ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

Cuaderno virtual del profesor

AUTOR: Sergio García Jiménez
DIRECTOR: Juan Ángel Pastor Franco

Septiembre / 2012

Universidad Politécnica de Cartagena

Propuesta de
Trabajo Fin de Máster

Departamento/ Comisión Académica	TIC
Curso académico	2011/2012
Fecha	31/07/2012
Título del Trabajo: Especificación de requisitos de una aplicación móvil para la gestión de la docencia.	

Titulación	Máster Universitario en Tecnologías de la Información y las Comunicaciones
Intensificación/ Especialidad/ Mención	
Tipo de Trabajo (investigación, estudio técnico, económico,…)	Investigación-Estudio Técnico

Director del trabajo:	Juan Ángel Pastor Franco
Tutor (en su caso)	
Codirector (en su caso):	
Departamento/Empresa/Institución:	*upct*
Alumno	*Sergio García Jiménez*
D.N.I.	23025507T
Fecha de inicio	01-03-2012

Requisitos previos recomendables/exigibles:

El Coordinador Académico (TFPFD)
o Responsable del Centro (TFM)

El Director del Trabajo

Fdo.:

Fdo.: Juan Ángel Pastor Franco

Propuesta de TFM

Objetivos

Se trata de especificar los requisitos de una aplicación que sea un libro de registro del profesor para terminales móviles tipo Android.

Esta aplicación debe permitir al profesorado prescindir de los grandes cuadernos de registro, incomodos y poco manejables, aprovechando las ventajas que proporcionan las nuevas tecnologías móviles.

La especificación deberá ser completos, consistentes, independientes de la implementación (si bien orientados a plataforma Android), precisos y verificables. La estructura del documento deberá facilitar su revisión, modificación y extensión y deberá organizarse por referencias. Se utilizarán convenciones UML siempre que sea posible (p. e.: diagramas de casos de uso y actividad).

Resumen

Según fases de trabajo.

Fases del Trabajo

Análisis de requisitos.
Definición de los datos de la aplicación y sus relaciones. Elaboración de diagrama ER.
Definición delas vistas de acceso a los datos, pantallas y formularios.
Definición de procedimientos de acceso. Casos de uso.
Elaboración de la memoria.

Bibliografía básica

Faulk, S., J. Brackett, P. Ward, and J. Kirby, Jr., The Core Method for Real-Time Requirements,IEEE Software,Vol. 9, No.5, September 1992.

The UML Reference Guide, J Rumbaugh et al, Addison Wesley 1999

Android Developers http://developer.android.com/training/index.html

Wei-Meng Lee, “Beginning Android Application Development”, Wiley Publishing Inc, ISBN 978-1-118-01711-1

COLL, C., MAURI, T. y ONRUBIA, J. (2008). Análisis de los usos reales de las TIC en contextos educativos formales: una aproximación sociocultural. Revista Electrónica de Investigación Educativa, 10 (1). http ://redie.uabc.mx/vol10no1/contenido-coll2.html.

http://c4lpt.co.uk/top-100-tools-for-learning-2011/

A mis dos milagros, Pablo y Sofía
y a la persona que los ha hecho realidad,
mi esposa y compañera, Rosa.

Contenido

1. Introducción

El actual sistema educativo español está enmarcado en la Ley Orgánica 2/2006, de 3 de mayo, de Educación. (LOE). Bajo esta Ley Orgánica surge en 2007 el Decreto 291/2007, de 14 de septiembre, por el que se establece el currículo de la Educación Secundaria Obligatoria en la CARM. Este sistema educativo, así como su currículo para Murcia, incentiva una evaluación individualizada y una atención a la diversidad del alumnado por parte del profesorado. Además, se promueve el uso de las TICs en las clases de manera que éstas sean más eficaces y eficientes en su contribución al proceso de enseñanza-aprendizaje **[Coll 08] [Herr 11]**.

Por otra parte, en el mundo actual, nos encontramos con un escenario tecnológico en el que la estrella es la movilidad y la comodidad. El teléfono móvil sustituye al fijo, los ordenadores de sobremesa son remplazados por ligeros portátiles, e incluso estos últimos empiezan a perder su puesto a favor de dispositivos de tipo "teléfono inteligente" (smartphone), que encierran, en la palma de nuestra mano, una capacidad de procesamiento que hace apenas 10 años no podíamos sino soñar. El negocio de las aplicaciones para móviles está en pleno auge, y los juegos y contenidos multimedia que se pueden disfrutar en este tipo de dispositivos no tienen nada que envidiar a los que ofrecen plataformas más clásicas.

Basándonos en estas premisas, surge la idea de la elaboración de una herramienta tecnológica adaptada a nuestros tiempos, que sirva de ayuda en la labor de los docentes. Se trata de un libro de registro del profesor en forma de aplicación para terminales móviles tipo Android. Esta aplicación permitirá al profesorado prescindir de los grandes cuadernos de registro del profesor, incomodos y poco

manejables, ahorrar el gasto de papel que acarrea y aprovecharse de las ventajas que puedan proporcionar las nuevas tecnologías, además de poder llevar el registro de todos los cursos en el bolsillo del pantalón.

En este documento nos centraremos en definir las especificaciones del software que queremos desarrollar, de modo que el programador tenga muy claro *qué construir*. Parafraseando al Dr. Brooks:

"*La parte única más difícil de construir un sistema de software es decidir con precisión qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requisitos técnicos detallados. Ninguna otra parte del trabajo hace ineficiente el sistema resultante si se hace mal. Ninguna otra parte es tan difícil de corregir más adelante.*" **[Brooks 87]**.

2. Objetivos y organización del documento

El objetivo de este trabajo es definir los requisitos de una aplicación que sirva de apoyo en la labor del registro de todos los datos importantes que necesita un docente en su labor diaria, el ***libro del profesor***. Más concretamente: Se trata de especificar los requisitos de una aplicación que sea un libro de registro del profesor para terminales móviles tipo Android. Como se ha dicho, esta aplicación debe permitir al profesorado prescindir de los grandes cuadernos de registro, incómodos y poco manejables, aprovechando las ventajas que proporcionan las nuevas tecnologías móviles.

La especificación deberá ser completa, consistente, independiente de la implementación (si bien orientada a la plataforma Android), precisa y verificable. La estructura del documento deberá facilitar su revisión, modificación y extensión y deberá organizarse por referencias **[Faulk 92]**. Se utilizarán convenciones UML (Lenguaje Unificado de Modelado) **[Booch 99]** siempre que sea posible (p. e.: diagramas de casos de uso y actividad).

Para realizar este trabajo, se realizará:

- Análisis de requisitos.
- Definición de los datos de la aplicación y sus relaciones. Elaboración de diagrama E-R.
- Definición de las vistas de acceso a los datos, pantallas y formularios.
- Definición de procedimientos de acceso. Casos de uso.

Aunque estos puntos se desarrollarán en esta memoria, los principales servicios que debe ofrecer la mencionada aplicación son los siguientes:

- Consultar horario.
- Consultar la lista de alumnos por clase.
- Consultar el perfil de cada alumno, por medio de una ficha individualizada en la que se incluye una foto del mismo.
- Definir los instrumentos de evaluación calificables, con su peso sobre la nota final, y los no calificables.
- Definir las fechas de inicio y final de cada evaluación.
- Definir las clases asignadas ese curso.
- Definir las aulas en las que se impartirán cada una de las clases.
- Introducción de las listas de alumnos de manera sencilla, bien por archivo .doc, .pdf suministrado por la secretaría del centro, bien por medio de la sincronización con la web de la Consejería de Educación, bien por voz…
- Mostrar tanto los resultados de las calificaciones finales por evaluación, como al final del curso.
- Mostrar gráficas de los resultados académicos de los alumnos, organizados franjas (de 0 a 4, de 5 a 6,…).
- Controlar las faltas del alumnado.
- Sincronizar la lista de control personal de faltas con la web de la Consejería de Educación.
- Conexión y sincronización con el web-blog personal del profesor.

La aplicación deberá crearse usando lenguaje Java, para funcionar sobre el sistema operativo Android y hacer uso de sus capacidades para integrar diferentes API y servicios ya existentes como por ejemplo el navegador web, la cámara fotográfica del terminal, las redes móviles, etc.

Como ya se ha indicado, se tratará en todo momento de cumplir con una buena especificación de requisitos software (System Requirements Specification o **SRS**), que cumpla que el software de la aplicación sea:

- Completo
- Implementación independiente
- Claro y coherente
- Preciso
- Verificable
- Modificable
- Legible
- Revisable y organizado en referencias

Por último, la aplicación contará con un diseño atractivo, modificable e intuitivo, al alcance de todos los usuarios, no debiendo éstos ser expertos en informática para el uso de la misma.

Este documento se organiza en 12 capítulos. Además de los dos ya vistos, se incluye un estado del arte, desarrollado en los capítulos 3, 4 y 5, en los que se explica y define qué es una especificación de requisitos de software, cuál es el escenario de las aplicaciones móviles y en qué consiste y cuál es la historia del sistema operativo Android.

Se continua en el capítulo 6 exponiendo y explicando la información que manejará el programa, además de sus relaciones por medio de un diagrama Entidad-Relación **(E-R)**.

Las pantallas con las que contará el programa para relacionarse con el usuario se enumerarán y explicarán en el capítulo 7.

El capítulo 8, estará ocupado por los requisitos del software y los capítulos 9 y 10 se encargarán del mapa de casos de uso y de los casos de uso respectivamente.

Por último, llegarán las conclusiones en el capítulo 11 para terminar con la bibliografía en el capitulo 12.

3. Especificación de requisitos de software

La experiencia indica, que para los desarrolladores, el problema más grande de ingeniería del software, es la definición de sus requisitos, cuyos objetivos son, decidir qué construir y qué resultados documentar. Si bien hay un desacuerdo considerable sobre cómo resolver el problema, muy pocos estarían en desacuerdo con la afirmación del Dr. Brooks sobre que ninguna otra parte de desarrollo resulta tan difícil de hacer bien y sobre sus desastrosas consecuencias cuando se hace mal.

3.1. ¿Por qué definir las especificaciones de requisitos software?

En el desarrollo de sistemas, las especificaciones de requisitos de software pueden producir una gran variedad de roles. Por ejemplo, los requisitos por lo general, documentan lo que el cliente espera que se le entregue y pueden proporcionar las bases contractuales para su desarrollo. Para los administradores pueden servir de base para la programación. Para los ingenieros de sistemas, define el papel del software dentro del sistema general y su relación con los componentes de hardware. Para los diseñadores de software, especifica las restricciones sobre el comportamiento y el rendimiento. Para los programadores define el rango de comportamiento aceptable de un sistema y es la autoridad final en los productos obtenidos. Para los evaluadores, es la base para la planificación de pruebas y verificación. En algunos sistemas los requisitos también son utilizados por grupos tan diversos como el marketing y los reguladores gubernamentales.

La figura 1 muestra los métodos para el desarrollo del software y para el desarrollo de sistemas **[Faulk 95]**.

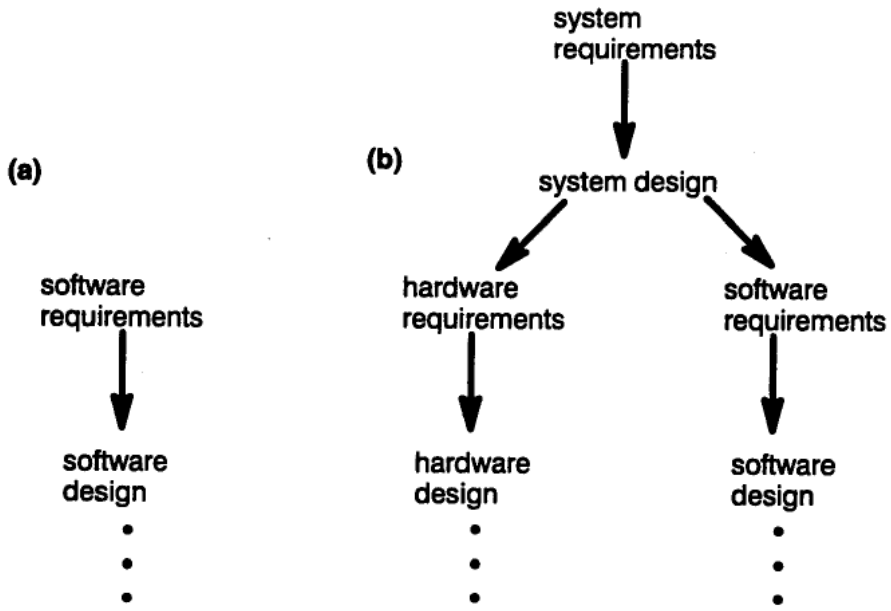

FIG. 1 – METODOS DE DESARROLLO DE (A) SOFTWARE Y (B) SISTEMAS

Es una práctica común clasificar los requisitos de software como "funcionales" o "no funcionales". Aunque las definiciones varían, "funcional" se refiere a los requisitos que limitan el comportamiento visible del sistema, específicamente a los valores deseados de las salidas para entradas dadas. "No funcional" típicamente se refiere a todas las demás restricciones, incluyendo pero no limitado a, rendimiento, fiabilidad, mantenimiento, reusabilidad, y los requisitos de seguridad.

Ni la terminología ni la clasificación en funcionales y no funcionales es eficaz. La palabra "función" es uno de los términos más usados en exceso en ciencias de la computación y su único sentido riguroso, es el de una función matemática, no como se entiende aquí. La clasificación no es muy útil porque se diferencian clases de requisitos con cualidades muy similares, mientras que se agrupan otros que no tienen nada en común.

Una distinción más útil es entre lo que puede ser descrito como "*requisitos de comportamiento*" y "*atributos de desarrollo de calidad*" con las siguientes definiciones:

- ***Requisitos de comportamiento:*** definen el comportamiento permitido y observable del sistema en función de las salidas esperadas para unas entradas y el estado actual del sistema. El concepto de estado incluye el tiempo, el estado del software y hardware. Los requisitos de comportamiento son el estado de seguridad, exactitud, rendimiento y tolerancia a fallos.

- ***Atributos de desarrollo de calidad:*** limitan las cualidades de la construcción estática del sistema. Estos incluyen propiedades como la capacidad de prueba, el mantenimiento, la facilidad de modificación y reusabilidad y su representación estática. Las evaluaciones de los atributos de calidad son relativistas en el sentido de que no hay una escala común o absoluta con la que se puedan medir sus logros.

Los requisitos de comportamiento tienen en común que son las propiedades del comportamiento en tiempo de ejecución del sistema. Los requisitos de comportamiento se pueden definir en términos de entradas, salidas, y de estado, se pueden validar objetivamente por la modificación de esas propiedades y observar el comportamiento del sistema.

En otras palabras, el objetivo de los requisitos es el de comprender y especificar el problema a resolver en lugar de dar la solución.

3.2. ¿Por qué es difícil definir los requisitos?

En general, se está de acuerdo en que el objetivo del SRS es definir el ***qué*** hacer y no el ***cómo*** hacerlo.

Por desgracia, distinguir el "***qué***" del "***cómo***" representa un dilema. Como Davis [Davis 88], entre otros, señala, la distinción entre el ***qué*** y el ***cómo*** es necesariamente función de la perspectiva. Una especificación en un sistema en cualquier nivel de abstracción puede ser vista como una descripción "qué" para el siguiente nivel. Por lo tanto las necesidades del cliente definen el "qué" y la descomposición del sistema en el hardware y el software correspondiente definen el "cómo". La descomposición del sistema define el "qué" del comportamiento de un componente de software, y su diseño el "cómo", y así sucesivamente. El resultado es que los requisitos no pueden ser discutidos con eficacia sin un acuerdo previo sobre el sistema y el nivel de abstracción del que se está hablando. Hay que ponerse de acuerdo sobre lo que constituye el espacio del problema y lo que constituye el espacio de la solución - el análisis y la especificación de los requisitos por tanto, pertenecen propiamente al espacio del problema.

Por otro lado, teniendo en cuenta el contexto del desarrollo multi-persona, multi-versión, nuestro objetivo básico al especificar lo que debe hacer el software se puede descomponer en los siguientes objetivos parciales:

1. Entender con precisión lo que se requiere del software.

2. Asegurarse de que se entienden los requisitos por todas las partes involucradas en el desarrollo.

3. Proporcionar un medio para controlar la producción para asegurar que el sistema final satisfaga los requisitos (incluyendo la administración de los efectos de los cambios).

De ello se desprende que la fuente de la mayoría de los errores de los requisitos, se deben al fracaso para llevar a cabo adecuadamente uno de estos objetivos, es decir:

1. Los desarrolladores no alcanza a comprender lo que se requiere del software por el cliente, usuario final, o de otras partes con interés en el producto final.

2. Los desarrolladores no capturaron completa y precisamente los requisitos o no comunicaron posteriormente los requisitos de forma efectiva a otras partes involucradas en el desarrollo.

3. Los desarrolladores no gestionaron con eficacia los efectos en el cambio de las necesidades o aseguraron la conformidad con las medidas de desarrollo en niveles inferiores, incluyendo el diseño, código, integración, prueba, o el mantenimiento de los requisitos del sistema.

3.3. Dificultades esenciales

Incluso teniendo muy claro el sistema final aparecen, a veces, una serie de problemas que hacen fallar la especificación de requisitos. El porqué de estos problemas se debe a diferencias de entendimiento sobre los objetivos que se pretenden conseguir. Para entender esto se hace imprescindible la distinción entre dos tipos de dificultades:

1. ***Dificultades esenciales al problema:*** aquellas inherentes al problema.

2. ***Dificultades accidentales:*** las derivadas de la práctica imperfecta. Aunque los requerimientos sean difíciles, a veces, esas dificultades, se ven multiplicadas por las deficiencias de la práctica.

Veamos las dificultades esenciales más comunes:

- ***Comprensión.*** Los clientes no saben con precisión lo que quieren o necesitan. Tienen una idea general del sistema, pero no cuentan con una comprensión precisa y detallada de qué funciones pertenecen al software, qué salida debe ser para cada posible entrada, la duración que debe tomar cada operación, cómo afecta una decisión a otra, y así sucesivamente. De hecho, a menos que el nuevo sistema sea simplemente una reconstrucción de uno antiguo, muchas de las decisiones sobre el comportamiento del sistema dependerán de las decisiones que otros todavía no han tomado y las expectativas

cambiarán a medida que el problema se entienda mejor. Sin embargo, es esta comprensión precisa y con riqueza de detalles del comportamiento esperado la que se necesita para crear diseños eficaces y desarrollar código correcto.

- ***Comunicación.*** Los requisitos de software son difíciles de transmitir con eficacia. Las estructuras conceptuales de sistemas de software son complejas, arbitrarias y difíciles de visualizar **[Brooks 87]**. Para empeorar las cosas, la mayoría de las estructuras conceptuales en el software no tienen análogo físico fácilmente comprensible por lo que son difíciles de visualizar.

  En la práctica, la comprensión sufre bajo todas estas restricciones. Trabajamos mejor con las estructuras regulares y predecibles, podemos comprender sólo una cantidad limitada de información de una sola vez, y entender mejor grandes cantidades información cuando las podemos visualizar. Así, la tarea de capturar y transmitir los requisitos de software es de por sí difícil.

  La dificultad inherente a la comunicación se ve agravada por la diversidad de propósitos y audiencias para una especificación de requisitos. Lo ideal es una especificación técnica que está escrita para una audiencia determinada.

- ***Control.*** El carácter arbitrario e invisible del software, hace que sea difícil anticiparse a qué requisitos serán superados fácilmente y cuáles no, lo cual complica mucho la planificación de su desarrollo.

  Esta situación se ve agravada por la maleabilidad inherente del software. De todos los problemas que acosan a los administradores de software, los más graves son los cambios frecuentes y arbitrarios de los requisitos. Para la mayoría de los sistemas, estos cambios continúan, incluso después de la entrega. Los continuos cambios hacen que sea difícil el desarrollo de especificaciones estables, la planificación efectiva, y el control de costes y calendario.

- ***Dependencias mutuas.*** En la búsqueda de soluciones a los problemas expresados anteriormente, nos encontramos con la dificultad adicional de que tales problemas no se pueden separar fácilmente y ser tratados poco a poco. Por ejemplo, los desarrolladores tratan de abordar el problema del cambio de requisitos por medio de la congelación de los mismos antes de que comience el diseño. Esto no siempre es factible debido al problema de la comprensión - el cliente no sabe completamente lo que quiere hasta que lo ve. Una forma de tratar con diferentes receptores es escribir una especificación para cada uno de ellos. Así, puede haber una especificación del sistema, una serie de requisitos entregados al cliente, un conjunto distinto de requisitos

técnicos escritos para el consumo interno de los desarrolladores de software, y así sucesivamente. Sin embargo, esta solución aumenta enormemente la complejidad, ofrece una vía abierta para las inconsistencias, y multiplica las dificultades de la administración de cambios.

Estos temas representan sólo una muestra de las relaciones inherentes entre las diferentes facetas del problema de requisitos. Los diferentes destinatarios de la especificación, tienen diferentes intereses en los requisitos de un sistema, que pueden resultar conflictivos entre sí para alcanzar una posible solución.

Las consecuencias son dobles. En primer lugar, nos vemos limitados en la aplicación de la estrategia más eficaz para hacer frente a problemas complejos: "*divide y vencerás*". Si un problema se considera de forma aislada, la solución es susceptible de agravar las dificultades de otros. Las soluciones efectivas para la mayoría de las dificultades de requisitos deben abordar más de un problema al mismo tiempo. En segundo lugar, el desarrollo de soluciones prácticas requiere hacer concesiones difíciles. Cuando los problemas entran en conflicto, se deben tomar compromisos. Debido a que el resultado de las concesiones no beneficia de igual manera a todas las partes, se deben establecer compromisos eficaces de negociación.

3.4. El papel de un enfoque disciplinado

Siguiendo el modelo integrado de Davis y de su terminología **[Davis 93]**, la fase de requisitos consta de dos actividades conceptualmente distintas pero superpuestas:

1. ***Análisis del problema:*** El objetivo del análisis es entender con precisión el problema que debe ser solucionado. Se incluye la identificación de la finalidad exacta del sistema, quién lo usará, las limitaciones de las soluciones aceptables, y los compromisos posibles en caso de conflicto.

   Las cuestiones básicas en el análisis de problemas son las siguientes:

   - ¿Cómo obtener un conjunto completo de requisitos del cliente o de otras fuentes?
   - ¿Cómo descomponer el problema en partes manejables intelectualmente?
   - ¿Cómo organizar la información para que pueda ser comprendida?
   - ¿Cómo hacer entender el problema a todas las partes involucradas?
   - ¿Cómo resolver las necesidades en conflicto?

- ¿Cómo saber cuándo parar?

2. ***Especificación de requisitos:*** El objetivo de la especificación de los requisitos es crear un documento, la Especificación de Requisitos de Software (SRS), que describe exactamente lo que se va a construir. El SRS captura los resultados de análisis de problemas y caracteriza el conjunto de soluciones aceptables para el problema.

Resulta poco beneficioso para el desarrollo, una comprensión profunda del problema, si ese entendimiento no se comunica eficazmente a los clientes, diseñadores, implementadores, evaluadores y técnicos de mantenimiento. Cuanto mayor y más complejo es el sistema, más importante resulta una buena especificación. Este es un resultado directo de las numerosas funciones que el SRS juega en el desarrollo multi-persona, multi-versión **[Pamas 86]**:

1. El SRS es el principal vehículo para el acuerdo entre el desarrollador y el cliente de lo que exactamente se va a construir. A menudo es la base para juzgar el cumplimiento de las obligaciones contractuales.

2. El SRS registra los resultados del análisis de los problemas a resolver. Constituye la base para determinar si los requisitos están completos o si el análisis adicional es necesario.

3. El SRS define qué propiedades debe tener el sistema y las limitaciones en su diseño e implementación. Se define el lugar donde hay, y donde no, libertad de diseño. Esto ayuda a asegurar que las decisiones sobre el sistema se tomen de forma explícita durante la fase de requisitos, y no implícitamente durante la programación.

4. El SRS es la base para estimar el costo y los plazos. Es la herramienta principal de gestión para el seguimiento del progreso de desarrollo y para determinar el avance efectivo de un proyecto.

5. El SRS es la herramienta principal del verificador para determinar si el comportamiento del software es aceptable.

6. El SRS ofrece la definición del comportamiento estándar esperado para los responsables del sistema y se utiliza para registrar los cambios de ingeniería.

3.5. Requisitos para la especificación de requisitos del software

Los objetivos del proceso de requisitos, las dificultades que conlleva, y el papel de la especificación de requisitos en un proceso disciplinado determinan las propiedades de una "buena" especificación de requisitos. Estas propiedades no estipulan ningún método en particular pero describen las características que un método eficaz debe abordar.

Al hablar de las propiedades de un buen SRS, es útil distinguir las *propiedades semánticas* de las *propiedades de empaquetamiento* **[Faulk 92]**. Las propiedades semánticas surgen de lo que dice la especificación (es decir, su significado o semántica). Las propiedades de empaquetamiento son el resultado de cómo están escritos los requisitos (el formato, organización y presentación de la información). Las propiedades semánticas determinan la precisión con la que un SRS captura los requisitos de software. Las propiedades empaquetamiento determinan la facilidad de uso y la legibilidad de la especificación resultante. A continuación se ilustra la clasificación de las propiedades de un buen SRS:

Propiedades semánticas del SRS	Propiedades de embalaje del SRS
Completo	Modificable
Implementación independiente	Legible
Claro y coherente	Revisable y organizado por referencias
Preciso	
Verificable	

Un SRS satisface las prop. semánticas de una buena especificación si es:

- ***Completo.*** El SRS define el conjunto de implementaciones aceptables. Debe contener toda la información necesaria para escribir software que sea aceptable para el cliente y no más. Cualquier aplicación que satisface todas las declaraciones de los requisitos es un producto aceptable. Cuando la información no está disponible antes de empezar el desarrollo, las áreas incompletas deben indicarse explícitamente **[Pamas 86]**.

- ***Independiente de la implementación.*** El SRS debería estar libre de decisiones de diseño e implementación a menos que esas decisiones reflejen las necesidades reales.

- ***Claro y coherente.*** Si el SRS está sujeto a una interpretación conflictiva, las distintas partes no estarán de acuerdo en lo que se va a construir o en si se ha construido el software adecuado. Todos los requisitos deben tener una sola interpretación posible.

- ***Preciso.*** El SRS debe definir con exactitud el comportamiento requerido. Se debe definir el valor de salida esperado para cada una de las posibles entradas.

- ***Verificable.*** Un requisito es verificable si es posible determinar sin ambigüedades si una aplicación dada lo satisface. Por ejemplo, un requisito funcional es comprobable si es posible determinar, para cualquier caso de prueba (es decir, una salida y una entrada), si la salida representa un comportamiento aceptable del software dada la entrada y el estado del sistema.

Un SRS que satisface las propiedades de empaquetamiento de una buena especificación es:

- ***Modificable.*** El SRS debe estar organizado para facilitar el cambio. La especificación se organiza para limitar el efecto de los posibles cambios en los requisitos.

- ***Legible.*** El SRS debe ser comprensible para las partes que lo utilizan. Evidentemente, debe relacionar los elementos del espacio del problema tal como se entiende por el cliente a la conducta observable (en relación a los requisitos de comportamiento) del software.

- ***Revisable y organizado en referencias.*** El SRS es la especificación técnica principal de los requisitos de software. Es el depositario de todas las decisiones tomadas durante el análisis sobre lo que debe ser construido. Es el documento revisado por el cliente o sus representantes. Es el árbitro principal de las controversias. Como tal, el documento debe ser organizado para una referencia rápida y fácil. Debe de ser clara en lo que a cada decisión sobre los requisitos se refiere. Debe de ser posible responder a preguntas específicas sobre los requisitos de forma rápida y sencilla.

Aunque el SRS ideal puede ser inalcanzable, sigue siendo importante tener un ideal **[Pamas 86]**. Tal ideal:

- Proporciona una base para la estandarización de los procesos de una organización y de los productos.

- Provee una norma con la que pueden medirse los progresos.

- Proporciona una guía, que ayuda a los desarrolladores a entender lo que hay que hacer en cada fase del proyecto y a determinar cuándo se ha conseguido un producto aceptable.

4. Aplicaciones móviles

4.1. Introducción

El mundo de las aplicaciones móviles está viviendo hoy en día, su momento más dulce. Puede que sea por el avance tecnológico de los dispositivos móviles, especialmente los *smartphones*. Este avance ha conseguido equiparar la capacidad gráfica y de procesamiento de los smartphones a la de los ordenadores que podemos encontrar hoy día en muchos hogares. También ha permitido que tecnologías como GPS y WiFi, se integren "de serie" en casi cualquier teléfono móvil, multiplicando sus posibilidades de uso.

Tal vez este auge se deba a las redes sociales, la necesidad de información inmediata, del qué estás haciendo y del qué está pasando; en fin, de la cultura, del "aquí y ahora". Noticias, eventos y datos parecen destinados a ser enviados y consumidos desde dispositivos móviles porque, si esperas a llegar a casa, habrán desaparecido enterrados bajo miles de cosas más recientes. Lo más probable es que sea una combinación de estos factores y muchos otros.

En cualquier caso, nos encontramos actualmente con un mercado, el de las aplicaciones móviles, que ha ido creciendo y evolucionando de forma pareja a como lo ha hecho el de los dispositivos móviles.

FIG. 2 – APPLE POWERBOOK G4 DE 2005 (IZQUIERDA) TIENE LA MISMA VELOCIDAD DE PROCESADOR, CANTIDAD DE MEMORIA RAM Y ESPACIO DE ALMACENAMIENTO QUE UN TELÉFONO MÓVIL SAMSUNG GALAXY S2 DE 2011 (DERECHA).

El mercado de las aplicaciones móviles ha pasado de ser casi inexistente, a estar orientado puramente al entretenimiento, hasta encontrar, hoy día, nichos de mercado en prácticamente todos los aspectos de nuestra vida: lectores de documentos para la oficina, aplicaciones para crear música, lectores de libros electrónicos, navegación mediante GPS... Aplicaciones todas ellas, que aprovechan las velocidades de transmisión de las redes actuales, la posibilidad de conocer la localización del usuario, la capacidad de proceso de los nuevos modelos y la ubicuidad del acceso a Internet y toda la información que alberga, para ofrecer al usuario la que necesita en ese momento.

4.2. Definición y orígenes

Entendemos como aplicación móvil, cualquier aplicación *software* creada por terceros y destinada a su instalación y ejecución en un dispositivo móvil, esto es de tamaño reducido e ideado para ser utilizado de manera inalámbrica.

Cuando a mediados de la década de 1990 aparecieron los primeros dispositivos móviles que se comercializaron con éxito, las famosas PDAs (del inglés *Personal Digital Assistant*), fig. 3, el mercado de las aplicaciones móviles era muy discreto. Estos dispositivos —destinados casi siempre a profesionales en continuo movimiento, como ejecutivos y comerciales, o relacionados con el mundo de la tecnología— estaban muy limitados tecnológicamente, y de hecho los primeros modelos requerían de una "estación base" conectada a otro equipo para poder transferir y almacenar la información en dicho equipo, dado que las PDAs no tenían memoria de almacenamiento propia.

FIG. 3 – PDA, MODELO PALM III DE 1999.

Además, las aplicaciones que el fabricante incluía por defecto: calendario, agenda de contactos, pequeños editores de texto y, más adelante, clientes de correo y navegadores web, eran más que suficientes para la utilización que se les daba. Esto suponía que las aplicaciones móviles existentes fueran desarrolladas casi siempre a petición de alguna empresa privada, para su utilización propia o, como mucho, ampliaciones y mejoras de las aplicaciones ya instaladas, dado que la tecnología del dispositivo no daba para más y, por tanto, no merecía la pena desarrollar aplicaciones adicionales. Aunque la evolución posterior propició que comenzarán a aparecer aplicaciones desarrolladas por terceros, el mercado ya había sido conquistado por los teléfonos móviles.

4.3. Telefonía móvil

4.3.1. J2ME y WAP

Lo que no sucedió con las PDA, sucedió con los teléfonos móviles: en la segunda mitad de los 90, la introducción de las tecnologías GSM y EDGE con su sistema de mensajes cortos SMS, el protocolo WAP, el abaratamiento de costes y la liberalización del mercado de operadores se unieron para provocar un *boom* en la utilización de telefonía móvil, pasando sólo en España de 1 millón de usuarios en 1996 a casi 30 millones en 2001.

Aunque los primeros terminales de 2ª generación (GSM) se limitaban a ofrecer la funcionalidad básica de comunicación por voz, envío de mensajes SMS y listín telefónico —con pantallas monocromas de dos o tres líneas de texto y, en muchos casos, ni siquiera letras minúsculas. Había algunas excepciones como el Nokia 9000 Communicator que, con sus aplicaciones de correo y navegación web, sólo estaba al alcance de los bolsillos más pudientes. A finales de los 90 empezaron a aparecer los primeros móviles destinados al público general que incorporaban aplicaciones sencillas de tipo calendario, reloj y agenda, e incluso juegos (fig. 4). Sin embargo, en todos los casos eran aplicaciones propietarias incluidas por los propios fabricantes; los terminales de entonces no

tenían ningún mecanismo sencillo que permitiera la instalación de aplicaciones de terceros.

FIG. 4 – DE IZQ. A DER.: ALCATEL "ONE TOUCH EASY" (1997); NOKIA "5110" (1998); NOKIA "7210" (2002).

El éxito de los terminales que incorporaban juegos y aplicaciones a su funcionalidad predeterminada, provocó que los fabricantes reorientaran parte de sus líneas de producción a crear dispositivos orientados un poco más hacia el entretenimiento. Un cambio de mentalidad que abrió el mercado de la telefonía a un público joven que exigía constantemente más novedades, más tecnología, más variedad, más posibilidades, y que hizo avanzar el mercado a un ritmo vertiginoso. Con el cambio de siglo llegaron al público general nuevos terminales, más pequeños, con más capacidad y con pantallas de miles de colores, algunos de los cuales incorporaban dos tecnologías que, juntas, formaban el caldo de cultivo perfecto para el mercado de las aplicaciones: J2ME y WAP.

Java 2 Mobile Edition (J2ME, hoy conocido como JME) surgió en 2001 como un subconjunto básico de las librerías y API de Java, diseñado para ser ejecutado en dispositivos embebidos como sistemas de control industrial, pequeños electrodomésticos y, por supuesto, terminales móviles y PDA. Su llegada al mundo de lo inalámbrico facilitó enormemente el desarrollo de aplicaciones para este tipo de dispositivos, puesto que ya no era necesario conocer lenguajes específicos de cada dispositivo o fabricante ni disponer obligatoriamente de un terminal en el que hacer las pruebas: los programadores podían crear sus aplicaciones cómodamente en su ordenador personal, con un emulador y usando un lenguaje fácil y potente como Java. J2ME abrió las puertas de los terminales móviles a los desarrolladores.

WAP (Wireless Application Protocol), por otra parte, es un conjunto de protocolos que facilita la interoperabilidad de dispositivos móviles en redes inalámbricas como GSM y CDMA. Aunque comenzó a desarrollarse en 1997, fue a partir de 2001 cuando empezó a extenderse su uso, con la introducción de la funcionalidad "WAP Push". Esta nueva característica permitía a los operadores

enviar a sus usuarios notificaciones con enlaces a direcciones WAP, de manera que el usuario sólo tenía que activarlas para acceder a contenidos en red. Gracias a esto, los terminales móviles podían intercambiar información con servidores web y descargar aplicaciones específicas de una manera sencilla y en cualquier parte. El protocolo WAP rompía así las barreras entre los usuarios de dispositivos móviles y los proveedores de contenidos.

Facilidad de desarrollo y facilidad de distribución formaron una combinación exitosa (no sería la última vez, como veremos más adelante) e hicieron surgir con verdadera fuerza el mercado de las aplicaciones móviles. Los desarrolladores creaban juegos y aplicaciones en Java que servían para múltiples modelos; los proveedores anunciaban estas aplicaciones en revistas especializadas, periódicos e incluso en televisión; y los usuarios descargaban en sus teléfonos fondos de pantalla, tonos y juegos, mediante el simple envío de mensajes SMS a números de tipo *premium* (tarificados de manera especial por la operadora para ofrecerle un beneficio a la compañía que distribuye las aplicaciones).

FIG. 5 – FRAGMENTO DE UNA PÁGINA DE REVISTA, ANUNCIANDO MULTITUD DE CONTENIDOS MULTIMEDIA ACCESIBLES MEDIANTE WAP Y MENSAJES PREMIUM.

No obstante, pronto el escenario volvería a cambiar: se estaba preparando la llegada de los *smartphones*.

4.3.2. Symbian

En el año 2000 (fig. 6), la compañía Ericsson provocó un salto cualitativo en el mercado de los dispositivos móviles: presentó su modelo Ericsson R380 Smartphone, el primer teléfono móvil con pantalla táctil y el primero que integraba toda la funcionalidad de una auténtica PDA manteniendo al mismo tiempo el tamaño, peso y diseño de los terminales de entonces. Aunque la interfaz se parecía demasiado a la de las PDA de la época y a pesar de que era un modelo "cerrado" que no permitía instalar aplicaciones de terceros, fue muy bien recibido por el público más técnico.

FIG. 6 – TELÉFONO MÓVIL ERICSSON MODELO R380 (2000). FUE EL PRIMERO EN COMBINAR UN TELÉFONO MÓVIL CON LAS FUNCIONALIDADES DE UNA PDA.

Este avance fue gracias a que el R380 usaba una versión de un sistema operativo, EPOC, que ya utilizaban algunas PDA de la época y que fue adaptado para su uso en terminales móviles. Dicha versión, conocida como Symbian, comenzó a desarrollarse en 1998 por una asociación conjunta de los fabricantes Nokia, Ericsson, Motorola y Psion, que tenían como objetivo aprovechar la incipiente convergencia entre teléfonos móviles y PDA para desarrollar un sistema operativo abierto que facilitara el diseño de los futuros dispositivos. El Ericsson R380 fue el primer modelo que salió al mercado con este sistema operativo y, ante el éxito cosechado y aprovechando el progreso de la tecnología utilizada en los terminales más recientes, Nokia y Ericsson enfocaron parte de sus esfuerzos a suplir las carencias de Symbian.

Nokia, comprendiendo el potencial del nuevo sistema operativo, decidió utilizarlo en sus modelos "Communicator" que, a pesar de ser los más avanzados (y caros) de toda su gama —ya desde varios años antes incorporaban acceso a Internet, correo, un teclado completo y varias otras aplicaciones—, estaban hechos sobre una plataforma que empezaba a quedarse pequeña.

Con su modelo Communicator 9210 (fig. 7), Nokia introdujo a mediados del año 2001 en el mercado la versión 6 de Symbian OS, con multitarea real y acompañado de la nueva interfaz gráfica "Series 80" a todo color. El modelo fue muy bien recibido en algunos sectores de público que no estaban familiarizados

con las PDA y para los cuales el Communicator suponía una novedosa solución "todo en uno" para sus negocios.

FIG. 7 – TELÉFONO MÓVIL NOKIA, MODELO "COMMUNICATOR" 9210 (2001), CON TECLADO INCORPORADO. FUE EL PRIMERO EN INCORPORAR SYMBIAN 6.0 Y LA INTERFAZ "SERIES 80".

Sin embargo, el 9210 carecía de pantalla táctil y esto fue visto como un paso atrás por muchos clientes, que estaban acostumbrados a las pantallas táctiles de las PDA y que esperaban que un modelo con características de PDA como era el 9210 también tuviera una. Además, su tamaño de "ladrillo" lo dejaba fuera de la tendencia a la reducción de tamaño que estaba experimentando el mercado generalista de dispositivos.

Nokia no estaba dispuesta a tirar la toalla y en el segundo cuarto de 2002 puso sobre la mesa su modelo 7650 (fig. 8): el primer *smartphone* auténtico, con cámara incorporada y con un diseño agradable y tamaño similar al del resto de teléfonos móviles del mercado. Traía Symbian OS 6.1 y una interfaz gráfica, "Series 60", diseñada específicamente para terminales con pantalla y teclado "cuadrados", que era el resultado del trabajo de Nokia sobre la anterior "Series 80" para facilitar al máximo su uso. Y lo más importante de todo: permitía instalar aplicaciones de terceros.

Este modelo de Nokia fue un éxito instantáneo y el culpable, en gran medida, de que el mercado de las aplicaciones móviles se llenara de programas compatibles con dispositivos basados en S60. La posibilidad de instalar aplicaciones tanto nativas como basadas en J2ME, y de hacer cosas como comprobar el correo, hacer fotos o jugar a un juego, todo ello con una sola mano y utilizando un teléfono elegante que cabía cómodamente en el bolsillo, fue la mejor carta de presentación posible para la plataforma S60 de Nokia y de hecho, hasta 2010, S60 fue la interfaz más utilizada en todo el mercado de terminales móviles.

FIG. 8 – TELÉFONO MÓVIL NOKIA 7650 (2002). CON SYMBIAN OS 6.1 Y 3,6 MB DE ESPACIO ADICIONAL, PERMITÍAN INSTALAR APLICACIONES COMO POR EJEMPLO UNA VERSIÓN DEL NAVEGADOR OPERA.

Sin embargo, todavía hacían falta un par de componentes más para desencadenar la siguiente explosión en el mercado de las aplicaciones móviles, y uno de ellos era el espacio disponible. El Nokia 7650, con sus 3,6 MB de espacio, apenas dejaba espacio para aplicaciones muy elaboradas, y ni siquiera un fichero MP3 de tamaño medio podía ser almacenado en él.

Ericsson, para entonces ya Sony Ericsson, se encargó de dar el siguiente paso en la evolución de Symbian, sacando al mercado su modelo P800 (fig.9).

FIG. 9 – TELÉFONO MÓVIL SONY ERICSSON P800 (FINALES DE 2002), CON TECLADO DESMONTABLE Y PANTALLA TÁCTIL.

Este teléfono móvil fue el primero en traer Symbian 7.0, la siguiente versión del sistema operativo, así como una nueva interfaz gráfica desarrollada por Ericsson (antes de su fusión con Sony) y conocida como UIQ, que permitía instalar aplicaciones desarrolladas por terceros bajo C++ o bajo J2ME. El Sony Ericsson P800, además de su pantalla a todo color y su cámara VGA, llegaba acompañado de 16 MB de espacio pero podía, aprovechando la tecnología de almacenamiento mediante tarjetas SD desarrollada por Sony, ampliarse hasta unos increíbles (para entonces) 128 MB de espacio, lo que permitía al usuario utilizar su teléfono también como reproductor de música y hasta de vídeo.

A partir de este momento, la evolución y expansión del sistema operativo Symbian sería, durante algunos años, imparable. Para los fabricantes, especialmente para Nokia y Sony Ericsson, era más fácil integrarlo en sus terminales al ser un sistema abierto; para los usuarios, era el sistema "de moda", el más compatible con todas las aplicaciones disponibles y el que más cosas podía hacer: varios modelos disponían de navegador web totalmente compatible con HTML, no solo WAP, y también aparecieron modelos con funcionalidades y diseño ideados para actividades específicas como escuchar música o jugar a videojuegos, conquistando así cuota de mercado en nichos tradicionalmente apartados del mundo de la telefonía.

Y aunque los teléfonos más económicos siguieron utilizando sistemas cerrados específicos para cada modelo (casi siempre compatibles con J2ME); y a pesar de la entrada en el mercado de otros competidores que daban un gran valor añadido al servicio de telefonía móvil —como es el caso del fabricante Research In Motion con sus dispositivos BlackBerry, de gran éxito en los países norteamericanos, que permitían mensajería instantánea y servicios de correo electrónico encriptado gracias a su red privada de servidores—; Symbian llegó a copar el mercado mundial de teléfonos móviles, con un 76% de cuota de mercado en 2006. Aún en 2008, más de la mitad de *smartphones* a la venta estaban basados en una versión u otra de Symbian.

	2005	2006	2007	2008
Symbian	68,0	76,1%	63,5%	52,4%
BlackBerry OS (RIM)	2,0	9,2%	9,6%	16,6%
Windows (Microsoft)	4,0	6,7%	12,0%	11,8%
iOS (Apple)	-	-	2,7%	8,2%
Android (Google)	-	-	-	0,5%
Otros (Linux, HP...)	28,0	8,0%	12,1%	10,5%

TABLA 1 – CUOTA DE MERCADO GLOBAL DE LOS PRINCIPALES SISTEMAS OPERATIVOS PARA TERMINALES SMARTPHONE, DE 2005 A 2008. FUENTES: CANALYS, GARTNER.

Sin embargo, ya mediado 2007 aparecerían dos nuevos competidores que impactarían profundamente el monopolio de Symbian y cambiarían la forma de ver el mercado de las aplicaciones móviles: iPhone y Android.

4.3.3. IPhone y Android

Hemos empezado este capítulo hablando de las PDAs. Hasta 1992, nadie había oído nunca hablar de una PDA. Sí, existían dispositivos portátiles que ofrecían una ayuda para organizar tareas y tomar notas; pero no fue hasta ese año que Apple acuñó el término *personal digital assistant (PDA)* para referirse a su última creación: el Apple MessagePad (fig. 10), la primera PDA auténtica, con un nuevo sistema operativo (Newton), pantalla táctil, reconocimiento de escritura y un aspecto e interfaz que serían después imitados por todos los modelos de PDA del mercado.

Hemos empezado este capítulo, por tanto, hablando de Apple; y es curioso, porque vamos a terminarlo hablando de Apple otra vez.

Apple, que a principios de los 90 había revolucionado el mercado de dispositivos móviles con la introducción de sus modelos basados en el sistema operativo Newton, no cosechó sin embargo el éxito que esperaba con ellos. Una autonomía muy limitada por el uso de pilas AAA, un sistema de reconocimiento de escritura defectuoso que requería demasiado tiempo de "aprendizaje" y la falta de conectividad inicial con el ordenador del usuario (los cables y software necesario se comercializaban aparte) fueron las principales causas de que, durante la década de los 90, el mercado fuera dominado por otras compañías con modelos más depurados y Apple se centrara más en su línea de ordenadores personales (llegando a cosechar enorme éxito a finales de siglo con su línea de ordenadores iMac y PowerBook y con su sistema operativo Mac OS).

FIG. 10 – A LA IZQUIERDA, APPLE MESSAGEPAD (1992), LA PRIMERA EN SER LLAMADA PDA. A LA DERECHA, APPLE IPHONE (2007), EL DETONADOR DE LA EXPLOSIÓN DEL MERCADO DE APLICACIONES MÓVILES.

Sin embargo, la mayor revolución de todas, la que en poco tiempo iba a convertir el mundo de las aplicaciones móviles en un negocio de miles de millones

de dólares, llegaría de la mano de Apple con el cambio de milenio... aunque por una vía poco convencional: con música.

Según declaraciones de Steve Jobs, CEO de Apple, por aquel entonces el comité ejecutivo de la compañía no creía que las PDA o los *tablet* PC (ordenadores portátiles con pantalla táctil que permitía su uso sin teclado, como si fueran una tabla) fueran buenas opciones para conseguir un gran volumen de negocio para Apple. Jobs creía que el futuro del mercado estaba en los teléfonos móviles; que éstos iban a hacerse dispositivos importantes para el acceso a la información. Por ello, en lugar de dedicarse a evolucionar su línea de PDA basada en el sistema Newton, los empleados de Apple pusieron todas sus energías en el reproductor de música iPod y la plataforma iTunes, un conjunto de *software* y tienda en línea que permitía a los usuarios del dispositivo la sincronización del mismo con su biblioteca de música así como la compra de nuevas canciones y discos de una manera fácil e intuitiva.

El éxito de la plataforma iTunes fue demoledor. No sólo rompió todos los esquemas de lo que hasta entonces había sido el negocio de las discográficas, liberalizando el mercado y cambiando para siempre el paradigma de la venta de música y otras obras con propiedad intelectual. Fue además la demostración más clara de que cuantos menos impedimentos se le pongan al usuario para que compre algo, más fácil es que lo compre; y la primera piedra de la estructura que faltaba en el mercado de las aplicaciones móviles para convertirlas en el negocio que son hoy en día: una plataforma de distribución que permite a los programadores y empresas publicar y dar visibilidad a sus aplicaciones reduciendo al máximo la inversión necesaria, y a los usuarios adquirir las aplicaciones de la manera más sencilla posible.

Ya en 2008, Apple creó dentro de la plataforma iTunes la App Store, una tienda en línea que permitía a los usuarios comprar y descargar directamente en el móvil o a través del software iTunes aplicaciones para el producto estrella de la compañía, el teléfono móvil iPhone. Con una política de aceptación muy estricta, los desarrolladores que consiguen cumplir todos los requisitos exigidos por Apple pueden publicar en esta tienda sus aplicaciones y obtener el 70% de la recaudación de su venta. Con estas condiciones, muchos desarrolladores adaptaron su trabajo y sus aplicaciones a la plataforma iOS de Apple, basada en el lenguaje Objective-C y en entornos de desarrollo sólo disponibles para sistemas Mac Os, comercializados también por Apple.

El mercado de las aplicaciones móviles disponía por fin de todos los elementos necesarios para su eclosión final, y así inició su despegue imparable, arropado por dispositivos con la tecnología necesaria para posibilitar aplicaciones interesantes y por plataformas de distribución de las mismas a un coste prácticamente cero. No obstante, la mejor parte del pastel, la que más beneficios generaba, estaba reservada a aquellos que abrazaran las tecnologías de Apple, y además era una parte forzosamente pequeña ya que no muchos usuarios podían permitirse adquirir dispositivos como el iPhone, con un coste de varios

cientos de dólares. Por este motivo, en 2008 el monopolio de Symbian aún no peligraba.

Afortunadamente, ese mismo año 2008 llegó la alternativa: Android. Mientras que el negocio de Apple estaba limitado a un único modelo y a una tecnología y lenguaje propietarios, Google se presentaba con un sistema operativo abierto, basado en Linux y Java y que podía ser incluido en cualquier dispositivo móvil independientemente del fabricante. Mediante numerosos acuerdos comerciales, Google consiguió que varios de los más importantes fabricantes de teléfonos móviles, que hasta ese momento estaban vendiendo sus terminales con sistemas operativos propietarios o con distintas versiones de sistemas Symbian o Microsoft, adoptaran el nuevo sistema operativo Android: un sistema operativo moderno, respaldado por una gran empresa como era Google y que les permitiría competir con el rey del mercado, el iPhone, ahorrándose de paso los costes de desarrollo del sistema.

Además, Android llegaba de la mano de una plataforma de distribución de aplicaciones, Android Market, similar a la de Apple pero unas políticas de aceptación mucho menos restrictivas. Con estas condiciones y con un entorno de desarrollo basado en el lenguaje Java y en herramientas disponibles gratuitamente tanto para sistemas Windows como para sistemas Linux/Unix, muchos desarrolladores y empresas con experiencia previa en Java se decantaron por Android como punto de entrada al mercado de las aplicaciones móviles.

Desde entonces, el ascenso de Android ha sido imparable. Si en 2008 tenía menos de un 1% de mercado, en 2011 ha conseguido desbancar a Symbian como sistema operativo más utilizado en *smartphones*, con un 53% del mercado.

4.3.4. Escenario actual

Hoy en día, el mercado de las aplicaciones móviles se reparte principalmente entre Apple con su sistema iOS y Google con su sistema Android.

En el caso de iOS, con tecnologías propietarias y sólo disponibles en sistemas del propio Apple, su fuerza está en una base de usuarios con un perfil económico medio-alto, que permite a los desarrolladores fijar unos precios más jugosos y que compensan de sobra la inversión inicial necesaria para entrar en la App Store. Así, actualmente más de la mitad de los beneficios generados mundialmente por la venta de aplicaciones móviles corresponden a aplicaciones diseñadas para el sistema iOS.

En el caso de Android, su rápida expansión y disponibilidad en multitud de dispositivos de distintos fabricantes, así como unas tecnologías abiertas que permiten un ciclo de desarrollo de aplicaciones con un coste de inversión casi nulo, le han servido para copar el mercado de dispositivos y aprovechar esta posición de fuerza para atraer a más y mejores desarrolladores. Existe ahora mismo una gran comunidad de desarrolladores escribiendo aplicaciones para extender la

funcionalidad de los dispositivos Android. Según datos ofrecidos por Google, la tienda de aplicaciones Android Market ha sobrepasado ya las 400.000 aplicaciones, sin tener en cuenta aplicaciones de otras tiendas no oficiales (como por ejemplo Samsung Apps, de Samsung, o la AppStore de Amazon). Multitud de aplicaciones están siendo portadas desde el sistema iOS al sistema Android, en busca de una base de usuarios más amplia, y es de esperar que Android se estabilice como sistema operativo dominante de aquí en adelante, igual que lo fue Symbian en su momento.

5. El sistema operativo Android

Android es un sistema operativo móvil, es decir, un sistema operativo diseñado para controlar dispositivos móviles como pueden ser un teléfono de tipo *smartphone*, una PDA o una *tablet* (ordenadores portátiles en los que la entrada de datos se realiza normalmente mediante una pantalla sensible al tacto), aunque existen versiones del sistema operativo utilizadas en otros dispositivos multimedia como por ejemplo televisores.

FIG. 11 – "ANDY", EL ROBOT IDEADO COMO LOGOTIPO DEL SISTEMA OPERATIVO ANDROID.

5.1. Historia

La primera versión de Android fue desarrollada por la compañía Android Inc., una firma fundada en 2003 con el objetivo de crear *"(...) dispositivos móviles más inteligentes y que estén más atentos a las preferencias y la ubicación de su propietario"* . Aunque la evolución del trabajo que se llevaba a cabo en Android Inc. se mantuvo prácticamente en secreto, el gigante Google empezaba a mostrar interés en el negocio de los dispositivos y comunicaciones móviles y en 2005

adquirió la compañía, haciéndola subsidiaria de Google Inc. e incorporando a su plantilla a la mayoría de programadores y directivos originales.

A partir de ese momento, Google inició un proceso de búsqueda de acuerdos con diferentes operadores de comunicaciones y fabricantes de dispositivos móviles, presentándoles una plataforma abierta y actualizable y ofreciéndoles diferentes grados de participación en la definición y desarrollo de sus características. La compra por parte de Google de los derechos de varias patentes relacionadas con tecnologías móviles hizo saltar la liebre y los mercados comenzaron a especular con la inminente entrada de la compañía en el negocio de las comunicaciones móviles.

Finalmente, el 5 de noviembre de 2007, el sistema operativo Android fue presentado al mundo por la Open Handset Alliance, un consorcio recién creado de más de 70 empresas del mundo de las telecomunicaciones (incluyendo a Broadcom, HTC, Qualcomm, Intel, Motorola, T-Mobile, LG, Texas Instruments, Nvidia, Samsung... por nombrar algunas), liderado por Google y cuyo objetivo es el desarrollo de estándares abiertos para dispositivos móviles. Bajo esta premisa, Google liberó la mayor parte del código fuente del nuevo sistema operativo Android usando la licencia Apache, una licencia libre, gratuita y de código abierto.

La evolución del sistema operativo desde su presentación ha sido espectacular. Sucesivas versiones han ido incorporando funcionalidades como pantalla multi-táctil, reconocimiento facial, control por voz, soporte nativo para VoIP (Voice over Internet Protocol), compatibilidad con las tecnologías web más utilizadas o novedosas como HTML5 y Adobe Flash, soporte de Near Field Communication, posibilidad de convertir el dispositivo en un punto de acceso WiFi, grabación de vídeo en 3D... así como ampliando el espectro de dispositivos soportados, pudiendo encontrar hoy en día Android en el corazón de teléfonos móviles, tablets, ordenadores portátiles y hasta televisores.

En la actualidad, los últimos datos indican que Android acapara más del 50% de cuota de mercado de *smartphones* a escala mundial, con un crecimiento en el último año de un 380% en número de unidades fabricadas; muy por delante de iOS, el sistema operativo del Apple iPhone que, con una cuota del 19%, ha relegado a su vez a Symbian OS a la tercera posición.

5.2. Diseño

El sistema operativo Android está compuesto de un núcleo basado en Linux, sobre el cual se ejecutan: por una parte, la máquina virtual Dalvik, basada en Java pero diseñada para ser más eficiente en dispositivos móviles; y por la otra, una serie de librerías o bibliotecas programadas en C y C++ que ofrecen acceso principalmente a las capacidades gráficas del dispositivo. Por encima de esta capa encontramos el marco de trabajo de aplicaciones, una colección de librerías Java

básicas adaptadas para su ejecución sobre Dalvik; y finalmente tenemos las aplicaciones Java, que son las que ofrecen la funcionalidad al usuario.

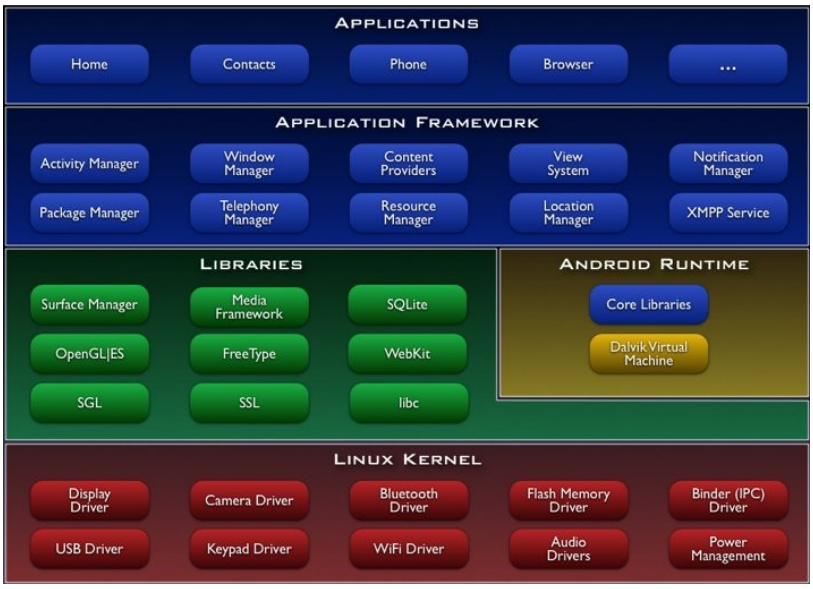

FIG. 12 – LAS DIFERENTES CAPAS QUE CONFORMAN EL SISTEMA OPERATIVO ANDROID.

A continuación presentamos en detalle cada uno de los componentes o capas que conforman el sistema operativo. Para cada uno se indica primero su denominación en inglés, en aras de una mejor identificación en el diagrama superior:

1. ***Applications* (aplicaciones)**: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

2. ***Application framework* (marco de trabajo de aplicaciones)**: los desarrolladores tienen acceso completo a los mismos APIs del *framework* usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del *framework*). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

3. ***Libraries* (bibliotecas)**: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del *framework* de aplicaciones de Android. Las bibliotecas escritas en lenguaje C/C++ incluyen un administrador de pantalla táctil (*surface manager*), un *framework* OpenCore (para el aprovechamiento de las capacidades multimedia), una base de datos relacional SQLite **[Celma 06]**, una API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, el protocolo de comunicación segura

SSL y una biblioteca estándar de C, llamada "Bionic" y desarrollada por Google específicamente para Android a partir de la biblioteca estándar "libc" de BSD.

4. ***Android runtime*** **(funcionalidad en tiempo de ejecución)**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".

5. ***Linux kernel*** **(núcleo Linux)**: Android dispone de un núcleo basado en Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores, y también actúa como una capa de abstracción entre el hardware y el resto de la pila de software. La versión para Android se encuentra fuera del ciclo normal de desarrollo del kernel Linux, y no incluye funcionalidades como por ejemplo el sistema X Window o soporte para el conjunto completo de librerías GNU, lo cual complica bastante la adaptación de aplicaciones y bibliotecas Linux para su utilización en Android. De igual forma, algunas mejoras llevadas a cabo por Google sobre el núcleo Linux original han sido rechazadas por los responsables de desarrollo de Linux, argumentando que la compañía no tenía intención de dar el soporte necesario a sus propios cambios; no obstante ya se han dado los pasos necesarios para asegurar que todas estas mejoras y controladores se incluyan en las próximas versiones del núcleo oficial.

5.3. Estructura básica de una aplicación Android

Las aplicaciones Android están escritas en lenguaje Java. Aunque ya hemos visto que la máquina virtual Dalvik no es una máquina virtual Java, Android toma ventaja de la utilización de un lenguaje de programación consolidado y de libre acceso como es Java para facilitar a los programadores el desarrollo de aplicaciones para su sistema. Así, el proceso de desarrollo y pruebas se puede llevar a cabo usando cualquier entorno de programación Java, incluso la simulación mediante dispositivos Android virtuales, dejando que sea el sistema operativo Android el que en tiempo de ejecución traduzca el código Java a código Dalvik. Si queremos distribuir la aplicación una vez completado el desarrollo de la misma, el SDK de Android compila todo el código Java y, junto con los ficheros de datos y recursos asociados (imágenes, etc.), crea un "paquete Android": un archivo con

extensión .apk que representa a una única aplicación y que es el archivo utilizado por los dispositivos Android para instalar nuevas aplicaciones.

Estas aplicaciones, una vez instaladas, se ejecutarán como un nuevo proceso en el sistema Linux subyacente, utilizando su propio identificador único de usuario y su propia instancia de la máquina virtual Dalvik. Cuando la aplicación deja de utilizarse o cuando el sistema necesita memoria para otras tareas, el sistema operativo Android finaliza la ejecución del proceso, liberando los recursos. De esta forma, Android crea un entorno bastante seguro en el que cada aplicación, por defecto, sólo puede acceder a los componentes estrictamente necesarios para llevar a cabo su tarea, ya que cada proceso está aislado del resto y no puede acceder a partes del sistema para las cuales no tiene permisos.

Como es normal, Android dispone de diferentes maneras de que dos aplicaciones puedan compartir recursos. El más habitual es el sistema de permisos, mediante el cual una aplicación indica qué funcionalidades del sistema necesitará utilizar; la responsabilidad de permitir el acceso recae en el usuario, quien puede aceptar o denegar dichos permisos durante la instalación de la aplicación.

5.4. Componentes de una aplicación

Según las tareas que necesite llevar a cabo, una aplicación Android puede estar formada por uno o varios de los siguientes componentes:

- ***Actividades (implementadas como subclases de** Activity**)***

  Una actividad representa una pantalla de la aplicación con la que el usuario interactúa. Por ejemplo, una aplicación que reproduzca música podría tener una actividad para navegar por la colección de música del usuario, otra actividad para editar la lista de reproducción actual, y otra actividad para controlar la canción que se está escuchando actualmente. Aunque una aplicación puede tener múltiples actividades, que juntas ofrecen la funcionalidad completa de la aplicación, las actividades son elementos independientes. Esto implica que otra aplicación distinta podría invocar cualquiera de estas actividades para utilizar su funcionalidad, si tiene los permisos adecuados. Por ejemplo, una aplicación de selección de tonos de llamada podría invocar la actividad que muestra la colección de música del usuario para permitirle elegir una canción como tono personal. Otro ejemplo que podemos ver a diario es la utilización de la actividad "Cámara" (incluida por defecto en Android) cada vez que una aplicación quiere permitir al usuario tomar una fotografía, ya sea para enviarla a un amigo, para fijarla como fondo de pantalla, para subirla a redes sociales, para dibujar sobre ella...

- ***Servicios (implementados como subclases de** Service**)***

  Un servicio no tiene interfaz de usuario, ya que se ejecuta en segundo plano para llevar a cabo operaciones de mucha duración y funcionalidades que, en general, no requieren de intervención del usuario. Siguiendo con el ejemplo de la aplicación musical, ésta podría implementar un servicio que continúe reproduciendo canciones en segundo plano mientras el usuario lleva a cabo otras tareas, como comprobar su correo o visualizar una presentación de fotografías. La descarga de contenidos desde Internet también se suele llevar a cabo mediante un servicio en segundo plano, que notifica la finalización de la misma a la aplicación original.

- ***Proveedores de contenido (implementados como subclases de** ContentProvider**)***

  Un proveedor de contenidos es un componente que controla un conjunto de datos de una aplicación. El desarrollador puede almacenar estos datos en cualquier ubicación a la que su aplicación pueda acceder (sistema de archivos, base de datos SQLite **[Celma 06]**, servidor web) y luego crear un proveedor de contenidos que permita a otras aplicaciones consultar o incluso modificar estos datos de manera controlada. El reproductor de música podría incluir un proveedor de contenidos que permita a otras aplicaciones consultar el catálogo de música del usuario y añadir nuevas entradas al mismo. Android ofrece a todas las aplicaciones con permiso para ello acceso a la lista de Contactos, también mediante un proveedor de contenidos.

- ***Receptores de notificaciones (implementados como subclases de** BroadcastReceiver**)***

  Un receptor de notificaciones responde a notificaciones lanzadas "en abierto" para todo el sistema. Muchas notificaciones las lanza el propio sistema, por ejemplo avisos de batería baja, de que la pantalla se ha apagado o de que hay una llamada entrante. Las aplicaciones también pueden lanzar notificaciones, por ejemplo para informar al resto del sistema de que se ha descargado algún tipo de información y que ya está disponible. Aunque los receptores de notificaciones no tienen interfaz de usuario, sí que pueden informar a éste de los eventos detectados, mediante un icono en la barra de notificaciones. En nuestro ejemplo, el reproductor de música podría tener un receptor de notificaciones para detectar las llamadas entrantes y detener la reproducción en curso o pasar el reproductor a segundo plano.

En Android, cualquier aplicación puede ejecutar cualquier componente de otra aplicación distinta. Es una característica única del sistema operativo Android que facilita muchísimo el desarrollo de nuevas aplicaciones ya que permite

reutilizar, para funcionalidades comunes, componentes ya existentes en otras aplicaciones, pudiendo así centrar los esfuerzos de desarrollo en las funcionalidades y componentes nuevos que generen valor añadido. Por ejemplo, podríamos crear una aplicación de tipo videojuego que permita asignarle a cada jugador una fotografía. En otro sistema operativo sería necesario codificar toda la lógica de acceso a la cámara del dispositivo y la interfaz de usuario para ello, o al menos incluir en nuestra aplicación las llamadas a la API y las librerías necesarias; en Android podemos simplemente invocar la actividad Cámara que, una vez finalizada su ejecución, nos retornará la imagen capturada.

Para conseguir esto, dado que por seguridad cada aplicación se ejecuta aislada de las demás y no tiene acceso directo al resto de aplicaciones, lo que tenemos que hacer es indicarle al sistema operativo que tenemos *intención* de lanzar una actividad o componente. Esta solicitud se efectúa mediante la creación de un objeto `Intent` con el identificador del componente o servicio. El sistema entonces comprueba los permisos, ejecuta (o reactiva) el proceso asociado a la aplicación propietaria del componente, e instancia las clases necesarias para ofrecer la funcionalidad solicitada. La lógica de estas clases se ejecutará en el proceso de la aplicación propietaria, no de la aplicación usuaria, con lo cual el componente sigue estando restringido por los permisos que tenía originalmente.

5.5. Manifiesto de la aplicación

Para que Android pueda ejecutar un componente cualquiera de una aplicación, la aplicación debe primero informar al sistema de cuáles son sus componentes mediante el fichero AndroidManifest.xml, conocido como "manifiesto" de la aplicación. En este fichero, cada aplicación proporciona entre otras cosas los siguientes datos:

- **Componentes de la aplicación**: actividades, servicios y proveedores de contenido deben ser declarados obligatoriamente en este fichero, mediante el nombre canónico de su clase principal. Este será el identificador utilizado en los objetos `Intent` por el resto de aplicaciones para solicitar al sistema la ejecución de un componente. Los receptores de notificaciones son los únicos componentes que además de en el manifiesto pueden "registrarse" en el sistema posteriormente, en tiempo de ejecución.

- **Capacidades de los componentes**: aunque una aplicación puede ejecutar un componente externo invocándolo directamente por su nombre canónico, existe un método más potente, basado en acciones. Utilizando acciones, la aplicación que crea el objeto Intent sólo tiene que indicar qué acción quiere llevar a cabo, dejando que sea el sistema el que busque componentes que puedan llevarla a cabo; por ejemplo, una aplicación podría indicar que tiene la intención de ejecutar la acción ACTION_SEND (enviar un mensaje), el sistema relacionaría la

acción con la actividad "enviar" del programa de mensajería y lanzaría dicha actividad al ser invocada esta acción. En el manifiesto de la aplicación, cada componente puede indicar opcionalmente una lista de acciones que está preparado para llevar a cabo, de manera que el sistema lo tendría en cuenta a la hora de ejecutar cualquiera de esas acciones.

- **Permisos necesarios**: cada aplicación debe informar en su manifiesto de todos los permisos que necesita para su funcionamiento. El usuario no puede aceptar o denegar permisos individuales: o bien los acepta todos e instala la aplicación, o los deniega todos y cancela la instalación.

- **Librerías y API externas utilizadas**: librerías externas a la propia API del sistema Android, que la aplicación utiliza para funcionar; por ejemplo, las librerías de Google Maps.

- **Requisitos mínimos**: en el manifiesto se indican tanto el nivel mínimo de la API de Android obligatorio para el funcionamiento de la aplicación, como características específicas de hardware y software sin las cuales la aplicación no pueda desarrollar sus capacidades. Por ejemplo, si nuestra aplicación utiliza las capacidades *multitouch* de Android (sensibilidad a múltiples puntos de contacto), tendremos que especificar en el manifiesto que nuestra aplicación requiere una pantalla capaz preparada para *multitouch*, y al menos un nivel 5 de API (primer nivel en el que se introdujeron los eventos relacionados).

El manifiesto de una aplicación se encuentra en el directorio raíz del paquete .apk y, aunque el sistema operativo no utiliza toda la información que contiene, otras aplicaciones como Android Market sí que utilizan esta información para saber si una aplicación es compatible con el dispositivo que está accediendo al servicio.

5.6. Recursos de la aplicación

Además de clases Java y el manifiesto, una aplicación Android se acompaña normalmente de una serie de recursos separados del código fuente, como pueden ser imágenes y sonidos, ficheros XML para las visuales y cualquier otro recurso relacionado con la presentación visual o la configuración de la aplicación. Estos recursos se organizan en directorios dentro del paquete .apk como sigue (se muestran sólo los más habituales):

- `res`: directorio raíz de los recursos.

- `drawable`: este directorio contiene imágenes en formato PNG, JPG o GIF, que después se cargarán en la aplicación como objetos de tipo Drawable.

- **`layout`**: directorio para almacenar las visuales XML de la aplicación, es decir, ficheros XML que contienen la definición de las interfaces de usuario.

- **`menu`**: contiene ficheros XML que definen los menús utilizados en la aplicación.

- **`values`**: agrupa ficheros XML que definen múltiples recursos de tipos sencillos, como cadenas de texto predefinidas, valores enteros constantes, colores y estilos. Para definir cadenas se suele utilizar el fichero `strings.xml`, para definir estilos el fichero `styles.xml`, etc.

- **`assets`**: directorio para almacenar ficheros que la aplicación accederá de manera clásica, mediante la ruta y el nombre, sin asignarle un identificador único ni transformarlo antes a un objeto equivalente. Por ejemplo, un grupo de ficheros HTML que se mostrarán en un navegador web, deben poder accederse mediante su ruta en el sistema de archivos para que los hiperenlaces funcionen.

Para cada uno de estos recursos (excepto los contenidos del directorio `assets`), el SDK de Android genera un identificador único y lo almacena como una constante dentro de la clase `R.java`. Así, una imagen situada en `res/drawable/thumb1.png` podrá referenciarse mediante la variable `R.drawable.thumb1`, y `R.layout.principal` apuntaría a una visual XML guardada en `res/layout/principal.xml`.

Además, el desarrollador puede añadir sufijos a cualquiera de los directorios para definir recursos que sólo estarán disponibles cuando se den las condiciones definidas por ese sufijo. Un uso muy habitual es definir directorios `drawable-ldpi` y `drawable-hdpi` para almacenar recursos que sólo deben utilizarse en casos de que el dispositivo tenga una resolución de pantalla baja (ldpi) o alta (hdpi). Otro uso es disponer de versiones "localizadas" de las distintas cadenas de texto utilizadas en la aplicación, creando directorios como `res/values-en` y `res/values-de` para guardar la traducción al inglés y al alemán de los textos de nuestra aplicación.

Con esto finalizamos la introducción al sistema operativo Android y sus aplicaciones. Para conocerlo en mayor profundidad, se puede acudir a la bibliografía **[Wei 11] [Android 12]**, aunque es más edificante construir nuestra propia aplicación Android y aprender sobre la marcha.

6. Información a manejar y sus relaciones

Este programa manejará toda la información necesaria para un docente en su práctica, siendo ésta muy numerosa:

- Horario.
- Grupos de clase.
- Fichas personales de los alumnos.
- Faltas de asistencia.
- Calificaciones.
- Instrumentos de evaluación (calificables y no calificables).
- Calendario docente.
- Temporalización.

Antes de proceder a la presentación de las pantallas de información y de acceso a las mismas, es interesante introducir una serie de terminología utilizada en el ámbito educativo, a fin de una mejor comprensión.

La evaluación se define como ***"la valoración del proceso de enseñanza-aprendizaje que se hace en función de una toma de datos sobre dicho proceso y que permite tomar decisiones ajustadas para que se desarrolle conforme a las finalidades propuestas en él"*** **[De Pablo 92]**.

De acuerdo a esta definición, llamaremos ***instrumentos de evaluación no calificables***, a aquellos indicativos que el docente tendrá en cuenta en la toma de decisiones finales de la evaluación del alumno pero que no tienen un valor numérico asociado a ellos. Por ejemplo, un docente puede a lo largo de una evaluación anotar los comportamientos negativos o positivos de un alumno, simplemente poniéndole en su ficha un negativo o un positivo, pero estas anotaciones sólo tomarán sentido cuando al final de la evaluación se utilicen en la toma de decisiones para calificar el ***comportamiento*** del alumno.

Por el contrario, definiremos los ***instrumentos de evaluación calificables*** como, todos aquellos que llevan asociados una valoración numérica, como por ejemplo una prueba evaluativa final o parcial.

Entendemos por ***grupo*** al conjunto de individuos que comparten un aula durante un periodo lectivo.

Una ***temporalización de contenidos*** es aquella especificación en la que se definen el orden y el número de periodos lectivos que ocuparán cada una de las unidades didácticas que engloban la programación, así como el resto de actividades necesarias para completar la evaluación del alumno.

Los ***periodos lectivos*** en el contexto de un horario, se entienden como cada una de las franjas horarias en las que está dividido y que están dedicados a una clase de una asignatura.

Vamos a introducir un diagrama Entidad-Relación en el que veremos las relaciones del docente, los alumnos, los grupos, las asignaturas, etc. Un **diagrama o modelo E-R** es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades.

Empezaremos exponiendo las partes principales del diagrama de manera que su comprensión resulte algo más sencilla. Todos los modelos que a continuación se presentan deben contener, al menos, los datos que se incluyen, quedando a criterio del diseñador/programador la inclusión de nuevos datos.

Casilla horaria: Este modelo corresponde a cada una de las casillas que conforman el ***horario del docente*** referenciado en el presente documento como “FIG. 18”.

Como podemos ver, la casilla horaria puede tener dos valores, o bien el ***grupo*** al que se imparte clase, o bien la actividad no lectiva correspondiente a esa hora (una guardia, atención a padres, preparación de prácticas,…).

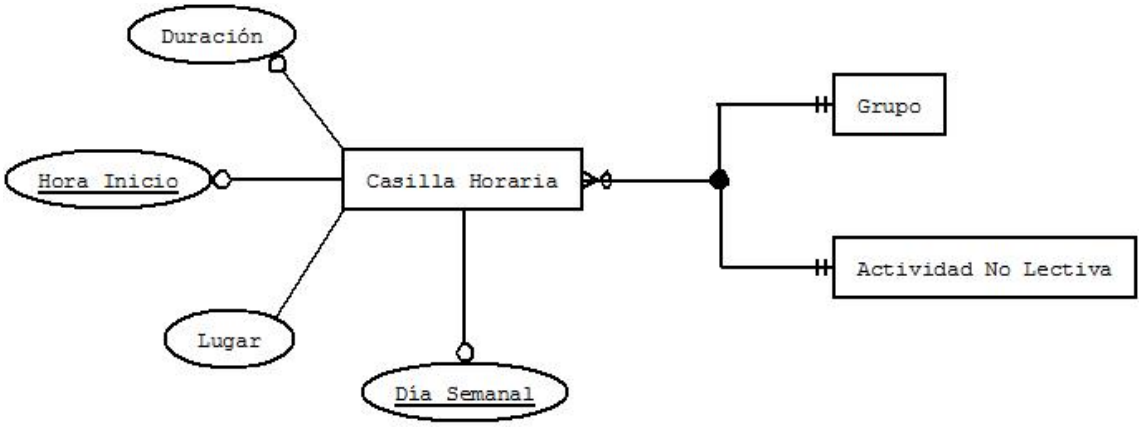

FIG. 13 – DIAGRAMA HORARIO

Fecha de Calendario Docente: Cada una de las fechas que conforman el calendario docente deben estar definidas por, al menos estos datos. Todas las fechas del curso formarán el ***calendario docente***.

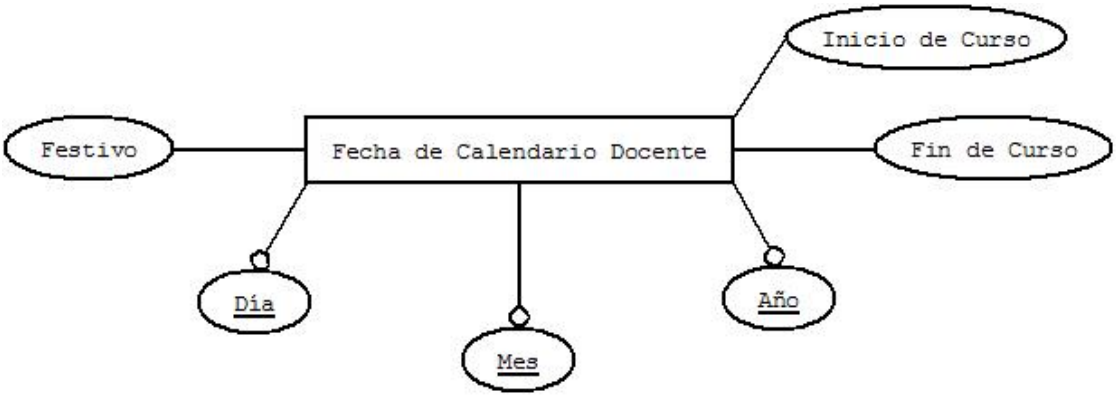

FIG. 14 – DIAGRAMA CALENDARIO DOCENTE

Temporalización: Destacar en este apartado lo que entendemos por ***actividad lectiva***; toda aquella actividad que se realiza en un aula teniendo como finalidad la explicación, práctica o evaluación de los contenidos de una asignatura. Decir que la duración de dichas actividades las mediremos en ***periodos lectivos***.

Destacar que la fecha y hora concreta de una actividad lectiva, la calcula el programa a partir de la totalidad de datos integrados en el programa (Calendario y horario del docente). En la "FIG. 26" podemos ver como se mostrará la ***temporalización de contenidos***.

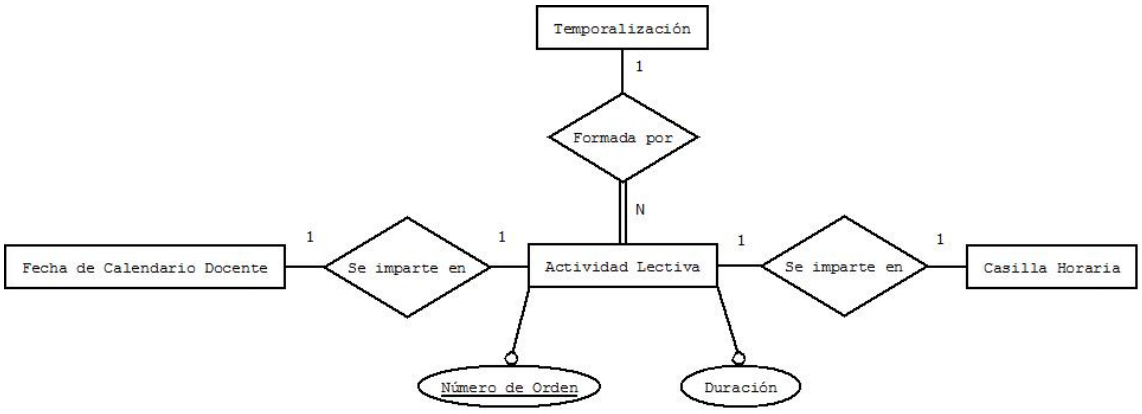

FIG. 15 – DIAGRAMA TEMPORALIZACIÓN

Grupo: Por último, vemos el diagrama de un grupo. Como se puede ver, este diagrama está estrechamente relacionado con el alumno, ya que éste no tiene sentido sin ellos.

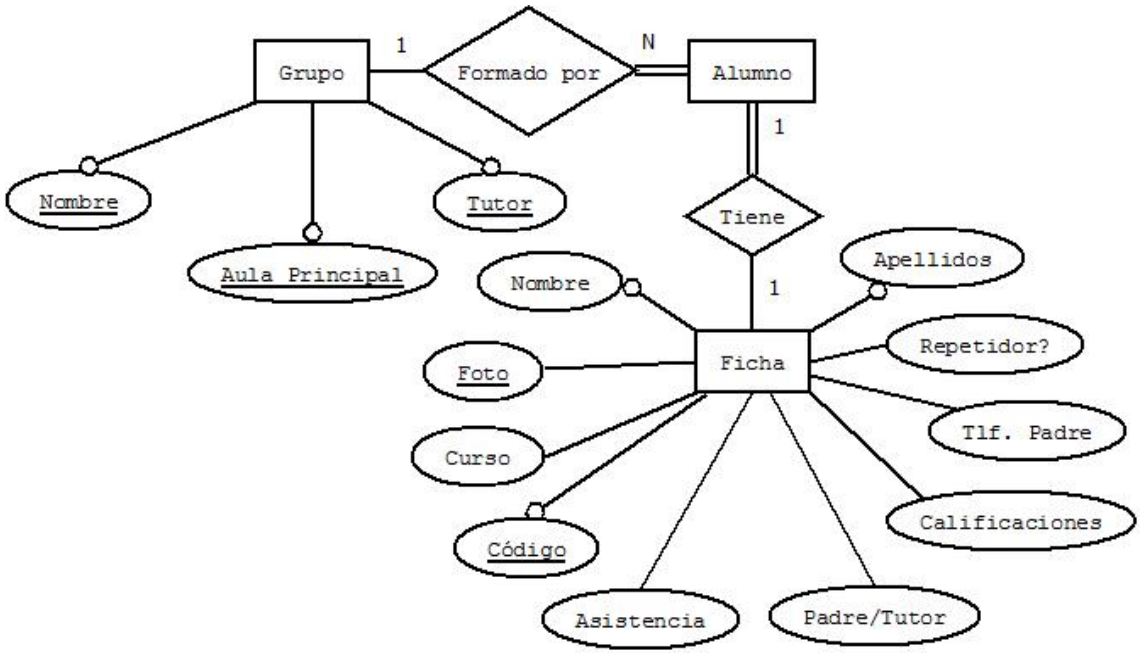

FIG. 16 – DIAGRAMA GRUPO

Finalmente, la figura 17 muestra todos los datos integrados en un diagrama E-R global.

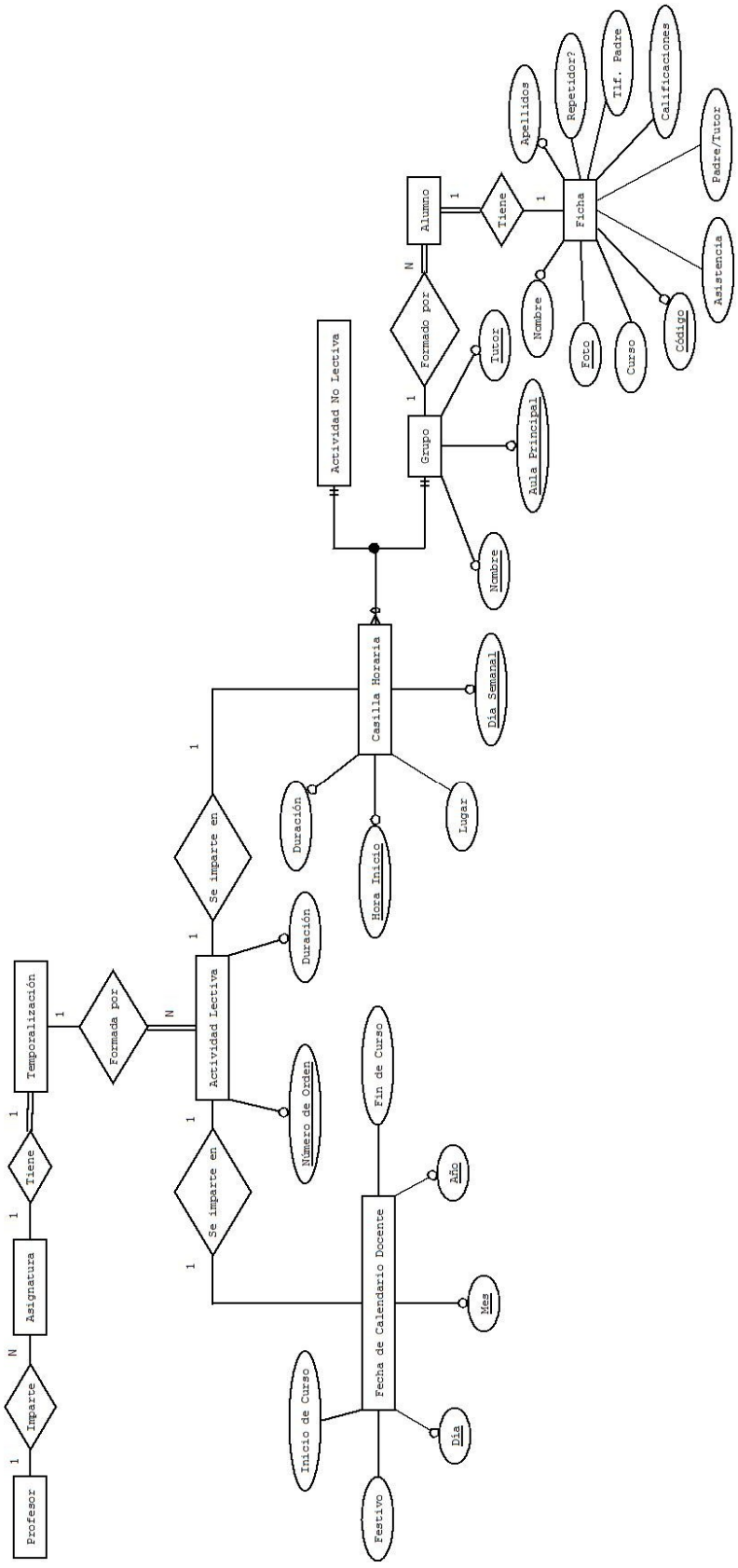

FIG. 17 – DIAGRAMA E-R

Veamos el aspecto de las pantallas y la información que se maneja en cada una de ellas.

7. Pantallas

7.1. Horario

La información horaria indica el grupo que tiene que impartir el docente en un día y hora determinados. Debe mostrarse al usuario en forma de tabla, de forma semejante a la indicada en la figura 18.

Según el **REQ01**, sobre la ventana será posible establecer el nombre del centro educativo y del docente (ver caso de uso 10.7) y el curso corriente.

Sobre cada una de las casillas de esta tabla deberán poder realizarse las siguientes acciones (10.12):

1. Fijar el grupo de cada casilla **(REQ01)**.
2. Fijar el color de la casilla **(REQ02)**.
3. Realizar las siguientes acciones sobre el grupo indicado en la casilla, siguiendo los pasos mostrados en el caso de uso 10.15, **(REQ03)**:

    a. Listado de alumnos.
    b. Control de asistencia.
    c. Calificación de alumnos.
    d. Temporalización del curso para el grupo seleccionado.

La aplicación deberá establecer un procedimiento claro para seleccionar las acciones 1, 2 ó 3; por ejemplo, pulsando de manera continuada sobre la casilla, se podrá fijar su grupo y color, con una pulsación corta saldrá un menú dando las tres opciones restantes.

FIG. 18 – EJEMPLO HORARIO FINAL

7.2. Fichas individuales de los alumnos

Una de las pantallas más importantes de la aplicación son las fichas individuales de los alumnos **(REQ04)**. Estas fichas están divididas en tres pantallas "Datos personales", "Faltas de asistencia" y "Calificaciones del alumno".

La pantalla de ***datos personales*** (fig. 19), debe incluir información que permita identificar al alumno y diferenciarlo del resto:

1. Nombre.
2. Apellidos.
3. Foto.
4. Número de clase.
5. Clase a la que pertenece.
6. Nombre del padre o tutor de contacto.
7. Teléfono del padre o tutor.
8. Si es o no repetidor.

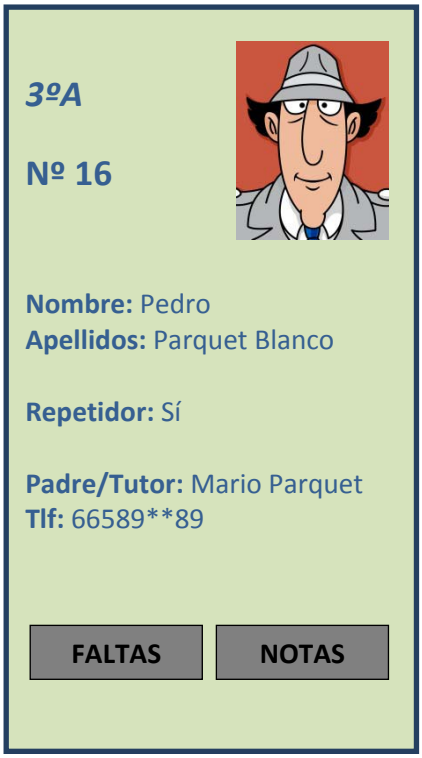

FIG.19 – DATOS PERSONALES

Se deberá establecer un procedimiento sencillo para que el usuario sea capaz de editar cualquier dato que aquí aparece.

La pantalla de ***faltas*** (fig.20), permitirá visualizar, editar, añadir y eliminar de una manera sencilla las faltas del alumno en el periodo de tiempo seleccionado. Igualmente debe ser posible seleccionar el periodo de tiempo a visualizar.

Se incluirá un sencillo código para diferenciar entre las faltas justificadas (J), las no justificadas (NJ) y los retrasos (R).

Se incluirá un botón para añadir una falta nueva, sin tener que salir a otras pantallas.

FIG.20 – FALTAS DE ASISTENCIA

En la pantalla de ***calificaciones del alumno*** (fig.21) veremos los diferentes instrumentos de evaluación utilizados (los calificables y los no calificables).

La pantalla de ***calificaciones*** permitirá visualizar, editar, añadir y eliminar de una manera sencilla las calificaciones del alumno en el periodo de tiempo seleccionado y su nota media para ese periodo. También debe ser posible seleccionar el periodo de tiempo a visualizar.

Junto a cada instrumento de evaluación se podrá ver la fecha de realización de dicho instrumento, así como su calificación, siempre y cuando el instrumento sea calificable.

Los instrumentos de evaluación se definen en la pantalla de configuración correspondiente (caso de uso 10.11).

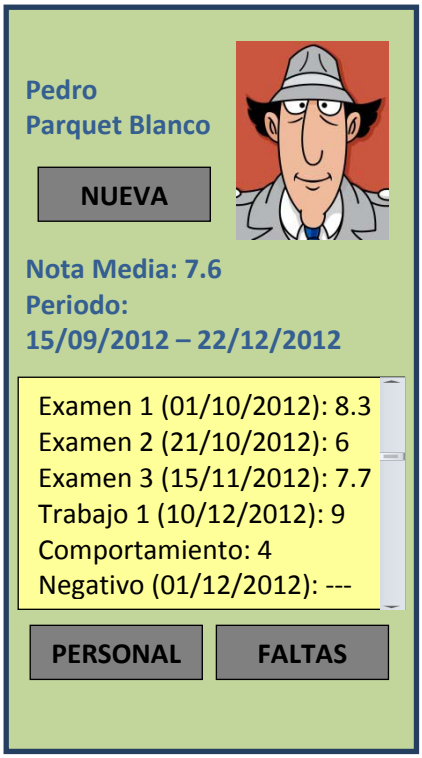

FIG.21 – CALIFICACIONES DEL ALUMNO

Todas las pantallas antes descritas, contarán con dos botones al final de la misma de manera que sea posible navegar entre ellas.

Por último, decir que se debe poder pasar al alumno anterior y al siguiente de manera sencilla sin abandonar la pantalla actual, por ejemplo, haciendo un movimiento de izquierda a derecha pasaríamos al alumno anterior y realizando un movimiento de derecha a izquierda al alumno siguiente.

7.3. Listado de alumnos

Vemos el listado de alumnos del grupo seleccionado **(REQ03)**.

PALENCIA PEREZ, JAVIER
PONCE GIMENEZ, MARCOS
PORTERO CEGARRA, JAVIER
POSTIGO GARCÍA, MARCOS
PREGO NIETO, JOSE
ROS SÁNCHEZ, IGNACIO
RUBIO GARCIA, LUIS
RUBIO MADRID, ARTURO
RUEDA PÉREZ, PAULA
SÁNCHEZ SOTO, MARGARITA
SÁNCHEZ GUERRERO, AMANDA
SÁNCHEZ MARTÍNEZ, PAULA
SÁNCHEZ RUIZ, JOSE JAVIER
SERNA DE HARO, PAULA
VALERO GALLEGO, LOLA
VELA GARCÍA, IRENE
VELASCO OTERO, LUCÍA
ZAMORA ÁLVAREZ, JOSÉ

FIG. 22 – LISTADO DE ALUMNOS

Desde esta pantalla, pulsando sobre cualquiera de los alumnos, accederemos a sus fichas personales.

Sobre el listado de alumnos se podrán definir diferentes criterios de ordenación y selección del alumnado:

I. Ordenación por faltas de asistencia.
II. Ordenación por nota media.
III. Ordenación por nota de una prueba evaluativa concreta.
IV. Selección de los alumnos con más de una falta de asistencia
V. Selección de los alumnos con más de dos pruebas evaluativas suspensas.

7.4. Datos de control de asistencia

En esta pantalla se deberá poder seleccionar los alumnos que han faltado a clase en el presente día **(REQ05)**.

La fecha se tomará de la que el terminal tenga configurada y quedará guardada en una base de datos **[Celma 06]** de la aplicación.

Una vez pasada lista, las faltas de los alumnos se reflejarán en sus fichas personales.

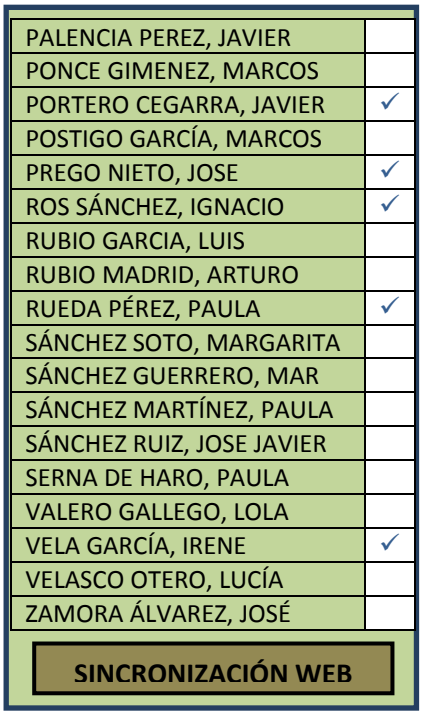

FIG. 23 – CONTROL DE ASISTENCIA

Será muy conveniente que una vez pasada lista se posibilite la opción de sincronizar nuestro registro de asistencia con el de la página web de la Consejería de Educación correspondiente al docente, en nuestro caso la de la Región de Murcia **(REQ08)**.

7.5. Introducción de calificaciones

La pantalla de introducción de calificaciones **(REQ06)** variará en función de que el instrumento de evaluación sea o no calificable. Si el instrumento es calificable (FIG. 24), al lado del nombre del alumno aparecerá una casilla en la que se pueda introducir el valor numérico del instrumento en cuestión de una manera sencilla; por ejemplo mediante el teclado del terminal.

Una forma de hacer esta pantalla sencilla de manejar sería, por ejemplo, empezar pulsando en la casilla de calificación del primer alumno a calificar, de manera que se despliega el teclado del terminal y se introduce la nota. Al pulsar "enter" en el teclado, el programa pasa automáticamente al siguiente alumno.

Si el instrumento es no calificable (FIG. 25), al lado del nombre del alumno aparecerá una casilla en la que se pueda seleccionar los alumnos a los que se les añadirá en su ficha personal dicho instrumento de evaluación.

PALENCIA PEREZ, JAVIER	8
PONCE GIMENEZ, MARCOS	6.1
PORTERO CEGARRA, JAVIER	4.5
POSTIGO GARCÍA, MARCOS	5
PREGO NIETO, JOSE	1.3
ROS SÁNCHEZ, IGNACIO	3
RUBIO GARCIA, LUIS	3
RUBIO MADRID, ARTURO	5
RUEDA PÉREZ, PAULA	9
SÁNCHEZ SOTO, MAR	10
SÁNCHEZ GUERRERO, MAR	8
SÁNCHEZ MARTÍNEZ, PAULA	8
SÁNCHEZ RUIZ, JOSE JAVIER	7
SERNA DE HARO, PAULA	6
VALERO GALLEGO, LOLA	4
VELA GARCÍA, IRENE	4.4
VELASCO OTERO, LUCÍA	8.7
ZAMORA ÁLVAREZ, JOSÉ	5

ACEPTAR

FIG. 24 – INSTRUMENTOS CALIFICABLES

PALENCIA PEREZ, JAVIER	
PONCE GIMENEZ, MARCOS	
PORTERO CEGARRA, JAVIER	✓
POSTIGO GARCÍA, MARCOS	✓
PREGO NIETO, JOSE	
ROS SÁNCHEZ, IGNACIO	
RUBIO GARCIA, LUIS	
RUBIO MADRID, ARTURO	✓
RUEDA PÉREZ, PAULA	
SÁNCHEZ SOTO, MAR	
SÁNCHEZ GUERRERO, MAR	
SÁNCHEZ MARTÍNEZ, PAULA	✓
SÁNCHEZ RUIZ, JOSE JAVIER	
SERNA DE HARO, PAULA	
VALERO GALLEGO, LOLA	
VELA GARCÍA, IRENE	
VELASCO OTERO, LUCÍA	
ZAMORA ÁLVAREZ, JOSÉ	

ACEPTAR

FIG. 25 – INSTRUMENTOS NO CALIFICABLES

7.6. Temporalización de contenidos

En esta pantalla podremos ver cuales son los contenidos previstos para el día en el que nos encontramos, cuáles nos tocan el día siguiente y cuáles dimos el día anterior **(REQ10)**.

Además podremos movernos de una manera fácil por la programación de contenidos, a fin de poder ver en un momento determinado en qué punto de ella nos encontramos.

FIG. 26 – TEMPORALIZACIÓN DE CONTENIDOS

Se incluirá una línea de tiempos en la que veremos cuál es el orden de las unidades didácticas y se remarcará en la que nos encontramos.

8. Requisitos del software

Código	REQ01	**Prioridad**	**Alta/Imprescindible**
Nombre	*Configuración del horario*		
Descripción	La pantalla principal del programa es el horario del docente. La configuración de este horario debe ser fácil e intuitiva, al mantener pulsada una casilla 2 segundos aparecerá un menú con todos los **grupos** configurados, donde elegiremos uno. Una vez elegido el grupo aparecerá otro submenú donde elegiremos la asignatura dada a ese grupo. Por último, se nos pedirá que introduzcamos el aula en el que se impartirá la clase. En esta pantalla aparecerá el nombre del docente y del centro educativo (introducido en el requisito REQ10). Al hacer una pulsación corta sobre una casilla aparecerá la pantalla de **selección de datos** (REQ03). Para poder configurar el número de horas que se imparten cada día, la duración de las clases, la hora de inicio, del recreo, y de finalización, se utilizará el menú "***Configuraciones generales***" (REQ10), incluido en el **menú principal**. Este menú tiene todas las opciones de configuración del programa. Desde cualquier pantalla de la aplicación podremos volver al horario con una opción que aparecerá al pulsar la tecla "atrás".		

Código	REQ02	**Prioridad**	**Baja/Opcional**
Nombre	*Selección del color de las casillas del horario*		
Descripción	Una posibilidad para la mejora del uso intuitivo de la aplicación sería, que se ofreciera como paso siguiente a la configuración del horario, la selección de un **color** para la casilla que estemos configurando. Esto permite al docente, una vez configurado todo el horario, ver que días y horas tiene clase con un grupo concreto, por ejemplo, siempre y cuando se hayan configurado del mismo color las casillas de clase con ese grupo.		

Código	REQ03	**Prioridad**	**Alta/Imprescindible**
Nombre	*Pantalla de selección datos*		
Descripción	Una vez seleccionado un grupo en el horario, aparecerá una pantalla de selección en el que se dan las opciones: a) Fichas individuales del alumnado. b) Asistencia. c) Instrumentos de evaluación. Podremos poner notas, ver estadísticas de un examen y ver las notas medias de los alumnos de todo el curso o de un periodo determinado. d) Temporalización.		

Código	REQ04	**Prioridad**	**Alta/Imprescindible**
Nombre	Fichas individuales de alumnos		
Descripción	Al pulsar sobre cada uno de los alumnos de la lista se abrirá una ficha individual de éste. En ella se incluirán sus datos personales (nombre, apellidos, número de clase, teléfono de padres, nombre padre de contacto y si es repetidor) y un recuadro para incluir la foto del alumno. Ésta será suficientemente grande para apreciar los rasgos del alumno pero dejando espacio para poder incluir el resto de los datos. Se incluirán dos botones (faltas y notas) con los que se podrán acceder a las faltas del alumno y a sus notas. Al pulsar sobre estos botones se accederá al registro de faltas y notas del alumno. Desde aquí, no sólo podremos ver las notas y las faltas, sino que podremos corregir cualquier posible error. En la pantalla de faltas aparecerá en número total de faltas acumuladas. Desde cualquiera de estas tres pantallas debe ser posible acceder a las otras dos pulsando sobre un simple botón.		

Código	REQ05	**Prioridad**	**Alta/Imprescindible**
Nombre	*Control de faltas del alumnado*		
Descripción	Aparece la lista de alumnos del curso elegido con una casilla al lado de cada nombre. Aquellos alumnos a los que se les marque la casilla adyacente, se les incluirá automáticamente en su ficha personal, la falta con la fecha del día. Se utilizará la fecha del terminal. Al terminar de pasar lista se pulsará "aceptar" para salir a la pantalla de selección de datos "REQ03".		

Código	REQ06	**Prioridad**	**Alta/Imprescindible**
Nombre	*Control de calificaciones*		
Descripción	De manera similar al control de faltas, aparecerá la lista de alumnos con una casilla en la que seleccionar la calificación adecuada. En la parte inferior de la pantalla aparecerá un selector en el que elegiremos un tipo de nota de entre las que hayamos configurado. Las notas quedarán grabadas al pulsar "aceptar" al pie de la pantalla. De nuevo los datos se incluirán en las fichas individuales de los alumnos.		

Código	REQ07	**Prioridad**	**Alta/Imprescindible**
Nombre	*Introducción de listado de alumnos*		
Descripción	Una de las opciones que dará el botón menú será la introducción del listado de alumnos. Esto se realizará por medio de un archivo hoja de cálculo, archivo de texto o si es posible según el requisito REQ08		

Código	REQ08	**Prioridad**	**Media/Recomendable**
Nombre	*Sincronización web*		
Descripción	Sería deseable que el programa ofreciese la posibilidad, al incluir el listado de alumnos, de acceder vía web a la página web, "*Plumier XXI*", de la Consejería de Educación y realizar una sincronización con los datos allí alojados. También se agradecería que se ofreciese la posibilidad de sincronizar las faltas tomadas en una clase con el *Plumier XXI.*		

Código	REQ09	**Prioridad**	**Alta/Imprescindible**
Nombre	*Captación de imágenes*		
Descripción	La aplicación será capaz de incorporar imágenes, tomadas con la cámara fotográfica del terminal, a las fichas individuales de los alumnos. Cada ficha contará con un recuadro del tamaño la foto deseada para la ficha, al pulsar de manera continuada 2 segundos sobre el recuadro se abre la cámara para poder tomar la foto del alumno. Una vez tomada la foto se incorporará a la ficha del alumno con el tamaño necesario para la aplicación.		

Código	REQ10	**Prioridad**	**Alta/Imprescindible**
Nombre	*Temporalización*		
Descripción	Se podrán introducir los contenidos temporalizados para cada grupo. Estos contenidos estarán disponibles para su consulta y modificación en cualquier momento por parte del usuario.		

Código	REQ11	**Prioridad**	**Alta/Imprescindible**
Nombre	*Configuraciones generales*		
Descripción	En esta opción se podrán configurar las siguientes variables: a) Fecha de inicio y final del curso b) Festividades del curso c) Duración y número de los periodos lectivos d) Hora de inicio de las clases y duración del recreo e) Nombre del centro educativo y del docente f) Introducción de la temporalización		

Código	REQ12	**Prioridad**	**Alta/Imprescindible**
Nombre	*Configuración de grupos o actividades*		
Descripción	En esta opción del menú podremos, de manera similar a como introducíamos los nombres de las notas, introducir los nombres o siglas de los grupos que tengamos ese curso; así como las actividades, por ejemplo, guardia, visita de padres, tutoría, etc..		

Código	REQ13	**Prioridad**	**Alta/Imprescindible**
Nombre	*Configuración de asignaturas*		
Descripción	Con esta opción podremos introducir el nombre, por medio de siglas, de las asignaturas que impartirá el docente ese curso.		

Código	REQ14	**Prioridad**	**Alta/Imprescindible**
Nombre	*Configuración de instrumentos de evaluación y ponderaciones*		
Descripción	*Esta será una de las opciones que dará el programa al pulsar el botón de menú. Podremos introducir por medio del teclado del terminal, la denominación de la nota que queremos que sea valorable (ej. Examen, actitud, libreta, ejercicios, trabajos,…) o simplemente informativa (negativo, positivo, no trae ejercicios,…). Una vez introducida nueva nota, ésta aparecerá en un menú junto a todas las notas existentes donde al pulsar sobre ella es posible configurar su peso. En esta pantalla se dará, por medio de un botón, la opción de introducir una nueva nota.*		

Código	REQ15	**Prioridad**	**Alta/Imprescindible**
Nombre	*Copias de seguridad (Backups)*		
Descripción	Una de las opciones que dará el botón menú será la opción de generar un archivo o carpeta de archivos que sirvan de copia de seguridad de los datos del programa. Otra de las opciones será la de restablecimiento de una copia de seguridad a partir de una anteriormente creada.		

Código	REQ16	**Prioridad**	**Alta/Imprescindible**
Nombre	*Usabilidad y robustez*		
Descripción	*Será obligatorio que la aplicación sea mínimamente robusta y no presente errores, reinicios ni bloqueos durante un uso "normal" de la misma en un dispositivo Android. Este uso normal incluye situaciones como: cambios de orientación de la pantalla del dispositivo, de horizontal a vertical y viceversa; paso de la aplicación a segundo plano por llamadas entrantes, alarmas o cualesquiera otras aplicaciones similares, etc. En todas ellas la aplicación deberá mantener su estado y continuar con la ejecución de manera normal siempre que sea posible, mostrando mensajes controlados de error cuando no lo sea.*		

9. Modelado de los casos de uso

9.1. Diagrama de caso de uso

Un diagrama de caso de uso son uno de los cinco tipos de diagrama UML, que se utilizan para el modelado de los aspectos dinámicos de un sistema (los otros cuatro tipos son los diagramas de actividades, de estados, de secuencia y de colaboración). Los diagramas de casos de uso son importantes para modelar el comportamiento de un sistema, un subsistema o una clase. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones.

Los diagramas de casos de uso se emplean para modelar la vista de casos de uso de un sistema. La mayoría de las veces, esto implica modelar el contexto del sistema, subsistema o clase, o el modelado de los requisitos de comportamiento de esos elementos.

En el caso que nos ocupa, el diagrama está compuesto por dos actores (profesor y Consejería), cuatro casos de uso, sus relaciones y el límite del sistema, representado por el cuadrado que engloba a los casos de uso.

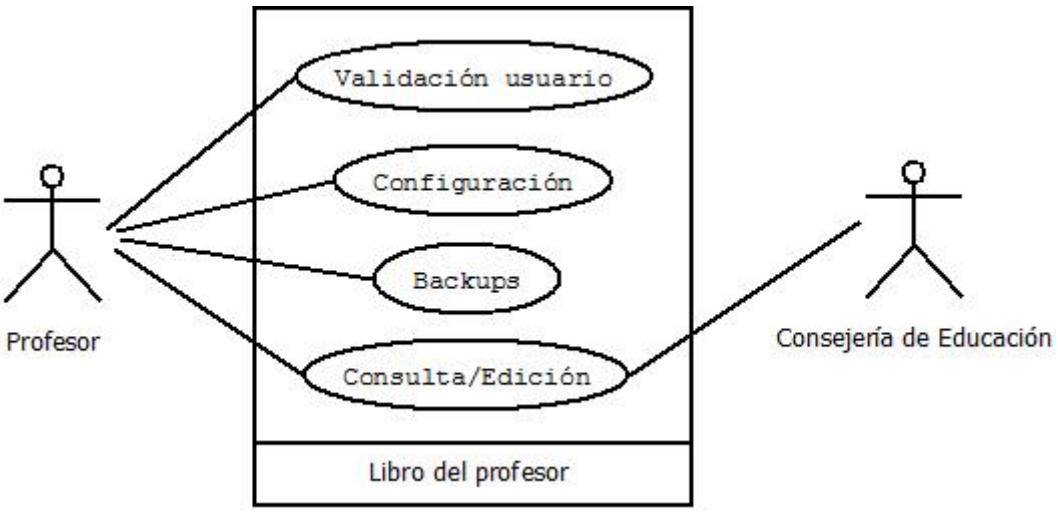

FIG. 27 – DIAGRAMA DE CONTEXTO

Pasemos a ver la organización de los casos de uso, en el siguiente subapartado.

9.2. Organización de casos de uso

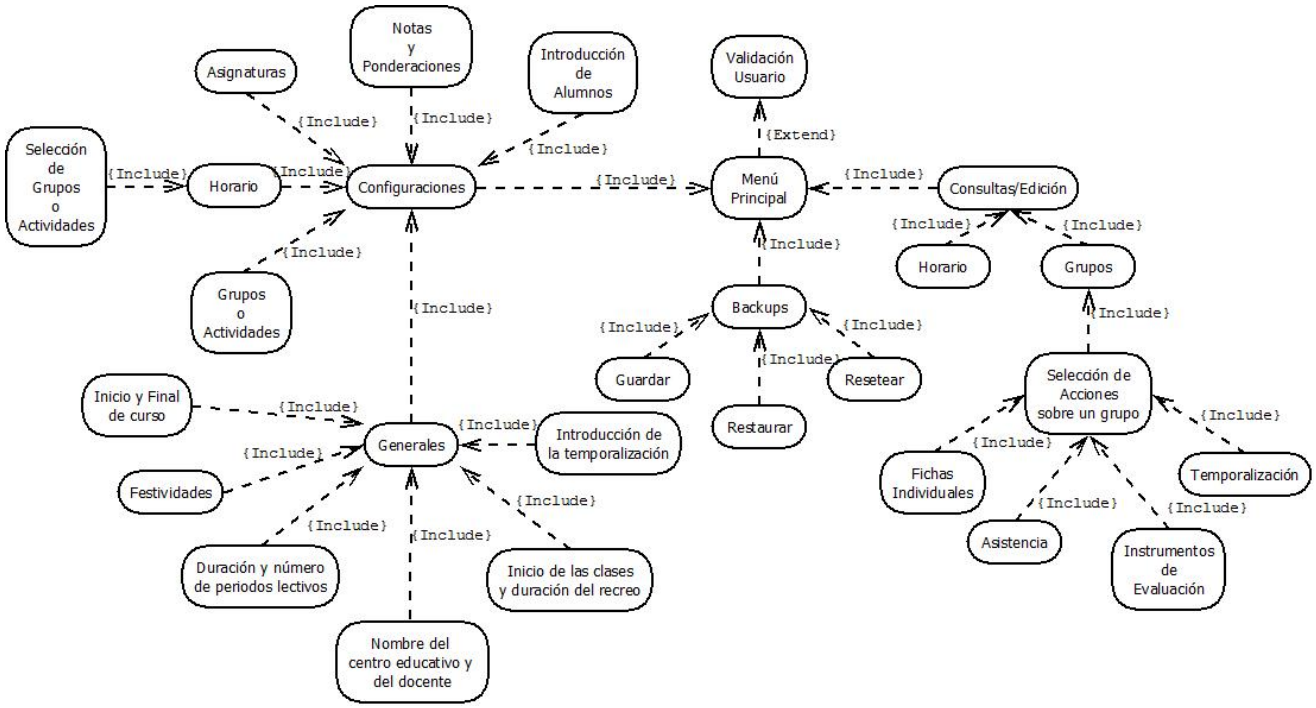

FIG. 28 – MODELADO DE CASOS DE USO

10. Casos de uso

Se denominan casos de uso a las descripciones de los pasos o actividades que se deben realizar para llevar a cabo un proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Decir, que es común a todos los casos de uso el cumplimiento del requisito 16 **(REQ16)**. Además, todas las pantallas que se presentan en el presente documento son explicativas y en ningún caso vinculantes, dejando de este modo libertad al diseñador gráfico y al programador para realizar su trabajo de la mejor forma posible.

10.1. Menú principal

Esta es la primera pantalla que aparece cuando abrimos el programa, previa ***validación (opcional) del usuario***, para desplegarlo **(REQ01)** debemos:

1. Pulsamos la tecla ***menú*** del terminal.
2. Aparece un menú con las siguientes opciones: "***Configuraciones generales***", "***Configuración de grupos o actividades***", "***Configuración de asignaturas***" "***Configuración de notas y ponderaciones***", "***Backups***" e "***Introducción de alumnos***".

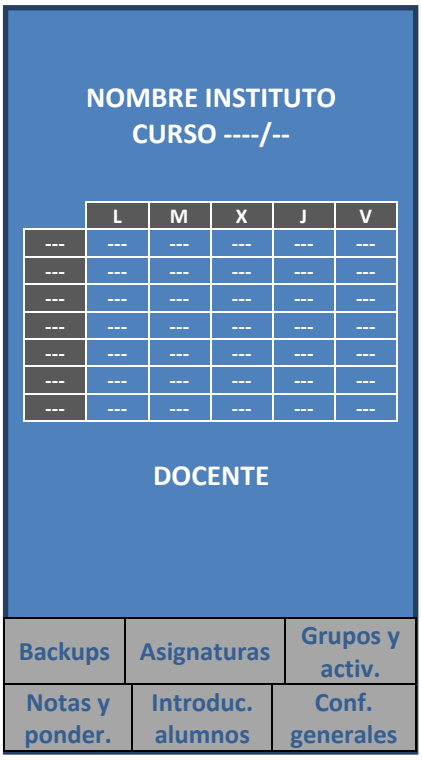

FIG. 29 –MENÚ PRINCIPAL

10.2. Configuraciones generales

Gracias a esta secuencia de pantallas podremos acceder al menú que permite realizar las configuraciones generales **(REQ11)** del programa:

1. Accedemos al ***menú principal***.
2. Pulsamos "***Conf. Generales***", aparece un nuevo submenú.

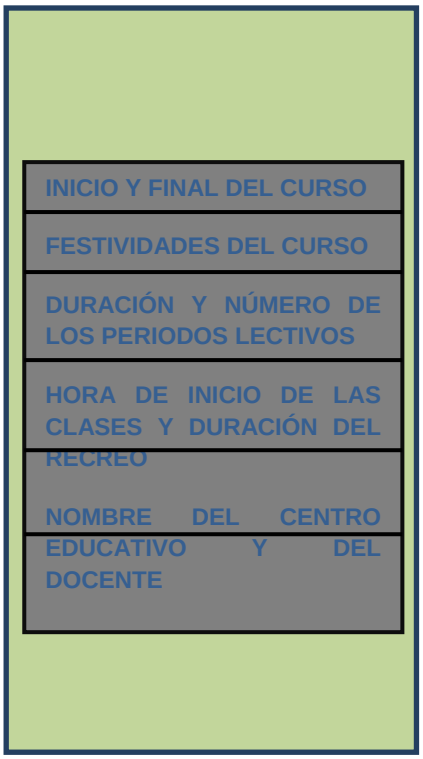

FIG. 30 – SUBMENÚ DE CONFIGURACIONES GENERALES

10.3. Inicio y final de curso

Gracias a esta secuencia de pantallas podremos configurar el inicio y final de curso **(REQ11)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "***Inicio y Final de curso***".

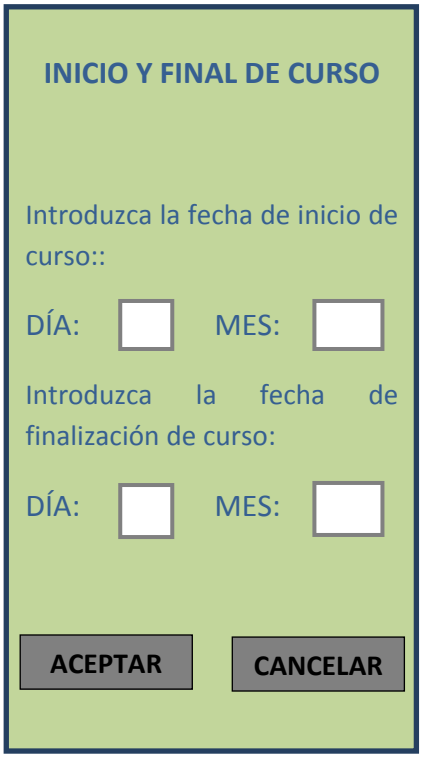

FIG. 31 – MENÚ DE INICIO Y FINAL DE CURSO

3. Introducimos los datos.
4. Pulsamos "ACEPTAR".

En el ***caso de uso 10.8*** se explicará de qué manera relacionaremos estos datos aquí introducidos con la temporalización

10.4. Festividades del curso

Gracias a esta secuencia de pantallas podremos configurar las festividades para el presente curso **(REQ11)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "***Festividades del curso***" (FIG. 32).
3. Introducimos los datos.
4. Pulsamos "SIGUIENTE" para introducir la siguiente festividad. Repetimos los pasos 3 y 4 hasta completar todas las festividades.
5. Pulsamos "ACEPTAR" y aparecerá una pantalla resumen con las festividades (FIG.33).

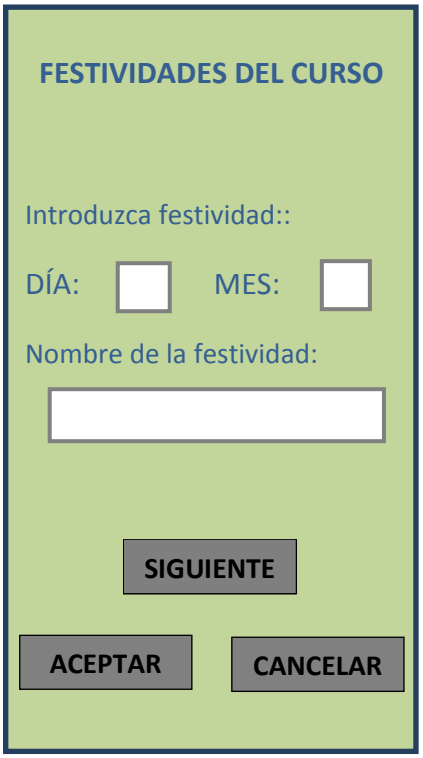

FIG. 32 – MENÚ DE DURACIÓN Y NÚMERO DE PERIODOS LECTIVOS

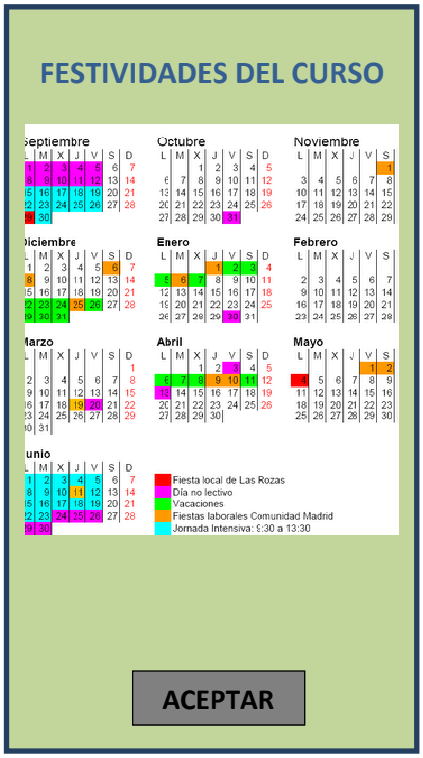

FIG. 33 – RESUMEN DE FESTIVIDADES

Será muy aconsejable que el programa sea capaz de introducir de manera automática aquellas festividades de carácter nacional y provincial, dejando de esta manera a cargo del profesor las de carácter local o las del centro.

En el ***caso de uso 10.8*** se explicará de qué manera relacionaremos estos datos aquí introducidos con la temporalización

10.5. Duración y número de los periodos lectivos

Gracias a esta secuencia de pantallas podremos configurar la duración y el número de los periodos lectivos para el presente curso **(REQ11)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "***Duración y número de los periodos lectivos***".

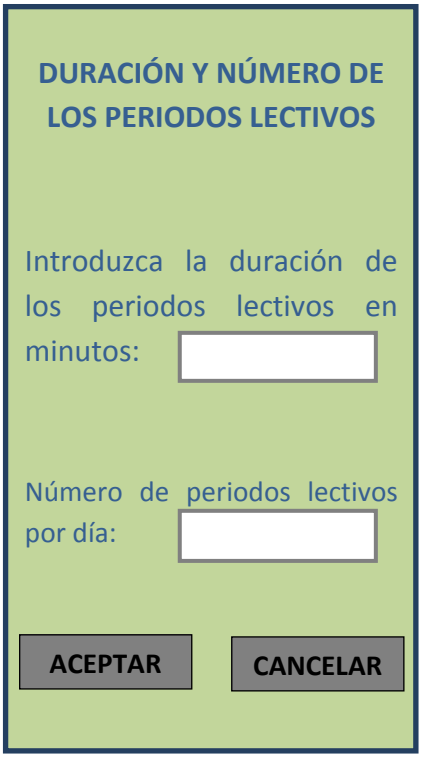

FIG. 34 – MENÚ DE DURACIÓN Y NÚMERO DE PERIODOS LECTIVOS

3. Introducimos los datos.
4. Pulsamos "ACEPTAR".

10.6. Inicio de las clases y duración del recreo

Gracias a esta secuencia de pantallas podremos configurar el inicio de las clases y la duración del recreo **(REQ11)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "***Inicio de las clases y duración del recreo***".

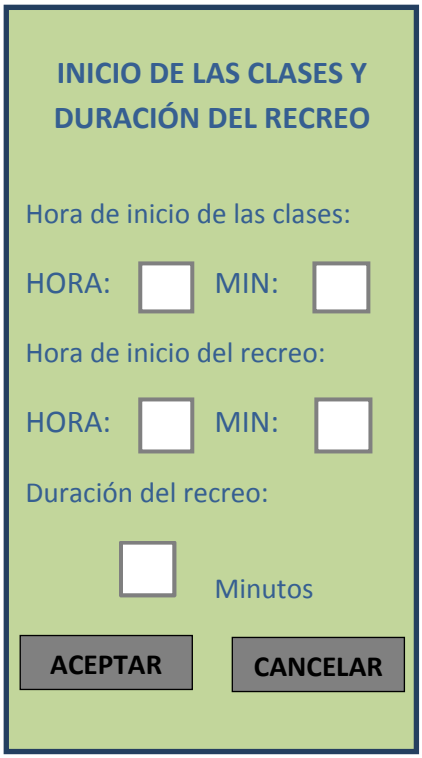

FIG. 35 – MENÚ DE INICIO DE LAS CLASES Y DURACIÓN DEL RECREO

3. Introducimos los datos.
4. Pulsamos "ACEPTAR".

10.7. Nombre del centro educativo y del docente

Gracias a esta secuencia de pantallas podremos configurar el nombre del centro educativo y del docente **(REQ11)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "**Nombre del centro educativo y del docente**".

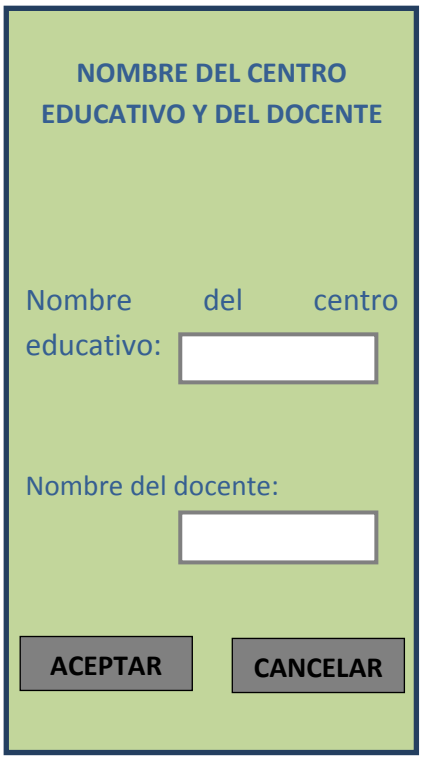

FIG. 36 – MENÚ DEL CENTRO EDUCATIVO Y DEL DOCENTE

3. Introducimos los datos.
4. Pulsamos "ACEPTAR".

10.8. Introducción de la temporalización

Gracias a esta secuencia de pantallas podremos configurar la secuencia de contenidos y duración para el curso **(REQ10)**.

1. Accedemos al menú de ***configuraciones generales***.
2. Seleccionamos "***Introducción de la temporalización***" (FIG. 37).
3. Introducimos los datos.
4. Pulsamos "SIGUIENTE" para introducir la siguiente actividad. Repetimos los pasos 3 y 4 hasta completar todas las actividades.
5. Pulsamos "ACEPTAR" y aparecerá una pantalla resumen (FIG.38).

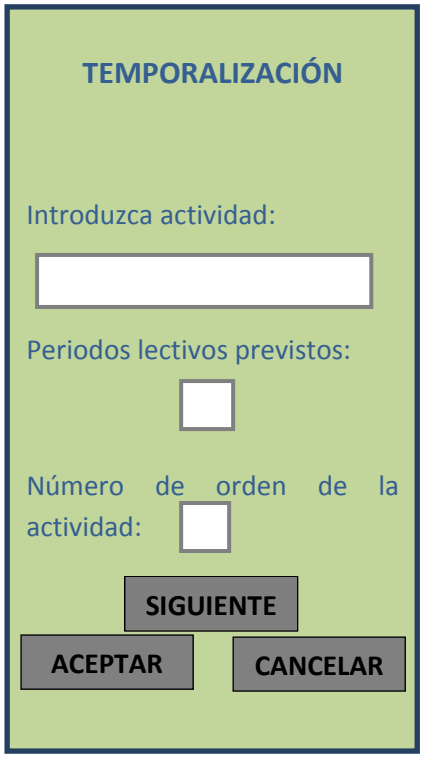

FIG. 37 – MENÚ PARA LA INTRODUCCIÓN DE LA TEMPORALIZACIÓN

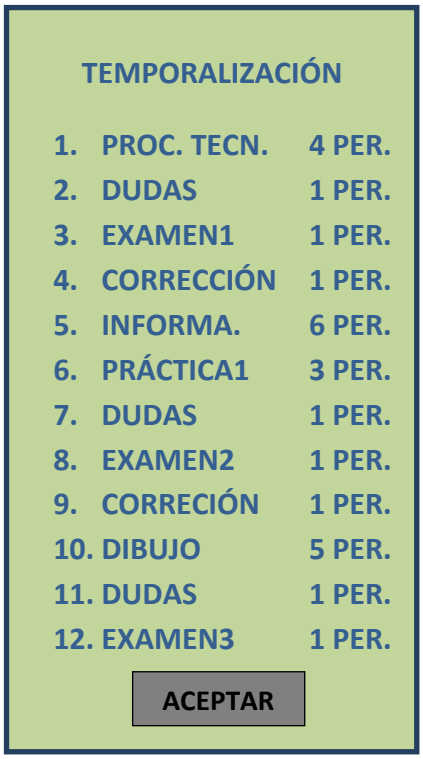

FIG. 38 – RESUMEN DE ACTIVIDADES

Decir, que la temporalización tendrá en cuenta todos los datos temporales que se han introducido en el programa, como son, las festividades, el inicio y final del curso o los días en los cuales cada grupo tiene clase, de manera que el programa podrá calcular con exactitud los días para cada una de las actividades.

Por ejemplo, un grupo tiene clase los lunes, martes y jueves, y la unidad didáctica 3 "Dibujo", ocupa 5 periodos lectivos. Imaginemos que el día que toca impartir la segunda sesión de la unidad es jueves y es fiesta, por tanto, el programa sabrá que la primera sesión será el martes 10 de octubre, la segunda sesión no será el jueves 12 de octubre, ya que es fiesta nacional en España, el día del Pilar, por tanto pasará automáticamente la segunda sesión para el lunes 16 de octubre, y seguirá temporalizando de esta manera.

10.9. Configuración de grupos o actividades

Gracias a esta secuencia de pantallas podremos acceder al menú que permite introducir los grupos de alumnos o las actividades **(REQ12)** que tiene el docente ese curso:

1. Accedemos al ***menú principal***.
2. Pulsamos "***Grupos/activ.***", aparece un nuevo submenú.

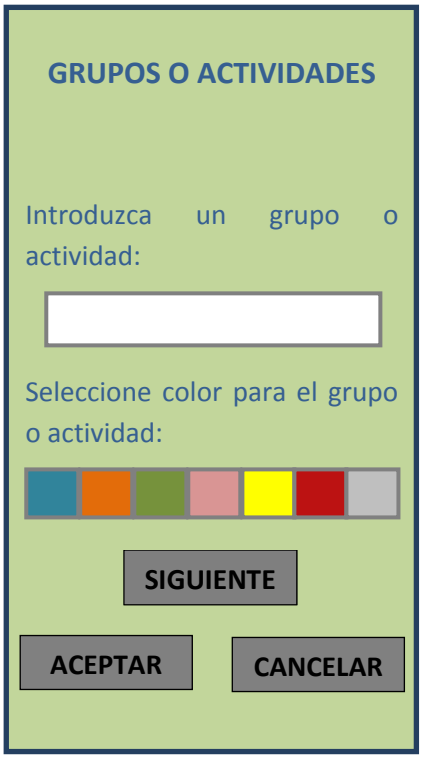

FIG. 39 – MENÚ PARA LA CONFIGURACIÓN DE GRUPOS O ACTIVIDADES

3. Introducimos los datos.
4. Pulsamos "SIGUIENTE" para introducir el siguiente grupo o actividad. Repetimos los pasos 3 y 4 hasta completar todos los grupos y actividades.
5. Pulsamos "ACEPTAR".

10.10. Configuración de asignaturas

Gracias a esta secuencia de pantallas podremos acceder al menú que permite introducir las asignaturas **(REQ13)** que tiene el docente ese curso:

1. Accedemos al ***menú principal***.
2. Pulsamos "***Asignaturas***", aparece un nuevo submenú.

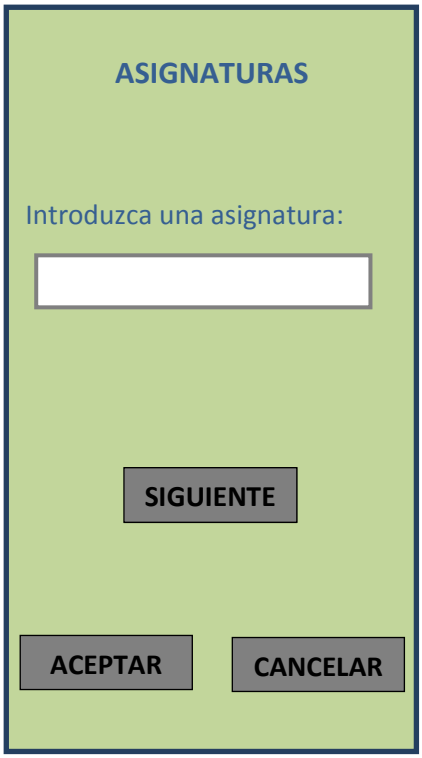

FIG. 40 – MENÚ PARA LA CONFIGURACIÓN DE ASIGNATURAS

3. Introducimos los datos.
4. Pulsamos "SIGUIENTE" para introducir la siguiente asignatura. Repetimos los pasos 3 y 4 hasta completar todas las asignaturas.
5. Pulsamos "ACEPTAR".

10.11. Configuraciones de notas y ponderaciones

Gracias a esta secuencia de pantallas podremos acceder al menú que permite introducir los instrumentos de evaluación, así como su peso en la nota final **(REQ14)**:

1. Accedemos al ***menú principal***.
2. Pulsamos "***Notas y ponder.***", aparece un nuevo submenú.

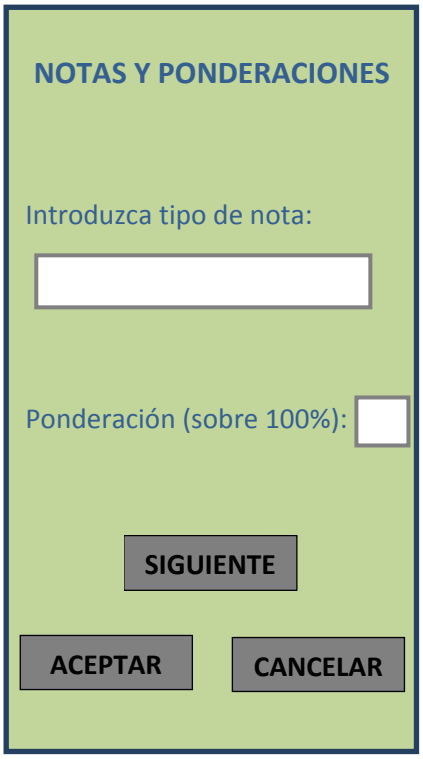

FIG. 41 – MENÚ PARA LA INTRODUCCIÓN DE NOTAS Y SUS PONDERACIONES

3. Introducimos los datos.
4. Pulsamos "SIGUIENTE" para introducir la siguiente nota y su ponderación. Repetimos los pasos 3 y 4 hasta completar todas las notas.
5. Pulsamos "ACEPTAR".

10.12. Horario

Esta es la pantalla principal de la aplicación, en ella se puede configurar y visualizar el horario de las clases del docente para el presente curso **(REQ01 y REQ02)**.

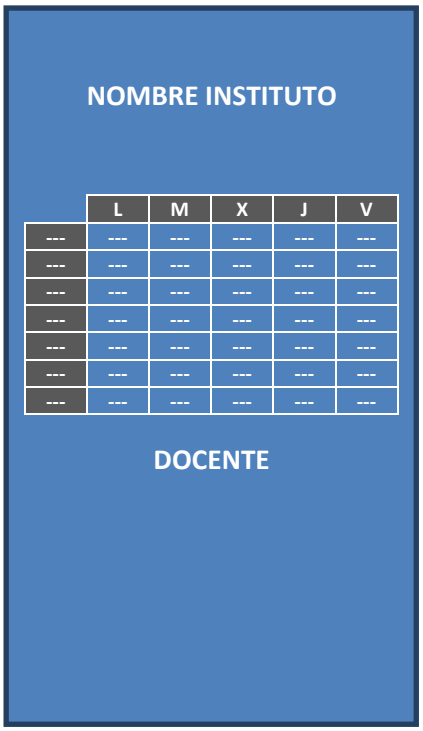

FIG. 42 – HORARIO EN BLANCO

FIG. 43 – HORARIO FINAL

Este es el aspecto que presenta el horario del docente la primera vez que abrimos la aplicación (FIG. 42). Para poder usarlo primero debemos configurarlo apropiadamente, siguiendo los siguientes pasos:

1. Abrimos el programa.
2. La tabla está en blanco.
3. Configuramos todas las opciones de las ***configuraciones generales***.
4. Configuramos los "***grupos de alumnos y actividades"*** que tiene el docente ese curso.
5. Hacemos una pulsación corta sobre la casilla en la que queremos seleccionar un grupo.
6. Seleccionamos uno de los grupos o actividades configurados del menú correspondiente (***Caso de uso 10.17***).
7. Se nos muestra el nuevo horario con la casilla modificada con el grupo que hemos seleccionado.
8. Repetimos los pasos 4 a 6 hasta completar todo nuestro horario.
9. Se nos muestra el horario terminado (FIG. 43).

En el ***caso de uso 10.8*** se explicará de qué manera relacionaremos estos datos aquí introducidos con la temporalización.

10.13. Introducción de alumnos

Gracias a esta opción podremos introducir los alumnos **(REQ07)** pertenecientes a cada grupo. Para que esto sea posible los alumnos deben estar en algún tipo de lista referenciada en el que cada alumno lleve asociado el grupo al que pertenece y el orden dentro de este.

Estas listas deben poder introducirse al terminal en forma de archivo creado en el ordenador del usuario, archivo dado por el centro de estudios o, como método más aconsejable, se debería poder descargar las listas de alumnos de la página web de la Consejería de Educación correspondiente **(REQ08)**. El método de introducción de esta información queda a cargo del programador.

Un a vez tengamos el archivo con las listas en el terminal, deben poder acoplarse al programa, por el siguiente procedimiento:

1. Accedemos al ***menú principal***.
2. Pulsamos "***Introducción de alumnos***".
3. El programa pedirá que introduzcamos la ubicación del archivo.
4. Las listas se incluirán en el programa.

Flujo alternativo 1:

4. El programa busca el archivo en la ubicación dada y no lo encuentra.
5. Aparece el mensaje "No se encuentra archivo".

10.14. Backups

Esta opción nos permitirá realizar una copia de seguridad de todos los datos incluidos en el programa **(REQ15)**, así como de restaurar una copia anterior o hacer un reseteo.

1. Accedemos al ***menú principal***.
2. Pulsamos "***Backups***".
3. El programa dará tres opciones.
    a. Realizar copia de seguridad
    b. Restaurar copia de seguridad.
    c. Resetear la copia de seguridad

Flujo alternativo 1:

4. Pulsamos ***Realizar copia de seguridad***.
5. El programa realiza un Backup de manera automática, guardando el archivo en una ubicación predeterminada. Si existe un archivo anterior se elimina.
6. Una vez terminada la copia de seguridad aparece el mensaje "Backup realizado".

Flujo alternativo 2:

4. Pulsamos ***Restaurar copia de seguridad***.
5. El programa busca en la ubicación predeterminada un archivo de backup creado anteriormente y lo incorpora al programa.
6. Una vez terminada la restauración aparece el mensaje "Backup restaurado".

Flujo alternativo 3:

4. Pulsamos ***Restaurar copia de seguridad***.
5. El programa busca en la ubicación predeterminada un archivo de backup creado anteriormente y no lo encuentra.
6. Aparece el mensaje "No se encuentra archivo de restauración".

10.15. Selección de acciones sobre un grupo

Esta es una pantalla sencilla que simplemente nos permite seleccionar entre las posibles acciones a realizar con un grupo seleccionado **(REQ03)**; a saber, ***"Fichas Individuales", "Control de asistencia", "Introducir Instrumentos de evaluación"*** o ***"Temporalización"***.

FIG. 44 – SELECCIÓN DE ACCIONES

La opción de ***fichas individuales***, nos llevará al listado de alumnos del grupo, donde podremos seleccionar el alumno del cual queremos consultar su ficha personal.

La opción de ***asistencia*** desplegará el listado del grupo, donde podremos seleccionar los alumnos que han faltado a la presente clase, el dispositivo tomará la fecha de configuración del terminal.

Al seleccionar ***instrumentos de evaluación***, podremos seleccionar de entre los tipos de instrumentos que hayamos definido previamente (***caso de uso 10.16***), aquel que vamos a introducir. Tras esto, veremos el listado de alumnos donde podremos calificar dicho instrumento, si es calificable, o seleccionar los alumnos que queremos cuenten con el instrumento seleccionado, si no es calificable; por ejemplo, negativo, positivo, no tareas…

Por último tendremos la opción de ver cuál es la ***temporalización*** que estamos siguiendo con dicho curso, que actividades tocan para el día presente y que tendremos que hacer en los días próximos.

10.16. Selección de los instrumentos de evaluación

Esta pantalla aparece cuando seleccionamos "***Instrumentos de evaluación***" en la pantalla de "***Selección de acciones sobre un grupo***" (***Caso de uso 10.15***).

La pantalla de selección de instrumentos de evaluación **(REQ06)** nos dará la opción de seleccionar una de entre las posibilidades existentes, configuradas en "***Notas y ponderaciones***" (***Caso de uso 10.11***). Posteriormente se evaluará este instrumento en la pantalla de introducción de calificaciones.

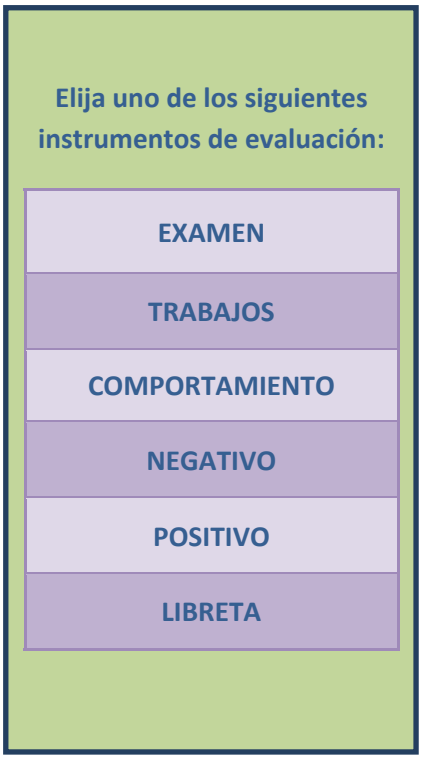

FIG. 45 – SELECCIÓN DE INSTRUMENTOS DE EVALUACIÓN

Flujo alternativo 1:

Al intentar acceder no hemos introducido las ***notas y ponderaciones***, el programa presentará el mensaje "No existen instrumentos de evaluación".

10.17. Selección de grupos y actividades

Esta pantalla **(REQ12)** aparece cuando hacemos una pulsación larga sobre una de las casillas del horario cuando lo estamos configurando (***Caso de uso 10.12***).

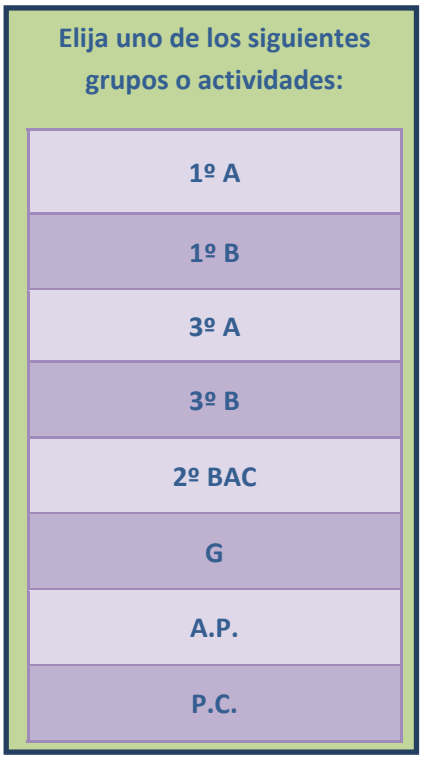

FIG. 46 – SELECCIÓN DE GRUPOS Y ACTIVIDADES

Estos grupos y actividades han debido introducirse anteriormente por medio del ***caso de uso 10.9***.

Flujo alternativo 1:

Al intentar configurar el horario no hemos introducido los ***grupos y las actividades***, el programa presentará el mensaje “No existen grupos o actividades”.

11. Conclusiones

En este documento se han definido los requisitos para la implementación de un ***libro del profesor***. Se han especificado los requisitos de una aplicación que debe permitir al profesor prescindir de los grandes cuadernos de registro, incómodos y poco manejables, aprovechando las ventajas que proporcionan las nuevas tecnologías móviles.

Tal y como se explicó en el capítulo 3 se ha tratado de que la especificación sea completa, consistente e independiente de la implementación, a la par que precisa y verificable. En este sentido, habría que comentar, que la orientación a plataformas Android no es una necesidad absoluta, sino que viene más bien dada por la disponibilidad y menor coste de estos terminales. En cualquier caso, la especificación proporcionada no depende en absoluto de la tecnología de implementación.

En la especificación se ha tratado de utilizar únicamente diagramas sencillos, pero expresivos, procurando en todo momento que fueran claros y precisos. Debe tenerse en cuenta que la especificación está hecha desde el punto de vista del usuario final, que no es, ni tiene porqué ser un experto en ingeniería del software. Aun así, la especificación creemos que es útil para aquellos que sí lo son, constituyendo de esta manera un documento válido tanto para la implementación como para la validación de la aplicación que se describe.

Del mismo modo, se ha tratado que la estructura del documento facilite su revisión, modificación y extensión. Se ha tratado también de referenciar los diferentes elementos para que puedan ser trazados a lo largo del documento.

La aplicación descrita admite, por supuesto, múltiples extensiones, que pueden incorporarse con facilidad al documento. Entre estas extensiones cabe destacar la adición de una conexión, o incluso una sincronización, con blog del profesor.

Bibliografía

[Android 12] Android Developers
http://developer.android.com/training/index.html

[Booch 99] Booch, G., Rumbaugh, J. and Jacobson, I., "El Lenguaje Unificado de Modelado", Addison Wesley Iberoamericana, Madrid, 1999.

[Brooks 87] Brooks, F., "No Silver Bullet: Essence and Accidents of Software Engineering", IEEE Computer, April 1987, pp. 10-19.

[Celma 06] Matilde Celma, "Bases de datos relacionales", Pearson Education, 2006

[Coll 08] COLL, C., MAURI, T. y ONRUBIA, J. (2008). Análisis de los usos reales de las TIC en contextos educativos formales: una aproximación sociocultural. Revista Electrónica de Investigación Educativa, 10 (1). http ://redie.uabc.mx/vol10no1/contenido-coll2.html.

[Davis 93] Davis, A, "Software Requirements (Revised): Objects, Functions, and States", Prentice Hall, New Jersey, 1993.

[De Pablo 92] De Pablo, P. y cols., "Diseño del currículo en el aula. Una propuesta de autoformación", Col. Forum Didáctico de Mare Nostrum Ediciones, Madrid, 1992.

[Faulk 92] Faulk, S., J. Brackett, P. Ward, and J. Kirby, Jr., "The Core Method for Real-Time Requirements", IEEE Software, Vol. 9, No.5, September 1992.

[Faulk 95] Faulk, Stuart R., “Software Requirements: A Tutorial”, Space and Naval Warefare Systems Command, November 1995.

[Herr 11] Herramientas TIC http://c4lpt.co.uk/top-100-tools-for-learning-2011/

[Pamas 86] Pamas, D., and P. Clements, “Rational Design Process: How and Why to Fake It", IEEE Transactions on Software Engineering, v. 12, no. 2, February 1986, pp. 251-257.

[Wei 11] Wei-Meng Lee, “Beginning Android Application Development”, Wiley Publishing Inc, ISBN 978-1-118-01711-1