realmente importantes o que más interesen para conseguir nuestro objetivo.

Teniendo en cuenta lo comentado, los elementos que vamos a considerar son los siguientes:

- Las **preguntas**, que sabemos que pertenecen a una asignatura concreta, la pregunta en sí, es decir, lo que sería la formulación de la misma, las cuatro opciones que se van a dar como posibles respuestas y la solución de la misma.

- Las **asignaturas**, de las que sólo contamos en un principio con su nombre.

- Las **iniciales** de los usuarios o jugadores. No es objetivo en este prototipo mantener una base de datos sobre los jugadores.

- La **puntuación**, que ha obtenido un jugador tras finalizar un test sobre una asignatura determinada.

- **Ranking** de puntuaciones, que dependerán de las asignaturas que se contemplen. Estos ranking contendrán las iniciales de los jugadores con su puntuación obtenida, ordenados por posición de mayor a menor, en función de la puntuación (un aspecto que está contemplado en los requerimientos funcionales es que los rankings de puntuaciones no sobrepasarán el tamaño de cinco registros por asignatura).

Una vez determinados cuales son los elementos de información del sistema, se deben obtener sus representaciones en forma de tablas de una base de datos [22]. Para ello, se debe realizar en primer lugar un diseño conceptual de la base de datos para, posteriormente, obtener las tablas requeridas y donde se puede utilizar el modelo Entidad-Relación.

4.4.1.1 Modelo Entidad-Relación

El **modelo Entidad-Relación** (también conocido por sus iniciales: E-R) es una técnica de modelado de datos que utiliza diagramas entidad-relación. No es la única técnica de modelado pero si es la más extendida y utilizada.

Un diagrama entidad-relación esta compuesto por tres tipos de elementos principales:

- **Entidades**: Objetos (cosas, conceptos o personas) sobre los que se tiene información. Se representan mediante rectángulos etiquetados en su interior con un nombre. Una instancia es cualquier ejemplar concreto de una entidad.

- **Relaciones**: Interdependencias entre uno o más entidades. Se representan mediante rombos etiquetados en su interior con un verbo. Si la relación es entre una entidad consigo mismo se denomina reflexiva, si es entre dos entidades se denomina binaria, ternaria si es entre tres y múltiple si es entre más.

- **Atributos**: Características propias de una entidad o relación. Se representan mediante elipses etiquetados en su interior con un nombre.

En los diagramas entidad-relación también hay que tener en cuenta otros aspectos como pueden ser:

- **Entidades débiles**: Son aquellas que no se pueden identificar unívocamente sólo con sus atributos, sino que, necesitan estar relacionadas con otras entidades para existir. Se representan con dos rectángulos concéntricos de distinto tamaño con un nombre en el interior del más pequeño.

- **Cardinalidad de las relaciones**: Existen tres tipos de cardinalidades de una relación según el número de instancias de cada entidad que involucren:

    - Uno a uno, una instancia de la entidad A se relaciona solamente con una instancia de la entidad B. (1:1)
    - Uno a muchos: cada instancia de la entidad A se relaciona con varias de la entidad B. (1:*).
    - Muchos a muchos: cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B. (*:*)

- **Claves**: cada entidad de un diagrama entidad-relación debe tener una clave, debe estar formada por uno o más de sus atributos.

Una vez conocidos los elementos que forman parte de un diagrama entidad-relación podemos empezar a desarrollar el modelo entidad-relación. Los pasos a seguir son los siguientes:

1. Convertir el enunciado del problema en un Esquema Conceptual del mismo.

2. Convertir este Esquema Conceptual (o EC) en uno más refinado conocido como Esquema Conceptual Modificado (ECM).

3. Obtener las tablas de la base de datos a partir del Esquema Conceptual Modificado.

4.4.1.2 Normalización en el modelo Entidad-Relación

La normalización es un proceso consistente en imponer a las tablas ciertas restricciones mediante una serie de transformaciones consecutivas. Con ello se asegura que las tablas contengan los atributos necesarios y suficientes para describir la realidad de la entidad que representan, separando aquellos que pueden contener información cuya relevancia permite la creación de otra nueva tabla.

Para asegurar la normalización, Codd estableció tres formas normales, las cuales hacen que una base de datos (si las cumple) esté normalizada. Estas formas normales son:

- **Primera forma normal** (F<1): Una tabla esta en FN1 si todos los atributos no clave, dependen funcionalmente de la clave, o lo que es lo mismo, no existen grupos repetitivos para un valor de clave.

- **Segunda forma normal** (F<2): Una tabla está en FN2 si está en FN1 y además todos los atributos que no pertenecen a la clave dependen funcionalmente de forma completa de ella. De esta definición se desprende que una tabla en FN1 y cuya clave esta compuesta por un único atributo está en FN2.

- **Tercera forma normal** (F<3): Una tabla está en FN3 si está en FN2 y además no existen atributos no clave que dependan transitivamente de la clave.

4.4.1.3 Esquema conceptual del proyecto

Necesitamos convertir nuestros elementos en entidades, con sus atributos y relaciones. En nuestro caso, y con las necesidades que queremos cubrir con este prototipo, tenemos entonces como entidades a Preguntas, Asignaturas y Rankings, las relaciones que se establecen son:

R1: Pertenece, cuya cardinalidad es (1:N), donde queremos expresar que a una asignatura pueden pertenecer N preguntas y donde una pregunta pertenece a una sola asignatura.

R2: Tiene, cuya cardinalidad es de (1:1), ya que una asignatura tiene solo un ranking de puntuaciones, y un ranking es de una sola asignatura.

Finalmente y considerando los atributos correspondientes a cada entidad, el Esquema Conceptual queda como en la figura 4.3 puede apreciarse:

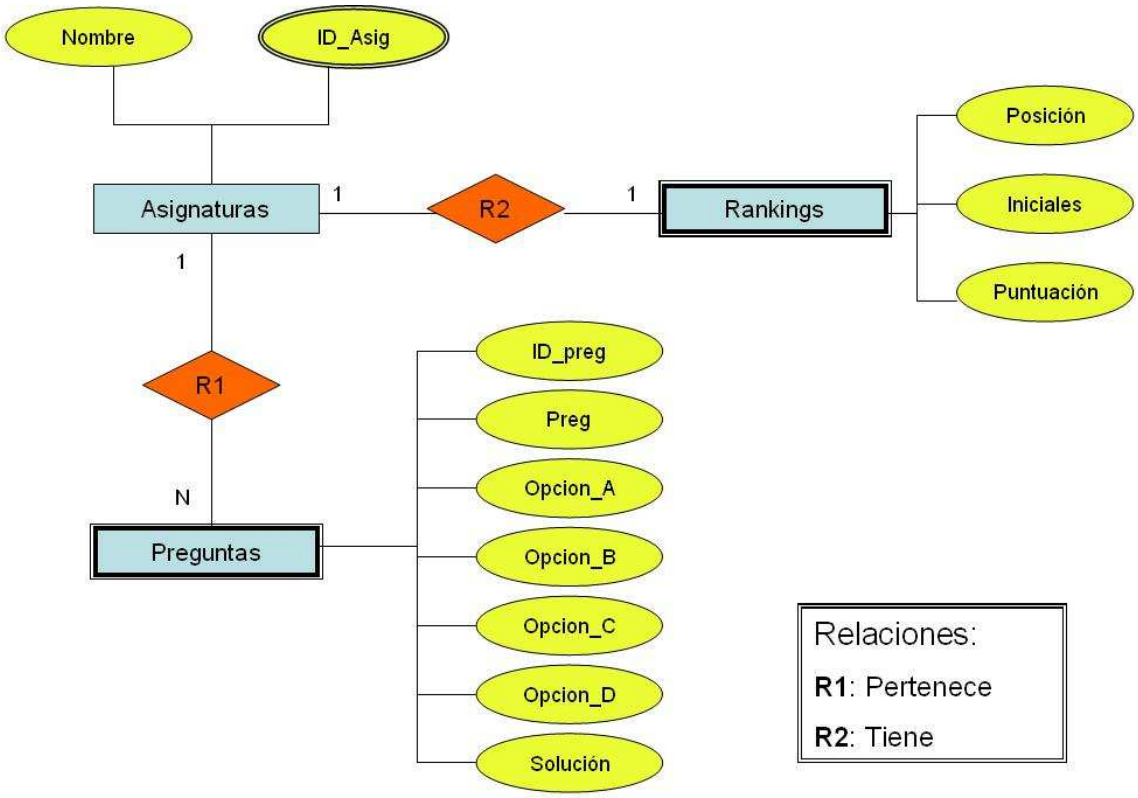


Figura 4.3: Esquema Conceptual

4.4.1.4 Esquema conceptual modificado

Para obtener el Esquema Conceptual Modificado a partir del Esquema Conceptual se deben hacer los cambios que siguen a continuación:

- Eliminar todas las entidades débiles.

- Eliminar las relaciones de muchos a muchos.

- Eliminar las relaciones con atributos que haya en nuestro Esquema Conceptual.

Como podemos observar en el Esquema Conceptual, no tenemos ni relaciones de muchos a muchos, ni relaciones con atributos, pero sí nos encontramos con dos entidades débiles, que son Preguntas y Rankings. Son débiles puesto que los atributos que poseen no son suficientes para establecer una clave que los identifique unívocamente. Así pues, para eliminar las entidades débiles, se le añaden los atributos clave de las entidades con las que tenga relación.

Por tanto, la entidad Preguntas adquirirá el atributo ID_Asig de la entidad Asignaturas y junto con el atributo ID_Preg, formaran la clave de la entidad, puesto que sólo con ID_Asig tampoco sería capaz de identificarla. La entidad Rankings también adquiere el atributo ID_Asig, ya que sólo tiene relación con dicha entidad y junto con el atributo Posición, formará la clave de dicha entidad.

Tras los pasos anteriormente realizados, el Esquema Conceptual Modificado (ECM) quedaría como se muestra a continuación, en la figura 4.4:

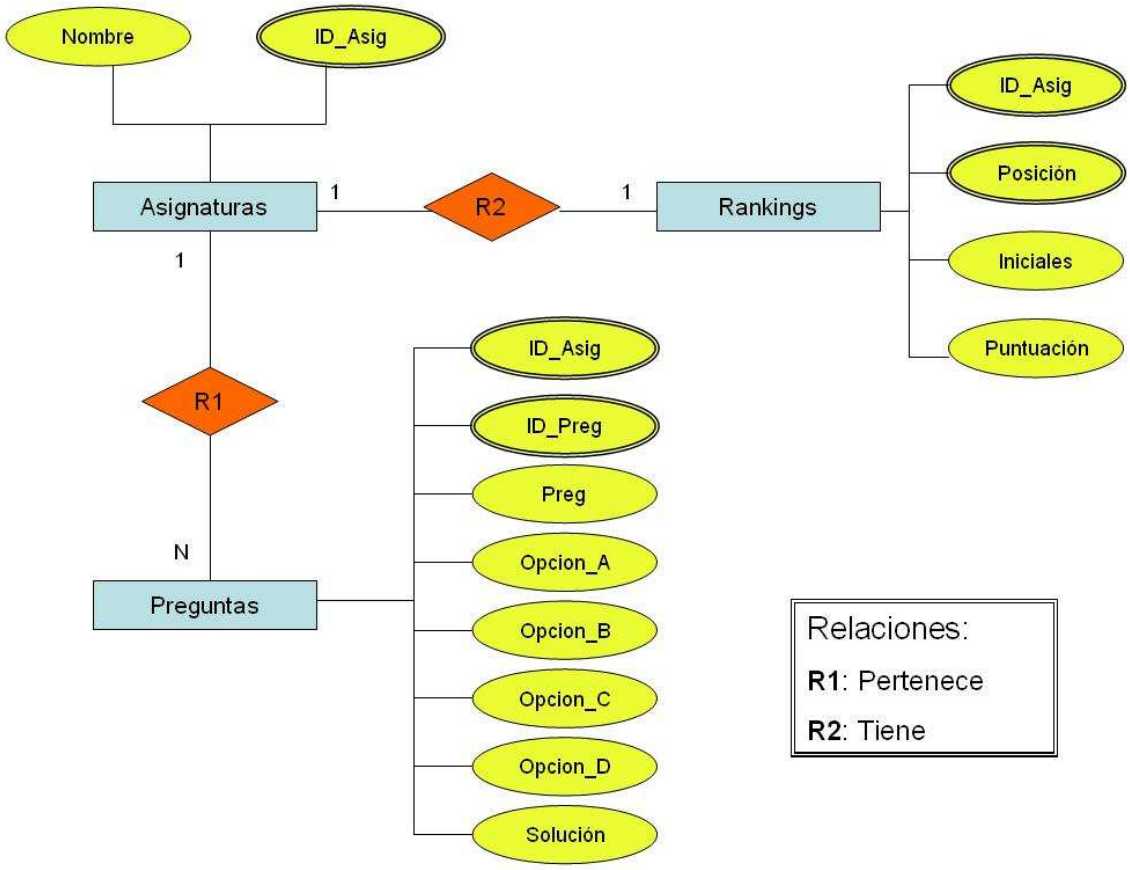


Figura 4.4: Esquema Conceptual Modificado

4.4.1.5 Tablas de la aplicación

Una vez obtenido el ECM, podemos pasar a identificar las tablas de las que se va a componer la base de datos, considerando que:

- Cada entidad del ECM se transforma directamente en una tabla.

- Los atributos de una entidad se convierten en los campos de las tablas respectivas.

Así pues, obtenemos las tablas:

- La entidad Preguntas pasa a ser la tabla PREGUNTAS.
- La entidad Asignaturas, da lugar a la tabla ASIGNATURAS.
- La entidad Rankings, se convierte en la tabla RANKINGS.

Para mejorar el entendimiento de lo que va a contener cada tabla de la base de datos, vamos a desglosar cada tabla con sus campos correspondientes:

ASIGNATURAS		
CAMPO	**TIPO**	**DESCRIPCION**
ID_Asig	Int	Identificador de la asignatura
Nombre	String	Nombre completo de la asignatura

Tabla 4.1: Tabla ASIGNATURAS.

PREGUNTAS		
CAMPO	**TIPO**	**DESCRIPCION**
ID_Asig	Int	Identificador de la asignatura, a la que pertenece
ID_Preg	Int	Identificador de la pregunta.
Preg	String	Texto de la pregunta
Opcion_A	String	Primera respuesta opcional
Opcion_B	String	Segunda respuesta opcional
Opcion_C	String	Tercera respuesta opcional
Opcion_D	String	Cuarta respuesta opcional
Solucion	Int	Indica el número de la respuesta correcta

Tabla 4.2: Tabla PREGUNTAS.

RANKINGS		
CAMPO	**TIPO**	**DESCRIPCION**
ID_Asig	Int	Identificador de la asignatura, a la que hace referencia
Posicion	Int	Indica la posición en el ranking de puntuación
Iniciales	String	Indica las iniciales del jugador
Puntuación	Long	Es la puntuación obtenida

Tabla 4.3: Tabla RANKINGS.

Sin embargo, como pudimos ver en el apartado de RMS (Record Management System) del Capítulo 3, los dispositivos móviles presentan unas peculiaridades y limitaciones en lo referente a almacenamiento de datos en memoria permanente, por lo que es necesario ajustar nuestro modelo de datos a las restricciones que se nos plantea. Como se comentó en el apartado 3.3.6 (Capítulo 3), el RMS es un pequeño sistema de base de datos muy sencillo, pero que nos permite añadir información en una memoria no volátil del dispositivo. El RMS no tiene nada que ver con un SGBD (Sistema de Gestión de Base de Datos) habitual, ya que el RMS no tiene implementado un lenguaje de consulta como SQL, el acceso y almacenamiento de la información se hace a más bajo nivel. En una base de datos RMS, el elemento básico es el registro (Record), siendo la unidad de información más pequeña que puede ser almacenada y estos registros son almacenados en un Record Store que puede visualizarse como una colección de registros, como puede apreciarse en la figura 3.9, del Capítulo 3, donde viene claramente la estructura completa de un Record Store. Cuando almacenamos un registro en el Record Store, a éste se le asigna un identificador único que identifica unívocamente al registro. Así pues, las aplicaciones hacen uso de esta herramienta para la gestión de sus datos.

Teniendo en cuenta los aspectos expuestos, nosotros necesitaremos utilizar para cada tabla de la base de datos, un Record Store para almacenar los datos correspondientes. Por lo que, siguiendo este planteamiento, para la tabla ASIGNATURAS emplearemos un Record Store, llamado igualmente Asignaturas, donde se almacenará el ID_Asig, y el nombre de la asignatura correspondiente.

Para la tabla PREGUNTAS, tendremos otro Record Store, que se llamara BDPreguntas, donde por cada registro del mismo se guardará, de forma concatenada, todos los datos correspondientes a una pregunta en el campo Datos, a excepción del ID_Preg, que nos bastará con usar el Record_ID del registro.

Para la tabla Rankings, hemos realizado unas modificaciones debido a que sobre esta tabla se van a realizar continuamente operaciones de inserción, borrado y ordenamiento. Al no existir un lenguaje de manipulación de datos dichas operaciones se realizan a bajo nivel (apartado 3.3.6, Capítulo 3), y el ordenamiento hay que implementarlo para mantener el ranking clasificado de mayor a menor según las puntuaciones. Todo esto supone una carga extra

para nuestro dispositivo y aplicación. Sólo para este caso se va a crear una tabla de rankings diferente, es decir, se van a crear dos Record Store, que hacen referencia cada uno exclusivamente a un ranking de puntos de una asignatura concreta, llamados RankingIngles y RankingInformatica, donde cada una tendrá como máximo el tamaño de cinco registros y se mantendrán ordenados en función de las puntuaciones obtenidas por los usuarios en los test referentes a las asignaturas correspondientes. La diferencia que se produce en tamaño, en bytes, no es realmente significativa, ya que al tenerlos por separado, mantenemos menos campos de datos. No nos hace falta almacenar el atributo Posición, puesto que el registro solo tiene cinco registros y estos se ordenarán por la puntuación, pero sí supone en tiempo de ejecución mantener un solo Record Store con todos los rankings de puntuaciones ordenados.

4.4.2 Diseño de la interfaz

En esta fase del diseño del sistema software se define cual va a ser la apariencia visual de la aplicación, es decir, se define la interfaz visual entre el usuario y la aplicación. Sin duda, realizar un buen diseño de la interfaz resulta primordial ya que ésta debe presentarse atractiva al usuario de la aplicación pero, a la vez, le debe de resultar fácil de entender y trabajar sobre ella [21].

Este aspecto, se puede presentar como un reto importante en nuestro caso ya que, por un lado, pretendemos que nuestro prototipo de juego educativo vaya dirigido a un diverso número de dispositivos móviles, donde cada uno presenta una interfaz propia, con sus peculiaridades y limitaciones inherentes, marcada por ejemplo por el tamaño de pantalla y su rango cromático, la distribución y forma de visualizar los elementos en pantalla, la distinta ubicación y funcionalidad del teclado, o por las características de los sonidos que son capaces de emitir, monofónicos o polifónicos, pero donde se busca que la interacción que se produzca entre el usuario y la aplicación, sea lo más semejante posible en todos los casos e igualmente intuitiva, fácil de entender y usar.

Ahora bien, la tecnología J2ME, a través del perfil MIDP, nos permite definir diversos aspectos gráficos, como diseñar interfaces de usuario, donde queremos resaltar que se hace una división entre interfaces de usuario definidas de alto nivel y de bajo nivel:

- Las **interfaces de usuario de alto nivel** usan componentes tales como botones, cajas de texto, formularios, etc, elementos que son implementados por cada dispositivo y la finalidad de usar las APIs de alto nivel es su portabilidad. Al usar estos elementos, se pierde el control del aspecto de nuestra aplicación ya que la estética o guía de estilo de estos componentes depende del dispositivo donde se ejecute. En cambio, usando estas APIs de alto nivel ganaremos un alto grado de portabilidad de la aplicación entre distintos dispositivos.

- Con las **interfaces de usuario de bajo nivel**, se tendrá un control completo sobre los recursos del dispositivo y podremos controlar eventos de bajo nivel como, por ejemplo, el rastreo de pulsaciones de teclas. Nos permitirán crear aplicaciones muy variadas en lo que se refiere al aspecto gráfico y por lo general pueden ser más vistosas. Aunque su programación se vuelve más tediosa y aún más importante es que hay que tener en cuenta las peculiaridades del dispositivo al que va dirigido. Por lo que, la aplicación pierde portabilidad entre distintos dispositivos [20,21].

Teniendo en cuenta las distintas ventajas e inconvenientes que se presentan al diseñar interfaces de usuario para aplicaciones dirigidas a dispositivos móviles, con cada una de las divisiones expuestas y dado que el objetivo marcado para nuestra aplicación es que sea compatible con el mayor número posible de dispositivos. Nos decantamos por usar para nuestro diseño una interfaz de usuario de alto nivel, ya que es la que nos brinda una mayor compatibilidad, pero además, conseguimos otro objetivo relevante en el diseño de una interfaz, que es que el usuario se va sentir familiarizado desde el principio con la interfaz de la aplicación, ya que es la que su propio dispositivo móvil utiliza, tanto en la asignación de las funciones del teclado, como el aspecto visual de menús y demás elementos gráficos.

Así pues, en todo diseño de una interfaz, hay que definir una serie de aspectos, entre los que se destaca el estilo de la aplicación, las metáforas, las pantallas, los caminos de navegación y secuencias de diálogo, que en nuestro caso van estar condicionados por la interfaz proporcionada por el dispositivo. Pon tanto, tendremos que el estilo, las metáforas y parte del diseño de las pantallas vendrán ya definidos por la interfaz propia del dispositivo móvil sobre el que se ejecute. El diseño final de las pantallas, en lo referente a qué

elementos mostrarán y parte de la distribución que presentarán los mismos, junto con los caminos de navegación y secuencias de diálogo, dependerán de nuestro criterio.

4.4.2.1 Estilo

El estilo trata acerca de la forma en que el contenido debe ser presentado al usuario, como es la fuente del texto, colores, alineado, cabeceras, etc. Para ello se definen guías de estilo, para mantener una consistencia en el estilo para toda la interfaz de la aplicación. Cuando van a participar en el diseño de una interfaz varios diseñadores, tener definida una guía de estilo ayuda a que no se produzcan incoherencias en la interfaz [21].

Sin embargo, a pesar de lo que pueda parecer en un principio, también es de gran utilidad definir una guía de estilo cuando sólo hay un diseñador encargado de la interfaz. Esto es debido a varias razones:

- En ocasiones es posible que mantener la coherencia y consistencia de una interfaz, si esta es muy grande o muy ambiciosa, sea complicado incluso si sólo hay un diseñador.

- El diseñador primitivo puede, por las más diversas razones, abandonar el diseño. Por tanto, es de gran utilidad para sus sustitutos contar con una guía de estilo predefinida para no tener que empezar desde cero de nuevo. Esto es también aplicable cuando no es el diseñador original el que se encarga de la actualización o el mantenimiento de la interfaz.

En nuestro caso, como se ha comentado, la mayor parte de los aspectos del estilo de la interfaz dependerá de la guía de estilo propia de la interfaz del dispositivo móvil sobre el que se ejecuta la aplicación, como es la fuente del texto, el color e incluso muchas veces el alineamiento sobre la pantalla. Sólo nos quedan algunos detalles que podemos definir, que son:

- Para el encabezado superior de cada pantalla, se indicará el nombre de la aplicación, que aparecerá en mayúsculas, seguido del texto que nos indica en qué pantalla nos encontramos.
- Las opciones, como Menú, Atrás, Ranking, Ver, vendrán con la primera letra en mayúscula, las demás en minúscula, a excepción de

la opción OK, que aparecerá totalmente en mayúscula. En el caso de los menús, seguirá el mismo criterio.
- Las preguntas y las respuestas de las mismas vendrán en formato tipo oración.
- El texto que aparece en la ayuda también se define con el formato tipo oración.
- Sólo para las pantallas Puntuaciones y Rankings, por su peculiaridad, se ha definido el fondo de color azul claro, con el texto centrado y de color blanco, menos para el encabezado, que es de color negro.
- En los demás casos, el texto siempre aparece alineado a la izquierda y sin justificar, como puede aparecer en un editor de texto, según ya la interfaz del dispositivo.

4.4.2.2 Metáforas

Una metáfora es el empleo de un objeto con un significado o dentro de un contexto diferente al habitual. Al diseñar una interfaz gráfica, la utilización de metáforas resulta muy útil ya que permiten al usuario, por comparación con otro objeto o concepto, comprender de una manera más intuitiva las diversas tareas que la interfaz permite desarrollar.

Al igual que pasa en el ámbito de la literatura, para que una metáfora cumpla con su cometido, el desarrollador de la aplicación y el usuario final de ésta deben tener una base cultural similar. Es muy posible que el uso de un icono de manera metafórica sea entendido de una manera por el usuario occidental y de otra bien distinta por un usuario oriental. Hay que intentar por lo tanto, que las metáforas empleadas sean lo más universales posibles para que, así sean comprendidas a la perfección por la mayor parte de los usuarios potenciales.

Las aplicaciones de escritorio, como las destinadas al Sistema Operativo Windows, suelen seguir una Guía de Estilo y utilizan una serie de metáforas con las que el usuario está plenamente familiarizado (por ejemplo, unos prismáticos en un icono, establece la función de buscar). Pero las metáforas no sólo dependen del tipo de aplicación (escritorio, Web, etc) sino también del ámbito de la misma. Por ejemplo, el carrito de la compra es una metáfora

conocida por todos pero, si nuestra aplicación no va a vender nada al usuario no resulta conveniente utilizarla ya que puede confundir.

Las metáforas, en nuestro caso, vendrían definidas por la interfaz del dispositivo móvil, aunque también hay que comentar que en este caso no se ha visto necesario el uso de metáforas, puesto que todas las opciones y menús de la aplicación serán mostradas en forma de texto, no se usarán iconos.

4.4.2.3 Pantallas de la aplicación

En este apartado vamos a definir la estructura de nuestra interfaz con el usuario. Mediante la elaboración de pantallas se pretende esbozar lo que será la interfaz de usuario de nuestra aplicación. Dichas pantallas no expresan el diseño final, simplemente una idea de lo que será nuestro sistema. Estas pantallas serán susceptibles de cambio durante el proceso de implementación y además, como ya se ha comentado anteriormente, el aspecto final puede diferir mucho de un dispositivo a otro, en función de su interfaz [21].

Usualmente para diseñar las pantallas se utilizan los prototipos de papel, que son una forma de crear una imagen palpable de lo que será una futura aplicación. Su creación y manipulación es rápida y elástica. Además permite a los usuarios imaginarse lo que será la futura aplicación en funcionamiento. Sin embargo, nosotros disponemos de una herramienta muy útil como el emulador predeterminado de aplicaciones de dispositivos móviles, desarrollado por Sun Microsystems y que se puede lanzar desde el entorno de desarrollo de Netbeans. Este emulador permite a los usuarios ejecutar previamente su aplicación antes de descargarla en su dispositivo móvil, para ver de una forma aproximada, lo que será su futura aplicación en funcionamiento, y permitiendo depurar los posibles fallos que pueda tener, aunque el aspecto final realmente dependerá de la interfaz del dispositivo sobre el que se ejecute, pero al menos cumple con el objetivo de mejorar y depurar el diseño de la interfaz y el funcionamiento general de la aplicación. Con él, mostraremos las diversas pantallas que se diseñarán y que compondrán nuestra aplicación.

- En la figura 4.5, tenemos de izquierda a derecha, la Pantalla de Inicio, la Pantalla Principal y la Pantalla de Menú Principal:

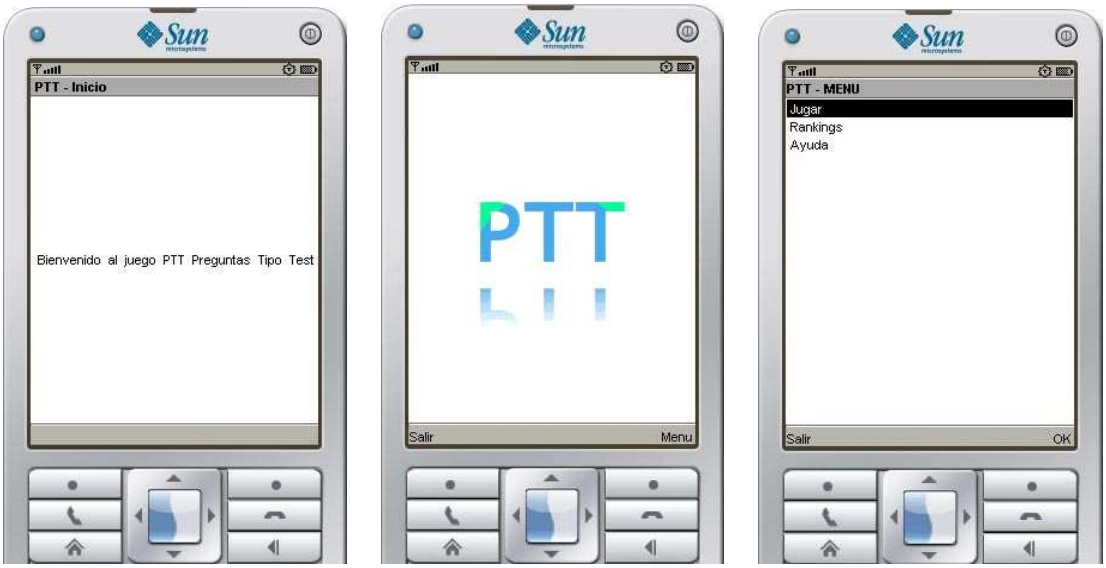


Figura 4.5: Pantalla Inicio, Pantalla Principal y Pantalla Menú Principal

- Igualmente, de izquierda a derecha, tenemos la Pantalla Iniciales, Pantalla Tipo Test y Pantalla Asignaturas en la figura 4.6:

Figura 4.6: Pantalla Iniciales, Pantalla Tipo Test y Pantalla Asignaturas

- En la figura 4.7, presentamos de izquierda a derecha, Pantalla Test, Pantalla Puntuaciones y Pantalla Menú Resultados:

Figura 4.7: Pantalla Test, Pantalla Puntuaciones y Pantalla Menú Resultados.

- Las pantallas Pantalla Solución, Pantalla Menú Ranking, Pantalla Ranking, mostradas en la figura 4.8:

Figura 4.8: Pantalla Solución, Pantalla Menú Rankings, Pantalla Ranking

- Y finalmente, tenemos la Pantalla Ayuda y Pantalla Salir en la figura 4.9:

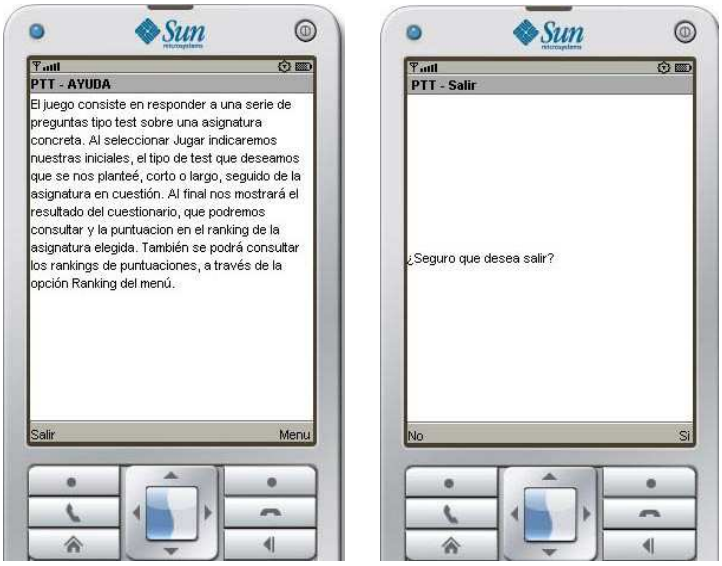


Figura 4.9: Pantalla Ayuda y Pantalla Salir.

4.4.2.4 Caminos de navegación

Hasta este momento tenemos un diseño visual de la interfaz estática, es decir, cada pantalla diseñada individualmente, pero no tenemos una idea de si en el conjunto de la interacción, la acción va a transcurrir de forma fluida y comprensible para el usuario. Para ello vamos a diseñar la interfaz en movimiento y comprobar que es usable.

Para estudiar los caminos de navegación se empleará una herramienta llamada **storyboard**, que consiste en mostrar, a modo de secuencia, las diferentes pantallas por las que se va pasando al realizar el usuario una determinada acción sobre la aplicación [21].

El procedimiento es el siguiente: se sitúan capturas de las pantallas de la interfaz unidas mediante flechas para indicar el camino que sigue la interacción. La posición de origen de las flechas debe ayudar a entender cuál es el elemento que ha desencadenado el paso de una pantalla a otra. Los storyboards también están muy ligados a los escenarios anteriormente vistos.

El storyboard sirve de prototipo para ser evaluado por el usuario y poder introducir correcciones en fases tempranas, ya que cuanto más tiempo se tarde en validar una interfaz, más coste de tiempo y trabajo acarreará.

A continuación, mostramos los storyboards para las acciones más importantes que se pueden llevar a cabo en nuestro sistema:

- Storyboard Jugar.
- Storyboard Consultar Resultados. Tenemos que partir de la base de que se ha terminado de resolver un test y se quiere ver los resultados y las distintas opciones que ofrece la aplicación.
- Storyboard Ver Ranking Informática.
- Storyboard Ver Ayuda.

Para evitar crear unos storyboard excesivamente complejos, la opción de Salir en cada una de las pantallas se ha obviado, por ser algo sencillo de comprender. Una vez diseñados los storyboard, deberían ser validados para comprobar que realmente la aplicación es usable.

Al igual que hemos utilizado el emulador de Sun Microsystems para mostrar el diseño de las pantallas de la aplicación, lo haremos para los storyboards.

El **Storyboard Jugar** se muestra en la figura 4.10:

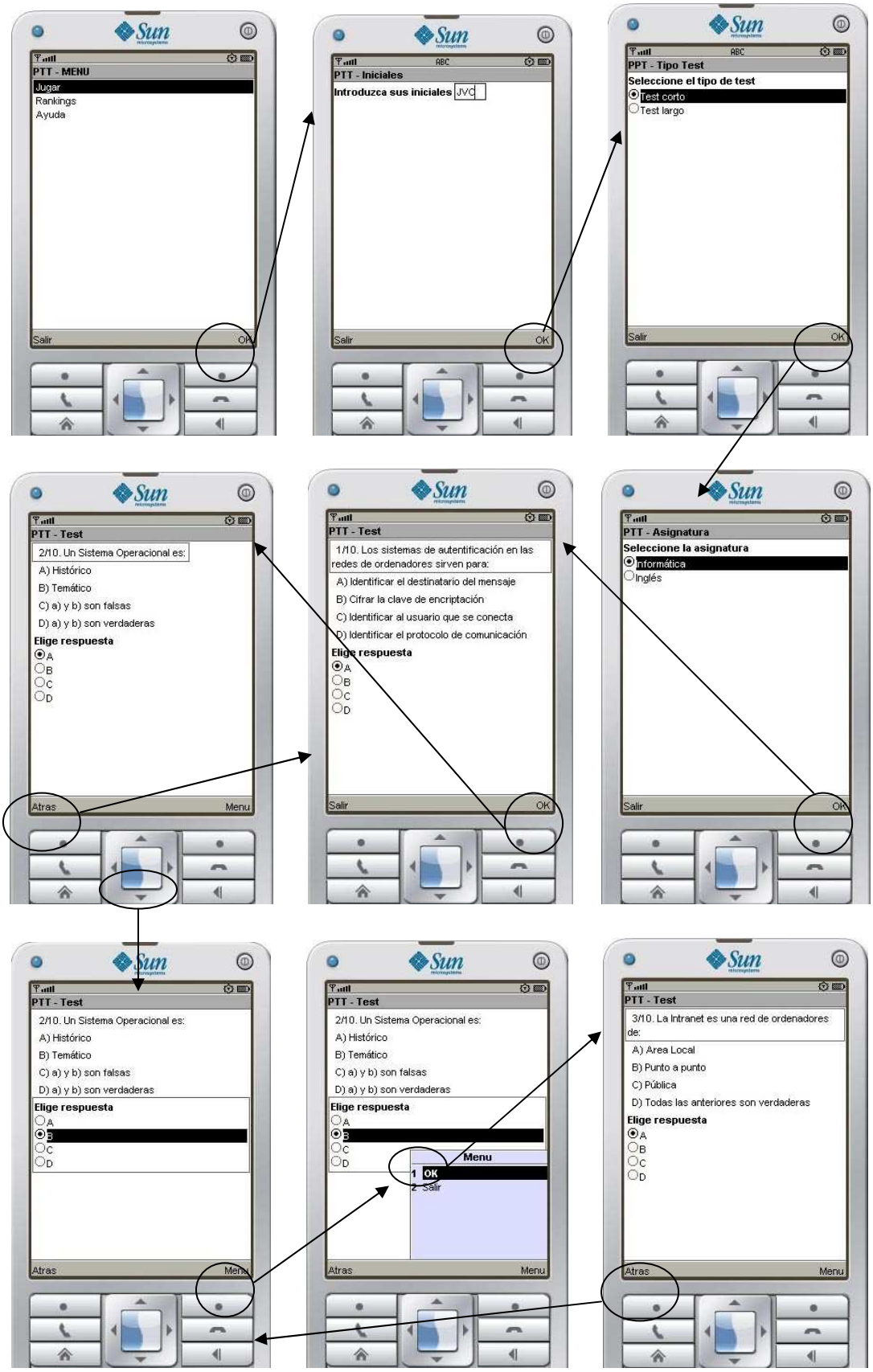


Figura 4.10: Storyboard Jugar.

Cuando se terminen de contestar todas las preguntas, se mostrará un resumen de las preguntas que han sido bien y mal contestadas y la puntuación obtenida, que corresponde con la Pantalla Puntuaciones.

A continuación, se muestra en la figura 4.11 el **Storyboard Consultar Resultados**:

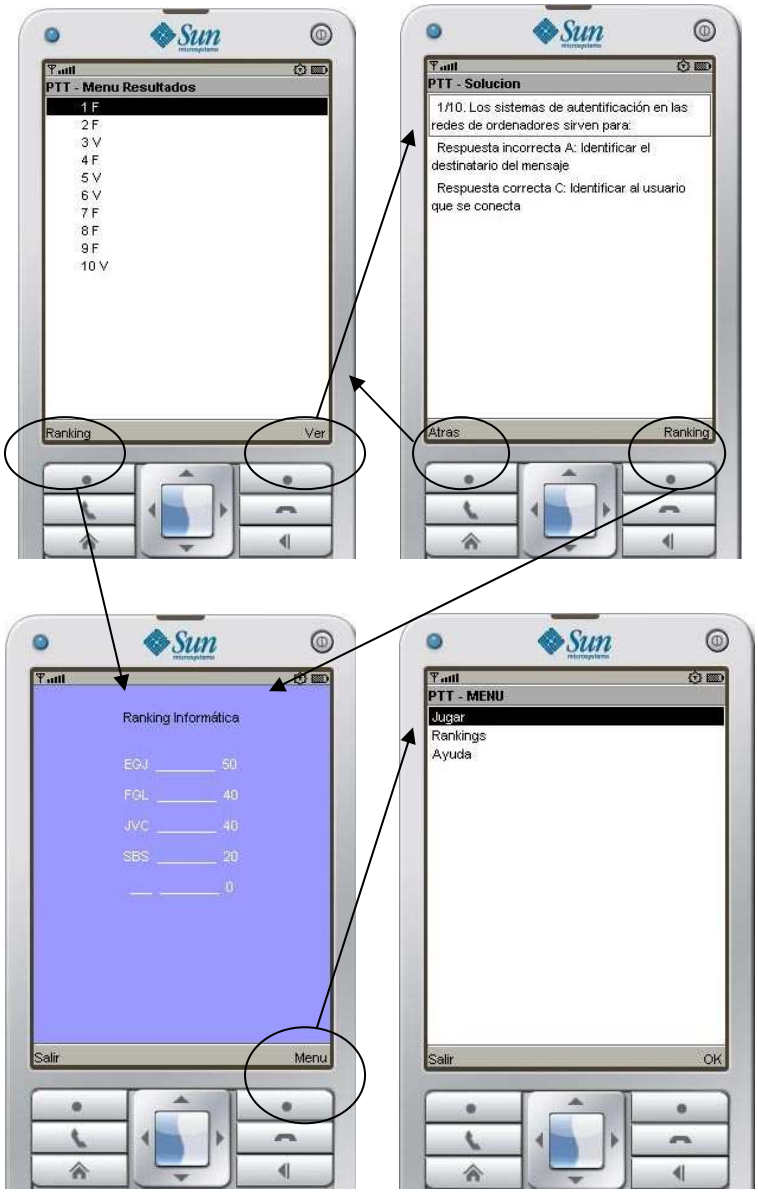


Figura 4.11: Storyboard Consultar Resultados.

En la figura 4.12, se presenta el Storyboard Ver Ranking Informática:

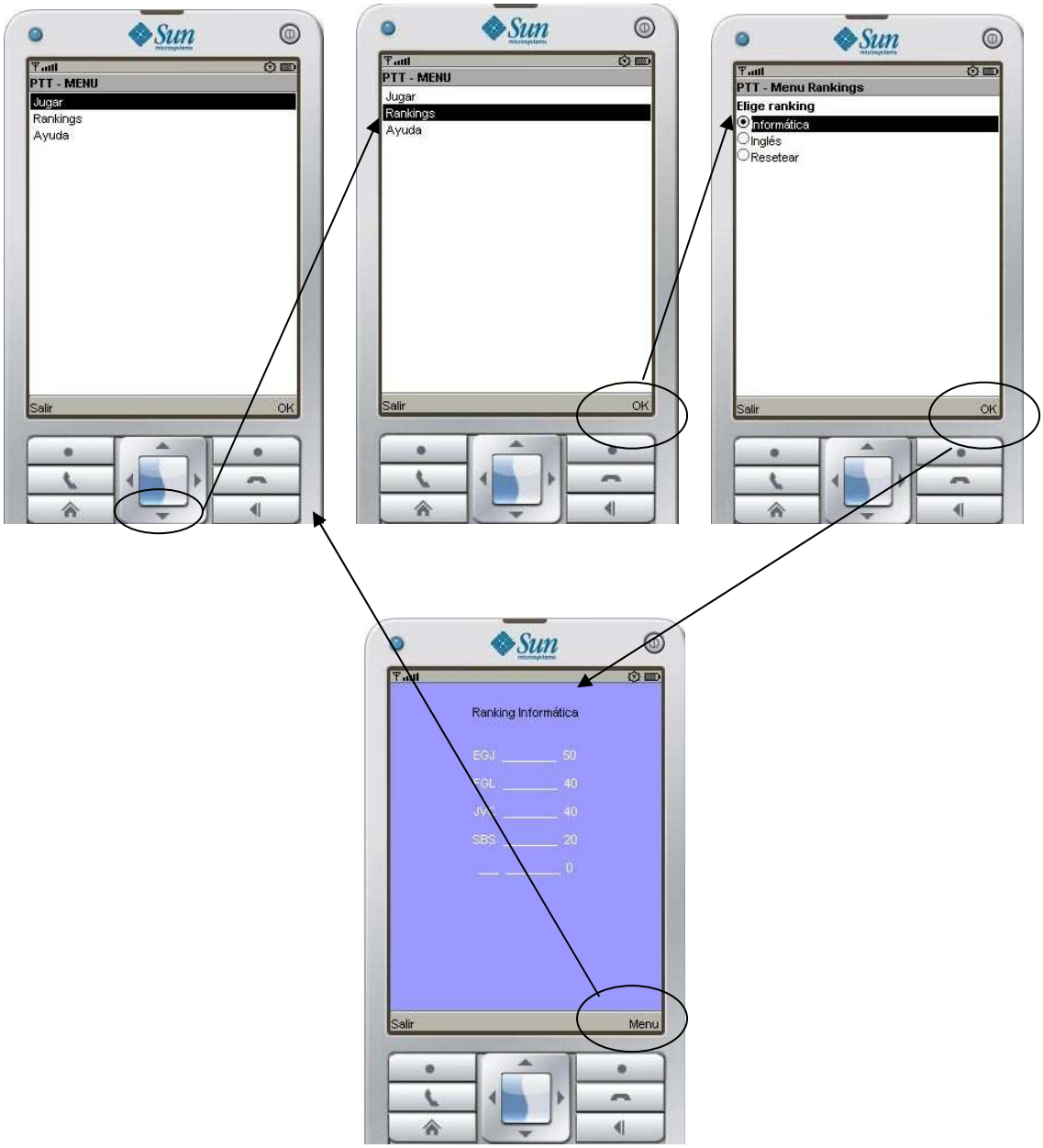


Figura 4.12: Storyboard Ver Ranking Informática.

Finalmente, en la figura 4.13, se muestra el **Storyboard Ver Ayuda:**

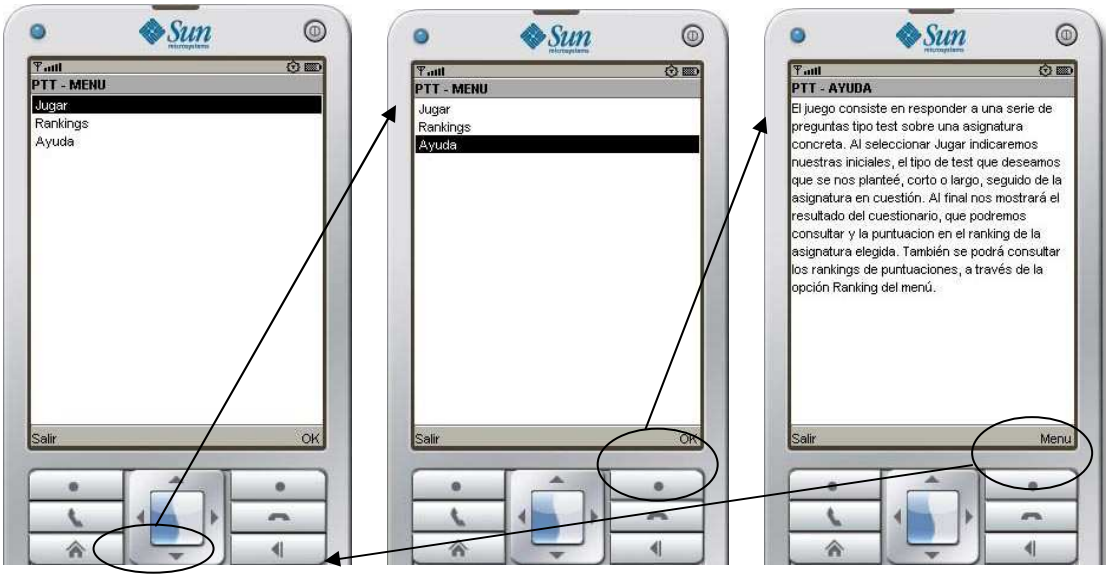


Figura 4.13: Storyboard Ver Ayuda.

4.5 Implementación

La implementación es la actividad final de la Ingeniería del Software, aquella en la que el modelo obtenido en las actividades anteriores se debe transformar en código fuente. Para ello se debe ser cuidadoso en la elección del lenguaje de programación empleado para la codificación y de la herramienta utilizada para generarla.

La elección del lenguaje de programación así como el de la herramienta utilizada para su desarrollo viene especificado desde la definición del proyecto y en el Capítulo 3 se ha tratado números aspectos sobre el lenguaje y sobre la implementación de aplicaciones para dispositivos móviles con dicho lenguaje.

4.5.1 Tipo de arquitectura de la aplicación

En nuestro caso, se trata de una sencilla aplicación destinada a una gran variedad de dispositivos móviles, los cuales deben disponer de la Plataforma Java correspondiente. Esta plataforma, como ya se comentó anteriormente en el Capítulo 3, es intermediaria entre la aplicación y el sistema operativo del

dispositivo, la cual permite que una aplicación implementada con el lenguaje J2ME se pueda ejecutar en este tipo de dispositivos, con características muy distintas entre ellos. Concretamente, esta aplicación va dirigida a los dispositivos que posean el perfil MIDP 2.0 (perfil para dispositivos de información móvil versión 2.0). Actualmente es el que más dispositivos móviles soporta, aunque no sea la versión más actual.

La aplicación ha de ser descargada en el dispositivo móvil, por alguna de las vías de comunicación que presente, como puede ser por el puerto USB, por Bluetooth, infrarrojos, por WIFI o acceso a Internet. Según el dispositivo móvil, puede ser que en el mismo proceso de descarga se realice la instalación, o bien, haya que localizar los archivos con extensión .jar y .jad, concretamente JuegoPTT.jar y JuegoPTT.jad y proceder a la instalación. Esta información se ampliará en el Anexo A, donde se documenta el manual de instalación.

4.5.2 Lenguajes de programación utilizados

Como ya se ha especificado con anterioridad, tanto en la definición del propósito y objetivos del proyecto y en el Capitulo 3 de esta memoria, la implementación de esta aplicación se ha realizado usando Java J2ME. Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. El lenguaje en sí mismo es similar en sintaxis a C y C++, pero tiene un modelo de objetos más sencillo y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. En este caso J2ME es una versión reducida de J2SE, que se puede traducir como Plataforma Java, Edición Estándar.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

No vamos a entrar en más detalles, ya que como se ha puntualizado, se ha tratado de forma detallada con anterioridad.

4.5.3 Herramienta de desarrollo

Como herramienta de desarrollo se ha elegido el entorno de desarrollo Netbeans 6.5, que es una plataforma para el desarrollo de aplicaciones de escritorio usando Java junto a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.

Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es una herramienta de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Además de la herramienta Netbeans, también se ha empleado un emulador predeterminado para simular la ejecución de aplicaciones destinadas a dispositivos móviles, desarrollado igualmente por Sun Microsystems, cuya versión es la 2.5.2 y forma parte como un paquete o módulo que se ejecuta en el Netbeans.

También es necesario instalar previamente la máquina virtual java en el equipo que se va a utilizar para diseñar y desarrollar la aplicación, para poder instalar el Netbeans y el emulador. En nuestro caso, se ha instalado la versión jdk 1.6.0_14.

Por último, mencionar que en el Anexo B se encuentra disponible el manual de usuario de la aplicación.

CAPÍTULO 5

CONCLUSIONES

En este capítulo, revisamos cuáles han sido las principales conclusiones de este proyecto y los posibles trabajos futuros que pueden desarrollarse en base al mismo.

Comenzamos viendo cómo en estos últimos años las Tecnologías de la Información y Comunicación (TIC) han experimentado un importante auge y desarrollo que ha generado grandes expectativas en el sistema educativo, el cual ha ido introduciendo y aplicando estas nuevas tecnologías en el proceso de aprendizaje, creando nuevos espacios educativos y desarrollando nuevas metodologías de enseñanza y aprendizaje. Entre estas nuevas metodologías educativas, hemos presentado el e-learning, que lleva a acabo la formación y aprendizaje del alumnado a través del uso de la tecnología de redes, Internet y las TIC en general y el m-learning, que integra el uso de dispositivos móviles que dispongan de conectividad inalámbrica, con el fin de producir experiencias educativas en cualquier lugar y momento. Ambas metodologías llevan a cabo procesos de aprendizaje que favorecen el autoentrenamiento y la autoevalución del alumno.

Así pues, se ha desarrollado un prototipo software que implementa un juego educativo basado en preguntas tipo test para dispositivos móviles, utilizando el lenguaje de programación y la tecnología J2ME (Java 2 Micro Edition), bajo la plataforma de desarrollo Neatbeans como soporte, guiándonos por los aspectos y fundamentos pedagógicos que sustentan las metodologías de enseñanza-aprendizaje e-learning y m-learning, en especial a esta última, ya que va dirigida precisamente a este tipo de dispositivos móviles y además conseguimos cumplir con los requisitos funcionales mínimos que se especificaban en los objetivos, entre los que hay que destacar como principal que el prototipo software permite al usuario autoentrenarse y autoevaluarse en varias temáticas registradas, a través de cuestionarios de preguntas tipo test, donde al término del mismo, el usuario puede consultar el resultado del test realizado, pregunta por pregunta.

Hay que señalar que este prototipo contempla dos temáticas, Informática e Inglés, para las cuales se han introducido 137 preguntas de Informática y 227 de Inglés, por lo que la base de datos de preguntas presenta un total de 364 preguntas. Cada pregunta plantea 4 posibles respuestas, siendo sólo una de ellas la correcta.

También hay que destacar que se ha diseñado una interfaz sencilla e intuitiva, siguiendo los principios de usabilidad, ya que se ha utilizado elementos de la interfaz de usuario que ofrece cada dispositivo móvil, consiguiendo que la interacción con nuestro prototipo sea familiar y fácil de usar para el usuario desde el primer contacto, ya que es la que el dispositivo muestra por defecto.

Se ha cumplido también como objetivo que nuestro prototipo software sea compatible con una gran variedad de dispositivos móviles actuales, usando la especificación MIDP 2.0 y que vaya dirigido a un amplio sector de la población, por su temática y facilidad de uso.

Se puede proponer, como trabajo futuro, mejorar las prestaciones de este prototipo, ampliando en número las asignaturas a elegir, incluso estructurando las asignaturas por especialidades, donde cada especialidad se compondría de varías asignaturas, todo ello rediseñando el modelo de datos de la base de datos y disponiendo de los datos necesarios para llevarlo a cabo. También se podrían considerar niveles de dificultad en las preguntas, ya en función de las posibilidades.

Otro aspecto que se considera deseable, es que se pudiera actualizar la base de datos, dentro de las capacidades que ofrecen estos dispositivos en cuestión de gestión y almacenamiento de datos en memoria permanente. Actualmente y con la especificación MIDP 2.0, sólo sería posible actualizar la base de datos desarrollando otra aplicación que tuviera los permisos pertinentes para acceder y modificar la base de datos creada por el prototipo y cuyos datos bien se pasaran por medio de la lectura de un fichero de texto en la memoria del dispositivo o bien que dicha aplicación nos solicitara la introducción de datos, a base de rellenar un formulario diseñado especialmente para ello. Es factible que en futuras versiones de la especificación MIDP, se faciliten las operaciones relacionadas con el manejo y gestión de datos en memoria permanente de dispositivos móviles, pudiéndose mejorar las prestaciones y opciones en las aplicaciones destinadas para estos dispositivos.

ANEXO A

MANUAL DE INSTALACIÓN

Con el manual de instalación se pretende resolver posibles dudas que pueden surgir al usuario a la hora de descargar e instalar la aplicación desde el PC a su dispositivo móvil, además de informarle sobre la configuración que debe presentar el dispositivo móvil para que se realice correctamente la instalación.

Aspectos previos a la instalación.

Hemos de decir que el manual de instalación se ha basado en la plataforma Microsoft Windows como sistema operativo sobre el que se realiza las operaciones en el PC, para llevar a cabo la descarga e instalación de la aplicación en un dispositivo móvil.

Previamente se ha comentado que la presente aplicación se ha desarrollado con el lenguaje J2ME de Java para el perfil MIDP 2.0, por lo que el usuario ha de asegurarse que su dispositivo soporta dicho perfil. Esta información la puede consultar en el manual de usuario del dispositivo, en las especificaciones del software. Además, deberá tener instalado en su PC el software facilitado por el fabricante del dispositivo móvil para establecer la comunicación entre el PC y el dispositivo, por alguna de las vías de comunicación que presente, es decir, bien por el puerto USB, Bluetooth, infrarrojos, etc.

El software del fabricante del dispositivo, por lo general, ofrece más funciones y opciones que se pueden realizar con dicho dispositivo. Actúa como un gestor de descarga y de aplicaciones sobre el dispositivo, aunque simplemente puntualizaremos que además de gestionar la descarga de las aplicaciones, puede llegar a iniciar la instalación de la misma. Pero finalmente la instalación se completa siempre desde la interfaz del dispositivo móvil.

También hay que destacar que la aplicación está formada por un archivo JAR, que es el que contiene a la aplicación en sí, llamado JuegoPTT.jar y un archivo JAD (Java Archive Descriptor) que contiene diversa información sobre la aplicación, JuegoPTT.jad. En la figura A.1 puede verse una ventana del explorador del sistema operativo Windows, donde puede localizarse dichos archivos en la carpeta *ejecutables* del CD del proyecto.

Figura A.1: Archivos .jad y .jar de la aplicación.

Descarga de los archivos ejecutables al dispositivo

Previamente se ha de conectar el dispositivo al PC, por alguno de los puertos que dispone y cerciorarse que se establece comunicación entre ellos, es decir, que el PC haya reconocido al dispositivo. Una ver hecho esto, el proceso de descarga de los archivos ejecutables de la aplicación al dispositivo puede llevarse a cabo de dos formas distintas:

- Con el gestor de descarga y de aplicaciones del dispositivo, que en función del fabricante, pueden presentarse una serie de pasos, los cuales han de consultarse en el manual de usuario que acompaña a dicho software. Normalmente estos pasos suelen ser:

  1. Localizar el gestor, instalado en el PC, e iniciarlo.
  2. Una vez iniciado, localizar los archivos de la aplicación presentes en la carpeta *ejecutables* del CD del proyecto, en el navegador de carpetas que dispone.
  3. Seleccionar los archivos JuegoPTT.jar y JuegoPTT.jad.
  4. Seleccionar el dispositivo y la memoria sobre la que se desea guardar estos archivos.
  5. Por último, aceptar la orden de descarga. El gestor, en este último paso, también puede iniciar el proceso de instalación, como veremos más adelante, en el apartado Instalación.

- Copiando directamente los archivos en el dispositivo, para ello hay que:

    1. Localizar los archivos de la aplicación en el explorador de carpetas, situados en la carpeta *ejecutables* del CD del proyecto.
    2. Seleccionar los archivos JuegoPTT.jar y JuegoPTT.jad.
    3. Copiarlos en alguna de las memorias de las que disponga el dispositivo.

Instalación.

La instalación de la aplicación puede realizarse también de dos formas distintas:

- Con el gestor de descarga y de aplicaciones del dispositivo. Una vez que se hayan seleccionado los archivos ejecutables de la aplicación para su descarga en el dispositivo y se haya producido su descarga, como se ha visto en el apartado anterior, el gestor puede preguntar al usuario si desea realizar la instalación de la aplicación. Si el usuario confirma dicha acción, el gestor inicia la instalación. Como último paso, la instalación se completará desde la interfaz del dispositivo, siguiendo sus indicaciones.

- Manualmente, que consiste en:

    1. Localizar, una vez descargados, los archivos ejecutables de la aplicación en la memoria del dispositivo a través de la interfaz del mismo.
    2. Seleccionar el archivo JuegoPTT.jar y ejecutarlo.
    3. El dispositivo nos preguntará si deseamos realizar la instalación de la aplicación.
    4. Si el usuario acepta, comenzará la instalación y si el dispositivo dispone de varias memorias, preguntará al usuario en cual de ellas quiere que se instale.

Una vez finalizada la instalación de la aplicación en el dispositivo, el usuario tendrá que ser capaz de localizarla en el mismo, el cual seguramente dispondrá de un menú donde muestre las aplicaciones instaladas. En el caso de que la aplicación estuviera ya instalada, a la hora de iniciar la instalación, el

dispositivo debe informar de este hecho al usuario, pudiendo cancelar el proceso.

En el siguiente apartado “Ejemplo de descarga de la aplicación en un móvil N73” mostraremos un ejemplo de cómo descargar e instalar nuestra aplicación en un teléfono N73, con el software facilitado por el fabricante Nokia, para conseguir una mayor comprensión sobre este tema.

Ejemplo de descarga de la aplicación en un móvil N73.

Para aclarar un poco el proceso de llevar nuestra aplicación a un dispositivo móvil, vamos a llevar a cabo un ejemplo con un móvil de la gama Nokia, concretamente el Nokia N73. Disponemos del software que proporciona el fabricante para comunicar el PC y el móvil y que sirve de gestor de descarga y aplicaciones, instalado en el PC. Este software se llama Nokia Application Installer y lo que tenemos que hacer primeramente es instalarlo en nuestro PC y una vez completada dicha instalación, lo localizamos y lo ejecutamos. Una vez iniciado, nos aparecerá la interfaz que se muestra en la figura A.2:

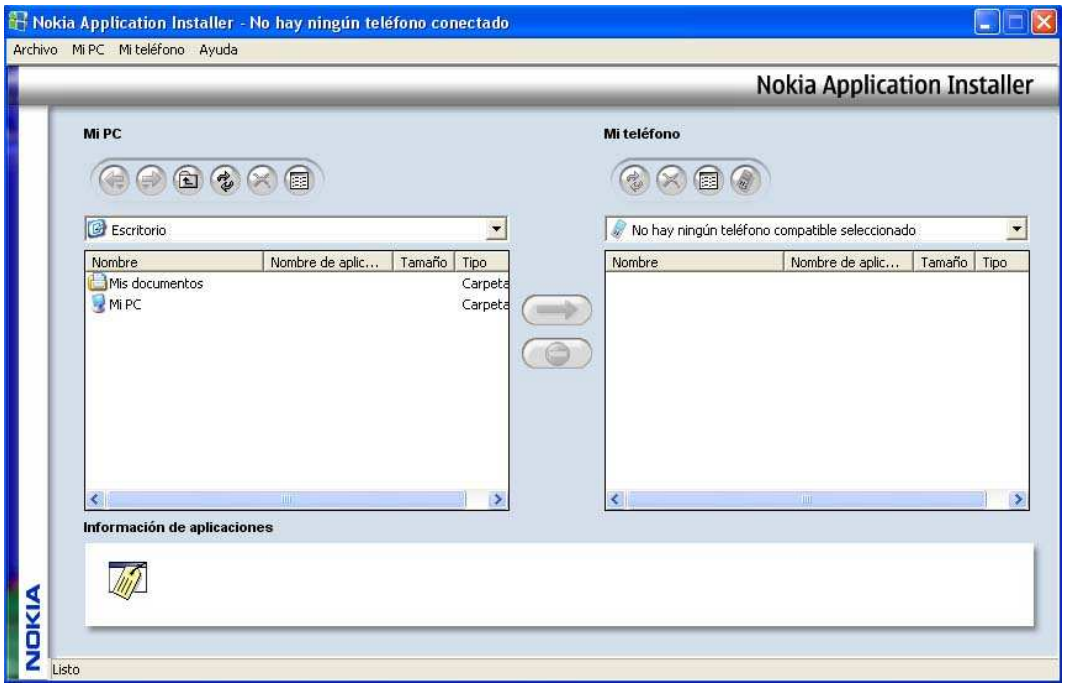


Figura A.2: Gestor Nokia Application Installer

Nos informa de que no tenemos ningún teléfono conectado, así pues, lo que tenemos que hacer es conectarlo al PC a través del cable USB que proporciona el fabricante para el teléfono, al puerto USB del PC. Una vez que detecta que se ha conectado, nos avisa de que ahora sí hay un teléfono conectado y cual es concretamente, N73, como se puede apreciar en la figura A.3. Fijándonos en la figura A.3, en el navegador de la izquierda localizamos los archivos correspondientes a nuestra aplicación, que en este caso sólo visualiza al archivo JuegoPTT.jar, que es el que reconoce que tiene la aplicación. Con el navegador de la derecha especificamos en qué memoria del teléfono móvil deseamos instalar la aplicación.

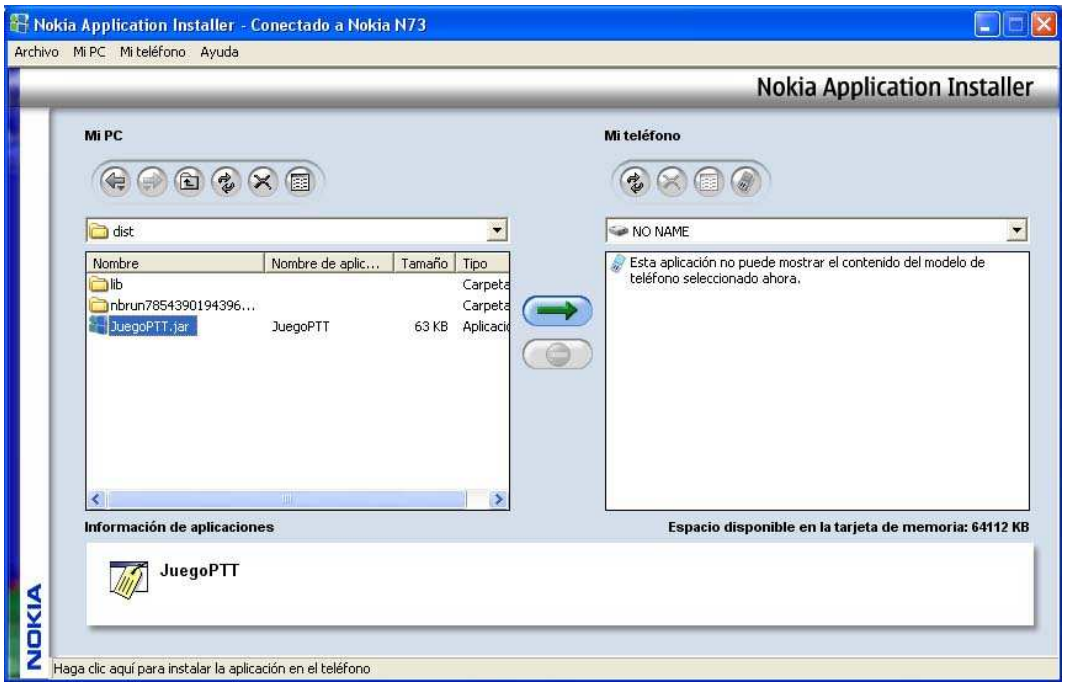


Figura A.3: Nokia Application Instaler conectado al teléfono N73

Seleccionamos JuegoPTT.jar y pulsamos sobre la flecha verde para iniciar la instalación y al poco nos informa con una ventana de texto que finalicemos la instalación de la aplicación sobre la interfaz del teléfono, como puede verse en la figura A.4. Seguidamente, en el teléfono nos aparecerá un aviso de si se desea instalar la aplicación en el dispositivo y se completará la instalación si pulsamos aceptar.


Nokia Application Installer - Conectado a Nokia N73
Archivo Mi PC Mi teléfono Ayuda
Nokia Application Installer
Mi PC
Mi teléfono
dist
Memoria del teléfono
Nombre
lib
nbrun7854390194396...
JuegoPTT.jar
Nokia Application Installer
Finalice la instalación de la interfaz del usuario del teléfono.
Aceptar
Información de aplicaciones
Espacio disponible en teléfono: 22702 KB
NOKIA
Listo

Figura A.4

ANEXO B

MANUAL DE USUARIO

El objetivo de este manual de usuario es que el usuario se familiarice con el uso de la aplicación desarrollada en este proyecto y resolver posibles dudas que se le pueden plantear en su manejo.

Pasaremos a explicar cómo es el funcionamiento general de la aplicación y las opciones que presenta, a la hora de interactuar con ella.

El primer paso que debe hacer el usuario es localizar la aplicación en su dispositivo y ejecutarla. En ese momento, cuando se inicia la aplicación, se muestra brevemente una pantalla de inicio, en la que se da la bienvenida al usuario y una vez cargada, se muestra la pantalla principal de la aplicación, donde se puede ver el logotipo de la misma y las opciones Salir, que es para terminar la aplicación y Menú, que sería para pasar a ver las distintas opciones que presenta la aplicación. En la figura B.1 puede verse las pantallas comentadas, junto con la pantalla que se mostraría una vez seleccionada la opción Menú de la pantalla principal:

Figura B.1: Pantalla Inicio, Pantalla Principal y Pantalla Menú Principal

Comenzar una partida o test.

Si el usuario desea jugar, es decir, realizar un test, desde el menú principal debe seleccionar la opción Jugar y pulsar OK (pantalla derecha, figura B.1). Seguidamente la aplicación mostrará una pantalla donde pedirá al usuario que introduzca sus iniciales (figura B.2), el cual puede hacerlo a través del uso del teclado del dispositivo. Una vez hecho esto, puede continuar pulsando OK.

Figura B.2: Pantalla iniciales

A continuación, puede verse en la figura B.3, en la pantalla de la izquierda, que la aplicación muestra una pantalla en la que el usuario debe elegir entre los dos tipos de test existentes, test corto y test corto, desplazándose con las teclas correspondientes de subir y bajar. El test corto consta de 10 preguntas y el largo de 20. Pulsando OK, la aplicación pasa a mostrar las distintas asignaturas sobre las que se puede basar el test, Informática e Inglés y entre las que el usuario puede escoger (pantalla derecha, figura B.3). Al igual que con el tipo de test, puede desplazarse entre las distintas asignaturas con las teclas de su dispositivo definidas como subir y bajar. Finalmente elegida la asignatura y pulsado OK, la aplicación presenta un test, con las opciones elegidas.

Figura B.3: De izquierda a derecha, pantalla tipo test y pantalla asignaturas

Contestar a una pregunta

Una vez iniciado un test, el usuario puede elegir la respuesta que considere correcta, de la pregunta en la que esté situado (ver figura B.4, pantalla izquierda, donde puede verse que está en la pregunta 2, de 10 preguntas), desplazándose con las teclas definidas por el dispositivo subir y bajar y por último pulsando OK.

Figura B.4: Pantallas durante el test.

Retroceder entre las preguntas

El usuario, mientras no se encuentre situado en la primera pregunta de un test, puede retroceder a la pregunta anterior, pulsando en la tecla Atrás, como viene señalada en la figura B.5, pantalla izquierda. Entonces la aplicación vuelve a mostrarle la pregunta anteriormente contestada, conservando la respuesta que había elegido previamente.

Figura B.5: Volver a la pregunta anterior.

Consultar resultado de un test

Cuando se ha llegado al final de un test, el usuario puede consultar el resultado del mismo, cuando nos encontremos en la pantalla menú resultados, que presenta una lista con el número de la pregunta en el test y una **V** si ha sido bien contestada y una **F** si no ha sido así, como puede apreciarse en la figura B.6.

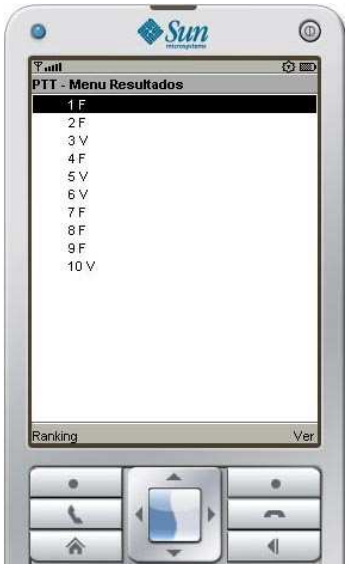


Figura B.6: Menú Resultados, tras un test.

El usuario, en el menú de resultados puede desplazarse con las teclas de subir y bajar y consultar el resultado de cada pregunta pulsando en Ver, como puede apreciarse en la figura B.6.

Consultar ranking de puntuaciones e inicializar ranking.

Los rankings de puntuaciones pueden consultarse seleccionando en el Menú principal la opción Rankings (pantalla izquierda figura B.7), donde la aplicación mostrará otro menú en el que pueden verse los rankings de puntos disponibles en función del número de asignaturas, que en nuestro caso son Informática e Inglés (pantalla derecha figura B.7). También puede consultarse un ranking, cuando finalizamos la consulta del resultado de un test, pulsando en Ranking (figura B.6).

Figura B.7: Menú principal y Menú Rankings.

Además, también se ofrece la opción de inicializar o resetear los rankings si lo desea el usuario, seleccionando la opción Resetear, en el menú de ranking (pantalla derecha, figura B.7) El usuario, estando en el ranking de puntuaciones, puede elegir pasar al menú principal pulsando en Menú, o bien salirse de la aplicación, Salir.

Consultar ayuda

La aplicación presenta una breve ayuda para el entendimiento del juego y explica por encima qué pasos hay que llevar a cabo para iniciar o realizar un test (pantalla derecha, figura B.8). Se accede desde la opción Ayuda del menú principal (pantalla izquierda figura B.8).

Figura B.8: Menú principal y pantalla Ayuda.

En la ayuda no se comenta la asignación de la funcionalidad en el teclado, ya que cada dispositivo dispone de un teclado, con una distribución y características distintas y su interfaz hace la asignación de las funciones a las teclas correspondientes. Lo que sí se puede decir es que al ser funciones y opciones muy comunes en estos dispositivos, realmente no presenta dificultad el uso de la aplicación ni saber en todo momento qué teclas o botones requiere para realizar las acciones pertinentes. Por ejemplo, la acción de aceptar estará asignada a la tecla OK de cualquier dispositivo, la acción volver/Atrás también tendrá asignada una tecla característica del dispositivo que realice esa función. Por lo tanto, no debe presentar dificultades el uso de la aplicación, si el usuario está familiarizado con el teclado de su dispositivo.

Salir de la aplicación

Finalmente, si decidimos salirnos de la aplicación, en alguna de las pantallas de la misma, se muestra un aviso en el que nos pide la confirmación de dicha acción. Esto siempre se hace por seguridad, ya que el usuario ha podido pulsar por error la tecla Salir, o si por el contrario, decide continuar con la ejecución de la aplicación. Esta pantalla puede verse en la figura B.9.

Figura B.9: Pantalla Salir

BIBLIOGRAFÍA

[1] Biblioteca del conocimiento, noticiasdoc.com.

[2] Telefonía Móvil e Historia del teléfono móvil, http://es.wikipedia.org/wiki/Telefon%C3%ADa_m%C3%B3vil.

[3] Lista de países por número de teléfonos móviles, basada en The World Factbook, web de la CIA: https://www.cia.gov/library/publications/the-world-factbook/rankorder/2151rank.html.

[4]CABERO, Julio (2006). «Bases pedagógicas del e-learning». Revista de Universidad y Sociedad del Conocimiento (RUSC). Vol. 3, n.° 1. UOC. <http://www.uoc.edu/rusc/3/1/dt/esp/cabero.pdf>

[5] Salinas, J. (2005). La gestión de los entornos virtuales de formación. Seminario Internacional de la Calidad de la Formación en Red en el Espacio Europeo de Educación Superior, Tarragona.

[6] Fernández Manjón, B. (2006). Especificaciones y estándares en e-learning. revista de Tecnologías de la Información y Comunicación educativas (Nº 6), en: http://reddigital.cnice.mec.es/6/Articulos/articulo_resumen.php?articulo=2

[7] García Aretio, L. (2006). Nuevos Ambientes de Aprendizaje. Boletín electrónico de Noticias de Educación a Distancia (BENED), número de junio.

[8] Cornella, A. (2000). Cómo sobrevivir a la infoxicación. Acto de entrega de títulos de los programas de Formación de Posgrado del año académico 1999-2000 de la UOC. Barcelona, 12 diciembre (Conferencia); en: www.infonomia.com/infonomia/alfons.php.

[9] Guía de innovación metodológica en E-learing , Depósito legal MA-1643-2008, ISBN 978-84-612-6519-0, http://www.portaleva.es/innovacion/

[10] Judy Brown, Mobile Learning ¿el futuro del aprendizaje?,Learning Review, http://www.learningreview.com/component/content/article/1609-mobile-learning-iel-futuro-del-aprendizaje

[11] Walter C. Franchini, M-learning: más allá de una promesa , Learning Review, http://www.learningreview.com.ar/tecnologias-para-la-formacion/articulos-y-entrevistas/1503-mobile-learning-malle-una-promesa.

[12] Fernández González J. y Elortegui Escartín (1996): ¿Qué piensan los profesores acerca de cómo se debe enseñar? Enseñanza de las Ciencias 14 (3).

[13] Rodriguez Andara, Alejandro y Lozano Salas, CARMEN, Evaluación del aprendizaje a través de exámenes tipo test, (2006) Universidad del País Vasco.

[14] P.J. Sánchez, L. Martínez, F. Mata, A. Bernardino, Una aplicación de entrenamiento y auto evaluación para un sistema e-learning, Universidad de Jaén, Departamento de Informática.

[15] Sergio Gálvez Rojas, Lucas Ortega Díaz. "Java a tope: J2ME (Java 2 Micro Edition)". Universidad de Málaga, Edición electrónica.

[16] Agustín Froufe Quintas y Patricia Jorge Cárdenes, "J2ME Java 2 Micro Edition", Ra-Ma, 2004.

[17] James Keogh, "J2ME: The Complete Reference", McGraw-Hill, 2003.

[18] Java ME Technology APIs & Docs http://java.sun.com/javame/reference/apis.jsp

[19]Freeman, E., Sierra, K., Bert Bates, "Head First Design Patterns". Editorial O'Reilly

[20] Brett D. McLaughli, Pollice, G., West, D. "Head First Object-Oriented Analysis and Design". Editorial O'Reilly

[21] Conde, F., ,Interacción Persona-Ordenador: ¿Pero qué narices es una buena interfaz de usuario". Apuntes de la asignatura.

[22] Feito Higueruela, F., Ruiz de Miras, J., Molina Aguilar, A. (1996), ,Análisis y Gestión de Datos", Editorial Universidad de Jaén.

[23] Manchón, E. (2002), ¿Qué es usabilidad?, disponible en Internet (http://www.ainda.info/que_es_usabilidad.htm)

[24] Krug, S. "No me hagas pensar: una aproximación a la Usabilidad en la Web". Editorial Prentice Hall

[25] Moreno, F. y Bailly-Baillière, M. (2002). Diseño instructivo de la formación on-line. Barcelona: Ariel.

[26] Plataformas e-learning. http://solucionesmka.com/blog/?p=6

[27] Lorenzo Garcia Aretio, Aprendizaje Movil, m-learning, Editorial del BENED, diciembre 2004.