2012

Proyecto Final de Carrera Ingeniería Informática

Aplicación web para dispositivos móviles (PDA)

Javier Hernández Higueras

Director
José Vicente Busquets Mataix

PALABRAS CLAVE

ANDROID: SISTEMA OPERATIVO MÓVIL PROPIEDAD DE GOOGLE.

APACHE: SERVIDOR HTTP DE CÓDIGO ABIERTO.

ECLIPSE: ENTORNO DE DESARROLLO INTEGRADO DE CÓDIGO ABIERTO Y MULTIPLATAFORMA

HTML: DOCUMENTO DE HIPERTEXTO EMPLEADO PARA LA VISUALIZACIÓN DE PÁGINAS WEB

JAVA: LENGUAJE DE DESARROLLO

MYSQL: SISTEMA DE GESTIÓN DE BASES DE DATOS.

PHP: LENGUAJE DE PROGRAMACIÓN INTERPRETADO

SMARTPHONE: TELÉFONO MÓVIL CON FUNCIONALIDADES AÑADIDAS.

CONTENIDO

1. INTRODUCCIÓN

Una aplicación para la gestión de asistencias es una herramienta que permite el control de la asistencia de estudiantes a diversas clases por el profesor o encargado, así como la gestión de otros factores como las notas, los datos de los estudiantes, incidencias relacionadas con dichos estudiantes, etc. La aplicación está destinada a dispositivos móviles que hagan uso del sistema operativo Android, centrado principalmente en smartphones.

Las acciones que pueden realizarse mediante el uso de esta aplicación consisten en controlar la asistencia de los alumnos a un grupo específico en el que esté matriculado, asignar notas a esos alumnos, así como crear nuevas notas asociadas a un grupo para poder calificar a los alumnos. También es posible notificar incidencias. Además de esto, se podrá consultar información relacionada con los alumnos, viendo sus datos personales, sus notas, asistencias y demás información introducida y tratada.

Por otro lado, existe un apartado web destinado a la gestión de la base de datos mediante el cual se pueden realizar diversos cambios con el fin de poder completar la funcionalidad que ofrece la aplicación Android. De esta forma, se pueden introducir datos y modificarlos.

1.1. MOTIVACIÓN

Existen diversos motivos por los que se decidió realizar y desarrollar este proyecto en concreto.

El primero de ellos está relacionado con el auge y popularidad actual de la tecnología a la que está destinada la aplicación, es decir, el éxito de los smartphones y las posibilidades que ofrecen. De esta forma, se han podido adquirir conocimientos del desarrollo destinado a estas plataformas en oposición a otras tecnologías como los ordenadores de sobremesa o las páginas web.

Otro de los motivos consiste en el sistema operativo al que va dirigida la aplicación, en este caso Android, un sistema operativo relativamente nuevo y cada vez más extendido en dispositivos móviles, tanto smartphones como tablets, con una cuota de mercado cada vez mayor. Además es un sistema operativo de código abierto.

Por último, la posibilidad de desarrollar una aplicación para dispositivos móviles es un ámbito que no se estudia ampliamente a lo largo de la ingeniería por lo que era una buena oportunidad para obtener conocimiento en dicho ámbito y poder desarrollar una aplicación acorde.

1.2. PLANTEAMIENTO TÉCNICO

En cuanto al planteamiento técnico, la idea principal es realizar una aplicación sencilla, intuitiva y lo más cómoda posible ya que el hecho de pasar lista podría ser algo tedioso y largo en caso de recargar la interfaz gráfica o de complicar en exceso el sistema. Además, debe ser posible realizar otras funciones relacionadas con el ámbito educativo, de forma que se facilite al usuario la tarea de informatizar datos.

En cuanto a los requisitos necesarios para el usuario tan solo es necesario disponer de un dispositivo móvil Android, instalar la aplicación y disponer de conexión a Internet. Por otro lado, para el almacenamiento de los datos será necesario un servidor, independiente del sistema operativo que emplee, en el que almacenar la base de datos. También debe ser capaz de interpretar PHP.

Para el desarrollo del proyecto se ha empleado un equipo con el sistema operativo Windows XP, trabajando con el entorno de desarrollo Eclipse y el kit de desarrollo de software, o SDK, de Android. Para la base de datos se ha empleado la aplicación XAMPP que proporciona un servidor web Apache, un sistema de gestión de bases de datos MySQL e intérpretes para PHP y Perl, aunque solo se ha empleado el primero de estos lenguajes.

2. ESPECIFICACIÓN DE REQUISITOS

A continuación se detalla una especificación de requisitos referentes a la aplicación.

2.1. INTRODUCCIÓN

En primer lugar es necesario definir algunas características del proyecto, tales como el ámbito, una visión global y una serie de definiciones que ayudarán a la lectura de la propia especificación.

2.1.1. PROPÓSITO

El propósito del presente apartado es definir los requerimientos que debe tener y cumplir la aplicación desarrollada. Esta especificación de requisitos tiene como objetivo formalizar las funcionalidades y prerrequisitos de forma que haya una base con la que contrastar el desarrollo de la aplicación, así poder realizar el desarrollo de una forma más sencilla y guiada.

2.1.2. ÁMBITO

La aplicación consistirá principalmente en una aplicación destinada a dispositivos móviles Android mediante la cual se realizarán todas y cada una de las distintas acciones posibles, ya sea asignar asistencias como consultar los datos asociados a un alumno.

Dentro de las características que implementa la aplicación, el usuario podrá realizar diferentes operaciones como asignar faltas de asistencia a un alumno en un grupo específico, crear incidencias, crear notas asociadas a un grupo y a su vez asignar notas a los alumnos de ese grupo y consultar los datos de cualquiera de los alumnos.

La funcionalidad principal podría resumirse en permitir la gestión de una serie de características relacionadas con el ámbito académico de una forma cómoda y sencilla mediante un dispositivo móvil de forma que no se dependa de un ordenador de sobremesa para el acceso o la modificación de dicha información.

Cabe mencionar que esta aplicación se complementa con otro proyecto consistente en el acceso mediante web a diversos datos y funcionalidades, de esta forma se permite el acceso a los datos a los padres de los alumnos. Los profesores también pueden tener acceso mediante la web a los datos necesarios. Así pues, mediante la aplicación de dispositivos móviles y la parte web se consigue mayor número de usuarios, ya no solo profesores, sino también padres, y además se consigue que el acceso a los datos pueda realizarse desde un ordenador, sin necesidad de disponer de un dispositivo móvil.

2.1.3. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

- Android: Es el sistema operativo de Google destinado a dispositivos móviles, smartphones y tablets. Lanzado en septiembre del año 2008 ha revolucionado el mundo de los smartphones debido a la cantidad de dispositivos que hacen uso de Android. Las últimas cifras apuntan a más de 225 mil millones de dispositivos con Android y dispone ya de más de 300.000 aplicaciones disponibles.

- Apache: Es un software libre. Apache es un servidor HTTP de código abierto para plataformas Linux, Windows y Mac OS X entre otros, desarrollado por Apache Software Foundation.

- MySQL: Es un sistema de base de datos relacional, multihilo y multiusuario. Desarrollado inicialmente por MySQL AB, posteriormente subsidiaria de Sun Microsystems, la cual es actualmente subsidiaria de Oracle Corporation. Actualmente se desarrolla como software libre bajo un licenciamiento dual, mediante la cual es posible distribuir un producto con licencia GNU GPL o adquirir una licencia comercial para distribución bajo otra licencia.

- PHP: Es un lenguaje de programación interpretado, en otras palabras, es el servidor el encargado de hacerlo funcionar de forma que el cliente tan solo recibe una respuesta sin necesidad de ejecutar código alguno. El nombre PHP es un acrónimo recursivo cuyo significado es PHP Hypertext Preprocessor.

- Smartphone: Es el término empleado para denominar a un teléfono móvil con más funcionalidades que un teléfono móvil común.

2.1.4. REFERENCIAS

Para la realización de esta especificación de requisitos se han tenido en cuenta los siguientes documentos:

- IEE Std. 830-1998. IEEE Recommended Practice for Software Requirements Specifications.
- Buendía García Félix.
  Una guía para la realización y supervisión de proyectos final de carrera en el ámbito web.
  Editorial UPV, 2008.
  ISBN 978-84-8363-325-0.

2.1.5. VISIÓN GLOBAL

En el resto del documento se tratará con detalle las características del proyecto a desarrollar, mostrando los objetivos y características marcadas que la aplicación final debe cumplir.

2.2. DESCRIPCIÓN GENERAL

A continuación se detallan una serie de apartados que describen el objetivo del producto, así como su funcionalidad y requisitos.

2.2.1. PERSPECTIVA DEL PRODUCTO

El desarrollo de la aplicación se puede dividir en dos partes, por un lado la aplicación Android destinada a dispositivos móviles y por otro lado, la parte del servidor en el que se incluye la base de datos y las diferentes funciones que la aplicación Android utiliza para obtener la información necesaria.

Para la aplicación Android se emplea el SDK Android utilizando el lenguaje de programación Java. En cuanto al lado del servidor, se utilizará el servidor Apache y una base de datos MySQL debido a la alta compatibilidad entre estos dos sistemas. Por otro lado, para la realización de las funciones que empleará la aplicación Android se empleará PHP para su desarrollo.

En cuanto a la aplicación Android en sí, el objetivo es desarrollar una aplicación de fácil acceso y aprendizaje, mostrando todas las opciones disponibles de la forma más sencilla. La presentación de la interfaz debe mostrarse igual independientemente de la resolución del terminal usado, por otro lado también deberá de ser compatible con el mayor número de versiones posibles del sistema operativo Android.

Para el desarrollo de la aplicación se hará uso del sistema operativo Microsoft Windows XP, del software gratuito XAMPP que proporciona un servidor Apache, un gestor de base de datos MySQL e intérprete para PHP. Además se empleará el entorno de desarrollo integrado Eclipse con el SDK Android.

2.2.2.FUNCIONES DEL PRODUCTO

Las funciones principales que la aplicación debe permitir son las siguientes:

1) Gestión de asistencia
    a) Añadir falta a un alumno
    b) Eliminar falta de un alumno

2) Gestión de notas
    a) Crear nueva nota
    b) Asignar notas a un alumno

3) Gestión de incidencias
    a) Crear incidencia de un alumno

4) Consultar datos de un alumno
    a) Consultar nombre, apellidos y otros datos
    b) Consultar los datos del tutor asignado
    c) Consultar asistencias de un alumno
    d) Consultar notas de un alumno
    e) Consultar incidencias de un alumno

2.2.3. CARACTERÍSTICAS DEL USUARIO

La aplicación debe poder ser manejada por un único usuario, el usuario principal que serán los profesores. Este usuario debe ser capaz de realizar todas las operaciones y funciones comentadas en el apartado anterior, de forma que pueda tanto gestionar las asistencias, como las notas o incidencias. Un detalle importante a tener en cuenta es que los profesores solo pueden acceder a los datos de sus alumnos, es decir, de aquellos alumnos que cursen una clase que imparta el profesor, con tal de mantener cierto nivel de privacidad. Por este motivo existen usuarios especiales, como el director del centro o el jefe de estudios, que además de tener el resto de funciones no disponen de limitaciones al consultar los datos de los alumnos.

2.2.4. RESTRICCIONES GENERALES

En cuanto a las restricciones que tiene el proyecto se puede mencionar la necesidad de trabajar con una base de datos con una estructura preestablecida. En caso de modificarse dicha estructura o eliminarse parcialmente, la aplicación funcionaría de manera incorrecta y no deseada o incluso podría dejar de ser funcional. En caso de tener que modificar la estructura de la base de datos, sería necesario realizar correcciones en la aplicación y/o en las funciones utilizadas para el acceso a la base de datos.

A la hora de emplear la aplicación hay que tener en cuenta que es necesario disponer de acceso a internet. Esto puede realizarse mediante conexión Wi-Fi a la red del centro en el que se use la aplicación o mediante conexiones móviles 3G. Hay que tener en cuenta que la velocidad con la que la aplicación responda a las peticiones dependerá tanto de la velocidad de la conexión como del rendimiento del servidor, así pues, siempre será mejor disponer de una conexión de alta velocidad y un buen servidor. En el caso ideal sería óptimo disponer del servidor en el mismo centro de forma que el acceso se realizase en la red local y por tanto se disminuyese el tiempo de respuesta de las peticiones a dicho servidor.

2.2.5.SUPUESTOS Y DEPENDENCIAS

En cuanto a las dependencias cabe mencionar que la aplicación funciona únicamente en dispositivos móviles Android. En cuanto a la base de datos y las funciones utilizadas para acceder a ella son independientes del servidor, por lo que podría alojarse en un servidor con cualquier sistema operativo siempre y cuando sea posible ejecutar Apache, interpretar PHP y alojar una base de datos MySQL.

2.3. REQUISITOS ESPECIFICOS

En el siguiente apartado se detallan los requisitos de la aplicación, tanto a nivel funcional como de interfaz, eficiencia y diseño.

2.3.1.REQUISITOS FUNCIONALES

- Asignar falta de asistencia: El usuario seleccionará el grupo en el que quiere pasar asistencia. A continuación se mostrará la lista de alumnos pertenecientes a ese grupo y el usuario tendrá la posibilidad de desmarcar la casilla asociada al alumno para indicar que el alumno ha faltado y por tanto se le debe asignar una falta de asistencia. Además, se enviará al tutor del alumno un correo electrónico informándole de la falta de asistencia.

- Eliminar falta de asistencia: El usuario seleccionará el grupo al que pertenece el alumno al que se le quiere eliminar la falta de asistencia. Una vez se muestre la lista de alumnos de ese grupo se buscará al alumno y se volverá a marcar la casilla para eliminar la falta de asistencia.

- Crear nota: El usuario seleccionará un grupo de la lista de grupos y a continuación indicará el nombre de la nota que se quiere crear

- Asignar nota: El usuario seleccionará el grupo y la nota que quiere asignar. A continuación se le mostrará una lista de alumnos pertenecientes al grupo seleccionado y tendrá la posibilidad de asignar una nota a cada alumno del grupo.

- Crear incidencia: El usuario buscará el alumno al que quiere asociar la asistencia. Una vez encontrado lo seleccionará y podrá añadir el asunto, la gravedad de la incidencia y el motivo por el que se ha creado la incidencia.

- Consultar datos de un alumno: El usuario buscará el alumno del que quiere obtener los datos y lo seleccionará. A continuación se le mostrarán los datos del alumno seleccionado. También podrán consultarse los datos del tutor de dicho alumno, así como las incidencias que tenga. Se podrá consultar también los grupos en los que está matriculado, y seleccionando uno de los grupos se mostrará la información relativa a ese grupo en referencia a las faltas de asistencia y las notas del alumno seleccionado en el grupo elegido.

2.3.2. REQUISITOS DE INTERFAZ

Con respecto a los requisitos de interfaz podemos diferenciar tres tipos: interfaz de usuario, interfaz software e interfaz hardware.

En cuanto a la interfaz de usuario, el principal objetivo al desarrollarla es conseguir una interfaz sencilla de manejar. Puesto que la aplicación es una aplicación destinada a móviles y con diversas funciones no existe un patrón básico que compartan las distintas interfaces. Aún así, debido al carácter de la aplicación y su funcionalidad orientada al uso educativo y por tanto la necesidad del uso de listas, la interfaz más común es la empleada por las funciones de asistencia y notas, las cuales consisten en una lista de alumnos, con la posibilidad de asignar una falta o una nota mediante un botón o la introducción de texto respectivamente. Por otro lado, a la hora de crear una incidencia la interfaz consiste en diversos campos a rellenar. Por último, para la funcionalidad de consultar datos, se dispone de una barra superior, dividida en pestañas, mediante las cuales es posible acceder a las distintas secciones.

En referencia a la interfaz software, el proyecto se desarrolla empleando el sistema operativo Microsoft Windows XP, empleando el entorno de desarrollo Eclipse y Apache como servidor. Java es el lenguaje principal puesto que Android hace uso de dicho lenguaje, pero también se emplea PHP. Por último para el sistema de gestión de base de datos se emplea MySQL.

Como requisito o interfaz hardware es importante mencionar la necesidad de emplear un dispositivo móvil con el sistema operativo Android y conexión a Internet y un servidor con soporte para las tecnologías PHP y MySQL.

2.3.3. REQUISITOS DE EFICIENCIA

El uso de la aplicación se realiza de forma individual, usando cada usuario su dispositivo móvil pero esto no evita que puedan existir varios usuarios accediendo de forma concurrente a los datos almacenados. Este servicio lo proporciona el propio servidor, de forma que se permita que varios usuarios accedan al servidor y por tanto a la base de datos. Se puede decir entonces que la eficiencia de la aplicación en el lado del servidor dependerá de éste.

Por otro lado, puesto que la aplicación se ejecuta en un dispositivo móvil la eficiencia de la propia aplicación dependerá de los requisitos de éste, a pesar de esto, cualquier terminal capaz de disponer del sistema operativo Android debería ser capaz de ejecutar la aplicación de forma eficiente.

2.3.4. RESTRICCIONES DE DISEÑO

A diferencia de una página web en la que sería adecuado seguir los estándares marcados por el W3C, a la hora de desarrollar una aplicación móvil para dispositivos móviles no existen unos estándares básicos que seguir. Aún así se han seguido en la medida de lo posible una serie de 'best practices' o mejores prácticas sugeridas por la guía de desarrolladores Android ofrecida en su página web, mediante la cual se ofrece información por ejemplo de qué hacer y cómo realizarlo para adaptar la interfaz a distintos tamaños.

2.3.5. ATRIBUTOS

La aplicación creada no debería necesitar mantenimiento puesto que funciona mediante los datos almacenados en la base de datos, así pues, el mantenimiento de los datos recae sobre el usuario ya que será el encargado de actualizarlos o modificarlos mediante el uso de la aplicación.

Por otro lado, ya que la aplicación hace uso de un sistema de acceso de usuarios, para poder acceder será necesario disponer de un nombre de usuario y una contraseña.

2.3.6.OTROS REQUISITOS

La aplicación hace uso de una base de datos MySQL. En esta base de datos se almacenará toda la información requerida por la aplicación, tanto datos de los alumnos como de los profesores para poder acceder, incluyendo datos de las asignaturas, notas, etc. Como ya se ha comentado en anteriores apartados, el acceso se realiza mediante diversas funciones en PHP puesto que existe un alto grado de compatibilidad entre dicho lenguaje y la base de datos utilizada.

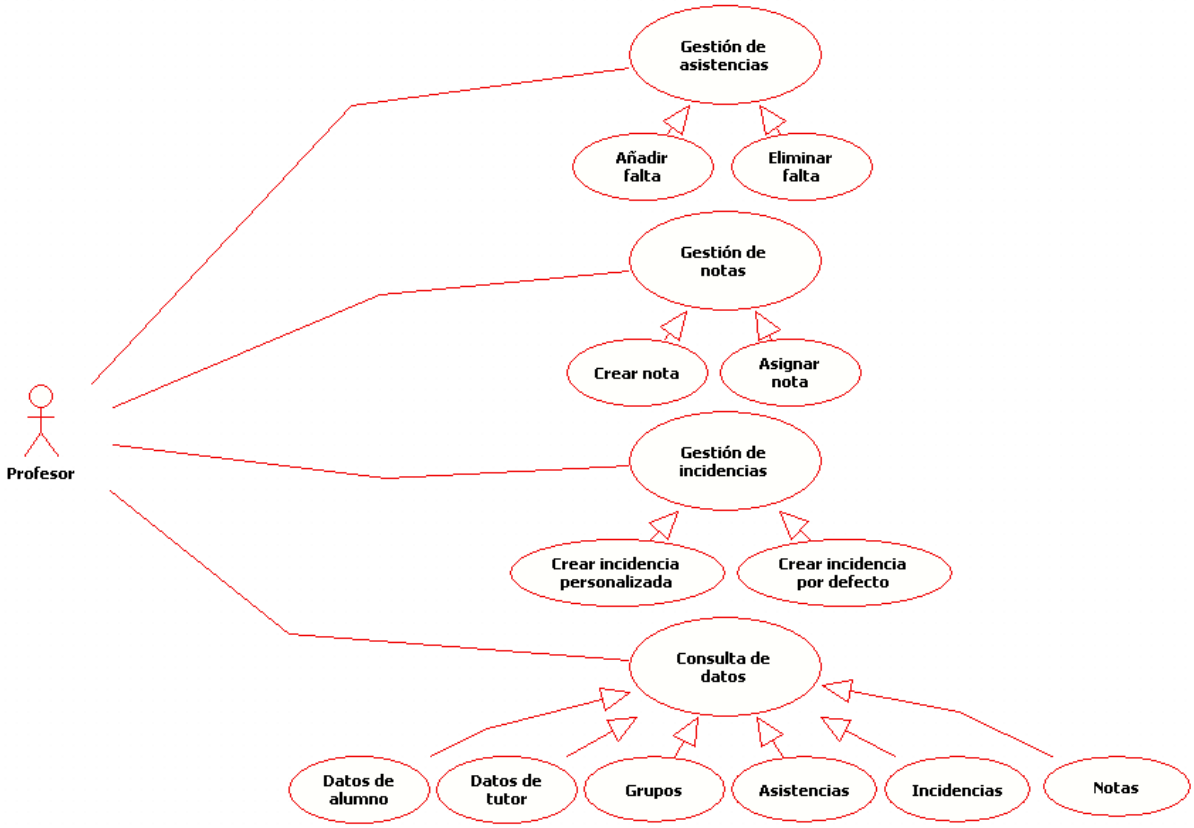
Dentro de los requisitos para la aplicación, es esencial que la aplicación funcione en la mayoría de dispositivos Android posibles. Por este motivo se establece como requisito que la aplicación funcione en dispositivos que dispongan de Android 1.6 o superior.

3. ANÁLISIS

A continuación se detallan los documentos y datos analizados y utilizados en el desarrollo de la aplicación.

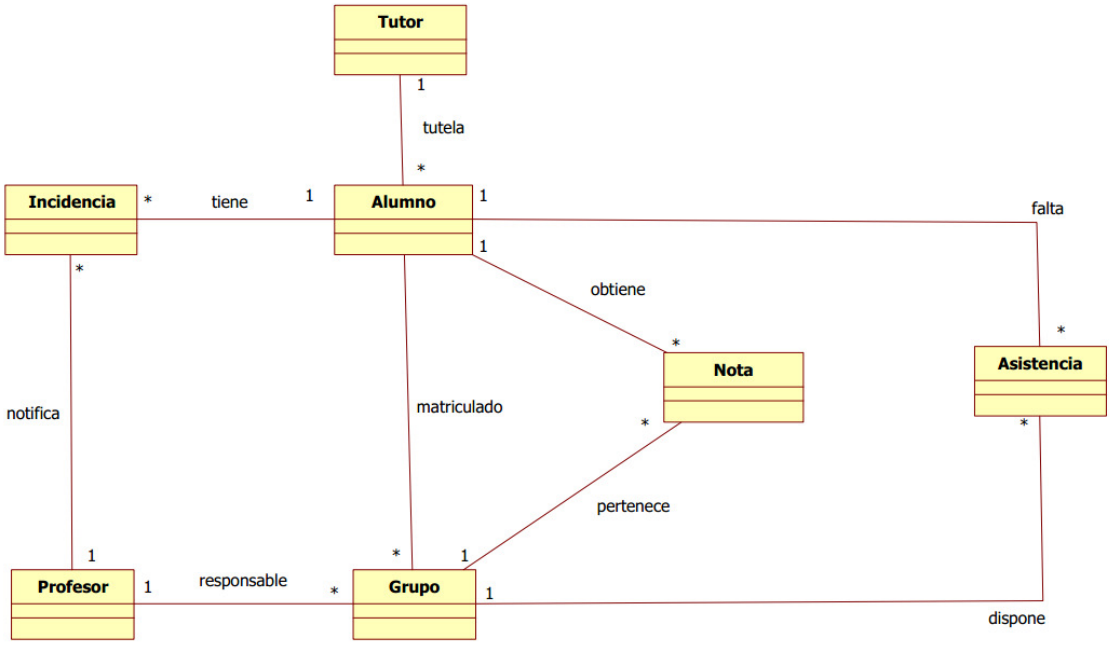
3.1. CASOS DE USO

En la etapa de análisis se hace uso de la metodología UML. El primer punto es la realización de los casos de uso. En este caso, como característica o rasgo, podemos destacar que al existir únicamente un usuario todas las acciones los lleva a cabo el mismo, por lo tanto solo es necesario un diagrama de caso de uso para especificar toda la funcionalidad de la aplicación.



3.2. DIAGRAMAS DE CLASES

El siguiente diagrama corresponde al diagrama de clase. La realización de este diagrama ayuda al entendimiento del sistema. Del diagrama de casos de uso se puede extraer una serie de clases necesarias y que ayudan al desarrollo de la aplicación, específicamente en la capa de lógica con el desarrollo de las clases necesarias.



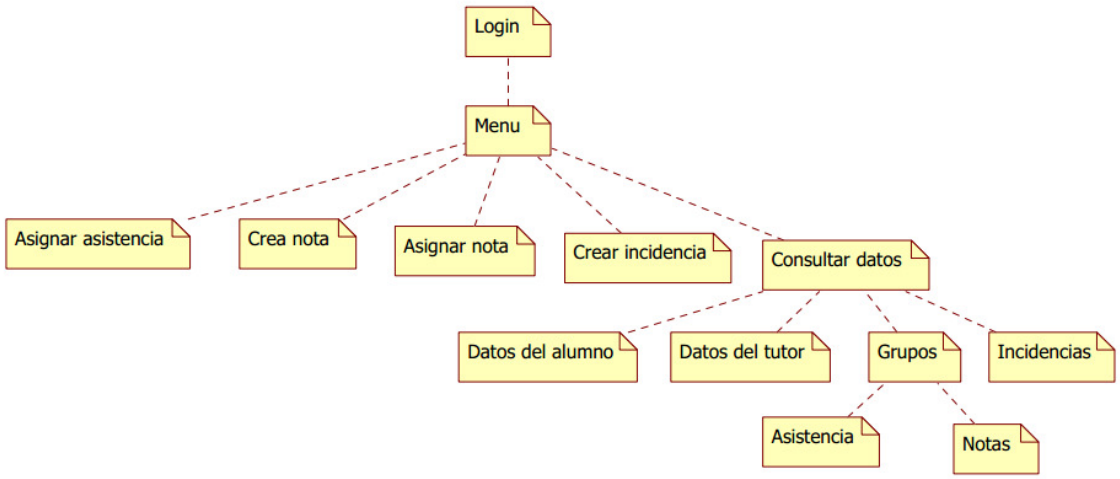
4. DISEÑO

Puesto que se hace uso de un lenguaje orientado a objetos, se emplea el patrón modelo-vista-controlador, dividiendo la aplicación en tres niveles, el nivel de datos correspondiente al modelo, el nivel de interfaz correspondiente a la vista y la lógica de la aplicación correspondiente al controlador.

4.1. VISTA

La vista corresponde con la interfaz de usuario que se muestra. El punto principal de la interfaz es la simplicidad de ésta, puesto que la intención es que la aplicación sea rápida, intuitiva y fácil de usar. Por otra parte, gracias a que Android emplea el patrón anteriormente comentado, modelo-vista-controlador, en caso de querer realizar cambios en la interfaz, tan solo tendremos que modificar la propia interfaz, sin alterar la lógica o funcionamiento de la aplicación.

En el siguiente diagrama se muestra de forma sencilla la navegación por las distintas funcionalidades, de forma que sea sencillo visualizar cómo acceder a la información y la estructura interna de la aplicación.



4.2. CONTROLADOR

En el controlador encontramos el código necesario para que la aplicación realice las funciones que se esperan de ella, es decir el código que se ejecuta cuando el usuario interacciona con la aplicación y ésta debe responder a la interacción. Se ha utilizado orientación a objetos para realizar la lógica de la aplicación, y como ya se ha comentado en el apartado correspondiente a la vista, debido a que Android hace uso del patrón modelo-vista-controlador, podemos modificar detalles del controlador sin que esto afecte al resultado mostrado.

4.3. MODELO

El modelo consiste en el comportamiento e información del dominio de la aplicación, así como del acceso a la base de datos y la propia base de datos.

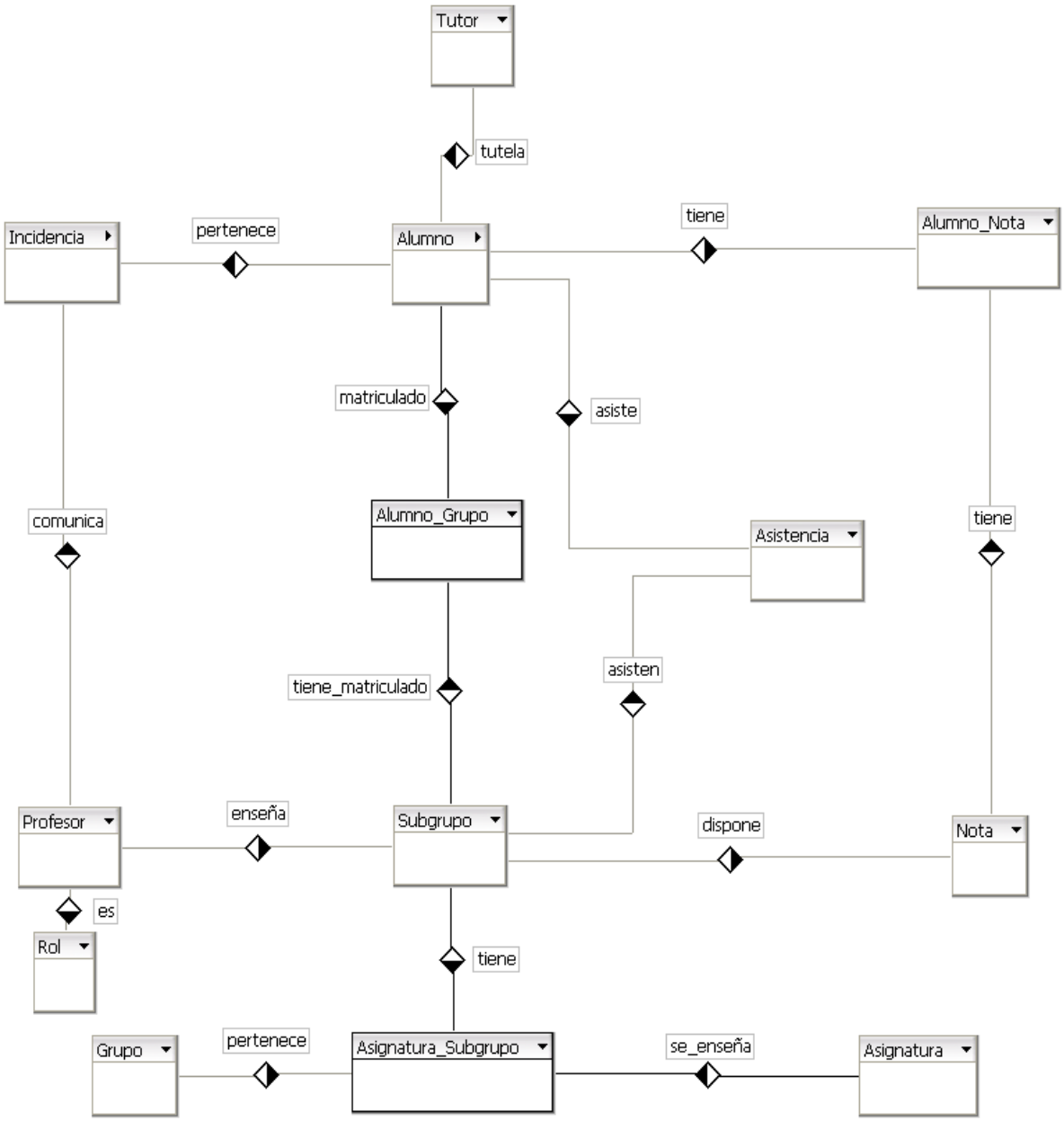
En cuanto al comportamiento e información del dominio destaca la colección de distintas clases empleadas para el desarrollo de la aplicación y vienen establecidas por el diagrama de clases anteriormente mostrado. Esto permite encapsular la información obtenida de forma que pueda manejarse de forma cómoda y sencilla mediante el apartado controlador.

En cuanto a la base de datos, es donde se almacena la información. Así pues, esta parte hace uso de un sistema de gestión de base de datos, el cual en este caso es MySQL.

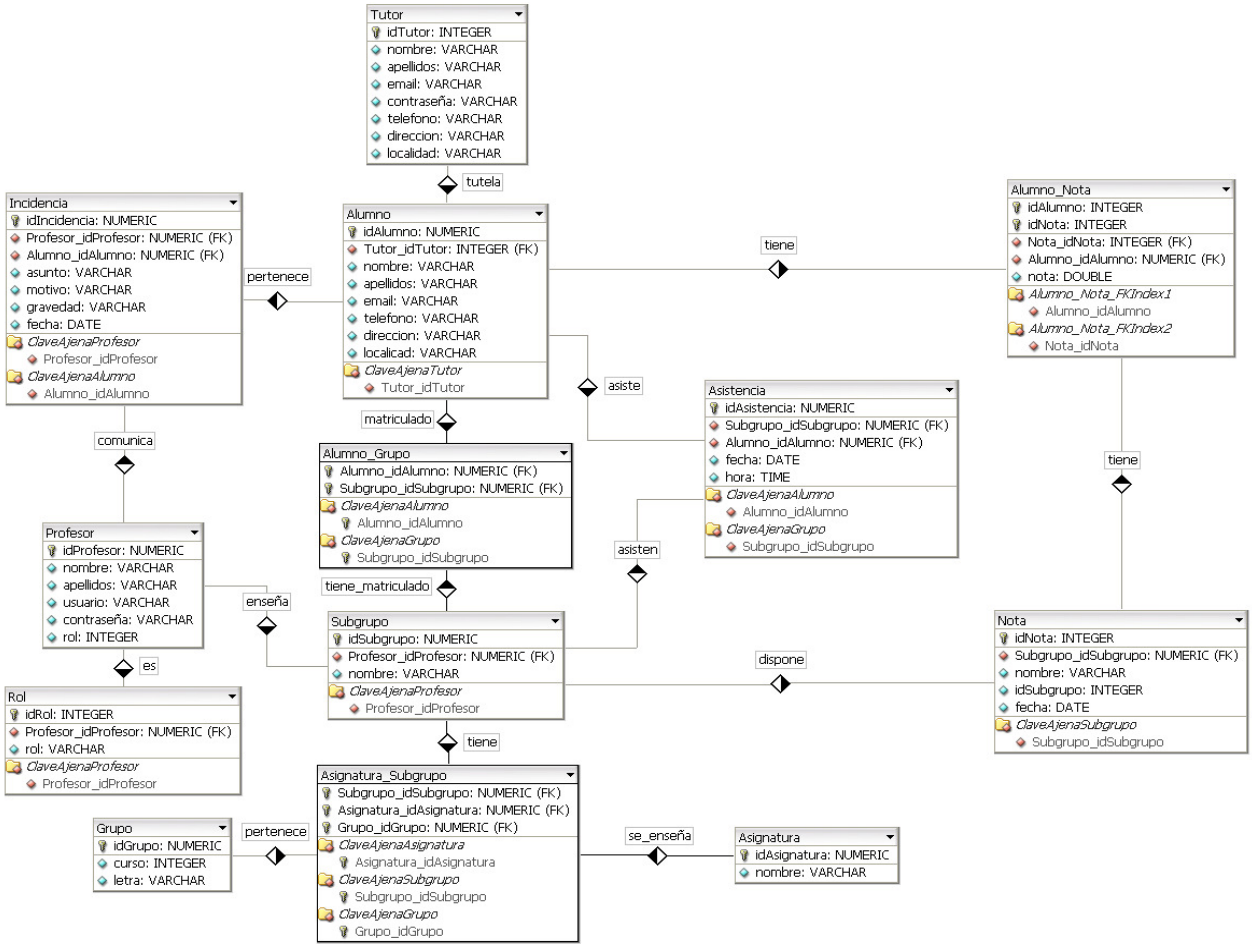
En la base de datos almacenamos la información relacionada con los alumnos, los profesores, asignaturas, etc. De esta forma, podemos acceder a ella mediante la aplicación, introducir nuevos datos o modificarlos. Para poder realizar este mantenimiento de la información se ha empleado el lenguaje PHP. Además, PHP ofrece una alta compatibilidad con el sistema de gestión MySQL, por lo que el uso conjunto del lenguaje y el sistema de gestión de base de datos es realmente sencillo.

A la hora de desarrollar la base de datos se realizó un diagrama entidad relación donde se plasmaron las entidades más relevantes. Este diagrama fue evolucionando y modificándose, añadiendo nuevas entidades para añadir información, mejorar la manera de acceder a los datos y mejorar su facilidad de uso.

En el siguiente diagrama se puede observar el diagrama entidad-relación empleado para modelar la base de datos. Este diagrama se muestra sin atributos para poder ofrecer una vista general de la estructura de la base de datos.



En el diagrama mostrado a continuación podemos ver la base de datos con todos sus atributos.



Por último se muestra el diseño lógico correspondiente a los diagramas entidad relación ya mostrados.

- Alumno (idAlumno: INTEGER, nombre: VARCHAR(100), apellidos: VARCHAR(100), email: VARCHAR(100), contraseña: VARCHAR(100), teléfono: VARCHAR(15), dirección: VARCHAR(100), localidad: VARCHAR(100))

  CP {idAlumno}

  VNN {nombre, apellidos, idTutor, email, contraseña, teléfono, dirección, localidad}

  CAj {idTutor} → Tutor

- Alumno_nota (idAlumno: INTEGER, idNota: INTEGER, nota: DOUBLE)

  CP {idAlumno, idNota}

  VNN {nota}

  CAj {idAlumno} → Alumno

  CAj {idNota} → Nota

- Alumno_subgrupo (idAlumno: INTEGER, idSubgrupo: INTEGER)

  CP {idAlumno, idSubgrupo}

  CAj {idAlumno} → Alumno

  CAj {idSubgrupo} → Subgrupo

- Asignatura (idAsignatura: INTEGER, nombre: VARCHAR(100), código: VARCHAR(3))

  CP {idAsignatura}

  VNN {nombre, código}

- Asignatura_grupo (idSubgrupo: INTEGER, idGrupo: INTEGER, idAsignatura: INTEGER)

  CP {idSubgrupo, idGrupo, idAsignatura}

  CAj {idSubgrupo} → Subgrupo

  CAj {idGrupo} → Grupo

  CAj {idAsignatura} → Asignatura

- Asistencia (idAsistencia: INTEGER, fecha: DATE, hora: TIME, idAlumno: INTEGER, idSubgrupo: INTEGER)

  CP {idAsistencia}

  VNN {fecha, hora, idAlumno, idSubgrupo}

  CAj {idAlumno} → Alumno

  CAj {idSubgrupo} → idSubgrupo

- Grupo (idGrupo: INTEGER, curso: INTEGER, letra: VARCHAR(1))

  CP {idGrupo}

  VNN {curso, letra}

- Incidencia (idIncidencia: INTEGER, fecha: DATE, hora: TIME, asunto: VARCHAR(100), motivo: VARCHAR(100), gravedad: VARCHAR(5), idProfesor: INTEGER, idAlumno: INTEGER)

  CP {idIncidencia}

  VNN {fecha, hora, asunto, motivo, gravedad, idProfesor, idAlumno}

  CAj {idProfesor} → Profesor

  CAj {idAlumno} → Alumno

- Nota (idNota: INTEGER, nombre: VARCHAR(100), idSubgrupo: INTEGER, fecha: DATE)

  CP {idNota}

  VNN {nombre, idSubgrupo, fecha}

- Profesor (idProfesor: INTEGER, nombre: VARCHAR(100), apellidos: VARCHAR(100), usuario: VARCHAR(16), password: VARCHAR(16))

  CP {idProfesor}

  VNN {nombre, apellidos, usuarios, password}

- Rol (idRol: INTEGER, rol: VARCHAR(100))

  CP { idRol }

  VNN { rol}

- Subgrupo (idSubgrupo: INTEGER, nombre: VARCHAR(100), idProfesor: INTEGER)

  CP {idSubgrupo}

  VNN {nombre, idProfesor}

- Tutor (idTutor: INTEGER, nombre: VARCHAR(100), apellidos: VARCHAR(100), email: VARCHAR(255), password: VARCHAR(255), teléfono: VARCHAR(15), dirección: VARCHAR(100), localidad: VARCHAR(100))

  CP {idTutor}

  VNN {nombre, apellidos, email, password}

5. IMPLEMENTACIÓN E INTEGRACIÓN

En este apartado se detallan las distintas tecnologías, herramientas, así como detalles de la implementación e integración.

5.1. TECNOLOGÍAS

La aplicación se ha desarrollado empleando las siguientes tecnologías:

Microsoft Windows XP como sistema operativo sobre el que se ha desarrollado el proyecto. Lanzado al mercado en octubre del año 2001 y en la actualidad, tras más de 10 años, el sistema operativo Windows XP aún sigue en uso, aunque decrece el número de usuarios del sistema operativo en favor de la última versión lanzada por Microsoft, Windows 7, a la venta desde octubre de 2009

Android de la empresa Google como sistema operativo bajo el cual la aplicación se ejecuta. En la actualidad, junto con iOS, Android es el sistema operativo móvil más utilizado y con un mayor número de aplicaciones disponibles y desarrolladores, lo que lo convierte en un sistema operativo con un amplio número de usuarios.

Apache como servidor web mediante el cual se proporciona acceso a la base de datos y a las funciones PHP que permiten a la aplicación comunicarse con la base de datos.

MySQL como sistema de gestión de base de datos y su herramienta PHPMyAdmin para gestionar la base de datos. Cabe mencionar que MySQL no sería la opción adecuada a la hora de llevar el proyecto a un uso real puesto que existen mejores sistemas que ofrecen un mayor rendimiento a la hora de acceder, modificar y eliminar información de una base de datos, a pesar de eso se escogió debido a su facilidad de uso y la compatibilidad con Apache gracias a la herramienta XAMPP.

5.2. HERRAMIENTAS

En cuanto a las herramientas usadas para el desarrollo de la aplicación son las siguientes:

Para el desarrollo del código fuente de la aplicación se ha empleado Eclipse como entorno de desarrollo integrado. Además, se ha usado el SDK de Android, el cual funciona con Eclipse a la perfección, gracias a la capacidad del entorno para permitir el uso de plugins.

Para la implementación de la base de datos, se ha usado la herramienta PHPMyAdmin, disponible en el paquete XAMPP. Mediante esta aplicación es posible administrar la base de datos y ofrece utilidades para la creación, modificación y eliminación de bases de datos, así como de las tablas y relaciones que forman la base de datos.

5.3. DETALLES DE LA IMPLEMENTACIÓN

A la hora de desarrollar la aplicación se ha realizado especial hincapié en poder adaptar la aplicación a cualquier versión existente de la aplicación. Así pues, la aplicación se desarrolló con el objetivo de hacerla compatible con el mayor número de dispositivos, evitando disponer de una versión de Android actual, de forma que incluso dispositivos antiguos con versiones anteriores del sistema operativo y sin posibilidad de actualizarse puedan utilizar la aplicación. En definitiva, la aplicación funciona en cualquier dispositivo que disponga de Android 1.6 en adelante, siendo compatible incluso con la reciente versión del sistema operativo, la versión 4.0.

Por otro lado, otro de los objetivos a la hora de desarrollar la aplicación ha sido adaptar la interfaz a cualquier tamaño de pantalla o disposición. De esta forma, la aplicación muestra toda la información necesaria independientemente del tamaño de la pantalla o de la orientación del dispositivo, ya que funciona tanto en pantallas pequeñas, como grandes, o cuando el smartphone se encuentra en posición vertical u horizontal. Este factor ha posibilitado también que la aplicación pueda utilizarse en tablets, debiendo de adaptar mínimamente la interfaz para que se muestre la información correctamente en la pantalla de mayor tamaño que poseen las tablets.

Dentro del código reseñable cabe mencionar entre otras cosas la construcción de interfaces en Eclipse mediante el SDK de Android. Para analizarlo se tomará de ejemplo la interfaz realizada para el menú cuando el dispositivo se encuentra en modo horizontal, también conocido como orientación horizontal o *landscape*. A continuación se muestra el código completo.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/logo"
            android:layout_gravity="center"/>
        <RelativeLayout
            android:id="@+id/relativeLayout1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" >
            <Button
                android:id="@+id/attendance_button"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dip"
                android:text="@string/attendance_label" />
            <Button
                android:id="@+id/grade_button"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_alignParentTop="true"
                android:layout_marginRight="10dip"
                android:text="@string/grade_label" />
            <Button
                android:id="@+id/trouble_button"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_below="@+id/grade_button"
                android:layout_marginLeft="10dip"
                android:text="@string/trouble_label"/>
            <Button
                android:id="@+id/students_button"
                android:layout_width="200dip"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_below="@+id/grade_button"
```

```
                android:layout_marginRight="10dip"
                android:text="@string/students_label"/>
        </RelativeLayout>
        <ImageButton
            android:id="@+id/exit_button"
            android:layout_width="45dip"
            android:layout_height="45dip"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dip"
            android:background="@null"
            android:scaleType="centerInside"
            android:src="@drawable/ic_close_button"/>
 </LinearLayout>
</LinearLayout>
```

Dentro de este código podemos destacar diversas características. En primer lugar Android emplea para las interfaces etiquetas para la distribución de los elementos, en este caso se utiliza una etiqueta *<LinearLayout>* para contener todos los elementos y a su vez se divide en otro *<LinearLayout>* que contiene a su ver un *<RelativeLayout>*. La diferencia principal entre *<LinearLayout>* y *<RelativeLayout>* es que el segundo permite alinear y situar componentes según la situación de otros componentes. Así pues, los botones se han situado según la posición del resto. Por ejemplo:

```
<Button
      android:id="@+id/trouble_button"
      android:layout_width="200dip"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_below="@+id/grade_button"
      android:layout_marginLeft="10dip"
      android:text="@string/trouble_label" />
```

En este botón indicamos que debe estar debajo de otro botón mediante el atributo *android:layout_below*, indicando en su valor el id del botón que se situará por encima de éste.

Todos los componentes disponen de un atributo *android_id* que sirve para identificarlos. Anteponiendo *@+id/* al id del botón estamos indicando que el id debe añadirse al fichero R.java, un fichero generado de forma automática que permite acceder a los elementos de la interfaz en el código.

Por otro lado, en el botón podemos observar como el texto que se muestra no es un texto normal, si no que se está accediendo al valor *trouble_label* del fichero strings.xml, en el cual tenemos entradas clave valor de la siguiente manera:

```
<string name="trouble_label">Incidencias</string>
```

Otro atributo destacable es *android:layout_marginLeft* puesto que se puede observar que el valor no está en píxeles o medidas similares, sino en *dip*, que no es otra cosa que pixeles independientes de la densidad, de forma que se mantiene la proporción al emplear distintas pantallas.

A continuación podemos ver el elemento *ImageButton*, utilizado para crear un botón que use una imagen y no texto como un botón habitual*:*

```
<ImageButton
	android:id="@+id/exit_button"
	android:layout_width="45dip"
	android:layout_height="45dip"
	android:layout_gravity="center_horizontal"
	android:layout_marginTop="20dip"
	android:background="@null"
	android:scaleType="centerInside"
	android:src="@drawable/ic_close_button" />
```

Dispone de los habituales atributos para el identificador y la disposición así como otros empleados para establecer el color del fondo del botón. Mediante *android:scaleType* se indica cómo se debe escalar la imagen de forma que se pueda centrar o ajustar a uno de los lados. Por último *android:src* nos permite seleccionar el recurso que se utilizará, en este caso la imagen *ic_close_button*.

ACCESO A LA BASE DE DATOS

Para acceder a la información almacenada se emplean unas funciones desarrolladas en PHP que realizan la comunicación entre la base de datos y la aplicación. En estas funciones se realiza la petición correspondiente a la base de datos y devuelve los datos.

```
public static boolean checkAttendance(int idGroup, int idStudent) {
	ArrayList<String> names = new ArrayList<String>();
	names.add("idGroup");
	names.add("idStudent");
	ArrayList<String> values = new ArrayList<String>();
	values.add(String.valueOf(idGroup));
	values.add(String.valueOf(idStudent));

	ArrayList<NameValuePair> nameValuePairs = putValues(names,
values);

	String method = "php/checkAttendance.php";

	JSONArray jArray = petition(nameValuePairs, method);

	try {
		for(int i=0;i<jArray.length();i++){
			JSONObject json_data = jArray.getJSONObject(i);
			String id = json_data.getString("idAsistencia");
```

```
                if(id.compareTo("")!=0)
                    return true;
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return false;
}
```

En este ejemplo vemos que se reciben los identificadores del grupo y del alumno, se almacenan en dos vectores para generar posteriormente un vector con pares clave valor y a continuación se llama al método petición que genera la petición y devuelve un vector con datos en formato JSON. Tan solo es necesario comprobar si existe un identificador de una falta de asistencia y entonces devolver true o en caso contrario al terminar la búsqueda, devolver false. Tanto el método *putValues* como *petición* son métodos propios empleados para reducir el número de líneas a escribir a la hora de realizar las distintas peticiones puesto que de lo contrario habría que realizar el mismo proceso largo y tedioso para conectar con el servidor.

DESARROLLO DEL CÓDIGO DE LA APLICACIÓN ANDROID

Al hablar de detalles de implementación es necesario mencionar algunas particularidades de Android como las mostradas en el siguiente código:

```
public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        String name = ((TextView) view).getText().toString();
        Iterator<Subgroup> it = subgroups.iterator();
        Subgroup sub = null;
        while(sub == null) {
                Subgroup aux = it.next();
                if(name.compareTo(aux.getFullName()) == 0)
                        sub = aux;
        }
        Bundle extras = getIntent().getExtras();
        Student s = (Student) extras.getParcelable("Student");

        Intent intent = new Intent(this, StudentsDataGroupsData.class);
        intent.putExtra("Subgroup", sub);
        intent.putExtra("Student",s);
        startActivity(intent);
}
```

El código mostrado corresponde al evento onItemClick disparado cuando se pulsa sobre una fila de una lista. El primer detalle es el tipo del parámetro *parent*, el cual es *AdapterView<?>* que corresponde a un manejador para las listas y tiene la posibilidad de ser personalizado de ahí que el tipo específico no venga definido y se muestre el símbolo *?*. Por otra parte en este código podemos ver como se crean nuevas vistas para mostrarlas mediante los *intent* al final del código.

```
Intent intent = new Intent(this, AlumnoDatosGruposDatos.class);
intent.putExtra("Subgroup", sub);
intent.putExtra("Student", s);
startActivity(intent);
```

Para añadir una nueva vista, llamadas *activities* en Android, tan solo debemos crear un objeto del tipo *intent* con el fichero *.class* que queramos utilizar y llamar al método *startActivity()* con el objeto de tipo *Intent* como parámetro. Cabe recordar que es necesario declarar todos y cada uno de los archivos *.class* que se vayan a utilizar en el fichero *AndroidManifest.xml,* que se emplea para añadir permisos tanto a nivel de funcionalidades, como por ejemplo permitir la conexión a Internet, como a nivel de los *activities* que se van a ejecutar. En caso contrario la aplicación dará error al intentar usar un fichero *.class* que no se ha declarado en el fichero *AndroidManifest.xml*.

Los datos añadidos que también aparecen en el código, por ejemplo, *intent.putExtra* se emplean para pasar objetos entre *activities*, de forma que los objetos sub y s se pasan de la *activity* actual a la nueva que se está creando, de forma que no se tiene que volver a crear el objeto, acceder a su información en la base de datos, etc. Para recuperarlo tan solo necesitamos utilizar las líneas siguientes:

```
Bundle extras = getIntent().getExtras();
Student s = (Student) extras.getParcelable("Student");
```

Por otro lado, mediante el siguiente código se puede ver el uso de dos interfaces distintas en una sola clase, es decir, se usan dos interfaces diferentes en la misma *activity*, de esta forma, vemos como es posible gestionar el uso de dos vistas sin necesitar para ello crear dos *activities* distintas y gestionar el paso de una a otra. Esto se realiza con el uso de dos métodos que intercambian la vista mediante el método *setContentView*, el cual establece el fichero XML que se usará para mostrar la interfaz.

```
private void initializeA () {
	creating = 0;
	defaultIncident = false;
	initializeSpinners();
	Spinner spin = (Spinner) findViewById(R.id.spinner_type);
	ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.type,
android.R.layout.simple_spinner_item);
	adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
	spin.setAdapter(adapter);
	spin.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
		public void onItemSelected(AdapterView<?> parent, View
view, int pos, long id) {
			switch (pos) {
				case 0:
					break;
				case 1:
					setContentView(R.layout.incident_alt_view);
					initializeB();
					break;
			}
		}
	  });
}

private void initializeB () {
	creating = 1;
	defaultIncident = true;
	initializeSpinners();
	Spinner spin = (Spinner) findViewById(R.id.spinner_type);
	ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.type,
android.R.layout.simple_spinner_item);
	adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
	spin.setAdapter(adapter);
	spin.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
		public void onItemSelected(AdapterView<?> parent, View
view, int pos, long id) {
			switch (pos) {
				case 0:
					if (creating >= 2) {
					    setContentView(R.layout.incident_view);
					    initializeA();
					}
					break;
				case 1:
					break;
			}
		 }
	  });
	spin.setSelection(1);
	creating = 2;
}
```

Los dos métodos mostrados se encargan de inicializar lo necesario según se vaya a usar una vista u otra mediante setContentView(). El resto de métodos, como por ejemplo el botón de aceptar a la hora de confirmar los cambios se gestiona de manera independiente a la interfaz, por ello no es necesario hacer activities distintas puesto que gran parte del código y funcionamiento es prácticamente similar.

A la hora de mostrar listas, Android no permite una gran personalización de forma sencilla por lo que es necesario desarrollar código para adaptarlo a la funcionalidad deseada. Por ejemplo, en el desarrollo del proyecto, la asignación de faltas de asistencia se realiza mediante una lista que muestra filas compuestas por un *checkbox* y el nombre del alumno. Para poder realizar distintas acciones al pulsar el *checkbox* es necesario realizar una subclase del objeto de Android *BaseAdapter* empleado para mostrar listas. En el siguiente código se muestra el código utilizado para aplicar o eliminar faltas de asistencia.

```
public View getView(int position, View convertView, ViewGroup
viewGroup) {
	TextView textView;
	CheckBox checkBox;

	if (convertView == null) {
		LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
		convertView =
inflater.inflate(R.layout.attendance_view_item, null);
	}
	textView =
(TextView)convertView.findViewById(R.id.student_item);
	final Student s = students.get(position);
	textView.setText(s.getFullName());
	boolean attendance = s.checkAttendance(subgroup);

	checkBox = (CheckBox)
convertView.findViewById(R.id.student_checkbox);
	checkBox.setOnCheckedChangeListener (
	new OnCheckedChangeListener() {
		@Override
		public void onCheckedChanged(CompoundButton button, boolean
valor)
		{
			boolean attendance = s.checkAttendance(subgroup);
			if(!valor) {
				if(!attendance)
					s.putAbsence(subgroup);
			} else {
				if(attendance)
					s.deleteAbsence(subgroup);
			}
		}
	});
```

```
        if(attendance && checkBox.isChecked()) {
             checkBox.setChecked(false);
        } else if(!attendance && !checkBox.isChecked()) {
             checkBox.setChecked(true);
        }
        return convertView;
}
```

Se muestra el método *getView*, encargado de dibujar cada fila y en el cual se añade al *checkbox* el método necesario, llamados en Android *listeners,* para responder a las interacciones del usuarios.

```
checkBox.setOnCheckedChangeListener (
new OnCheckedChangeListener() {
     @Override
     public void onCheckedChanged(CompoundButton button, boolean
valor)
     {
          boolean attendance = s.checkAttendance(subgroup);
          if(!valor) {
               if(!attendance)
                  s.putAbsence(subgroup);
          } else {
               if(attendance)
                    s.deleteAbsence(subgroup);
          }
     }
});
```

Al añadir este método se sobreescribe el método *onCheckedChanged* que se dispara cuando el *checkbox* cambia de estado entre marcado y sin marcar. Cuando se ejecuta ese método se comprueba el estado y en caso de cambiarse se añade la falta o se elimina según sea necesario.

A lo largo del código mostrado se puede ver la llamada al método *findViewById*, el cual permite acceder a elementos mostrados en la interfaz. Como ya se comentó con anterioridad, existe un fichero llamado R.java que dispone de los identificadores de los elementos que componen una interfaz, así por ejemplo, para añadir un *listener* a un botón, tan solo debemos recuperarlo con el método *findViewById* y a continuación agregarle el *listener* correspondiente.

```
View okButton = findViewById(R.id.ok_button);
okButton.setOnClickListener(this);
```

Mediante estas dos líneas de código recuperamos un botón, utilizado en el login, y le añadimos el *listener* que responde a los clicks sobre el botón. Una vez realizado esto, tan solo es necesario implementar el método *onClick* y realizar las acciones pertinentes.

Otra de las particularidades de Android surge a la hora de crear interfaces que dispongan de pestañas. Para realizar esto es necesario crear una vista que contenga las pestañas y su clase correspondiente donde se configuran cada una de ellas como se ve a continuación:

```
Resources res = getResources();
TabHost tabHost = getTabHost();

Intent intent = new Intent(this, StudentsDataGroupAtendance.class);
intent.putExtra("Student", s);
intent.putExtra("Subgroup", sub);

TabHost.TabSpec spec =
      tabHost.newTabSpec("tab1").setIndicator("Asistencia",
res.getDrawable(R.drawable.ic_tab_attendance)).setContent(intent);
tabHost.addTab(spec);

intent = new Intent(this, StudentsDataGroupGrades.class);
intent.putExtra("Student", s);
intent.putExtra("Subgroup", sub);

spec = tabHost.newTabSpec("tab2").setIndicator("Notas",
res.getDrawable(R.drawable.ic_tab_grade)).setContent(intent);
tabHost.addTab(spec);

tabHost.setCurrentTab(0);
```

Podemos ver como se crea un *intent* para cada pestaña, en este caso dos, se añaden los objetos necesarios mediante *putExtra* y a continuación se crea un objeto de tipo *TabSpec* que dispone del identificador de la pestaña, el nombre, el icono y el *intent* creado que permite cargar una nueva *activity* para mostarla. A la hora de indicar el icono se realiza mediante un archivo XML, llamados en este caso *ic_tab_attendance* y *ic_tab_grade,* con el siguiente formato:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/ic_tab_attendance_selected"
        android:state_selected="true"
        android:state_pressed="false" />
    <item android:drawable="@drawable/ic_tab_attendance_unselected" />
</selector>
```

En este fichero se está indicando el icono que se debe usar para cuando la pestaña esté activa y cuando esté inactiva, mediante el acceso a ficheros locales del proyecto almacenados en la carpeta *drawable* y accedidos mediante *@drawable*.

Por último comentar la manera de mostrar ventanas emergentes. Esto se realiza mediante el siguiente código:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage(incident).setCancelable(false).setPositiveButton("A
ceptar", null);
AlertDialog alert = builder.create();
alert.show();
```

Se puede ver como se crea un nuevo objeto, se indica el mensaje, si existirán botón para cancelar y por último, el botón para aceptar. A continuación tan solo es necesario crear el mensaje emergente mediante el objeto creado, y mostrarlo con *show.*

ESTRUCTURA DEL PROYECTO

Por último, con el fin de permitir el análisis del proyecto, su continuación o mejora, por parte de personas ajenas al desarrollo inicial se explicará la estructura que tiene, para así poder facilitar su entendimiento.

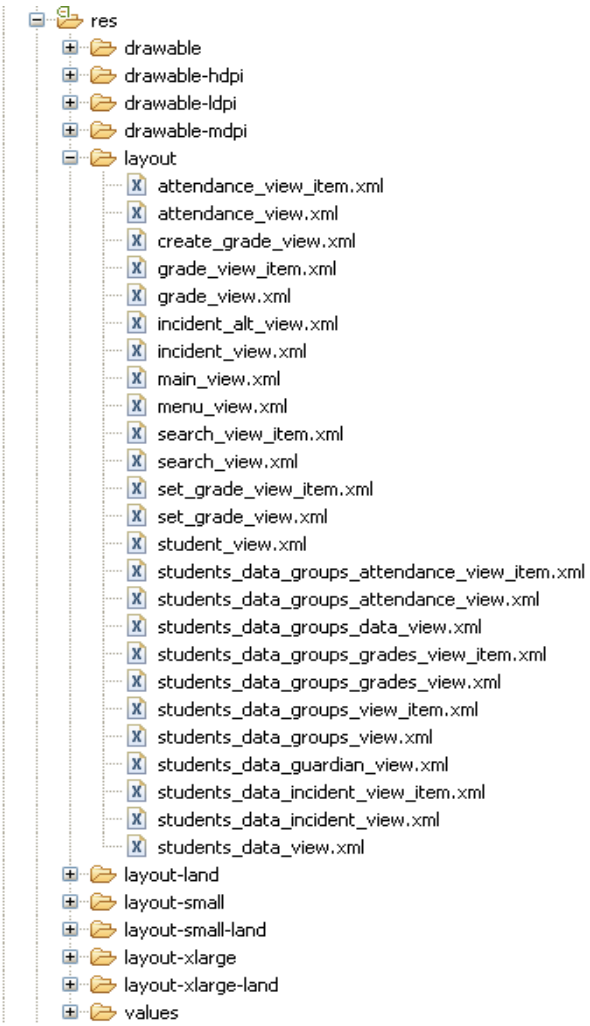
En primer lugar, se explicará cómo estructura Android los recursos necesarios para crear la interfaz de la aplicación. Para ello, al crear un proyecto se crea una carpeta llamada *res* que contiene varias carpetas dentro de ella. Las tres carpetas principales dentro de *res* son *drawable, layout* y *values.*

Drawable contiene las imágenes utilizadas por la aplicación, así como ficheros XML utilizados para crear el estilo de los botones y de las pestañas y poder mostrar una imagen cuando están pulsados y otra cuando no. Así pues, esta carpeta contiene las imágenes del proyecto y los ficheros XML utilizados para crear los estilos de las pestañas del apartado de consulta de datos.

Por su parte la carpeta *layout* contiene las vistas encargadas de crear la interfaz, es decir, los ficheros XML a partir de los cuales se crea la interfaz. Existen diversos tipos de carpetas *layout*, como por ejemplo *layout-land, layout-small,* etc. Esta característica se utiliza para crear distintas interfaces según los distintos tipos de interfaz, tamaños u orientaciones. El proyecto hace uso de la carpeta *layout* para establecer la interfaz básica para dispositivos móviles con una resolución de tamaño normal, alrededor de los 480x800, correspondiente a la mayoría de móviles, entre ellos el Google Nexus One. Puesto que ha sido necesario adaptar algunas de las interfaces para otras pantallas se han empleado otras carpetas específicas. Así pues, para pantallas de tamaño normal y orientación horizontal se ha utilizado la carpeta *layout-land*. Para pantallas pequeñas, alrededor de los

240x320, como la disponible en teléfonos HTC Wildfire, se ha empleado la carpeta *layout -small* y *layout -small-land* en el caso de orientaciones horizontales. Por último, para adaptar la interfaz en pantallas grandes correspondiente a tablets se ha empleado la carpeta *layout-xlarge* y en el caso de orientaciones horizontales *layout-xlarge-land.*

La última carpeta importante dentro de *res* es la carpeta *values*, utilizada para almacenar valores de la forma clave valor. Se utiliza principalmente para almacenar cadenas de caracteres y accederlas posteriormente haciendo uso del id. Es útil a la hora de realizar una aplicación puesto que permite modificar una cadena repetida varias veces a lo largo de la aplicación con solo cambiarla en un fichero. Dentro de *values* encontramos los archivos *string* y *arrays*. El primero de ello se usa para almacenar simples cadenas y el segundo permite crear arrays de cadenas, utilizados en los *spinners*, también llamados menús desplegables.



En cuanto al código que implementa las funcionalidades puede encontrarse en la carpeta *src*. Aquí hay que hacer una distinción entre archivos que actúan como controladores y archivos que actúan como objetos. Los primeros forman parte del patrón modelo-vista-controlador y se utilizan para comunicar las interacciones del usuario con el dispositivo con los cambios que se requieren hacer en el modelo. Los segundos se utilizan para crear los objetos utilizados en la funcionalidad y se encuentran en la carpeta *classes.* En cuanto a los controladores, se muestra a continuación una lista con el nombre y la funcionalidad de la que se encargan de forma que se pueda localizar fácilmente donde se encuentra una funcionalidad determinada para poder modificarla.

- *Attendance.java*: Implementa la funcionalidad correspondiente al control de la asistencia. Es aquí donde se marca una falta o se elimina. Hace uso del archivo AttendanceAdapter.java encargado de gestionar la interacción con el *checkbox* utilizado para poner las faltas. Los ficheros de la vista correspondiente son *attendance_view.xml* y *attendance_view_item.xml*

- *CreateGrade.java:* Implementa la funcionalidad encargada de crear nuevas notas. Su vista es *create_grade_view.xml*.

- *GradeData.java*: Implementa la funcionalidad utilizada para elegir qué nota se va a asignar. Hace uso de *grade_view.xml* y *grade_view_item.xml* como archivos para la vista.

- *IncidentData.java*: Implementa la funcionalidad usada para la creación de las incidencias. Puesto que existen dos interfaces para las incidencias hace uso de dos ficheros para la vista *incident_view.xml* y *incident_alt_view.xml*.

- *Menu.java:* Implementa la funcionalidad correspondiente al menú principal desde el que se accede a los distintos apartados de la aplicación. Su vista se encuentra en el archivo *menu_view.xml.*

- *Project.java*: Implementa la funcionalidad encargada del login. El fichero de la vista de esta funcionalidad es *main_view.xml.*

- *Search.java:* Implementa la funcionalidad utilizada para realizar la búsqueda. Puesto que se muestra mediante una lista hace uso de la vista *search_view.xml* para la interfaz y *search_view_item.xml* para los elementos de la lista.

- *SetGrade.java:* Implementa la funcionalidad usada para asignar notas a los alumnos. Usa los ficheros *set_grade_view.xml* y *set_grade_view_item.xml* para crear la interfaz.

- *Students.java:* Implementa el código necesario para crear la pantalla principal de consulta con las pestañas y enlazar cada pestaña con la *activity* que tiene que emplear. El fichero *student_view.xml* corresponde a su vista.

- *StudentsData.java:* Implementa la funcionalidad utilizada para mostrar los datos de los alumnos. Su vista se encuentra en el fichero *students_data_view.xml*

- *StudentsDataGroupAttendance.java:* Implementa la funcionalidad correspondiente a mostrar las faltas de un alumno en un grupo. Su vista usa dos ficheros, *students_data_groups_attendance_view.xml* y *students_data_group_attendance_view_item.xml.*

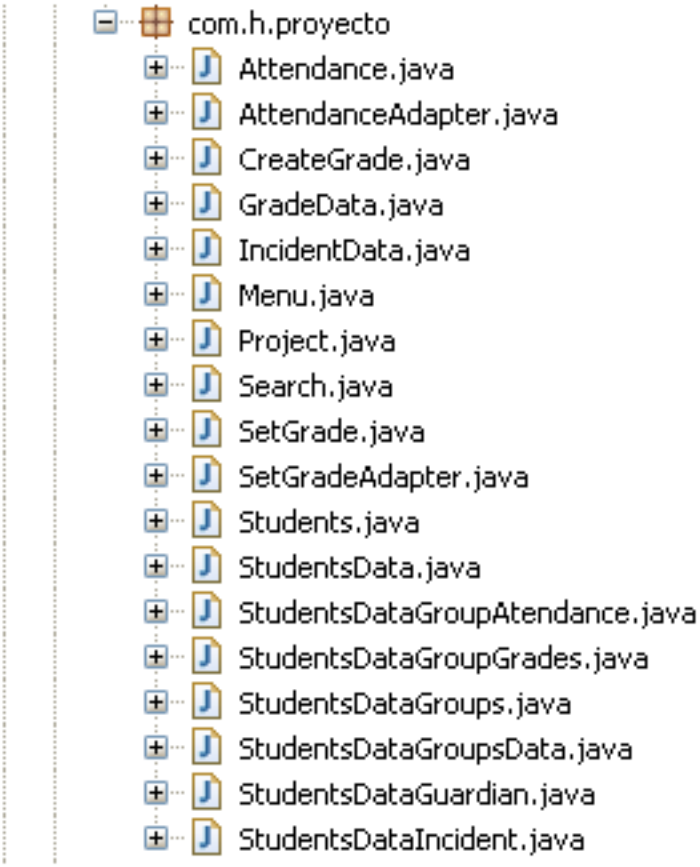
- *StudentsDataGroupGrades.java:* Implementa el código necesario para mostrar las notas de un alumno en un grupo específico. Los ficheros que se utilizan para su vista son *students_data_group_grades_view.xml* y *students_data_group_grades_view_item.xml.*

- *StudentsDataGroups.java:* Se utiliza para mostrar los grupos a los que pertenece el alumno del cual se están consultando sus datos. Sus ficheros para la vista son *students_data_groups_view.xml* y *students_data_groups_view_item.xml.*

- *StudentsDataGroupsData.java:* Se utiliza para crear la interfaz correspondiente a las pestañas usadas para mostrar las notas y asistencias según grupo. *Students_data_groups_data_view.xml* es el fichero utilizado para crear la interfaz.

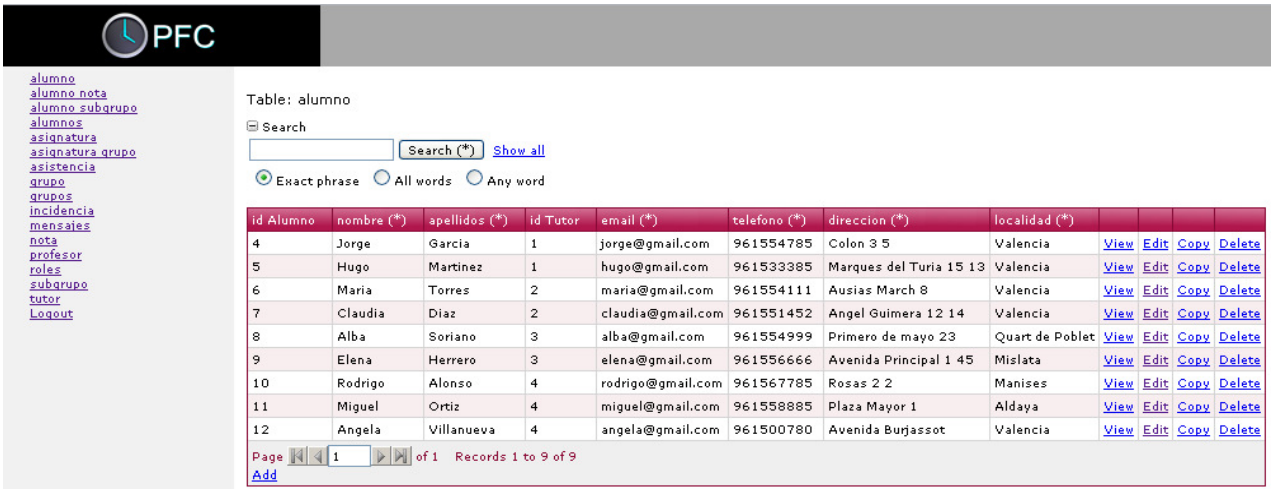
- *StudentsDataGuardian.java:* Se emplea para mostrar los datos correspondientes al tutor del alumno. Para la vista se utiliza el fichero llamado *students_data_guardian_view.xml.*

- *StudentsDataIncident.java:* Muestra las incidencias asociadas a un alumno, así como la ventana emergente que se muestra al pulsar sobre las incidencias. Su vista se encuentra en los ficheros *students_data_incident_view.xml* para crear la interfaz y el fichero *students_data_incident_view_item.xml* para los elementos de la lista.



BACK END WEB

Se ha creado una web para poder modificar los datos en la base de datos, puesto que la aplicación Android no está destinada a dichos fines. Mediante esta web se pueden corregir erratas, por ejemplo en nombres u otros datos de los alumnos, e incluso introducir nuevos datos.



Para poder acceder es necesario un usuario y contraseña, ya que de otra forma cualquier usuario podría hacer uso de la web y acceder para modificar datos relevantes. Así pues, los profesores disponen de un usuario y contraseña que corresponden con el usuario y contraseña de la aplicación móvil, mientras que el director y jefe de estudios disponen de su usuario y contraseña correspondiente. Al igual que con los profesores, el usuario y contraseña es el mismo que en la aplicación móvil. Además de estos usuarios, existe el perfil de administrador con permisos para modificar todos los datos de la base de datos, a diferencia del resto de usuarios.

En primer lugar, los profesores tienen acceso a los datos de los alumnos, asignaturas, tutores y profesores. Este acceso es de solo consulta, de forma que un profesor no pueda modificar datos. Por otra parte, el jefe de estudios y el director disponen de acceso a los mismos datos pero en este caso es posible modificar, insertar y borrarlos, ya que se presuponen mayores privilegios con respecto al profesorado. Por último, el administrador tiene capacidad para modificar, insertar y eliminar datos de cualquier registro almacenado en la base de datos, por lo que en caso de modificar algún dato que el resto de usuarios no pueda modificarlo, habría que recurrir al administrador del sistema para realizar la modificación.

Es importante destacar que a pesar de estos permisos, ninguno de los usuarios tiene capacidad de consultar o modificar los datos asociados al usuario de los profesores o sus contraseñas, de forma que se garantiza la privacidad de los usuarios del sistema.

El hecho de que los datos a los que pueden acceder los profesores, jefe de estudios o director sea limitado se debe al hecho de que si se diese libertad a la hora de modificar los valores, podrían darse casos de datos inconsistentes o pérdida de información, lo que provocaría que la aplicación fallase u omitiese información. Por ejemplo, si se eliminase la relación existente entre un alumno y un grupo en el que está matriculado, al consultar sus datos no se podría acceder a las notas de ese alumno en ese grupo ya que no se reconocería que el alumno sí está en dicho curso.

6. EVALUACIÓN Y PRUEBAS

Para comprobar el correcto funcionamiento de la aplicación se han llevado a cabo diversas evaluaciones y pruebas descritas a continuación.

6.1. EVALUACIÓN

Una vez terminada la implementación se han realizado distintas pruebas para garantizar el correcto funcionamiento, probando cada funcionalidad una a una y corrigiendo aquellas en las que se presentaba algún error, de forma que se ha conseguido depurar la aplicación lo máximo posible con el fin de evitar fallos a la hora de realizar uso del proyecto desarrollado.

6.2. PRUEBAS

Una de las pruebas más importantes era probar la aplicación en funcionamiento en distintos dispositivos, para ello se ha empleado el simulador incorporado en el propio SDK de Android. Mediante el simulador se pueden configurar diversos dispositivos virtuales, de forma que simulen las capacidades técnicas de un dispositivo real.

Así pues se configuraron varios dispositivos virtuales, principalmente para probar tanto si la aplicación funcionaba bajo diversas versiones del dispositivo móvil como para probar la aplicación funcionando en distintos tamaños de pantalla.

Se configuraron cinco dispositivos con distintas versiones del sistema operativo y tres dispositivos con distintos tamaños de pantalla. Los cinco dispositivos con distintas versiones corresponden a las versiones de Android 1.6, 2.2, 2.3.3, 3.0 y 4.0. La aplicación se probó en todas ellas y funcionó correctamente en los cinco casos. Por otro lado, los dispositivos con distintos tamaños de pantalla corresponden a tamaños de pantalla pequeños (240x320), normales (entre 240x400 y 480x854) y pantallas de tablets (entre 1024x600 y 1280x800). Al igual que con las distintas versiones la aplicación funciona y muestra toda la información sin problema. Además de estos dispositivos virtuales se ha probado la aplicación en un dispositivo real, un smartphone HTC Wildfire con Android 2.2 con una pantalla de 240x320, procesador de 528 MHz, 512 MB de ROM y 384 MB de RAM.

A continuación se muestran distintas imágenes de la aplicación desarrollada funcionando en distintas versiones del sistema operativo Android.

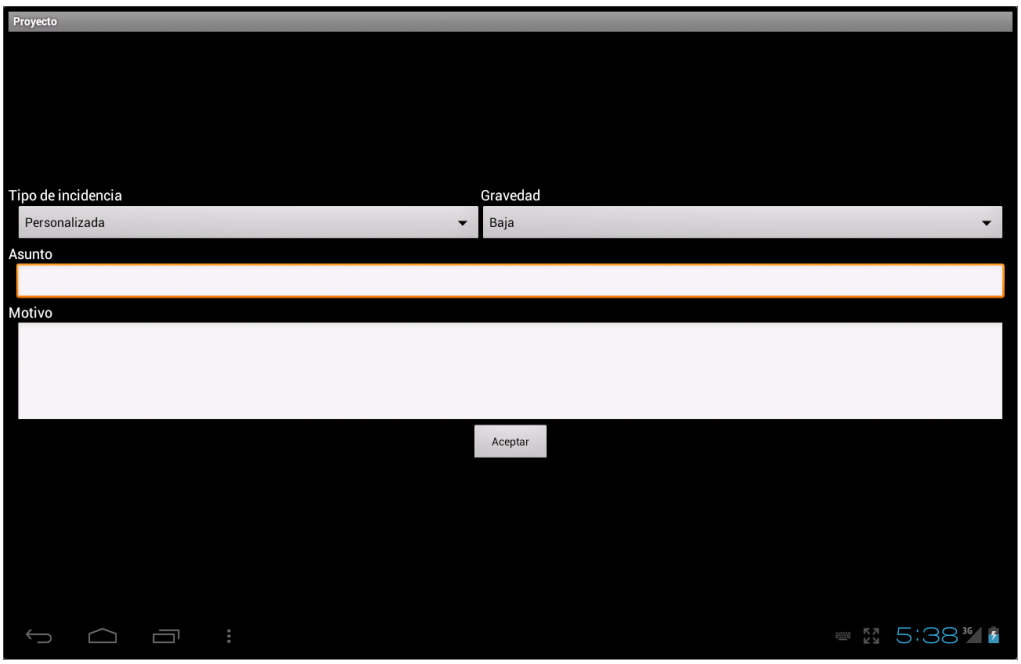
Aplicación funcionando en Android 1.6

Aplicación funcionando en Android 2.2

Aplicación funcionando en Android 2.3.3

Aplicación funcionando en Android 4.0

En las siguientes imágenes, se muestra la aplicación funcionando en dispositivos con distintos tamaños de pantalla.



7. CONCLUSIONES

El último apartado a tratar es el relacionado con las conclusiones del proyecto que se ha desarrollado. Cabe mencionar que el proyecto, a lo largo de todo su desarrollo, ha logrado cumplir con los objetivos y motivaciones que se habían marcado al iniciar el proyecto.

He podido realizar el desarrollo de una aplicación destinada a dispositivos móviles desde cero hasta lograr una aplicación que funciona de la manera esperada. De esta forma, he adquirido conocimientos relacionados con el desarrollo de aplicaciones para dispositivos móviles, en este caso la programación destinada a Android así como el patrón modelo-vista-controlador. Además, se han empleado conocimientos ya aprendidos como el uso de base de datos mediante el sistema de gestión MySQL. Por otro lado, el hecho de tener que desarrollar la aplicación de forma individual, sin un código inicial o referencia ha hecho que consultar diferentes fuentes tales como libros o páginas web haya cobrado gran relevancia a la hora de abordar y solventar diversos problemas o cuestiones surgidos a lo largo del desarrollo.

Una de las etapas más costosas a la hora de desarrollar el proyecto fueron el inicio del desarrollo y la creación de las interfaces. El inicio del desarrollo fue complicado debido a la diferencia existente entre el desarrollo para Android del desarrollo de una aplicación de escritorio usando Java. Esto supuso tener que realizar una adaptación al método de desarrollo para Android con sus peculiaridades, a pesar de esto, al tener conocimientos del lenguaje Java no conllevo graves problemas o estancamientos a la hora de desarrollar, pero sí algo de lentitud a la hora de iniciar el desarrollo. Por otro lado, la creación de las interfaces es dificultosa debido a que la herramienta empleada no está debidamente desarrollada por lo que es costoso ajustar los distintos componentes de la interfaz. Cabe también mencionar la poca optimización a la hora de emplear un dispositivo virtual, lo que lleva a que las pruebas sean un proceso lento debido a la poca fluidez que presenta esta característica del SDK.

Como conclusión mencionar que el proyecto me ha servido para aprender el uso de tecnologías antes desconocidas como es la programación para dispositivos Android, así como para poner en práctica conocimientos adquiridos durante la carrera, como por ejemplo los referentes al uso y gestión de la base de datos.

8. BIBLIOGRAFIA

- Libros
  - Buendía García Félix.
    - Una guía para la realización y supervisión de proyectos final de carrera en el ámbito web.
    - Editorial UPV, 2008.
    - ISBN 978-84-8363-325-0.
  - Burnette, Ed.
    - Hello, Android. Introducing Google's Mobile Development Platform.
    - Editorial Pragmatic Programmers, 2010.
    - ISBN 978-1-934356-56-2.
- Páginas web
  - Android Developers (http://developer.android.com/index.html)
  - PHP: Hypertext Preprocessor (http://www.php.net/)
  - Wikipedia, the free encyclopedia (http://en.wikipedia.org/)
  - Eclipse – The Eclipse Foundation (http://www.eclipse.org/)
  - Apache friends – xampp (http://www.apachefriends.org/es/xampp.html)
  - DBDesigner (http://fabforce.net/dbdesigner4/)
  - StarUML – The Open Source UML/MDA Platform (http://staruml.sourceforge.net/en/)
  - PHPMaker (http://www.hkvstore.com/phpmaker/)
- Documentos
  - IEE Std. 830-1998. IEEE Recommended Practice for Software Requirements Specifications