UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENERÍA TÉCNICA DE TELECOMUNICACIÓN

ESPECIALIDAD SONIDO E IMAGEN

Implementación de un Sistema de Notificación para Dispositivos Móviles orientado a Usuarios de la Tercera Edad con uso de Multimedia

PROYECTO DE FIN DE CARRERA

Autor: ALEJANDRO MORALES INDIAS

Tutor: FAUSTO J. SAINZ DE SALCES

JUNIO, 2009

ÍNDICE

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

Motivación del trabajo

La motivación principal de este trabajo es llegar a crear un dispositivo que cohesione dos aspectos que, a menudo, parecen ser incompatibles, el uso de las últimas tecnologías y las personas mayores.

En la actualidad, se esta presenciando una "tercera revolución industrial" en el ámbito de las tecnologías de la información, y paralelamente a ésta, se esta produciendo una "revolución demográfica", el incremento de personas mayores de 65 años está aumentando en valores muy superiores al resto de grupos de edad **[1]**. Este incremento de personas mayores se puede ver en la Figura 1 que se muestra abajo.

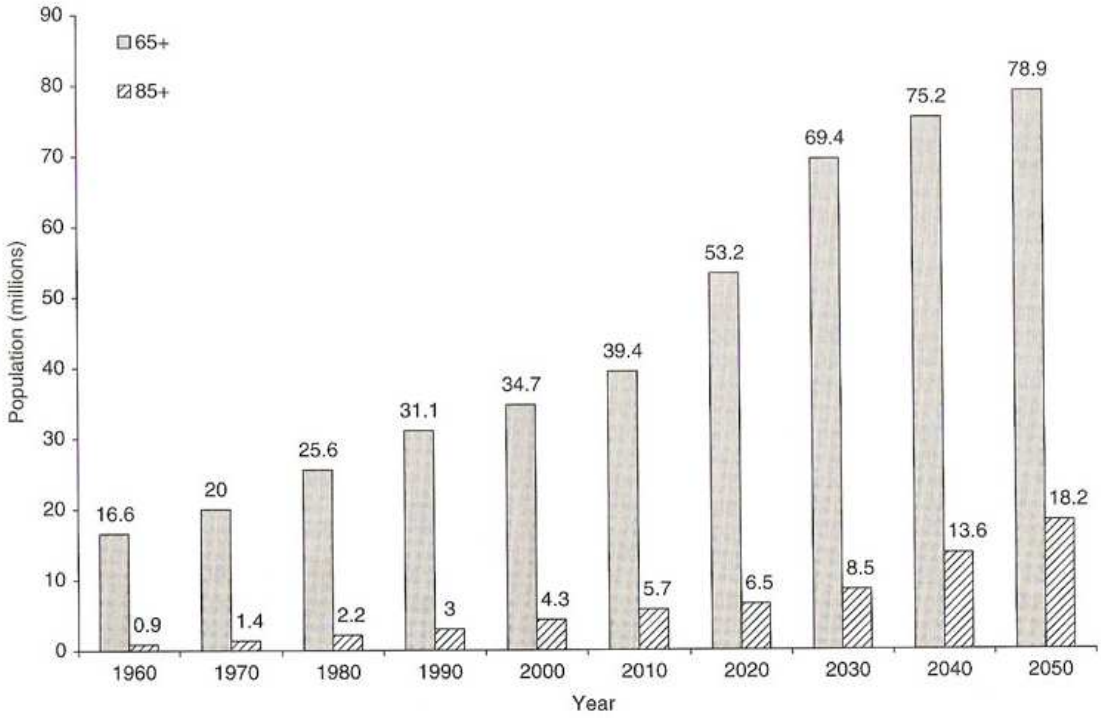


Figura 1.-Población estimada para los grupos de edad mayores de 65 y de 85 años

Acercándonos al ámbito de estudio del proyecto, se plantean cuestiones como: ¿llegan las últimas tecnologías a los mayores?, ¿quieren ellos utilizarlas?, ¿son accesibles a estas personas?, ¿creen en su utilidad?, ¿podemos hacer que les sean útiles?, o incluso yéndonos más lejos ¿podemos hacer que les sean indispensables en la vida diaria?

Intentando responder a estas preguntas aparecen términos como la usabilidad, la accesibilidad o la ergonomía, es decir, estas personas debido a su edad pueden tener deficiencias visuales, auditivas, motrices, o cognitivas y de lenguaje, lo que les suponen unas limitaciones a la hora de acceder a estas tecnologías. Luego, en el punto en el que hay que insistir, es en eliminar estas barreras, de manera que el grupo creciente de personas mayores de 65 años se pueda beneficiar del progreso tecnológico.

Si se consigue vencer estos problemas de acceso a las tecnologías por parte de los mayores, se entrará en un mercado de gran expansión, ya que son un gran número de la población, y con muchas necesidades de apoyo, que se pueden cubrir a través del progreso tecnológico.

Como se esta viendo, la motivación de este proyecto engloba varios aspectos, por un lado crear una herramienta para la notificación en dispositivos móviles, y por otro lado hacer que esta herramienta sea útil para las personas mayores.

Resumen breve del estado del arte

Para comenzar con el proyecto, se tiene que plantear en qué momento nos encontramos dentro de los diferentes temas a tratar, es decir como se mencionó antes, se sabe que estamos en plena revolución de la sociedad de la información, y que esta sociedad cada vez es de edad más avanzada, hay más personas dentro del grupo de la tercera edad, pero más específicamente se quiere asegurar en qué punto se encuentra actualmente la usabilidad, la ergonomía, la interacción persona-ordenador, la accesibilidad, el diseño centrado en el usuario…

La usabilidad se define coloquialmente como facilidad de uso, ya sea de una página Web, una aplicación informática o cualquier otro sistema que interactúe con un usuario. Según Jakob Nielsen **[2]**, que es uno de los gurús más influyentes en este campo, la usabilidad básica no ha cambiado en los últimos 25 años. En 1983, John Gould y Clayton Lewis, presentaron los tres objetivos principales de la usabilidad, que aún hoy aprendemos: dar atención a los usuarios, ejecutando diferentes estudios sobre éstos antes de cualquier diseño; realizar varios estudios empíricos en todo el desarrollo; y hacer un proceso iterativo de diseño. Lo que intenta lograr como objetivo final la usabilidad, es que nos sintamos cómodos al utilizar la tecnología, la veamos como algo práctico, y no que seamos esclavos de ella. Actualmente, a pesar de que estas ideas llevan años en los libros, la mayoría de las empresas sólo están

empezando a utilizar estas técnicas y el crecimiento esperado en los próximos años en esta área es incalculable.

La Interacción Persona-Ordenador (IPO), es la disciplina relacionada con el diseño, evaluación e implementación de sistemas informáticos interactivos para el uso de seres humanos, y con el estudio de los fenómenos más importantes con los que está relacionado **[3]**. El tema principal de esta disciplina está en la interacción y más específicamente en la interacción entre unos o más seres humanos y uno o más sistemas informáticos. Aunque, en los últimos años, la IPO no siempre es un clásico usuario sentado delante de un ordenador de mesa, sino que aparece la interacción con "ordenadores" como PDA's, teléfonos móviles y otros muchos, en los que los usuarios no disponen de la interfaz gráfica clásica y, a pesar de ello, ésta tiene que satisfacer los requisitos de éstos.

Para poder diseñar interfaces satisfactorios, se ha de tener en cuenta muchas disciplinas para comprender toda la problemática que supone el desarrollo de interfaces. En el diseño de cualquier producto, tienen que aparecer las preguntas como: ¿Qué van a sentir los clientes cuando usan ese producto, Aburrimiento, Ansiedad, Serenidad, Sueño?, ¿Qué sentimientos les provocan: Confianza, Miedo, Orgullo, Satisfacción?, ¿Qué sentimientos quisiera el diseñador que ellos tuvieran? Debemos tener un sólido conocimiento de quiénes son nuestros clientes y cuáles son sus necesidades y aspiraciones. Los productos y servicios no deben ser creados sólo como una herramienta para completar una tarea, deben ser vistos como objetos vivos con los que la gente tiene relaciones. Esto es lo que se conoce como Diseño Centrado en el Usuario **[4]**.

En el caso de este proyecto, los usuarios van a ser personas mayores, estas personas no se han considerado importantes en el pasado con el resultado de que los diseñadores no han tenido ninguna oportunidad de diseñar para ellos. Sin embargo, las cosas están cambiando, ya que el mercado que supone este grupo de edad está aumentando. Hay que pensar en los requisitos que estas personas van a valorar, más aun de los que realmente van a necesitar, porque cuando nos acercamos al diseño centrado en el usuario, lo que queremos no es aplicar los cuatro pasos básicos de diseño para "especiales" que aparecen en las múltiples guías, sino intentar acercarnos realmente a ellos, conocer su punto de vista acerca de este software y qué es lo que haría que resultara verdaderamente atractivo y fácil de manejar para ellos.

Descripción del problema

El problema al que hay que enfrentarse es la implementación de un sistema de notificación con uso de multimedia para personas de la tercera edad.

La primera dificultad que se nos plantea es la extracción de requisitos que se necesita hacer para conseguir un diseño centrado en el usuario. Hay que volcar el mayor de nuestros esfuerzos en este punto, porque el sistema a diseñar va a ser principalmente utilizado por personas mayores, y si éstas lo ven complicado de manejar, poco accesible o innecesario, estaríamos ante el fracaso de este proyecto **[5]**.

Para la extracción de estos requisitos se cuenta con varios caminos de investigación, la lectura de literatura relacionada con este tema, y las entrevistas a personas mayores en residencias o centros sociales.

Otro problema que aparece es la integración de este sistema en un dispositivo móvil. En la actualidad, existe una gran variedad de dispositivos móviles en el mercado y las fabricaciones distribuyen constantemente nuevos modelos. Como resultado de esta gran variedad de dispositivos, aparecen diversos problemas, diferente soporte, diferentes capacidades,…Tendremos que hacer un sistema que se pueda integrar en la mayoría de estos dispositivos, adaptándose a la capacidad de cada uno y no incluyendo al usuario nuevos problemas en cuanto a que tipo de dispositivo debe tener **[6]**.

La idea que se nos plantea es llegar al punto de intersección entre los parámetros Producto, Persona que va a utilizarlo y Función para la que se quiere utilizar.

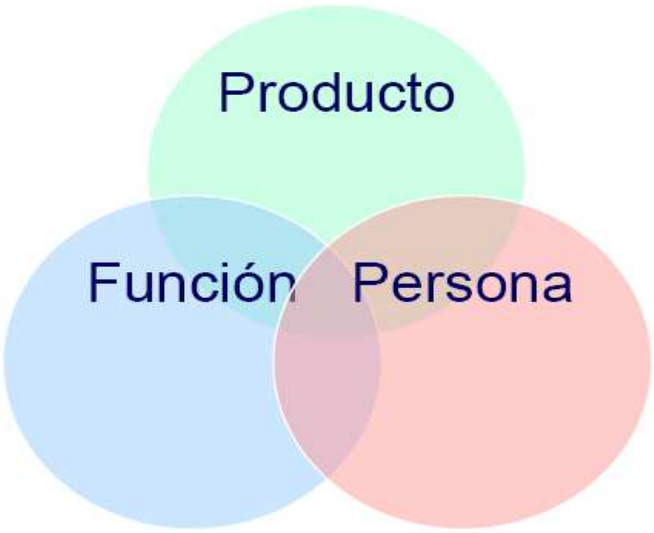


Figura 2.-Metáfora de la unión producto-función-persona

2. ESTADO DEL ARTE

El mundo se encuentra en plena revolución de la sociedad de la información. A diario, se puede experimentar como la tecnología avanza a pasos de gigante, desarrollando nuevos productos que desbordan todas las expectativas.

Pero a menudo, se asocian estas nuevas tecnologías a las personas jóvenes, que han nacido prácticamente a la vez que éstas. Sin embargo, no hay que olvidar que las personas mayores con una correcta formación, y con una serie de pequeñas adaptaciones, pueden sacar grandes ventajas de la tecnología, y hacer que ésta les mejore la calidad de vida.

El concepto de ordenador se ha redefinido de manera sustancial debido a varios factores; la continúa evolución de las tecnologías de la comunicación, que ha hecho que se implanten estándares móviles e inalámbricos; la mejora de la capacidad de proceso, que permite nuevos canales de entrada como el habla, el tacto, etc.; y el aumento de potencia y capacidad de almacenamiento, que ofrece la posibilidad de crear interfaces multimedia.

Dispositivos Móviles

Los avances en informática y telecomunicaciones han provocado el nacimiento de un nuevo mercado para el desarrollo de diversas aplicaciones: los dispositivos móviles. La potencia computacional de éstos es ya comparable con la de algunos ordenadores de mesa, y su difusión y aceptación entre la población es impresionante; por tanto se puede considerar a los dispositivos móviles como la escenario idóneo para la implementación de nuevas soluciones innovadoras.

Los dispositivos móviles son aquellos suficientemente pequeños para ser transportados y que pueden ser utilizados durante su transporte **[7]**.

Tipos de dispositivos móviles:

En la actualidad, existe una gran variedad de dispositivos móviles en el mercado, y las principales empresas del sector distribuyen constantemente nuevos modelos.

Ahora vamos a hacer un pequeño análisis de los distintos tipos:

- Teléfonos móviles con funcionalidades Web

Los teléfonos móviles son los dispositivos más extendidos y utilizados del mercado. Aunque nacieron para el uso de la comunicación por voz, la introducción en ellos de mensajería de texto instantánea, Internet, etc., hace que cada día se utilicen para la comunicación de datos.

El problema de estos dispositivos, es que aunque su utilización es muy elevada, la mayor parte de los usuarios aún los utiliza simplemente para la comunicación por voz o texto, y no ve en ellos la posibilidad de navegar, realizar pequeñas consultas, jugar en red, etc.

Figura 3.- Teléfono móvil con funcionalidades Web

- Dispositivos de mano y PDA's

Estos dispositivos, como mayor ventaja, poseen una pantalla gráfica que supera ampliamente a la de los dispositivos de telefonía.

Los más comunes en el mercado son los Pocket PC y las Palm. Los Pocket PC tiene características de funcionamiento ligeramente mejores a las Palm

Actualmente, se está tendiendo a lanzar dispositivos combinados de telefonía móvil y PDA, con lo que el mercado de ambos dispositivos se está fusionando.

Figura 4.- PDA

- Teléfonos inteligentes: Smart Phones

Son la combinación de la que hablaba antes, la telefonía móvil y las PDA's. Los principales fabricantes son los mismos que los de la telefonía móvil, Nokia, Motorola, etc.

Figura 5.-Smart Phone

- Tablet PC's

Son dispositivos móviles más pequeños que los ordenadores portátiles, y buscan ofrecer las mismas funcionalidades. Suelen tener un teclado, aunque siempre se pueden introducir los datos a través de su pantalla táctil.

Figura 6.-Tablet PC

Sistemas operativos

Desde la aparición de estos dispositivos ha surgido una gran diversidad de sistemas para ellos. Dependiendo del sistema operativo, el dispositivo tendrá un número mayor de funcionalidades, y el desarrollador tendrá una mayor gama de posibilidades a la hora de programar.

En el mercado existen cuatro grandes familias de sistemas, aunque hay dispositivos que vienen con un sistema diseñado por el propio fabricante:

- Microsoft

Sus inicios fueron a través del sistema Windows CE, y con el tiempo se ha ido diversificando para crear una versión adaptada para cada dispositivo.

El sistema Windows CE, tiene una interfaz similar al resto de SO de la familia Windows, por lo que debido a su gran acogida en ordenadores personales es una interfaz muy conocida por la mayoría de usuarios, lo que le hace tener un gran "hueco" en el mercado. Además Microsoft ha desarrollado las versiones de Internet

Explorer, Microsoft Office, Windows Media Placer, lo que le permite al usuario realizar las mismas funciones que con su ordenador de mesa.

Existen otras plataformas de Microsoft que debemos destacar, como Windows Mobile SmartPhone y Pocket PC Phone Edition.

Pocket PC Phone Edition proporciona capacidades de telefonía a los PC de bolsillo, como gestión de la SIM, mensajería de texto y multimedia, etc.

Windows Mobile SmartPhone, proporciona una plataforma elegante para los teléfonos inteligentes, integra servicios de voz proporcionando una serie de programas conocidos por los usuarios de Windows: IExplorer, ActiveSync, Windows Media Player, etc.

El inconveniente de los sistemas de Microsoft, es el elevado consumo de recursos, lo que produce problemas con la alimentación de las baterías, algo muy valorado en los dispositivos móviles.

Figura 7.-Ejemplo de PDA con WindowsCE

- SO Palm

Uno de los sistemas más populares y con mayor experiencia en el sector. En los inicios fue el líder destacado, ocupando más del 75% del mercado, y a día de hoy sigue siendo de los más utilizados, ya que se integra perfectamente con el hardware concreto de cada dispositivo.

Se diseñó específicamente para la gestión de información móvil, optimiza la facilidad de uso, el tamaño, la duración de la batería y la portabilidad. El consumo de batería es uno de sus puntos fuertes, ya que sus funciones utilizan menos memoria y energía que sus homólogos del mercado.

El problema de este sistema, es que en sus inicios, cuando llegó a todos los dispositivos, tenía varios problemas para los desarrolladores, ya que por ejemplo no soportaba multitarea. Y aunque en sus últimas versiones ya proporciona multitarea, mejoras en la conectividad con redes, etc., estos cambios han llegado tarde al mercado, ya que competidores como Microsoft ya lo implementaban en versiones anteriores.

Figura 8.-Ejemplo de dispositivo con SO Palm

- SO Symbian

Este sistema fue creado a través de una alianza entre los principales fabricantes de dispositivos móviles (Ericsson, Nokia, Motorola).

Es un sistema desarrollado exclusivamente para la telefonía móvil, y su característica principal es que es un sistema abierto, por lo que hay una mayor facilidad de personalización y creación de aplicaciones.

Se está convirtiendo en la principal y más importante opción en los dispositivos móviles, ya que se trata del sistema operativo más flexible y con mayor número de funcionalidades. En sus últimas versiones, se ha diseñado pensando en los requerimientos de los dispositivos de tercera generación, aprovechando las ventajas del hardware actual y el que está por venir (GPRS, UNTS, POP3, IMAP4, SMTP, IrDA, Bluetooth, USB, WiFi, OpenGL, etc.)

Además Symbian, proporciona un kit de desarrollo que facilita el trabajo a los desarrolladores de aplicaciones, con C++ o Java.

- Linux

Otra alternativa más a considerar son las versiones Embedded Linux. Linux está haciendo progresos importantes en esta área, y dado que es un sistema abierto al público, cualquier persona interesada en la personalización de su dispositivo móvil, puede descargar el kernel y sus aplicaciones libremente de Internet y empezar a usarlo y a desarrollarlo.

Figura 9.-Ejemplo dispositivo con Linux

Plataformas de desarrollo

Hay también una gran variedad de modelos para el desarrollo de aplicaciones móviles. El tipo de dispositivos móviles que usen los usuarios finales, determinará el tipo de entorno de desarrollo que se debe utilizar. Aunque muchos de estos entornos están disponibles para la mayoría de dispositivos móviles.

- Microsoft Embedded Visual Tools

Está formado por el conjunto de herramientas que facilitan a los desarrolladores la implementación de aplicaciones software sobre dispositivos móviles basados en el sistema operativo Microsoft Windows CE. Estas herramientas, incluyen los sistemas de desarrollo Microsoft Embedded Visual C++ y Microsoft Embedded Visual Basic, SDKs, herramientas remotas y documentación.

Ofrece soporte para todos los dispositivos móviles actuales basados en Windows CE.

- ASP.NET Mobile Controls

Permite diseñar, ejecutar y depurar páginas Web en el entorno Visual Studio .NET e Internet Explorer

No instala ningún componente en el dispositivo del cliente, la interfaz de usuario puede soportar los lenguajes CHTML, WML, o HTML. Además permite adaptar a un dispositivo concreto, o a una clase de dispositivos móviles, la interpretación y soporte de las páginas Web dentro del mismo modelo de programación, sin necesidad de rehacer la aplicación Web móvil.

- .NET Compact FrameWork

Es la plataforma de desarrollo llevada a cabo por Microsoft. NET. Utiliza código controlado y XML para desarrollar y ejecutar aplicaciones y servicios Web en dispositivos móviles basados en Windows CE.

Los desarrolladores pueden reutilizar gran parte de las capacidades, seguridad, herramientas, modelos de programación y código de la plataforma .NET Framework, lo que aumenta notablemente la eficiencia de .NET Compact Framework en el desarrollo de aplicaciones sobre dispositivos móviles.

- J2ME

El éxito del lenguaje de programación Java y de los diversos estándares que orbitan a su alrededor es, hoy por hoy, un hecho indiscutible **[8]**.

J2ME, forma parte de la plataforma Java 2, y es la parte enfocada a los dispositivos móviles. Es un conjunto de tecnologías y especificaciones que están diseñadas para cada una de las diferentes partes del mercado de los pequeños dispositivos. Por tanto, no hay una solución única que englobe a la gran variedad de dispositivos móviles existentes.

Se divide básicamente en configuraciones, perfiles y paquetes opcionales. Cada combinación de éstos, se optimiza para la memoria, la capacidad de procesamiento y de entrada/salida de una categoría específica de dispositivos.

Algunas de las múltiples ventajas de J2ME, son: compatibilidad de aplicaciones entre cualquier dispositivo J2ME, rápido desarrollo, fácil distribución, simplicidad en la integración wireles, interfaces robustas, aumento de la flexibilidad y acceso completo a las funcionalidades de los dispositivos móviles, etc.

Debido a todo esto, J2ME es la tecnología del futuro para la industria de los dispositivos móviles. Actualmente las compañías telefónicas y los fabricantes de móviles están implantando los protocolos y dispositivos necesarios para soportarla.

Por tanto, razones como la compatibilidad entre distintos dispositivos, el rápido crecimiento de desarrolladores de J2ME y su flexibilidad, hacen que sea la opción idónea para elaborar la aplicación de este proyecto.

Interfaces de usuario

Las interfaces gráficas pueden mejorar la vida a las personas, ya que ayudan de manera sustancial en multitud de tareas, desde pilotar un avión hasta aprender a leer. Sin embargo, al hablar de ellas no se puede evitar decir que también causan múltiples frustraciones, ya que una interfaz compleja puede hacer que un usuario no encuentre solución a su problema, o éste se le haga inabordable.

Dentro de la parcela que se ocupa en este estudio, las personas mayores, es fundamental facilitar el acceso a los sistemas informáticos por parte de éstas. "De igual forma que puede hacerse más segura la conducción para conductores y peatones con señales de tráfico más grandes, luces de tráfico más brillantes y con mejor iluminación nocturna, también el escritorio, la Web y los dispositivos móviles pueden mejorarse para todos los usuarios, proporcionándoles control sobre los tamaños de fuente, contrastes de la visualización y niveles de audio" **[9]**.

Actualmente se están creando nuevas interfaces que se adapten al entorno, a los dispositivos, y principalmente a las personas. Las líneas de trabajos seguidas en estos últimos años son, la multimodalidad, la realidad aumentada, las interfaces adaptativas, los aspectos emocionales y la usabilidad **[10]**.

- Realidad aumentada

Se basa principalmente en añadir información digital a los objetos reales, permite al usuario ver el mundo real con una capa de información adicional superpuesta. La diferencia más significativa con la realidad virtual, es que la interacción se da en el mundo real.

Esto permitiría a un médico mientras explora tu cuerpo poder ir viendo los distintos órganos internos, y hacer algo así como un recorrido interno para poder diagnosticar tumores u otras enfermedades.

Para ser considerado un sistema informático de realidad aumentada debe cumplir tres requisitos **[10]**: combinar objetos reales y virtuales en un entorno real; funcionar de forma interactiva en tiempo real; y registrar los objetos reales y virtuales de forma recíproca.

Un ejemplo en España de realidad aumentada, es el proyecto Vilars RA **[11]** que presenta un escenario en el cual un visitante se encuentra en el yacimiento de Vilars interactuando en el mundo real con un mundo virtual, mediante la ayuda de un ordenador en forma de tableta. En el proyecto se introduce una visión futurista de cómo la tecnología puede evolucionar con el fin de ofrecer al usuario un sistema

interactivo con el que queda aumentada la información que le ofrece el mundo real en el que se encuentra inmerso durante la visita.

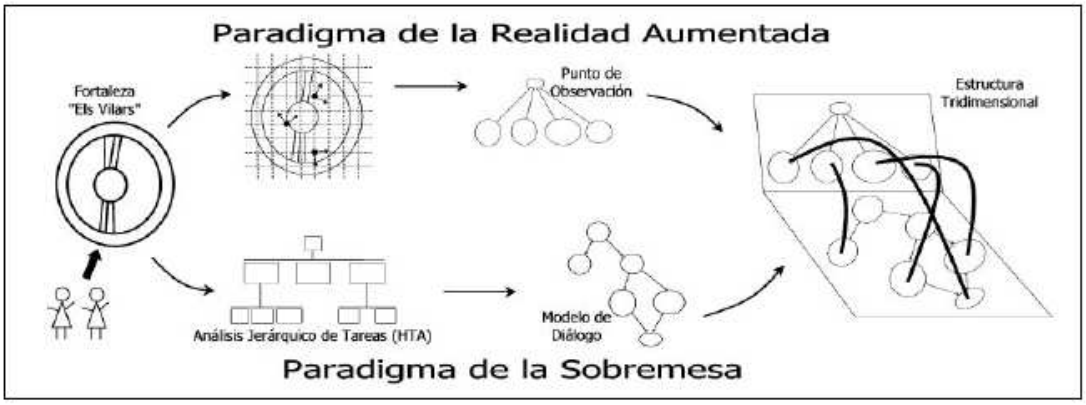


Figura 10.- Paradigma de realidad aumentada

- Usabilidad

La usabilidad se puede definir como "el grado de eficacia, eficiencia y satisfacción con la que usuarios específicos pueden lograr objetivos específicos, en contextos de uso específicos" **[12]**.

En esta definición se puede ver que aparece el adjetivo "satisfacción", el denominar a algo como satisfactorio es muy subjetivo, y depende del usuario a quien se haga la pregunta. Por tanto, la primera idea que se puede obtener de la usabilidad, es que hay que tener en cuenta las peculiaridades de los usuarios potenciales del sistema a diseñar.

Los diez principios de la usabilidad más recomendados por el gurú de la usabilidad Jacob Nielsen **[2]** son:

1. El estado del sistema debe ser siempre visible: El sistema debe mantener a los usuarios informados de lo que está ocurriendo

2. Tiene que haber un emparejamiento entre el sistema y el mundo real: El sistema tiene que hablar el lenguaje de los usuarios

3. Dar al usuario el control y la libertad: Poder retroceder ante acciones erróneas o que haya caminos alternativos.

4. Consistencia y estándares: No utilizar palabras diferentes para denotar un mismo hecho.

5. Prevención de errores: Evitar en la mayor medida posible los errores.

6. Reconocimiento antes que recuerdo: Intentar que todos los objetos, acciones y opciones sean visibles

7. Flexibilidad y eficiencia de uso: Utilizar atajos para usuarios expertos

8. Estética y diseño mínimos: Evitar información irrelevante

9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores

10. Ayuda y documentación: Si es necesario, hay que proporcionar la documentación y ayuda que se requiera.

La necesidad de incorporar el factor humano en el diseño se debe a que, las características de los usuarios pueden afectar al modo de trabajo y condicionar el proceso de comunicación con el sistema.

El analizar al usuario es, conocer aspectos tales como:

→ Habilidades físicas y sensoriales: Estas habilidades determinan la adaptación del entorno de trabajo a las características del usuario, puede ser desde cambiar el tamaño de la letra o el color hasta tener que cambiar el dispositivo.

→Habilidades cognitivas: Las diferencias en cuanto a la capacidad de razonamiento y conocimiento también son muy importantes para el diseño, ya que no es lo mismo una interfaz para una persona experta en la informática que para una persona novel o con discapacidad.

→Personalidad: Dependiendo de si una persona es más tímida o introvertida, también transmitirá este comportamiento al sistema que utilicen.

→Cultura: Por razones evidentes, no es lo mismo diseñar una interfaz para una persona que vive en La India a una persona de occidente.

Otro factor importante en la usabilidad, es el entorno de trabajo en el que se mueven las personas, y aquí aparece el concepto de la ergonomía.

La ergonomía se puede definir como "un enfoque que pone las necesidades y capacidades humanas como el foco del diseño de sistemas tecnológicos. Su propósito es asegurar que los humanos y la tecnología trabajen en completa armonía, manteniendo los equipos y las tareas en acuerdo con las características humanas" **[13]**.

La función de la ergonomía, en cuanto a la interfaz de usuario, es la de lograr una adecuada transmisión de información entre el hombre y la máquina, para lo que se basa en el estudio sensorial del ser humano. Por tanto para que una interfaz sea identificada como correctamente utilizable, debe conseguir que el usuario tenga la sensación de satisfacción del trabajo realizado y que su modo de realización sea fácil e intuitivo.

- La multimodalidad

El concepto de multimodalidad en las interfaces, presenta la idea de emplear varias formas de comunicación entre el usuario y el dispositivo. La interacción se lleva a cabo a través de diversos canales de comunicación simultáneos, estos pueden ser: la voz, el teclado, el tacto, etc.

Lo que se quiere lograr con cualquier tipo de interfaz, es una interacción natural. Y el objetivo de una interacción natural es permitir a los usuarios emplear todos los recursos disponibles para la comunicación, lo que significa utilizar la multimodalidad.

Existen dos causas principales **[14]** para que el desarrollo de las interfaces multimodales sea uno de los principales temas a tratar en todos los congresos de Tecnologías de la Información y las Comunicaciones (TIC):

→ La primera causa, es que las interfaces actuales, de texto y gráficos, son muy interesantes cuando se cuenta con una pantalla de gran tamaño y un teclado que permite un eficiente uso de la información. Pero la aparición de los nuevos dispositivos móviles, hace que normalmente se carezca de esa gran pantalla, y de teclado, o se tenga uno de tamaño muy reducido, por tanto hay que dar cabida a la comunicación por voz o tacto para poder utilizar de manera óptima estos dispositivos.

→ La segunda causa, es que el uso de una sola forma de interacción limita mucho las posibilidades, por ejemplo, si se tuviera un dispositivo que funciona sólo a través de la voz, si se está en un lugar público con un ligero silencio, puede resultar incómodo estar hablando al propio dispositivo. La alternativa en ese caso de utilizar un pequeño teclado o el tacto sería mucho más conveniente. Por tanto, la unimodalidad no parece tener cabida en el progreso de la interacción.

Para describir de manera conceptual, los componentes básicos de una interfaz multimodal, es posible basarse en las pautas que recoge el W3C **[15]**. Así la interfaz, que podemos ver en la Figura 11, contaría con seis partes principales, el usuario, es el que interacciona con el sistema a través de las distintas acciones y recibe respuesta con los diferentes modos de de comunicación; el módulo de entrada, muestra al usuario los diferentes modos de interacción y es en él donde se hace el proceso de reconocimiento e interpretación; el módulo de salida, a través de él se proporciona la información al usuario, ya sea textual, por voz etc.; el gestor de interacción, es el

encargado de gestionar el flujo de ejecución a partir de los componentes de los objetos de interfaz asociados a los diferentes modos de entrada y salida; el componente de sesión, permite soportar la gestión de estados y las sesiones temporales y persistentes en las aplicaciones multimodales; y el componente de sistema y entorno, hace que el gestor de interacción responda a los cambios que se producen en el terminal.

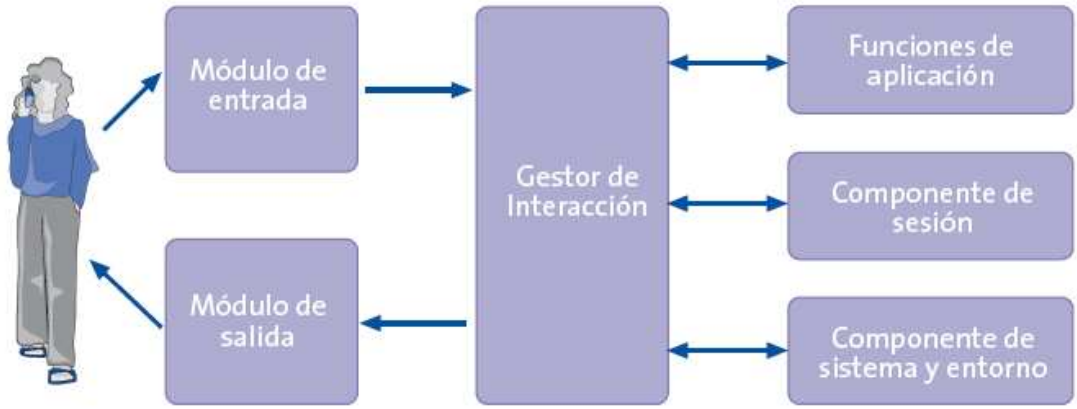


Figura 11.-Componentes de un sistema multimodal

Estas interfaces presentan varios problemas como son, las exigentes necesidades computacionales que demanda algunos modos de interacción, por ejemplo los conversores voz-texto; la dificultad vinculada al control temporal o sincronismo de los diferentes eventos asociados a los distintos modos de interacción posibles; y como no, la complejidad que supone la integración de las diferentes tecnologías de interacción, entre estas tecnologías aparece el reconocedor de voz, el identificador biométrico, etc.

Pero, a pesar de estos problemas, estas interfaces ofrecen un mundo nuevo en la accesibilidad, ya que, un usuario cualquiera podrá utilizar cualquier servicio desde cualquier sitio y casi independientemente del tipo de terminal que en ese momento tenga disponible. Además, para las personas con discapacidad, supone un gran avance, ya que todos los servicios estarán también preparados para ellas.

Las Personas Mayores y el Uso de la Tecnología

Las personas mayores, al igual que los discapacitados suelen presentar mayores problemas a la hora del acceso a las nuevas tecnologías.

Sin embargo, en los últimos años ha habido una evolución del concepto de integración de personas con discapacidad. Esta evolución se basa en las ideas de protección de éstas, en la inclusión y la no discriminación.

El concepto de discapacidad en este proyecto tiene que ser entendido desde el punto de vista de la ergonomía, desde este punto la discapacidad es el desajuste entre las habilidades funcionales de la persona y los requerimientos funcionales de un producto.

El modelo ergonómico de discapacidad, puede representarse gráficamente a través de una distribución normal, como se ve en la Figura 12, de las habilidades de las personas y los requerimientos del producto.

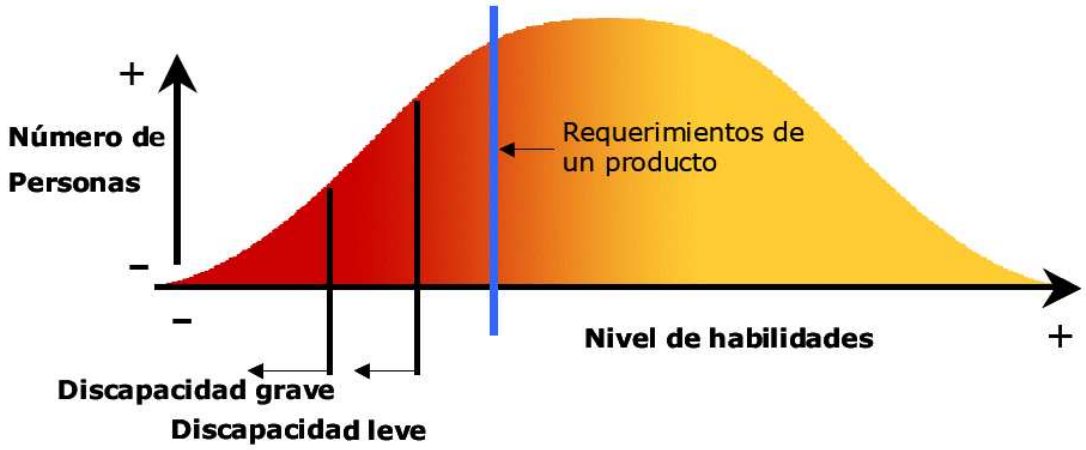


Figura 12.-Modelo ergonómico de la discapacidad

Como se puede observar en la Figura 12, a medida que aumentan los requerimientos del producto, aumenta el nivel de habilidades que tiene que tener el usuario y por tanto los usuarios con menos habilidades, con discapacidades, no pueden usar el producto. Por lo que, cuanto menos requerimientos exija el producto mayor número de personas podrán utilizarlo, será un diseño más inclusivo.

Centrándose en las personas mayores, en las distintas guías de software accesible para ellas, se habla **[16]** de cuatro criterios claves para hacer este software accesible, estos son:

→Evitar los fallos técnicos o de aplicación: Se debe hacer unas exhaustivas pruebas de fallos anteriores a la presentación del software a estas personas, ya que muchas veces los fallos de aplicación producen frustraciones y hace que el aprendizaje se vuelva costoso.

→Facilidad para utilizar el software: Ya se ha hablado anteriormente que la usabilidad es una de las piezas claves del diseño, ya que la facilidad de uso nos invita a acceder a la tecnología, y a evitar los miedos que nos produce el desconocimiento de algo.

→Facilidad para encontrar la información: Una vez inmersos en la aplicación, tiene que ser lógico e intuitivo el encontrar lo que buscamos, es decir, si nuestra aplicación sirve para proporcionar información del tráfico, una vez que entremos en ella no tiene que ser difícil encontrar dicha información.

→Facilidad para comprender la información: La compresión de la información es necesaria para poder utilizar el software, el diseñador suele tener conocimientos técnicos, por lo que suele utilizar un lenguaje pero a la hora de diseñar la aplicación tiene que darse cuenta que el lenguaje debe ser el más claro y básico posible.

La importancia de las personas mayores ha crecido en una considerable medida en los últimos años, en España la aparición del "Plan de Acción para las personas mayores 2003 – 2007", ha supuesto un vuelco definitivo a la concepción que se tenía de ellas dentro de la sociedad. Por poner un ejemplo una de las medidas de este plan es la "puesta en marcha de los trabajos tendentes a desarrollar titulaciones académicas nuevas relacionadas con la atención a las personas mayores, poniendo especial interés en la formación profesional específica / inicial (título de Técnico en Atención sociosanitaria) que den respuesta a nuevas necesidades" **[17]**.

Este tipo de medidas, ha hecho que el desarrollo tecnológico aplicado a estas personas haya aumentado en los últimos años.

La accesibilidad a la Web es uno de los puntos en los que se reúnen más iniciativas para las personas mayores, hay multitud de ejemplos, como http://www.jubilatas.com/ , que es llevada por un grupo de personas mayores de 50 años, o como http://www.redmayores.net/ , que se desarrolla dentro de un programa que se ha puesto en marcha para integrar a las personas mayores en el uso de las nuevas tecnologías.

Uno de los proyectos más destacados en España, para hacer más accesible la tecnología las personas mayores, es el proyecto denominado Software Senior, se ha creado una página Web totalmente accesible, en la que puede entrar cualquier persona y según su discapacidad, el ordenador que disponga, el sistema operativo, e incluso el dinero que disponga, puede descargarse aplicaciones que le ayuden en el uso del PC, esta página es: http://softsenior.cesga.es/index.php . Se pueden encontrar

programas que ayuden a una mejor visión de la pantalla, como "Adobat e-book reader" o "Microsoft reader"; programas que ayuden al uso del teclado, como "TotiPm"; o programas que ayuden al uso del ratón como "Ratón visual".

Como se ve, en la actualidad existe una nueva perspectiva hacia el cambio demográfico, la vejez ya no se ve como una época de involución y ausentismo, sino que está empezando a ser visto como una nueva oportunidad económica. Las instituciones públicas hacen cada vez más hincapié en la necesidad de realizar reformas estructurales e innovaciones tecnológicas que puedan, al mismo tiempo, mitigar los problemas económicos de una población envejecida, contribuir a su mejor calidad de vida y crear nuevas oportunidades económicas y de negocios **[18]**.

Además, desde el punto de vista de las Tecnologías de la información y la comunicación (TIC), éstas están obligadas a dar solución a las múltiples necesidades que las personas mayores necesitan, creando una sociedad más igualitaria, donde todas las personas tengan al alcance de su mano las nuevas posibilidades de la tecnología. Y estas soluciones, a la vez, suponen una forma de potenciar el mercado de las nuevas tecnologías, ya que están ayudando a que los distintos investigadores y desarrolladores tengan que buscar alternativas para dar accesibilidad **[18]**.

Por tanto, resulta paradójico, pero las personas mayores son el grupo de personas que puede tener un mayor beneficio debido a las nuevas tecnologías, y el objetivo de los creadores de estas nuevas tecnologías debe ser hacérselas llamativas y accesibles, para que realmente las vean como algo necesario y prioritario, y no como algo "pesado" y de mucha dificultad.

Se puede analizar los datos estadísticos nacionales **[19],** que aparecen en la tabla 1para ver en qué punto se encuentra en estos momentos el uso de la tecnología por parte de las personas mayores:

Edad (años)	Personas que disponen de Teléfono móvil (%)	Personas que han usado alguna vez el ordenador (%)
16-74	88,8	67,4
55-64	78,7	35,4
65-74	57,9	16,2

Tabla 1.-Los mayores y el uso de la tecnología

Los datos que se han obtenido **[19]**, que se pueden visualizar en la tabla, reflejan un menor porcentaje de utilización de las Tecnologías de la Información y la Comunicación (TIC) a medida que se aumenta la edad, lo que se esperaba desde un principio. Sin embargo, se puede ver que el grupo de edad de 55 a 64 años, aumenta de forma significativa sus porcentajes en relación con el grupo de mayores de 65, por tanto se puede llegar a la conclusión que en próximos años, la demanda de tecnología por parte de las personas de la tercera edad va a incrementarse fuertemente.

En la tabla anterior, también se puede ver la diferencia de la utilización de los teléfonos móviles y los ordenadores. Se observa que los teléfonos móviles tienen una mayor acogida entre todos los grupos de edad, y sobre todo si nos fijamos en las personas mayores de 65 años, apenas un 16,2% ha usado alguna vez un ordenador, y sin embargo más de la mitad dispone de teléfono móvil. Por tanto, el crear un software para personas mayores de utilización en teléfonos móviles, tiene más interés a crearlo para el ordenador.

3. METODOLOGÍA DE DESARROLLO

Introducción

La metodología es un conjunto de técnicas que nos proporcionan la manera de proceder para conseguir unos propósitos deseados en un producto. En este proyecto se ha utilizado la metodología adecuada para desarrollar un sistema interactivo bajo los parámetros de la usabilidad y la accesibilidad, ya que el fin último del proyecto es que sea útil para un determinado grupo de personas que necesitan ciertos requisitos de accesibilidad. Para conseguir estos objetivos se ha seguido el modelo de proceso conocido como "Ingeniería de la Usabilidad y la Accesibilidad" **[29].** Las fases en las que se estructura este modelo, son las que se ven en la figura 13, que aparece a continuación.

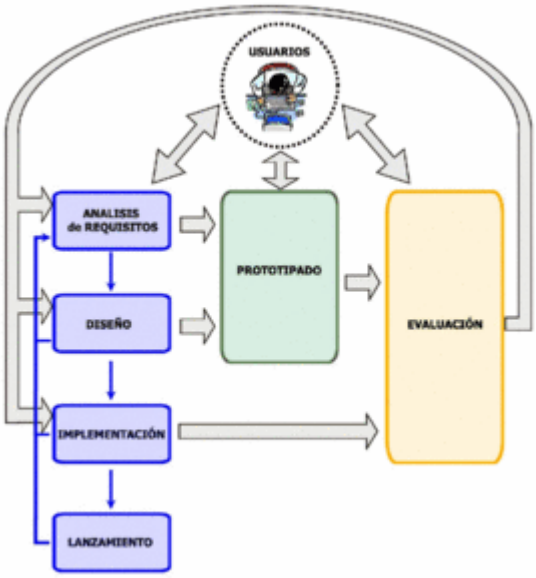


Figura 13.- Modelo de Ingeniería de la Usabilidad y la Accesibilidad

El modelo de Ingeniería de la Usabilidad y la Accesibilidad, proporciona una organización conceptual de los pasos a seguir para el desarrollo de un producto interactivo. Si nos fijamos en la figura 13, se puede ver que el modelo distingue, tres partes fundamentales:

1. La ingeniería del Software (en azul), que comprende todo lo relativo a la creación del sistema.

2. El prototipado (en verde), que engloba las técnicas para la puesta en pruebas del sistema

3. La Evaluación (en amarillo), que contiene las diferentes técnicas o métodos de evaluación del sistema

En la figura 15, se puede apreciar que el centro del modelo y por donde concurren todas las fases son los usuarios, mostrando la evidencia de que es un proceso de diseño centrado en el usuario **[24].**

A continuación, se presenta en detalle cada uno de los módulos que forman el proceso.

Análisis de requisitos

El análisis de requisitos que se conoce en la Ingeniería de Software "clásica", establece los servicios que el sistema tiene que proporcionar y las restricciones bajo las cuales debe operar. Con este análisis se intenta contestar a las preguntas: "¿qué debe hacer el sistema?" y "¿cómo debe hacerlo?", de este modo aparecen dos tipos de requisitos **[24]**:

1. Funcionales: Describen la funcionalidad o servicio del sistema, contestan a la primera pregunta.

2. No funcionales: Son restricciones al propio sistema (por ejemplo el tamaño de letra), o restricciones al proceso de desarrollo (por ejemplo utilizar un determinado lenguaje de programación).

El análisis de los requisitos en el modelo de la Ingeniería de la Usabilidad y la Accesibilidad utiliza lo descrito anteriormente, pero añade un cambio de calidad en los conceptos para conseguir un diseño centrado en el usuario. Este cambio es principalmente la implicación del usuario dentro del análisis, dándole prioridad a sus necesidades **[24].**

Para poder hacer partícipe al usuario en el análisis de los requisitos, hay que interactuar con él dentro del contexto en el que va a utilizar el sistema, y adaptar éste al modelo mental del usuario. Para una óptima captura de requisitos desde este punto de vista se siguen una serie de actividades, que se describen a continuación:

Análisis etnográfico

Este análisis estudia el modo de vida de un grupo de personas, los usuarios. Para poder realizarlo, no basta con ver al grupo desde un punto de vista exterior, sino que hay que adentrarse en él, hay que conocer las ideas arraigadas que poseen, los condicionantes históricos en que viven, en definitiva, hay que ponerse en el papel de uno de ellos **[24]**.

El uso de la investigación etnográfica ayuda a responder a las cuestiones sobre organizaciones y mercados que hasta ahora no tenían respuesta, por ello, en la actualidad, está proporcionando resultados impresionantes a las grandes empresas de software que se han dado cuenta de la cantidad de información útil que les aporta.

Dentro del análisis etnográfico existen múltiples técnicas, que ayudan a lograr ponerse en el papel del usuario, algunas de las más utilizadas son: obtener el perfil del usuario, o el perfil del entorno.

Perfil del usuario

Consiste en obtener una descripción de las características más relevantes de la población potencial que usará la interfaz de usuario que se va a diseñar. Entre estas características se tendrá que obtener, por ejemplo, el nivel de uso de los equipos informáticos, el nivel de estudios que poseen, el entorno social… **[24]**

Estas características tendrán un impacto fundamental en el diseño de la interfaz de usuario, ya que no es lo mismo diseñar una interfaz para un experto en informática que para una persona que no utiliza frecuentemente el ordenador.

Perfil del entorno

El entorno donde se realiza un determinado trabajo, influye fuertemente en cómo se realiza éste. Por tanto, es otro factor a tomar en cuenta en diseño de la interfaz de usuario. Es muy diferente diseñar un sistema para una persona que tiene que utilizarlo mientras está conduciendo, a diseñarlo para otra que lo utiliza en su ordenador de mesa.

Análisis de implicados

Es el análisis de todo los que influyen en el desarrollo del proyecto, desde el programador hasta el usuario final. Para ello se han creado una serie de herramientas que ayudan a identificar a cada uno de los implicados **[24]**:

Actores

Son los individuos o personas implicadas en el proyecto. Es de gran utilidad poder identificar a cada uno de ellos, para posteriormente asignar las tareas convenientes

Roles

Es una subdivisión de clases de actores, que tienen asignadas similares tareas. Los roles pueden variar en el tiempo, y atribuirse varios a una misma persona.

Organización

Es la relación existente entre los actores y roles, en el contexto de las tareas a realizar. La organización describe la estructura del conjunto de actores, así como su papel en los distintos roles.

Objetos

Todo lo que sea relevante en el uso del sistema en una cierta situación es un objeto. Pueden ser cosas físicas, como una mayor o menor inclinación de la pantalla, o conceptuales, como un mensaje, una contraseña…

Análisis contextual de tareas

Consta de la realización de un estudio de las tareas actuales de los usuarios, cómo las realizan, que patrones de trabajo utilizan, y llegar a especificar y entender los objetivos de los usuarios.

Es ente análisis es donde aparecen los requisitos funcionales y no funcionales de la Ingeniería de Software "clásico", sin embargo se amplían los objetivos no funcionales, introduciendo los objetivos de usabilidad y accesibilidad, que aseguren un buen funcionamiento del sistema, parte de estos objetivos se obtienen a través de el análisis de los implicados y el análisis etnográfico **[30]**.

Diseño

El aspecto que se considera más importante en un sistema interactivo reside en el diálogo con el usuario. La interfaz permite que este diálogo sea más o menos fluido, facilitando al usuario el acceso a los recursos del ordenador.

El usuario no se interesa por como funciona el sistema interiormente, sino en cómo tiene que utilizarlo, cómo se le presenta a él la aplicación. Por tanto si se diseña sin tener en cuenta como primer punto la interfaz, se obtendrán diseños que dependerán totalmente del diseño de los datos y las funciones, sin tener en cuenta que el diseño tiene que ser por y para el usuario. Una vez se tiene hecha la especificación, propuesto un diseño y el código está implantado, es realmente difícil cambiar las características de la interacción y presentación de la información. Por tanto, se debe empezar con un idea clara de cómo se quiere la interfaz y como serán las interacciones con el usuario para después, desarrollar las especificaciones funcionales que sirvan de guía al diseño posterior **[31].**

Para realizar el diseño, se debe seguir una serie de actividades o métodos, que llevan a obtener un diseño centrado en el usuario. Los métodos a seguir aparecen en la figura 14.

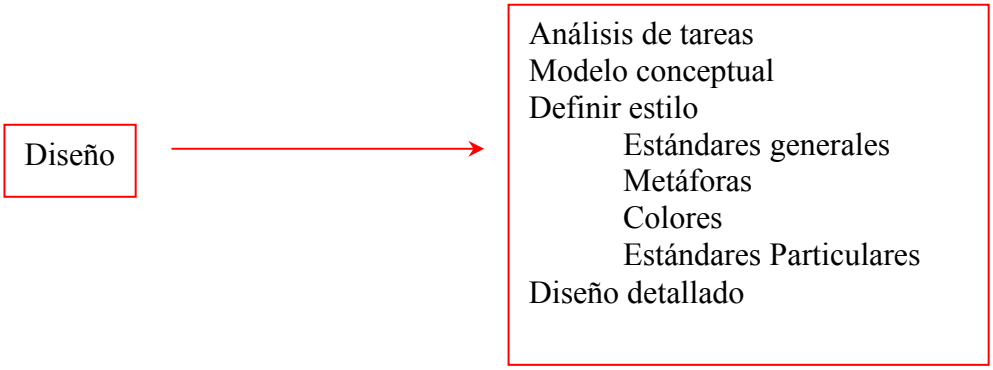


Figura 14.-Métodos a seguir en el diseño

Análisis de tareas

Con ayuda de los requisitos obtenidos en la fase anterior, se rediseñarán las tareas de usuario para racionalizar la organización del trabajo y explotar las capacidades que la automatización proporciona.

El análisis de tareas supone el paso intermedio entre las descripciones de las tareas obtenidas y su codificación, estando orientado a describir las interacciones usuario sistema de manera sistemática y eficiente. En la figura 15 se describe de manera conceptual el proceso de obtención de las tareas **[24]**.

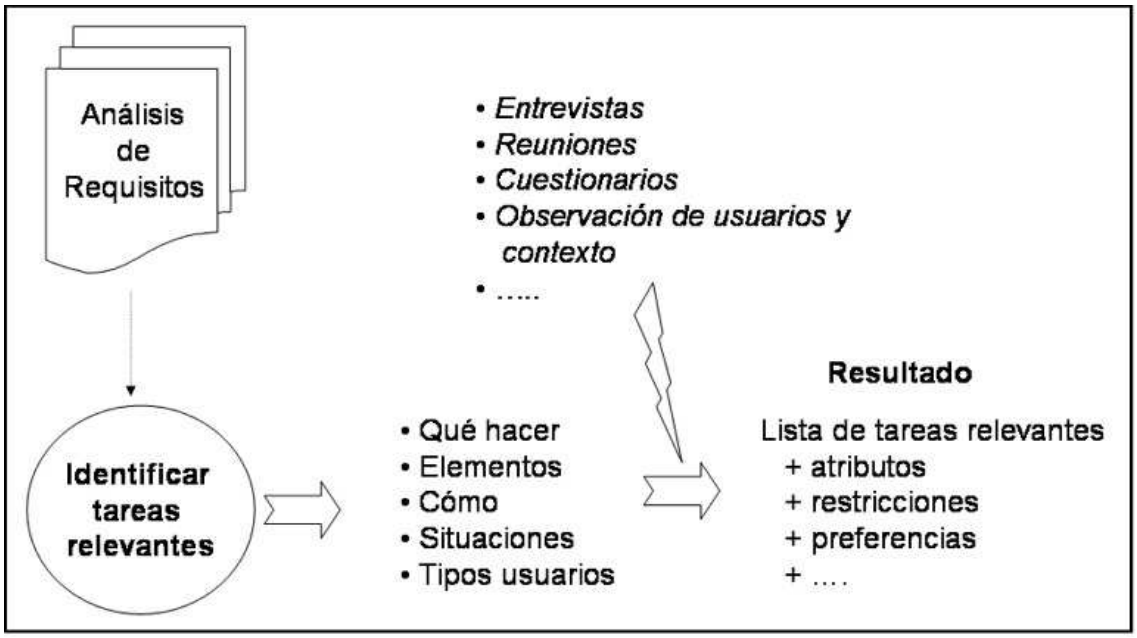


Figura 15.-Proceso de obtención de la lista de tareas a analizar.

Existen multitud de métodos, para realizar el análisis de tareas que se diferencian entre sí básicamente por el grado de formalismo de su notación y por su poder de expresividad y finalidad. Todos parten de un objetivo, estado que el usuario quiere alcanzar dentro de una aplicación, que para conseguirlo se seguirá una serie de tareas, y éstas a su vez estarán estructuradas en unas determinadas acciones.

Modelo conceptual

El modelo conceptual se basa en dar forma a las primeras ideas de diseño mediante diagramas y anotaciones, desde un punto de vista exterior, sin entra en detalles del sistema. Es lo que el usuario debe conocer del funcionamiento del sistema **[24]**.

En esta fase, se definen las primeras pantallas por las que navegará el usuario, y los caminos de navegación que seguirá. Puede entenderse, como una descripción del sistema propuesto en términos de un conjunto integrado de ideas y conceptos sobre lo que éste debe hacer, y cómo lo debe hacer, perfectamente comprensible para los usuarios.

Finalmente, se puede decir que el modelo conceptual es, una visión global del producto final, basándose en las necesidades y limitaciones del usuario obtenidas en el análisis de requisitos anteriormente hecho.

Definir estilo

Se tiene que "definir un estilo que garantice la coherencia general de toda aplicación, entendiendo que la coherencia engloba todas las facetas de la interfaz (visual, sonora...), la funcionalidad completa del sistema y la interactividad con la que éste ofrece dicha funcionalidad al usuario mediante la interfaz" **[24]**.

- Estándares generales

Existen varios estándares generales que son principios y guías que se deben seguir. Algunos de estos estándares son Macintosh Toolbook, MS Windows, o en aplicaciones para la Web W3C, en telecomunicaciones IEEE, etc.

Estos estándares generales, se diseñan con el objetivo de mantener una línea común en los productos desarrollados, y de esta manera hacer más fácil al usuario el manejo de distintos sistemas

- Metáforas

Una metáfora, en un contexto de diseño, es la asociación de conceptos abstractos con conceptos que resultan más familiares y accesibles al usuario. Hay multitud de ejemplos que han tenido gran éxito, como el "escritorio" del PC, o las señales de tráfico utilizadas en multitud de programas informáticos.

Con esto puede verse, que si se definen correctamente las metáforas, pueden aportar un mayor entendimiento de la funcionalidad al usuario, con lo que necesitará menor tiempo de aprendizaje y recordará su uso con facilidad.

Sin embargo, los errores a la hora de crear metáforas nuevas son muy comunes entre los diseñadores, ya que algo que puede significar para un diseñador por ejemplo borrar la pantalla, para el usuario puede significar borrar todo el documento. Por tanto a la hora de crear estas metáforas, hay que tener en cuenta los usuarios potenciales, y solo crear aquellas que son más obvias o más comunes a éstos.

- Color

Utilizar los colores apropiados para el público al que va dirigida la aplicación es un aspecto muy importante. Dependiendo de los contrastes entre ellos y de su combinación, se podrá obtener una interfaz más o menos agradable para el usuario. Es muy importante también donde se va a mostrar esa interfaz, ya que no es lo mismo una pantalla de ordenador o televisión a una pantalla de un teléfono móvil.

También los colores pueden ayudar a crear buenas metáforas, como los colores rojo y verde representando lo no permitido y lo permitido, o las letras en rojo para advertir de algo.

- Estándares particulares

Estos estándares se crean para un grupo de usuarios en concreto, puede ser para una gran empresa, en el que aparezca el logotipo, los colores corporativos, etc. O puede ser para un tipo de personas que tienen alguna limitación, discapacidad visual, discapacidad motriz, que requieren que la interfaz se adapte a estas limitaciones.

Las metáforas y colores de estos estándares suelen venir predeterminados por el usuario final, ya no se basan en un trabajo de campo hecho por el diseñador, sino que suele ser el propio usuario el que le da las pautas.

Diseño detallado

Es el procedimiento que se basa en recoger todos los pequeños detalles que apareciesen en las tareas anteriores, hasta que se disponga de la versión definitiva del software. Esta fase puede integrarse en el resto, si se tiene el riguroso cuidado de cumplir todas las tareas **[24]**.

Prototipado

Los prototipos son documentos, diseños o sistemas que simulan o tienen implementadas partes del sistema final que se desarrollará posteriormente. El concepto de prototipado engloba todas las herramientas que permiten realizar estas simulaciones a los diseñadores de sistemas **[24]**.

En el modelo de Usabilidad y Accesibilidad no se marca el punto exacto de hacer el prototipo, puede ser tras el análisis de los requisitos o tras hacer el diseño, se deja a elección del diseñador, porque dependiendo de lo que sea conveniente en el producto se podrá hacer antes o después

Los prototipos ayudan al diseñador a responder muchas cuestiones que han quedado pendientes sobre los requisitos o el diseño. Además clarifican los aspectos técnicos y ayudan a decidirse por una opción determinada.

Algunos tipos de prototipos que se suelen utilizar son:

- Escenarios

Se crea un escenario que consiste en recrear un entorno similar al que se encontrarán los usuarios a la hora de utilizar el software. En los escenarios se busca objetivos sugeridos por la apariencia y comportamiento del sistema, se responde a preguntas como "qué es lo que las personas quieren hacer con el software", "qué procedimientos usan y cuáles no", "cuales se realizan o no satisfactoriamente", "qué interpretaciones hacen de lo que sucede".

- Herramientas de diagramación
    - Narrativa: Una historia completa de la interacción hecha con la interfaz existente, o con un diseño nuevo
    - Flowchart: El diagrama de flujo es una representación gráfica de las series de acciones y decisiones extraídas de la narrativa
    - Texto procedimental: Una descripción paso a paso de las acciones del usuario y las respuestas del sistema
- Prototipos Software

Los prototipos software son primeras versiones de ciertas funcionalidades del sistema, las cuales son realizadas con el lenguaje de programación escogido para desarrollar la aplicación, o con otro visual más orientado al prototipo.

Evaluación

El objetivo de los prototipos es poder ser evaluados, para poder comprobar el funcionamiento del sistema. Es una herramienta muy útil para hacer participar al usuario, y evaluar el producto antes del la implementación final y el lanzamiento.

Hay tres formas de evaluar un producto, la inspección, la indagación y el test **[32]**. A continuación se explican las tres fases:

Inspección

La inspección sirve para evaluar la usabilidad del sistema. Hay una serie de métodos para evaluarla, desarrollados por conocidos expertos, que explican el grado de usabilidad de un sistema basándose en la inspección o examen de la interfaz del mismo.

Los métodos más importantes son:

- Heurística

Método desarrollado por Nielsen y Molich que consiste en analizar la conformidad de la interfaz con unos principios de usabilidad mediante la inspección de varios evaluadores expertos.

- Recorrido de la usabilidad plural

Método desarrollado por Bias en los laboratorios IBM. Las principales características de este método son que se realiza con tres tipos de participantes que evalúan el modelo a partir de prototipos de papel, y con una especie de debate final entre los participantes.

- Recorrido cognitivo

Este método se centra en evaluar la facilidad de aprendizaje del sistema. Se realiza básicamente de la forma que la mayoría de los usuarios prefieren o suelen aprender software: por exploración.

- Estándares

Para evaluar este método se precisa de un evaluador que sea un experto en el o los estándares a evaluar. Dicho evaluador va pasando por la interfaz comprobando el cumplimiento o incumplimiento de dichos estándares.

Indagación

Este proceso trata de llegar al conocimiento de algo discurriendo, o por conjeturas y señales. En los métodos de evaluación realizados por indagación hay un gran trabajo de hablar con los usuarios y observarlos detenidamente usando el sistema de manera real, u obteniendo respuestas a preguntas orales o escritas **[32]**.

Los métodos principales que se utilizan por indagación son:

- Observación de campo

Este procedimiento lo describe Nielsen sobre la base de trabajo que se realiza al visitar el lugar o lugares de trabajo donde se estén realizando las actividades que son objeto de estudio, y donde se encuentran los usuarios representativos. El principal objetivo, consiste en observarlos para entender cómo realizan sus tareas y qué clase de modelo mental tienen sobre ellas. Esta información será completada con preguntas o entrevistas personales. Este método se puede utilizar en las etapas de prueba y del despliegue del desarrollo del producto **[32]**.

- Focus Group

El Focus Group es una técnica de recolección de datos donde se reúne de 6 a 9 usuarios para discutir aspectos relacionados con el sistema. Un ingeniero de factores humanos hace de moderador, y tiene que preparar la lista de aspectos a discutir y recoger la información que necesita de la discusión. Esto permite capturar reacciones espontáneas e ideas de los usuarios que evolucionan en el proceso dinámico del grupo.

- Entrevistas

Las entrevistas aportan información muy valiosa sobre aspectos que a veces no son tenidos suficientemente en cuenta para los diseñadores. Son realmente efectivas si el evaluador la ha preparado de manera que conduce la misma para tratar los temas que son realmente necesarios.

- Logging

También es conocida es a técnica como “grabación de uso”, se basa en recoger todas las actividades realizadas por el usuario con el sistema para su posterior análisis. Se debe tener una aplicación secundaria, no percibida por el usuario, que realice la labor de recoger las actividades automáticamente.

- Cuestionarios

Es menos flexible que la entrevista, pero se puede realizar a un número mayor de personas. Normalmente se suele usar la entrevista con un número pequeño de personas para sacar los aspectos más destacados, y con éstos realizar un cuestionario que se pase a número mucho mayor de usuarios.

Test

En los métodos de usabilidad por test, algunos usuarios representativos trabajan en tareas utilizando el sistema o el prototipo, y los evaluadores emplean los resultados para ver cómo la interfaz soporta a los usuarios en sus tareas.

Los métodos principales de evaluación por test son:

- Medida de las prestaciones

Tiene como primer objetivo mejorar la usabilidad del producto gracias a realizar el test con usuarios reales realizando labores habituales.

- Thinking aloud

Este método, también descrito por Nielsen, pide a los usuarios que expresen en voz alta sus pensamientos, sentimientos y opiniones mientras que interaccionan con el sistema, o un prototipo. Es muy útil en la captura de un amplio rango de actividades cognitivas.

Se realiza con usuarios únicos que expresan libremente todo lo que piensan sobre el diseño y funcionalidad del sistema.

- Test retrospectivo

Esta técnica se utiliza como complemento de las demás. Se trata de realiozar alguno de los métodos anteriores y grabarlo en vídeo, así se puede volver a ver indefinidamente, para capturar todos los detalles.

Implementación

Es la fase en la que se agrupa todo el trabajo de codificación de la aplicación que se ha ido construyendo hasta ese punto.

Hay que escoger previamente a la codificación, las herramientas que se van a utilizar, así como el lenguaje de programación que se adapte al proyecto y todo la tecnología que necesitemos para el sistema.

Esta fase depende en gran medida del diseño realizado, ya que en base a éste se programará la aplicación, cumpliendo cada uno de los requisitos. Y si se ha realizado un buen diseño, siguiendo los pasos del modelo de la Usabilidad y la Accesibilidad, se obtendrá un sistema implementado que cumplirá ambos principios **[24]**.

Lanzamiento

Esta fase, suele ser una de las más críticas de todo el proceso. Es el punto en el que se ven concretadas en mayor e o menor medida las expectativas puestas en el producto. El éxito total del producto dependerá de dos factores muy importantes:

1. La comodidad del usuario, entendiendo ésta como que no le aparezcan continuos errores, no le resulte complicado utilizarlo y recuerde con facilidad el manejo de las diferentes opciones

2. Que proporcione a los responsables los resultados esperados

Si se sigue el modelo de proceso de la Usabilidad y la Accesibilidad, se asegura que ambos factores se vean satisfechos. Ya que al contar con los usuarios durante todo el proceso de desarrollo, por un lado son participes del propio proyecto, y dado que lo han evaluado, no muestran ningún problema de aprendizaje del sistema **[24]**.

Una vez instalado el producto, y puesto en fase de pruebas durante un cierto periodo de tiempo, se recoge lo que se llama “feedback” del usuario, posibles problemas que se han encontrado los usuarios. A partir de ahí, se hacen modificaciones, y se vuelve a dejar en fase de pruebas, hasta que se obtenga una satisfacción total por parte del usuario **[32]**.

4. ANÁLISIS Y DISEÑO DEL SISTEMA

Problemas de desarrollo del sistema que nos compete

En este proyecto nos enfrentamos a los problemas que derivan de la creación de una interfaz de usuario en un dispositivo móvil para avisos de emergencia a las personas mayores.

La interfaz de usuario es el principal punto de contacto entre el usuario y el sistema, es la parte que el usuario ve, oye, toca y con la que se comunica. Dependiendo de la experiencia del usuario con la interfaz, el sistema puede resultar muy útil para el usuario, o de lo contrario, no prestarle ninguna ayuda. El tipo de problemas que origina una interfaz de usuario pobre incluye la reducción de la productividad, un tiempo de aprendizaje inaceptable y niveles de errores inaceptables que produce frustración y probablemente el desechar dicho sistema **[20]**.

Al enfrentarse con el diseño de un sistema interactivo hay que tener en cuenta dos aspectos fundamentales de éste. El primero es centrarse en los requerimientos que éste debe tener en cuanto a interactividad y eficiencia. Para ello necesitamos trabajar los aspectos psicológicos del usuario, la ergonomía del equipamiento, los aspectos sociales, etc. Esto se conoce como "dominio del comportamiento", ya que la interacción se describe desde el punto de vista del comportamiento del usuario y la interfaz trabajando conjuntamente **[21]**.

El segundo aspecto, es el que se denomina "dominio constructivo" y está asociado al diseño como sistema informático, ya que un sistema interactivo es un producto informático **[21]**.

Ambos aspectos podemos unificarlos en la metodología conocida como "diseño centrado en el usuario". Esta metodología de diseño se basa en que las personas puedan llevar a cabo sus actividades productivamente con simplicidad, fiabilidad, seguridad, comodidad y eficacia **[21]**.

El acercamiento que se pretende con las interfaces a las personas humanas, se basa en ofrecer modos de interacción cada más simples y naturales. El proceso de adaptación y naturalidad se está logrando a través de la capacidad de entrada y salida de forma multimodal.

Como se expone en el proyecto *Las telecomunicaciones y la movilidad en la sociedad de la información:* "La problemática asociada al desarrollo de las tecnologías de interfaz debe concentrarse sobre la exigencia de ofrecer modos de interacción simples y con un alto grado de naturalidad, adaptados a los futuros terminales y redes de comunicación. Es en este campo donde dichas tecnologías se enfrentan a su mayor reto: buscar la integración de los diferentes modos de comunicación (visual, oral, auditivo, gestual, etc.) para ofrecer nuevas y más potentes vías de interacción con el usuario, que suelen englobarse bajo el término "interacción natural", superando así las limitaciones de las interfaces actuales" **[14]**.

El objetivo último de esta "interacción natural", es la posibilidad de que los usuarios usen las múltiples formas de comunicación con las que interactuamos los humanos a través de los distintos canales sensoriales, lo que supone recurrir a múltiples modos de interacción **[14]**. Estos diferentes modos pueden ser **[22]**:

- "Un mensaje visual en una pantalla: hacer que aparezca un texto, ampliar un icono en una ventana, mover un cursor por la pantalla o un cambio completo de la información que aparezca en la pantalla".

- Audible: un sonido de atención, un comentario grabado u otro tipo de sonido

- Táctil: A través de un lápiz en la pantalla o con la propia mano, o al pulsar la tecla con más o menos presión, etc.

Las interfaces multimodales, buscan integrar los diferentes modos de interacción, de manera que la comunicación hombre-máquina se produzca de la forma más natural posible **[3]**.

Las ventajas que presenta una interfaz multimodal frente a una interfaz unimodal son inmensas, por ejemplo si disponemos de una interfaz sólo por voz y tenemos que dar una gran lista de nombres sería mucho más eficaz poder dar esta lista de manera visual, así dependiendo de la información que se quiera dar es conveniente poder usar uno u otro modo. También depende de donde nos encontremos será mejor utilizar uno u otro modo, es decir en un museo, por ejemplo, sería un gran inconveniente tener que utilizar un sistema que interaccione por voz, ya

que el resto de la gente tendría que oír tus instrucciones, sería más conveniente utilizar un modo táctil o visual.

Además, como se expone en el proyecto *Las telecomunicaciones y la movilidad en la sociedad de la información* "estudios recientes han demostrado que para la realización de una misma tarea, una interfaz basada en la utilización de un lápiz electrónico y la voz es entre 3,2 y 8,7 veces más rápida que una interfaz visual tradicional, y también un diez por ciento más rápida que una interfaz vocal unimodal" **[14]**.

Si nos fijamos en la recuperación ante errores y la robustez, el poder utilizar más de un modo de interacción nos evita muchos de estos errores, ya que si tenemos, por ejemplo, dos modos de reconocimiento, uno por voz y otro por escritura, el sistema tendrá muchos menos fallos, ya que ambos se realimentarán y podrán recuperarse de diversos errores **[5]**. Aquí aparecen las ventajas de las tecnologías de seguridad multimodal, "en este sentido, la integración de varios modos de autenticación biométrica (voz, huella, firma, cara, etc.) surge como una de las alternativas más prometedoras para superar las importantes limitaciones tecnológicas de cada modo aislado, y alcanzar así los elevados niveles de seguridad exigidos por el uso y acceso de futuros servicios y aplicaciones seguras" **[14]**.

La multimodalidad, también nos aporta, la ventaja de que personas con deficiencias o personas mayores puedan utilizar estos sistemas, ya que el uso de diferentes modos de presentación de la información abre las puertas a personas con problemas de visión, de oído o problemas cognitivos. Por ejemplo, no es lo mismo que una persona mayor tenga que utilizar una lupa para poder leer en la pantalla de su móvil a tener la opción de poderlo escuchar.

El tipo específico de información que nuestro sistema va a proporcionar son notificaciones de emergencia (terremotos, atentados, inundaciones, etc.). Dado que el momento en que se va a proporcionar esta información es un momento de mucha tensión y nerviosismo el uso de diferentes canales sensoriales para suministrar la información es necesario, está muy justificado que debemos usar tanto la notificación visual como la verbal para dar a conocer la información.

Análisis de requisitos:

En la actualidad, el diseño de cualquier aplicación necesita del conocimiento del usuario final para lograr una interacción adecuada y el aprovechamiento total del software. Dentro de este conocimiento del usuario, hay que saber sus necesidades y sus limitaciones no solo de la persona sino también del entorno donde se va a utilizar la aplicación, ya que no es lo mismo diseñar un GPS para llevarlo en el coche a uno para tenerlo en casa **[23]**.

Este proyecto como está dirigido a un grupo muy determinado de personas, este apartado cobra aún mayor importancia. La extracción con éxito de los requisitos, hace que el sistema aporte ventajas reales a este colectivo, y con ello cumple el objetivo principal en la realización de este proyecto, ayudar a las personas mayores en situaciones de emergencia.

Para analizar lo requisitos se va a utilizar la metodología explicada en el apartado número cuatro del proyecto, el modelo de porceso de la Usabilidad y la Accesibilidad.

Extracción de requisitos:

Para la extracción de los requisitos se va a hacer un análisis del sistema y la población a la que va dirigida, para ello nos remitiremos a los estudios realizados hasta la fecha, con los que se han elaborado las normas para la creación de software dirigido a personas mayores.

La recogida de requisitos la vamos a dividir en tres grupos principales de análisis **[24]**: el análisis etnográfico, el análisis de implicados y el análisis contextual de tareas.

- Análisis etnográfico:

Este análisis cubre el entorno en el que se va a utilizar el sistema, hay que describir el contexto.

El contexto, en el que se va a utilizar este sistema, es una situación de emergencia, estas situaciones han sido estudiadas en psicología, y en artículos como "El comportamiento humano en situaciones de emergencia" se expone: "La mayoría de las personas no han tenido la experiencia de hallarse ante una situación de peligro inminente y cuando esto ocurre algunas personas toman decisiones que incrementan al peligro para ellas y también para los demás. Así, los comportamientos que se producen van desde una actitud de calma hasta un verdadero pánico" **[25]**. Por tanto,

hay que ver que requisitos debe cumplir el sistema de manera que no incremente esta situación de pánico o estrés, y ayude en un momento así.

A través de este sistema lo que queremos es notificar de la emergencia, luego lo que buscamos es transmitir la información de manera adecuada a la situación, para ello hay que tener en cuenta ciertos requisitos:

- La información debe ser la suficiente para que no se generen dudas, y se actúe en función de lo que ocurre **[25].**
- La información debe ser adaptada al grupo de personas mayores **[25].**
- Se debe utilizar todos los posibles modos de comunicación (multimodalidad), para garantizar la adecuada percepción y el conocimiento **[25].**
- Se tiene que evitar la abundancia de información, ésta debe ser explícita, clara y comprensible para el usuario **[25].**

o Análisis de implicados:

En este apartado se va a estudiar a todos los que participan en el desarrollo y puesta en marcha de la aplicación, pero se va a estudiar a los usuarios finales, es decir vamos a analizar las necesidades y limitaciones de éstos de cara a obtener los requisitos de nuestro sistema.

Podemos basarnos en la norma española UNE 139802 **[26],** en la cual aparecen los requisitos que debe cumplir cualquier aplicación informática de manera que pueda ser utilizada por personas con discapacidad y personas mayores. En la norma aparecen los principios generales que se deben cumplir, y se especifica luego para cada aspecto (sonido, pantalla, etc.) qué requisitos han de cumplirse. Dichos requisitos están divididos en tres prioridades, prioridad 1, prioridad 2 y prioridad 3, desde los más básicos hasta los más sofisticados **[26]**. En nuestro sistema al estar desarrollado para personas mayores, habrá requisitos que hay que cumplir de prioridad 1 pero también habrá de prioridad 3.

Los requisitos que utilizaremos para nuestro sistema serán una selección de los que aparecen en la norma, ya que escogeremos los adecuados para este sistema en particular.

Principios generales **[26]**:

Prioridad 1

- El software debe estar diseñado para minimizar el número de pasos que debe realizar el usuario para activar cualquier opción.
- Se debe permitir al usuario elegir dispositivos de entrada/salida.
- Se debe poder manejar el software de forma efectiva utilizando sólo uno de los posibles dispositivos de entrada.

- Las aplicaciones deben utilizar los servicios ofrecidos por el sistema operativo para facilitar su accesibilidad.
- Las aplicaciones no deben desactivar o interferir en las características de accesibilidad del sistema operativo o de otros productos.

Prioridad 2

- Se debe proporcionar una función que permita a los usuarios deshacer los efectos de acciones no intencionadas. Si una acción no puede deshacerse, se debe pedir confirmación antes de realizarla.
- Si el usuario cambia de tarea, al regresar a la anterior, su interfaz debe recordar cuál era el control que tenía el foco.
- Se debe permitir la definición de perfiles con las preferencias del usuario.
- Cada usuario debe poder cambiar y mantener sus propias preferencias de usuario mediante la interfaz del sistema.
- Las aplicaciones deben usar los servicios estándar de entrada/salida del sistema operativo.

Prioridad 3

- Se debe permitir intercambiar rápidamente dispositivos alternativos para la entrada/salida.

- Se debe permitir al usuario transferir sus preferencias a otro sistema compatible.

Teclado **[26]**:

Prioridad 1

- Se deben poder activar todas las funciones (incluyendo la navegación) sólo mediante teclado.
- El usuario debe poder cambiar de una ventana de trabajo o aplicación a otra utilizando el teclado.
- Los comandos de navegación por teclado no deben activar los objetos de interfaz.
- Se deben ofrecer alternativas a la pulsación simultánea de varias teclas.

Prioridad 2

- La navegación entre elementos de la interfaz debe ser circular.
- Las aplicaciones deben respetar las convenciones de funcionamiento del teclado en el sistema operativo.

Pantalla **[26]**:

Prioridad 1

- Todos los textos presentados en pantalla deben ser generados mediante las funciones del sistema dedicadas a mostrar texto.
- No debe usarse el color como única fuente de información.
- Deben existir opciones para modificar el tipo de letra, el tamaño y el color de todos los controles de la interfaz.
- Se debe evitar presentar elementos que parpadeen o destellen con una frecuencia entre 2 y 50 Hz.

Prioridad 2

- Todos los iconos deben poder tener asociada una etiqueta de texto y debe existir la posibilidad de visualizar sólo esa etiqueta.
- Debe poder ajustarse el tamaño y posición de las ventanas.
- Deben proporcionarse opciones para minimizar, maximizar, restaurar y cerrar las ventanas.
- El usuario debe poder ajustar el tamaño de iconos y otras imágenes.
- El usuario debe poder ajustar, de forma individual o en grupos, la posición de aquellos iconos y objetos gráficos que puedan ser activados.
- La interfaz de usuario debe adaptarse a la configuración de contraste, color, tamaño y demás atributos de visualización que haya definido el usuario en el sistema operativo.
- Debe existir al menos un modo de presentación de información visual que sea legible para usuarios con agudeza visual entre 6/18 y 6/60 sin depender del sonido.
- Deben proporcionarse combinaciones de colores predefinidas que hayan sido diseñadas teniendo en cuenta las necesidades de las personas con deficiencias visuales.
- El sistema debe proporcionar acceso a la información que se presenta fuera de la región visible de la pantalla.
- Las etiquetas de los campos de entrada o visualización de datos de los formularios deben estar próximas a estos campos.

Sonido y multimedia **[26]**:

Prioridad 1

- Los contenidos relevantes en formato audio o vídeo deben ofrecerse también en otros formatos alternativos.
- El usuario debe poder activar la presentación visual de avisos sonoros.
- Deben ofrecerse funciones que permitan enviar cualquier información textual a una salida mediante síntesis de voz.
- La salida en síntesis de voz debe aparecer inmediatamente después de ocurrir el evento que la originó.

Prioridad 2

- El usuario debe poder ajustar el volumen de los sonidos.
- Debe existir la posibilidad de ajustar la frecuencia fundamental de los avisos sonoros.

Notificación al usuario **[26]**:

Prioridad 1

- Los mensajes emitidos deben ser cortos, sencillos y redactados en un lenguaje claro para el usuario no técnico.
- Los mensajes del mismo tipo deben ser claramente identificables: siempre deben aparecer en la misma posición de pantalla, deben tener el mismo formato y deben estar etiquetados de forma unívoca y estándar.

Información de objetos **[26]**:

Prioridad 1

- Se debe proporcionar a otras aplicaciones información semántica sobre los objetos de la interfaz de usuario.
- Todos los controles, objetos, iconos e imágenes de la interfaz de usuario deben tener un texto asociado que indique su función o significado.
- Las ayudas técnicas deben poder acceder a las características de los objetos de la interfaz de usuario.

Tiempo **[26]**:

Prioridad 1

- El usuario debe poder pausar o detener la presentación dinámica de información.
- La información sobre errores o los avisos relevantes para la tarea actual deben persistir hasta que el usuario confirme su lectura.
- Si se requiere una respuesta del usuario en un intervalo de tiempo determinado, se debe poder ajustar dicho intervalo, incluyendo la posibilidad de desactivar todos los límites de tiempo.

Documentación **[26]**:

Prioridad 1

- La documentación del producto debe estar redactada de la forma más clara y sencilla posible, dentro del vocabulario del dominio de la aplicación.

- La documentación del producto debe estar disponible en formatos alternativos bajo petición del usuario, ajustándose a sus necesidades específicas y sin coste adicional.
- La información sobre las características de accesibilidad del producto debe estar disponible en formatos alternativos bajo petición del usuario, ajustándose a sus necesidades específicas y sin coste adicional.

Otros **[26]**:

Prioridad 1

- Las aplicaciones deben ofrecer la opción de finalizar.

- Análisis contextual de tareas

En este análisis se busca especificar y entender los objetivos de los usuarios. Hay que distinguir entre los objetivos funcionales y los objetivos no funcionales. Los objetivos funcionales, será la lista donde aparecerán cada una de las funciones que debe realizar el sistema; y los objetivos no funcionales son los objetivos de accesibilidad que debe cumplir la aplicación para que pueda ser un usada por el colectivo de las personas de la tercera edad **[24]**.

Los objetivos básicos funcionales de nuestra aplicación son:

- Avisar de manera gráfica y sonora de una emergencia.
- Dar al usuario la información necesaria en caso de emergencia
- Poder mostrar la información para afrontar una emergencia en cualquier momento

Los objetivos no funcionales, son el conjunto de requisitos del análisis de implicados que debemos utilizar en nuestra aplicación, no para cubrir las funcionalidades de la aplicación, sino para poder hacer un uso más fácil y eficiente de ella.

- Se tiene que minimizar los pasos para realizar cualquier opción.
- Se debe poder definir perfiles con las características del usuario
- Deben existir opciones para modificar el tipo de letra, el tamaño y el color de todos los controles de la interfaz.
- Los iconos deben tener asociada una etiqueta de texto, y debe poder visualizarse sólo dicha etiqueta.
- Los contenidos relevantes en formato audio o vídeo deben ofrecerse también en otros formatos alternativos.
- Deben ofrecerse funciones que permitan enviar cualquier información textual a una salida mediante síntesis de voz.

- La salida en síntesis de voz debe aparecer inmediatamente después de ocurrir el evento que la originó.
- Los mensajes emitidos deben ser cortos, sencillos y redactados en un lenguaje claro para el usuario no técnico.
- Los mensajes del mismo tipo deben ser claramente identificables: siempre deben aparecer en la misma posición de pantalla, deben tener el mismo formato y deben estar etiquetados de forma unívoca y estándar.
- El usuario debe poder pausar o detener la presentación dinámica de información.
- La información sobre errores o los avisos relevantes para la tarea actual deben persistir hasta que el usuario confirme su lectura.
- Las aplicaciones deben ofrecer la opción de finalizar.

Diseño de la aplicación

Plataforma de funcionamiento

Antes de centrarse el proyecto en el diseño de la aplicación en concreto, se va a mostrar el escenario global en el que se desarrolla la aplicación.

Se utiliza una técnica vista en el apartado de los prototipos (en el epígrafe "Metodología del proyecto"), conocida como "escenario", que consiste en recrear una situación real en qué usaríamos la aplicación. Utilizar esta técnica antes del diseño es realmente útil, ya que permite contestar a pregunta de "qué elementos necesitamos para el correcto funcionamiento de la aplicación"

Este escenario estará compuesto por el usuario, el dispositivo móvil del usuario, donde se encuentra la aplicación, y el servidor de red al que estará conectado el usuario.

Figura 16.- Esquema del escenario de funcionamiento de la aplicación

En el momento en que se produzca una emergencia, el servidor será informado a través de Internet, y éste dará el aviso al dispositivo móvil. El usuario deberá estar suscrito a dicho servidor, y estar conectado a la red permanentemente, esto no es un problema ya que ahora los principales distribuidores de Internet en los dispositivos móviles ofrecen conexión permanente, y se paga por los datos descargados no por el tiempo de conexión. Así con este sistema, sólo se pagará por el aviso ya que la aplicación es totalmente independiente de la red y está instalada en el propio dispositivo.

Además, dado que la aplicación es independiente de la red, se podrá consultar fuera de red, para mirar las indicaciones, establecer una configuración adecuada al usuario o familiarizarse con la aplicación antes de que se produzca una emergencia.

El objetivo de este proyecto, es diseñar un prototipo de la aplicación que las personas mayores tendrán en sus teléfonos móviles, por tanto la parte de red quedará pendiente para posteriores trabajos.

Una vez vistos los elementos necesarios para el correcto funcionamiento del sistema, se pasa a la fase del diseño de la aplicación que se va a desarrollar en este proyecto.

Análisis de tareas

En este punto se rediseñarán las tareas del usuario, para racionalizar la organización del trabajo y explotar las capacidades que la automatización proporciona. Se trata de poner en orden la parte funcional de la información obtenida del análisis de requisitos.

Las tareas que se van a realizar por medio de la interfaz son estructuradas en tres puntos:

- Avisar de una emergencia
- Informar de los pasos a seguir en caso de emergencia
- Consulta de información de las distintas emergencias que puedan producirse

Avisar de una emergencia

Se mostrará una pantalla en la que aparecerá lo que ha ocurrido (un terremoto, un huracán, una inundación, o un atentado), y se escuchará un sonido de alarma.

Informar de los pasos a seguir en caso de emergencia

Se mostrará una pantalla con las indicaciones que hay que seguir en caso de producirse dicha emergencia, y se escuchará estas mismas indicaciones por voz

Consulta de información de las distintas emergencias que puedan producirse

En todo momento, aunque no se haya producido ninguna emergencia, el usuario podrá acceder a la aplicación y consultar los pasos a seguir en cada uno de los posibles casos de emergencia, para así familiarizarse con la aplicación y estar más preparado si se produce alguna emergencia.

Modelo conceptual

Ahora, se establece el modelo conceptual que es clave para el desarrollo posterior de la interfaz. Para ello se ha hecho uso de los requisitos analizados anteriormente y de un diagrama UML de actividad, que especifica el comportamiento de la aplicación y define la lógica de cada operación. A partir de

este diagrama se va a entrar en un diseño más detallado donde se especifique la navegación en cada una de las pantallas.

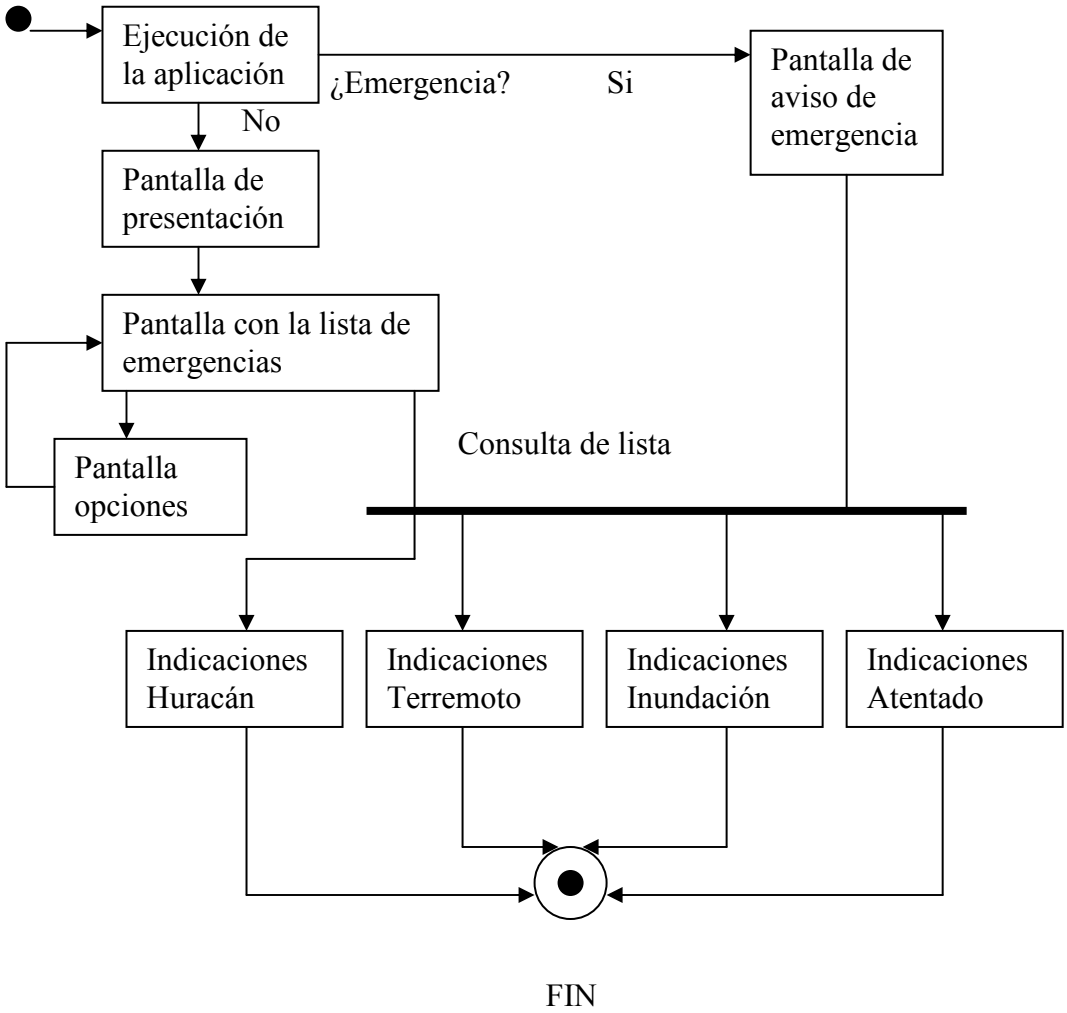


Figura 17.-Diagrama UML de la actividad de la aplicación

Como se puede ver en el diagrama UML de actividad de la aplicación, ésta al ejecutarse tendrá dos maneras de inicializarse, la principal es al producirse una situación de emergencia, en ese momento aparecerá una pantalla que nos indique peligro, y la otra manera es que el usuario puede entrar en la aplicación, como en cualquier otra de su dispositivo para simplemente ver las recomendaciones en caso de emergencia.

Si la aplicación se inicializa al producirse una emergencia, tras esta pantalla podremos ir a otra donde se explica que debemos hacer en este caso, o simplemente salir ya que conocemos los pasos a seguir.

Si es el usuario el que accede al sistema, tiene la opción de ver qué hacer en cada uno de los casos, o ir a una pantalla de opciones donde cambiar el tamaño de la letra o el volumen del sonido.

Una vez visto el diseño para cubrir los objetivos funcionales, habrá que ver como cubrimos los objetivos no funcionales.

Si se produce una emergencia, la pantalla inicial de aviso, nos avisará gráficamente a través de texto y un icono simbólico, además aparecerá un sonido constante de alarma. De esta forma, el aviso se produce con dos modalidades asociadas, lo que produce un mayor entendimiento por parte del usuario.

Los mensajes de emergencia serán muy semejantes, sólo cambiará el nombre de la emergencia dependiendo de la que sea, así el usuario puede ver que son mensajes de un mismo tipo, aparecen con el mismo formato e incluso el mismo sonido de alerta.

Las opciones que hay una vez que aparece la pantalla de alerta, son salir, ya que siempre se debe dar la opción de finalizar, o ir, en este caso iría el usuario a la pantalla en la cual le dan instrucciones sobre qué hacer en este caso, con esto se consigue minimizar los pasos a realizar por el usuario.

La pantalla de alerta y el sonido que la acompaña se mantienen hasta que el usuario decide hacer alguna de las opciones, con lo que así nos aseguramos que ha leído el mensaje.

Si no se ha producido ninguna emergencia, pero el usuario decide entrar en la aplicación, se encontrará una primera pantalla con el nombre de la aplicación, y en la que tendrá las opciones de salir (siempre tiene que poder finalizar) o entrar. Una vez que el usuario entra se encuentra una pantalla en la que aparecen una lista con los distintos tipos de emergencia, pulsando en cualquiera de ellos podrá ir a la información de qué hacer en cada una de las emergencias. Habrá también dos comandos, uno para salir de la aplicación, y otro llamado opciones. Si el usuario pulsa el comando opciones, este nos llevará a una pantalla donde podrá modificar el tamaño de la letra o el volumen del sonido y así adecuarlo a sus necesidades, además estos cambios quedarán guardados para posteriores iniciaciones de la aplicación, es una manera de guardar el perfil, aunque siempre pueden ser modificados. El diagrama se puede visualizar en la figura 18.

Una vez que el usuario pulse uno de los tipos de emergencia, aparece en una pantalla en la que se nos da unas instrucciones o indicaciones sobre qué tener que hacer en caso de ocurrir esa emergencia, además aparece una pequeña imagen que simboliza lo que ocurre, y esta misma información es sintetizada por voz, ya que un largo texto es muchas veces incomodo de leer en un teléfono móvil. Esta pantalla es la misma que se mostrará cuando el usuario pulse ir en la pantalla de emergencia.

Para personas que decidan leer estas instrucciones en un lugar público, sin llamar mucho la atención, existe la opción de silenciar el mensaje sintetizado por voz. En caso de quererlo escuchar otra vez, existe el comando reproducir que nos devolverá nuevamente la información sintetizada por voz.

El esquema conceptual del diseño sería el que se muestra en las siguientes figuras.

Si hay una situación de emergencia:

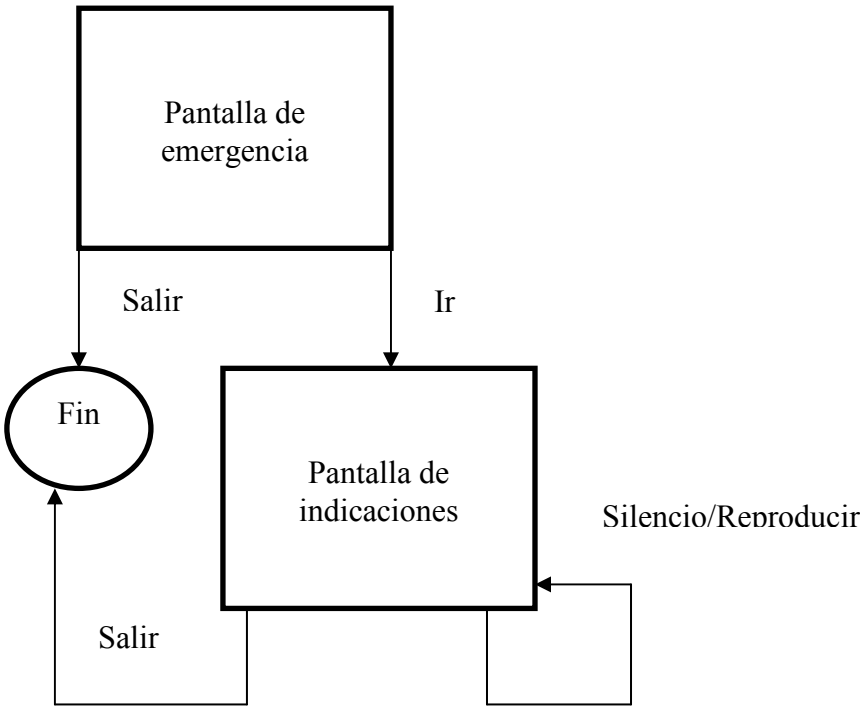


Figura 18.- Esquema de diseño de la aplicación en caso de emergencia

Se puede ver en la figura 18, que el aspecto más importante del diseño es su sencillez, es decir, debido al momento en el que se va a utilizar, lo más conveniente es que la complejidad sea la mínima y los pasos a realizar por el usuario también sean los menos posibles, ya que éste estará nervioso y la capacidad de razonamiento será menor.

Si no se ha producido ninguna situación de emergencia:

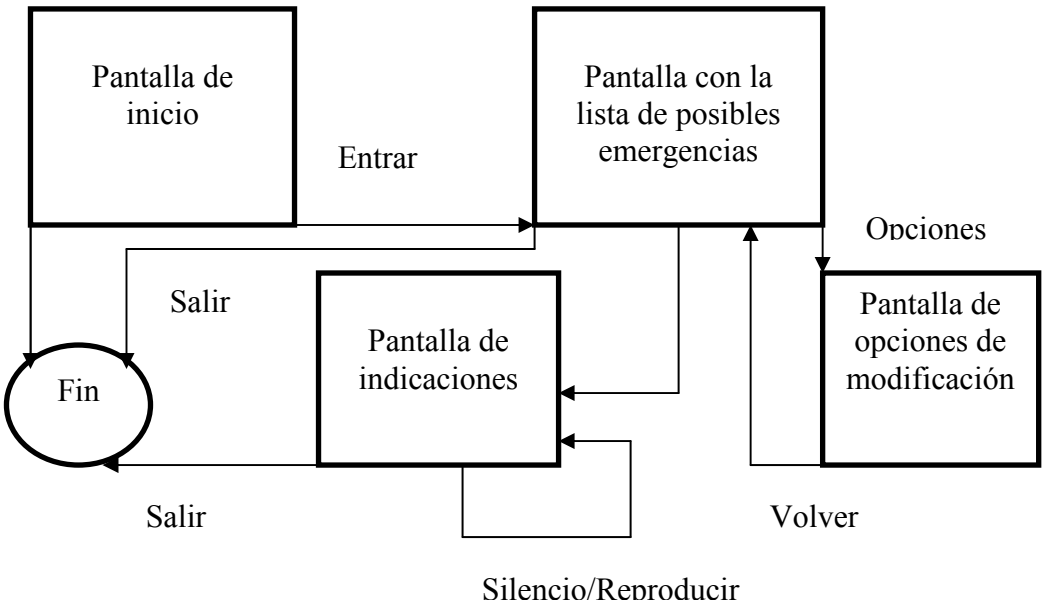


Figura 19.- Esquema de diseño de la aplicación si no se produce emergencia

El esquema de la figura 19 es más complejo, debido a que la utilización se puede dar en momentos de menos tensión, y por tanto con mayor capacidad de razonamiento. Sin embargo, también se busca conseguir los objetivos con el menor número de pasos posible.

Estilo

Dado que es una aplicación dirigida a un grupo de usuarios con unas características muy definidas, no se sigue un estándar general, sino un estándar particular que obedece a la norma española UNE 139802 **[26],** en la cual aparecen los requisitos que debe cumplir cualquier aplicación informática de manera que pueda ser utilizada por personas con discapacidad y personas mayores.

Las metáforas visuales que se introducen siempre van acompañadas de un texto adicional, por si el usuario no interpreta su significado. Las metáforas que se han introducido en la aplicación son:

- Señal de tráfico de peligro, para indicar que se ha producido una emergencia.
- Icono de información al lado de cada una de las opciones de lista de emergencias
- Barras de progreso para modificar el volumen

En cuanto a los colores de la interfaz, se han seguido las indicaciones que aparecen el libro de Shneiderman **[9]**. Lo principal es que los colores de las letras y el fondo no sean muy parecidos, ya que perdemos contrates, y debido a que la interfaz es sobre la pantalla de un teléfono móvil y está dirigida a personas mayores que pueden tener problemas de visión, el contraste debe ser muy alto. Por ello, se ha utilizado sobre todo texto negro sobre fondo blanco, salvo la pantalla de aviso de emergencia que aparece texto rojo sobre fondo negro, dando a entender un mayor peligro.

5. IMPLEMENTACIÓN

Herramientas empleadas

Para la realización del proyecto, la plataforma de desarrollo que se ha empleado es J2ME.

J2ME es una arquitectura para diseñar aplicaciones Java en dispositivos móviles.

Las razones principales para utilizar J2ME son:

- Es una plataforma estándar para el desarrollo de aplicaciones en dispositivos limitados
- Se ha extendido como el lenguaje preferido entre los programadores, luego hay una mayor facilidad y rapidez en el desarrollo de nuevas aplicaciones
- Portabilidad de las aplicaciones entre diferentes dispositivos y distintos fabricantes

Se estructura en tres niveles:

- Máquina virtual

Se encarga de interpretar el bytecode (programa Java compilado) a código máquina (órdenes del sistema).
Proporciona la independencia de plataforma
Existen dos máquinas virtuales que se usan normalmente, CVM, orientada a dispositivos embebidos y electrónica de consumo; y KVM, diseñada para

dispositivos con poca memoria, capacidad de proceso limitada y con conexión de red inalámbrica.

- Configuración

Es el mínimo conjunto de clases disponibles en una categoría de dispositivos. Las categorías se establecen según requisitos similares de memoria y procesamiento.

Existen dos configuraciones, y cada una de ellas está asociada a una máquina virtual:

→CDC (Connected Device Configuration): Está dirigida a dispositivos con cierta capacidad computacional y de memoria. Asociada a la máquina virtual CVM.

→CLDC (Connected Limited Device Configuration): Concebida para dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica. Está asociada a la máquina virtual KVM. Los principales fabricantes de móviles la implementan en la mayoría de sus modelos.

- Perfiles

Conjunto de clases Java que complementan una configuración para un conjunto específico de dispositivos.

Son los encargados de definir APIs que controlan, el ciclo de vida de la aplicación, la interfaz de usuario y la gestión de eventos.

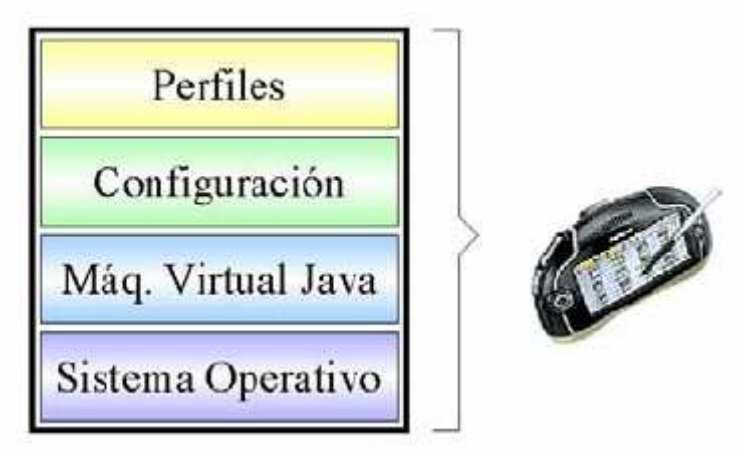


Figura 20.- Entorno de ejecución de J2ME

Actualmente el perfil más extendido en el mercado es el MIDP, está construido sobre la configuración CLDC, y fue el primer perfil definido para esa plataforma.

Las aplicaciones que se desarrollan para el perfil MIDP se denominan MIDlets. Un MIDlet es una clase Java que hereda de una clase abstracta de MIDP (javax.microeditionMIDlet), y que se ejecuta en el entorno de ejecución dentro de la máquina virtual, la cuál provee un ciclo de vida perfectamente definido y controlado a través de métodos de la clase MIDlet, que cada aplicación desarrollada debe implementar.

J2ME Wireless Toolkit 2.5

Es el conjunto de herramientas o entorno de desarrollo para J2ME proporcionado por SUN, y que vamos a utilizar para la implementación de nuestra aplicación.

Las razones para su utilización son:

→Descarga gratuita desde la página oficial de SUN
→Existen versiones para Windows y para Linux
→Contiene sus propios emuladores y puede integrar emuladores de terminales reales

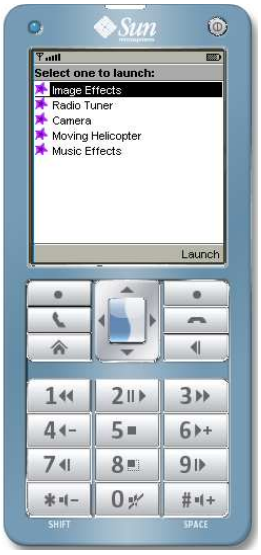


Figura 21.- Emulador "DefaultColorPhone" propio de la herramienta

Introducción a la implementación

La implementación del sistema se ha realizado con un MIDlet, que como se explica en el apartado de herramientas, es una clase Java que hereda de una clase abstracta de MIDP (javax.microedition.midlet.MIDlet).

MIDlet es como se denomina a las aplicaciones que se desarrollan para el perfil MIDP, que está construido sobre la configuración CLDC, que como se menciona en el apartado anterior, es la configuración concebida para dispositivos móviles dotados de conexión y baja computación gráfica. En estos dispositivos se carece de una línea de comandos desde donde se pueda ejecutar las aplicaciones, sino que disponen de un software encargado de ejcutar los MIDlets y gestionar los recursos que ocupan **[8]**.

El gestor de aplicaciones

El gestor de aplicaciones o Aplication Management System (AMS), es el software que gestiona los MIDlets. Pertenece al propio dispositivo, y cumple dos funciones muy importantes, la primera es la gestión del ciclo de vida de los MIDlets, y la segunda es controlar los estados por los que está el MIDlet mientras está en la memoria del dispositivo, que es cuando está en ejecución **[8]**.

El ciclo de vida de un MIDlet sigue 5 pasos funadamentales: localización, instalación,ejecución, actualización y borrado o desinstalación, en la figura 22 se observa un esquema de este ciclo.

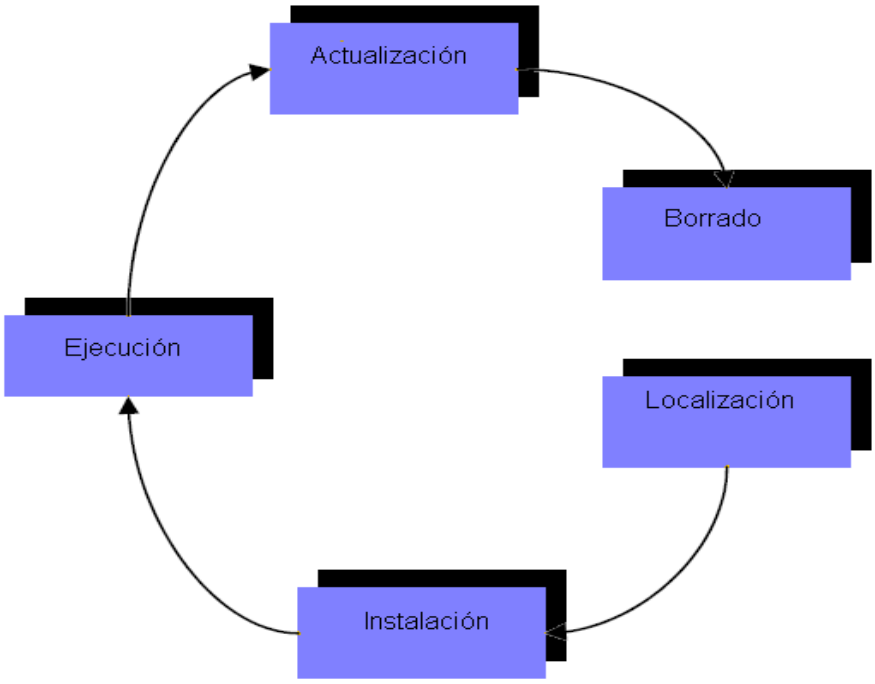


Figura 22. Ciclo de vida de un MIDlet.

El AMS, como ya se mencionó anteriormente, es el encargado de gestionar cada una de estas fases **[8]**:

1. Localización: En este paso seleccionamos a través del gestor de aplicaciones el MIDlet que queremos descargar. Por tanto, el gestor de aplicaciones nos tiene que proporcionar los mecanismos necesarios para realizar la elección del MIDlet a descargar.

2. Instalación: En esta fase el gestor de aplicaciones informa al usuario de la evolución de la instalación.

3. Ejecución: Durante la ejecución, el AMS tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan.

4. Actualización: El AMS 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, tiene que darnos la opción de realizar la actualización o no.

5. Borrado: En esta fase el AMS es el encargado de borrar el MIDlet seleccionado del dispositivo.

Los estados por los que pasa un MIDlet durante su ejecución también son gestionados a través del AMS. Normalmente en ejecución un MIDlet está en el estado "activo", pero al poder ser una aplicación dentro de un teléfono móvil, si se recibe una llamada, el gestor debe interrumpir la aplicación pero sin destruirla, el AMS lo pasará al estado de "pausa". Una vez que queramos finalizar la ejecución, pasaremos al estado "destruido", donde se libera la memoria volátil que estemos utilizando **[8]**.

Por tanto un MIDlet puede pasar por tres estados:

- Activo: Se encuentra en ejecución.
- Pausado: No consume ningún recurso compartido, y no está en ejecución hasta volver a pasar a activo.
- Destruido: No está en ejecución y no puede pasar a otro estado. Se liberán todos los recursos.

Un MIDlet puede cambiar de estado mediante una llamada a los métodos MIDlet.startApp(), MIDlet.pauseApp() o MIDlet.destroyApp(). En la figura 23, que hay a continuación, se aprecia como a través de la llamada a estos métodos, el gestor de aplicaciones cambia de uno a otro estado **[8]**.

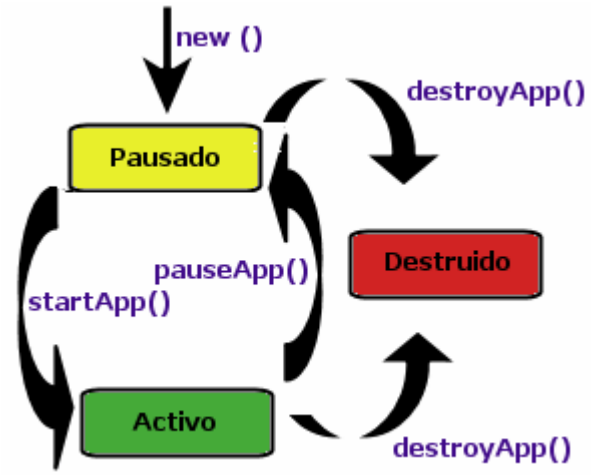


Figura 23.- Estados en ejecución de un MIDlet

La clase abstracta javax.microedition.midlet.MIDlet

Es la clase abstracta base para todos los MIDlets. Una aplicación tiene que extender de esta clase para que el AMS pueda gestionar sus estados y tener acceso a sus propiedades.

Los métodos más importantes que tiene esta clase son **[27]**:

- *protected MIDlet( )* : Es el constructor que nos sirve para crear un MIDlet, sin argumentos.
- *public final void notifyDestroyed( )*: Comunica al AMS que el MIDlet ha limpiado la memoria y ha termiando.
- *public final void notifyPaused ( )*: Comunica al AMS que el MIDlet está en pausa.
- *public final String getAppProperty (String key)*: Se le llama para obtener las propiedades del MIDlet.

Todos los MIDlets tienen un esqueleto común, ya que está clase posee tres métodos abstractos que hay que implementar, estos son **[27]**:

- *protected abstract void startApp ( ) throws MIDletStateChangeException*: Se llama, como se puede ver en la figura de los estados del MIDlet, cuando pasamos del estado pausado al activo, es obligatorio para comenzar la ejecución.
- *protected abstract void pauseApp ( )*: Se llama cuando el MIDlet pasa del estado activo al pausado.
- *protected abstract void destroyApp() (boolean unconditional) throws MIDletStateChangeException*: Se llama cuando el MIDlet pasa al estado destruido, la llamada puede hacerse tanto desde el estado activo como desde el pausado.

Análisis de la aplicación

La aplicación está formada por un solo MIDlet formado por un conjunto de clases anidadas.

La utilización de clases anidadas se justifica en que evita la proliferación de clases muy pequeñas que en muchos casos sólo se usan dentro de otra clase **[28]**, por tanto con el uso de clases anidadas, evitamos un gasto de memoria muy alto para estos dispositivos que poseen una capacidad reducida de memoria.

Además con el uso de clases internas se logra un control de la complejidad y de visibilidad de las clases, manteniendo un buen diseño orientado a objetos **[28]**.

La organización de las clases es la que se puede ver en el diagrama posterior, la clase contenedora o principal es Emergencias.java, que hereda de la clase MIDlet que, como se explicó anteriormente, es la clase base para todos los MIDlets. Y hay cuatro clases internas que forman el interfaz de usuario,dos pantallas del API de alto nivel (Formulario.java y Notas.java) y otras dos pantallas del API de bajo nivel (Alarma.java y Grafico.java). Más adelante se detalla cuál es la función de cada una dentro del grupo y su función individual.

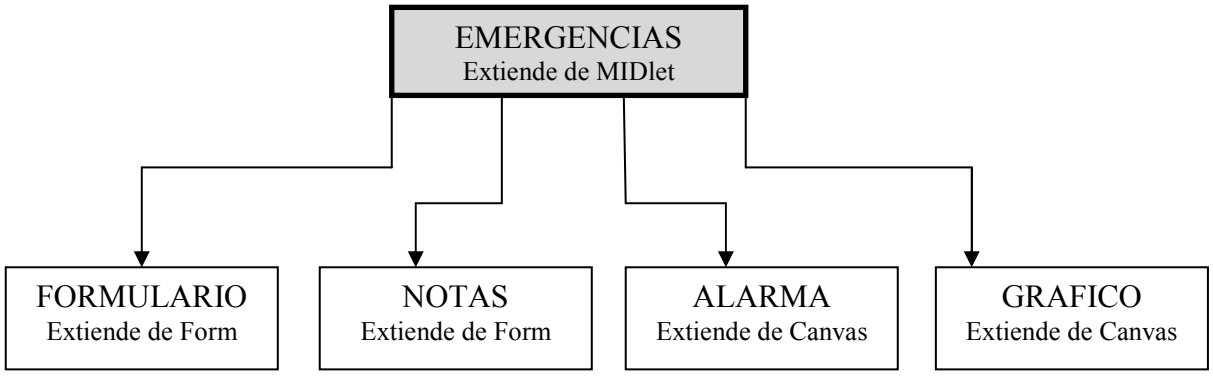


Figura 24.- Organización de las clases dentro del MIDlet

Antes de pasar a explicar la implementación de cada una de las clases, se va a detallar algunos de los APIs de J2ME que se han utilizado en la programación de esta aplicación, para lograr un mayor entendimiento del código implementado.

API del interfaz gráfico

El paquete que contiene los elementos que utilizaremos en la interfaz gráfica es el javax.microedition.lcdui.

Las interfaces de usuario en MIDP, se dividen en interfaces de usuario de alto nivel e interfaces de usuario de bajo nivel **[27]**.

Las interfaces de usuario de alto nivel utilizan componentes como botones, cajas de texto o formularios, no se tiene un control de la posición exacta de la pantalla donde se ubican, ya que su ventaja es la gran portabilidad que ofrecen al adaptar su tamaño y su posición al terminal en que se instale la aplicación.

Las interfaces de usuario de bajo nivel nos ofrecen un control gráfico de la pantalla a nivel de píxel, además estas APIs nos proporcionan un control completo sobre los recursos del dispositivo móvil, con lo que podemos capturar eventos de bajo nivel como la pulsación de teclas.

Sin embargo el uso de API de bajo nivel o alto nivel no es exclusivo, dentro de un MIDlet puedes tener pantallas basadas en Screen y pantallas basadas en Canvas.

Para pasar a explicar las clases que componen el API del interfaz gráfico, primero se va a explicar el modelo de eventos que sigue, ya que difiere del utilizado en Java.

Eventos en MIDP

Al desarrollar un MIDlet hay que proporcionar al usuario un mecanismo para navegar a través de las diferentes pantallas de la aplicación. La clase Command es la que va a implementar los comandos que se presentan en la pantalla del dispositivo móvil para permitir la navegación **[8]**.

El objeto Command encapsula el nombre y toda la información relacionada con la semántica de la acción correspondiente al comando. El objetivo principal es mostrar posibles acciones al usuario. El comando seleccionado por el usuario estará definido en un objeto de tipo CommandListener (el escuchador) asociado a la pantalla, y será lo que se tratará en el programa para saber qué acción realizar **[27]**.

→Eventos de alto nivel

El control de eventos de alto nivel se basa en el modelo listener o escuchador que se utiliza en Java. Las clases Screen y Canvas disponen de escuchadores de eventos de comandos. Un escuchador de eventos de tipo Command se crea simplemente implementando la interfaz CommandListener. Para registrar un escuchador de eventos a un comando hay que llamar al método setCommandListener ( ) **[27]**.

La interfaz CommandListener

Se implementa si se necesita recibir eventos de alto nivel de la aplicación. La intefaz sólo el método commandAction(Command c, Displayable d) que ha y que implementar, en donde indicaremos la acción que queremos que se realice cuando se produzca un evento en el Command c que se encuentra en el objeto Displayable d **[8]**.

La interfaz ItemStateListener

Para que la aplicación pueda recibir eventos correspondientes a cambios internos de los componentes de la interfaz de usuario incorporados en un objeto Form, hay que implementar la interfaz ItemStateListener. Los eventos se generan cuando **[8]**:

- Cambia el valor de un indicador de un objeto tipo Gauge
- Se modifica el valor de un TextField
- Se introduce una nueva fecha u hora en un DateField
- Se cambia la selección de un objeto de tipo ChoiceGroup

La interfaz ItemStateListener define el método itemStateChanged ( ) que se debe implementar por las clases que implementen esta interfaz. Además para registarar un escuchador de eventos de esta interfaz hay que llamar al método setItemStateListener ( ) de la clase Form.

La clase Display

Display es la clase principal del API de interfaces gráficas, la jerarquía que sigue este paquete se puede ver en la figura 25, que hay a continuación.

La clase Display representa el manejador de la pantalla y los dispositivos de entrada. Todo MIDlet debe tener asociado siempre un único objeto Display. En este objeto Display podemos incluir tantos objetos Displayables como queramos, los objetos Displayables son las pantallas que forman nuestra aplicación. La clase Display puede obtener información sobre las características de la pantalla del dispositivo donde se ejecute el MIDlet, además de ser capaz de mostrar los objetos que componen nuestras interfaces **[27]**.

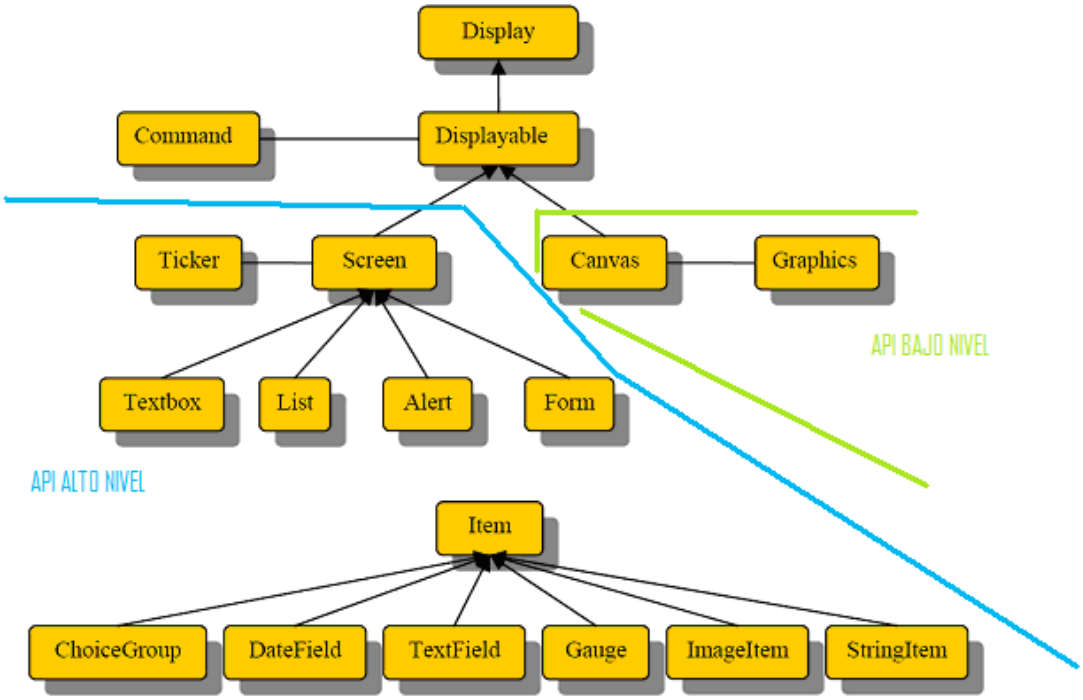


Figura 25.-Jerarquía de clases derivadas de Display e Item

La clase Display contiene los métodos para controlar la visualización de objetos Dispalyable. Los más utilizados son:

Método	Descripción
Static Display getDisplay(Midlet m)	Obtiene el objeto Display correspondiente a nuestro MIDlet
Displayable getCurrent()	Obtiene el objeto que se está visualizando
void setCurrent(Displayable d)	Establece el objeto a visualizar

Tabla 2.-Métodos de la clase Display

La clase Displayable

La clase Displayable representa las pantallas que hay en nuestra aplicación. Existen dos tipos de pantallas, las pantallas del API alto nivel y las pantallas del API bajo nivel, las de alto nivel descienden de Screen y las de bajo nivel descienden de Canvas, esto se puede apreciar en la figura de la Jerarquía de clases derivadas de Dispaly e Item **[27]**.

Displayable contiene los métodos encargados de manejar los eventos de pantalla y añadir o eliminar comandos. Los más utilizados son:

Método	Descripción
void addCommand(Command cmd)	Añade el comando cmd
void setComandListener(ComandListener cl)	Establece un listener para la captura de eventos

Tabla 3.-Métodos de la clase Displayable

La clase Command

La clase Command contiene información sobre un evento, se implementa en el MIDlet para detectar y ejecutar una acción simple, como ir de una pantalla a otra o ir a un menu de opciones.

Para construir un objeto Command hay que definir tres elementos **[27]**:

- Etiqueta: Texto que representa el significado del comando, y es lo que aparece en la pantalla
- Tipo: Entero que especifica qué va a relizar el comando. Los tipos definidos son BACK, CANCEL, HELP, EXIT, ITEM, OK, SCREEN y STOP.
- Prioridad: Entero que indica la importancia del comando, mayor importancia cuando menor sea el parámetro.

Pero para que el Command realice la acción que deseamos, además de crearlo tenemos que implementar la interfaz CommandListener, esta intefaz sólo contiene un método commandAction(Command c, Displayable d) en donde indicaremos la acción que queremos que se realice cuando se produzca un evento en el Command c que se encuentra en el objeto Displayable d.

Las tres clases anteriores son comunes a las interfaces gráficas tanto de bajo nivel como de alto nivel, ahora se va a explicar las clases que se utilizan en la aplicación, primero del API de alto nivel y luego del API de bajo nivel.

API ALTO NIVEL

La clase List

La clase List nos permite crear pantallas con una lista de opciones, lo que es muy útil para crear menús. Cada una de las opciones tiene asociada una cadena de texto y puede tener un icono de tipo Image asociado **[8]**.

Al ejecutar un comando de la lista, el sistema envía notificaciones a la aplicación, que podrán ser capturadas a través del CommandListener.

La clase List implementa la interfaz Choice, donde están definidos tres tipos de objetos Choice, lo que nos permite hace tres tipos distintos de lista, estos son:

1. Exclusivo (Choice.EXCLUSIVE)

Presenta al usuario la selección de un único elemento y su apariencia en pantalla se realiza mediante botones radio. Cuando el usuario selecciona un elemento, cualquier selección previa se desmarca automáticamente.

Figura 26.-Ejemplo de lista tipo Exclusivo

2. Implícito (Choice.IMPLICIT)

Se utiliza a la hora de realizar una selección rápida en la que solamente se requiere que el usuario seleccione un elemento. La presentación en pantalla es un menú de opciones en el que puedes desplazarte y elegir cualquier opción.

Figura 27.-Ejemplo de lista tipo Implícito

3. Múltiple (Choice.MULTIPLE)

Permite al usuario seleccionar varios elementos o ninguno, en cualquier combinación. En pantalla aparecen cajas de selección que indican si la opción está marcada o no.

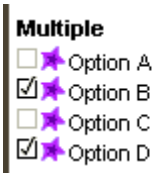


Figura 28-Ejemplo de lista tipo Múltiple

La clase Form

La clase Form actúa como contenedor de un número indeterminado de objetos. Todos los objetos que puede contener derivan de la clase Item. El número de objetos a insertar dependerá del tamaño de la pantalla, pero si el tamaño es mayor a la pantalla hará Scroll **[27]**.

Los métodos más usados para manipular los objetos Item almacenados en el Form son:

Método	Descripción
int append (Image imagen)	Añade una imagen al formulario
int append (Item elemento)	Añade un Item al formulario
int append (String texto)	Añade un texto al formulario
void delete (int numElemento)	Elimina el Item que ocupa la posición numElemento
void insert (int numElemento, Item elemento)	Inserta un Item antes del que ocupa la posición numElemento
void set (int numElemento, Item elemento)	Reemplaza el Item que ocupa la posición numElemento
Item get (int numElemento)	Devuelve el Item que se encuentra en la posición numElemento

Tabla 4.-Métodos de la clase Form

La clase Item

La clase Item es la superclase de todos los componentes de la interfaz gráfica que pueden ser incorporados a un objeto Form. Todos los objetos Item tienen una etiqueta de tipo String que corresponde al título del Item, y se presenta normalmente al lado del Item cuando éste aparece en la pantalla del dispositivo. Si la pantalla se desplaza el sistema intenta mantener la etiqueta visible al mismo tiempo que el Item **[27]**.

Los elementos que descienden de Item son:

Elementos	Descripción
ChoiceGroup	Item que implementa la interfaz Choice, pero sólo los tipos exclusivo y múltiple. Es similar a List pero puede aparecer en pantalla con otros Items
StringItem	Es una cadena no modificable de texto
ImageItem	Sirve para incluir imágenes
TextField	Texto que pueden manejar los usuarios
DateField	Fecha y hora editables por el usuario
Gauge	Indicador de progreso a través de un diagrama de barras.

Tabla 5.-Elementos de la clase Item

API BAJO NIVEL

La clase Canvas

La clase Canvas es una subclase abstracta de Displayable que permite realizar interfaces gráficos de bajo nivel en MIDP. Es la responsable del control de todo lo

que se dibuja sobre la pantalla y del control de todos los eventos que se producen en el dispositivo **[8]**.

Para programar en API de bajo nivel es necesario obtener el tamaño del display y tener en cuenta estas dimensiones, para ello existen los métodos int getWidth (), int getHeight () **[8]**.

Toda clase que extienda de Canvas debe implementar el método paint (void paint (Graphics g)), ya que es el encargado de pintar todos los píxeles de la pantalla.

Tiene varios métodos con los que gestiona los eventos de bajo nivel, como la presión de una tecla, pero dado que en este proyecto no se utilizan, no se detallará en su explicación.

La clase Graphics

La clase Graphics proporciona métodos para pintar elementos gráficos básicos como líneas, rectángulos, texto o imágenes sobre un objeto Canvas. Su funcionamiento es similar a java.awt.Graphics.

Un objeto Graphics se pasa como parámetro al método paint de un Canvas, y proporcionar para dibujar los diferentes elementos gráficos.

La clase Font

La clase Font nos permite seleccionar el tipo de letra, su tamaño y estilo.

API multimedia

Existen tres paquetes que proporcionan soporte para la generación, reproducción y grabación de datos multimedia **[27]**:

- javax.microedition.media
- javax.microedition.media.control
- javax.microedition.media protocol

Las principales características es que es de tamaño reducido, para poderse ejecutar en dispositivos con restricciones de memoria; es independiente del tipo de contenido o del protocolo utilizado; es extensible; puede soportar sólo un subconjunto (por ejemplo audio básico); y facilita al desarrollador incluir sólo determinada funcionalidad.

La especificación define cuatro partes principales que realizan el procesado multimedia **[27]**:

- *Datasource:* Encapsula y abstrae la gestión del protocolo, permitiendo al objeto Player tratar el contenido.
- *Player*: Lee y procesa los datos del Datasource, y finalmente es el encargado de controlar y ejecutar el contenido sonoro
- *Manager*: Controla los recursos de audio disponibles en el dispositivo, y es solicitado por la aplicación para crear los objetos Player.
- *Control*: Permite el acceso a los diferentes controles que proporciona el Player.

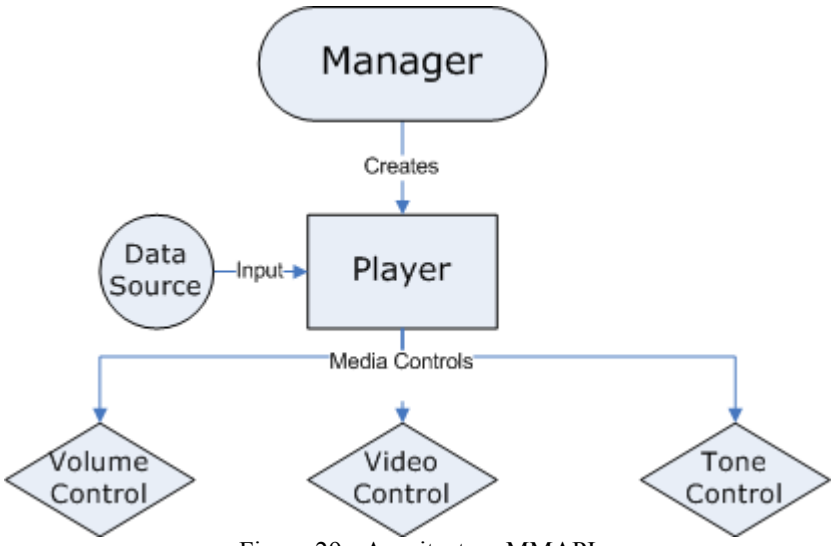


Figura 29.- Arquitectura MMAPI

La clase Manager

Es el punto de partida para procesar o reproducir datos multimedia. Puede reproducir tonos simples, crear objetos Player, e interrogar al dispositivo sobre protocolos y tipos que soporta.

Los métodos más importantes son **[8]**:

Método	Descripción
createPlayer (DataSource source)	Crea un Player a partir de un DataSource
createPlayer (String url)	Crea un Player a partir de una url
createPlayer(imput stream in, String tipo)	Crea un Player a partir de un recurso y un tipo
playTone (int nota, int duration, int volumen)	Reproduce un tono simple

Tabla 6.-Métodos de la clase Manager

La interfaz Player

La interfaz Player dispone de métodos para el control y reproducción de los diversos formatos multimedia.

Un objeto Player pasa por diferentes estados para controlar la asignación de recursos del dispositivo móvil. El ciclo de vida de un Player tiene cinco estados, y seis de los métodos de la clase Player controlan las transiciones entre estados **[27]**:

- UNREALIZED: Tras su creación
- REALIZED: Tras la llamada al método realice ( ), inicializa la información para adquirir recursos multimedia.
- PREFETCHED: Tras llamar a prefetch ( ), establece la conexión de red para el streaming de datos y otras inicializaciones. Al acabar el procesado vuelve a este estado.
- STARTED: Tras la llamada a start ( ), el Player puede procesar datos.
- CLOSED: Tras la llamada a close( ).

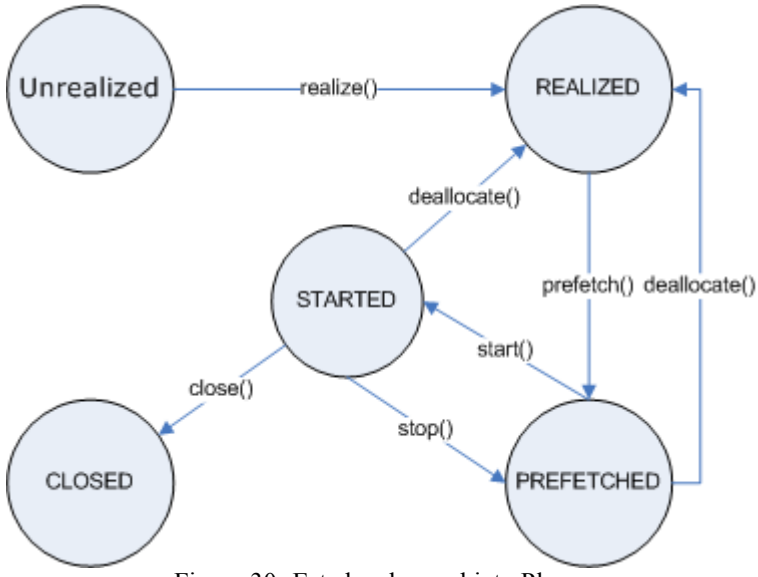


Figura 30.-Estados de un objeto Player

Una vez que el objeto Player está en el estado Realized se puede obtener información sobre el medio, como por ejemplo, la duración del audio o vídeo a través del método getDuration(), tenemos la posibilidad de variar el tiempo de reproducción a través del método setMediaTime(tiempo), o se puede poner el número de ves que se reproduce el medio a través del método setLoopCount (int numero) **[27]**.

Para saber en qué estado estamos o qué acción se ha producido en el objeto Player, éste genera eventos asíncronos que deberán ser capturados para poder manejar el reproductor. La recepción de eventos generados por el Player se realiza a

través de la interfaz PlayerListener, ésta se añade a un objeto Player con el método addPlayerListener(), y además debe implementar el método playerUpdate (Player player, String evento, Object datos).

Los tipos de eventos definidos para un objeto Player son **[8]**:

Evento	Descripción
CLOSED	Cuando se llama al método close() , se cierra el Player
DEVICE_AVAILABLE	Se libera un dispositivo disponible para el Player
DEVICE_UNVAILABLE	Una aplicación toma el control del dispositivo
DURATION_UPDATED	Se modifica la duración
END_OF_MEDIA	Cuando se llega al final de la reproducción
ERROR	Si ocurre un error
STARTED	El Player comienza
STOPPED	Cuando se llama al método stop(), se produce una parada del Player
VOLUME_CHANGED	El volumen de audio se ha modificado

Tabla 7.-Eventos de la interfaz Player

La interfaz Control

Permite el acceso a los diferentes controles que proporciona un Player. Un control es una manera de interactuar con el medio. Hay dos maneras de obtener un control de un Player, obtener un array de controles disponibles mediante el método getControls(), u obtener un control específico para un tipo de medio, a través del método getControl (nombreControl) **[27]**.

Los tipos de controles definidos más interesantes son:

Control	Descripción
RecordControl	Controla la grabación, graba lo que reproduce el Player
VideoControl	Controla la reproducción de vídeo
VolumeControl	Manipula el volumen de un Player

Tabla 8 -Controles de la interfaz Control

API de almacenamiento (RMS)

RMS permite almacenar datos de forma persistente para recuperarlos en un futuro, es decir, permite guardar registros entre distintas ejecuciones del MIDlet, independientemente del sistema de almacenamiento concreto del dispositivo **[8]**.

El paquete que soporta esta funcionalidad es javax.microedition.rms.

Las unidades principales del RMS son el registro (Record) y el almacén de registros (RecordStore).

El registro es el elemento individual de datos puede ser cualquier tipo de información contenida en una secuencia de bytes, a nivel de API es un array de bytes. El único problema es que no puede ser diferenciado en distintos campos, por tanto complica la tarea de organización al programador pero permite que el API RMS sea más sencillo y simple **[8]**.

El almacén de registros es una colección de registros, cada uno de ellos con un identificador único que se asigna consecutivamente según se van creando pero que no funciona como un índice. La estructura de un RecordStore se puede ver en la figura 31, que aparece a continuación.

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

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

Figura 31.- Estructura de un RecordStore

En la figura 31, aparece una cabecera, que está compuesta por: nombre, que identifica al almacén dentro de la suite de MIDlets, el nº de versión, que se actualiza automáticamente al insertar, modificar o borrar datos en el RecordStore, y el timestamp, que representa el número de segundos desde el 1 de enero de 1970 hasta el instante de la última actualización. El área de de almacenamiento de registros está formada por dos elementos, el identificador de registro (Record ID) y la información del registro o datos **[27]**.

Las operaciones más comunes que se van a realizar sobre un RecordStore son, abrir, leer, escribir y cerrar. Para realizar estas operaciones, se utilizan métodos estáticos, ya que no tiene constructor **[8]**.

Para abrir un RecordStore se utiliza el método:

```
static RecordStore openRecordStore (String nombre, boolean creaSiEsNecesario)
```

Existen otras dos versiones alternativas a este método que lo que hacen, es incluir más parámetros como la autorización, preguntar si se puede modificar por cualquier MIDlet, etc.

Para cerrar el almacén se usa:

```
closeRecordStore () throws RecordSiteNotFoundException, RecordStoreException
```

Se utiliza este método para cerrar la comunicación con el almacén una vez se acabe de trabajar con él, y hay que cerrarlo las mismas veces que se haya abierto si queremos cerrar de manera correcta la comunicación.

Para la manipulación de los registros, es decir la lectura, escritura, o borrado de éstos, existen distintos métodos **[27]**:

Método	Descripción
int addRecord (byte [] data, int offset, int numBytes)	Añade un registro al RecordStore
void deleteRecord (int Id)	Borra el registro Id del RecordStore
byte[] getRecord (int Id)	Devuelve el registro con identificador Id
int getRecord (int Id, byte[] buffer, int offset)	Devuelve el registro con identificador Id, en 'buffer' a partir de 'offset'
void setRecord(int Id, byte[] datos, int offset, int numBytes)	Sustituye el registro Id con el valor 'datos'

Tabla 9.-Métodos de la clase RecordStore

Para escribir un registro, se tiene que crear un array de bytes con la información que se quiere añadir, para ello se puede utilizar un flujo DataOutputStream, y una vez creado el array, se puede añadir el registro con el método int addRecord (byte [] data, int offset, int numBytes), que añade un registro devolviendo la posición de éste. O con el método void setRecord(int Id, byte[] datos, int offset, int numBytes), que sobreescribirá la posición indicada. Abajo se muestra un ejemplo de escritura de un registro **[8]**:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream ();
```

```
DataOutputStream dos = new DataOutputStream (baos);
dos.writeUTF(nombre);
dos.writeInt(telefono);
bytes [] datos = baos.toByteArray();
setRecord(id, datos, 0,datos.length);
```

Para leer un registro, si se dispone del identificador del registro podemos acceder a él con el método byte [] getRecord (int Id), y si se ha codificado la información utilizando un flujo, como en el ejemplo anterior, se puede decodificar con DataInputStream de la siguiente manera **[8]**:

```
byte [] datos = rs.getRecord (id);
DataInputStream dis = new DataInputStream (bais);
String nombre = dis.readUTF ();
int telefono = dis.readInt ();
dis.close ();
```

Emergencias.java

Es la clase principal, ya que hereda de javax.microedition.midlet.MIDlet, con lo que es la clase que gestiona los estados de vida del MIDlet a través de los métodos startApp(), pauseApp() y destroyApp(boolean unconditional). También implementa las interfaces CommandListener, ItemStateListener y Runnable, por tanto se encarga de manejar los eventos y las tareas de la aplicación.

```
public class Emergencias extends MIDlet implements
CommandListener,ItemStateListener,Runnable
```

En el constructor de la clase, se realizan todas las acciones previas a lanzar la aplicación el móvil (esto se hace a través del método startApp()), por tanto primero se obtiene el objeto Display, correspondiente a nuestro MIDlet, para ello utilizamos el método getDisplay(MIDlet m):

```
display = display.getDisplay(this);
```

Una vez obtenido el Display, se leen los registros para mostrar al usuario la misma configuración que dejó la última vez que utilizó el programa. Primero se abre el almacén, luego se lee cada registro y posteriormente se cierra la comunicación, cerrando el almacén.

```
   abrirRecordStore();
      try{
        leerRegistro(rs.getNumRecords()-1);
        leerRegistro(rs.getNumRecords());
      }
         catch(Exception e)
        {
          System.out.println("no puedo leer nada");
        }
cerrarRecordStore();
```

Ya establecida la configuración, se crean todos los objetos que aparecerán en aplicación, y se añaden a éstos los escuchadores de comandos e ítems que necesitamos.

En el método startApp(), se hace una diferencia entre lanzar la aplicación de manera "normal", el usuario entra para ver la información sobre las emergencias sin que haya ocurrido nada, o lanzar la aplicación al producirse una emergencia, en este caso el propio servidor que nos alerta sería en el encargado de lanzar la aplicación. En este proyecto, al no implementar la parte de red, simulamos el aviso del servidor

a través de un número aleatorio, al pulsar el botón "Launch" para lanzar la aplicación, sacamos un número aleatorio que puede valer entre 1 y 5, y sólo en el caso en que vale 3, en un 20% de las ocasiones, se produce emergencia, en el resto de casos se inicia de manera "normal". Si se lanza de manera "normal" aparece la pantalla inicial, que nos muestra un rótulo que pone "EMERGENCIAS", si se lanza al producirse una "emergencia", nuevamente obtenemos un número aleatorio, entre 1 y 4 para ver qué tipo de emergencia se produce y así ir a una pantalla u otra.

```
protected void startApp() {
        Random rnd = new Random();
        int dado;
       dado = (int)(rnd.nextDouble() * 5.0);

       if(dado==3)
        {
          Random rnd2 = new Random();
          int dado2;
          dado2 = (int)(rnd2.nextDouble() * 4.0);
        if(dado2==1)
          {
            display.setCurrent(alarmaHuracan);

         }
          if(dado2==2)
          {
            display.setCurrent(alarmaTerremoto);

          }
          if(dado2==3)
          {
            display.setCurrent(alarmaInundacion);

          }
          if(dado2==4)
          {
            display.setCurrent(alarmaAtentado);
           }
```

Si se ha producido una emergencia, además se llama a la tarea o hilo que nos permite reproducir el sonido de alarma.

```
t = new Thread( this );
        t.start();
```

Se utiliza un hilo para la reproducción del sonido, ya que el hilo nos permite la reproducción mientras que el interfaz de usuario se mantiene activo. Al llamar al método start(), se inicia la ejecución del hilo, llamamos al método run, y es dentro de este método donde se implementa lo que queremos hacer durante la ejecución del hilo.

```
    public void run() {
      try {
        liberaReproductor();
        System.out.println("liberas reproductor");
        if (dameTicker()==1&&numMusica==0)
        {
           InputStream is=getClass().getResourceAsStream("alarma.wav");
          p = Manager.createPlayer(is,"audio/x-wav" );
          //indicamos que se va a repetir infinitas veces
          p.setLoopCount( -1 );
        }
        else
        {

          if(numMusica==1)
          {

            InputStream is=getClass().getResourceAsStream("huracan.wav");
            p = Manager.createPlayer(is,"audio/x-wav" );

          }
          else if(numMusica==2)
          {
            InputStream is=getClass().getResourceAsStream("terremoto.wav");
            p = Manager.createPlayer(is,"audio/x-wav" );
          }
          else if(numMusica==3)
          {
            InputStream is=getClass().getResourceAsStream("inundacion.wav");
            p = Manager.createPlayer(is,"audio/x-wav" );

          }
          else if(numMusica==4)
          {
            InputStream is=getClass().getResourceAsStream("atentado.wav");
            p = Manager.createPlayer(is,"audio/x-wav" );
```

```
            }

        }

        p.realize();

        vc=(VolumeControl)p.getControl("VolumeControl");
        if (vc!=null)
        {
          vc.setLevel(dameVolumen());
        }

        p.prefetch();
        p.start();
      }
        catch( Exception e ) {
          System.out.println("problemas con reproductor");

          e.printStackTrace();

          p = null;
        }
    }
```

Tanto la variable numMusica como el método dameTicker modifican sus valores en función de la pantalla en que estemos, en este caso numMusica es igual a 0 y el método dameTicker devuelve un uno, por tanto se reproducirá el archivo de sonido "alarma.wav" infinitas veces porque al método del reproductor setLoopCount se le pasa un -1 como parámetro, y con el volumen configurado en la última vez que se utilizó la aplicación, si es la primera vez que se utiliza el programa, el valor por defecto es el máximo.

Dado que el flujo del programa es distinto si se ha producido una "emergencia" a si se inicia de manera "normal", se va a analizar la aplicación diferenciando ambos supuestos.

→Se inicia de manera "normal"

El método startApp() nos lleva a la pantalla inicial, esta pantalla es un objeto de tipo gráfico.

La clase gráfico es una de las clases anidadas que extiende de Canvas, la pantalla muestra una imagen que contiene un rótulo en el que se lee

"EMERGENCIAS", y el fondo se pinta de negro. Además tiene dos comandos, "salir" y "entrar".

Al pulsar en cualquiera de los comandos, el evento se maneja a través del método commandAction(Command c, Displayable d). Si el comando pulsado es salir se finalizará la ejecución del MIDlet, con una llamada al método destroyApp (boolean unconditional), y si el comando pulsado es entrar iremos a la pantalla llamada "información"

```
public void commandAction(Command c, Displayable d)
    {
      if(d.equals(pantallaInicial))


        if (c==comandoSalir)
        {

          destroyApp(false);
          notifyDestroyed();

        }
        else if(c==comandoEntrar)
        {

          display.setCurrent(informacion);

        }
      }
```

Antes de finalizar la ejecución del MIDlet, en el método destroyApp, tendremos que guardar en el almacén de registros la última configuración utilizada por el usuario. Para conseguir esto, se destruye el anterior almacén, se crea uno nuevo y es escriben los registros, así siempre se tiene un almacén con sólo dos registros ya que de lo contrario se estaría desperdiciando memoria del dispositivo móvil.

```
public void destroyApp(boolean unconditional)
    {
      destruirRecordStore();
      abrirRecordStore();
      escribirRegistro(posTamaño);
      escribirRegistro(posVolumen);
      try{
        leerRegistro(rs.getNumRecords()-1);
        leerRegistro(rs.getNumRecords());
     }
```

```
        catch(Exception e)
        {
          System.out.println("no puedo leer nada");
        }

      cerrarRecordStore();

    }
```

La pantalla información es un objeto de tipo List, una lista que como opciones tiene los diferentes tipos de emergencia, y además tiene asociados dos comandos, uno "salir" y otro "opciones".

Si el comando pulsado es salir, se finalizará la ejecución del MIDlet, igual que lo hacía en la pantalla anterior. Si el comando es opciones nos llevará a una nueva pantalla llamada "opciones".

Además en este caso al ser una lista, cada una de las opciones de ésta tienen función de comando, y al pulsar sobre cada una de ellas, primero iremos a una u otra pantalla en la que se dan las indicaciones a tomar en cada emergencia, luego modificaremos la variable numMusica, para indicar el mensaje que debe escucharse, y finalmente se ejecutará el hilo, que en este caso dependiendo del valor de numMusica reproducirá uno u otro mensaje en la pantalla asociada.

```
else if(c==List.SELECT_COMMAND)

        {
        switch (((List)d).getSelectedIndex())
          {
           case 0:
              display.setCurrent(huracan);
              numMusica=1;
              t = new Thread( this );
              t.start();
              break;
           case 1:
              display.setCurrent(terremoto);
              numMusica=2;
              t = new Thread( this );
              t.start();
              break;
           case 2:
              display.setCurrent(inundacion);
              numMusica=3;
              t = new Thread( this );
              t.start();
```

```
            break;
          case 3:
            display.setCurrent(atentado);
            numMusica=4;
            t = new Thread( this );
            t.start();
            break;
        }
      }
```

La pantalla opciones es un objeto de tipo Formulario, que es una clase que hereda de Form. La clase Formulario es donde establece las opciones de configuración el usuario, que son el tamaño del texto y el volumen del sonido.

Para modificar estas opciones se han usado dos objetos que heredan de Item, ya que estamos en un objeto que hereda de Form. El objeto utilizado para manejar el tamaño del texto es un ChoiceGroup con tres opciones posibles, pequeño, mediano y grande. El objeto utilizado para modificar el volumen es un Gauge, que es un objeto que visualmente representa perfectamente la idea de volumen, con cuatro posibles intensidades, de 0 a 100, que son 20, 50, 75 y 100.

```
String [] tipos= {"grande","mediano", "pequeño"};
 tamaño=new ChoiceGroup("Tamaño del texto",Choice.EXCLUSIVE,tipos,null);
volumen=new Gauge("Volumen",true,4,4);
```

Los objetos Items que modificamos además tienen un escuchador asociado, para que al lanzar un evento, podamos subir o bajar el volumen, o cambiar el tamaño del texto. El método con el que manejamos los eventos lanzados por los Items es el itemStateChanged (Item i).

```
          public void itemStateChanged(Item i)
    {
        if(i==opciones.tamaño)
      {



      switch(((ChoiceGroup)i).getSelectedIndex())
      {

        case 0:

          huracan.ponTamaño(Font.SIZE_LARGE);
          terremoto.ponTamaño(Font.SIZE_LARGE);
```

```
            inundacion.ponTamaño(Font.SIZE_LARGE);
            atentado.ponTamaño(Font.SIZE_LARGE);
            posTamaño=0;
            break;

          case 1:

            huracan.ponTamaño(Font.SIZE_MEDIUM);
            terremoto.ponTamaño(Font.SIZE_MEDIUM);
            inundacion.ponTamaño(Font.SIZE_MEDIUM);
            atentado.ponTamaño(Font.SIZE_MEDIUM);
            posTamaño=1;
            break;

          case 2:

            huracan.ponTamaño(Font.SIZE_SMALL);
            terremoto.ponTamaño(Font.SIZE_SMALL);
           inundacion.ponTamaño(Font.SIZE_SMALL);
            atentado.ponTamaño(Font.SIZE_SMALL);
            posTamaño=2;
            break;
        }
      }

    else if(i==opciones.volumen)
    {

       switch(((Gauge)i).getValue())
       {
          case 1:
            ponVolumen(20);
            posVolumen=1;
            break;

          case 2:

            ponVolumen(50);
            posVolumen=2;

            break;

          case 3:

            ponVolumen(75);
            posVolumen=3;
```

```
            break;

          case 4:

            ponVolumen(100);
            posVolumen=4;

            break;

      }
    }
```

Como se puede ver en el código, hacemos uso de los métodos ponVolumen (int i) y ponTamaño(int tam), el primero lo único que hace es dar un valor a la variable volumen y luego en el metodo run() que es donde se controla el volumen, a través del método dameVolumen() se establece un el volumen en la reproducción.

```
vc=(VolumeControl)p.getControl("VolumeControl");
        if (vc!=null)
        {
          //con el metodo setLevel se pone el nivel que el usuario haya seleccionado
          vc.setLevel(dameVolumen());
        }
```

El método ponTamaño (int tam), está dentro de la clase interna Notas y establece el tamaño del texto de las indicaciones que hay que tomar en cada emergencia.

```
      public void ponTamaño(int tam)
      {
        Font fuente=Font.getFont(Font.FACE_SYSTEM,Font.STYLE_PLAIN,tam);
        texto.setFont(fuente);
      }
```

Además, la clase Formulario, tiene asociado un único comando, “volver”, que al pulsar sobre él regresamos a la pantalla anterior, información, para que una vez establecidas las preferencias podamos seguir con la aplicación.

En la pantalla información, ya hemos visto que si se selecciona alguna de las opciones de la lista, iremos a la pantalla asociada y escucharemos a través del altavoz la información escrita. Las opciones posibles a escoger son huracán, terremoto, inundación o atentado, todos son objetos de tipo Notas, que es una clase que hereda

de Form, y contiene dos Items, un StringItem en donde aparecen escritas las indicaciones a tener en cuenta, y un ImageItem, una imagen que simboliza la emergencia. Pero dependiendo del número que se le pasa como parámetro al constructor, veremos un texto y una imagen distinta.

La clase Notas tiene tres comandos, "salir", "reproducir" y "silencio", al pulsar sobre el primero se finaliza la ejecución del MIDlet, como se hacía en los anteriores comandos "salir", pero en éste antes se libera el reproductor, para evitar problemas si en ese momento estaba en uso. Si se hace un clic sobre reproducir lo que se hace es iniciar de nuevo la reproducción de las indicaciones, por si el usuario no ha podido escucharlo bien, o quiere volver a escucharlo.

```
else if(c==comandoReproducir)
        {
          t=new Thread(this);

          t.start();
        }
```

Si se pulsa sobre el commando "silencio", lo que hace es quitar el volumen, por si el usuario se siente incómodo porque está en un sitio público, o simplemente no quiere escucharlo.

```
else if (c==comandoSilencio)
        {

          if(vc!=null)

          {
            if(vc.isMuted()==true)
              vc.setMute(false);
            else if(vc.isMuted()==false)
              vc.setMute(true);
          }

        }
```

→ Se inicia la producirse una emergencia

Si el número aleatorio que obtenemos en el método startApp() es un 3, se simula el momento de producirse una emergencia. Se obtiene un nuevo número aleatorio, esta vez entre 1 y 4, y dependiendo de éste, se simulará que se ha

producido un huracán, un terremoto, una inundación, o un atentado. Y sonará el archivo de sonido alarma.wav.

La primera pantalla en aparecer será alarmaHuracan, alarmaTerremoto, alarmaInundación o alarmaAtentado, todos objetos de tipo Alarma que es una clase que hereda de Canvas, y muestra en pantalla un texto indicando peligro más la emergencia que se ha producido, y una señal que indica peligro.

La clase Alarma contiene dos comandos, "salir" e "ir", si se hace un clic sobre "salir" se libera el reproductor y posteriormente se finaliza la ejecución del MIDlet, por si acaso el usuario ya conoce las indicaciones y no necesita mirarlas ni escucharlas. Si se pulsa el comando "ir", nos lleva a la pantalla donde se muestran las indicaciones referentes a la emergencia que se ha producido, además aparece una imagen que simboliza lo ocurrido y escuchamos las indicaciones por voz, esta pantalla es la misma que se ha explicado cuando se inicia la aplicación de manera normal y seleccionamos una posible opción de la lista.

```
else if(d.equals(alarmaAtentado))
      {

        if(c==comandoIr)

        {
          liberaReproductor();
          display.setCurrent(atentado);
          numMusica=4;
          t = new Thread( this );
          t.start();
        }

        else if(c==comandoSalir)
        {
          liberaReproductor();
          destroyApp(false);
          notifyDestroyed();
        }
   }
```

Ejemplo de aplicación

La aplicación tiene dos contextos diferentes de uso:

- Si no se ha producido una emergencia, modo "consulta".
- Si se ha producido una emergencia, modo "peligro".

Modo Consulta

El usuario entra en la aplicación como lo hace habitualmente en otras aplicaciones del teléfono, pulsa sobre el botón "Launch" (Lanzar), y una vez lanzada la aplicación aparece una pantalla con un título a modo de presentación. En esta pantalla aparecen dos botones que muestran dos posibles acciones a realizar por el usuario, "salir", ya que siempre hay que dar la oportunidad al usuario de finalizar la aplicación, o "entrar", para pasar al núcleo del programa. La pantalla inicial se puede ver a continuación, en la figura 32.

Figura 32.- Pantalla inicial

Una vez presionado el botón "Entrar", la aplicación lleva al usuario a una nueva pantalla donde aparece una lista con los posibles casos de emergencias, sobre cada una de estas opciones puede pulsar el usuario y le llevará a indicaciones propias del suceso seleccionado. Además aparecen dos botones más, "salir" para finalizar la ejecución y "opciones", que dirige a una pantalla donde se puede definir el perfil o configuración del usuario, eligiendo un tamaño del texto (grande, mediano o pequeño) y un determinado volumen, estos datos se quedan guardados para

posteriores ejecuciones de la aplicación. En esta pantalla aparece un botón "volver" que nos devuelve a la pantalla donde aparece la lista con las posibles emergencias. En la figura 33 se puede ver la pantalla con la lista de posibles emergencias y la pantalla de "opciones".

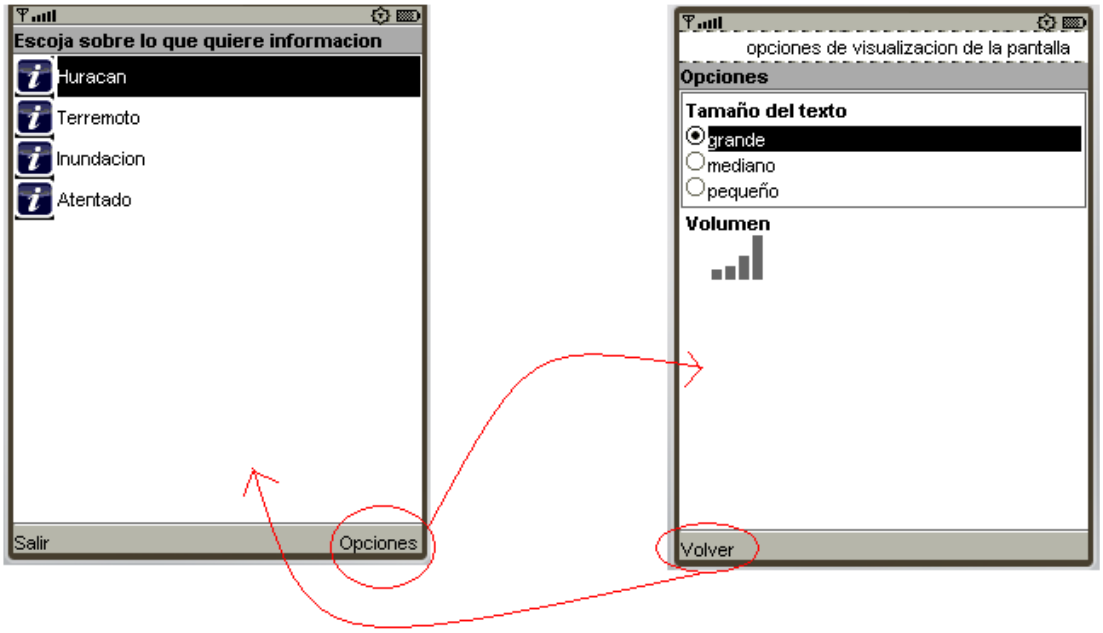


Figura 33.-Pantalla de la lista (izquierda) y pantalla de opciones (derecha)

Una vez establecidos los valores de la configuración, se puede pulsar sobre alguna de las opciones de la lista, y se verá una pantalla en la que aparecerá una imagen que simboliza la emergencia, se escucharán las indicaciones de qué hacer en ese caso, y si se baja con el "scroll" se podrán ver estas indicaciones escritas en texto. También aparecen dos botones en la parte baja de la pantalla, "salir", para cerrar la aplicación, y "menu", que al desplegarlo da la opción de "reproducir", para volver a escuchar las indicaciones, o "silencio", para no escuchar nada, por si el usuario se encuentra en un sitio público o prefiere no escucharlo por el altavoz .La pantalla que aparece, si se hubiese pulsado la opción de "huracán", se puede observar en la figura 34.

Figura 34.- Pantalla de indicaciones sobre un huracán

Modo peligro

Esta situación debería de indicarse por un servidor remoto, pero en este proyecto no se estudia la parte de red, por tanto la situación de emergencia se simula a través de un número aleatorio al ejecutarse la aplicación.

El tipo de emergencia que se produce también se simula a través de un número aleatorio, dependiendo del cual se producirá un huracán, un terremoto, una inundación o un atentado.

En este caso, la aplicación se lanza con una pantalla inicial advirtiendo de la emergencia a través de una señal de peligro, un sonido de alarma y un mensaje corto que avisa de lo producido. Los botones que aparecen en esta pantalla son, "salir", en caso de que el usuario ya conozca las medidas que debe tomar, o "ir", que nos lleva a la pantalla de las indicaciones a seguir en caso de haberse producido dicha emergencia, esta pantalla es la misma que se muestra en la aplicación al pulsar sobre una de las opciones de la lista, que se ha explicado antes. En la figura 35, se puede ver la pantalla inicial que aparece al simularse que se ha producido un terremoto.

Figura 35.-Pantalla inicial en caso de producirse un terremoto

6. EVALUACIÓN

En este proyecto debido a que es una aplicación interactiva, es fundamental hacer una evaluación de la usabilidad con los usuarios. El objetivo de la evaluación es comprobar si los potenciales usuarios se muestran satisfechos al manejar la aplicación, es decir si ésta es fácil de usar.

Para realizar la evaluación se ha utilizado la técnica de recorrido cognitivo, que se basa en hacer que los usuarios aprendan a manejar el software de forma exploratoria, para así evaluar la facilidad de aprendizaje del sistema. Además, a los usuarios se les realizó una entrevista semi-formal, en la que se usó una guía de preguntas, pero a medida que aparecían temas que le parecían interesantes se indagó en ellos, para obtener una opinión ajustada a sus propias necesidades y requerimientos.

Los usuarios analizados fueron, dos mujeres de mediana edad, un hombre y una mujer de unos 60 años y dos mujeres jóvenes.

El método seguido fue el siguiente: se les explicó en qué consistía la aplicación, cuál sería su uso y brevemente cómo funcionaba. A partir de ahí se les enseñó el mismo prototipo en dos plataformas diferentes, primero sobre el emulador "J2ME Wireless Toolkit 2.5" ,dentro del ordenador portátil, con el que se dejó que exploraran un tiempo indefinido hasta que vieron todas sus funcionalidades y aprendieron a utilizarlo. Luego, se les mostró el mismo prototipo instalado en el móvil Nokia 6300, con el que igualmente se les dejó que interactuaran y aprendieran a manejarlo.

Una vez acabado el período exploratorio, se pasó a realizar unas breves entrevistas con una guía de preguntas sobre usabilidad, donde se evaluó el tamaño del texto, la navegabilidad de la aplicación, los colores, etc. Pero estas preguntas se abrieron a los comentarios, sugerencias u opiniones de los usuarios, de manera que se obtuvo una visión más real de sus valoraciones.

En términos generales, los resultados obtenidos fueron realmente satisfactorios, ya que todos los usuarios valoraron positivamente la navegabilidad de la aplicación, la distinción de los colores utilizados, el tamaño del texto o la

funcionalidad de los botones, y un comentario generalizado entre ellos, fue el manejo sencillo y práctico del sistema.

Centrándonos en aspectos más concretos, les pareció más sencillo el uso de la aplicación en el teléfono móvil que en el emulador, ya que están más acostumbrados a usarlo, y valoraron negativamente el tamaño del texto de los botones de comandos en el emulador ya que les parecían muy pequeños, sin embargo en el móvil su tamaño les pareció correcto.

Si dividimos los resultados por edades, las dos personas jóvenes no tuvieron ningún tipo de problema, y los demás usuarios solo valoraron negativamente el tamaño del texto en los botones de comandos.

Un dato curioso es que a una de las mujeres de mediana edad que se le realizó la evaluación, no tenía teléfono móvil y sólo había usado alguno en contadas ocasiones. Y aun así el manejo de la aplicación le pareció sencillo, aunque destacó que le gustaría pulsar los botones directamente en la pantalla, la posibilidad de hacer esto sería utilizar un teléfono móvil con pantalla táctil.

Los usuarios también aportaron grandes ideas de cara a futuros trabajos sobe la aplicación, ya que ésta les pareció realmente interesante y práctica, quisieron aportar su ayuda para mejorarla. Algunas mejoras sobre usabilidad que plantearon fueron:

- Pedir la aprobación de salir al pulsar el botón de salida, utilizar la conocida pregunta: "¿Estás seguro de que quieres salir?". Esto ayudaría en caso de nerviosismo a no salir de la aplicación por equivocarse al pulsar sobre ese botón.

- Menor número de instrucciones: Ya que así se ganaría en claridad, aunque el número de instrucciones que aparecen en la aplicación no es muy alto.

- Ordenar la prioridad de las instrucciones por número, porque aunque están ordenadas por prioridades, los usuarios creen que un número delante les ayudaría a comprender más la prioridad de cada una.

- Dar las instrucciones a través de imágenes, ya que aunque les parece muy positivo que aparezcan a través de texto y voz, les resultaría más fácil que apareciesen también con imágenes de ayuda.

- Repetir continuamente las instrucciones por voz, ya que ahora para repetirlas hay que pulsar sobre el botón reproducir, y los usuarios piensan que es más útil que continuamente se repitan.

7. FUTUROS TRABAJOS

Esta aplicación puede ser el principio de un posible proyecto de gran alcance, ya que la idea que se intenta transmitir con la realización de este estudio, es que hay que buscar la manera de hacer llegar a todas las personas la información necesaria de qué tienen que hacer en una situación de emergencias, en la que debido al pánico no se sabe reaccionar o se reacciona de manera inadecuada.

Pero para que esta aplicación pueda comercializarse y extenderse al público hay que realizar una serie de mejoras, muchas de las cuales han sido sugeridas por los propios usuarios al realizar la evaluación, por lo que son de gran interés ya que son sus propias necesidades en estas situaciones. Las posibles mejoras que se han planteado son:

- Aumentar el número de emergencias a las que responde la aplicación, otras emergencias que podrían cubrirse serían incendios, accidentes de tráfico, etc. Esta mejora es la propia evolución del proyecto, ya que con él se ha hecho un pequeño ejemplo con sólo cuatro posibles emergencias, pero el trabajo de aumentarse el número de emergencias no costaría un gran trabajo de desarrollo y podría cubrir hasta ocho, doce, o todas las que sean necesarias.

- Otra mejora importante, es que al estar el servicio supuestamente conectado con protección civil, o algún otro grupo de ayuda en caso de emergencia, es que la primera indicación que apareciese y se dijese por voz fuera por ejemplo: "Estese tranquilo que los servicios de emergencia ya están en camino". Este mensaje proporcionaría a los usuarios una seguridad mayor y les tranquilizaría en gran medida. Introducir esta mejora en la aplicación tampoco supondría gran esfuerzo, ya que sólo sería introducir una indicación más al comienzo de la lectura de las instrucciones.

- Una mejora o un aspecto importante que debe tenerse en cuenta a la hora de llevar a cabo la implantación de esta herramienta en la población, es "¿a quién debe mandarse?". Dependiendo de la emergencia se deberá enviar a un radio de población menor o mayor, ya que no es lo mismo la población que hay que informar si se produce un terremoto a si se produce un incendio, por tanto esto es un tema importante a tratar si se quiere comercializar.

- Un aspecto que no se ha podido evaluar es la accesibilidad, ya que no hemos podido realizar la evaluación con personas discapacitadas, por tanto esto queda pendiente para futuros trabajos.

8. ORGANIZACIÓN DEL PROYECTO

Organización temporal

El calendario del proyecto es una parte muy importante de la planificación y programación del proyecto.

El calendario se puede presentar en una tabla o de forma gráfica. Lo más utilizado en la actualidad es el diagrama de Gantt.

El gráfico de Gantt permite identificar la actividad en que se estará utilizando cada uno de los recursos y la duración de esa utilización.

Este gráfico consiste simplemente en un sistema de coordenadas en que se indica:

→En el eje Horizontal: un calendario, o escala de tiempo definido en términos de la unidad más adecuada al trabajo que se va a ejecutar: hora, día, semana, mes, etc.

→En el eje Vertical: Las actividades que constituyen el trabajo a ejecutar. A cada actividad se hace corresponder una línea horizontal cuya longitud es proporcional a su duración en la cual la medición efectúa con relación a la escala definida en el eje horizontal.

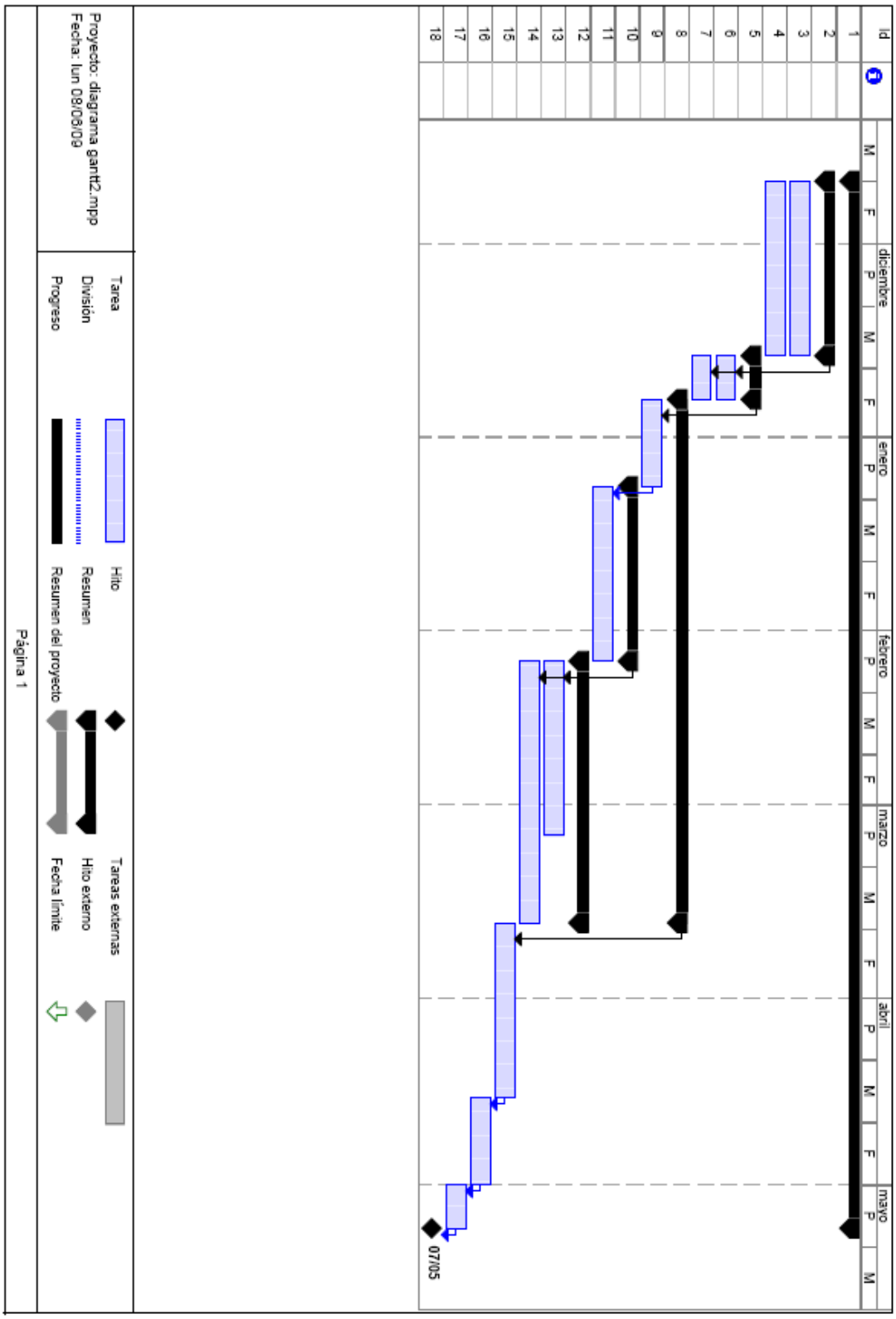
En este proyecto se han realizado dos diagramas de Gantt, uno se hizo al comienzo, para planificar el tiempo aproximado o esperado de cada una de las tareas y ver los costes estimados. Y otro se ha hecho al finalizar el proyecto, con los tiempos reales de duración de cada tarea y los costes finales, para así poder ver la variabilidad existente entre lo planificado y lo real.

Tareas del proyecto con la duración planificada inicialmente

Id	Nombre de tarea	Duración	Comienzo
1	**Sistema multimedia de información**	120 días	Vie 21/11/08
2	**Introducción en el proyecto**	20 días	Vie 21/11/08
3	*Lectura de literatura sobre el tema*	4 sem	Vie 21/11/08
4	*Estudio del estado del arte*	4 sem	Vie 21/11/08
5	**Organización del proyecto**	5 días	Vie 19/12/08
6	*Organización temporal*	1 sem	Vie 19/12/08
7	*Estudio de las herramientas de diseño*	1 sem	Vie 19/12/08
8	**Análisis y diseño del sistema**	60 días	Vie 26/12/08
9	*Problemas que se nos plantean*	2 sem	Vie 26/12/08
10	*Análisis de requisitos*	20 días	Vie 09/01/09
11	Extracción de requisitos	4 sem	Vie 09/01/09
12	*Diseño aplicación*	30 días	Vie 06/02/09
13	Iniciación en el lenguaje de programación	4 sem	Vie 06/02/09
14	Diseño	6 sem	Vie 06/02/09
15	**Implementación**	4 sem	Vie 20/03/09
16	**Evaluación**	2 sem	Vie 17/04/09
17	**Conclusiones**	1 sem	Vie 01/05/09
18	**Fin del proyecto**	0 días	Jue 07/05/09

Tabla 10.-Tareas organizadas al principio del proyecto

Diagrama de Gantt inicial

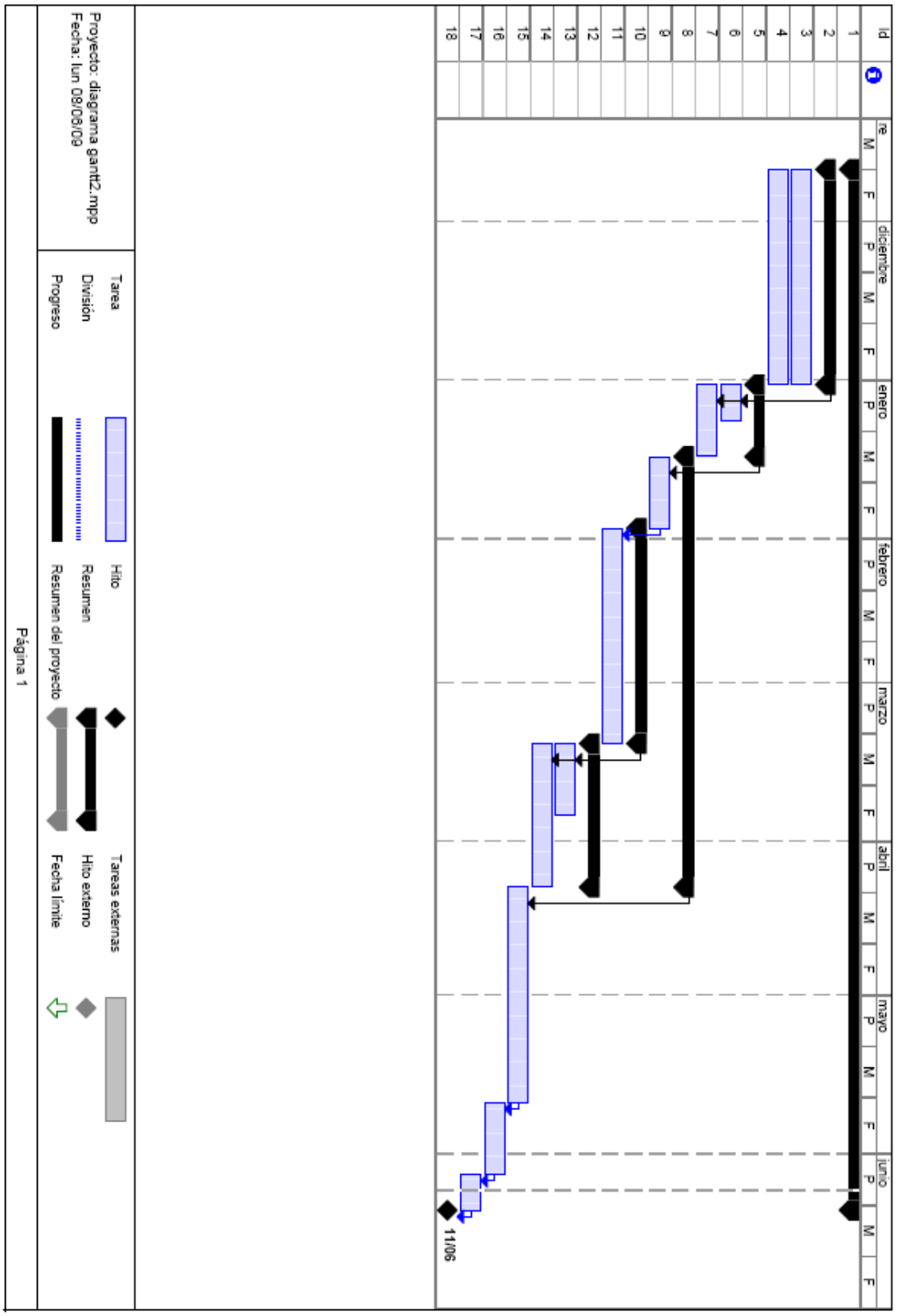


Tareas del proyecto con la duración final

Id	Nombre de tarea	Duración	Comienzo
1	**Sistema multimedia de información**	145 días	Vie 21/11/08
2	**Introducción en el proyecto**	30 días	Vie 21/11/08
3	*Lectura de literatura sobre el tema*	6 sem	Vie 21/11/08
4	*Estudio del estado del arte*	6 sem	Vie 21/11/08
5	**Organización del proyecto**	10 días	Vie 02/01/09
6	*Organización temporal*	1 sem	Vie 02/01/09
7	*Estudio de las herramientas de diseño*	2 sem	Vie 02/01/09
8	**Análisis y diseño del sistema**	60 días	Vie 16/01/09
9	*Problemas que se nos plantean*	2 sem	Vie 16/01/09
10	*Análisis de requisitos*	30 días	Vie 30/01/09
11	Extracción de requisitos	6 sem	Vie 30/01/09
12	*Diseño aplicación*	20 días	Vie 13/03/09
13	Iniciación en el lenguaje de programación	2 sem	Vie 13/03/09
14	Diseño	4 sem	Vie 13/03/09
15	**Implementación**	6 sem	Vie 10/04/09
16	**Evaluación**	2 sem	Vie 22/05/09
17	**Conclusiones**	1 sem	Vie 05/06/09
18	**Fin del proyecto**	0 días	Jue 11/06/09

Tabla 11.-Tareas organizadas al final del proyecto

Diagrama de Gantt final



Presupuesto

En este epígrafe del proyecto se realiza un cálculo del coste total estimado para la realización del sistema propuesto. El presupuesto se desglosa en dos partes: el coste de personal y el coste de material.

Coste de personal

Las tareas realizadas durante el proyecto son las expuestas en el diagrama Gantt, estas tareas han sido llevadas a cabo por una única persona, desde noviembre del 2008 hasta junio de 2009. La duración total ha sido de 6 meses y 20 días. Los primeros 5 meses se han dedicado unas 4 horas diarias (en días laborables) y un mes 20 días siguientes se han invertido 6 horas diarias (en días laborables).

Según los cánones establecidos, el coste de una hora de trabajo de ingeniero técnico de telecomunicaciones es de 45€. Si se suman todas las horas de trabajo se obtienen 656 horas, con lo que el coste del ingeniero técnico de telecomunicaciones es de 29520 €.

Al aumentar el tiempo en la realización del proyecto también aumentan los costes. Si se hubiera finalizado el proyecto en el tiempo que se estimó en al antes de comenzarlo, se hubiera ahorrado 23días laborables, lo que suponen 6210 €.

El salario del director del proyecto se estima de forma general, como un 7% del coste total del proyecto. Dado que el coste del proyecto, sin tener en cuenta el del director, es de 30300 €, el del director será 2121 €.

Personal	Cantidad	Honorarios	Importe
Ingeniero técnico de telecomunicaciones	1	50€/hora	29520 €
Director del proyecto	1	7% del total	2121 €
			TOTAL: 31641 €

Tabla 12.-Costes de personal

Coste del material utilizado

En este importe se incluyen tanto los dispositivos utilizados en el sistema como los costes de desarrollo de éste (equipos, licencias de software). Los costes asociados al material son:

Material	Unidades	Importe
PC portátil	1	600 €
Teléfono móvil	1	150€
Micrófono	1	30€
		TOTAL: 780 €

Tabla 13.-Costes del material utilizado

Coste total

Finalmente, el resumen de los costes totales, teniendo en cuenta la aplicación de 16% de impuestos, es el siguiente:

Costes	**Importe**
Costes del personal	31641 €
Costes de material	780 €
I.V.A. (16%)	5187.36 €
Coste total	**37608.36€**

Tabla 14.-Costes totales

9. CONCLUSIONES

Al comenzar el proyecto el conocimiento que tenía sobre la tecnología orientada al usuario, y más en concreto hacia las personas mayores, no era muy alto, sin embargo a medida que fui leyendo bibliografía sobre este tema y otros proyectos que se habían realizado, se despertó en mi un gran interés.

Desde un principio me di cuenta que realizar software orientado al usuario era dar un gran paso en el mercado. Durante muchos años se han creado y comercializado productos pensando en la demanda que éstos podían tener, basándose en la idea del diseñador y de los expertos de mercado, pero sin preguntar al verdadero cliente cuál era su necesidad. Ahora, conociendo la mecánica de las técnicas del Diseño Centrado en el Usuario, el producto antes de aparecer en el mercado se conoce hasta qué punto va ser demandado porque se conoce hasta qué punto es necesario. Por eso, el realizar un estudio previo de los potenciales clientes, y diseñar implicándoles a ellos, es el método adecuado para que el producto cumpla todas las expectativas.

El problema que existe en la actualidad sobre la notificación de emergencias, es que la última persona a la que se le informa es la persona implicada. En cuanto se produce alguna emergencia, los cuerpos de la policía, de los bomberos y de protección civil son avisados inmediatamente, proporcionándoles las indicaciones de lo que deben hacer, sin embargo son los propios afectados los que no son avisados hasta que llegan a socorrerles. Durante el tiempo en que ocurre la incidencia y vienen a atenderte, existe un gran desconcierto, porque no saben qué hacer, no saben qué les puede pasar y entran en una situación de nerviosismo y pánico que puede hacer que no sepan reaccionar o reaccionen de manera errática. Esto mismo se multiplica en personas mayores, que se sienten aun más inseguras en estos momentos y que debido a su edad su capacidad de reacción es menor. Por todo ello, el avisar en el momento en que se produce una emergencia de qué pasos hay que seguir a la población afectada, es una manera de proporcionarles tranquilidad, independencia y seguridad, con lo que se consigue una reacción más adecuada que puede llevar incluso a salvar vidas.

Debido a la importancia y responsabilidad que supone realizar un proyecto para ayudar en momentos tan difíciles, supone un gran reto utilizar las herramientas adecuadas para realizarlo y tener en cuenta cada pequeño detalle. En este proyecto el desarrollo técnico en sí no es la gran dificultad, sino el saber manejarlo para obtener un objetivo claro, la facilidad de uso. Ese ha sido el gran reto al que me he enfrentado en el proyecto, he tenido que estudiar exhaustivamente los requisitos necesarios para personas en momentos de nerviosismo y personas mayores. Pero el objetivo final propuesto al plantearse este proyecto, realizar una interfaz amigable con la que se pudiese notificar de diversas emergencias a las personas mayores, se puede decir que se ha cumplido, ya que los resultados de la evaluación han sido muy satisfactorios, y aunque siempre hay detalles que mejorar, en líneas generales ha cumplido con las expectativas del comienzo.

En cuanto a la organización del proyecto, se podría decir que a pesar de haberse retrasado la finalización un mes aproximadamente, debido a toda la información que ha habido que recopilar, se han cumplido los objetivos de acabar antes del verano, que era lo que se buscaba en un principio. Durante la realización, he aprendido sobre todo a organizar las diferentes tareas en que se divide el proyecto, ya que hasta ahora no me había enfrentado a un proyecto así, y me he dado cuenta de lo importante que resulta llevar una buena organización porque de lo contrario sería imposible finalizarlo.

Sobre las herramientas técnicas utilizadas, hay que mencionar especialmente que he trabajado con el lenguaje J2ME, versión de Java para dispositivos reducidos, y aunque ya había trabajado con este lenguaje en una asignatura durante la carrera, la realización del proyecto me ha hecho expandir mucho mis conocimientos y conocer prácticamente al detalle su API.

10. APÉNDICES

Java 2 SDK (Software Development Kit)

Se trata de un kit de desarrollo de software compuesto por un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema bastante concreto, por ejemplo, ciertos paquetes de software, frameworks, plataformas de hardware, ordenadores, videoconsolas, sistemas operativos, etc..

Es básicamente una interfaz de programación de aplicaciones o API (Application Programing Interface) creada para permitir el uso del lenguaje de programación de alto nivel Java. Las herramientas también incluyen soporte para detección de errores de programación y un entorno de desarrollo integrado o IDE (Integrated development Enviroment). Además, incluye códigos de ejemplo y notas técnicas de soporte para ayudar a clarificar ciertos aspectos del material de referencia primario.

Instalación para Windows XP

Se descarga del j2sdk versión 1.5 como mínimo para esta aplicación:
-http://java.sun.com/javase/downloads/index.jsp

1. Se hace doble clic en el programa de instalación del jdk (versión 1.6.0_14):
"jdk-6u14-windows-i586.exe"

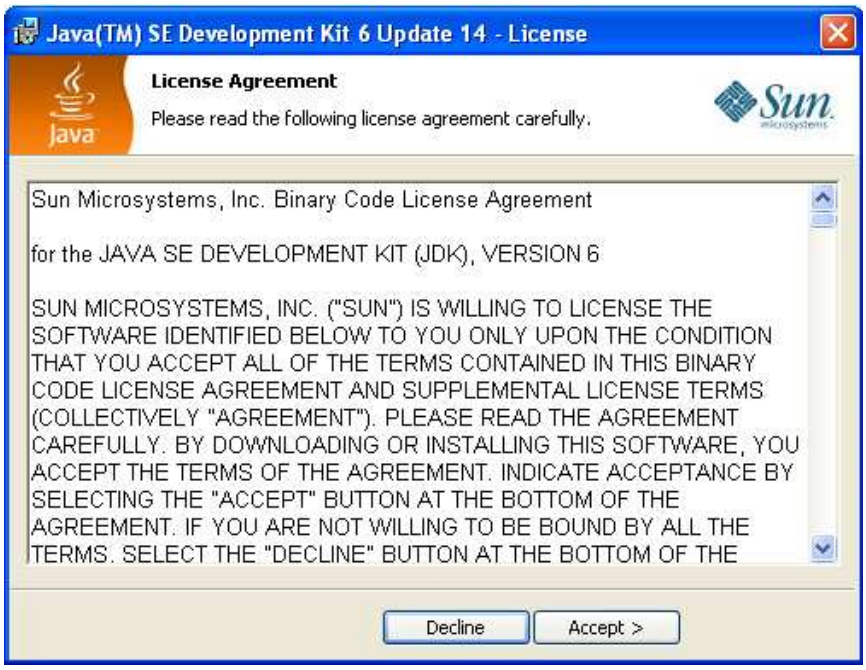


Figura 36.- Cuadro de licencia

2. Cuando aparece el cuadro de licencia, Figura 36, se pulsa "*Accept>*".

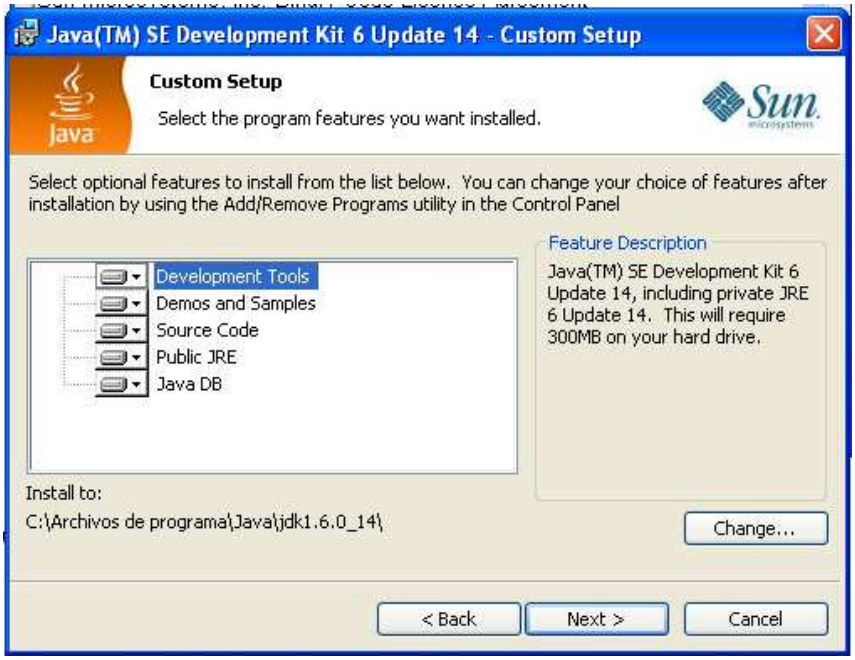


Figura 37.- Pantalla con los componentes a instalar

3. En la siguiente pantalla, Figura 37, aparecen los componentes que se quiere instalar, de modo que por defecto se instalarán todos, no se cambia nada y se pulsa "*Next>*".

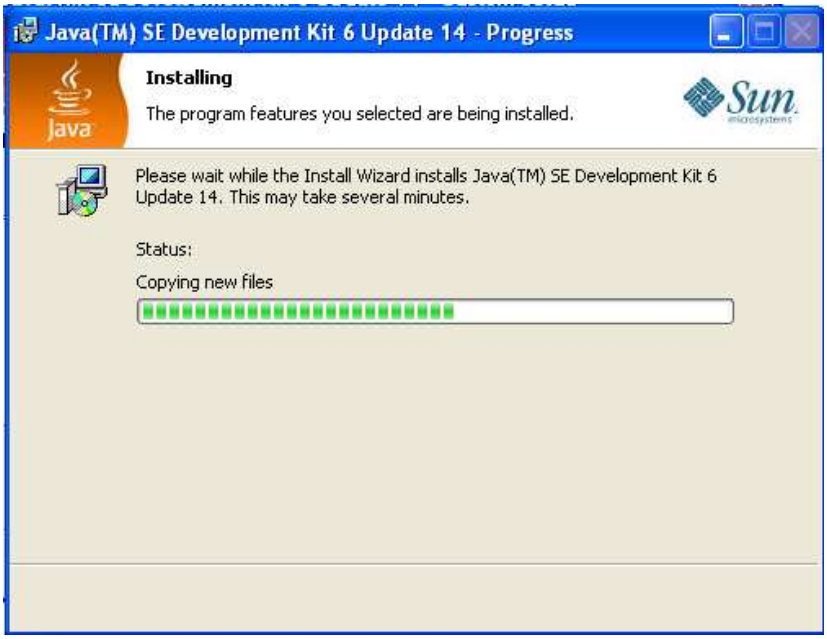


Figura 38.- Pantalla de progreso de la instalación

4. Se espera a que acabe de copiar todos los ficheros, aparece la pantalla de progreso de la instalación, la figura 38. Puede llevar varios minutos.

Figura 39.- Pantalla de instalación correcta

5. Cuando termine, aparecerá la pantalla de instalación correcta, figura 39, se pulsa "*Finish*" y se reinicia el ordenador.

Configuración del j2sdk

Se abre la "Consola del Sistema", llamada "Símbolo del Sistema". Habitualmente se puede encontrar en:

-"Menú Inicio"→Programas→Accesorios. O sino también se puede abrirla yendo a "Menú Inicio"→Ejecutar y escribir "cmd".

En el símbolo del sistema, se escribe:
"*javac*"

Si se obtiene un mensaje similar al que se muestra en la figura 40, no hace falta seguir con este paso porque ya está configurado.

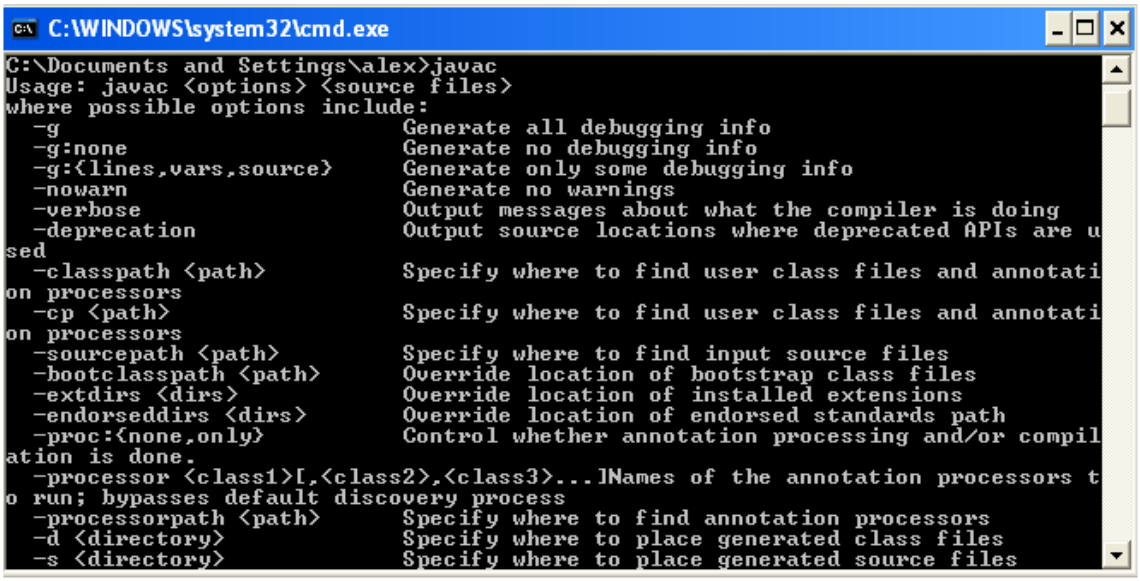


Figura 40.- Consola del sistema

Sin embargo, si se obtiene un mensaje comunicando que no se reconoce el comando escrito, hay que seguir los siguientes pasos.

1 .Se pulsa con el botón derecho del ratón sobre "Mi PC" y se selecciona "Propiedades".

2. Se selecciona la pestaña "Opciones avanzadas" y en la pantalla que se ve en la figura 41 se pulsa en el botón inferior "Variables de entorno".

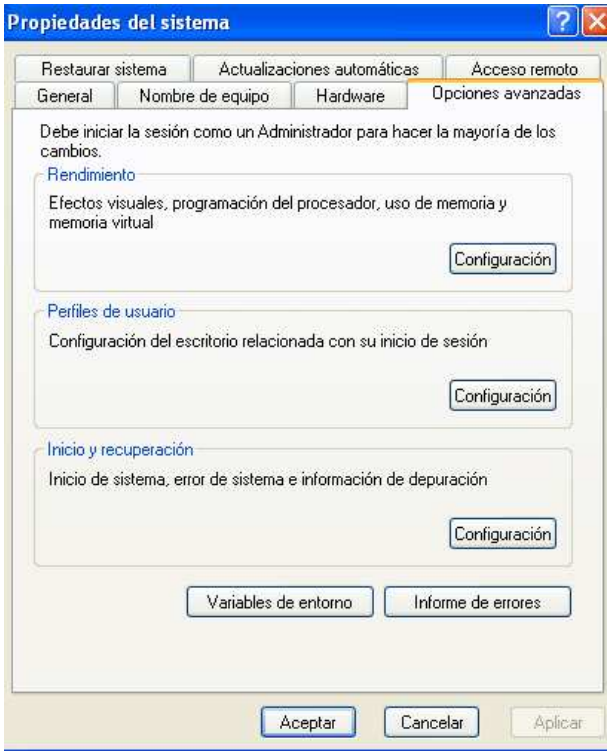


Figura 41.- Pantalla de propiedades del sistema

3. Tanto en el cuadro superior como el inferior de la pantalla que aparece en la figura 42, hay que buscar una variable que se llama "Path", hacer clic sobre ella y pulsar el botón "Modificar" (si aparece en los dos cuadros, hay que hacer lo siguiente dos veces).

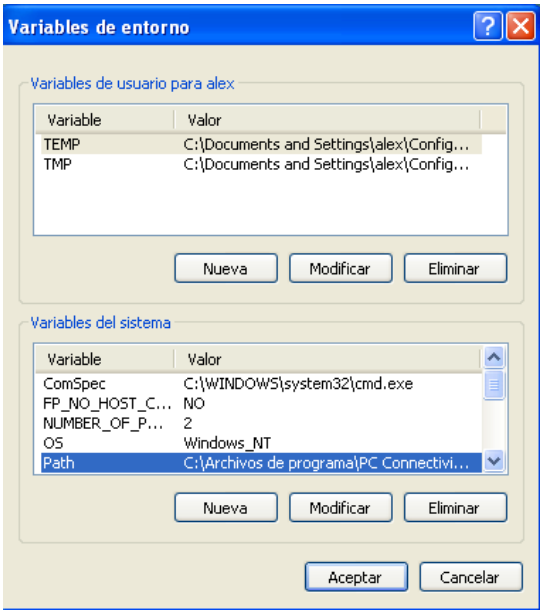


Figura 42.- Pantalla de variables de entorno

4. En el cuadro de valor de variable, figura 43, se añade al final, sin borrar nada:
<lo que hubiera antes>;c:\Archivos de programa\Java\jdk1.6.0_14\bin
Y se pulsa "*Aceptar*".

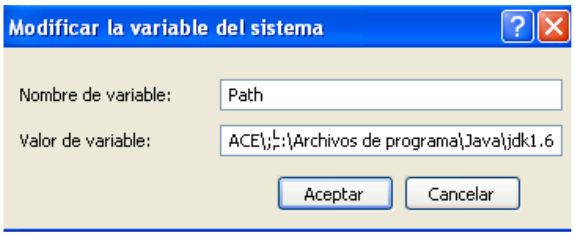


Figura 43.- Pantalla para modificar la variable del sistema

5. Se busca otra vez en la lista de variables si existe "CLASSPATH". Si no existe, no pasa nada, se sigue con el siguiente punto. Si existe, clic sobre ella y se pulsa el botón "Modificar" (si aparece en los dos cuadros, haz lo siguiente dos veces). En el cuadro de valor de variable, se añade al final, sin borrar nada:

<lo que hubiera antes>;.

y se pulsa "Aceptar".

6. Se cierra el cuadro de diálogo de variables de entorno con "*Aceptar*" y otra vez "*Aceptar*" para "Propiedades del sistema".

Si se tenía abierta la consola del sistema, se cierra para que se carguen los nuevos valores. Se abre la consola otra vez y se comprueba que al escribir "*javac"* y "*java"* se obtienen los mensajes habituales de Java. Si sigue apareciendo el mensaje "*iavac*" no se reconoce como un comando interno o externo, programa o archivo por lotes ejecutable; hay que revisar que se han seguido todos los pasos correctamente y se vuelve a repetir el paso 3.

API J2SE
-http://java.sun.com/j2se/1.5.0/docs/api/

JGrasp

Es un entorno de desarrollo, creado para una mejor comprensión de código escrito de lenguajes como Java, ADA, C, C++, etc. jGrasp (Graphical Representations of Algoritms, Structures and Proccesses) ha sido desarrollado por la universidad de Auburn; está escrito en java, por lo que es necesario tener instalada la máquina virtual java, y es de uso libre. El código del programa se puede bajar para diferentes plataformas.

Este programa se ha empleado para desarrollar las clases .java de la aplicación

Instalación

Se descarga desde la página de jGRASP:

-http://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?;dl=download_jgrasp.html

1. Hacer doble clic sobre el archivo de instalación: "*jgrasp1.8.6_15.exe*".

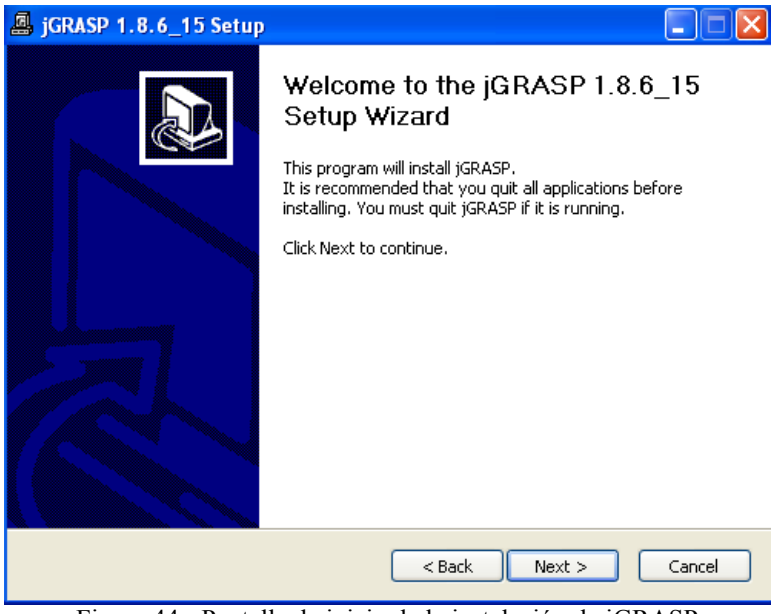


Figura 44.- Pantalla de inicio de la instalación de jGRASP

2. En pantalla de inicio de la instalación. Hay que pulsar sobre el botón "'*Next*>".

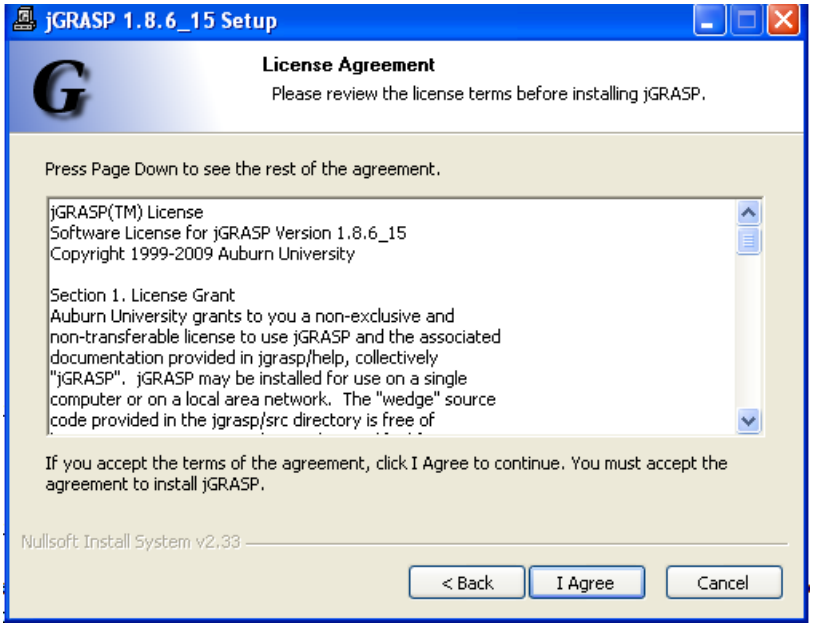


Figura 45.- Pantalla donde aparecen los términos de la licencia

3. Se pulsa sobre el botón "*I agree*" en la pantalla de la figura 45.

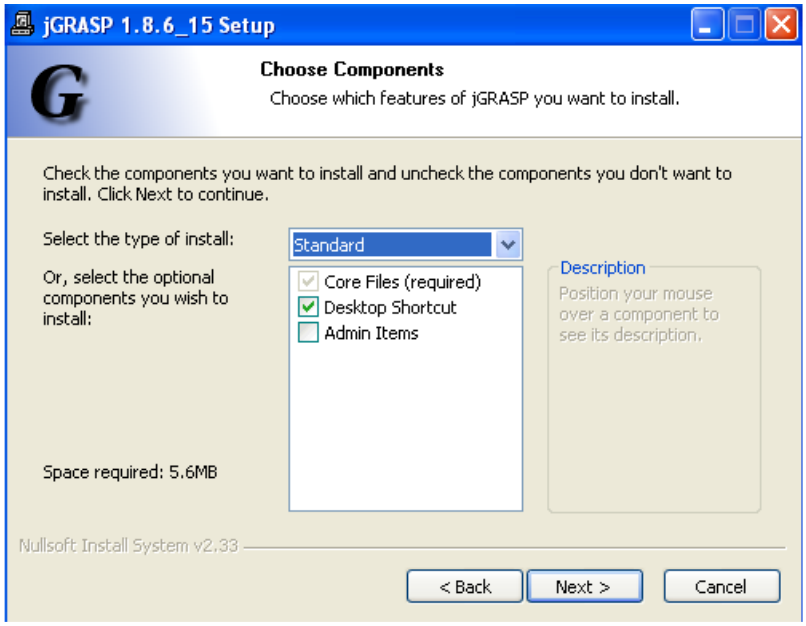


Figura 46.- Pantalla de los componentes

4. Se muestran los componentes a instalar, figura 46, y hay que dejar las opciones que vienen marcadas por defecto y pulsar sobre "*Next* >".

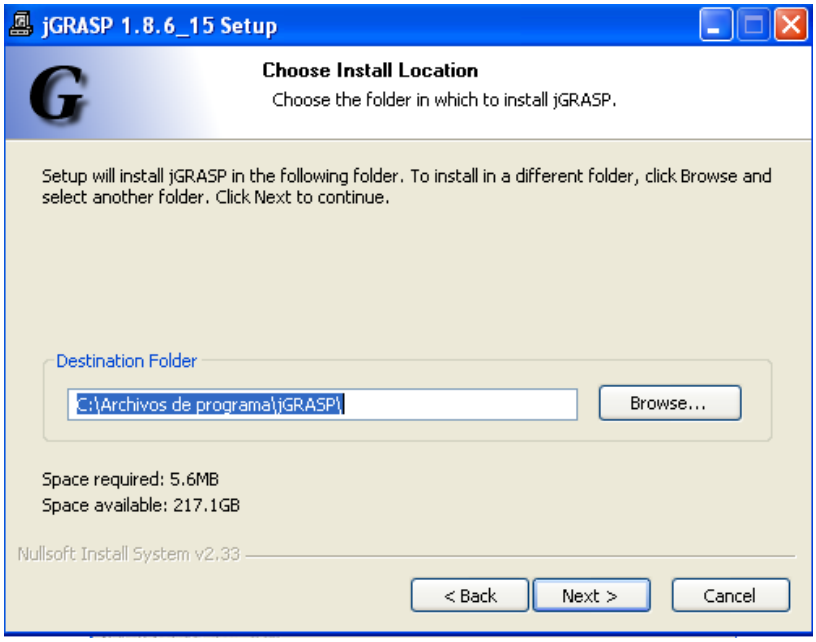


Figura 47.- Pantalla que muestra la carpeta de destino

5. No hay que modificar la carpeta de destino que viene por defecto, figura 47, para evitar problemas en el futuro y se pulsa "*Next*>".
6. Se Pulsa de nuevo sobre "*Next*>".

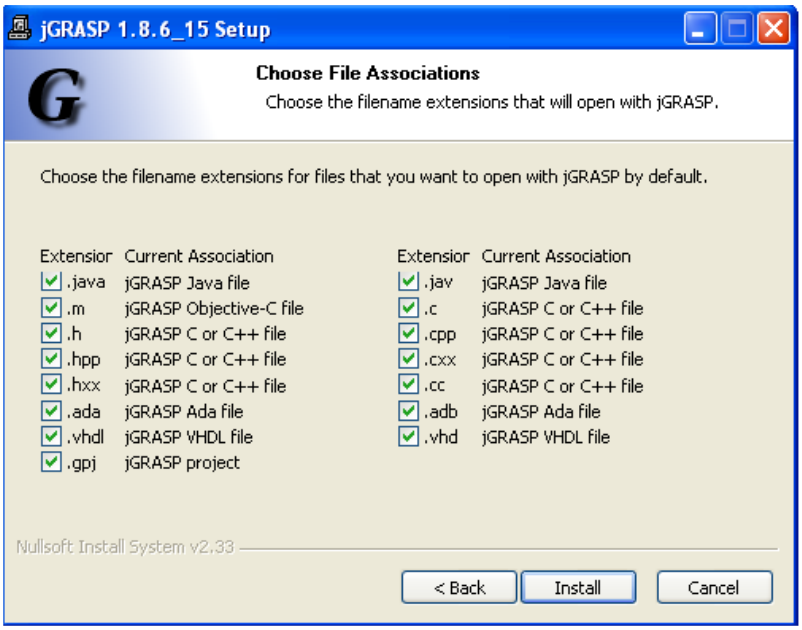


Figura 48.- Pantalla que muestra las posibles opciones para programar en JGRASP

7. Dejar las opciones que vienen marcadas por defecto y pulsar sobre "*Install*". Esperar unos minutos a que finalice la instalación.

8. Finalmente pulsar sobre "*Finish*".

Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC

Se trata de un entorno de desarrollo de MIDP/CLDC/KVM proporcionado por SUN que contiene:

-Desarrollo de MIDlets MIDP 2.0/CLDC 1.1
-Soporta los siguientes APIs opcionales:

- Java technology for the wireless Industry (JTWI) 1.0 (JSR 185)
- Wireless Messaging API (WMA) 2.0 (JSR 205)
- Mobile Media API (MMAPI) 1.1 (JSR 135)
- PDA Optional Packages for the J2ME Platform (JSR 75)
- Java APIs for Bluetooth (JSR 82)
- J2ME Web Services Specification (JSR 172)
- Mobile 3D Graphics API for J2ME (JSR 184)

Instalación:

Se descarga desde la página oficial de Sun:

- *http://java.sun.com/products/sjwtoolkit/download.html*

-Es necesario tener instalado previamente Java 2 SDK versión 1.5 o superior.

1. Se hace doble clic sobre el archivo: "sun_java_wireless_toolkit-2.5.2_01-win.exe"

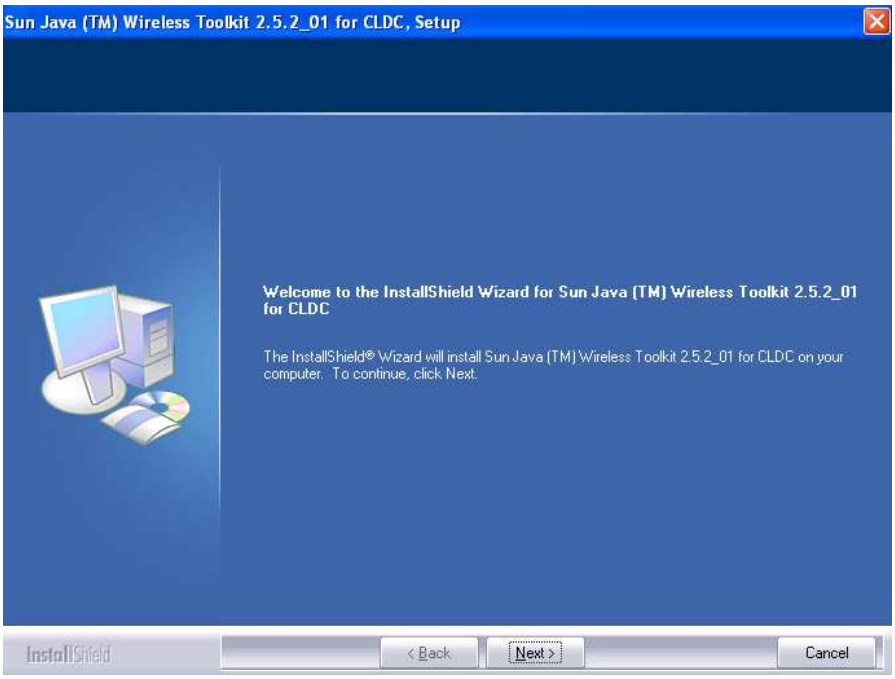


Figura 49.- Pantalla de inicio de la instalación

2. En la pantalla que aparece en la figura 49, se pulsa sobre "*Next* >".

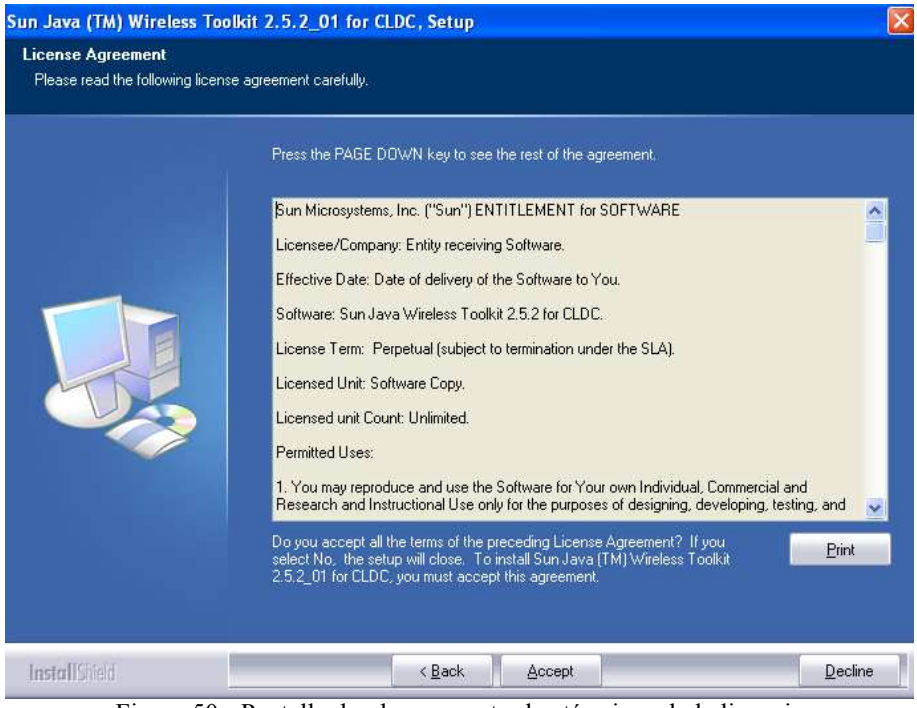


Figura 50.- Pantalla donde se muestra los términos de la licencia

3. En la pantalla de la figura 50, se acepta la licencia pulsando sobre "*Accept*".

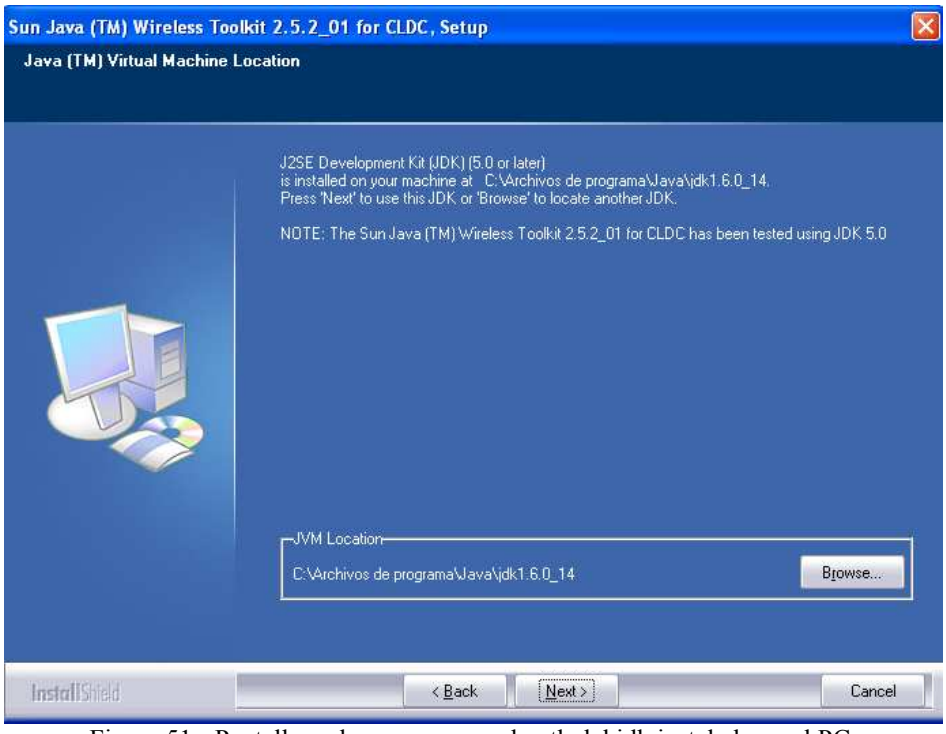


Figura 51.- Pantalla en la que aparece el path del jdk instalado en el PC.

4. Se Comprueba que se ha localizado bien el path en el que se sitúa el jdk, en la figura 51, y se pulsa sobre "*Next* >".

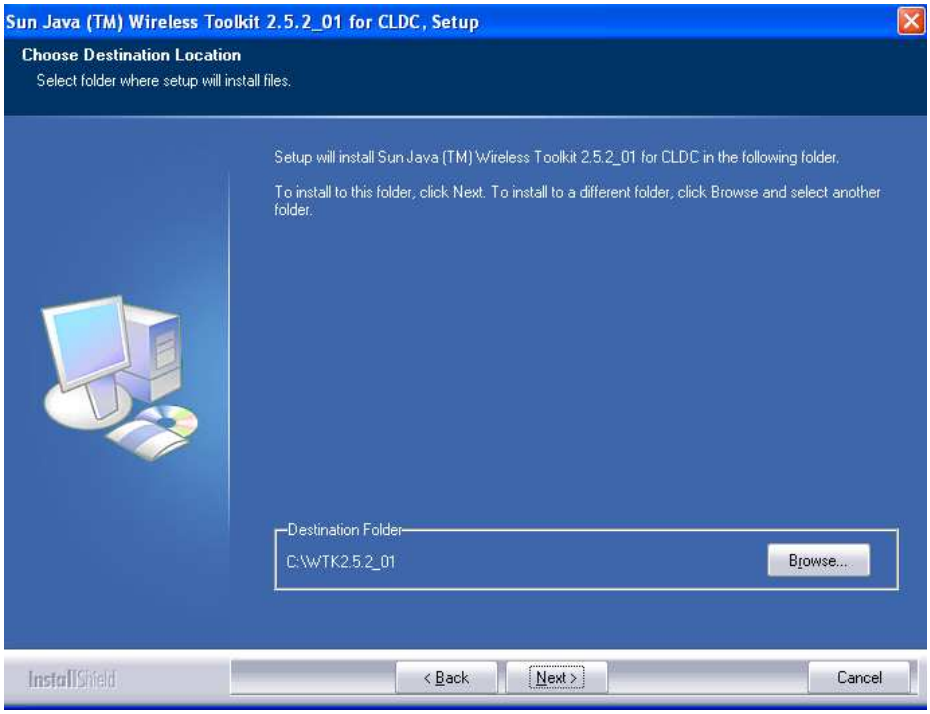


Figura 52.- Pantalla en la que aparece el destino del software en el PC

5. Se deja el path de instalación por defecto, que aparece en la figura 52, para evitar futuros problemas, y se pulsa sobre "*Next* >".

6. Se pulsa de nuevo sobre "*Next* >", en las dos pantallas siguientes.

7. Se esperan unos minutos a que finalice la instalación, y se pulsa sobre "Finish".

Existen versiones para Linux y para Windows. Contienen sus propios simuladores y además pueden integrarse emuladores de terminales reales (por ejemplo: Nokia).

Crear un proyecto:

Un MIDlet Suite es un conjunto de MIDlets contenidas en un mismo .jar. Incluye los ficheros de clases y otros recursos asociados al MIDlet, por ejemplo imágenes. El fichero .jar puede contener varios MIDlets que comparten recursos:

- Un proyecto está asociado a un MIDlet Suite.

- Tras la creación, se crea el directorio C:\WTK25\apps\nombre_proyecto, con subdirectorios:

  - src: ficheros fuente (ficheros .java).
  - res: recursos asociados (p.ej fichero .midi o .jpg).
  - bin: contiene JAR, JAD y manifiesto.
  - lib: librerías externas JAR o ZIP.

El fichero manifiesto (MANIFIEST.MF) está incluido en el .jar y contiene información acerca de los contenidos del fichero jar. El descriptor (.jad) es un fichero de texto con extensión .jad y que permite al AMS comprobar si el MIDlet es adecuado para descargarlo.

Pasos para el desarrollo:

Una vez creado el proyecto, se crean los .java correspondientes y se dejan en el directorio src del proyecto.

- Si es necesario modificar alguna propiedad en el manifiesto o en el .JAD se debe pulsar Settings y realizar las modificaciones.
- Se pulsa el botón Build:

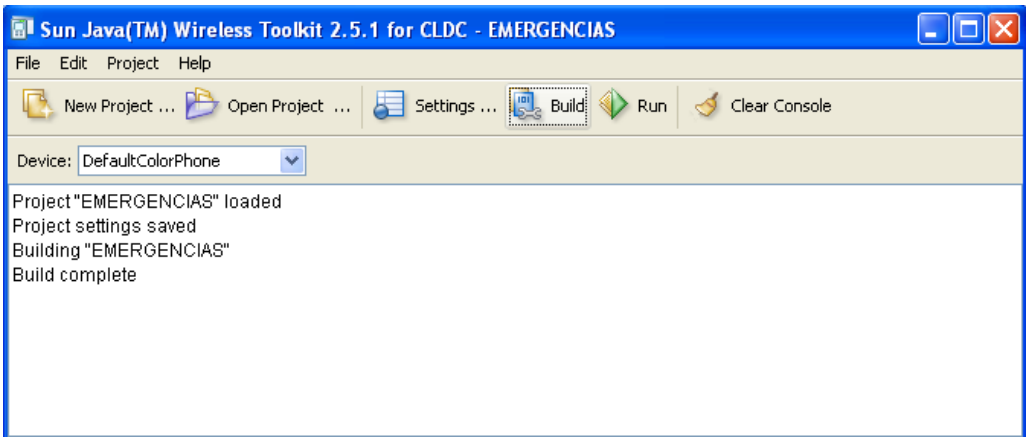


Figura 53.- Consola del Ktoolbar

-Si se producen errores de compilación aparecerán en la consola del ktoolbar.

- Si la compilación se ha realizado con éxito, se puede comprobar el funcionamiento en un emulador:
  -Se selecciona el emulador en Device.
  -Se pulsa el botón Run.

Para copiarse la aplicación a un teléfono móvil se debe ir al menú, Project → Package → Create Package, figura 54, que generará los archivos .jar, .jad y

manifiesto. Y con copiarse el archivo .jar en el móvil, la aplicación queda instalada en el dispositivo.

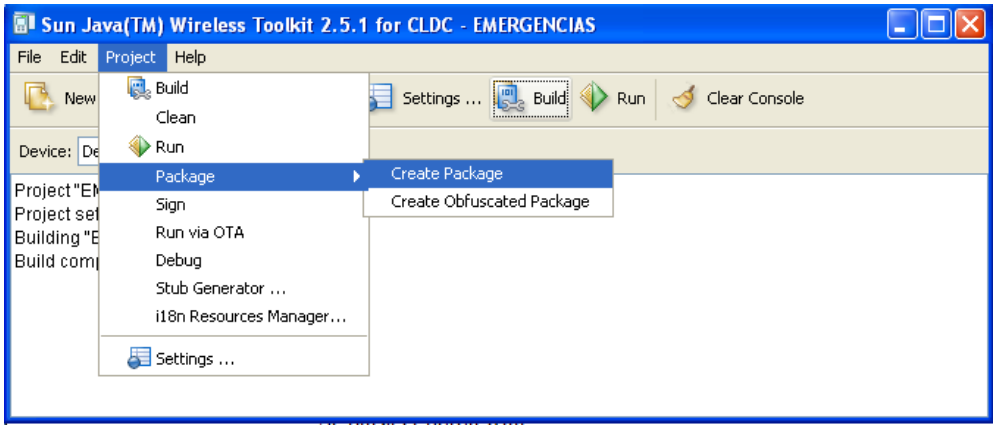


Figura 54.- Pasos para crear el archivo .jar

11. REFERENCIAS

[1] Salvendy, Gavriel, *Handbook of human factors and ergonomics:* John Wiley & Sons, 2006

[2] Nielsen, Jakob, http://www.useit.com/: Jakob Nielsen's Website, 2003

[3] Asociación Interacción Persona-Ordenador, http://www.aipo.es/definicion_aipo.php

[4] Francisco Jesús Martín Fernández, Yusef Hassan Montero, "Conociendo a nuestros usuarios", http://www.nosolousabilidad.com/articulos/conocer_usuarios.htm , 2003

[5] Fausto J. Sainz Salces, Michael Baskett, David Llewellyn-Jones, David England, "Ambient Interfaces for Elderly People at Home", *Ambient Intelligence in Everyday Life*, Springer Berlin / Heidelberg, 2006

[6] Cesar Colado Rodríguez, "Diseño y desarrollo de aplicaciones Web multidispositivo", Germinius XXI, 2003

[7] Antonio Calero Monteagudo, "Tecnologías móviles con Java", Revista del Instituto Tecnológico de Informática

[8] Sergio Gálvez Rojas, Lucas Ortega Díaz, *Java a Tope: Java 2 Micro Edition,* Universidad de Málaga

[9] Shneiderman, Ben, *Diseño de interfaces de usuarios: estrategias para una interacción persona-computadora efectiva,* Pearson Educación, 2005

[10] Mireia Ribera Turró, "Evolución y tendencias en la interacción persona–ordenador", El profesional de la información, 2005

[11] Toni Granollers i Saltiveri, *MPIu+a*, Universidad de Lleida, 2004

[12] International Organization for Standardization (ISO), "http://www.iso.org/iso/iso_catalogue"

[13] The Ergonomics Society**,** http://www.ergonomics.org.uk/

[14] Telefónica, *Las telecomunicaciones y la movilidad en la sociedad de la información,* capítulo 17: Las interfaces Multimodales

[15] W3C World Wide Web Consortium, http://www.w3c.es/

[16] Asociación Española de Terapeutas Ocupacionales. *Guía para desarrollar software accesible para mayores y personas discapacitada.* 2008

[17] Ministerio de Trabajo y Asuntos Sociales. PLAN DE ACCIÓN PARA LAS PERSONAS MAYORES 2003-2007

[18] Miranda de Larra, Rocío. "Los mayores en la sociedad de la información". Cuadernos Fundación Orange. 2007

[19] Instituto Nacional de Estadística. "Uso de productos TIC por características demográficas y tipo de producto". 2008

[20] Jesús Lorés, Toni Granollers, Sergi Lana. *La interacción persona-ordenador*. Capítulo 1: "Introducción". Asociación Interacción Persona Ordenador (AIPO). 2001

[21] Moreno Muñoz, Antonio. Diseño ergonómico de aplicaciones hipermedia. Capítulo 6: "Metodología de diseño". Paidós Papeles de Comunicación. 2000

[22] Moreno Muñoz, Antonio. Diseño ergonómico de aplicaciones hipermedia. Capítulo 4: "Normativa y guías de diseño". Paidós Papeles de Comunicación. 2000

[23] Leah M. Reeves, Jennifer Lai, James A. Larson, Sharon Oviatt, T.S. Balaji, Stéphanie Buisine, Penny Collings, Phil Cohen, Ben Kraal, Jean-Claude Martin, Michael McTear, TV Raman, Kay M. Stanney, Hui Su, y QianYing Wang. "GUIDELINES FOR MULTIMODAL User Interface Design". COMMUNICATIONS OF ACM. Enero 2004.

[24] Toni Granollers i Saltiveri, Jesús Lorés Vidal, José Juan Cañas Delgado. *Modelo de Proceso de la Ingeniería de la usabilidad y de la accesibilidad. MPIu+a.* http://griho.udl.es/mpiua/mpiua/index.htm .Capítulo 5: Modelo de Proceso de la Ingeniería de la Usabilidad y de la Accesibilidad. Noviembre, 2005

[25] Mutua Universal. "El comportamiento humano en situaciones de emergencia". SERVICIOS CONCERTADOS DE PREVENCIÓN Territorial Galicia. 2006

[26] AENOR. UNE 139802. "Aplicaciones informáticas para personas con discapacidad. Requisitos de accesibilidad al ordenador. Software". Septiembre 2003

[27] Nacho Díaz Asenjo. "Programación en MIDP". Departamento de ingenería Telemática. Universidad Carlos III de Madrid. 2007

[28] Jesús Sanchez Allende, Gabriel Huecas Fernández Toribio y otros. *Java 2 Iniciación y referencia*. McGraw-Hill Interamericana. 2005

[29] Héctor Lozano Guadalajara. *Usabilidad y diseño de una interfaz para una aplicación logística*. Proyecto Fin de Carrera de ingeniería industrial. Universidad Carlos III de Madrid. Mayo, 2005.

[30] Jesús Lorés, Toni Granollers, Sergi Lana. *La interacción persona-ordenador*. Capítulo 2: "El Factor Humano". Asociación Interacción Persona Ordenador (AIPO). 2001

[31] Jesús Lorés, Toni Granollers, Sergi Lana. *La interacción persona-ordenador*. Capítulo 5: "Diseño". Asociación Interacción Persona Ordenador (AIPO). 2001

[32] Jesús Lorés, Toni Granollers, Sergi Lana. *La interacción persona-ordenador*. Capítulo 4: "Evaluación". Asociación Interacción Persona Ordenador (AIPO). 2001