Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Aplicación de gestión de faltas de asistencia

Grado en Ingeniería Mecánica

ALUMNO: Rafael García Ruiz

DIRECTOR: Norberto Fernández García

CURSO ACADÉMICO: 2016-2017

Universidade Vigo

Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Aplicación de gestión de faltas de asistencia

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval

Cuerpo General

Universidade Vigo

RESUMEN

La gestión de faltas de asistencia es muy importante en la Escuela Naval Militar para conseguir la formación de excelencia que se imparte en ella. Actualmente, esta gestión se realiza en papel, lo cual presenta una serie de carencias, tales como el coste de gestión de los partes en papel o la dificultad de realizar un análisis de los datos así recabados. Para solucionar estas carencias se propone digitalizar el proceso mediante el desarrollo de una aplicación Android con la que los usuarios puedan capturar toda la información contenida en el actual parte de asistencia. La aplicación implementada es configurable, mediante el uso de bases de datos basadas en archivos XML, que permiten que, conforme se avanza en las distintas ventanas de la aplicación, se pueda ir seleccionando los valores correctos de curso, asignatura, profesor, etc. Tras la implementación de la aplicación se llevaron a cabo una serie de pruebas realistas en distintos escenarios para comprobar su correcto funcionamiento, así como encuestas a un grupo de usuarios para recabar información sobre su usabilidad.

PALABRAS CLAVE

Escuela Naval Militar, asistencia, Java, Android, XML, tableta.

AGRADECIMIENTOS

En primer lugar quería agradecer a la Jefatura de Estudios de la ENM el interés mostrado en el trabajo, proporcionando archivos e información relativa a los planes de estudios. También me gustaría agradecer a mis compañeros de la 5ª Brigada su ayuda prestada a la hora de permitir que usara sus nombres en las bases de datos así como rellenando una encuesta de uso de la aplicación.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contextualización

La Real Compañía de Guardias Marinas fue creada hace 300 años y desde ese momento la formación científica, física, marinera y moral de todos y cada uno de los futuros Oficiales de la Armada Española ha sido impartida, con distintos nombres y localización geográfica, por la Escuela Naval Militar. Hoy en día y desde el año 1943, esta se encuentra situada en la ciudad de Marín, en la provincia de Pontevedra.

También conocida como ENM, la Escuela Naval Militar siempre se ha caracterizado por una formación de excelencia multidisciplinaria, pero es a partir del curso académico 2010/2011 cuando comienza junto al Centro Universitario de la Defensa adscrito a la Universidad de Vigo, a impartir un grado en Ingeniería Mecánica con Intensificación Naval. Todo esto sin dejar de lado el intenso adiestramiento en el ámbito físico y militar, tanto con clases teóricas como prácticas.

En la Escuela Naval Militar se imparte un gran número de planes de estudios, abarcando los distintos cuerpos de la Armada Española y pudiendo tener o no titulación previa. Pero la mayor parte de los alumnos son aquellos que entran sin titulación de los Cuerpos General y de Infantería de Marina.

Figura 1-1. La Escuela Naval Militar [1]

1.2 La asistencia a clase en la ENM

La presencia del alumno en clase es una parte fundamental del proceso de aprendizaje de este, así como la participación activa del alumno en el desarrollo de la asignatura y el trabajo continuo. Cuando un estudiante no asiste al aula está perdiendo no solo las explicaciones que se dan, sino también las posibles dudas, debates en conjunto con el resto de compañeros, práctica de la escucha activa, etc. Además, recuperar el tiempo de ausencia no es nada fácil y requiere un mayor esfuerzo el aprendizaje autónomo que el guiado por un profesor.

Pero la asistencia a clase no es sólo importante en cuanto al estudio o trabajo que tiene que realizar el alumno, sino que también tiene efectos en la evaluación. En la mayoría de centros universitarios los alumnos tienen que superar un porcentaje de asistencia en ciertas asignaturas para poder ser calificados, pudiendo darse el caso de no superar una asignatura por faltar a clase un número de veces mayor al permitido.

Además, existen multitud de artículos y estudios que relacionan la asistencia a clase con el rendimiento académico. Destacando las conclusiones de [2], se pueden llegar a las siguientes reflexiones:

- Una gran cantidad de los alumnos que presentan una gran inasistencia ofrece un rendimiento académico considerado como deficiente.
- La falta de asistencia tiene, en la gran mayoría de los casos, como consecuencia final la deserción escolar, ya que el estudiante al interrumpir de manera constante el proceso educativo acaba por alejarse de este de forma definitiva.

Figura 1-2. Importancia de la asistencia a clase. [3]

Al ser un centro docente militar, la Escuela Naval Militar cumple unas ciertas peculiaridades en cuanto a la asistencia a clase de los alumnos, que la diferencian de otros centros y ayudan a alcanzar y mantener ese objetivo de formación de excelencia.

En la ENM un alumno no puede faltar a ninguna clase, siendo la asistencia obligatoria salvo en casos excepcionales como pueda ser estar de guardia o encontrarse enfermo. En la gran mayoría de centros en los que se imparte una educación universitaria la asistencia a clase no es obligatoria salvo en algunas asignaturas especifícas. Como ya se ha dicho, no es así en la Escuela Naval Militar y el principal motivo es el exigente plan de estudios que no permite perder horas de clase para poder alcanzar los objetivos lectivos. A pesar de esto, hay ciertas ocasiones en las que es inevitable la ausencia de un alumno, debido a esto en la ENM se impone un máximo número de faltas para los alumnos en clases prácticas tanto de la formación civil como de la militar. Asimismo, si alguno de los estudiantes no cumple con el mínimo de créditos en las asignaturas de Instrucción y Adiestramiento automáticamente pasará a repetir curso.

HOJA DE CONTROL DE FALTAS DE ASISTENCIA. CURSO 2016/2017

Escala Cuerpo Curso Grupo Fecha

HORA	ASIGNATURA	ALUMNOS FALTOS (1)	MOTIVO(2)	
1				PROFESOR FIRMA
2				PROFESOR FIRMA
3				PROFESOR FIRMA
4				PROFESOR FIRMA
5				PROFESOR FIRMA
6				PROFESOR FIRMA
7				PROFESOR FIRMA
8				PROFESOR FIRMA

***PONER ÚNICA Y EXCLUSIVAMENTE LOS MOTIVOS RELACIONADOS EN EL PUNTO (2)**

ALUMNO JEFE DE GRUPO:_____
(empleo, apellidos)

NOTAS:

1. Nº de clase, apellidos y nombre de los alumnos ausentes.
2. Código de motivos;(H) hospitalizado, (RM) revista médica, (P) permiso, (C) comisión, (RD) rebajado en domicilio, (R) rebajado de ejercicio y/o instrucción, (G) guardia, (T) tutoría, (CV) convalidada.
3. Los jefes de grupo entregarán esta Hoja de Control al Brigadier de Guardia que a su vez se la entregará al Profesor de Servicio.
4. El Profesor de Servicio la presentará junto co n el Parte de Situación de Alumnos al Jefe de Estudios.

Figura 1-3. Actual parte de clase de la ENM

1.3 Control de la asistencia a clase en la ENM

Debido a la importancia de la asistencia a clase en la Escuela Naval Militar es necesario realizar un control exhaustivo y diario de la presencia de los alumnos en todas y cada una de las horas lectivas.

Hoy en día este control se realiza mediante la firma de los profesores en el parte de clase como se puede ver en la Figura 1-3. Este parte de clase es una hoja tamaño A5 que tiene una fila destinada para cada una de las 8 horas lectivas que tiene el horario de la ENM, en las que el alumno "Jefe de Grupo" deberá escribir el nombre de los componentes de su grupo que no asistan a dicha clase y el motivo de la falta. Los motivos justificables de faltar a clase son muy reducidos y vienen contemplados en el propio parte: Hospitalizado (H), Revista médica (RM), Permiso (P), Comisión (C), Rebajado en Domicilio (RD), Rebajado de ejercicio y/o instrucción (R), Guardia (G), Tutoría (T) y Convalidada (CV).

El anteriormente nombrado Jefe de Grupo es un alumno de cada uno de los grupos de los distintos cursos de la ENM. Los grupos eligen internamente a este Jefe de Grupo que va rotando según el propio criterio de los alumnos. Con esta figura no solo se consigue tener un encargado de controlar la asistencia a clase del resto de integrantes del grupo, sino que además se le otorga una cierta responsabilidad frente al resto de alumnos y permite el ejercicio del mando hasta en las actividades más cotidianas del día a día de la Escuela Naval Militar.

A pesar de las ventajas expuestas anteriormente del actual sistema de control de asistencia en la Escuela Naval Militar, el paso a la tecnología parece necesario e incluso obvio en la era en la que nos encontramos. Es bien sabido que un gran número de prestigiosos centros de formación Universitaria cuentan con este tipo de sistemas, de manera que con el objetivo de seguir perteneciendo a este selecto grupo la ENM debe comenzar cuanto antes esta apuesta por la modernización del sistema de control de asistencia.

1.4 Objeto de este Trabajo

Actualmente se usan a diario en la Escuela Naval Militar un total de 55 partes de clase, lo que hace una cantidad aproximada de 1100 partes al mes. Esto es porque cada grupo debe entregar un parte distinto, lo cual genera un gasto en papel y residuos excesivo. En el presente trabajo se pretende reducir de gran manera este gasto a la vez que facilitar la labor de los alumnos encargados del control de asistencia de sus compañeros de grupo. Para ello se pretende crear una aplicación Android que realice la misma función que el parte de clase, pero de una manera aún más sencilla y visual. Para esto cada grupo debería portar un dispositivo de tipo tableta en el que estaría instalada la aplicación.

Esta digitalización del modelo que se puede ver en la Figura 1-3 tiene como objetivos principales:

- Modernización del método empleado en la Escuela Naval Militar.
- Eliminar la dependencia del papel.
- Disminuir el enorme gasto en papel que se realiza día a día en la ENM, así como el gasto de personal encargado en procesar todo este papel.
- Al estar los datos en formato electrónico se facilita su posterior procesado y consulta.

1.5 Estructura de la memoria

En la presente memoria se procederá a explicar todo lo relacionado con el Trabajo de Fin de Grado: Aplicación de gestión de faltas de asistencia.

Para ello se procede a una división en 6 capítulos. En el primer capítulo se presentará la introducción así como la motivación y objeto de este Trabajo. En el segundo, el Estado del Arte será

expuesto, hablando de otros trabajos en el mismo ámbito que este Proyecto y de las herramientas utilizadas para la realización de éste. El tercer capítulo versará acerca del desarrollo del Trabajo dejando constancia de todas y cada una de las decisiones tomadas en la consecución de los objetivos. A continuación, en el cuarto capítulo se observarán los resultados obtenidos y las pruebas realizadas, dando lugar al quinto capítulo, en el que se presentarán las conclusiones, dando una valoración objetiva del éxito o no del Proyecto, tras lo cual se propondrán una serie de posibles líneas futuras en relación a este ámbito, buscando siempre la mejora continua. Finalmente, en el sexto y último capítulo se expondrá la bibliografía usada.

Tras los apartados anteriormente descritos irán incluidos anexos con los recursos usados durante la realización del Proyecto.

2 ESTADO DEL ARTE

2.1 Aplicaciones existentes en el mercado

A continuación se expondrán una serie de aplicaciones existentes actualmente en el mercado con objetivos similares a los del presente proyecto, procediendo a la explicación de su funcionamiento.

2.1.1 Inika

Inika es un software de gestión académica y administrativa basado en una aplicación Web para centros de enseñanza. Sus principales características son:

- Puede ser adaptado a centros de Educación Primaria, Preescolar, Educación Secundaria, Bachiller, Formación profesional o Formación para adultos.
- Trabajo en línea en múltiples idiomas.
- Pueden acceder tanto los alumnos, como los profesores y los padres a través de Internet, pero cada uno tiene acceso a distintas funciones. Los profesores pueden poner mensajes, meter notas y faltas de asistencia, los padres pueden mantener comunicación con los tutores y comprobar calificaciones de sus hijos, mientras que los alumnos solo tienen acceso a las notas.
- Incluye un módulo SMS para mantener puntualmente informados a los padres que así lo deseen.
- Los principales aspectos funcionales son: gestión de alumnos, profesores, padres, notas, faltas, boletines o estadísticas.
- Se pueden generar listados, boletines y actas. Para los usuarios avanzados dispone de un creador SQL (*Structured Query Language*, Lenguaje de Consulta Estructurado) para generar consultas y gráficos dinámicos. Se pueden ajustar a las necesidades personales o crear de cero.

Para la configuración de esta aplicación se comienza por el año académico, siguiendo por los cursos, asignaturas, profesores y alumnos. Hay que configurar todos los parámetros para que el sistema funcione y se puede adaptar a las necesidades del centro de enseñanza. Una vez configurado el centro los profesores pueden introducir notas y faltas así como comunicarse con las familias. [4]

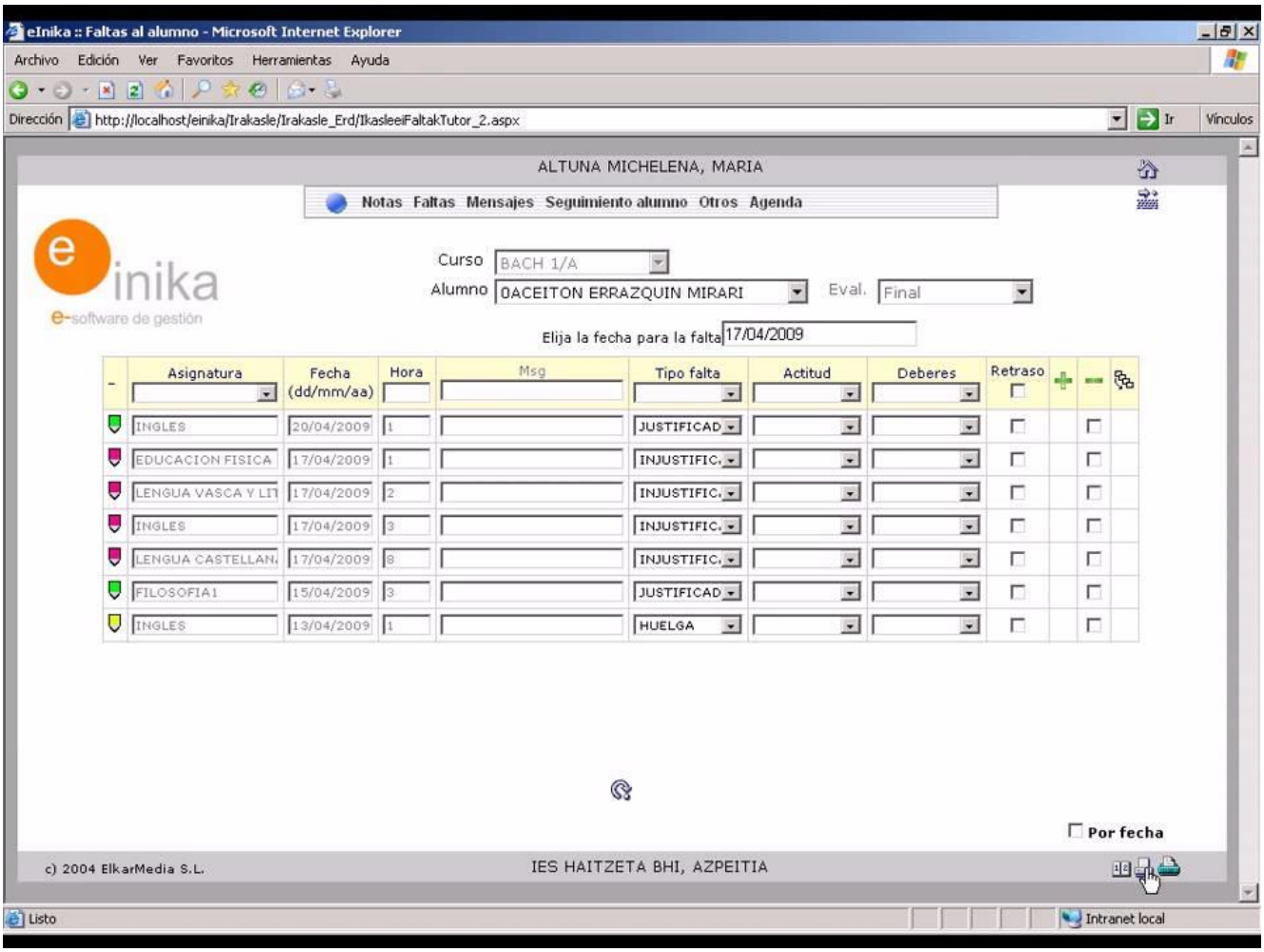

Figura 2-1. Pantalla de faltas de asistencia [4]

Esta aplicación no nos sería de utilidad para la consecución del objeto de este Proyecto ya que lo que se busca solo es una manera de gestionar el control de las faltas de asistencia, mientras que Inika nos ofrece más funciones de las que se necesitan. Además es una aplicación web, lo que no nos interesa puesto que no en todas las aulas se dispone de acceso a Internet. Adicionalmente, a diferencia del propósito de este proyecto, las faltas de asistencia son gestionadas por los profesores. Otro factor a tener en cuenta es el precio, que exige la compra de una licencia anual para su uso.

2.1.2 Alexia Educación

Alexia es una plataforma de gestión y comunicación presente en más de 1200 colegios, situándose como uno de los referentes en cuanto a servicios de apoyo a la enseñanza. Se basa en 3 pilares básicos:

- Gestión: Se busca una gestión adaptada al centro, flexible, gestión de transporte, comedor, actividades extraescolares, seguimiento del alumno, informes y listados personalizables, etc.
- Comunicación: Esta se realiza a través de la Web, permitiendo una comunicación en tiempo real a través de múltiples canales: Web, SMS, correo electrónico, etc.
- Integración: Se conecta con herramientas de aprendizaje, horarios, bibliotecas, lo que permite una gestión mucho más eficiente. Además permite extraer información para incluirla en otras herramientas.

Entre sus principales características podemos citar:

- Gestión integral.
- Comunicación global.

- Evolución.
- Usabilidad.
- Seguridad.
- Flexibilidad.
- Herramientas complementarias.
- Adaptación.

Además hace referencia a las herramientas integradas de las que dispone este servicio, como son un Entorno de aprendizaje (Xtend), un generador de horarios (Untis), una herramienta de gestión contable (Asesor), gestión de bibliotecas (SophiA) y gestión de calidad (ISOTools). [5]

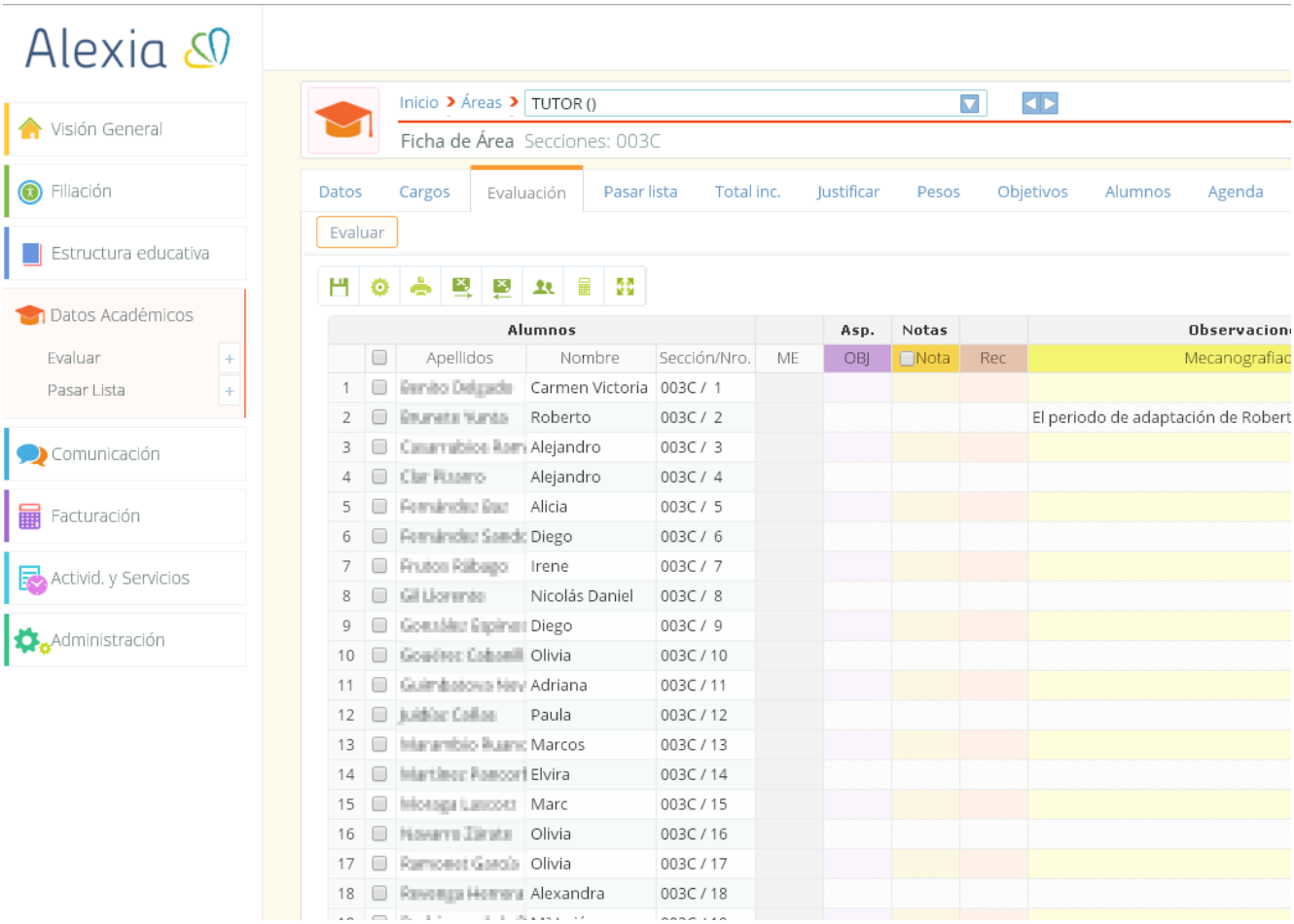

Figura 2-2. Ejemplo de uso de Alexia. [5]

Esta aplicación no permite explicar el motivo de las faltas de asistencia por lo que no sería útil para nuestro caso. También cabe destacar que esta aplicación está orientada a centros de enseñanza en los que se informe a los padres de los alumnos de las calificaciones y demás información de interés, esto no es algo que se busque en la ENM ya que todos los alumnos son mayores de edad y no existe esa necesidad de informar a los padres. Al igual que con el anterior producto requiere de una licencia anual para poder usarlo.

2.1.3 DocCF

DocCF es un Software de Gestión Escolar creado por la empresa Grupo CF Developer. Permite un control escolar, académico y administrativo para todo tipo de centros de enseñanza, desde institutos hasta jardines infantiles. Se caracteriza por una gran flexibilidad y adaptabilidad y está disponible tanto en Latinoamérica como en España.

Entre la gran cantidad de procesos que es capaz de gestionar este producto, destaca el control de procedimientos académicos y administrativos como pueden ser: la matriculación, asignación de horarios, registro de calificaciones, generación de boletines o gestión de faltas de asistencia. Además optimiza los procesos de comunicación entre cargos directivos, profesores, alumnos y familias.

Todas las funciones de DocCF se encuentran organizadas por módulos, lo que permite adaptar la organización del centro educativo a las necesidades que se tengan. Por este motivo la aplicación DocCF podría ser una opción para realizar la función que requiere la Escuela Naval Militar y que es el objeto de este Proyecto. Sin embargo, a pesar de que se podría contratar solo el módulo de gestión de faltas de asistencia, diferenciando así este producto del resto que se ha mencionado, dicho módulo no permite personalizar el tipo de falta, algo clave en el caso de querer implementarlo en la ENM. Además, esta aplicación trabaja igualmente a través de la Web y requiere conexión continua a Internet para funcionar. [6]

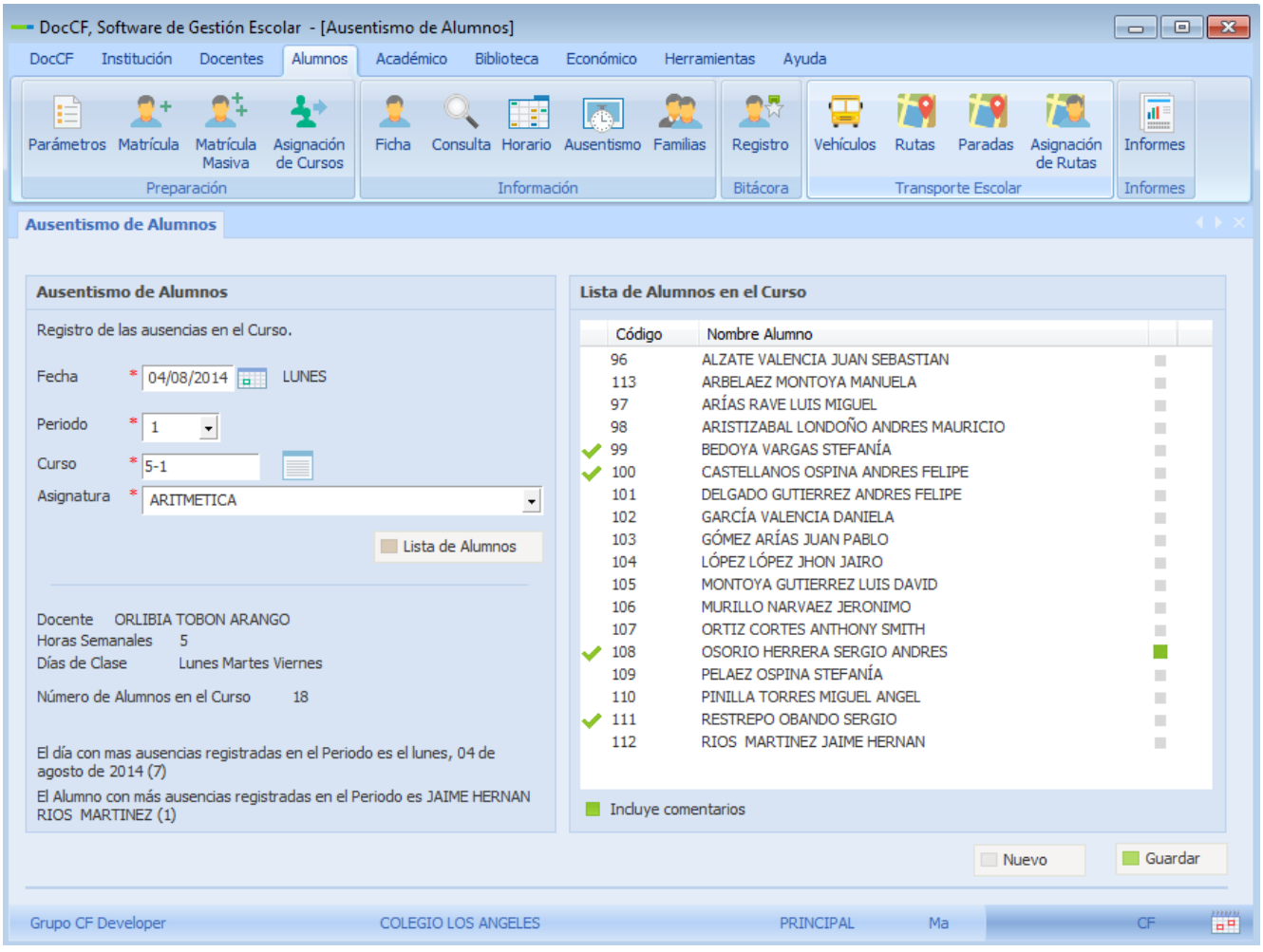

Figura 2-3. Módulo de Control de Asistencia de DocCF. [6]

2.1.4 ITeacherBook

ITeacherBook es una aplicación creada por la empresa iEstudiez que sólo está disponible para dispositivos con sistema operativo iOS. Esta aplicación permite realizar gestión de horarios, control de asistencia y calificaciones de alumnos. Todo ello orientado a dispositivos portátiles como teléfonos móviles o tabletas. [7]

Esta aplicación permite, entre otras cosas:

- Planear las clases para una mejor gestión del aprendizaje.
- Importar información de los estudiantes por múltiples vías, a través de base de datos o incluso desde los propios contactos del dispositivo.
- Mandar tareas a los alumnos.
- Control de asistencia. Muestra gráficos con estadísticas de asistencia a clase y marca en el calendario los días que haya faltado o llegado tarde a alguna clase un alumno.
- Introducir las calificaciones de los estudiantes.
- Programar notificaciones.
- Optimizar al gusto, ofreciendo multitud de opciones de diseño.

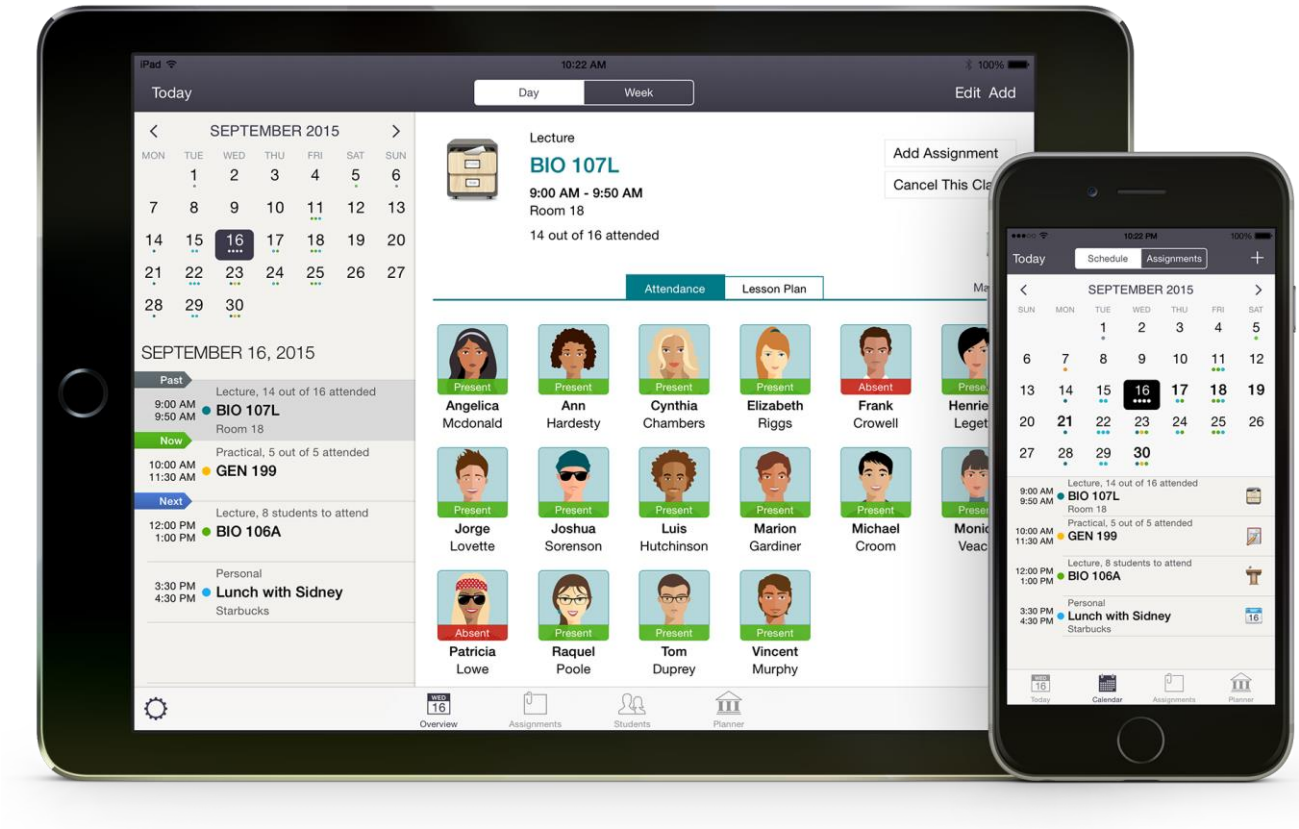

Figura 2-4. Actividad de asistencia a clase de iTeacherBook. [7]

Este producto, a diferencia de las anteriores, está orientado a dispositivos portátiles que no requieren conexión continua a Internet, sin embargo no consigue cumplir todas las exigencias de la Escuela Naval Militar. Esta herramienta está pensada para que sea el profesor el que la maneje, mientras que, por particularidades de la ENM, lo que se pretende en este Proyecto es que sea el alumno el responsable de su buen uso y manejo.

2.1.5 Teacher Aide

Esta aplicación para telefónos móviles o dispositivos tableta está disponible tanto para Android como para iOS. Tiene dos versiones, una gratuita y otra de pago, la primera solo incluye control de las calificaciones de los alumnos, por lo que sería necesario obtener la versión de pago para poder contar con la gestión de faltas de asistencia. [8]

Las características que ofrece la aplicación son:

- Control de faltas de asistencia, permitiendo distinguir entre distintos motivos.
- Gestión de las calificaciones obtenidas por los alumnos.
- Agenda para organizar las clases.
- Pantalla interactiva para controlar la posición de los alumnos en clase.

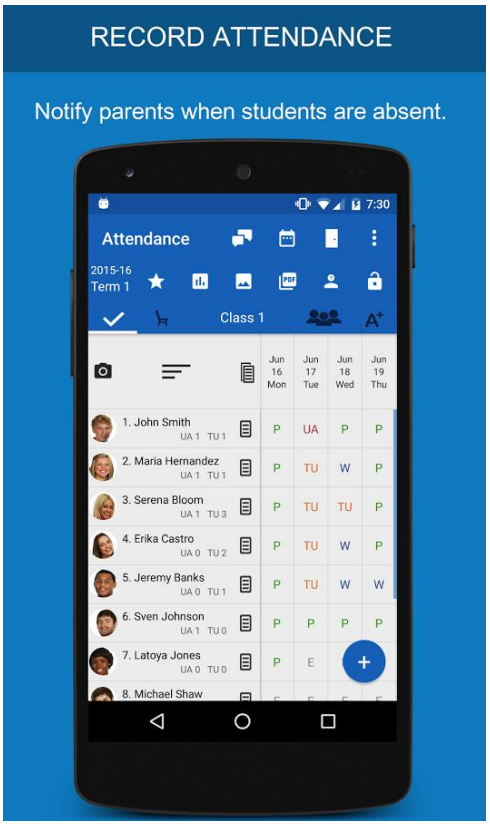

Figura 2-5. Control de asistencia en Teacher Aide. [8]

Al igual que la anterior aplicación, está orientada al control de la asistencia por parte del profesor, motivo principal por el que no serviría en la Escuela Naval Militar.

2.1.6 MIDS

A diferencia de los sistemas y aplicaciones expuestos anteriormente, este está pensado y adaptado al mundo militar. La Escuela Naval de Annapolis de la *United States Navy* (USNA) cuenta con este sistema.

El nombre, MIDS, proviene del diminutivo de *Midshipmen* (Guardiamarinas), empleo de los alumnos de la USNA, pero también proviene de *Midshipmen Information System* (Sistema de Información de Guardiamarinas). Es una herramienta informática que permite gestionar las notas, trabajos, correos y faltas de asistencia de los alumnos.

En cuanto al control de asistencia, presenta una interfaz sencilla, en la que al profesor se le muestra por pantalla una tabla con el nombre de los alumnos que tienen que asistir a clase, junto con un

desplegable que permite elegir los distintos motivos de retrasos o faltas. Además también permite visualizar fotos de los alumnos.

Esta aplicación permite gestionar la asistencia por parte del profesor, que como ya se ha dicho es algo que no nos interesa, puesto que el objetivo es que sea el alumno el encargado de llevar este control. Además este sistema, sería imposible de instalar en la Escuela Naval Militar puesto que es exclusivo de la USNA.

2.2 Herramientas usadas en este Proyecto

A continuación se procederá a explicar las herramientas utilizadas durante el presente Proyecto. La programación se ha realizado en Java, más concretamente orientada hacia Android y el programa usado para realizar la aplicación es Android Studio.

2.2.1 Java

Todo nace del equipo de ingenieros de Sun Microsystem (Actualmente Oracle) que, encabezados por Bill Joy, trabajaban en un proyecto llamado "Green" para desarrollo de aplicaciones destinadas a una amplia variedad de periféricos y sistemas transportables (en particular teléfonos móviles y televisores interactivos). En un principio optaron por desarrollar en C++ que había demostrado sus buenas capacidades en programación orientada a objetos (POO), pero pronto se dieron cuenta de los límites que este lenguaje ofrecía.

Debido a esto llegaron a la conclusión de que era más rentable crear un nuevo lenguaje alrededor de una nueva plataforma de desarrollo. A esto se dedicaron varios de los ingenieros de Sun: James Gosling, Patrick Naughton y Mike Sheridan que pueden verse en la Figura 2-6. Para la creación del lenguaje se inspiraron en otros tales como C++, Objective C, Ada o Perl; dando como resultado una plataforma idónea para el desarrollo de aplicaciones en diversos dispositivos.

El primer nombre que recibió fue el de C++-- (C++ sin los problemas que este ofrecía), pasando luego a llamarse Oak y finalmente Java. Y así es como en 1991 nació el lenguaje Java. [9]

Como se ha mencionado anteriormente, Java nace por la necesidad de suplir una serie de carencias y cumplir unos objetivos. Este lenguaje de programación cubre las siguientes necesidades:

- Sintaxis sencilla y orientada a objetos.
- Lenguaje interpretado que optimiza el tiempo y el ciclo de desarrollo (compilación y ejecución).
- Las aplicaciones pueden ser usadas en un gran número de plataformas y dispositivos sin necesidad de modificación.
- Aplicaciones resistentes, ya que el motor de ejecución de Java (*Java Runtime Environment o JRE*) se encarga de gestionar la memoria y es más fácil gestionar los errores.
- Aplicaciones gráficas eficientes debido a la puesta en marcha y asunción de funcionamiento de algunos procesos ligeros o hilos (*Thread* y *multithreading*).
- Funcionamiento seguro, ya que el entorno Java introduce restricciones que reducen la probabilidad de realizar ninguna operación o modificación peligrosa.

Pero Java no es solo un lenguaje de programación, sino también una plataforma de desarrollo. [9]

2.2.1.1 Java como lenguaje de programación

Como puede verse en [9], Java se caracteriza como lenguaje de programación:

"[…] *Java como un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, securizado, independiente de las arquitecturas, portable, eficaz, multihilo y dinámico."*

A continuación se procede a explicar las características más destacadas:

- Sencillo: Sólo existen tres tipos primitivos, autogestión de la memoria, no hay preprocesadores ni ficheros de encabezamiento, herencia simple para evitar problemas de ambigüedad, no hay punteros sino referencias a objetos.
- Orientado a objetos: En Java, a excepción de los tipos primitivos, todo es un objeto. Incluso existen clases ya incorporadas que encapsulas los primitivos. La programación orientada a objetos proporciona un mejor dominio de la complejidad, es más fácil reutilizar y hace que las correcciones sean más sencillas.
- Interpretado: Un programa Java no es ejecutado, sino interpretado. Esto hace que sea más lento, pero evita tener que editar los enlaces a librerías antes de ejecutar el programa. Por lo tanto, solo existen dos fases en Java: compilación y ejecución.
- Robusto: Java es un lenguaje muy estricto, por ejemplo, las variables han de ser siempre explícitamente declaradas. Además, el código es verificado en el momento de la compilación y en el de la ejecución, evitando así errores.
- Securizado: El motor de ejecución en Java (JRE) es el que se encarga de vigilar la seguridad de las aplicaciones y los sistemas.
- Independiente de las arquitecturas: El compilador genera *bytecode*, que es un lenguaje binario independiente de arquitecturas, sistemas operativos y dispositivos de gestión de interfaz gráfica de usuario (GUI, *Graphic User Interface*).
- Multihilo: Java permite tener en ejecución varios hilos de manera simultánea, con el fin de aumentar la velocidad de la aplicación.

2.2.1.2 Java como plataforma de desarrollo

Una plataforma es un entorno, físico o virtual, que permite la ejecución de un programa. Una gran parte de estas plataformas se compone de un elemento de hardware y otro de software. Java se distingue por ser un software compatible con multitud de plataformas físicas y sistemas operativos. [9]

La máquina virtual Java es la base de la plataforma Java. Sin ella los programas no podrían ser ejecutados. La JVM (*Java Virtual Machine*) se encarga de:

- Cuando un programa invoca un objeto o llama a miembros de una clase, su función es cargar el *bytecode* a interpretar.
- Se encarga totalmente de la gestión de los punteros, y por tanto de cada referencia hecha a un objeto. Se libera automáticamente la memoria en el momento en que ninguna variable le hace referencia
- De la seguridad: Al cargar el programa comprueba que no ocurra ningún error como llamar a memoria no inicializada, conversiones ilegales o manipulación de punteros de memoria.
- Interacción con código nativo (por ejemplo C/C++). Esto puede usarse para accesos a periféricos o funciones no implementadas en Java.

La API (*Application Programming Interface,* Interfaz de programación de aplicaciones*)* Java es una colección de componentes de software que aportan una gran cantidad de funcionalidades. Se organiza en paquetes, y estos en clases e interfaces, disponiendo de alrededor de 4000 clases e interfaces. La plataforma Java proporciona las APIs básicas, pero en el sitio Java de Oracle se encuentran disponibles numerosas extensiones. Las APIs se dividen en tres categorías: Las APIs básicas, las APIs de acceso a datos e integración con datos existentes y las APIs de gestión de la interfaz usuario. [9]

Figura 2-6. Gosling, Naughton y Sheridan, principales desarrolladores de Java.

2.2.2 Android

Android es un sistema operativo y una plataforma de software, basado en Linux y pensado sobre todo para teléfonos móviles. Pero este SO también puede verse en tabletas, *netbooks*, reproductores de música e incluso ordenadores. Android permite programar en un *framework* (entorno de trabajo) de Java y su principal característica es que cualquier persona que tenga los suficientes conocimientos de programación puede crear nuevas aplicaciones, *widgets* (pequeñas aplicaciones que facilitan el acceso a funciones frecuentes) o incluso modificar el propio sistema operativo ya que Android es de código libre. Por lo tanto lo único necesario para programar en esta plataforma es conocer el lenguaje Java. [10]

Figura 2-7. Logo de Android.

Fue creado por Android Inc., empresa que compró Google en 2005, pero no fue hasta 3 años más tarde cuando se popularizó, gracias al proyecto de Open Handset Alliance, una asociación de 48 empresas para desarrollo de hardware, software y telecomunicaciones, en el que se decidió promover este software libre. La mayor parte del código fuente ha sido publicado por Google con licencia Apache, versión 2.0.

Android puede acceder a recursos como controladores de pantalla, cámara o memoria flash gracias a que se encuentra en una capa por encima del *kernel* (núcleo) de Linux. En la Figura 2-8 pueden verse cada una de las distintas capas en las que trabaja Android: el propio *kernel* desde donde puede acceder a los controladores, las librerías, la organización de los administradores de recursos y la capa de aplicaciones. [10]

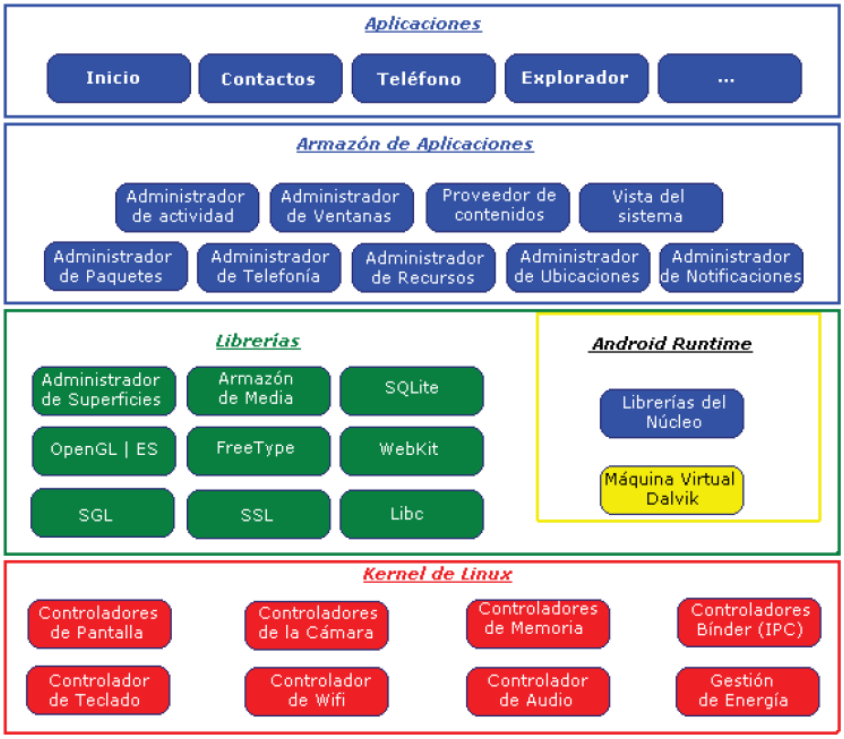

Figura 2-8. Capas en las que trabaja Android. [10]

Como puede verse en la Figura 2-8, las capas en las que trabaja Android son:

- Aplicaciones: Existen una cierta cantidad de aplicaciones base entre las que se incluye un cliente de correo electrónico, SMS, calendario, teléfono, mapas, navegador de Internet, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- Armazón de aplicaciones: Los desarrolladores tienen acceso completo a las mismas APIs del *framework* que usan las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes. Además, cualquier aplicación puede publicar sus capacidades, por lo que otra aplicación puede luego hacer uso de éstas. Este mecanismo también permite que el usuario pueda reemplazar componentes.
- Librerías: Android cuenta con un conjunto de librerías de C/C++ que son usadas por varios componentes del sistema. Entre otras destacan: *System C library* (librería C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQL.
- Android Runtime: Incluye un conjunto de librerías base que aportan la mayor parte de las funciones disponibles del lenguaje de programación Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Esta permite que un dispositivo pueda tener activas múltiples máquinas virtuales de forma eficiente.
- *Kernel* de Linux: Android depende de Linux para servicios como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El *kernel* también actúa como una capa intermedia entre el hardware y el resto de la pila de software.

La empresa Symantec de sistemas de seguridad publicó un informe [11] en el que analiza las plataformas móviles con sistema operativo Android e iOS. Centrándonos en las conclusiones sacadas acerca de Android, podemos destacar que Google ha optado por un modelo de certificación menos riguroso que iOS, lo cual permite que cualquier desarrollador pueda crear y lanzar aplicaciones de

forma anónima. Esto desemboca en el creciente volumen actual de software malicioso existente para Android.

Además, Android otorga a las aplicaciones mucho más control sobre las funciones del dispositivo y deja en manos del usuario la decisión de otorgar o no dichos permisos. Estso permite a los desarrolladores crear aplicaciones más complejas y útiles, pero a la vez deja demasiadas decisiones de seguridad en manos del cliente, lo que provoca una mayor exposición a riesgos. Los dispositivos con *jaibreak* (liberados), o aquellos cuya seguridad ha sido deshabilitada, son un atractivo blanco para atacantes, ya que hoy en día la mayor parte de los usuarios de *smartphones* (teléfonos inteligentes) tienen en estos una gran cantidad de información personal.

Desde su lanzamiento Android ha ido evolucionando, pasando por diferentes versiones en las que se incluyeron distintas funcionalidades:

- Android 1.6 Donut: En esta versión se introduce por primera vez el conocido cuadro de búsqueda rápida de Android, también permite el desarrollo del sistema operativo en dispositivos con distintas resoluciones de pantalla y se introducen cambios en el Android Market (predecesor de Google Play) para hacerlo más accesible a los usuarios.
- Android 2.1 Eclair: Introduce Google Maps Navigation, permite personalizar la pantalla de inicio y se lanzó el software de Síntesis de voz, que aparecía en los teclados representado con el icono de un micrófono.
- Android 2.2 Froyo: Teléfonos muy rápidos controlables por voz. Además, gracias a los puntos de acceso Wi-Fi, se podía tener acceso a Internet en cualquier lugar.
- Android 2.3 Gingerbread: Ofrecía una interfaz más sencilla tanto para el usuario como para el desarrollador. Los juegos sufrieron una gran mejora y la duración de la batería era mayor.
- Android 3.0 Honeycomb: Se optimizó el diseño para tabletas usando patrones de diseño más grandes, se sustituyen la mayoría de botones por controles de navegación en pantalla y se desarrollan los ajustes rápidos.
- Android 4.0 Ice Cream Sandwich: Aún más personalización en la pantalla de inicio, permitía controlar el uso de datos y surge Android Beam, que consistía en un método de compartir contenido al instante entre dos dispositivos Android a través de NFC (*Near Field Communication*, Comunicación de campo cercano).
- Android 4.1 Jelly Bean: Google Now ofrece la información que necesita el usuario en el momento preciso, como información meteorológica, notificaciones accionables, etc. Ofrecía la posibilidad de que hubiera más de un usuario en un mismo dispositivo.
- Android 4.4 Kit Kat: Surge OK Google para realizar comandos y controlar aún más funcionalidades del dispositivo por voz, diseño envolvente de la pantalla y teléfono inteligente que destaca los contactos a los que más llamadas se realizan o busca sitios cercanos.
- Android 5.0 Lollipop: Se desarrolla Material Design que renueva el desplazamiento por el dispositivo haciéndolo más sencillo, la Multipantalla permite cambiar rápidamente el contenido del teléfono a la tableta, o reloj Android o Android TV. Además las notificaciones se muestran en la pantalla de bloqueo de forma organizada.
- Android 6.0 Marshmallow: Acceso más rápido y sencillo a Google Now, permite activar y desactivar permisos de aplicaciones en cualquier momento y optimiza el uso de la batería.
- Android 7.0 Nougat: Permite disponer de más de un idioma a la vez, cambiar de aplicación con un doble toque o utilizar dos a la vez. También incluye una mejora de rendimiento para graficos en 3D. [12]

2.2.3 Android Studio

Android Studio es el IDE (*Integrated Development Environment*, Entorno de desarrollo integrado) oficial para Android. Proporciona herramientas para crear aplicaciones en cualquier dispositivo Android. Permite editar códigos fuente, depurarlos, cuenta con herramientas de rendimiento y un sistema de compilación flexible e instantáneo.

Permite escribir código proporcionando un tiempo reducido de respuesta en el flujo de trabajo. La función Instant Run aplica cambios en el código o en los recursos de la aplicación en ejecución y permite visualizar los cambios sin necesidad de reiniciar la aplicación. Contiene un editor de código inteligente que analiza y proporciona sugerencias, buscando escribir un código eficiente y productivo. Además gracias a Android Emulator se pueden probar las aplicaciones en cualquier configuración de dispositivo sin necesidad de disponer físicamente de este.

La estuctura de proyectos y el mecanismo de compilación basado en Gradle de Android Studio proporcionan flexibilidad para generar APK (*Android Application Package*, Paquete de aplicación de Android) para toda clase de dispositivos. Ofrece sistemas de compilación automatizados, administración de dependencias y configuraciones de compilación personalizables. Proporciona un entorno unificado en el que se pueden desarrollar aplicaciones para teléfonos móviles, tabletas, Android Wear, Android TV y Android Auto. Integra herramientas de control de versión para mantener el software actualizado.

Android Studio dispone de una gran cantidad de plantillas de código con patrones establecidos como paneles laterales de navegación o un paginador de vistas e incluso permite importar aplicaciones totalmente funcionales desde GitHub (plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git).

Dispone de herramientas GUI que simplifican el desarrollo de aplicaciones. Permite seleccionar iconos de diseño de *material design* proporcionados por Google o cargar archivos propios. Además proporciona una vista única de todos los recursos traducidos. [13]

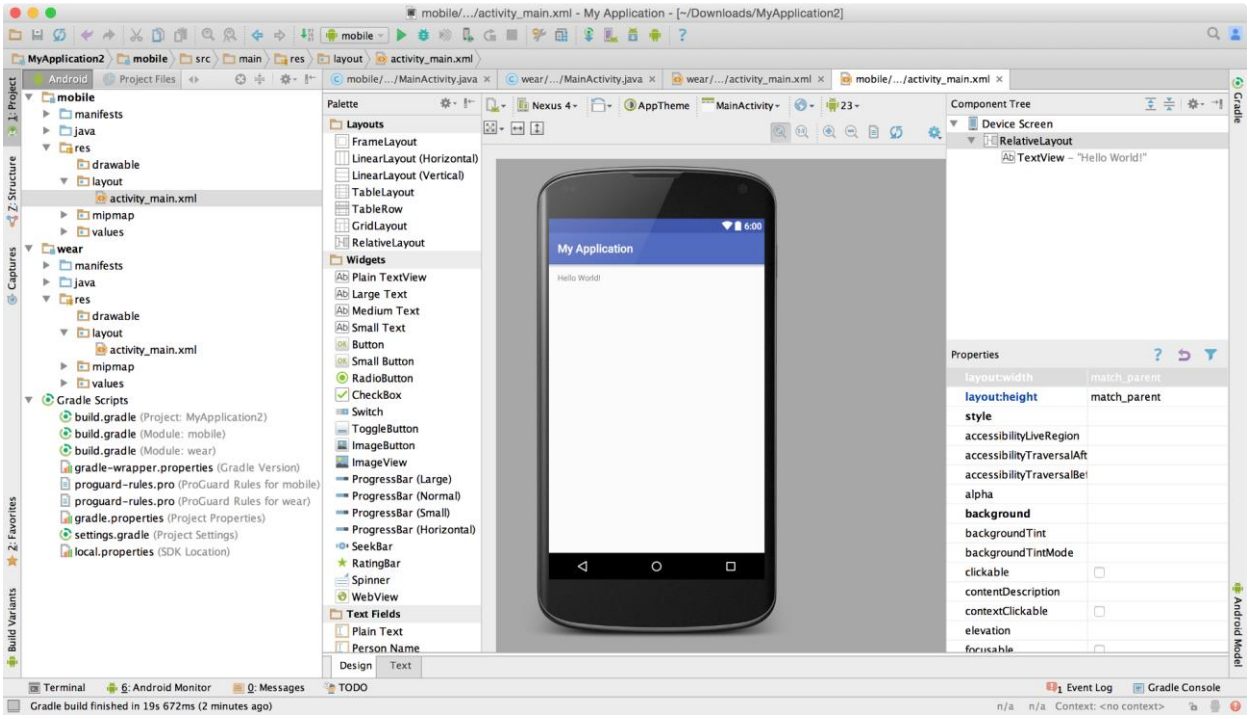

Figura 2-9. Interfaz de Android Studio.

Cada proyecto en Android Studio contiene al menos un módulo con archivos de código fuente y recursos. Hay distintos tipos de módulos, entre los que se incluyen:

- Módulos de aplicaciones para Android.
- Módulos de librerías.
- Módulos de Google App Engine.

De manera predeterminada se muestran los archivos del proyecto en la vista Android, como se muestra en la Figura 2-10. Esta vista se organiza en módulos facilitando el acceso a los distintos archivos.

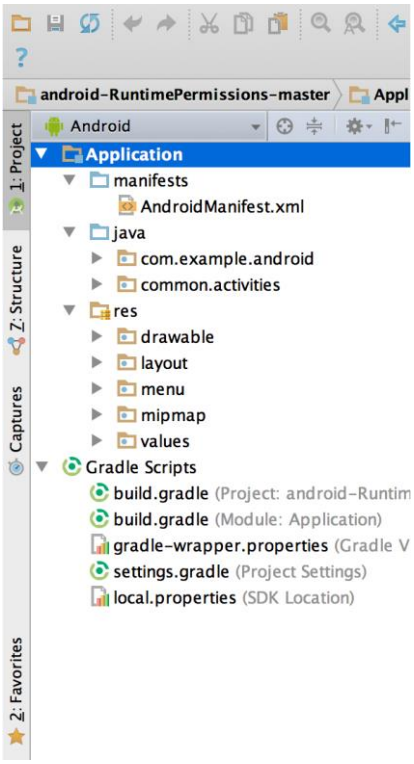

Figura 2-10. Archivos del proyecto en la vista Android. [13]

Cada módulo de la aplicación contiene las siguientes carpetas:

- *Manifests:* Contiene el archivo *AndroidManifest.xml*, donde se pueden aplicar las configuraciones básicas de la aplicación tales como: los permisos que se le concede a la aplicación (acceso a internet, escribir en memoria externa…), el nombre de las distintas *activities* (actividades, cada una de las distintas pantallas con las que interactúa el usuario) o cuál de ellas es la que se inicia al abrir la aplicación.
- Java: Contiene archivos de código fuente de Java.
- Res: Contiene recursos, como los diseños XML (*eXtensible Markup Language*, Lenguaje de marcado extensible) de las distintas *activities*, imágenes a las que tener acceso desde la aplicación o distintas cadenas de texto (*strings)*, estilos y colores que se encuentran en la carpeta *values*.

Sin embargo la estructura del proyecto en el disco no es igual a esta vista Android. Para ver la estructura real, es necesario seleccionar la vista *Project.* (Figura 2-11).

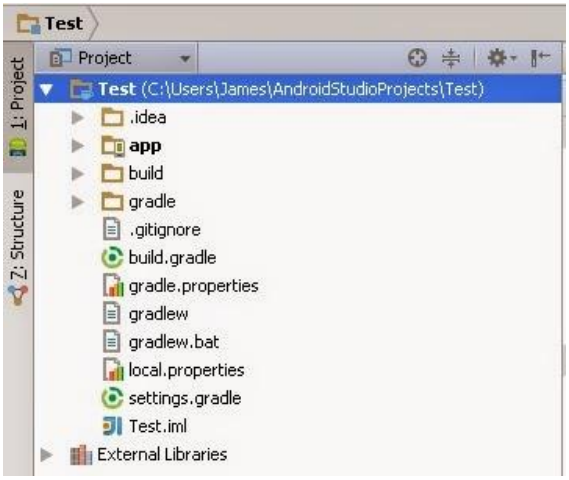

Figura 2-11. Vista Project en Android Studio. [13]

3 DESARROLLO DEL TFG

3.1 Requerimientos del Proyecto

Lo primero a la hora de plantear el desarrollo del Trabajo de Fin de Grado, es recopilar una lista de los requerimientos o necesidades del proyecto. Por lo tanto, lo que se pretende obtener con el desarrollo de una aplicación de gestión de faltas de asistencia es:

- Mantener la misma información contenida en el parte de clase actual.
- Debido al elevado número de planes de estudios que se están llevando a cabo actualmente en la Escuela Naval Militar, es necesario que la aplicación sea capaz de poder cambiar entre ellos rápidamente, obteniendo su información de configuración de un servidor.
- En vista a un posterior análisis de datos y estadísticas, es necesario que la aplicación envíe los datos generados a un servidor, de manera ordenada y con un formato establecido.
- Interfaz gráfica sencilla e intuitiva que permite detectar y corregir errores al instante.
- Capacidad de trabajar sin conexión. Las novedades se almacenan temporalmente en el dispositivo y se envían al servidor cuando vuelva a tener conexión.
- Mantener la figura del Jefe de Grupo, alumno responsable de realizar la novedad de su grupo. Asimismo, también será el responsable de portar el dispositivo en el que vaya instalada la aplicación.

Una vez analizados los requerimientos del proyecto, se opta por desarrollar una aplicación en un dispositivo tableta con sistema operativo Android.

3.2 Diseño de la Interfaz

Un paso muy importante a la hora de desarrollar una aplicación es pensar el diseño de la interfaz. Esta tiene que ser sencilla a la vez que intuitiva, ya que va a ser con lo que el usuario interaccione. El objetivo sería que un usuario que usase la aplicación por primera vez fuera capaz de saber usarla. Por ello se procede a incluir numerosos botones que explican su función, así como texto aclarativo en las distintas *activities.*

Tras varios bocetos iniciales y cambios, el diseño final es el que puede verse en la Figura 3-1. Puede observarse que la aplicación dispone de nueve *activities*, relacionadas entre sí y a excepción de la primera, *MainActivity*, todas disponen de la opción de volver hacia atrás en el caso de que el usuario cometiera un error a la hora de rellenar los distintos campos. También se puede ver el flujo que siguen los datos dentro de la aplicación, así como en flechas de color blanco el proceso “normal” de trabajo, mientras que las flechas negras indican donde se puede volver atrás. Cabe destacar que las flechas que indican comunicación con el servidor son distintas puesto que no requieren de interacción con el usuario.

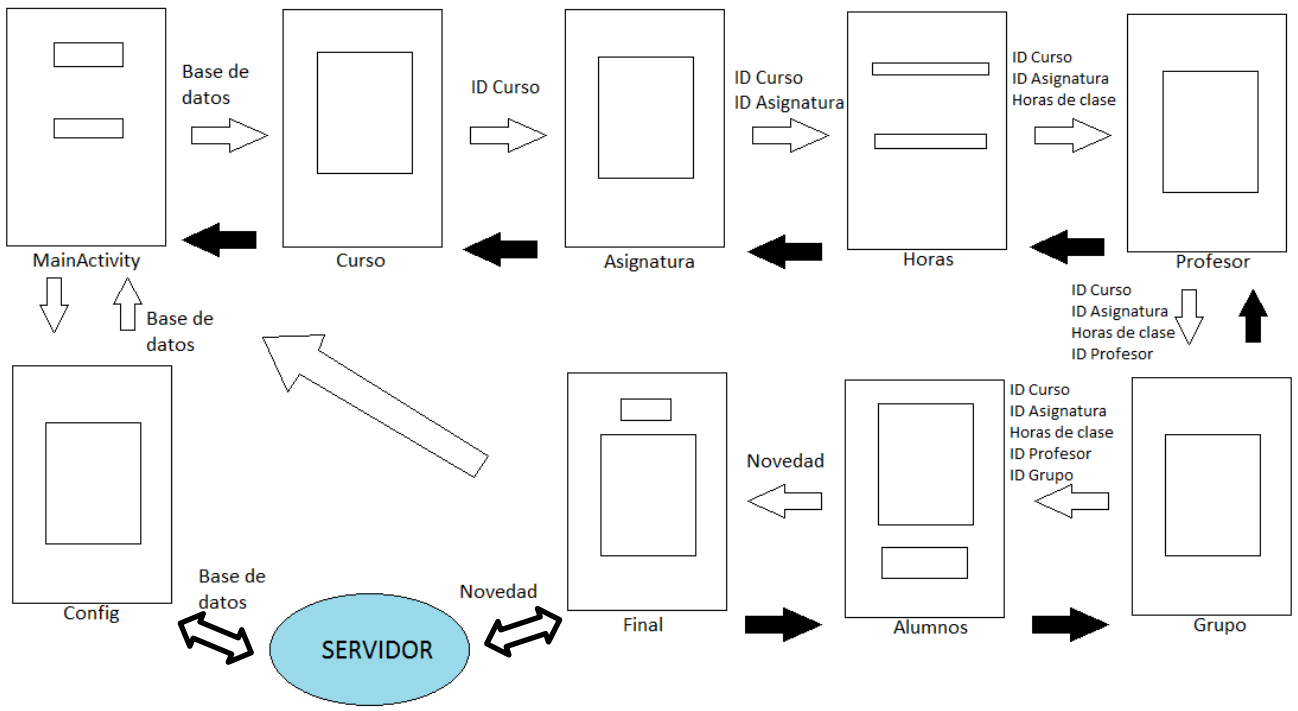

Figura 3-1. Esquema del diseño de la interfaz.

Antes de empezar a explicar las funciones de cada *activity*, cabe destacar que todo el proceso de la aplicación parte de la descarga de alguna de las bases de datos alojadas en el servidor, que servirán para configurar la aplicación con los datos (asignaturas, profesores, alumnos, etc) específicos de cada titulación. Habrá una base de datos por titulación, todas ellas siguiendo el mismo formato. Más adelante se profundizará acerca de ellas. En el anexo se incluye el código de las que se han usado en este proyecto.

Sintetizando lo que puede verse en la Figura 3-1, el *workflow* (flujo de trabajo) de la aplicación sería el siguiente:

- En la primera *activity*, MainActivity, el usuario tendrá la opción de elegir entre iniciar una novedad, acceder a la configuración o mandar las novedades que hayan podido quedar previamente pendientes por falta de conexión.

- En la ventana de configuración, Config, el usuario podrá elegir entre las distintas bases de datos existentes, una por titulación o plan de estudios, y se conectará al servidor para descargar dicha base de datos.

- Al iniciar la novedad, ventana Curso, la aplicación crea todos los recursos obtenidos de la base de datos descargada y muestra una lista de los cursos que el usuario puede utilizar para seleccionar el que proceda.

- En la siguiente *activity*¸Asignatura, y a partir del identificador del curso seleccionado se muestra una lista de las asignaturas correspondientes a dicho a curso.

- A continuación, ventana Horas, se elige la hora de clase, que por defecto vendrá ya seleccionada según la hora real que sea aunque permite que el usuario la modifique.

También se puede especificar en esta ventana el número de períodos lectivos que comprenda la clase. Estos períodos podrán ser 1, 2 o 3, correspondiéndose con clases de teoría o seminarios, prácticas o actividades de instrucción y adiestramiento.

- En la *activity* Profesor, se procederá a elegir el profesor que imparte la clase, para ello existen varias opciones. La principal sería seleccionar uno de los nombres que aparece directamente al iniciar la actividad. Estos nombres pertenecen a los profesores recomendados para la asignatura elegida, esta sugerencia proviene de selecciones previas del usuario. En caso de que ninguno de los profesores sugeridos fuera válido se dispone adicionalmente de otros dos métodos para seleccionar al profesor. Uno de ellos sería introducir parte del nombre de este y se mostraría una lista de los profesores que contengan esa parte. La otra opción sería mostrar en pantalla los nombres de todos los profesores registrados en la base de datos.

- Posteriormente, ventana Grupo, aparecerá en pantalla los grupos de clase en los que se divide el curso seleccionado.

- La ventana con la que más tendría que interactuar el usuario, sería la *activity* Alumnos. Se mostrará un listado de los alumnos que componen el grupo seleccionado, así como un desplegable al lado de cada nombre de los integrantes de dicho grupo. En el desplegable se podrá seleccionar la situación en la que se encuentra cada alumno, viniendo seleccionada por defecto la situación “Sin Novedad”. Además también habrá otro desplegable en el que el usuario tendrá que seleccionar el Jefe de Grupo. Por último en la parte inferior de la pantalla habrá un campo en el que se podrá dibujar para que el profesor firme la novedad.

- Ya en la última *activity*, ventana Final, se visualizaría un resumen de todos los datos que se han seleccionado a lo largo de todo el proceso de creación de la novedad, dando la opción de volver atrás en caso de que hubiera que modificar algún parámetro. El último paso sería enviar la novedad al servidor.

A continuación se procede a explicar más en profundidad las bases de datos usadas para este proyecto así como cada una de las *activities* de las que se compone la interfaz de la aplicación.

3.3 Bases de datos

Para poder trabajar la aplicación necesita ser adecuadamente configurada con los datos de los alumnos, profesores, asignaturas, cursos, grupos y situaciones. Estos datos se obtienen del servidor, donde se representan mediante un archivo XML. Dado que hay varias titulaciones disponibles en la Escuela Naval Militar, habrá varios XML en el servidor (Ver Anexo I: Bases de Datos usadas).

Para un correcto funcionamiento del programa es necesario que las bases de datos se realicen siguiendo el siguiente formato:

- El elemento raíz ha de contener los atributos “versión” y “titulación”, en los que se indica el año académico y el plan de estudios respectivamente.

  Por ejemplo: <novedades version="20162017" titulacion="CG_CT">

- El resto de elementos seguirán la misma estructura:

```
<elementos>
 <elemento atributo="...">
  <nombre>NombreElemento</nombre>
 </elemento>
 .
 .
 .
 </elemento>
</elementos>
```

Figura 3-2. Estructura de los elementos de la base de datos.

- Siguiento el modelo de la Figura 3-2, los “cursos” tendrán que contener un atributo como su identificación.

  Ejemplo elemento cursos:

  ```
  <cursos>
     <curso id="c1">
           <nombre>Primer curso</nombre>
     </curso>
  </cursos>
  ```

- Las “asignaturas” tendrán un atributo de identificación, otro atributo “curso” con el ID del curso al que pertenezcan y un último atributo “militar” cuyo contenido será o “S” o “N” indicando respectivamente si la asignatura es militar o no. El ID de las asignaturas se corresponde con el código que recibe dicha asignatura en el documento del plan de estudios. Dicho documento ha sido facilitado por la Jefatura de Estudios de la Escuela Naval Militar.

  Ejemplo elemento asignaturas:

  ```
  <asignaturas>
     <asignatura id="a6018" curso="c1" militar="N">
           <nombre>Termodinámica y Transmisión de calor</nombre>
     </asignatura>
  </asignaturas>
  ```

- Los “grupos” contendrán su código de identificación así como un atributo “curso” haciendo referencia al curso al que pertenecen.

  Ejemplo elemento grupos:

  ```
  <grupos>
     <grupo id="g11" curso="c1">
           <nombre>Grupo 1-1</nombre>
     </grupo>
  </grupos>
  ```

- Los “profesores” tendrán como único atributo su ID. Este código de identificación se correspondería con el Documento Nacional de Identidad de dicho profesor.

Ejemplo elemento profesores:

```
<profesores>
    <profesor id="p12654963">
            <nombre>Norberto Fernández García</nombre>
    </profesor>
</profesores>
```

- Los “alumnos” necesitarán atributos de identificación así como de referencia al grupo en el que se encuentren. Al igual que los “profesores”, los códigos de identificación se corresponderán con el DNI de los alumnos.

  Ejemplo elemento alumnos:

```
<alumnos>
    <alumno id="a52111655" grupo="g11">
            <nombre>Rafael García Ruiz</nombre>
    </alumno>
</alumnos>
```

- Por último, los “situaciones” sólo contendrán el atributo ID.

  Ejemplo elemento situaciones:

```
<situaciones>
    <situacion id="s1">
            <nombre>Sin Novedad</nombre>
    </situacion>
</situaciones>
```

- Para permitir una posterior validación del documento será necesario definir el contenido de los elementos descritos anteriormente, para lo cual se utiliza DTD (*Document Type Definition*, Definición de tipo de documento). Este DTD va incluido al principio del fichero pero también podría ir incluido en otro documento aparte al que habría que hacer referencia en el XML a validar [14]. Los documentos XML no solo han de ser validados a través del DTD (en este caso) sino que también tienen que estar bien formados. Esto lo comprueban las propias reglas del lenguaje XML, por ejemplo si alguna de las etiquetas no está correctamente cerrada provocaría un error [15]. A continuación puede verse el DTD utilizado en las bases de datos de este proyecto:

```
<!DOCTYPE novedades [
   <!ELEMENT novedades (cursos, asignaturas, grupos, profesores, alumnos, situaciones)>
            <!ATTLIST novedades version CDATA #REQUIRED>
            <!ATTLIST novedades titulacion CDATA #REQUIRED>
   <!ELEMENT cursos (curso)+>
   <!ELEMENT curso (nombre)>
            <!ATTLIST curso id ID #REQUIRED>
   <!ELEMENT nombre (#PCDATA)>
   <!ELEMENT asignaturas (asignatura)+>
   <!ELEMENT asignatura (nombre)>
            <!ATTLIST asignatura id ID #REQUIRED>
            <!ATTLIST asignatura curso IDREF #REQUIRED>
            <!ATTLIST asignatura militar CDATA #REQUIRED>
   <!ELEMENT grupos (grupo)+>
   <!ELEMENT grupo (nombre)>
            <!ATTLIST grupo id ID #REQUIRED>
            <!ATTLIST grupo curso IDREF #REQUIRED>
   <!ELEMENT profesores (profesor)+>
   <!ELEMENT profesor (nombre)>
            <!ATTLIST profesor id ID #REQUIRED>
   <!ELEMENT alumnos (alumno)+>
   <!ELEMENT alumno (nombre)>
            <!ATTLIST alumno id ID #REQUIRED>
            <!ATTLIST alumno grupo IDREF #REQUIRED>
   <!ELEMENT situaciones (situacion)+>
   <!ELEMENT situacion (nombre)>
            <!ATTLIST situacion id CDATA #REQUIRED>
]>
```

3.4 Activities

Antes de comenzar con la explicación de las actividades, es necesario para una mejor comprensión de dichas explicaciones que se describan los distintos elementos que ofrece Android relativos al diseño de las *activities*.

De entre los múltiples elementos disponibles se procede a explicar los que se han usado para el desarrollo de este proyecto [16]:

- Button: Consiste en un texto o un icono (o ambos) que comunica qué acción ocurre cuando el usuario lo pulsa.
- TextView: Muestra una cadena de texto que puede ser modificada desde el código, pero no por el usuario.
- EditText: Muestra un campo en el que el usuario puede introducir una cadena de texto.
- ListView: Es un grupo de vistas que muestra una lista de elementos desplazables y seleccionables.
- Spinner: Ofrece una manera de seleccionar un valor de un conjunto de ellos. Por defecto vendrá seleccionado el primero de todos esos valores. Al tocar el Spinner se mostrará una lista desplegable que mostrará el resto de valores disponibles, de los que el usuario podrá seleccionar uno.

- Toast: Proporciona información al usuario sobre una operación en forma de texto emergente. Ocupa tan solo el espacio necesario para mostrar el texto requerido y no interrumpe la actividad en la que se encuentre.

También es necesario explicar que cada una de las *activities* que a continuación se describen, tienen asociadas a ellas una clase Java en la que se introduce el código de la función que se desea que realice dicha actividad y un fichero XML con el diseño de la ventana. En la Figura 3-3 se puede ver un ejemplo de un diseño sencillo que contiene un TextView y un Spinner.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <Spinner
        android:layout_width="0sp"
        android:layout_height="wrap_content"
        android:id="@+id/spinner"
        android:layout_weight="2" />

    <TextView
        android:text="TextView"
        android:layout_width="0sp"
        android:layout_height="wrap_content"
        android:id="@+id/textView"
        android:layout_weight="3"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textSize="24sp"
        android:background="@color/common_action_bar_splitter" />
</LinearLayout>
```

Figura 3-3. Ejemplo del fichero XML de una actividad.

3.4.1 MainActivity

Esta es la primera ventana que visualiza el usuario al abrir la aplicación. En ella se muestran tres Buttons: uno con el texto “Iniciar Novedad”, que una vez pulsado iniciaría la *activity* Curso siempre y cuando exista una base de datos almacenada en los archivos de la aplicación, otro con el texto “Configuración” que abriría la *activity* Config y un último botón cuya función es mandar al servidor las novedades que no hayan podido ser enviadas en el momento de su creación por falta de conexión, el proceso de envío al servidor se explicará más adelante en la sección 3.4.9. Debajo de este Button se informa al usuario de cuántas novedades quedan pendientes de enviar en un TextView. Además también se mostraría otro TextView con un texto con el que se pretende aclarar que la primera vez que se inicie la aplicación es necesario configurarla. En la Figura 3-4 puede verse el *layout* (diseño) de esta ventana.

Además de cuando se abre la aplicación, esta actividad también se inicia en los siguientes casos:

- Al volver atrás en la actividad Curso.

- Al volver atrás en la actividad Config. Este paso no es considerado como de corrección ya que forma parte del *workflow* de la aplicación, al menos la primera vez que se use o cuando sea necesario cambiar la base de datos por la de otra titulación.

- Al finalizar todo el proceso de creación de novedad se accede otra vez a esta ventana.

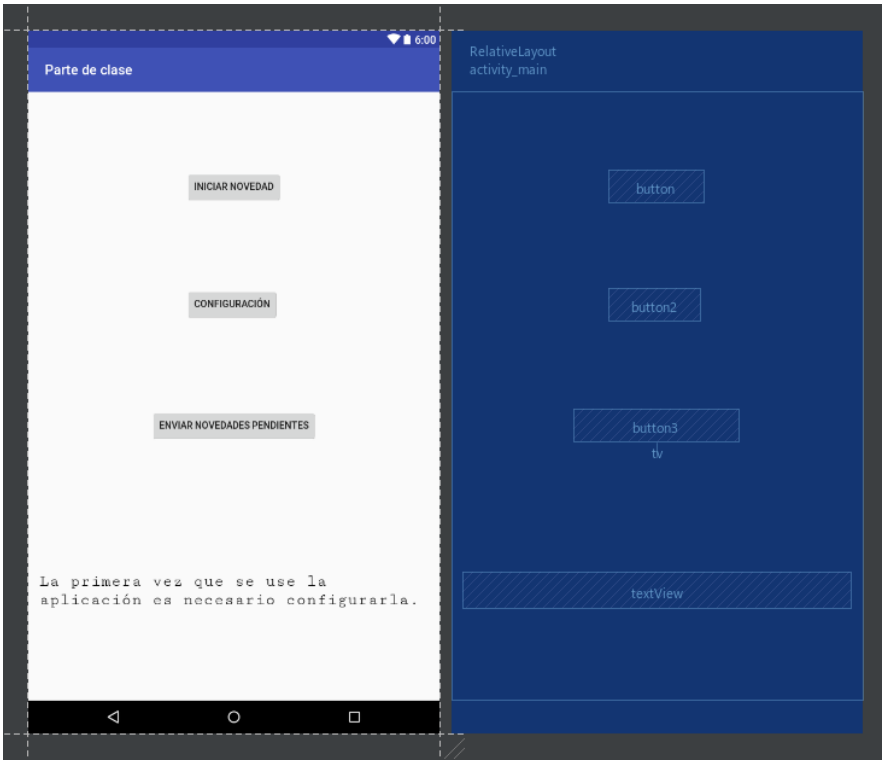

Figura 3-4. *Layout* de la actividad MainActivity.

El texto que muestra la cantidad de novedades pendientes de enviar se actualiza con una función que se ejecuta cada 1000 milisegundos. El motivo de esto es evitar que si se inicia la actividad (al final del proceso) antes de que una novedad termine su proceso de envío muestre esta novedad como si estuviera pendiente de enviar.

3.4.2 Config

En lo referente al diseño, la *activity* Config se compone de un TextView que muestra "Elija titulación" y una ListView con los nombres de cada una de las titulaciones que se han considerado en el desarrollo de la aplicación. Pulsando uno de estos nombres se iniciaría el proceso de descarga descrito anteriormente. Además, en la parte inferior dispone de un Button para volver a la ventana MainActivity.

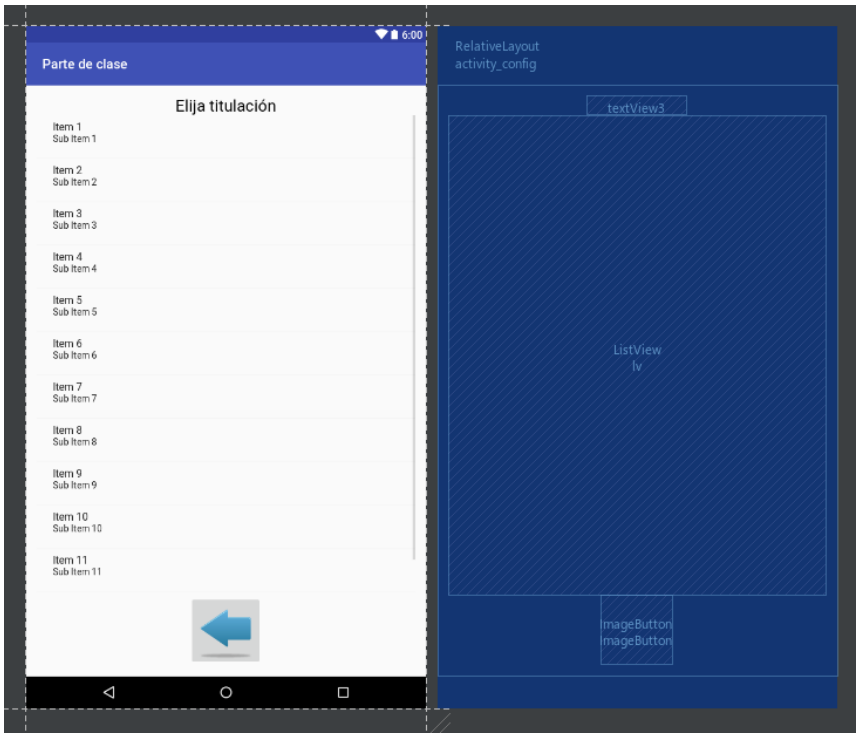

Figura 3-5. *Layout* de la actividad Config.

En esta *activity* es donde se realiza la descarga de la base de datos pertinente en función de la titulación que se desee. Para esto se establece una conexión HTTPS (*HyperText Transfer Protocol Secure*, Protocolo seguro de transferencia de hipertexto) con el servidor y en función de la titulación que seleccione el usuario en la ventana de esta actividad se accede a la URL (*Uniform Resource Locator*, Localizador Uniforme de recursos) correspondiente de cada una de las bases de datos. Una vez descargado el archivo, este se almacena en la memoria interna del dispositivo, dentro de la carpeta correspondiente a la aplicación en un fichero llamado "Base_de_datos.xml".

Se decide usar una conexión HTTPS debido a que en los documentos de las bases de datos aparece información sensible como el nombre de profesores tanto civiles como militares y el nombre de los alumnos de la ENM. De esta manera se consigue que aunque se interceptara el contenido de esta conexión lo único que se obtendría sería un flujo de datos cifrado. Ahora bien, para que un servidor acepte conexiones HTTPS es necesario que se cree un certificado de clave pública para el servidor web. Este certificado debe estar firmado por una autoridad de certificación para que sea aceptado. Esta autoridad confirma que el titular del certificado es quien dice ser. [17]

```
    private static SSLSocketFactory sslSocketFactory;

    private static final String FICHERO_CERT = "ssl-cert-snakeoil.pem";

    public static SSLSocketFactory getSSLSocketFactory(Context context)
            throws CertificateException, IOException, GeneralSecurityException {

        if (sslSocketFactory == null) {
            CertificateFactory cf = CertificateFactory.getInstance("X.509");
            InputStream caInput = new BufferedInputStream(context.getAssets()
                    .open(FICHERO_CERT));
            Certificate ca;
            try {
                ca = cf.generateCertificate(caInput);
            } finally {
                caInput.close();
            }

            String keyStoreType = KeyStore.getDefaultType();
            KeyStore keyStore = KeyStore.getInstance(keyStoreType);
            keyStore.load(null, null);
            keyStore.setCertificateEntry("ca", ca);

            String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
            TrustManagerFactory tmf = TrustManagerFactory
                    .getInstance(tmfAlgorithm);
            tmf.init(keyStore);

            SSLContext sslContext = SSLContext.getInstance("TLS");
            sslContext.init(null, tmf.getTrustManagers(), null);

            sslSocketFactory = sslContext.getSocketFactory();
        }
        return sslSocketFactory;
    }
}
```

Figura 3-6. Clase CustomSSLSocketFactory.

En nuestro caso tenemos un certificado autofirmado, no firmado por una CA (*Certification Authority*, Autoridad de certificación), por eso se necesita configurar la aplicación para que lo acepte. Para hacer esto en la aplicación son necesarios tres elementos. El primero es un certificado de clave pública que se almacena en la carpeta *assets* (bienes, ver Figura 3-7) del proyecto. El segundo elemento necesario es la creación de una clase, que en este caso se llama CustomSSLSocketFactory (ver Figura 3-6), la cual devuelve un SSL *socket* (*Secure Socket Layer*, Capa de *sockets* seguros, *socket*: concepto abstracto que define el intercambio de flujo de información entre dos programas) que acepta el certificado anteriormente nombrado. Por último, se necesita otra clase, DownloadTask, que a partir de la URL seleccionada (eligiendo la titulación) y del SSL *socket* anteriormente creado, establece la conexión HTTPS y descarga la base de datos. Además, una vez se ha descargado la nueva base de datos, se borra la caché que almacena la lista de profesores que han sido seleccionados recientemente por el usuario y que se utilizan para hacer sugerencias en la actividad Profesor.

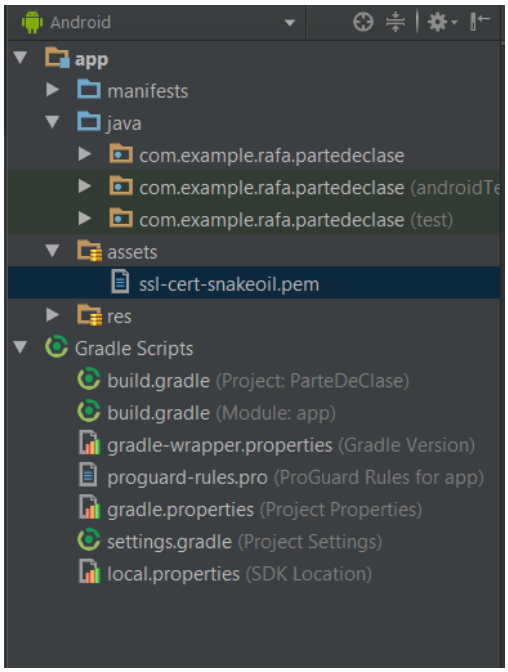

Figura 3-7. Ubicación de la carpeta *assets*.

3.4.3 Curso

Lo primero que se realiza al iniciar esta *activity* es comprobar si los datos de configuración existentes en la base de datos han sido cargados en la aplicación (en memoria).

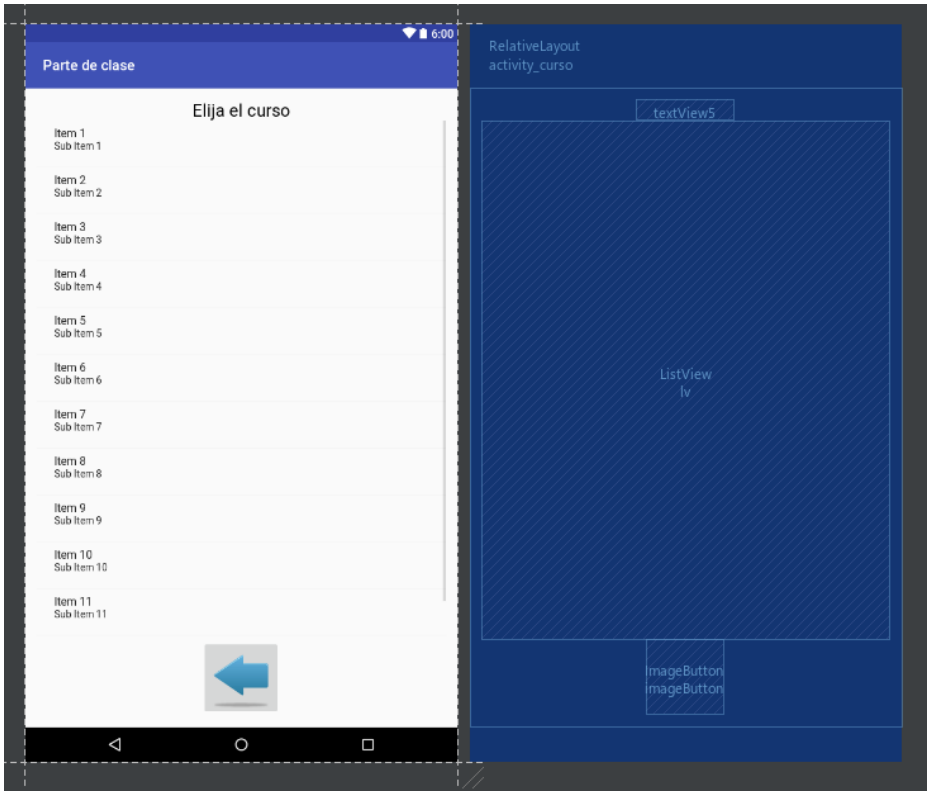

Figura 3-8. *Layout* de la actividad Curso.

En caso de que no existan estos datos, se procede a cargarlos en memoria recorriendo todo el documento almacenado como “Base_de_datos.xml”. Para permitir manejar en la aplicación los datos

de la base de datos se han creado una serie de clases auxiliares dentro de otro paquete denominado "datos" como puede verse en la Figura 3-9. En este paquete se crea una clase por cada uno de los elementos principales de los que se compone la base de datos (profesores, asignaturas, cursos, grupos, alumnos y situaciones). Además se han creado otras dos clases cuyas funciones son la de procesar el documento XML y la de permitir obtener tanto el código de identificación (ID) como el nombre de cualquiera de los elementos de la base de datos.

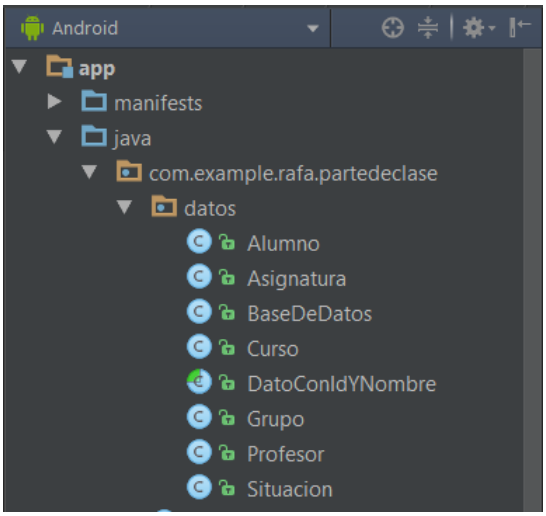

Figura 3-9. Clases del paquete datos.

En caso de que los datos de configuración estén disponibles, simplemente mostraría una lista de los cursos, debajo del texto "Elija el curso" y encima de un botón para volver atrás (ver Figura 3-8). Cuando el usuario pulsa cualquiera de los cursos que se muestran en la ventana se inicia la siguiente actividad, Asignatura.

3.4.4 Asignatura

En esta *activity* se muestra un TextView con el nombre del curso seleccionado en la ventana anterior, así como otro que muestra "Elija asignatura". Debajo de estos dos TextViews aparece una ListView con las asignaturas del curso previamente elegido. Asimismo, un Button en la parte inferior permite volver atrás, posibilitando corregir un posible error en la elección del curso. Pulsando el nombre de la asignatura que se desea, se da paso a la elección de la hora.

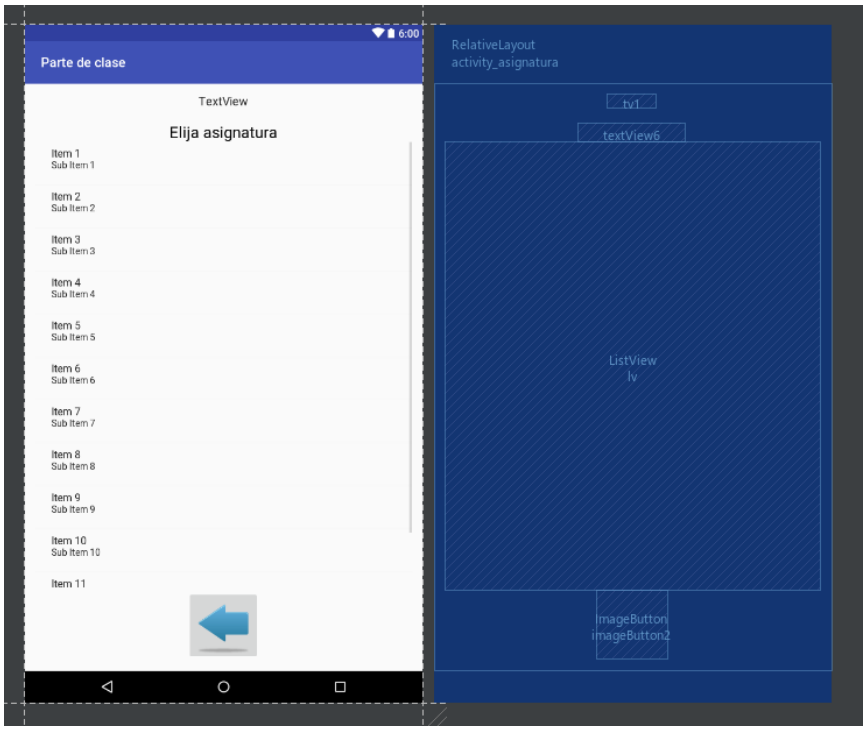

Figura 3-10. *Layout* de la actividad Asignatura.

3.4.5 Horas

Esta ventana, a diferencia de las *activities* previas no muestra una ListView sino que muestra dos Spinners para la elección de la hora de clase y del número de horas lectivas (duración) de la clase. Además, en la parte superior se muestra el curso y la asignatura que han sido seleccionados previamente y encima de cada uno de los desplegables un TextView que pone “Introduzca hora” e “Introduzca número de horas de clase” respectivamente (Figura 3-11).

En el desplegable de la hora de clase vendrá seleccionado por defecto el período en el que se encuentre en base a la hora del dispositivo. Estas horas van desde 1ª a 8ª hora y a pesar de aparecer una ya elegida permite que el usuario modifique la selección. Esto ayuda a que el proceso de crear la novedad sea más rápido y a la vez permite que si se realiza en otro momento se pueda elegir cualquier otra hora.

Por último, un botón con el texto “Continuar” permite proseguir con el *workflow* de la aplicación. Sin embargo, no permitirá avanzar si no se ha seleccionado un valor válido en ambos Spinners.

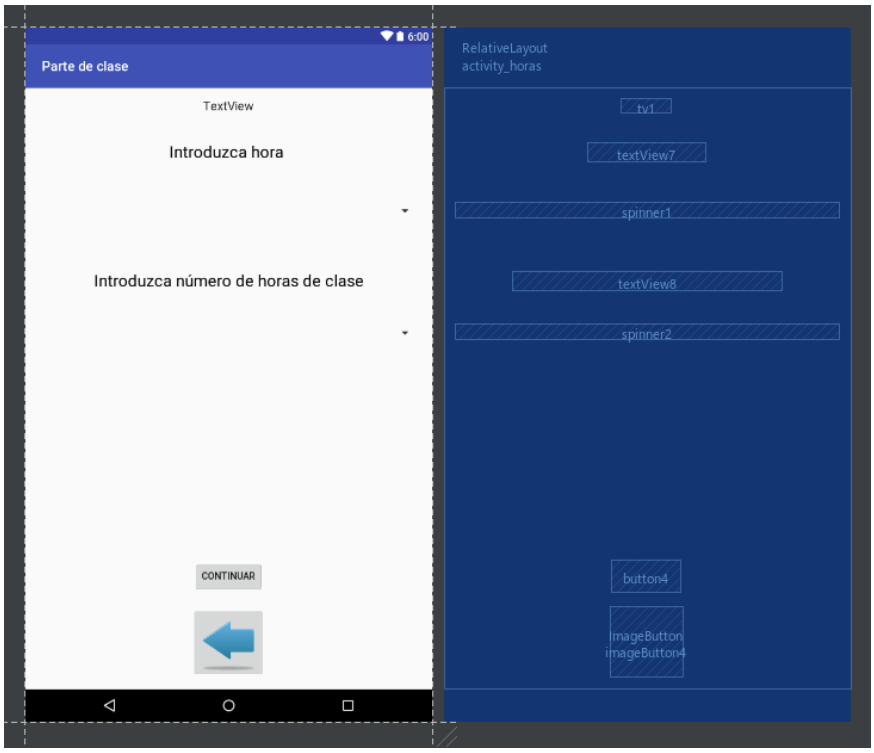

Figura 3-11. *Layout* de la actividad Horas.

3.4.6 Profesor

En cuanto al diseño, esta *activity* muestra en la parte superior el curso, asignatura y horas seleccionadas previamente. Debajo de esto se encuentra un EditText que permite escribir al usuario, y dos Buttons con los textos "Buscar" y "Mostrar todos" respectivamente (Figura 3-12). En la parte inferior otro Button permite volver a la actividad anterior. El resto de la pantalla muestra una ListView con los nombres de los profesores, esta ListView será la misma para mostrar los nombres de los profesores independientemente del método que se utilice, dichos métodos serán descritos más adelante.

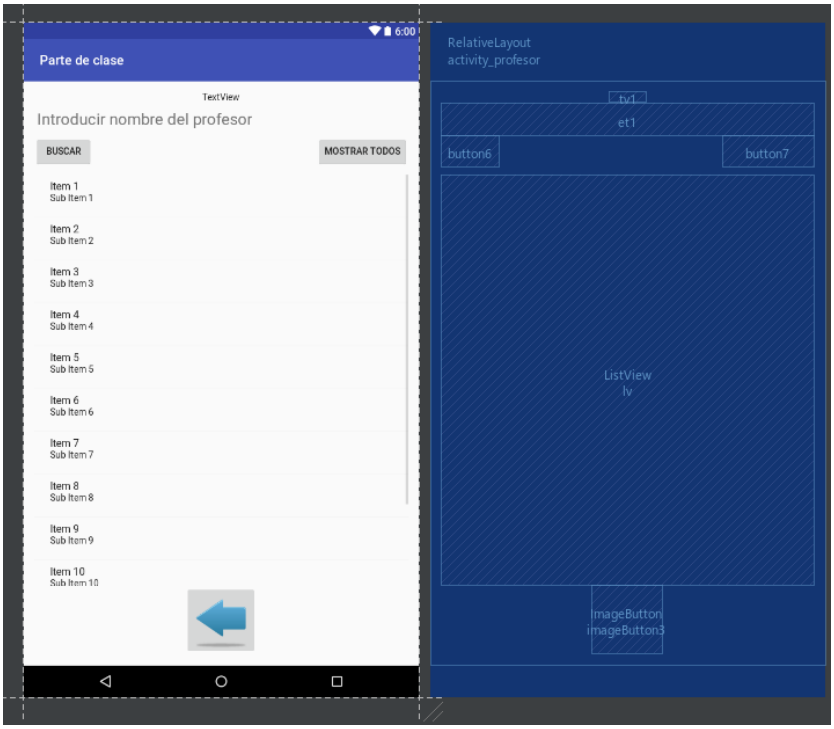

Figura 3-12. *Layout* de la actividad Profesor.

En esta actividad será donde el usuario podrá elegir el nombre del profesor que imparte la clase. Se ha decidido no relacionar en la base de datos los profesores con las asignaturas ya que esto es algo que cambia todos los años o incluso cambia dentro del mismo curso. También existen asignaturas en las que un cierto profesor tan sólo impartirá uno o dos períodos lectivos, o la misma asignatura en distintos planes de estudios es impartida por distintas personas. Todo esto lleva a la resolución de no incluir esta relación asignatura/profesor en la base de datos.

El usuario dispone de tres métodos para poder elegir el nombre del profesor:

- Pulsando el Button "Mostrar todos", se presenta una lista con los nombres de todos los profesores registrados en la base de datos. El usuario puede buscar en esta lista y seleccionar el nombre correcto.

- Introduciendo parte del nombre en el EditText y dándole al Button "Buscar", se muestran en pantalla todos los nombres que contienen la cadena de texto introducida por el usuario. No se hacen distinciones entre letras mayúsculas y minúsculas para agilizar el proceso de creación de la novedad.

- Los profesores presentan localidad temporal, es decir, cuando dan clase una vez en la asignatura es muy probable que vuelvan a hacerlo otra vez, así que lo que se hace es almacenar en una base de datos en el dispositivo los profesores que se han elegido en el pasado para una asignatura y se le sugieren al usuario. La primera vez que se realiza la novedad de una asignatura, no se generará ninguna lista porque todavía no se ha seleccionado ningún profesor para dicha asignatura. La intención es que el usuario use la mayoría de las veces este procedimiento para la elección del nombre.

Para implementar el sistema de recomendación previamente descrito, se hace uso de dos clases auxiliares, CacheProfesores y CacheProfesoresBDHelper. Esta última (ver Figura 3-13) se encarga de crear una base de datos SQL con una tabla cuyas columnas son los nombres de la asignatura y el profesor seleccionados. También se establece que cuando se descargue una nueva versión del archivo "Base_de_datos.xml" se borrarán todas las entradas, es decir se reiniciaría cuando el usuario descargue una nueva base de datos desde la *activity* Config.

```java
public class CacheProfesoresBDHelper extends SQLiteOpenHelper {

    public static class CacheEntry implements BaseColumns {

        public static final String TABLE_NAME = "cache";
        public static final String COLUMN_NAME_ASIG = "asignatura";
        public static final String COLUMN_NAME_PROF = "profesor";
    }

    private static final String SQL_CREATE_ENTRIES =
            "CREATE TABLE " + CacheEntry.TABLE_NAME + " (" +
                    CacheEntry.COLUMN_NAME_ASIG + " TEXT," +
                    CacheEntry.COLUMN_NAME_PROF + " TEXT, " +
                    "PRIMARY KEY (" + CacheEntry.COLUMN_NAME_ASIG +
                    "," + CacheEntry.COLUMN_NAME_PROF + ") )";

    private static final String SQL_DELETE_ENTRIES =
            "DROP TABLE IF EXISTS " + CacheEntry.TABLE_NAME;

    public static final String DATABASE_NAME = "CacheProfesores.db";

    public CacheProfesoresBDHelper(Context context, int version) {
        super(context, DATABASE_NAME, null, version);
    }

    public void onCreate(SQLiteDatabase db) { db.execSQL(SQL_CREATE_ENTRIES); }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(SQL_DELETE_ENTRIES);
        onCreate(db);
    }

    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        onUpgrade(db, oldVersion, newVersion);
    }
}
```

Figura 3-13. Contenido de la clase CacheProfesoresBDHelper.

La otra clase, CacheProfesores, permite añadir nuevos valores (IDs) a las columnas del SQL de asignatura y profesor, y obtener una lista de los nombres de todos los profesores que aparecen junto a una asignatura. En esta actividad el método que nos interesa es el de obtener el listado para mostrarlo por pantalla (ver Figura 3-14). El profesor elegido no se añade al SQL hasta el final del proceso de creación de novedades para evitar que, si el usuario se ha equivocado y vuelve atrás, se añada una fila errónea.

Por lo tanto, se crearía una caché (Tabla 3-1) cuyas columnas serían cadenas de texto con los IDs de asignaturas y profesor. La primera vez que se realizase una novedad, esta tabla se encontraría vacía.

CACHE	
ASIGNATURA	PROFESOR

Tabla 3-1. Caché de profesores.

Cuando se finalizara una novedad, se añadirían los datos a ambas columnas, quedando por ejemplo el resultado mostrado en la Tabla 3-2. Añadir línea en caché de profesores.Tabla 3-2.

CACHE	
ASIGNATURA	PROFESOR
a0001	p1234

Tabla 3-2. Añadir línea en caché de profesores.

Para obtener el listado de profesores sería necesario ejecutar el método *get* de la clase CacheProfesores (ver Figura 3-14) de la siguiente forma:

cache.get(“a0001”);

Dando como resultado el identificador del profesor “p1234” a partir del que se puede obtener su nombre de la base de datos de configuración.

```
public List<String> get(String asignaturaID) {

    db = cacheDBHelper.getReadableDatabase();

    String[] projection = {
            CacheProfesoresBDHelper.CacheEntry.COLUMN_NAME_PROF
    };

    String selection = CacheProfesoresBDHelper.CacheEntry.COLUMN_NAME_ASIG + " = ?";
    String[] selectionArgs = { asignaturaID };

    String sortOrder = CacheProfesoresBDHelper.CacheEntry.COLUMN_NAME_PROF + " ASC";

    Cursor cursor = db.query(
            CacheProfesoresBDHelper.CacheEntry.TABLE_NAME,
            projection,
            selection,
            selectionArgs,
            null,
            null,
            sortOrder
    );


    List<String> profesores = new ArrayList<>();
    while(cursor.moveToNext()) {
        String profID = cursor.getString(
                cursor.getColumnIndexOrThrow(CacheProfesoresBDHelper.CacheEntry.COLUMN_NAME_PROF));
        profesores.add(profID);
    }
    cursor.close();

    return profesores;
}
```

Figura 3-14. Método *get* para obtener el listado de profesores en la clase CacheProfesores.

3.4.7 Grupo

Esta ventana (ver Figura 3-15) tiene el mismo diseño que las actividades Curso y Asignatura. En la parte superior muestra todas las selecciones que ha realizado el usuario hasta el momento: curso, asignatura, horas y profesor. Debajo de lo cual se observa un TextView con el texto “Elija el grupo de clase” y una ListView con los grupos pertenecientes al curso que se eligió previamente. Un Button en la parte inferior permite volver a la actividad previa. El nombre de los grupos tiene el formato: “Grupo (Curso elegido)-(Número de grupo)”, por ejemplo Grupo 2-1 indicaría que es el Grupo 1 del segundo curso.

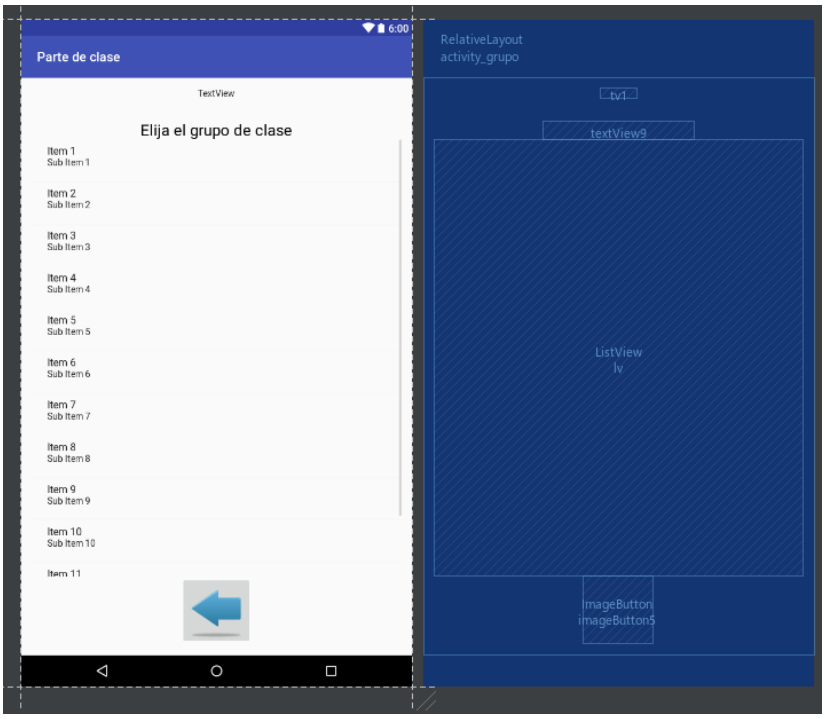

Figura 3-15. *Layout* de la actividad Grupo.

3.4.8 Alumnos

En esta ventana el usuario tendrá que elegir la situación en la que se encuentra cada uno de los alumnos integrantes del grupo previamente seleccionado, así como elegir de un desplegable el nombre del Jefe de Grupo. También será aquí donde el profesor tendrá que firmar.

En lo referente al diseño, en la parte superior se puede ver lo que el usuario ha ido seleccionando a lo largo del proceso, debajo de lo cual se muestra una ListView en la que cada elemento estará a su vez compuesto de dos elementos, un Spinner con las situaciones y un TextView con el nombre de cada alumno. Debajo de esto, otro Spinner permite elegir el Jefe de Grupo y un cuadro que permite dibujar con el dedo para que el profesor firme. En la parte inferior un Button permite regresar a la anterior actividad, otro permite borrar la firma y el último avanza a la siguiente ventana. También se incluye a modo de prueba para el diseño final de todas las actividades de la aplicación el escudo de la Escuela Naval Militar en una escala de grises.

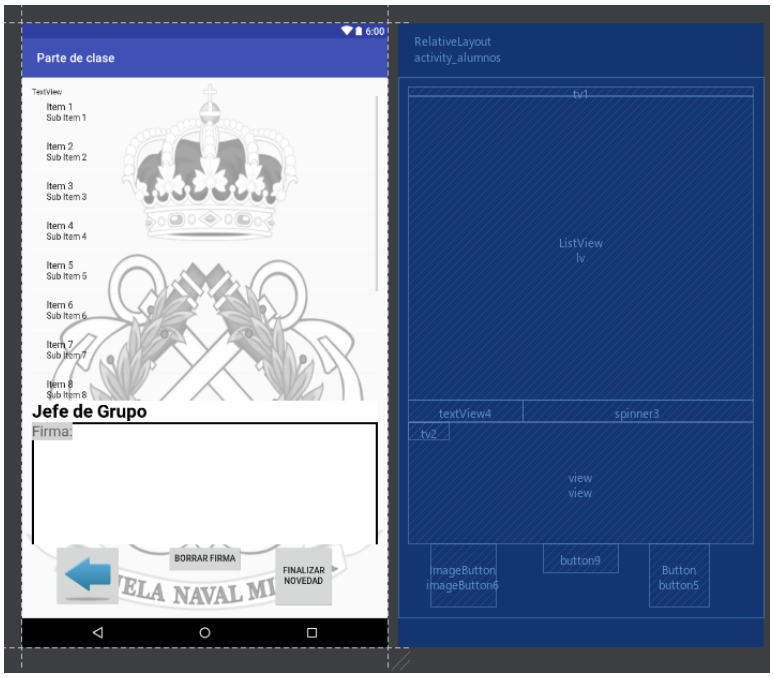

Figura 3-16. *Layout* de la actividad Alumnos.

Como se puede ver en la Figura 3-16 cada uno de los ítem de la lista se compone de un solo elemento pero como se ha dicho anteriormente es necesario mostrar dos componentes, un Spinner y un TextView. Para ello es necesario crear una clase a la que se ha llamado AlumnosArrayAdapter en la cual se personaliza cada uno de estos ítems. De esta manera y usando el diseño de la Figura 3-17 se asigna a cada elemento de la lista un Spinner y un TextView.

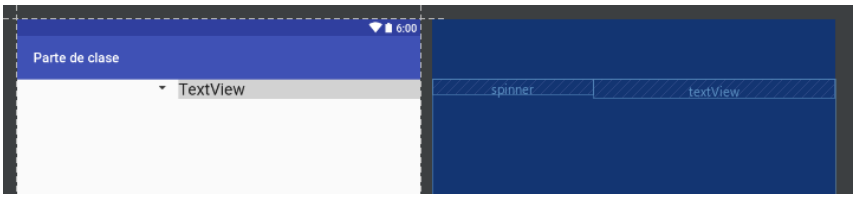

Figura 3-17. Diseño de cada ítem.

El cuadro que permite dibujar incluido en la parte inferior del *layout* será usado para que el profesor firme. Este lienzo se genera con una clase auxiliar llamada Firma, en la cual se utiliza Canvas para hacer el dibujo. Canvas permite generar un espacio en el que la aplicación puede dibujar o también realizar funciones dependiendo de la interacción con el usuario (apoyar el dedo en la pantalla, moverlo, levantarlo) [16]. En esta clase lo primero que se hace es establecer los parámetros con los que se quiere realizar el dibujo, es decir, color, estilo, tamaño del trazo… La aplicación obtiene las coordenadas X e Y en donde toca el usuario dentro del cuadro generado y en función de las distintas interacciones posibles anteriormente nombradas va generando trazos. El elemento de la firma también es necesario almacenarlo y enviarlo al servidor para lo cual se genera un mapa de bits (fichero de datos que representa una sección rectangular de píxeles o puntos de color, denominada matriz [14]) que contiene la firma generada por el profesor.

3.4.9 Final

En esta actividad (ver Figura 3-18) se muestran los datos que se van a enviar al servidor en un formato más legible para el usuario mediante un TextView. Un Button con el texto “Enviar” finalizaría el proceso de creación de la novedad, mandando los datos al servidor. Otro Button permitiría volver atrás en caso de que se hubiese cometido algún error.

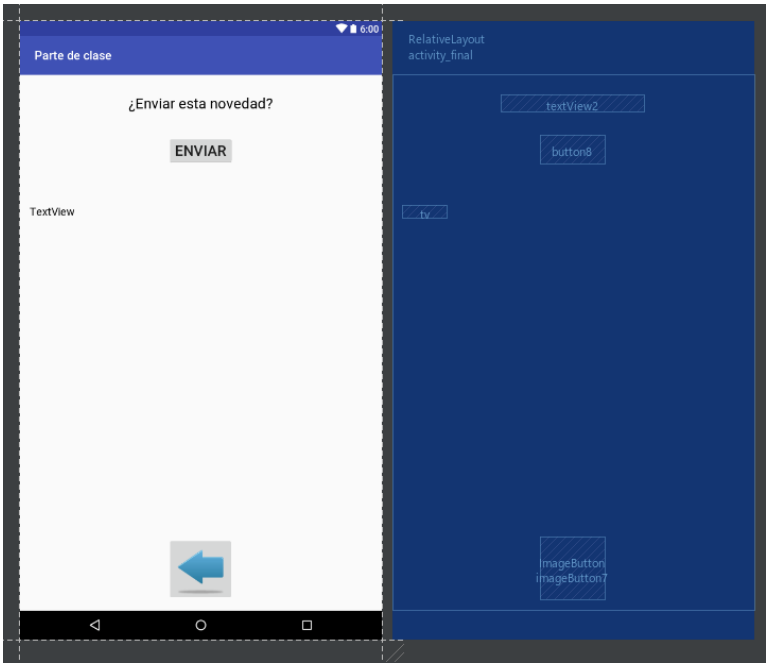

Figura 3-18. *Layout* de la actividad Final.

Cuando el usuario pulsa el Button "Enviar" se genera un documento XML que representa toda la información de la novedad. El nombre del XML será "novedad" seguido de un número autogenerado que varía en función de la hora que sea en ese momento en escala de milisegundos (sello temporal). En este archivo se refleja la siguiente información (ver Figura 3-19):

- Como atributos del elemento raíz se incluyen la versión de la base de datos, la titulación y el momento (fecha y hora) en que se genera la novedad con el formato dd/mm/aaaa hh:mm.
- ID del curso.
- ID de la asignatura indicando mediante un atributo si la asignatura es militar o no.
- Período de clase y número de horas.
- ID del profesor.
- ID del grupo.
- ID de los alumnos indicando también mediante un atributo el ID de la situación en la que se encuentran.
- Es necesario mandar la firma generada por el profesor en la actividad anterior. Sin embargo, debido a que XML es un formato textual y no se puede incluir un elemento binario como el mapa de bits que se genera con la firma, es necesario convertirlo a texto. Por lo que, usando una función Base64, se convierte a cadena de texto y es esta cadena la que se manda al servidor.

```
▼<novedad version="20172018" titulacion="CG_ST" fecha="01/03/2017 12:39">
   <curso>c1</curso>
   <asignatura militar="S">a6013</asignatura>
   <hora>5</hora>
   <horas_clase>1</horas_clase>
   <profesor>p85123964</profesor>
   <grupo>g11</grupo>
 ▼<alumnos>
     <alumno situacion="s1">a52111655</alumno>
     <alumno situacion="s3">a93670655</alumno>
     <alumno situacion="s1">a11726733</alumno>
     <jefe_grupo>a52111655</jefe_grupo>
   </alumnos>
 ▼<firma>
     /9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
     AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCADYAjgDASIA
     AAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3
     p6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEA
     BhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElK
     uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDgv+Df
     zElwkVxILi4EpadirMoK5P7b/wDEIP8A8EW/+hJ/aA/8P3r3/wApaT/g0F/5Qt+DP+zgfjz/AOn7
     /iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCL
     w0/+N0f8RwH7KH/Rin7QH/hx/hp/8boooAP+I4D9lD/oxT9oD/w4/wANP/jdH/EcB+yh/wBGKftA
     OA/ZQ/6MU/aA/wDDj/DT/wCN0f8AEcB+yh/0Yp+0B/4cf4af/G6KKAD/AIjgP2UP+jFP2gP/AA4/
     f+HH+Gn/AMbooooAP+I4D9lD/oxT9oD/AKMU/aA/8OP8NP8A43R/xHAfsof9GKftAf8AhG6KKAD/i
     w0/+N0f8RwH7KH/Rin7QH/hx/hp/8boooA/Aj/gvx/wX3+Dv/BYz4+/D/wBv/BYz4Pfs9/DT4afs9/C
   </firma>
 </novedad>
```


Figura 3-19. Ejemplo de documento XML de novedad.

El envío al servidor, al igual que la descarga de la base de datos, se realiza estableciendo una conexión HTTPS. Además, hay que contemplar la posibilidad de que no haya conexión a Internet en el momento de enviar la novedad. En este caso el archivo XML se almacenaría en la memoria interna del dispositivo, quedando pendiente su envío

Tanto desde el Button que se muestra en la MainActivity “Enviar novedades pendientes” como la próxima vez que se complete un proceso de creación de novedad en el que sí haya conexión, se mandarían todas las novedades que se encuentren almacenadas. Para esto se obtiene el número de archivos de la memoria interna que contengan el nombre “novedad” y se repite la tarea de enviar al servidor tantas veces como archivos haya. Tras cada envío exitoso el archivo mandado se borra, para evitar su reenvío futuro.

4 PRUEBAS

4.1 Pruebas realizadas

Se realizan una serie de pruebas para comprobar el correcto funcionamiento de la aplicación. Para ello, se utilizan unas bases de datos simuladas que pueden verse en Anexo I: Bases de Datos usadas.

Lo primero es comprobar que la aplicación se instala correctamente en el dispositivo tableta. Para la realización de las pruebas se usa una tableta Lenovo TAB3 7 con Android 5.0. Como se puede observar en la Figura 4-1 se ha incluido el escudo de la Escuela Naval Militar como icono.

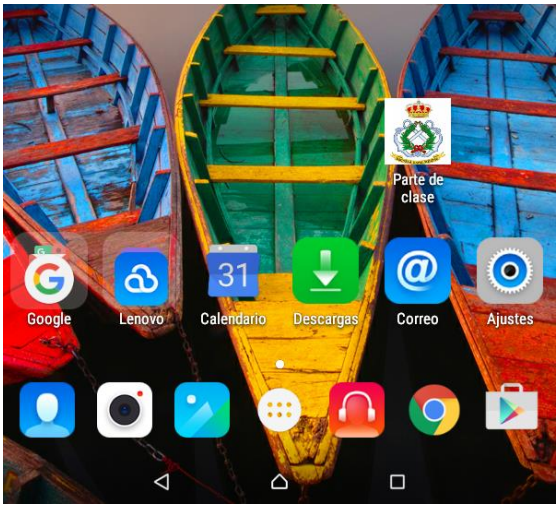

Figura 4-1. Icono de la aplicación.

4.1.1 Flujo de trabajo con conexión a Internet

Se procede a comprobar que la aplicación es capaz de realizar un proceso completo de creación y envío de novedad. Más tarde se comprobarán específicamente los puntos críticos de este proceso.

Comenzamos iniciando el programa. Se comprueba (ver Figura 4-2) que informa correctamente del número de novedades pendientes (en este caso 0) y muestra correctamente los 3 botones y el texto que se pretendía.

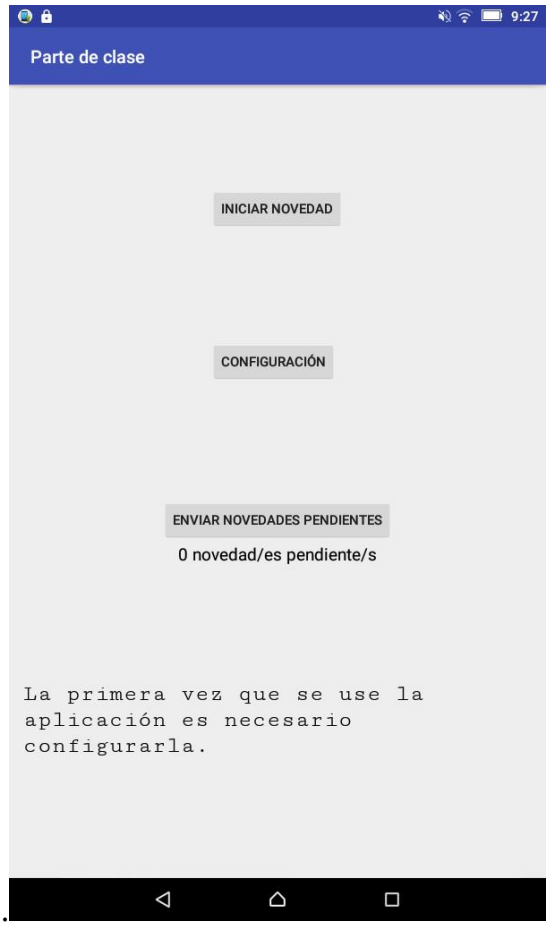

Figura 4-2. MainActivity.

Asimismo, si se pulsa el botón “Enviar novedades pendientes” se muestra un Toast indicando que no existe ninguna novedad almecenada (Figura 4-3).

Figura 4-3. Notificación de novedades pendientes.

Puesto que sería la primera vez que se usa la aplicación, sería necesario configurarla por lo que se pulsa el botón “Configuración”. Se inicia la siguiente actividad, en la que se puede observar las titulaciones disponibles. Para esta primera prueba se procede a descargar la base de datos de Cuerpo General sin titulación (ST). La aplicación informa correctamente de si se descarga el archivo o no (ver Figura 4-4).

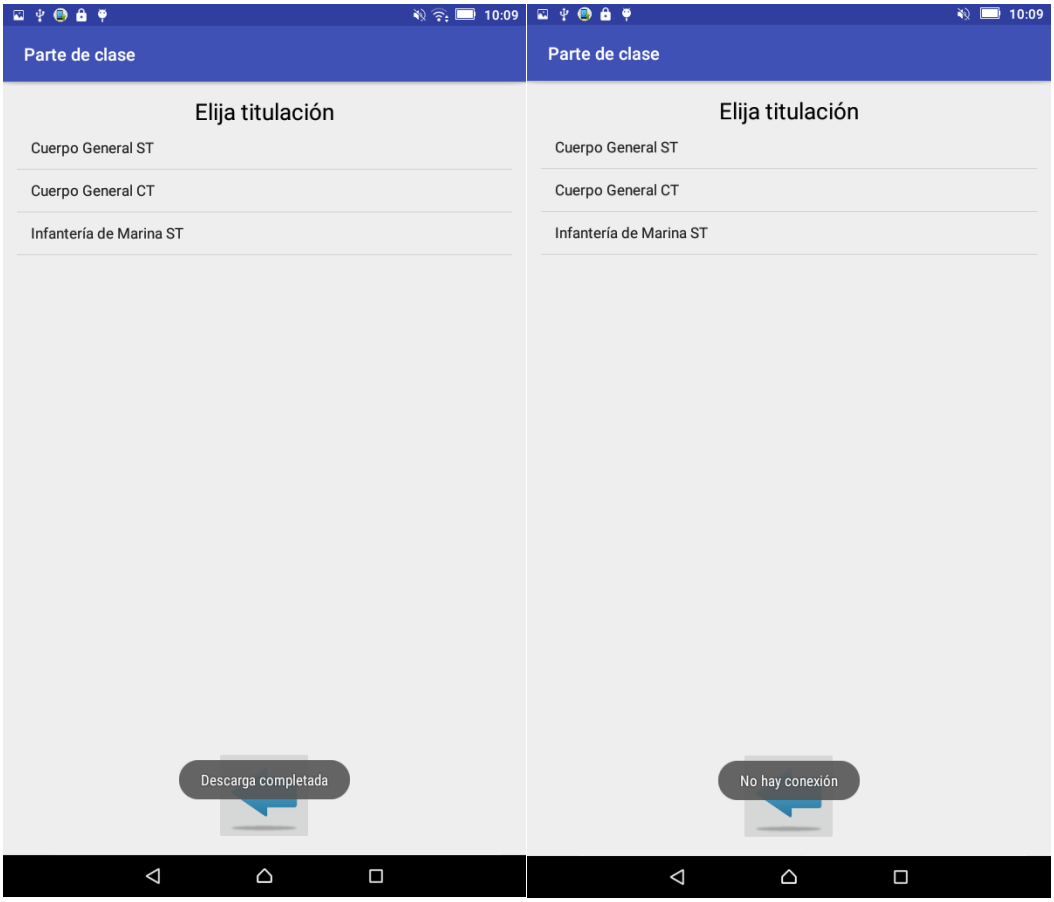

Figura 4-4. Actividad Config (descarga de base de datos).

Una vez descargada la base de datos deseada, se vuelve a la actividad anterior y se pulsa el botón "Iniciar novedad". En caso de que no haya ninguna base de datos almacenada en el dispositivo no permitirá avanzar a la actividad Curso, mostrando un Toast que informa de ello (Figura 4-5).

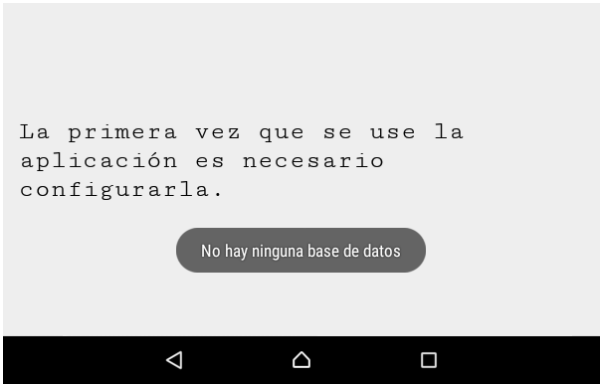

Figura 4-5. Toast que indica que no hay ninguna base de datos.

Si se descargó correctamente la base de datos, al pulsar el botón “Iniciar novedad” se inicia la *activity* Curso (Figura 4-6), mostrando los cursos que se encuentran en la base de datos que se descargó previamente. En este caso, Base de datos de Cuerpo General sin titulación.

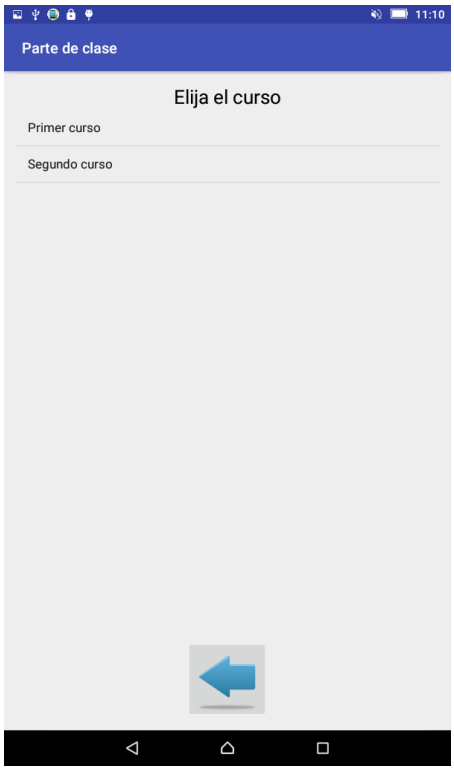

Figura 4-6. Actividad Curso.

A continuación, dependiendo de si el usuario pulsa un curso u otro deberá aparecer un listado de asignaturas distintas. Se comprueba que, efectivamente, las asignaturas no son las mismas seleccionando “Primer Curso” que si se elige “Segundo Curso” (Figura 4-7).

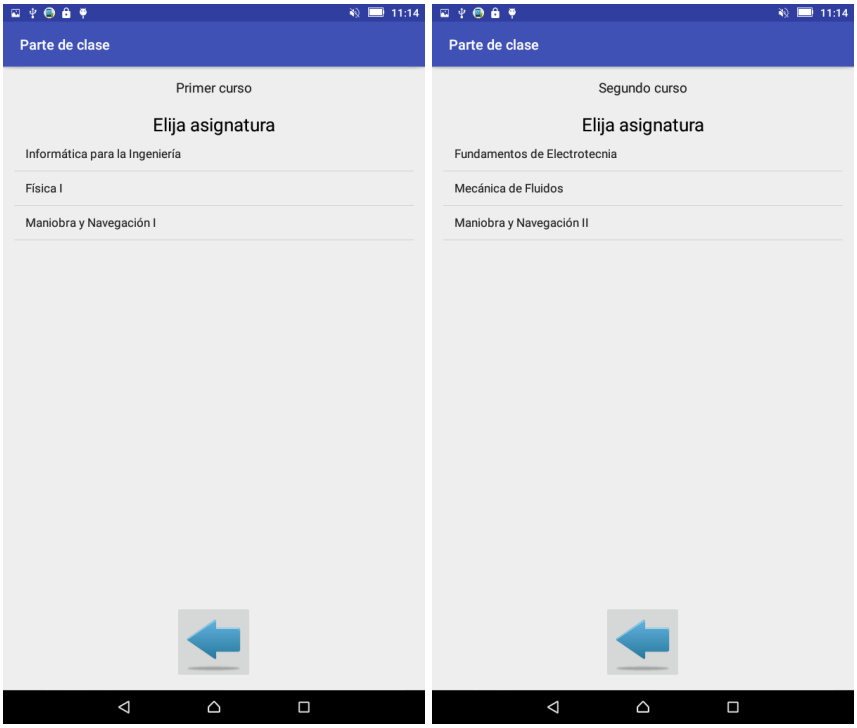

Figura 4-7. Distintas asignaturas en función del curso.

Una vez se elija cualquiera de las asignaturas se inicia la actividad Horas, en la que habrá que elegir las horas de clase y cuántos períodos lectivos comprende. Si se intenta avanzar dejando sin seleccionar cualquiera de los dos Spinner, se muestra un Toast informando de ello y no permite iniciar la siguiente ventana (Figura 4-8).

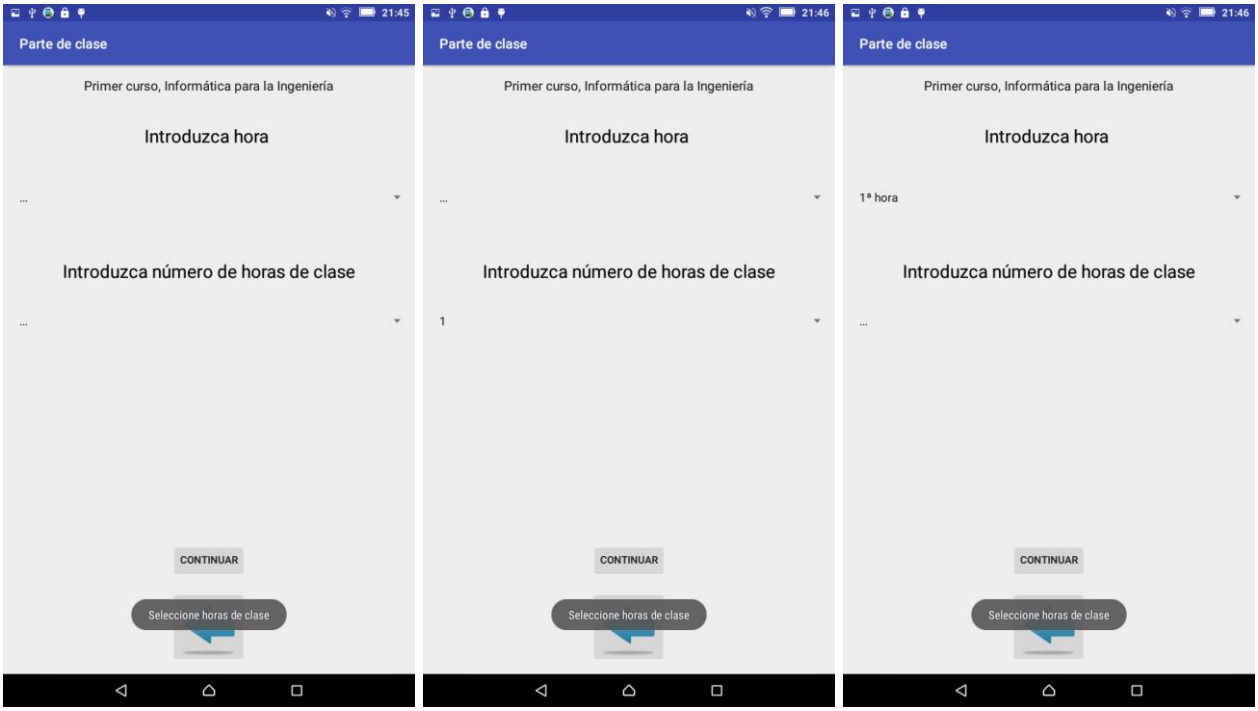

Figura 4-8. Notificación de error en Actividad Horas.

También se observa que el primer Spinner, el de selección de la hora de clase, muestra por defecto correctamente la hora en la que se encuentra en función de la hora del dispositivo. En el caso de la Figura 4-9, al ser las 09:40 el Spinner selecciona "2ª hora".

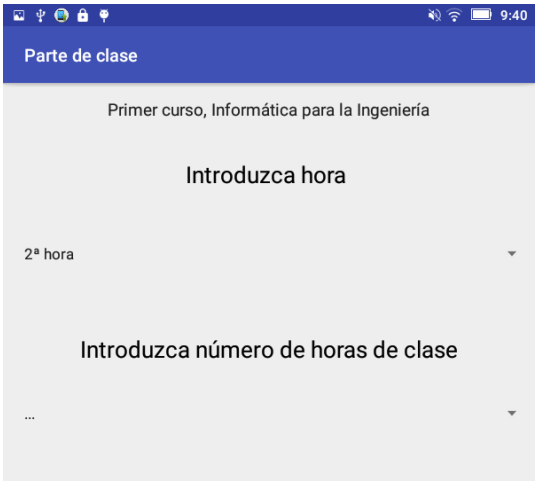

Figura 4-9. Selección automática de la hora.

Una vez seleccionados correctamente los valores de los Spinner se pulsa el Button "Continuar" y comienza la actividad Profesor. Puesto que es la primera vez que se realiza una novedad en el dispositivo, la caché de profesores está vacía y la aplicación no recomienda ningún profesor. Se prueba

que introduciendo en el EditText la cadena de texto “b” muestra en la ListView todos los profesores que contienen dicho carácter y pulsando el Button “Mostrar todos” muestra en la misma ListView el nombre de todos los profesores (Figura 4-10).

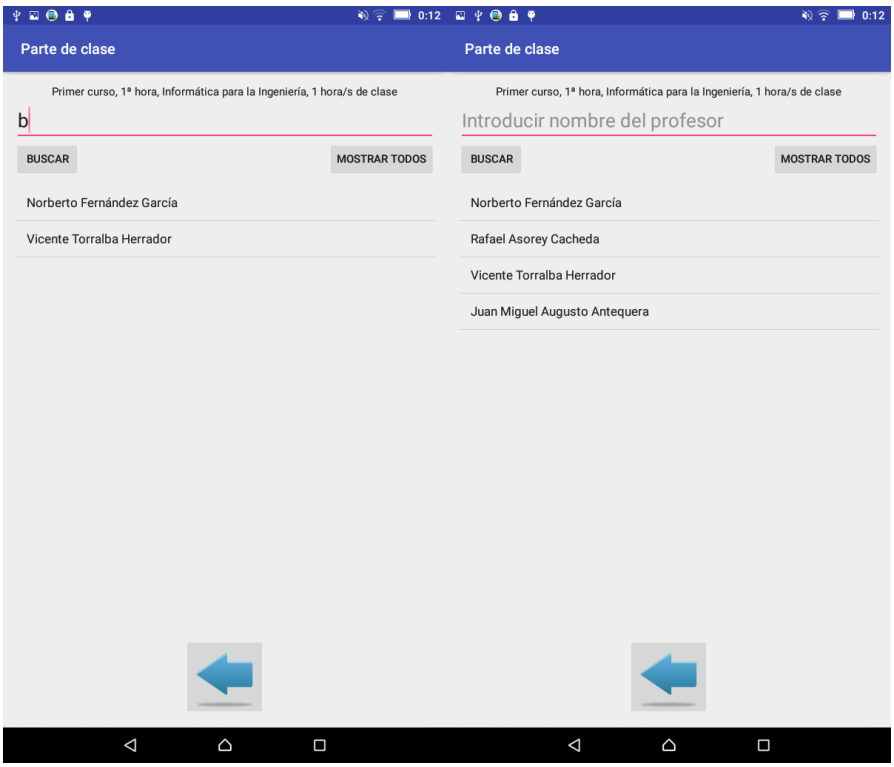

Figura 4-10. Actividad Profesor.

Se selecciona uno de estos profesores y se comprueba que en la *activity* Grupo aparecen los grupos que pertenecen al curso que se seleccionó previamente (Figura 4-11).

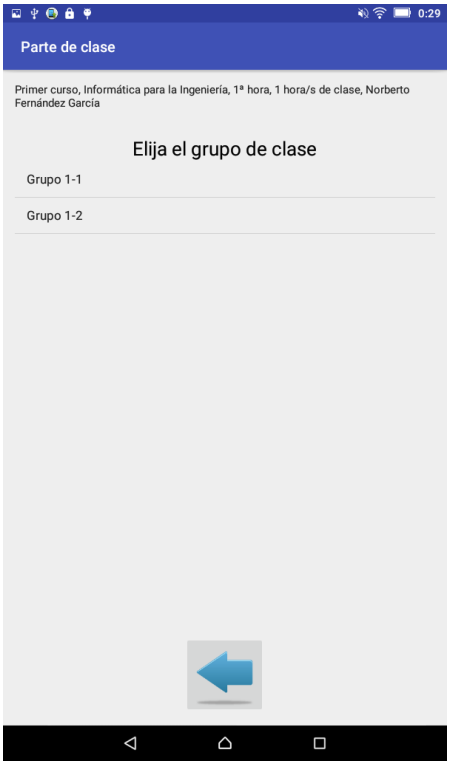

Figura 4-11. Actividad Grupo.

A continuación, tras elegir el grupo pertinente, se inicia la actividad Alumnos y, como se dijo anteriormente, en cada ítem de la ListView se muestra un TextView con el nombre de uno de los alumnos del grupo previamente seleccionado y un Spinner con las posibles situaciones recogidas en la base de datos. Aparece correctamente seleccionada por defecto la situación "Sin novedad". Así mismo, en el otro Spinner, el de selección del Jefe de Grupo, se muestra al primero de los alumnos del grupo, permitiendo elegir a cualquier otro alumno del grupo (Figura 4-12).

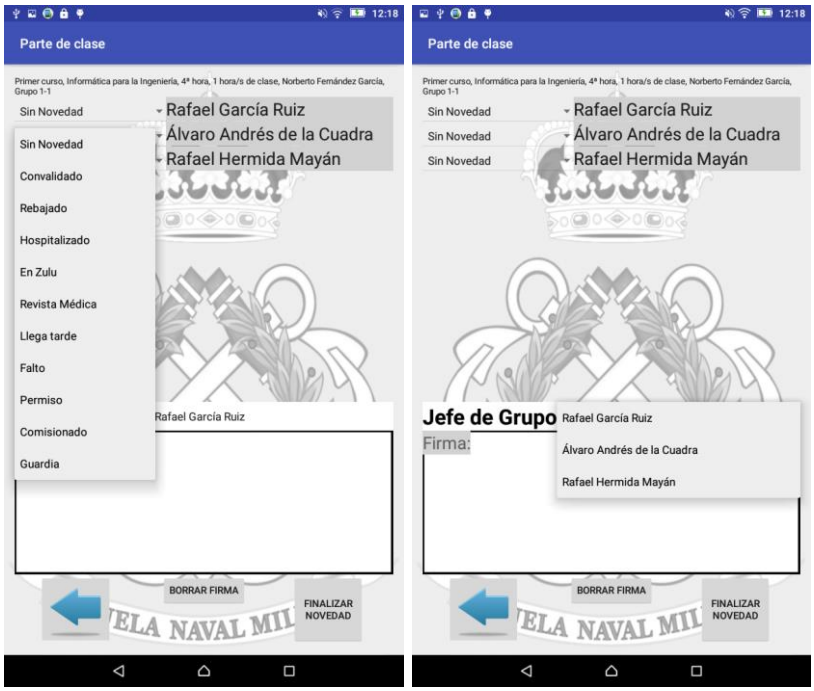

Figura 4-12. Selección de la situación de un alumno y del Jefe de Grupo.

El cuadro destinado a la firma del profesor dibuja correctamente los trazos realizados en la pantalla y el botón "Borrar firma" vuelve a dejar en blanco este lienzo (Figura 4-13).

Figura 4-13. Firma del profesor.

Una vez seleccionados los valores pertinentes de los Spinners y firmado en el espacio destinado a ello, se pulsa el Button "Finalizar novedad", iniciándose la *activity* Final. Se comprueba que muestra todos los datos seleccionados anteriormente así como la fecha y hora en que se ha realizado la novedad (Figura 4-14).

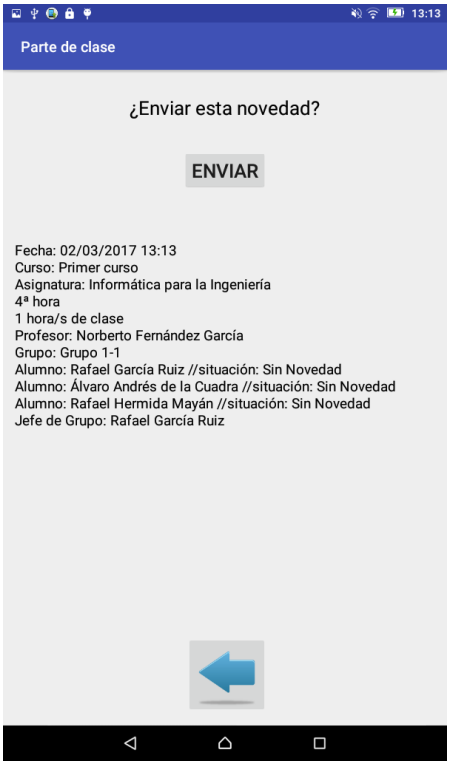

Figura 4-14. Actividad Final.

En esta primera prueba se considera que sí que hay conexión a Internet, por lo que al pulsar el Button "Enviar", se genera el documento XML de la novedad y se manda el servidor. Además, se informa al usuario de que el envío ha sido correcto mediante un Toast (ver Figura 4-15) en la ventana MainActivity ya que esta actividad se inicia inmediatamente tras pulsar el Button.

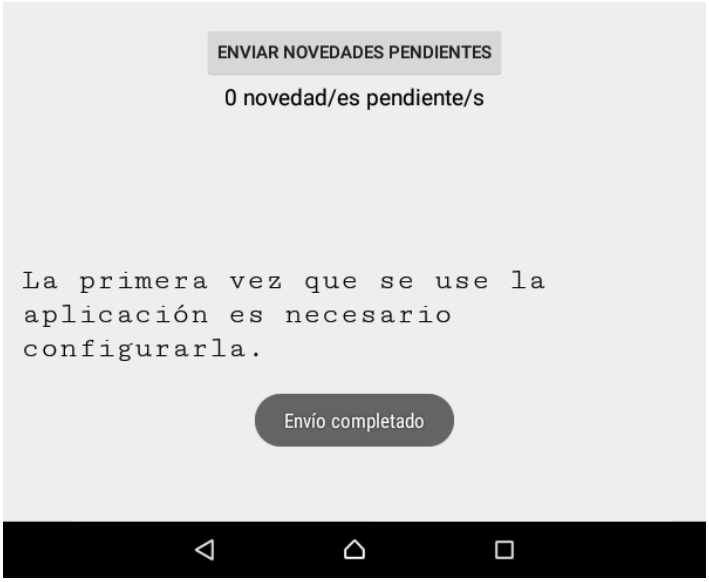

Figura 4-15. Notificación de envío correcto.

A continuación se comprueba que el servidor recibe correctamente el fichero XML de la novedad. Se puede observar en la Figura 4-16 el archivo que se recibe en el servidor, comprobando a partir del atributo “fecha” que es la novedad que se generó en la Figura 4-14. El contenido de la etiqueta “firma” ha sido recortado para evitar tener una figura excesivamente grande. Puede verse el fichero en el Anexo II: Documento XML de novedad.


```
<novedad version="20162017" titulacion="CG_ST" fecha="02/03/2017 13:13">
  <curso>c1</curso>
  <asignatura militar="N">a6003</asignatura>
  <hora>4</hora>
  <horas_clase>1</horas_clase>
  <profesor>p12654963</profesor>
  <grupo>g11</grupo>
▼<alumnos>
    <alumno situacion="s1">a52111655</alumno>
    <alumno situacion="s1">a93670655</alumno>
    <alumno situacion="s1">a11726733</alumno>
    <jefe_grupo>a52111655</jefe_grupo>
  </alumnos>
▼<firma>
    /9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
    AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCADYAjgDASIA
    AAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3
    p6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEA
    BhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElK
    uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDgv+Df
    zElwkVxILi4EpadirMoK5P7b/wDEIP8A8EW/+hJ/aA/8P3r3/wApaT/g0F/5Qt+DP+zgfjz/AOn7
    /iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCL
    /wCUtH/EIP8A8EW/+hJ/aA/8P3r3/wApa/p+ooA/mB/4hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg/
    gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCLf/Qk/tAf+H717/5S0f8AEIP/AMEW/wDoSf2gP/D969/8
```


Figura 4-16. Fichero XML de novedad.

Además, utilizando XSLT (*eXtensible Stylesheet Language Transformation*, Lenguaje que permite transformar un documento XML) en el servidor se transforma el XML de la novedad a formato HTML lo que permite una mejor presentación en un navegador web. Puede observarse que la firma es la que se hizo en la Figura 4-13.

Parte de asistencia

Titulación:	CG_ST
Curso:	Primer curso
Asignatura:	Informática para la Ingeniería
Profesor:	Norberto Fernández García
Grupo:	Grupo 1-1
Fecha:	02/03/2017 13:13
Hora:	4
Duración:	1 hora/s

Alumno	Situación	Jefe
Rafael García Ruiz	Sin Novedad	Sí
Álvaro Andrés de la Cuadra	Sin Novedad	No
Rafael Hermida Mayán	Sin Novedad	No

Figura 4-17. Formato HTML de la novedad.

4.1.2 Flujo de trabajo sin conexión a Internet

Una vez que se ha comprobado que el flujo de trabajo de la aplicación se realiza correctamente cuando hay conexión a Internet, se pasa a comprobar el correcto funcionamiento de la misma cuando no exista esta conexión. Para ello se genera una nueva novedad siguiendo todos los pasos expuestos en 4.1.1 hasta la *activity* Final (Figura 4-18).

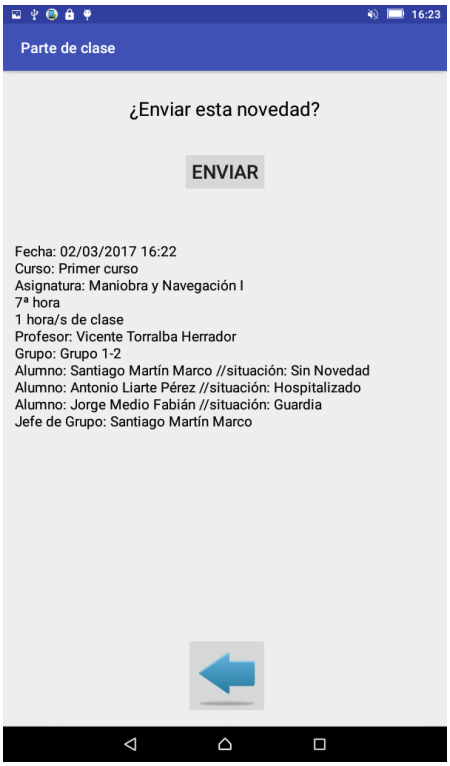

Figura 4-18. Generación de novedad sin conexión.

Se comprueba (ver Figura 4-19) que pulsando el Button “Enviar” la notificación que muestra es distinta así como que el TextView de la actividad MainActivity que muestra el número de novedades pendientes de enviar, se actualiza pasando a indicar que hay una novedad pendiente.

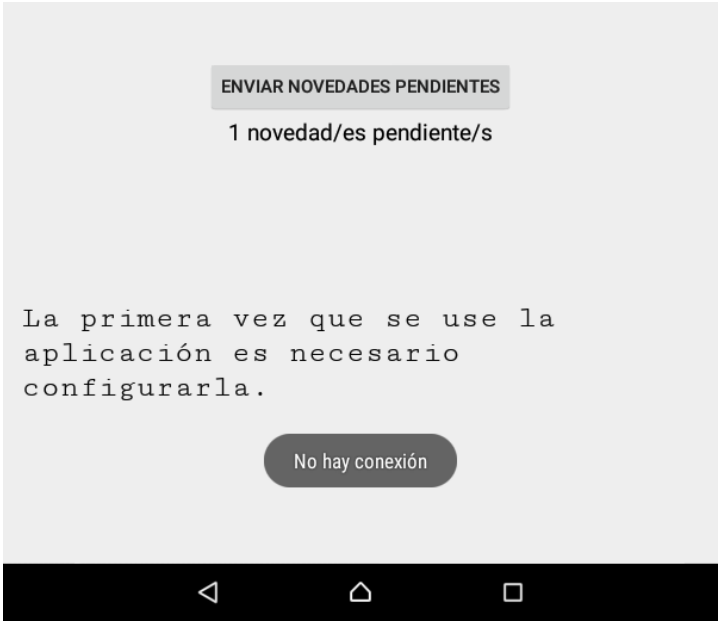

Figura 4-19. Notificación de no hay conexión.

En este momento es necesario comprobar que la novedad almacenada se manda de las dos maneras posibles que permite la aplicación: desde el Button "Enviar novedades pendientes" de la actividad MainActivity y completando otra novedad. La novedad generada en la Figura 4-18 se mandará mediante el Button de la MainActivity y posteriormente se generará otra novedad sin conexión seguida de otra que sí tenga para comprobar que se mandan ambas.

En la Figura 4-20 se puede observar el Toast que informa de que el envío es correcto, así como que se borra del dispositivo. En la Figura 4-21 se comprueba, comparando la hora, que la novedad recibida se corresponde con la que se puede ver en la Figura 4-18.

Figura 4-20. Envío desde la actividad MainActivity.


```xml
<novedad version="20162017" titulacion="CG_ST" fecha="02/03/2017 16:22">
  <curso>c1</curso>
  <asignatura militar="S">a6013</asignatura>
  <hora>7</hora>
  <horas_clase>1</horas_clase>
  <profesor>p85123964</profesor>
  <grupo>g12</grupo>
▼<alumnos>
    <alumno situacion="s1">a50665943</alumno>
    <alumno situacion="s4">a79978988</alumno>
    <alumno situacion="s11">a95983709</alumno>
    <jefe_grupo>a50665943</jefe_grupo>
  </alumnos>
▼<firma>
    /9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
    AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCADYAjgDASIA
    AAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3
    p6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEA
    BhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElK
    uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDgv+Df
    zElwkVxILi4EpadirMoK5P7b/wDEIP8A8EW/+hJ/aA/8P3r3/wApaT/g0F/5Qt+DP+zgfjz/AOn7
    /iFH/wCCLf8A0JP7OH/h+9e/+UtH/FTP/wAFW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiFH/AOCI
```


Figura 4-21. Novedad recibida desde la actividad MainActivity.

Como se dijo anteriormente para comprobar el otro método de envío de novedades pendientes es necesario generar otra novedad sin conexión seguida de otra con conexión. En la Figura 4-22 se observa las distintas horas a las que se generan ambas novedades.

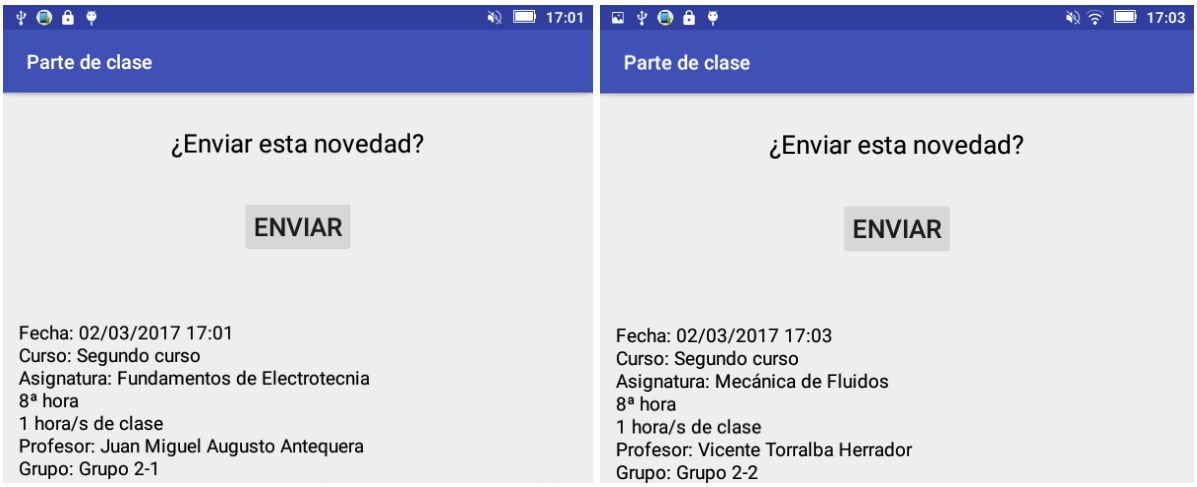

Figura 4-22. Novedad generada sin conexión y novedad generada con conexión.

Pulsando el Button "Enviar" se comprueba que se reciben en el servidor ambas novedades (Figura 4-23).

Figura 4-23. Recepción de ambas novedades.

4.1.3 Caché de profesores

4.1.3.1 Recomendación

Para comprobar que la caché de profesores funciona correctamente se comienza a generar una novedad con la misma asignatura que se puede ver en la Figura 4-17. Recordando lo que hace la clase CacheProfesores, cuando se crea el documento XML de la novedad se añade en una tabla de una base de datos SQL el ID de la asignatura y del profesor en sus respectivas columnas y la próxima vez que se inicie una novedad con la misma asignatura aparecerá dicho profesor recomendado en la *activity* Profesor.

Como se puede ver en la Figura 4-24 la aplicación correctamente recomienda el profesor que se eligió previamente.

Figura 4-24. Recomendación de profesor basada en la base de datos.

4.1.3.2 Borrado de la caché

Como se dijo anteriormente la caché de profesores se borra cuando se descarga una nueva base de datos desde la actividad Config. En esta prueba se descargará la Base de datos de Infantería de Marina sin titulación y se comprueba que, al llegar a la actividad Profesor habiendo elegido la misma asignatura que se seleccionó previamente, la aplicación no recomienda ningún profesor puesto que la caché se encuentra vacía.

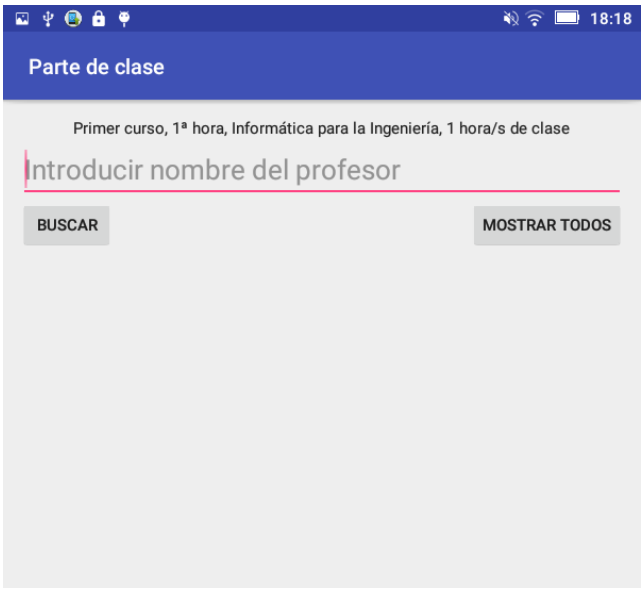

Figura 4-25. Caché borrada.

4.2 Encuesta de uso de la aplicación

Uno de los requerimientos que se planteaba al principio del desarrollo de este proyecto era que la aplicación fuese sencilla e intuitiva de usar. Debido a que este parámetro es difícil de medir y es una opinión subjetiva de cada individuo, se decide realizar una sencilla encuesta.

Esta encuesta se realiza a 30 Alféreces de Fragata y Alféreces Alumno de las promociones 417 y 147 respectivamente. En ella se le propone a cada uno de los alumnos que simulen ser el Jefe de Grupo y tienen que usar la aplicación para rellenar la novedad de su grupo. Se les dan directrices de qué titulación, curso, asignatura, profesor, grupo y situaciones de los alumnos tienen que elegir, pero no de cómo manejar la aplicación. Tras esa pequeña prueba se les pide que valoren de 0 a 10 lo sencilla e intuitiva de usar que les ha parecido la aplicación, entendiéndose como sencilla que la aplicación no se encuentra excesivamente cargada de elementos y como intuitiva que es fácil de usar aun siendo la primera vez que se tenga contacto con la aplicación. En el Anexo III: Encuesta de uso de la aplicación se puede ver la tabla con cada uno de los valores de la encuesta.

SENCILLA	INTUITIVA
8.884101, 9.582565	8.41518, 9.18482

Tabla 4-1. Intervalos de confianza del 95% obtenidos de la encuesta.

Observando las valoraciones obtenidas, cabe destacar que un 90% de los encuestados valoran con un sobresaliente la sencillez de la aplicación, mientras que un 66,66% valoran de igual manera lo intuitiva que es de usar la aplicación. Analizando esos datos, así como los intervalos de confianza del 95% que pueden verse en la Tabla 4-1 se puede decir que, como se pretendía, la aplicación es sencilla e intuitiva de usar.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Una vez realizas las pruebas anteriormente descritas y basándonos en los requerimientos de la aplicación que se nombraron en el capítulo 3.1, se puede llegar a las siguientes conclusiones:

- Entre los datos que se mandan al servidor se encuentran todos los que se incluyen en el actual parte de clase así como algún dato más, como la hora a la que se genera la novedad o alguna posible situación más de las que hay actualmente, por ejemplo, que el alumno llegue tarde a clase.
- Con esta aplicación y una vez se configurase el servidor que se usaría se eliminaría totalmente la dependencia de papel que existe ahora mismo en la Escuela Naval Militar, tanto material como personal, para la gestión de faltas de asistencia.
- Se mantiene la figura del Jefe de Grupo que será el alumno responsable de realizar la novedad de su grupo así como de portar el dispositivo en el que vaya instalada la aplicación.
- La aplicación permite cambiar al instante entre una titulación y otra, posibilitando que un mismo dispositivo puedan usarlo alumnos de distintos planes de estudios. El único requerimiento es que el dispositivo tenga conexión a Internet para poder descargar las distintas bases de datos.
- Los datos generados de cada novedad se mandan a un servidor en forma de documento XML.
- Basándose en la encuesta que se puede ver en la sección 4.2 se puede concluir que la aplicación tiene una interfaz sencilla e intuitiva que, además, permite volver atrás en todas las ventanas para poder corregir posibles errores.
- El programa permite trabajar sin conexión, almacenando las novedades que no hayan podido enviarse en la memoria interna del dispositivo. Estas novedades se podrán enviar más tarde cuando haya conexión desde la ventana principal de la aplicación o cuando se genera otra novedad, mandándose ambas.

En resumen, la aplicación cumple todos los requisitos de los que se partía al inicio de este proyecto. Sin embargo, para poder integrar este sistema totalmente en la Escuela Naval Militar será necesario configurar un servidor que analice los datos que recibe de la aplicación, así como una serie de posibles mejoras que se expondrán a continuación.

5.2 Líneas Futuras

En este apartado se pretende proponer una serie de líneas futuras en el ámbito de este proyecto, así como mejoras de la propia aplicación que por falta de tiempo no se han podido implementar.

- Lo primero, aunque no sería una tarea específica de la aplicación, sería comprobar que los archivos XML de las bases de datos son válidos y están bien formados. En este caso la validación se haría comprobando el DTD del archivo. La comprobación de que los documentos XML de las bases de datos son válidos y están bien formados debería ser realizada por el servidor desde el que se descargarían dichas bases de datos y al cual se mandarían las novedades generadas con la aplicación, que en este caso sería gestionado por la Jefatura de Estudios de la Escuela Naval Militar.

- Una posible línea futura para un próximo proyecto sería el de analizar los datos obtenidos en el servidor al recabar las distintas novedades. Se podrían generar estadísticas, gráficos e incluso avisos en caso de que un alumno se ausente en demasiadas clases. Además, todo esto se podría realizar en un entorno de trabajo personalizable y más sencillo a la vista que trabajar sobre un terminal.

En cuanto a las mejoras específicas de la aplicación se podrían incluir multitud de funcionalidades adicionales, algunas de las cuales se indican a continuación:

- Introducción de un identificador propio de cada dispositivo. Esto podría hacerse usando un UUID (*Universally Unique Identifier*, Identificador único universal) que, como su propio nombre indica, identifique cada dispositivo de manera única e inequívocamente. Esto sería relativamente sencillo de implementar ya que en Java existe una clase UUID que genera un código de 128 bits [18]. Podría incluirse el identificador del dispositivo en el XML a la hora de mandar la novedad al servidor y, de esta manera, poder saber rápidamente qué dispositivo es el que ha realizado el envío. Esto sería útil en caso de que hubiera algún problema con alguno de los dispositivos.

- Una posible medida de seguridad para evitar que un alumno genere una novedad sin que el profesor la firme sería que siempre que se mandara al servidor una novedad de un profesor, se le mandara a este un correo electrónico informando de ello. De esta manera si un profesor firma 7 novedades a lo largo de un día, deberá recibir 7 correos electrónicos, si no, es que hay algún problema que habría que investigar. Para esto sería necesario modificar las bases de datos para introducir un atributo en los elementos “profesor” con la dirección de correo electrónico de los mismos.
  Sin embargo, este método podría ocasionar demasiados correos electrónicos a los profesores pudiendo llegar a ser molesto. Para solucionar esto se podrían hacer dos cosas: crear una dirección de correo electrónico distinta de la personal de cada profesor cuyo único propósito sea el de recibir estas notificaciones o incluir la posibilidad en la aplicación de elegir si se quiere que se mande dicho correo electrónico o no.

- Para mejorar la rapidez de envío de novedades así como su posterior gestión en el servidor, estas podrían mandarse como archivos comprimidos en lugar de como texto. Al igual que con el UUID, esto no sería excesivamente complicado de implementar puesto que existe una librería en Java, java.utils.zip, con métodos que permiten generar ficheros comprimidos en formato ZIP.

- Ahora mismo en la aplicación sólo se autentifica utilizando HTTPS al servidor, pero el cliente no necesita ningún tipo de autentificación. Una posible mejora sería incluir esta autentificación del cliente a través de un certificado. Para ello se podría reutilizar la clase CustomSSLSocketFactory que se usa en este proyecto.

- Por último, y relativo a la interfaz de la aplicación, se podría incluir que al pulsar el nombre de alguno de los alumnos que se pueden ver en la *activity* Alumnos se viera la foto de dicho alumno en una ventana emergente. Para poder realizar esto sería necesario, antes de nada, disponer de una foto de cada alumno, lo cual en la ENM no supone ningún problema puesto que todos los años se realizan orlas con las fotos de todos los alumnos. Lo siguiente sería subirlas al servidor e introducir en las bases de datos un atributo en los elementos "alumno" que fuera la URL en la que se encuentre cada una de las fotos. Por último, en la aplicación solo sería necesario incluir que al pulsar un nombre se accediese a la URL de su respectiva foto y mostrarla por pantalla. El problema que puede ocasionar esta funcionalidad es que para poder visualizar las imágenes sería necesario disponer de conexión a Internet o que ya se hubiese accedido alguna vez a la URL deseada y se guardaran las imágenes en la memoria interna del dispositivo.

6 BIBLIOGRAFÍA

[1] «Armada Española» [En línea]. Available: http://www.armada.mde.es. [Último acceso: 23/01/2017].

[2] B. A. Z. Rodríguez, «La inasistencia a clases de los estudiantes influye en el rendimiento académico» Universidad Tecnológica Equinoccial, Chone, Manabí, Ecuador, 2012.

[3] «Universia» [En línea]. Available: http://noticias.universia.com.ar/educacion/. [Último acceso: 23/01/2017].

[4] «Inika School Management System» [En línea]. Available: www.inika.net/es/que-es-inika. [Último acceso: 29/01/2017].

[5] «Alexia Educación» Cospa & Agilmic [En línea]. Available: http://www.alexiaeducacion.com/. [Último acceso: 29/01/2017].

[6] «Grupo CF Developer» Grupo CF Developer, [En línea]. Available: http://www.grupocfdeveloper.com. [Último acceso: 31/01/2017].

[7] iStudiez, «iTeacherBook» iStudiez, [En línea]. Available: http://iteacherbook.com/. [Último acceso: 31/01/2017].

[8] Teacher Aide Pro, «Teacher Aide» Teacher Aide Pro, [En línea]. Available: http://www.teacheraidepro.com/. [Último acceso: 19/02/2017].

[9] T. Groussard, Java 7. Los Fundamentos del Lenguaje de Java, ediciones ENI, 9782746073180, 2014.

[10] M. Báez, Introducción a Android, E.M.E. Editorial, 978-84-96285-39-5, 2012.

[11] Symantec, «A Window into Mobile Device Security» 2011. [En línea]. Available: http://www.symantec.com/content/en/us/about/media/pdfs/symc_mobile_device_security_june2011.pdf. [Último acceso: 17/02/2017].

[12] «Android» Google, [En línea]. Available: https://www.android.com/intl/es_es/history. [Último acceso: 04/02/2017].

[13] «Android Studio» Google, [En línea]. Available: https://developer.android.com/studio/index.html?hl=es-419. [Último acceso: 04/02/2017].

[14] «Desarrollo Web» [En línea]. Available: http://www.desarrolloweb.com. [Último acceso: 01/02/2017].

[15] «Abrir Llave» [En línea]. Available: http://www.abrirllave.com/dtd/. [Último acceso: 10/02/2017].

[16] Android, «Android Developer» Android, [En línea]. Available: https://developer.android.com/index.html.

[17] R. Asorey Cacheda, Apuntes Ampliación de Informática, Centro Universitario de la Defensa, Curso 2016/2017.

[18] Oracle, «Oracle Java Documentation» Oracle, [En línea]. Available: http://docs.oracle.com/javase/7/docs/index.html.

[19] «SGOLIVER» [En línea]. Available: www.sgoliver.net.

[20] «Stack overflow» [En línea]. Available: www.stackoverflow.com.

[21] «Programación Ya» [En línea]. Available: http://www.tutorialesprogramacionya.com/javaya/androidya/androidstudioya/index.php?inicio=0. [Último acceso: 14/01/2017].

[22] Youtube, «Desarrollo de un Paint en Android» [En línea]. Available: https://www.youtube.com/watch?v=GAr_agEokr8. [Último acceso: 18/02/2017].

[23] Youtube, «Hilos y AsyncTask» [En línea]. Available: https://www.youtube.com/watch?v=cxUJJh29PBg&t=44s. [Último acceso: 25/01/2017].

ANEXO I: BASES DE DATOS USADAS

Base de datos de Cuerpo General sin titulación

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

<!DOCTYPE novedades [
    <!ELEMENT novedades (cursos, asignaturas, grupos, profesores, alumnos, situaciones)>
        <!ATTLIST novedades version CDATA #REQUIRED>
        <!ATTLIST novedades titulacion CDATA #REQUIRED>
    <!ELEMENT cursos (curso)+>
    <!ELEMENT curso (nombre)>
        <!ATTLIST curso id ID #REQUIRED>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT asignaturas (asignatura)+>
    <!ELEMENT asignatura (nombre)>
        <!ATTLIST asignatura id ID #REQUIRED>
        <!ATTLIST asignatura curso IDREF #REQUIRED>
        <!ATTLIST asignatura militar CDATA #REQUIRED>
    <!ELEMENT grupos (grupo)+>
    <!ELEMENT grupo (nombre)>
        <!ATTLIST grupo id ID #REQUIRED>
        <!ATTLIST grupo curso IDREF #REQUIRED>
    <!ELEMENT profesores (profesor)+>
    <!ELEMENT profesor (nombre)>
        <!ATTLIST profesor id ID #REQUIRED>
    <!ELEMENT alumnos (alumno)+>
    <!ELEMENT alumno (nombre)>
        <!ATTLIST alumno id ID #REQUIRED>
        <!ATTLIST alumno grupo IDREF #REQUIRED>
    <!ELEMENT situaciones (situacion)+>
    <!ELEMENT situacion (nombre)>
        <!ATTLIST situacion id CDATA #REQUIRED>
]>


<novedades version="20162017" titulacion="CG_ST">

 <cursos>
  <curso id="c1">
```

```
  <nombre>Primer curso</nombre>
 </curso>
 <curso id="c2">
  <nombre>Segundo curso</nombre>
 </curso>
</cursos>

<asignaturas>
 <asignatura id="a6003" curso="c1" militar="N">
  <nombre>Informática para la Ingeniería</nombre>
 </asignatura>
 <asignatura id="a6023" curso="c2" militar="N">
  <nombre>Fundamentos de Electrotecnia</nombre>
 </asignatura>
 <asignatura id="a6001" curso="c1" militar="N">
  <nombre>Física I</nombre>
 </asignatura>
 <asignatura id="a6020" curso="c2" militar="N">
  <nombre>Mecánica de Fluidos</nombre>
 </asignatura>
 <asignatura id="a6029" curso="c2" militar="S">
  <nombre>Maniobra y Navegación II</nombre>
 </asignatura>
 <asignatura id="a6013" curso="c1" militar="S">
  <nombre>Maniobra y Navegación I</nombre>
 </asignatura>
</asignaturas>

<grupos>
 <grupo id="g11" curso="c1">
    <nombre>Grupo 1-1</nombre>
 </grupo>
 <grupo id="g12" curso="c1">
    <nombre>Grupo 1-2</nombre>
 </grupo>
 <grupo id="g21" curso="c2">
    <nombre>Grupo 2-1</nombre>
 </grupo>
 <grupo id="g22" curso="c2">
    <nombre>Grupo 2-2</nombre>
 </grupo>
```

```
</grupos>

<profesores>
 <profesor id="p12654963">
  <nombre>Norberto Fernández García</nombre>
 </profesor>
 <profesor id="p78951456">
  <nombre>Rafael Asorey Cacheda</nombre>
 </profesor>
 <profesor id="p85123964">
  <nombre>Vicente Torralba Herrador</nombre>
 </profesor>
 <profesor id="p59874512">
  <nombre>Juan Miguel Augusto Antequera</nombre>
 </profesor>
</profesores>

<alumnos>
 <alumno id="a52111655" grupo="g11">
   <nombre>Rafael García Ruiz</nombre>
 </alumno>
 <alumno id="a93670655" grupo="g11">
   <nombre>Álvaro Andrés de la Cuadra</nombre>
 </alumno>
 <alumno id="a11726733" grupo="g11">
   <nombre>Rafael Hermida Mayán</nombre>
 </alumno>
 <alumno id="a50665943" grupo="g12">
   <nombre>Santiago Martín Marco</nombre>
 </alumno>
 <alumno id="a79978988" grupo="g12">
   <nombre>Antonio Liarte Pérez</nombre>
 </alumno>
 <alumno id="a95983709" grupo="g12">
   <nombre>Jorge Medio Fabián</nombre>
 </alumno>
 <alumno id="a18734080" grupo="g21">
   <nombre>Laura Vitalia González Martínez</nombre>
 </alumno>
 <alumno id="a36729252" grupo="g21">
   <nombre>Borja Moriano Rivera</nombre>
```

```
 </alumno>
 <alumno id="a44928896" grupo="g21">
   <nombre>Ernesto Golmayo Fernández</nombre>
 </alumno>
 <alumno id="a54290619" grupo="g22">
   <nombre>Javier Gago Codesal</nombre>
 </alumno>
 <alumno id="a62536929" grupo="g22">
   <nombre>Alejandro Ameneiro Hernández de Armijo</nombre>
 </alumno>
 <alumno id="a16185260" grupo="g22">
   <nombre>Antonio Delage Román</nombre>
 </alumno>
</alumnos>

<situaciones>
 <situacion id="s1">
  <nombre>Sin Novedad</nombre>
 </situacion>
 <situacion id="s2">
  <nombre>Convalidado</nombre>
 </situacion>
 <situacion id="s3">
  <nombre>Rebajado</nombre>
 </situacion>
 <situacion id="s4">
  <nombre>Hospitalizado</nombre>
 </situacion>
 <situacion id="s5">
  <nombre>En Zulu</nombre>
 </situacion>
 <situacion id="s6">
  <nombre>Revista Médica</nombre>
 </situacion>
 <situacion id="s7">
  <nombre>Llega tarde</nombre>
 </situacion>
 <situacion id="s8">
  <nombre>Falto</nombre>
 </situacion>
 <situacion id="s9">
```

```
  <nombre>Permiso</nombre>
 </situacion>
 <situacion id="s10">
  <nombre>Comisionado</nombre>
 </situacion>
 <situacion id="s11">
  <nombre>Guardia</nombre>
 </situacion>
</situaciones>

</novedades>
```

Base de datos de Infantería de Marina sin titulación

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

<!DOCTYPE novedades [
    <!ELEMENT novedades (cursos, asignaturas, grupos, profesores, alumnos, situaciones)>
            <!ATTLIST novedades version CDATA #REQUIRED>
            <!ATTLIST novedades titulacion CDATA #REQUIRED>
    <!ELEMENT cursos (curso)+>
    <!ELEMENT curso (nombre)>
            <!ATTLIST curso id ID #REQUIRED>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT asignaturas (asignatura)+>
    <!ELEMENT asignatura (nombre)>
            <!ATTLIST asignatura id ID #REQUIRED>
            <!ATTLIST asignatura curso IDREF #REQUIRED>
            <!ATTLIST asignatura militar CDATA #REQUIRED>
    <!ELEMENT grupos (grupo)+>
    <!ELEMENT grupo (nombre)>
            <!ATTLIST grupo id ID #REQUIRED>
            <!ATTLIST grupo curso IDREF #REQUIRED>
    <!ELEMENT profesores (profesor)+>
    <!ELEMENT profesor (nombre)>
            <!ATTLIST profesor id ID #REQUIRED>
    <!ELEMENT alumnos (alumno)+>
    <!ELEMENT alumno (nombre)>
            <!ATTLIST alumno id ID #REQUIRED>
            <!ATTLIST alumno grupo IDREF #REQUIRED>
    <!ELEMENT situaciones (situacion)+>
```

```
    <!ELEMENT situacion (nombre)>
        <!ATTLIST situacion id CDATA #REQUIRED>
]>


<novedades version="20162017" titulacion="IM_ST">

 <cursos>
  <curso id="c1">
   <nombre>Primer curso</nombre>
  </curso>
  <curso id="c2">
   <nombre>Segundo curso</nombre>
  </curso>
 </cursos>

 <asignaturas>
  <asignatura id="a6003" curso="c1" militar="N">
   <nombre>Informática para la Ingeniería</nombre>
  </asignatura>
  <asignatura id="a6023" curso="c2" militar="N">
   <nombre>Fundamentos de Electrotecnia</nombre>
  </asignatura>
  <asignatura id="a6001" curso="c1" militar="N">
   <nombre>Física I</nombre>
  </asignatura>
  <asignatura id="a6020" curso="c2" militar="N">
   <nombre>Mecánica de Fluidos</nombre>
  </asignatura>
  <asignatura id="a6029" curso="c2" militar="S">
   <nombre>Táctica Anfibia II</nombre>
  </asignatura>
  <asignatura id="a6013" curso="c1" militar="S">
   <nombre>Táctica Anfibia I</nombre>
  </asignatura>
 </asignaturas>

 <grupos>
  <grupo id="g11" curso="c1">
     <nombre>Grupo 1-1</nombre>
  </grupo>
```

```
 <grupo id="g21" curso="c2">
    <nombre>Grupo 2-1</nombre>
 </grupo>
 <grupo id="g22" curso="c2">
    <nombre>Grupo 2-2</nombre>
 </grupo>
</grupos>

<profesores>
 <profesor id="p12654963">
  <nombre>Norberto Fernández García</nombre>
 </profesor>
 <profesor id="p78951456">
  <nombre>Rafael Asorey Cacheda</nombre>
 </profesor>
 <profesor id="p85123964">
  <nombre>Vicente Torralba Herrador</nombre>
 </profesor>
 <profesor id="p59874512">
  <nombre>Juan Miguel Augusto Antequera</nombre>
 </profesor>
</profesores>

<alumnos>
 <alumno id="a52111655" grupo="g11">
  <nombre>Rafael García Ruiz</nombre>
 </alumno>
 <alumno id="a93670655" grupo="g11">
  <nombre>Álvaro Andrés de la Cuadra</nombre>
 </alumno>
 <alumno id="a11726733" grupo="g11">
  <nombre>Rafael Hermida Mayán</nombre>
 </alumno>
 <alumno id="a50665943" grupo="g11">
  <nombre>Santiago Martín Marco</nombre>
 </alumno>
 <alumno id="a79978988" grupo="g11">
  <nombre>Antonio Liarte Pérez</nombre>
 </alumno>
 <alumno id="a95983709" grupo="g11">
  <nombre>Jorge Medio Fabián</nombre>
```

```
 </alumno>
 <alumno id="a18734080" grupo="g21">
   <nombre>Laura Vitalia González Martínez</nombre>
 </alumno>
 <alumno id="a36729252" grupo="g21">
   <nombre>Borja Moriano Rivera</nombre>
 </alumno>
 <alumno id="a44928896" grupo="g21">
   <nombre>Ernesto Golmayo Fernández</nombre>
 </alumno>
 <alumno id="a54290619" grupo="g22">
   <nombre>Javier Gago Codesal</nombre>
 </alumno>
 <alumno id="a62536929" grupo="g22">
   <nombre>Alejandro Ameneiro Hernández de Armijo</nombre>
 </alumno>
 <alumno id="a16185260" grupo="g22">
   <nombre>Antonio Delage Román</nombre>
 </alumno>
</alumnos>

<situaciones>
 <situacion id="s1">
  <nombre>Sin Novedad</nombre>
 </situacion>
 <situacion id="s2">
  <nombre>Convalidado</nombre>
 </situacion>
 <situacion id="s3">
  <nombre>Rebajado</nombre>
 </situacion>
 <situacion id="s4">
  <nombre>Hospitalizado</nombre>
 </situacion>
 <situacion id="s5">
  <nombre>En Zulu</nombre>
 </situacion>
 <situacion id="s6">
  <nombre>Revista Médica</nombre>
 </situacion>
 <situacion id="s7">
```

```
  <nombre>Llega tarde</nombre>
 </situacion>
 <situacion id="s8">
  <nombre>Falto</nombre>
 </situacion>
 <situacion id="s9">
  <nombre>Permiso</nombre>
 </situacion>
 <situacion id="s10">
  <nombre>Comisionado</nombre>
 </situacion>
 <situacion id="s11">
  <nombre>Guardia</nombre>
 </situacion>
</situaciones>

</novedades>
```

Base de datos de Cuerpo General con titulación

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

<!DOCTYPE novedades [
    <!ELEMENT novedades (cursos, asignaturas, grupos, profesores, alumnos, situaciones)>
            <!ATTLIST novedades version CDATA #REQUIRED>
            <!ATTLIST novedades titulacion CDATA #REQUIRED>
    <!ELEMENT cursos (curso)+>
    <!ELEMENT curso (nombre)>
            <!ATTLIST curso id ID #REQUIRED>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT asignaturas (asignatura)+>
    <!ELEMENT asignatura (nombre)>
            <!ATTLIST asignatura id ID #REQUIRED>
            <!ATTLIST asignatura curso IDREF #REQUIRED>
            <!ATTLIST asignatura militar CDATA #REQUIRED>
    <!ELEMENT grupos (grupo)+>
    <!ELEMENT grupo (nombre)>
            <!ATTLIST grupo id ID #REQUIRED>
            <!ATTLIST grupo curso IDREF #REQUIRED>
    <!ELEMENT profesores (profesor)+>
    <!ELEMENT profesor (nombre)>
```

```
            <!ATTLIST profesor id ID #REQUIRED>
        <!ELEMENT alumnos (alumno)+>
        <!ELEMENT alumno (nombre)>
            <!ATTLIST alumno id ID #REQUIRED>
            <!ATTLIST alumno grupo IDREF #REQUIRED>
        <!ELEMENT situaciones (situacion)+>
        <!ELEMENT situacion (nombre)>
            <!ATTLIST situacion id CDATA #REQUIRED>
]>


<novedades version="20162017" titulacion="CG_CT">

 <cursos>
  <curso id="c1">
   <nombre>Primer curso</nombre>
  </curso>
 </cursos>

 <asignaturas>
  <asignatura id="a6018" curso="c1" militar="N">
   <nombre>Termodinámica y Transmisión de calor</nombre>
  </asignatura>
  <asignatura id="a6023" curso="c1" militar="N">
   <nombre>Fundamentos de Electrotecnia</nombre>
  </asignatura>
  <asignatura id="a6033" curso="c1" militar="N">
   <nombre>Tecnología Electrónica</nombre>
  </asignatura>
  <asignatura id="a6020" curso="c1" militar="N">
   <nombre>Mecánica de Fluidos</nombre>
  </asignatura>
  <asignatura id="a6013" curso="c1" militar="S">
   <nombre>Maniobra y Navegación I</nombre>
  </asignatura>
 </asignaturas>

 <grupos>
  <grupo id="g11" curso="c1">
     <nombre>Grupo 1-1</nombre>
  </grupo>
```

```
 <grupo id="g12" curso="c1">
    <nombre>Grupo 1-2</nombre>
 </grupo>
 <grupo id="g13" curso="c1">
    <nombre>Grupo 1-3</nombre>
 </grupo>
</grupos>

<profesores>
 <profesor id="p12654963">
  <nombre>Norberto Fernández García</nombre>
 </profesor>
 <profesor id="p78951456">
  <nombre>Rafael Asorey Cacheda</nombre>
 </profesor>
 <profesor id="p85123964">
  <nombre>Vicente Torralba Herrador</nombre>
 </profesor>
 <profesor id="p59874512">
  <nombre>Juan Miguel Augusto Antequera</nombre>
 </profesor>
</profesores>

<alumnos>
 <alumno id="a52111655" grupo="g11">
   <nombre>Rafael García Ruiz</nombre>
 </alumno>
 <alumno id="a93670655" grupo="g11">
   <nombre>Álvaro Andrés de la Cuadra</nombre>
 </alumno>
 <alumno id="a11726733" grupo="g11">
   <nombre>Rafael Hermida Mayán</nombre>
 </alumno>
 <alumno id="a50665943" grupo="g11">
   <nombre>Santiago Martín Marco</nombre>
 </alumno>
 <alumno id="a79978988" grupo="g12">
   <nombre>Antonio Liarte Pérez</nombre>
 </alumno>
 <alumno id="a95983709" grupo="g12">
   <nombre>Jorge Medio Fabián</nombre>
```

```
 </alumno>
 <alumno id="a18734080" grupo="g12">
   <nombre>Laura Vitalia González Martínez</nombre>
 </alumno>
 <alumno id="a36729252" grupo="g12">
   <nombre>Borja Moriano Rivera</nombre>
 </alumno>
 <alumno id="a44928896" grupo="g13">
   <nombre>Ernesto Golmayo Fernández</nombre>
 </alumno>
 <alumno id="a54290619" grupo="g13">
   <nombre>Javier Gago Codesal</nombre>
 </alumno>
 <alumno id="a62536929" grupo="g13">
   <nombre>Alejandro Ameneiro Hernández de Armijo</nombre>
 </alumno>
 <alumno id="a16185260" grupo="g13">
   <nombre>Antonio Delage Román</nombre>
 </alumno>
</alumnos>

<situaciones>
 <situacion id="s1">
  <nombre>Sin Novedad</nombre>
 </situacion>
 <situacion id="s2">
  <nombre>Convalidado</nombre>
 </situacion>
 <situacion id="s3">
  <nombre>Rebajado</nombre>
 </situacion>
 <situacion id="s4">
  <nombre>Hospitalizado</nombre>
 </situacion>
 <situacion id="s5">
  <nombre>En Zulu</nombre>
 </situacion>
 <situacion id="s6">
  <nombre>Revista Médica</nombre>
 </situacion>
 <situacion id="s7">
```

```
  <nombre>Llega tarde</nombre>
 </situacion>
 <situacion id="s8">
  <nombre>Falto</nombre>
 </situacion>
 <situacion id="s9">
  <nombre>Permiso</nombre>
 </situacion>
 <situacion id="s10">
  <nombre>Comisionado</nombre>
 </situacion>
 <situacion id="s11">
  <nombre>Guardia</nombre>
 </situacion>
</situaciones>

</novedades>
```

ANEXO II: DOCUMENTO XML DE NOVEDAD

<novedad version="20162017" titulacion="CG_ST" fecha="02/03/2017 13:13">
<curso>c1</curso>
<asignatura militar="N">a6003</asignatura>
<hora>4</hora>
<horas_clase>1</horas_clase>
<profesor>p12654963</profesor>
<grupo>g11</grupo>
<alumnos>
<alumno situacion="s1">a52111655</alumno>
<alumno situacion="s1">a93670655</alumno>
<alumno situacion="s1">a11726733</alumno>
<jefe_grupo>a52111655</jefe_grupo>
</alumnos>
<firma>
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/2wBDAQEBAQEBAQEBAQEBAQEBA
QEB
AQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQEBAQH/wAARCADYAjgDA
SIA
AhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQA
AAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3
ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWm
p6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEA
AwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgECBAQDBAcFBAQAAQJ3AAECAxEEBSEx
BhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElK
U1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3
uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDgv+Df
P/g31/4Jvf8ABR7/AIJveHP2mv2mvDXxY1P4m6l8Wfip4Purvwf8VdV8K6KdE8K6rp1po6Jo9rpt
zElwkVxILi4EpadirMoK5P7b/wDEIP8A8EW/+hJ/aA/8P3r3/wApaT/g0F/5Qt+DP+zgfjz/AOn7
R6/p/oA/mB/4hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg//BFv/oSf2gP/AA/evf8Aylr+n6igD+YH
/iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCL
f/Qk/tAf+H717/5S0f8AEIP/AMEW/wDoSf2gP/D969/8pa/p+ooA/mB/4hB/+CLf/Qk/tAf+H717
/wCUtH/EIP8A8EW/+hJ/aA/8P3r3/wApa/p+ooA/mB/4hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg/
/BFv/oSf2gP/AA/evf8Aylr+n6igD+YH/iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2
gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCLf/Qk/tAf+H717/5S0f8AEIP/AMEW/wDoSf2gP/D969/8
pa/p+ooA/mB/4hB/+CLf/Qk/tAf+H717/wCUtH/EIP8A8EW/+hJ/aA/8P3r3/wApa/p+ooA/mB/4
hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg//BFv/oSf2gP/AA/evf8Aylr+n6igD+YH/iEH/wCCLf8A
0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCLf/Qk/tAf+H71
7/5S0f8AEIP/AMEW/wDoSf2gP/D969/8pa/p+ooA/mB/4hB/+CLf/Qk/tAf+H717/wCUtH/EIP8A
8EW/+hJ/aA/8P3r3/wApa/p+ooA/mB/4hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg//BFv/oSf2gP/
AA/evf8Aylr+n6igD+YH/iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KW
v6fqKAP5gf8AiEH/AOCLf/Qk/tAf+H717/5S0f8AEIP/AMEW/wDoSf2gP/D969/8pa/p+ooA/mB/
4hB/+CLf/Qk/tAf+H717/wCUtH/EIP8A8EW/+hJ/aA/8P3r3/wApa/p+ooA/mB/4hB/+CLf/AEJP
7QH/AIfvXv8A5S0f8Qg//BFv/oSf2gP/AA/evf8Aylr+n6igD+YH/iEH/wCCLf8A0JP7QH/h+9e/
+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8AiEH/AOCLf/Qk/tAf+H717/5S0f8AEIP/
AMEW/wDoSf2gP/D969/8pa/p+ooA/mB/4hB/+CLf/Qk/tAf+H717/wCUtH/EIP8A8EW/+hJ/aA/8
P3r3/wApa/p+ooA/mB/4hB/+CLf/AEJP7QH/AIfvXv8A5S0f8Qg//BFv/oSf2gP/AA/evf8Aylr+
n6igD+YH/iEH/wCCLf8A0JP7QH/h+9e/+UtH/EIP/wAEW/8AoSf2gP8Aw/evf/KWv6fqKAP5gf8A
.
.
.
</firma>
</novedad>

ANEXO III: ENCUESTA DE USO DE LA APLICACIÓN

SENCILLA	INTUITIVA
9	8
10	10
10	9
10	10
10	9
10	8
10	10
9	9
9	9
9	10
9	10
10	8
10	8
9	7
10	9
9	7
10	10
9	10
10	9
10	9
9	8
10	9
9	8
9	9
7	8
8	9
9	10

6	6
9	9
9	9

Tabla A3-1. Resultados de la encuesta de uso de la aplicación.