Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

TRABAJO FIN DE GRADO

APLICACIÓN DE ATENCIÓN PERSONALIZADA DINÁMICA PARA INTERIORES BASADA EN TERMINALES ANDROID

Alumno: **David Miguel Poyatos Reina**

Tutor: Prof. D. Juan Carlos Cuevas Martínez
Depto.: Ingeniería de Telecomunicación

Septiembre 2014

Índice General

Índice de Figuras

Índice de Tablas

1 MEMORIA

1.1 Introducción

El objetivo de este Trabajo Fin de Grado es el de crear un servicio destinado a dar soporte a la comunicación de horarios de tutorías y avisos a los alumnos.

Se creará una aplicación para terminales con sistema operativo Android, que funcionará tanto en los terminales de profesores como en los alumnos y será la herramienta principal para hacer uso del servicio desarrollado. Los profesores publicarán sus horarios de tutorías y avisos, mientras que los alumnos podrán consultar los horarios y avisos de los profesores que hayan elegido en la aplicación. Asimismo, los alumnos recibirán automáticamente una notificación cuando un profesor realice algún cambio en sus tutorías.

Conjuntamente se desarrollará una aplicación servidora en la que se apoyará la aplicación móvil para guardar en la base de datos información de cuentas de profesores, horarios, avisos, etc., y para comunicarse con el servicio encargado de enviar las notificaciones automáticas a los alumnos.

El servicio se apoya en varias tecnologías:

- Android como sistema operativo para el que se desarrolla la aplicación.
- NFC y QR para la obtención de datos de profesores.
- HTTP como protocolo de comunicación entre aplicación y servidor.
- RSA, AES y SHA1 para garantizar autenticidad e integridad de los mensajes.
- MySQL como gestor de base de datos en el servidor y SQLite en la aplicación.

En adelante se referirá al proyecto como TutoGuia.

TutoGuia

Figura 1. Logo de TutoGuia

1.2 Objetivos

El objetivo principal de este trabajo de fin de grado es generar una aplicación cliente para un terminal Android que, a través de todas las capacidades de comunicación de que disponen dichos terminales, permita la recepción de información personalizada y de interés para el usuario.

Objetivos secundarios:

- La aplicación deberá permitir cierto grado de personalización y ajuste dentro del ámbito para el que se desarrolla.
- Se deberá dotar de un servicio externo que aporte la información a seleccionar dentro del ámbito en el que se desarrolla el proyecto.
- Se deberá crear/simular la infraestructura de apoyo al servicio que permita el control, localización y suministro de información al usuario.
- La aplicación a desarrollar deberá estar preparada para adaptarse a diferentes tipos de terminal de los que existan en el momento de la realización del trabajo.

Objetivos opcionales:

- Muestra de la información en dispositivos externos pertenecientes a la infraestructura del servicio, tales como tablets fijos, televisores, PCs, etc.
- Capacidad de mostrar publicidad según lo permita la plataforma de desarrollo.
- Accesos basados en certificados digitales.
- Cobro y tarificación de servicios a través de la misma aplicación.

1.3 Alcance

El proyecto pretende facilitar la consulta y gestión de avisos y tutorías a alumnos y profesores.

Con el uso de TutoGuia se daría un salto en la interacción profesor-alumno, ya que el sistema facilitaría la consulta de horarios de tutorías, lo que conllevaría a que mayor número de alumnos hiciesen uso de las tutorías.

Además, ante cualquier cambio o incidencia en el horario de tutorías se notifica automáticamente a los alumnos, evitando así posibles desplazamientos innecesarios, como podría suceder con el actual sistema (pequeños 'post-it' en la puerta del despacho).

A continuación se resumen las ventajas y desventajas que se presentan:

Ventajas:

- Simplificación en la publicación y consulta del horario de tutorías.
- Simplificación en la publicación y consulta de avisos.
- Ante cualquier cambio, se notifica al alumnado al instante.
- Optimización del tiempo del estudiante (puede ajustar su horario con precisión si quiere acudir a tutoría).

Desventajas:

- Es necesario que profesores y alumnos posean un terminal Android para instalar la aplicación. En caso de no poseerlo, se debe seguir utilizando el sistema tradicional.

1.4 Antecedentes

Actualmente los profesores informan de sus tutorías y avisos mayoritariamente a través de notas informativas colocadas en la puerta del despacho, o en menor medida, a través de las webs personales, correo electrónico o la plataforma de Docencia Virtual (ILIAS). Esta forma resulta poco práctica para los alumnos, ya que para consultar el horario de tutorías deben acudir a la puerta del despacho del profesor, ocasionando una pérdida de tiempo en caso de que no se encuentre en ese momento, o en caso de estar en horario de tutoría, haya tenido algún imprevisto y hubiese tenido que ausentarse.

En la actualidad el uso de teléfonos inteligentes está ampliamente extendidos en todos los ámbitos de la vida cotidiana, facilitando muchas tareas. Asimismo, en el ámbito universitario esta expansión no ha sido tan notable, como es el caso que se refiere en este proyecto. Por tanto, es de gran interés desarrollar una aplicación que facilite a la comunidad universitaria esta tarea.

1.5 Estado del arte

Actualmente no existe ninguna aplicación Android parecida a la que se ha desarrollado. Se ha encontrado un sistema [1], desarrollado por el Grupo de Tecnología Informática de la Universidad Politécnica de Valencia con las mismas bases que TutoGuia, pero no es válido para terminales actuales, por lo que no se puede utilizar.

No obstante, la tarea que se pretende facilitar con TutoGuia podría realizarse utilizando otra serie de herramientas:

- Calendario compartido de Google: con este método el profesor crearía un calendario con su horario de tutorías utilizando este servicio y lo compartiría con sus alumnos (debería añadir cada uno de los correos electrónicos de los alumnos con los que quiere compartir el calendario, lo que supone una tarea engorrosa para los profesores). En caso de insertar un aviso, tendría que añadir un nuevo evento al calendario. Esta forma podría resultar útil, pero no estaría tan optimizada como TutoGuia, ya que el envío de notificaciones no sería igual. Con el calendario se crearía una alarma que se activaría a cierta hora, o durante un día, mientras que con el sistema de notificaciones de TutoGuia, se avisaría al instante.
- Noticias de ILIAS (RSS): este método no es viable, ya que habría que utilizar una tercera aplicación que leyera las noticias, y dependiendo de la aplicación que se use (hay muchas en las tiendas de aplicaciones) se pueden tener unas funciones o no, por ejemplo, ante una nueva noticia, no se avisa al usuario, el usuario es el que debe abrir la aplicación y ver si hay algo nuevo. Además, para que este sistema funcionase y se crease una noticia, el profesor tendría que subir algún archivo (doc, pdf) a la plataforma ILIAS, lo cual sería tedioso realizar desde un dispositivo móvil.

1.6 Tecnologías usadas

La gran explosión en los últimos años de la tecnología y la telefonía ha causado que la mayoría de la población posea un teléfono móvil de última generación, con capacidades de cómputo y funciones similares, y a veces superiores a las de un ordenador.

1.6.1 Android

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, y también para relojes inteligentes, televisores y automóviles, inicialmente desarrollado por la empresa Android Inc., Google la respaldó económicamente y más tarde compró esta empresa en 2005. Android fue presentado en 2007 junto con la fundación Open Handset Alliance: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles. Es de código abierto, lo que quiere decir que cualquier desarrollador puede crear y desarrollar aplicaciones para este sistema. Su entorno de ejecución está basado en Java.

Está instalado en dispositivos de una gran variedad de marcas. Esto hace que según las características del terminal, podremos instalar una versión u otra del sistema y, por lo tanto, no se podrán ejecutar ni las mismas aplicaciones ni con las mismas prestaciones en todos ellos.

1.6.2 Near Field Communication

Near Field Communication (NFC – comunicación de campo cercano) es una tecnología inalámbrica que funciona en la banda de los 13,56 MHz (no es necesaria licencia para usarla) y que deriva de las etiquetas RFID (*Radio Frequency IDentification* – identificación por radiofrecuencia). Puede alcanzar velocidades de hasta 424 kbits/s, por lo que su enfoque más que para la transmisión de grandes cantidades de datos es para una comunicación instantánea sin necesidad de emparejamiento previo. Su alcance es muy reducido, como máximo 20 cm.

Puede funcionar en dos modos:

- Activo, en el que ambos equipos con chip NFC generan un campo electromagnético e intercambian datos.

- Pasivo, en el que solo hay un dispositivo activo y el otro aprovecha ese campo para intercambiar la información.

1.6.3 Quick Response (QR)

Un código QR (*Quick Response* - de respuesta rápida) es un código de barras de dos dimensiones que puede escanearse con la cámara de un teléfono móvil y que puede, según del tipo que sea, redirigir a una página web, mostrar un texto, enlazar a un número de teléfono, etc.

Comparados con el código de barras, los códigos QR pueden contener aproximadamente 100 veces más información. La matriz es leída usando un sensor CCD (*Chargue-Coupled Device* – dispositivo de carga acoplada – es un sensor ampliamente utilizado en cámaras digitales), los datos que se leen se guardan en la memoria y, usando el software, se analiza la imagen, se buscan los patrones necesarios y la posición en la que está la imagen y a partir de ahí se decodifica la información. Es resistente a símbolos distorsionados, por ejemplo, que no se encuentren en una superficie lisa o que se capte con una cierta inclinación. Además, implementa un error de corrección, por lo que sería posible leer un código que hubiera perdido como máximo un 30% de su información, como por ejemplo, códigos medio tapados o recortados.

Para hacer uso de los códigos QR en la aplicación se ha utilizado una librería, *zxing [2]*, que proporciona la herramientas necesarias para crear y leer códigos QR.

1.6.3.1 Estructura de un código QR

Estos códigos son símbolos matriciales con una estructura en rejilla en forma de cuadrado. Tiene varios patrones funcionales para que la lectura de los datos sea fácil. A continuación se van a explicar cada uno de los patrones de los que se compone un código.

- **Información de la versión:** indica la versión de código QR que se utiliza. Según la versión, la densidad del código QR puede ser mayor y por tanto contener más información. Hay 40 versiones: versión 1 de tamaño 21x21 hasta la versión 40 de tamaño 177x177. Cuanto mayor es la versión, mayor dificultad presenta la lectura.
- **Información del formato:** indica el formato de la información que contiene el código QR, si es una URL, es sólo texto, etc.

- **Corrección de errores y datos.** Los datos serán codificados en el área de datos, que es lo que representa la zona verde de la Figura 2. La información será codificada en símbolos binarios 1 y 0 que se convertirán en celdas negras y blancas y serán colocadas en el cuadrado. Esta área también incorporará códigos de corrección de errores.
- **Posición:** este patrón sirve para detectar la posición del código. Como este patrón está colocado en tres esquinas del símbolo, la posición, el tamaño y el ángulo del mismo pueden ser detectados. Este patrón consiste en una estructura que puede ser detectada en todas las direcciones.
- **Alineamiento:** este patrón sirve para corregir la distorsión del código. Es muy efectivo para corregir las distorsiones no lineales. Para corregir la distorsión, se comprueba la coordenada central de este patrón. Para este propósito, se coloca una celda negra en el centro para que sea más fácil detectar la coordenada central.
- **Sincronización:** este patrón sirve para identificar la coordenada central de cada celda en el código con cuadros negros y blancos colocados alternativamente. Se usa para corregir la coordenada central de las celdas de datos cuando el símbolo está distorsionado o cuando hay un error en el tono de la celda. Se coloca tanto vertical como horizontalmente.

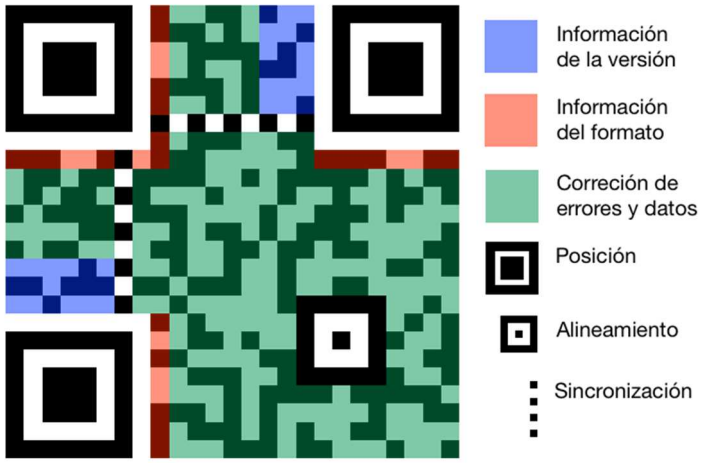


Figura 2. Estructura de un código QR
Fuente: http://www.rociomiro.es/crea-y-personaliza-tus-codigos-qr/

1.6.4 Comunicación entre aplicación y servidor

La comunicación entre aplicación y servidor se realiza mediante servicios web (en inglés, *web services*), de forma que se establezcan las conexiones cuando sean necesarias (se establece conexión TCP, se envía un mensaje HTTP con una petición,

se recibe la respuesta HTTP con el resultado de la operación solicitada, se cierra conexión TCP).

Para poder utilizar dichos servicios web se ha utilizado una librería cliente SOAP para la plataforma Android, conocida como *ksoap2* [3]. Esta librería *ksoap2* es un API SOAP basado en kXML, que es un analizador sintáctico (parser) XML ligero basado en Java, diseñado para ejecutarse en sistemas embebidos limitados tales como dispositivos móviles.

En cuanto a la forma de crear el servicio web en el servidor, se ha utilizado el asistente del entorno de desarrollo, es decir, no ha sido necesario hacer uso de ninguna librería.

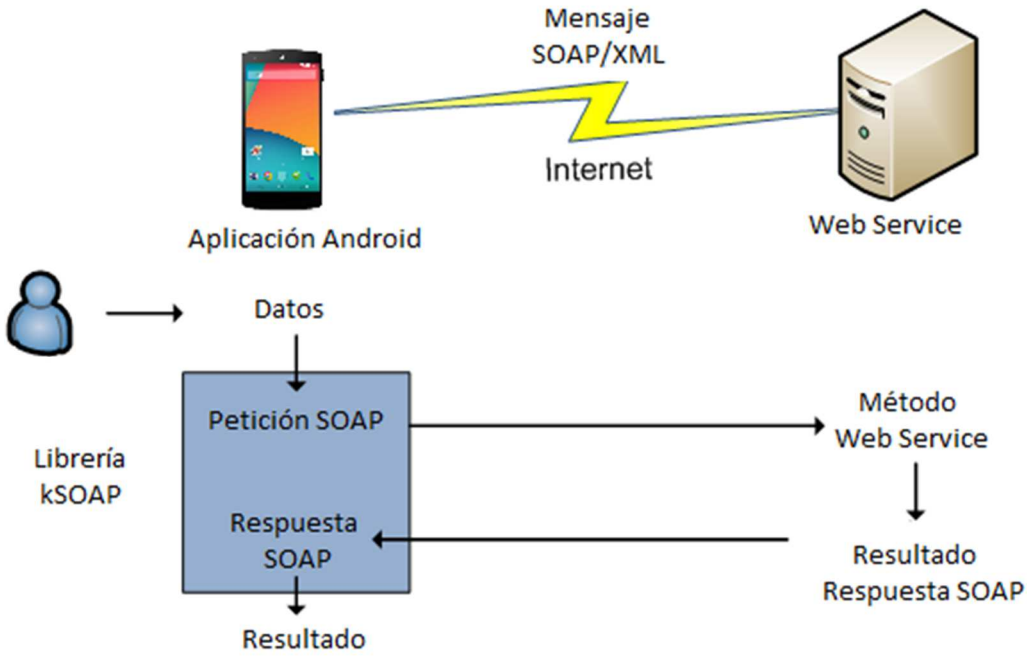


Figura 3. Arquitectura comunicación aplicación-servidor

1.6.5 Seguridad en las comunicaciones

Para proporcionar seguridad a las comunicaciones entre aplicación y servidor, se hace uso de varias tecnologías de cifrado y resumen. En función del tipo de mensaje que se utilice se puede utilizar un nivel más alto que en otro tipo de mensajes. La forma concreta en que se aplican según el tipo de mensaje se verá en el apartado ‘1.7.4 Protocolo de comunicaciones’.

- RSA [4]: utilizando el esquema de cifrado asimétrico, con clave pública y privada, se garantiza que el mensaje sólo puede procesarlo el destinatario del mensaje, (se cifra con la clave pública y se descifra con la clave privada).

Debido a problemas con la API de Android, muchas funciones no están totalmente soportadas por lo que era imposible realizar este tipo de cifrado. Para solucionarlo se utiliza una librería adicional que si permite realizar este tipo de cifrado, se conoce como *SpongyCastle* [5]. Para evitar posibles incompatibilidades, esta librería se utiliza tanto en la aplicación móvil como en el servidor.

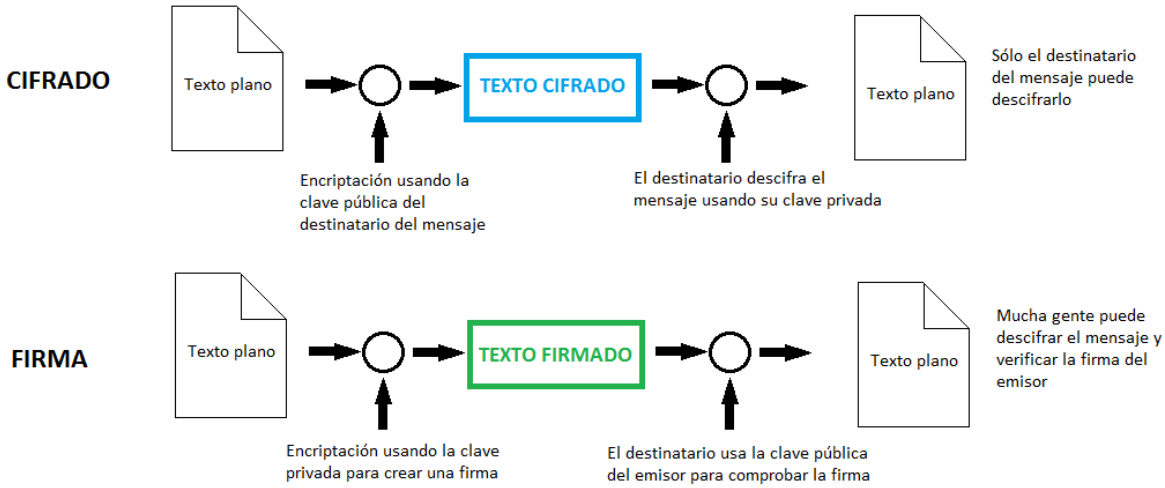


Figura 4. Esquema RSA (cifrado y firma)

Como método de operación de cifrado de bloque se utiliza ECB (Electronic CodeBook), de forma que el mensaje se divide en bloques y se encripta por separado. Es la forma más fácil de operación, no necesita vector de inicialización, y su principal desventaja es que siempre la misma entrada va a tener la misma salida.

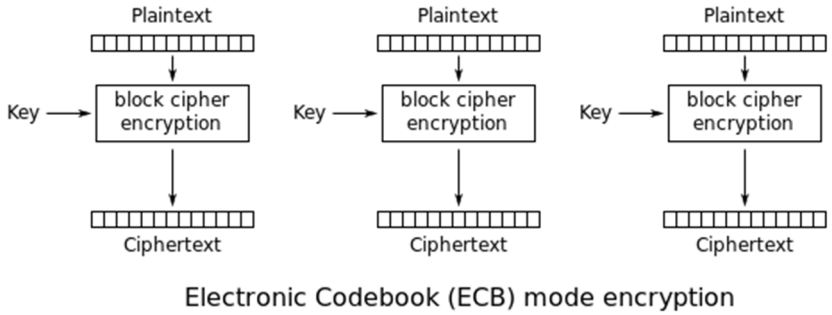


Figura 5. Esquema cifrado ECB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

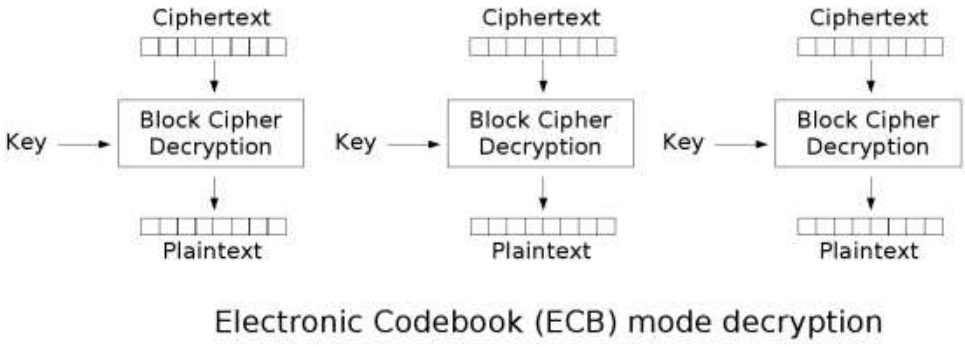


Figura 6. Esquema descifrado ECB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Obligatoriamente al utilizar el algoritmo RSA, hay que utilizar un estándar de *padding* (relleno), ya que si no se utiliza no se garantiza total seguridad. Se utiliza PKCS1, el primero de la familia de estándares conocida como *Public-Key Cryptography Standards*. Define las propiedades matemáticas de las claves públicas y privadas, operaciones primitivas para encriptar y firmar, esquemas seguros de criptografía, etc.

Este algoritmo se utiliza para garantizar que el servidor TutoGuia es el único que puede procesar la información codificada.

- AES (Advanced Encryption Standard) [6]: se garantiza la confidencialidad, cifrando los datos enviados/recibidos con una clave que comparten los usuarios que intervienen en la comunicación. El modo de operación utilizado es CFB (Cipher FeedBack), el cual utiliza un vector de inicialización para 'aleatorizar' la salida del cifrador. En este caso no se utiliza relleno.

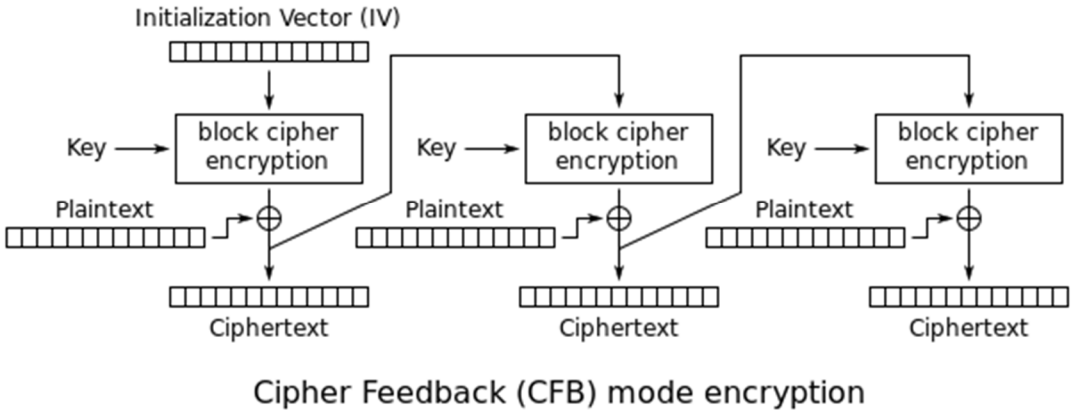


Figura 7. Esquema modo de cifrado CFB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

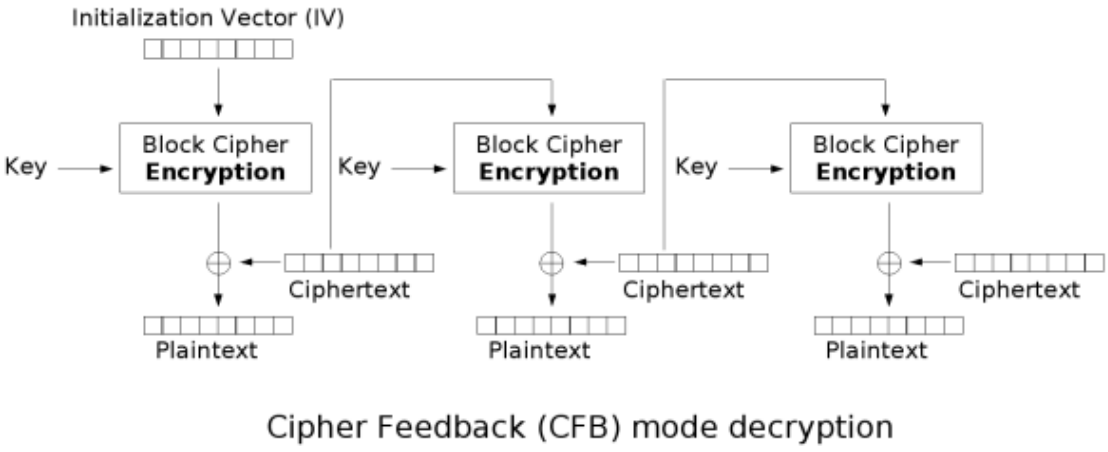


Figura 8. Esquema modo de descifrado CFB
Fuente: http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

- SHA-1 [7]: se garantiza la integridad de los datos envidados/recibidos, calculando una función resumen en función del mensaje. Se utiliza para garantizar que la información intercambiada no ha sufrido ningún cambio.

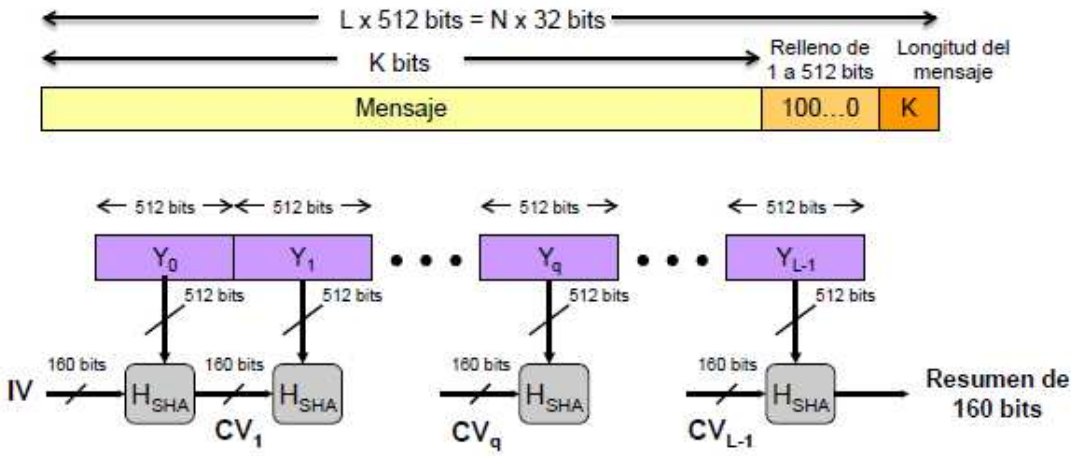


Figura 9. Esquema SHA-1
Fuente: http://en.wikipedia.org/wiki/Hash_function

1.6.6 Gestor de base de datos MySQL

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS), propiedad de Oracle, para bases de datos relacionales.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo,

permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java, y su integración en distintos sistemas operativos.

También es muy destacable, la condición de Open Source de MySQL, que hace que su utilización sea gratuita para uso no comercial. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

1.6.7 Gestor de base de datos SQLite

SQLite es una base de datos Open Source, muy popular en muchos dispositivos pequeños, como Android.

Las ventajas que presenta utilizar SQLite es que no requiere configuración, no tiene un servidor de base de datos ejecutándose en un proceso separado y es relativamente simple su empleo.

Esta base de datos se utiliza para guardar los datos de profesores en la aplicación móvil.

1.7 Descripción del sistema

El sistema desarrollado consta de dos partes: la aplicación para terminales Android, encargada de la recepción de notificaciones y captura de información a través de códigos QR y NFC, y el servidor, que proporciona el apoyo al sistema sirviendo de plataforma de gestión y almacenamiento de la información.

1.7.1 Arquitectura del sistema

La infraestructura necesaria para el despliegue del sistema consta de los siguientes elementos:

- Servidor: desarrollado en Java, es el centro del sistema, ya que recibe las actualizaciones de horarios e inserciones de avisos, y es el encargado de enviar los mensajes necesarios para notificar dichas actualizaciones e inserciones.
- Base de datos: utilizando el sistema gestor MySQL, se almacena toda la información necesaria relacionada con cuentas de usuarios (datos personales, contacto, contraseñas), datos de horarios, datos de avisos y datos de alumnos que desean ser notificados. En el apartado '1.7.2 Base de datos' se detallan las tablas utilizadas.
- Conexión a Internet: para comunicación con dispositivos móviles. Se pretende que el sistema pueda utilizarse desde cualquier parte, aprovechando la posibilidad de acceder a Internet de los teléfonos móviles. Por ello, el servidor debe poder ser accesible desde cualquier parte de Internet, es decir, poseer una dirección IP pública. Además es necesario el acceso a Internet para hacer uso del sistema de mensajería.
- Sistema de mensajería: servidor Google Cloud Messaging. Se utiliza para enviar a la aplicación móvil de los alumnos las notificaciones cuando alguno de los profesores que "siguen" realice algún cambio. En el apartado '1.7.3 Notificaciones' se detalla el funcionamiento de este sistema.

En la Figura 10 puede verse el esquema del sistema:

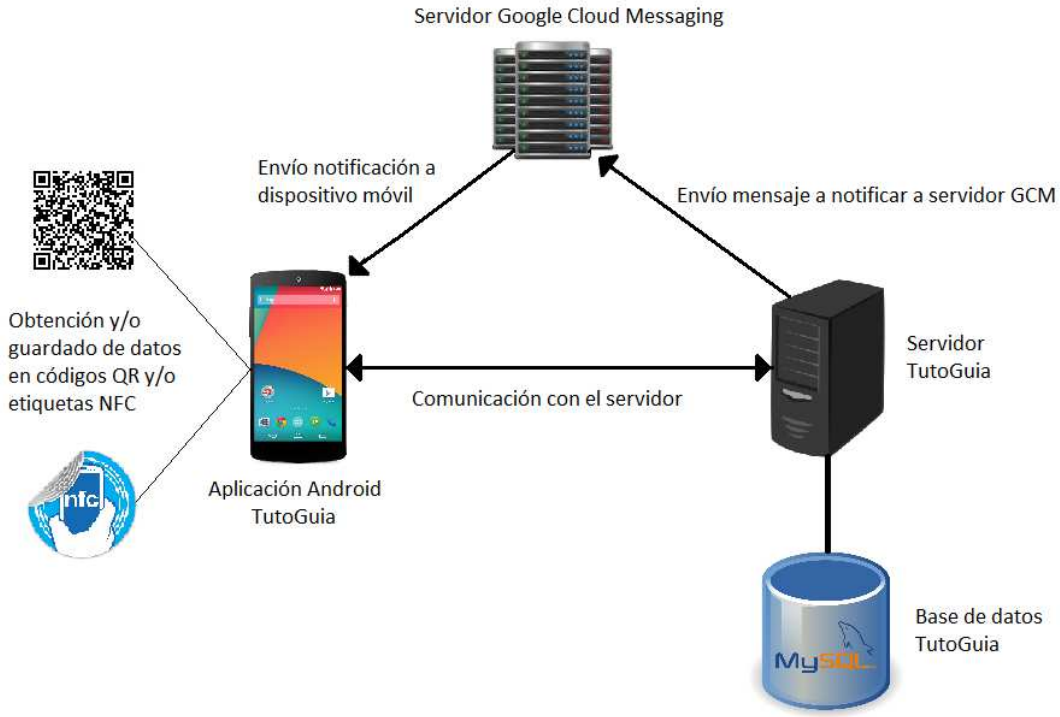


Figura 10. Arquitectura del sistema

1.7.2 Base de datos

La base de datos MySQL utilizada por el servidor consta de seis tablas:

- Usuarios: almacena los datos personales de los profesores.
- Tutorías: almacena el horario de tutorías de cada profesor y la última fecha de actualización.
- Avisos: almacena los avisos que publiquen los profesores con su fecha de inicio, fecha de fin y descripciones.
- Ult_acceso: almacena el último número de secuencia utilizado de cada usuario.
- Vuelvodiezmin: si se activa el modo 'Vuelvo en 10 minutos', la variable opc se pondrá a '1', lo cual significa que el modo está activado.
- Notificar: almacena a qué alumnos (identificados con el google_id de su terminal) se les enviará notificaciones utilizando Google Cloud Messaging.

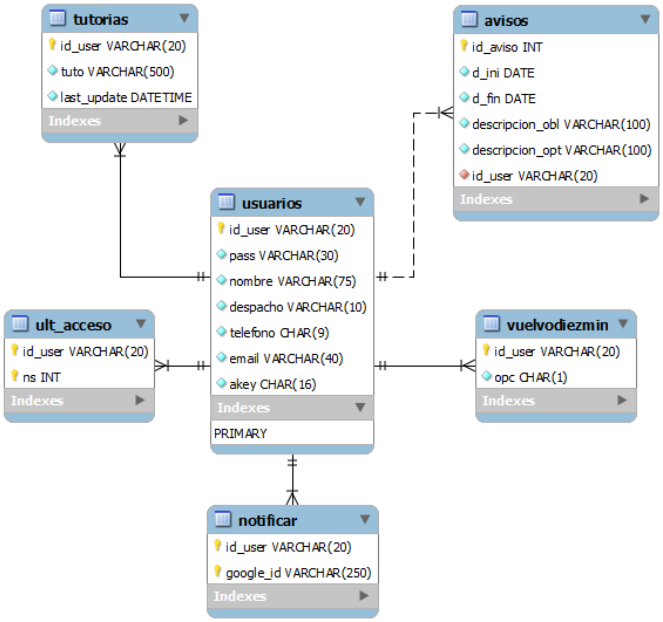


Figura 11. Esquema de tablas y variables de la Base de Datos

1.7.3 Notificaciones

Cuando un profesor haga algún cambio en su horario de tutorías o inserte un aviso, se enviará una notificación a aquellos alumnos que se hayan registrado. Para dicha tarea se hace uso de un servicio: Google Cloud Messaging (GCM).

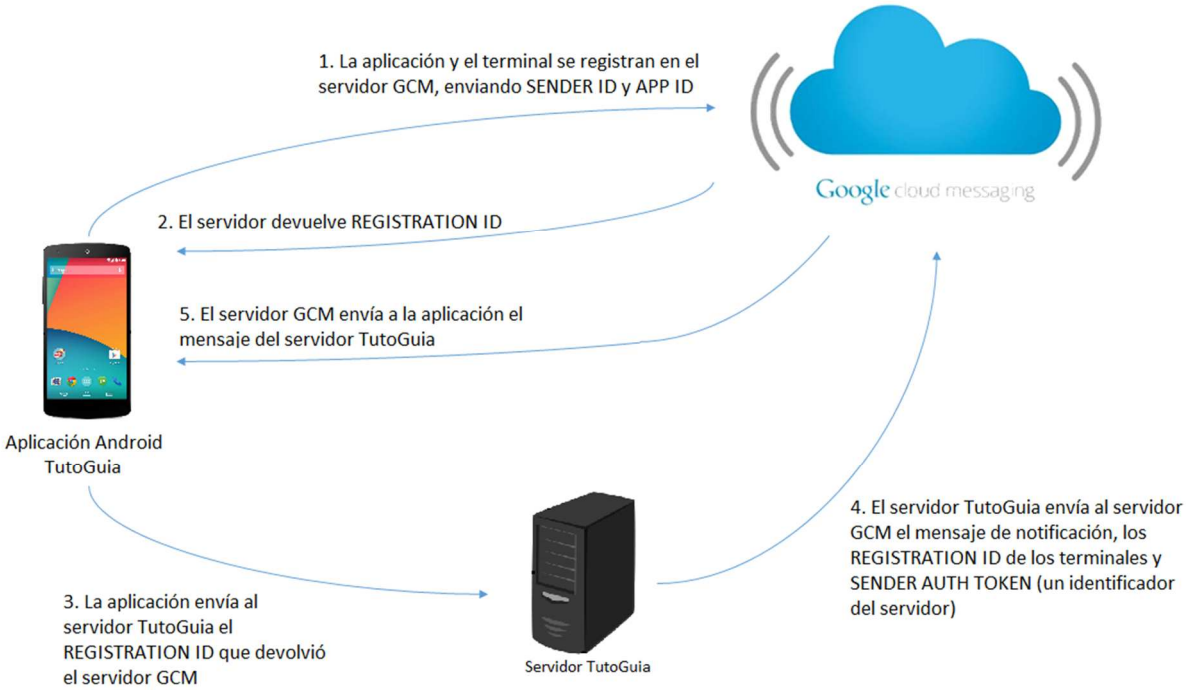


Figura 12. Proceso registro y envío de notificaciones

1. La aplicación se registra en los servidores de GCM enviando los valores 'Sender ID' (proporcionado por Google al activar el API) y 'App ID' (nombre del paquete de la aplicación).
2. Si el registro es satisfactorio se devuelve a la aplicación un 'Registration ID' (identificador único del dispositivo móvil).
3. La aplicación envía al servidor el 'Registration ID' (si el estudiante activa las notificaciones).
4. Cuando haya algún cambio en tutorías o avisos, el servidor TutoGuia envía al servidor GCM el mensaje a notificar, los 'Registration ID' de los móviles a los que va a enviar el mensaje y el identificador de 'Sender', conocida como 'API Key'. El servidor GCM mantiene el mensaje en cola hasta que el terminal pueda recibirlo.
5. La notificación es enviada a cada dispositivo.

En el Anexo III se explica con más detalle los pasos necesarios para desplegar este servicio.

1.7.4 Protocolo de comunicaciones

Para garantizar una comunicación óptima y segura a través del servicio web, se ha diseñado un protocolo para el intercambio de información entre aplicación y servidor. Este protocolo está basado en las tecnologías de seguridad ya mencionadas: RSA, AES y SHA1.

Dependiendo de la consulta o acción a realizar, se consideran tipos de mensajes distintos, y por tanto van a tener un nivel de seguridad distinto:

- Registro de profesores: el registro sólo es obligatorio para los profesores, los estudiantes pueden utilizar la aplicación sin identificarse. Dentro de este tipo de mensaje hay dos subtipos: registro de un usuario nuevo (no hay datos anteriores en la base de datos) y registro de un usuario existente (el usuario ya está registrado en la base de datos). El segundo subtipo se utiliza para aquellos casos en que el usuario cambie de teléfono o desinstale la aplicación o algún caso similar.

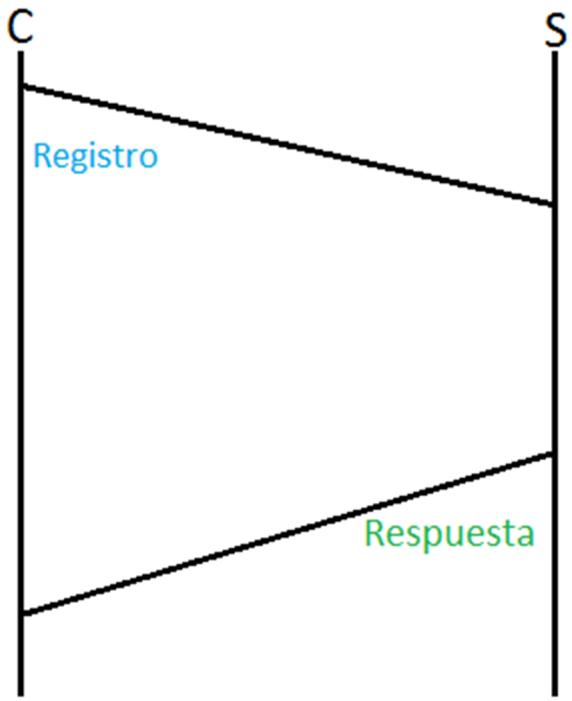


Figura 13. Diagrama de mensajes 'Registro de usuarios'

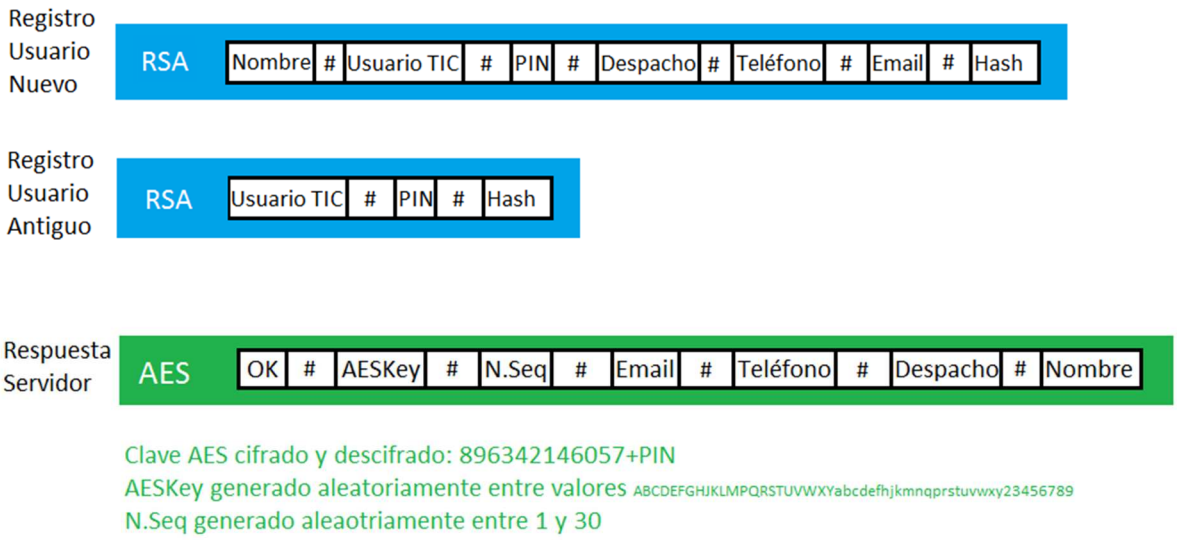


Figura 14. Formato mensajes 'Registro de usuarios'

La aplicación envía el mensaje de registro cifrado con RSA utilizando la clave pública del servidor. El servidor descifra con su clave privada y obtiene los datos del mensaje. Sea del subtipo que sea la respuesta tendrá el mismo formato, enviando datos necesarios para el segundo subtipo (nombre, teléfono, despacho, email) así evitamos al usuario tener que volver a introducirlos.

La respuesta irá codificada con AES utilizando como clave la siguiente cadena: 896342146057+PIN (PIN facilitado por el usuario en el mensaje de registro). La clave AES debe ser de 16 caracteres.

Una vez completado el proceso, el cliente utilizara como clave AES en sucesivas operaciones, el valor contenido el campo AESKey de la respuesta. Asimismo deberá utilizar e incrementar cuando sea necesario el número de secuencia inicial proporcionado en el campo N.Seq.

En caso de error en el proceso, se devuelve un mensaje 'ER'.

- Gestiones de profesores: en este tipo de mensajes se proporciona el nivel de seguridad más alto posible, ya que es donde los profesores insertan en la base de datos los cambios en su horario de tutorías, insertan avisos y activan (o desactivan) el modo 'Vuelvo en 10 minutos'. En total hay 4 subtipos.

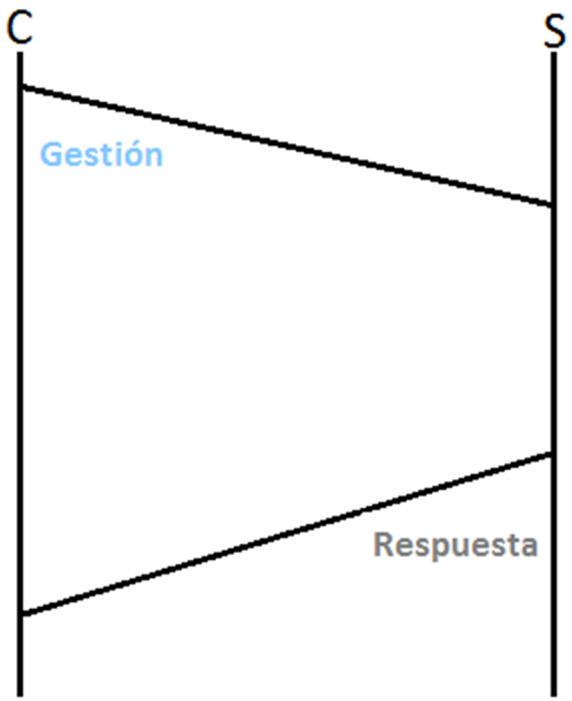


Figura 15. Diagrama de mensajes 'Gestiones de usuarios'

Figura 16. Formato mensajes 'Gestiones de usuarios'

En función del valor del campo 'OPC', el 'Mensaje' se procesará de una forma u otra, asimismo la respuesta puede ser distinta, ya que en el subtipo 'Insertar aviso' la respuesta cuando el proceso es satisfactorio será el ID que se haya asignado a ese aviso (utilizado para futuras modificaciones o borrados de la base de datos), mientras que en los subtipos restantes, el mensaje enviado será un 'OK'.

- Consultas de alumnos: en este tipo de mensajes no se proporciona seguridad alguna, ya que los datos que se manejan no presentan riesgo alguno, son datos públicos de horarios de tutorías y avisos.

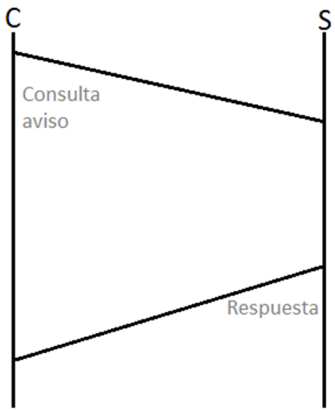


Figura 17. Diagrama de mensajes 'Consultar avisos'

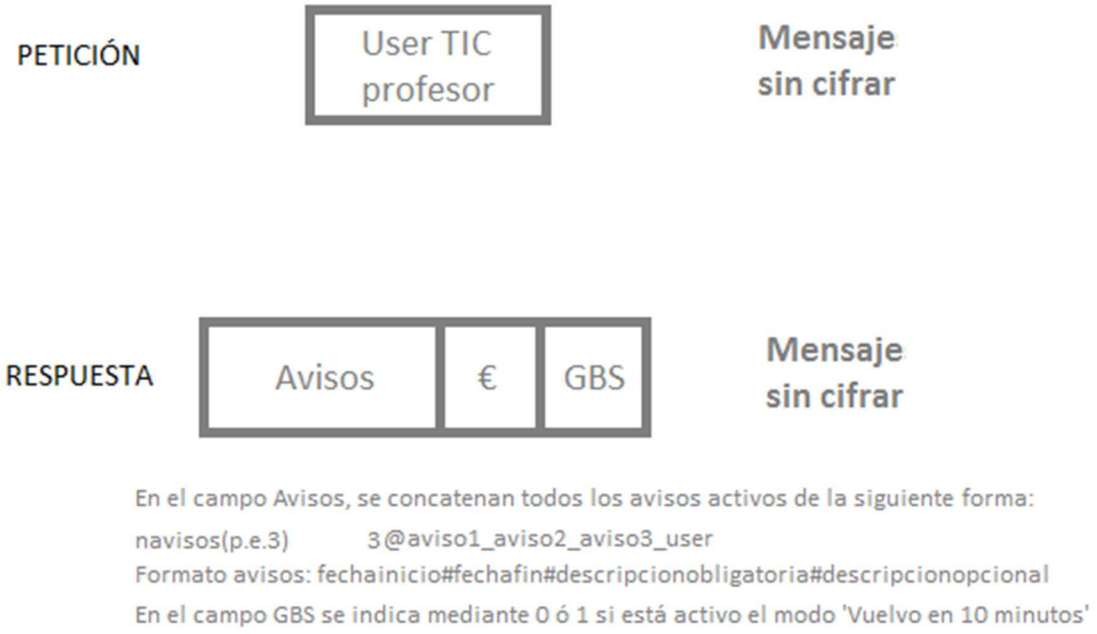


Figura 18. Formato mensajes 'Consultar avisos'

Para consultar los avisos, se envía sólo el usuario TIC del profesor a realizar la consulta y se recibe una cadena con los avisos activos (si los hubiera) y si el profesor está ausente momentáneamente.

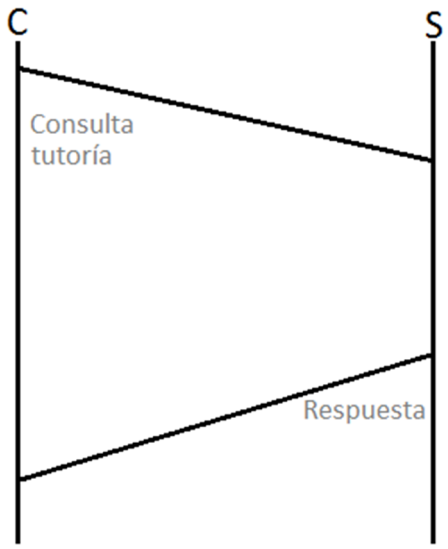


Figura 19. Diagrama mensajes 'Consultar tutorías'

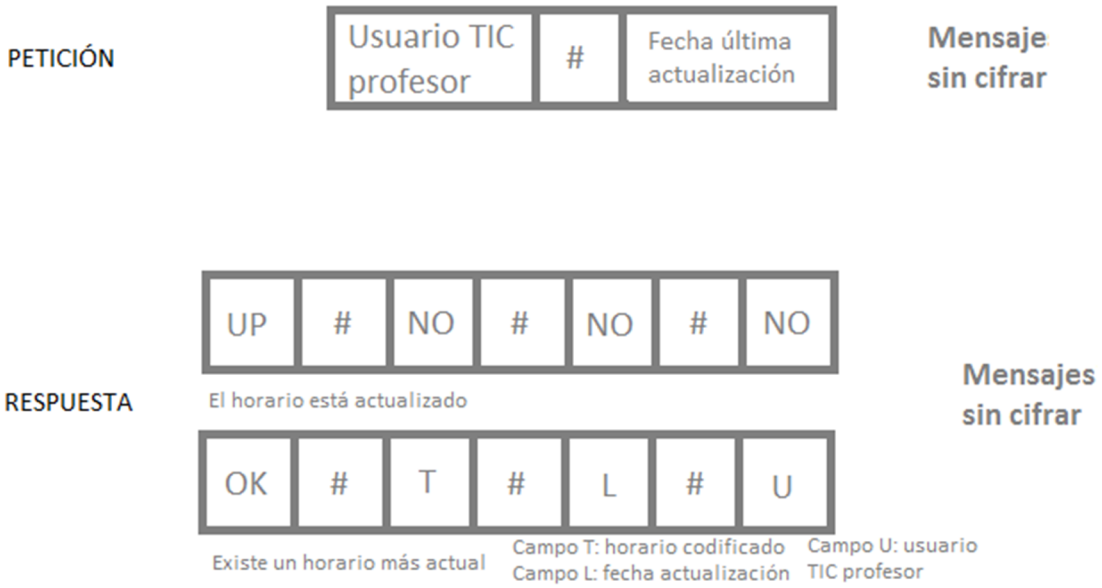


Figura 20. Formato mensajes 'Consultar tutorías'

Dependiendo de si hay un horario más actual o no, se envía una respuesta u otra, pero deben tener el mismo formato, ya que si no lo tienen se produciría un error al separar las cadenas de texto con el símbolo '#'.

La forma en que se codifican los horarios de tutorías se detalla en el apartado '1.7.4.1 Codificación de horario de tutorías'.

- Registro notificaciones: con este tipo de mensaje, los alumnos activan o desactivan el envío de notificaciones ante cualquier actualización en el horario de tutorías o inserción de avisos.

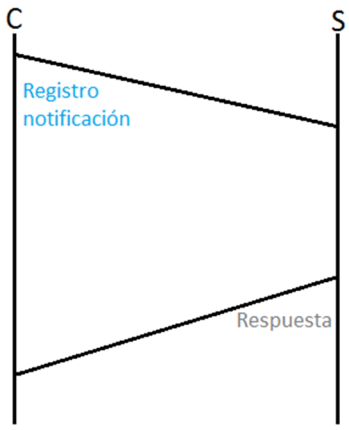


Figura 21. Diagrama mensajes 'Registro notificaciones'

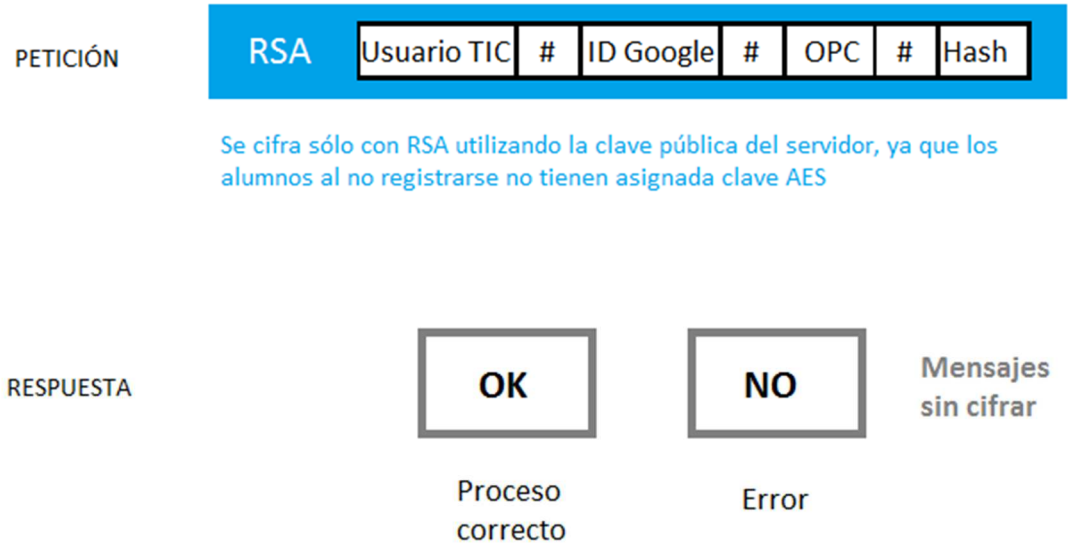


Figura 22. Formato mensajes 'Registro notificaciones'

En el campo 'Usuario TIC' se indica el profesor del que se quiere activar o desactivar las notificaciones.

En el campo 'ID Google' se indica el código de identificación del dispositivo móvil proporcionado por Google.

En el campo 'OPC' se indica la acción a realizar, activar o desactivar las notificaciones.

1.7.4.1 Codificación de horario de tutorías

Considerando 12 tramos horarios de 1 hora, de 8:30 a 14:30 y 15:30 a 21:30, 5 días a la semana, obtenemos una cuadrícula como la de la Figura 23.

Tan sólo basta con marcar con 1 los tramos en los que hay tutoría y con 0 los tramos en los que no hay tutoría, y de esta forma obtenemos una cadena de 1 y 0, es decir, un byte que puede convertirse a hexadecimal para ocupar menos espacio, exactamente 2 caracteres.

Un horario de tutorías codificado puede ser éste: 'f0:e0:e0:e0:e0:e0:e0:e0:e0:e0:e0:e0', en el que sólo hay tutorías el lunes en el primer tramo horario (f0 = 111 10000).

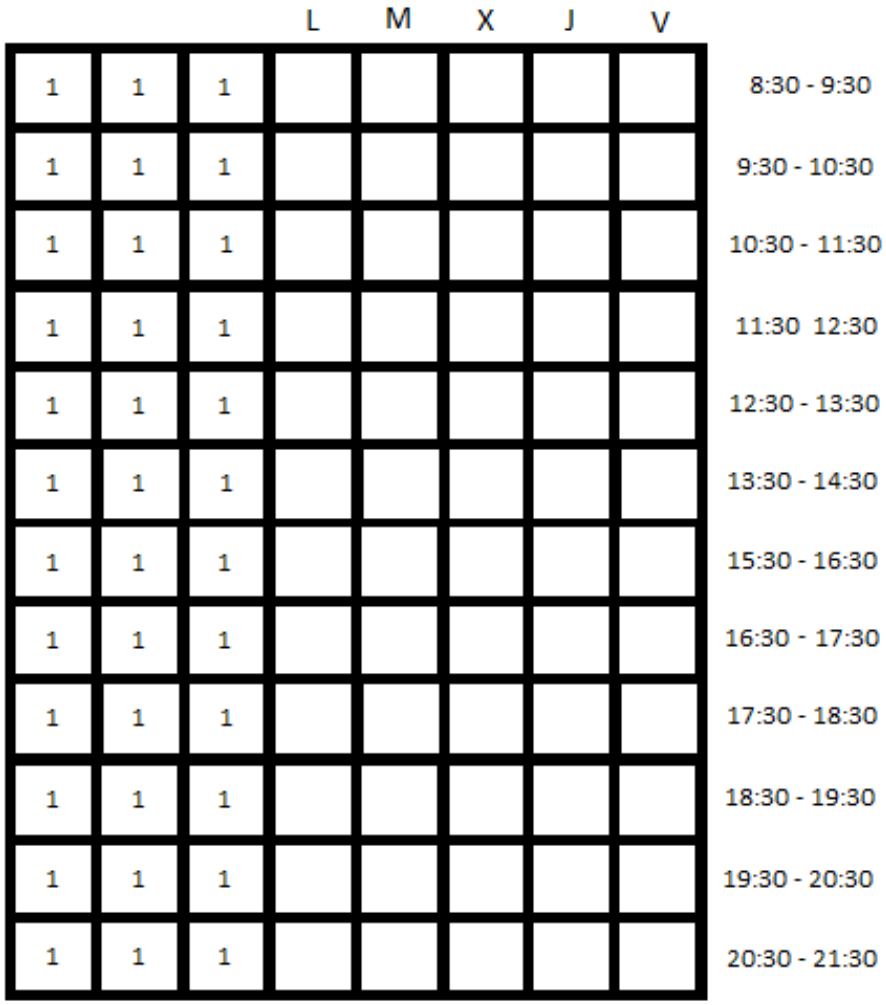


Figura 23. Cuadrícula horario de tutorías

1.7.4.2 Formato de la información en códigos QR y etiquetas NFC

La forma en que la información se guarda en los códigos QR y en las etiquetas NFC es la misma y tiene el siguiente formato:

`Nombre & UserTIC & Tutorías & Email & Teléfono & Despacho & FechaActual`

En el campo `Tutorías` se inserta el horario de tutorías codificado como se ha explicado en el anterior apartado.

En el campo `FechaActual` se inserta la fecha en que se crea el código QR o la etiqueta NFC. Sirve para posteriormente comparar si existe o no un horario más actual. El formato de la fecha es el siguiente: 'yyyy-MM-dd HH:mm:ss'

1.7.5 Descripción aplicación móvil

La aplicación está diseñada de forma que su uso sea fácil e intuitivo, para ello se basa, principalmente, en dos elementos:

- Navigation Drawer [8]: es un panel que aparece desde la parte izquierda de la pantalla y muestra las opciones de navegación de la aplicación. Este panel puede desplegarse de dos formas, bien pulsando el icono de la aplicación situado en la parte superior izquierda, o bien deslizando el dedo de izquierda a derecha. Una vez abierto, se selecciona la opción deseada y se mostrará en pantalla.

Para utilizar este elemento hay que hacer uso de la librería de soporte de Android 'android.support.v4'.

En la siguiente figura se muestra un ejemplo de Navigation Drawer. En la primera imagen se muestra una pantalla principal de la aplicación, indicando las formas de desplegar el panel. En la segunda imagen se muestra el panel desplegado con las distintas opciones. En la tercera imagen se muestra la segunda pantalla abierta tras pulsar el correspondiente botón del panel.

Al utilizar este elemento se asegura una navegación rápida y fácil por la aplicación.

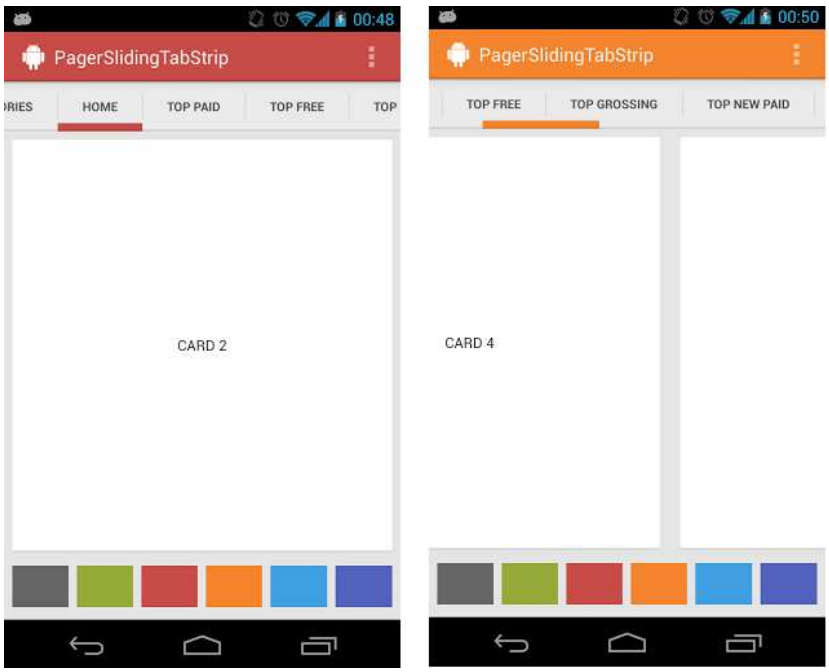
a b c

Figura 24. Ejemplo 'Navigation Drawer'. Pantalla principal indicando formas de desplegar el panel a), panel desplegado b), Pantalla secundaria c)
Fuente: http://developer.android.com/intl/es/design/patterns/navigation-drawer.html

- PagerSlidingTabStrip [9]: este elemento permite navegar entre pestañas de forma fácil y rápida, simplemente deslizando el dedo por la pantalla. De esta forma la navegación entre los distintos días de la semana sea lo más intuitiva posible.

Para utilizar este elemento hay que hacer uso de una librería externa 'com.astuetz.pagerslidingtabstrip'.

En las imágenes inferiores se muestra un ejemplo de PagerSlidingTabStrip. La transición entre las pantallas se puede hacer mediante las etiquetas superiores o deslizando las pantallas.



a b

Figura 25. Ejemplo 'PagerSlidingTabStrip' a) y b)
Fuente: https://github.com/astuetz/PagerSlidingTabStrip

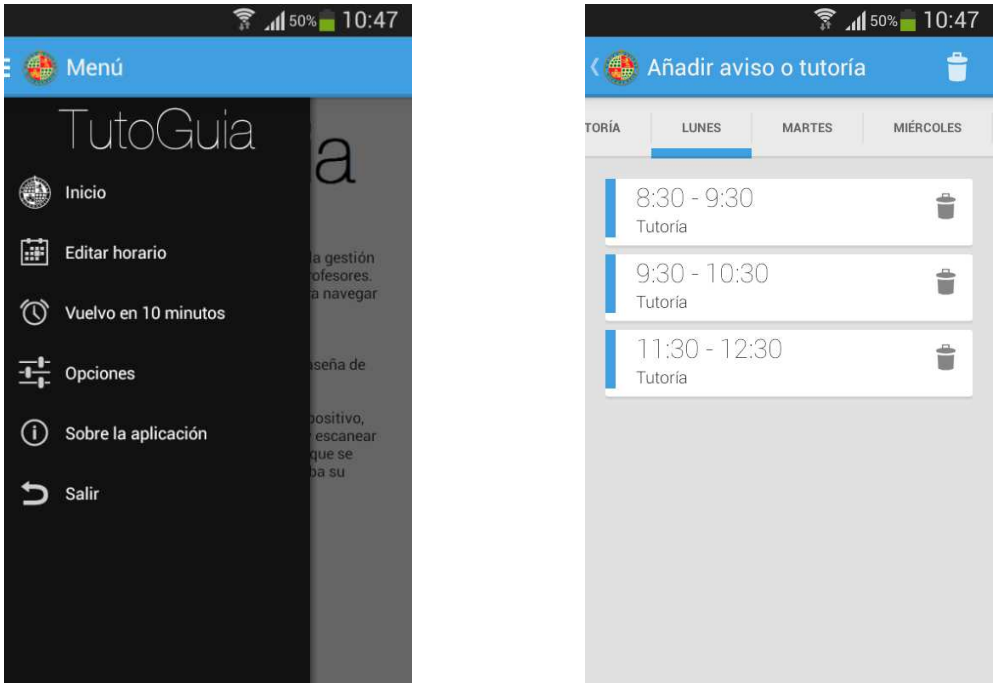
- Cards UI [10]: utilizando una librería adicional (*com.fima.cardsui*), se puede hacer uso del diseño de 'tarjetas' similar a las aplicaciones Google Now y Google Play. Se utiliza para mostrar los avisos y horarios de tutorías.
- Action Bar [11]: es una característica que identifica la localización del usuario en la aplicación y añade acciones de usuario y modos de navegación. Ofrece una interfaz familiar de la aplicación en distintos tipos de pantalla, ya que automáticamente el sistema se adapta al tipo de pantalla. Para usar esta característica hay que hacer uso de las librerías de soporte de Android.

Figura 26. Action Bar que incluye [1] el icono de la aplicación, [2] dos botones de acción de usuario y [3] un botón para desplegar un menú flotante
Fuente: http://developer.android.com/intl/es/guide/topics/ui/actionbar.html

En las figuras inferiores se muestra el resultado de aplicar los elementos mencionados.

En la imagen izquierda se muestra el panel desplegado con las distintas opciones de navegación (cambia según el tipo de usuario de la aplicación: profesor o estudiante).

En la imagen derecha se muestran las horas de tutorías por días de la semana, pudiendo navegar por los días con el elemento PagerSlidingTabStrip.



a) b)

Figura 27. Resultado 'DrawerLayout' y 'ActionBar' a) y 'PagerSlidingTabStrip', 'ActionBar' y 'CardUI' b)

Para continuar con la descripción de la aplicación se va a explicar los paquetes y clases más importantes del proyecto:

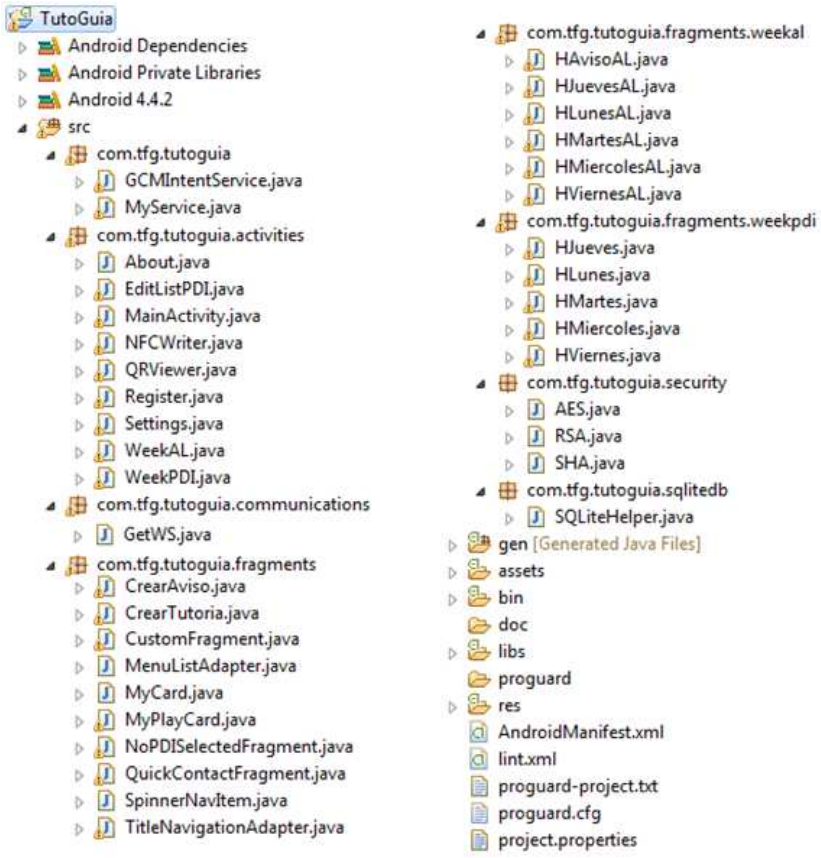


Figura 28. Esquema proyecto aplicación móvil

Paquete com.tfg.tutoguia: en este paquete se encuentran las clases con los servicios que se ejecutan en la aplicación.

- GCMIntentService: es un servicio encargado de recoger y procesar los mensajes enviados por el servidor de Google Cloud Messaging (GCM), para ello consta de 4 métodos: onError (procesa los mensajes de error), onMessage (procesa los mensajes y se crea la notificación que se mostrará al usuario), onRegistered (procesa el mensaje que devuelve el servidor de GCM cuando el terminal se registra en el servidor, devuelve un identificador único de terminal), onUnregistered (procesa el mensaje que devuelve el servidor de GCM cuando el terminal se borra del servidor).
- MyService: es un servicio encargado de avisar (se muestra una notificación) al profesor cuando activa el modo 'Vuelvo en 10 minutos' de desactivarlo pasados 15 minutos.

Paquete com.tfg.tutoguia.activities: en este paquete se encuentran las clases con todas las actividades de la aplicación.

- MainActivity: es la actividad principal de la aplicación y por tanto la encargada de varias funciones iniciales (lanzar la llamada para registrar el terminal en los servidores de GCM, mostrar la visualización de aplicación para estudiante o profesor, procesar los resultados de la captura de código QR y etiqueta NFC, etc.).
- NFCWriter: es la actividad encargada de escribir la información del profesor y horario de tutorías en una etiqueta NFC.
- QRViewer: es la actividad encargada de generar el código QR con la información del profesor y horario de tutorías y mostrarlo en pantalla. Además permite guardar el código QR en la memoria del terminal y compartir el código QR por correo electrónico o mensajería instantánea.
- Register: es la actividad encargada de recoger los datos del profesor para realizar el registro en el servidor. Según la opción elegida, el profesor puede realizar un registro nuevo (debe introducir todos los datos) o realizar un registro antiguo (sólo introduce usuario y PIN).
- Settings: es la actividad encargada de mostrar las opciones y configuraciones de la aplicación y guardarlas por medio de SharedPreferences [12].
- WeekAL: muestra el elemento PagerSlidingTabStrip con los días de la semana adecuados a la visualización de tutorías por parte de los alumnos.
- WeekPDI: muestra el elemento PagerSlidingTabStrip con las opciones para crear avisos, tutorías y ver los días de la semana en que hay tutorías creadas.

En la Figura 29 se muestra un diagrama de navegación entre las actividades.

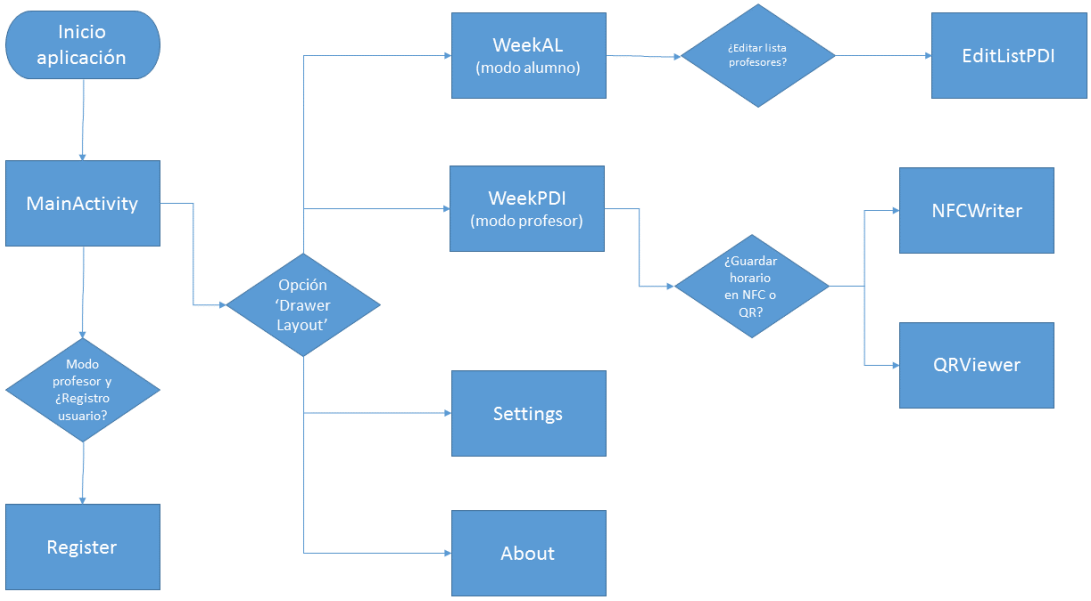


Figura 29. Diagrama de actividades

Paquete com.tfg.tutoguia.communications: contiene la clase encargada de la comunicación con el servidor mediante el servicio web.

- GetWS: contiene los métodos necesarios para realizar la comunicación con el servidor de acuerdo al protocolo diseñado. En función del tipo de mensaje a enviar, se encadenan y se cifran de una forma u otra. Una vez obtenida la cadena de caracteres que se enviará, se usa una tarea asíncrona, *AsyncTask* [13], de forma que la aplicación no se 'congele[1] mientras se produce la comunicación. Mientras se produce la comunicación se muestra un mensaje indicando al usuario que espere. Una vez que se obtiene la respuesta se pasa a la actividad perteneciente para que la procese.

Paquete com.tfg.tutoguia.fragments: contiene las clases necesarias para crear los distintos fragmentos en los que se compone el diseño de la aplicación.

Paquete com.tfg.tutoguia.security: en este paquete están las clases de seguridad utilizadas en el protocolo.

- AES: contiene los métodos necesarios para cifrar y descifrar un texto utilizando una clave AES.

[1] La aplicación se bloquea y no responde a acciones del usuario mientras se realiza la acción, lo que provoca que el usuario crea que la aplicación ha dejado de funcionar y la cierre, provocando una pérdida de información y posible desincronización con el servidor con los números de secuencia.

- RSA: contiene los métodos necesarios para cifrar un texto utilizando la clave pública del servidor. La clave pública se guarda en la carpeta *res/raw*.
- SHA: contiene los métodos necesarios para calcular la función resumen de una cadena de texto.

Paquete com.tfg.tutoguia.sqlitedb:

- SQLiteHelper: se crea la base de datos, tablas y variables necesarias para almacenar la información de profesores y avisos.

Para más información, consultar la documentación *Javadoc* situada en el CD en la carpeta: DOCUMENTACIÓN/Javadoc APP/index.html

1.7.6 Descripción aplicación servidor

A diferencia de la aplicación, el servidor no necesita interfaz ninguna para su funcionamiento, simplemente ejecutarlo como se explica en el punto "2.1 Anexo I: Manual de administración del servidor".

La estructura de clases se presenta en la siguiente Figura:

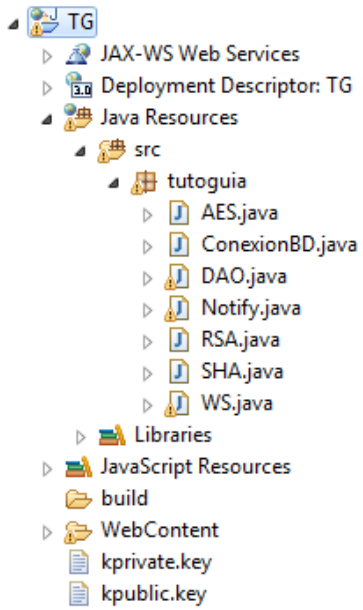


Figura 30. Esquema proyecto aplicación servidor

En este caso sólo existe un paquete en el que se agrupan las 8 clases necesarias para el funcionamiento del sistema.

La clase WS es el punto de partida, ya que es la clase encargada de recibir los mensajes de la aplicación móvil y la encargada de llamar a los distintos métodos de las demás clases para procesar el mensaje según corresponda.

- AES: contiene los métodos necesarios para cifrar y descifrar un texto utilizando una clave AES. Cuando se registra un usuario nuevo, se asigna la clave AES (*AESKey*) que deberá usar en las futuras comunicaciones. La clave generada debe ser de 16 caracteres para el correcto funcionamiento del sistema. En la siguiente Figura se muestra el método encargado de generar aleatoriamente las claves.

```
public String generaKey(){
    String key = null;
      String elegibleChars = "ABCDEFGHJKLMPQRSTUVWXYabcdefhjkmnpqrstuvwxy23456789";
      char[] chars = elegibleChars.toCharArray();

      StringBuffer finalString = new StringBuffer();

      for ( int i = 0; i < 16; i++ ) {
        double randomValue = Math.random();
        int randomIndex = (int) Math.round(randomValue * (chars.length - 1));
        char characterToShow = chars[randomIndex];
        finalString.append(characterToShow);
      }
      key = finalString.toString();

    return key;
}
```

Figura 31. Generación AESKey

- ConexionBD: contiene los métodos necesarios para acceder a la base de datos, tales como localización de la base de datos, driver que se utiliza para acceder (en este caso *Java Database Connectivity (JDBC*) y credenciales de acceso.
- DAO: contiene los métodos necesarios para realizar las consultas, inserciones y actualizaciones en la base de datos.
- Notify: contiene los métodos necesarios para enviar al servidor de GCM los mensajes para notificar a los alumnos cualquier cambio realizado por un profesor.

```
public Notify(String op, String user, String gid, String mensaje1, String mensaje2, String tipo){
    u = user;
    g = gid;
    m1 = mensaje1;
    m2 = mensaje2;
    t = tipo;

try {

    DAO dao = new DAO();

    Sender sender = new Sender("AIzaSyAPpbRJVuZDyoO75AudOKtnTiOcSyEy---");
    ArrayList<String> devicesList = null;
    MulticastResult mresult = null;

    //El mensaje se envía a muchos usuarios
    if(op.equals("1")==true){
        devicesList = dao.getGIDPDI(u);

        Message message = new Message.Builder()
        .collapseKey("1")
        .timeToLive(3)
        .delayWhileIdle(true)
        .addData("pdi",u)
        .addData("m1",m1)
        .addData("m2",m2)
        .addData("t", t)
        .build();


        mresult = sender.send(message, devicesList, 1);
        sender.send(message, devicesList, 1);
    }
```

Figura 32. Captura método envío mensaje

- RSA: contiene los métodos necesarios para descifrar un texto utilizando la clave privada del servidor. En la Figura 33 se muestran los métodos públicos que se llaman para cifrar y descifrar. Los parámetros que se pasan son los datos a cifrar o descifrar y la localización de la clave con la que se realizará el proceso.

```
public class RSA {

    /**
     * Método que será llamado para encriptar una cadena con RSA
     * @param data Cadena de datos a encriptar
     * @param fileLoc Localización del archivo a utilizar (clave pública o privada)
     * @return Devuelve cadena encriptada
     */
    public String encrypt(String data, String fileLoc) {
        byte[] enc = rsaEncrypt(data.getBytes(), fileLoc);
        return toHex(enc);
    }

    /**
     * Método que será llamado para desencriptar una cadena con RSA
     * @param data Datos a desencriptar
     * @param fileLoc Localización del archivo a utilizar (clave pública o privada)
     * @return Devuelve cadena desencriptada
     */
    public String decrypt(String data, String fileLoc){
        byte[] enc = toByte(data);
        byte[] decryptedData = rsaDecrypt(enc, fileLoc);
        return new String(decryptedData);
    }
```

Figura 33. Métodos públicos RSA

- SHA: contiene los métodos necesarios para calcular la función resumen de una cadena de texto.

```
public class SHA {

    private static byte[] genHash(String frase) {
          try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            md.update(frase.getBytes());
            return md.digest();
          } catch (NoSuchAlgorithmException e) {
            return null;
          }
        }

    private static String stringHexa(byte[] bytes) {
           StringBuilder s = new StringBuilder();
           for (int i = 0; i < bytes.length; i++) {
               int parteAlta = ((bytes[i] >> 4) & 0xf) << 4;
               int parteBaixa = bytes[i] & 0xf;
               if (parteAlta == 0) s.append('0');
               s.append(Integer.toHexString(parteAlta | parteBaixa));
           }
           return s.toString();
        }

    /**
     * Método que será llamado para la generación del hash
     * @param frase String al que se le calculará el hash
     * @return Devuelve el hash de 'frase'
     */
    public String hash(String frase){
        return stringHexa(genHash(frase));
    }

}
```

Figura 34. Métodos clase SHA

- WS: al igual que la clase GetWS de la aplicación móvil, esta clase se encarga de realizar la comunicación de acuerdo al protocolo. Contiene los métodos necesarios para procesar la información según el tipo de mensaje. En la Figura 35 se muestra el diagrama de flujo para la recepción de un mensaje de gestión de los profesores.

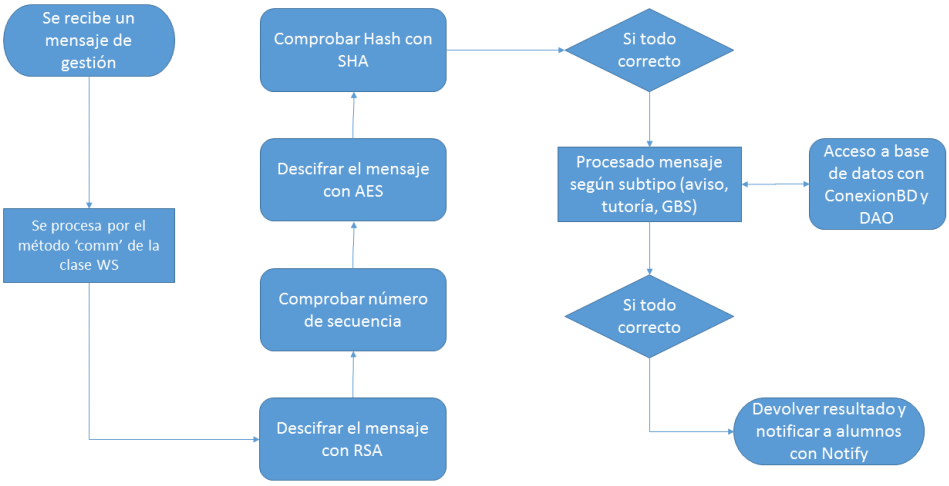


Figura 35. Diagrama de flujo mensaje de gestión recibido

Para más información, consultar la documentación *Javadoc* situada en el CD en la carpeta: DOCUMENTACIÓN/Javadoc Server/index.html

1.8 Conclusiones

Una vez terminado el desarrollo del Trabajo Fin de Grado, se llega a la conclusión de que se han cumplido todos los objetivos, principales y secundarios:

- Se ha desarrollado una aplicación cliente para Android que permite a los usuarios recibir información personalizada y de interés, en concreto se puede recibir información de tutorías y avisos de los profesores.
- La aplicación permite al usuario la personalización: permitiendo seleccionar los profesores de los que desea consultar la información y permitiendo ajustar la vibración del terminal ante la recepción de notificaciones.
- Por otro lado, se ha desarrollado una aplicación servidora en la que se apoya la aplicación cliente para almacenar información de profesores, alumnos, tutorías y avisos.
- Se ha diseñado una plantilla que pueden utilizar los profesores para publicar sus horarios de tutorías por medio de etiquetas NFC y códigos QR.
- La aplicación cliente puede adaptarse a diferentes tipos de terminal, ya sea en cuanto a versión de Android (compatibles desde versión 2.3) o en cuanto a tamaño de pantalla.

Además, se han añadido funciones adicionales que aportan gran valor añadido al proyecto como son:

- Servicio de notificaciones instantáneas con Google Cloud Messaging.
- Protocolo de comunicaciones seguro.

1.9 Líneas futuras

El sistema queda abierto para futuras mejoras o cambios que se puedan realizar, con el objetivo de mejorar lo presente.

En primer lugar, el servidor se tendría que subir a un dominio público para poder estar disponible al alcance de todos los usuarios. En el desarrollo del proyecto se ha utilizado un servidor “Apache Tomcat” que sólo ha funcionado en una red local sobre un PC normal. Sería ideal implementarlo en un servidor más potente, tanto en el aspecto software como hardware, de forma que pueda cubrir el servicio sin problemas. Además, habría que utilizar un motor de base de datos más potente, ya que se está usando “MySQL”, un motor gratuito y que seguramente no sea capaz de afrontar las demandas de un servicio que se presupone que será ampliamente utilizado.

En segundo lugar se puede implementar una opción de ‘Reserva de tutoría’, en la que el alumno solicita acudir a tutoría a cierta hora, y el profesor responde aceptando o denegando esta solicitud.

En tercer lugar, la base de datos podría estar integrada en IILAS, de forma que se utilizase las credenciales que se utilizan con el sistema SIDUJA y de forma que las tutorías y avisos pudiesen consultarse desde la plataforma de Docencia Virtual, por ejemplo.

En cuarto lugar se podría desarrollar la aplicación para el resto de sistemas operativos móviles, ya que aunque Android ocupa el 85% [14] del mercado, muchos alumnos utilizan otro tipo de sistemas operativos, tales como iOS y Windows Phone.

1.10 Referencias

[1] Proyecto para la gestión de tutorías en ámbito universitario UPV (última vez consultada Julio 2014):
http://www.gti-ia.upv.es/sma/tools/tourism/archivos/documentation/artutorias.pdf
[2] Librería QR Android (última vez consultada Julio 2014):
https://github.com/zxing/zxing

[3] Using ksoap2 for Android, and parsing output data (última vez consultada Julio 2014):
http://www.helloandroid.com/tutorials/using-ksoap2-android-and-parsing-output-data
[4] Rivest Shamir Adleman (RSA) (última vez consultada Julio 2014):
http://es.wikipedia.org/wiki/RSA
http://stackoverflow.com/questions/12471999/rsa-encryption-decryption-in-android
[5] SpongyCastle (última vez consultada Julio 2014):
http://rtyley.github.io/spongycastle/
[6] Advanced Encryption Standard (AES) (última vez consultada Julio 2014):
http://stackoverflow.com/questions/6788018/android-encryption-decryption-with-aes
http://iamvijayakumar.blogspot.com.es/2013/10/android-example-for-encrypt-and-decrypt.html
[7] Secure Hash Algorithm 1 (SHA-1) (última vez consultada Julio 2014):
http://es.wikipedia.org/wiki/Secure_Hash_Algorithm
[8] Navigation Drawer Android (última vez consultada Julio 2014):
http://developer.android.com/intl/es/design/patterns/navigation-drawer.html

[9] PagerSlidingTabStrip Android (última vez consultada Julio 2014):
https://github.com/astuetz/PagerSlidingTabStrip

[10] Cards UI Library for Android (última vez consultada Julio 2014):
http://nadavfima.com/cardsui-view-library/

https://github.com/nadavfima/cardsui-for-android

[11] ActionBar for Android (última vez consultada Julio 2014):
http://developer.android.com/intl/es/guide/topics/ui/actionbar.html

[12] SharedPreferences Android (última vez consultada Julio 2014):
http://developer.android.com/guide/topics/data/data-storage.html#pref

[13] AsyncTask Android (última vez consultada Julio 2014):
http://developer.android.com/reference/android/os/AsyncTask.html

[14] Cuota mercado Android segundo trimestre 2014 (consultado en Julio 2014):
http://www.elandroidelibre.com/2014/07/android-alcanza-el-85-de-cuota-de-mercado-en-el-segundo-trimestre-del-ano.html

[15] Instalar Tomcat como servicio de Windows (última vez consultado Julio 2014):
http://antuansoft.blogspot.com.es/2013/07/instalar-tomcat-como-servicio-de.html

1.11 Definiciones y abreviaturas

AES	Advanced Encryption Standard (Estándar avanzado de encriptación)
API	Application Programming Interface (Interfaz de programación de aplicaciones)
GCM	Google Cloud Messaging (Sistema de mensajería en la nube de Google)
HTML	HyperText Markup Language (Lenguaje de marcado de hipertexto)
HTTP	HyperText Transfer Protocolo (Protocolo de transferencia de hipertexto)
ILIAS	Entorno virtual de enseñanza de código abierto.
Kernel	Software que constituye una parte fundamental del sistema operativo.
kSOAP	Al igual que SOAP, pero diseñado para dispositivos más limitados.
kXML	Analizador sintáctico de XML ligero
Linux	Combinación de kernel libre similar a Unix con el sistema GNU.
NFC	Near Field Communication (Comunicación por campo cercano)
OHA	Open Handset Alliance (Alianza comercial formada por Google entre otros)
PKCS	Public Key Criptography Standard (Estándar criptográfico de clave pública)
QR	Quick Response (Respuesta rápida)
RSA	Rivest Shamir Adleman (Iniciales de los apellidos de los creadores de la empresa y del algoritmo de encriptación RSA)
RSS	Really Simple Syndication (Formato XML para compartir contenido en la web)
SDK	Software Development Kit (Kit de desarrollo software)
SHA	Secure Hash Algorithm (Algoritomo de *Hash* Seguro)
SOAP	Simple Object Access Protocol (Define la comunicación de dos objetos en diferentes procesos intercambiando datos XML)
TCP	Transmission Control Protocol (Protocolo de control de transmisión)
XML	eXtensible Markup Language (Lenguaje de marcas)

2 ANEXOS

2.1 Anexo I: Manual de administración del servidor

Suponiendo un escenario inicial sin servidor alguno, los pasos a seguir para conseguir un servidor totalmente operativo son:

- Ordenador conectado a Internet (con requisitos hardware necesarios para ejecutar Tomcat, Java y MySQL).
- Servidor web Apache Tomcat 7 instalado y configurado en el ordenador.
- Máquina virtual Java instalada y configurada en el ordenador.
- Base de datos MySQL instalada en el ordenador (usuario: root, clave: root y puerto: 3306).

Una vez cumplidos los anteriores requisitos, ya se puede importar el paquete con el proyecto del servidor a la carpeta 'webapps' de Tomcat (Directorio de instalación/webapps). Para ello el servicio [15] de Tomcat no debe estar ejecutándose, una vez copiado, se ejecuta.

Una vez está ejecutado el servicio, debemos crear la base de datos. Ejecutando, por ejemplo, MySQL Workbench (utilizado en el desarrollo del proyecto), seguimos los siguientes pasos:

1. Abrimos una conexión con la base de datos.

Figura 36 Conexión con la base de datos

2. Abrimos el archivo 'tutoguia_db.sql' y lo ejecutamos pinchando en el tercer icono con forma de rayo.

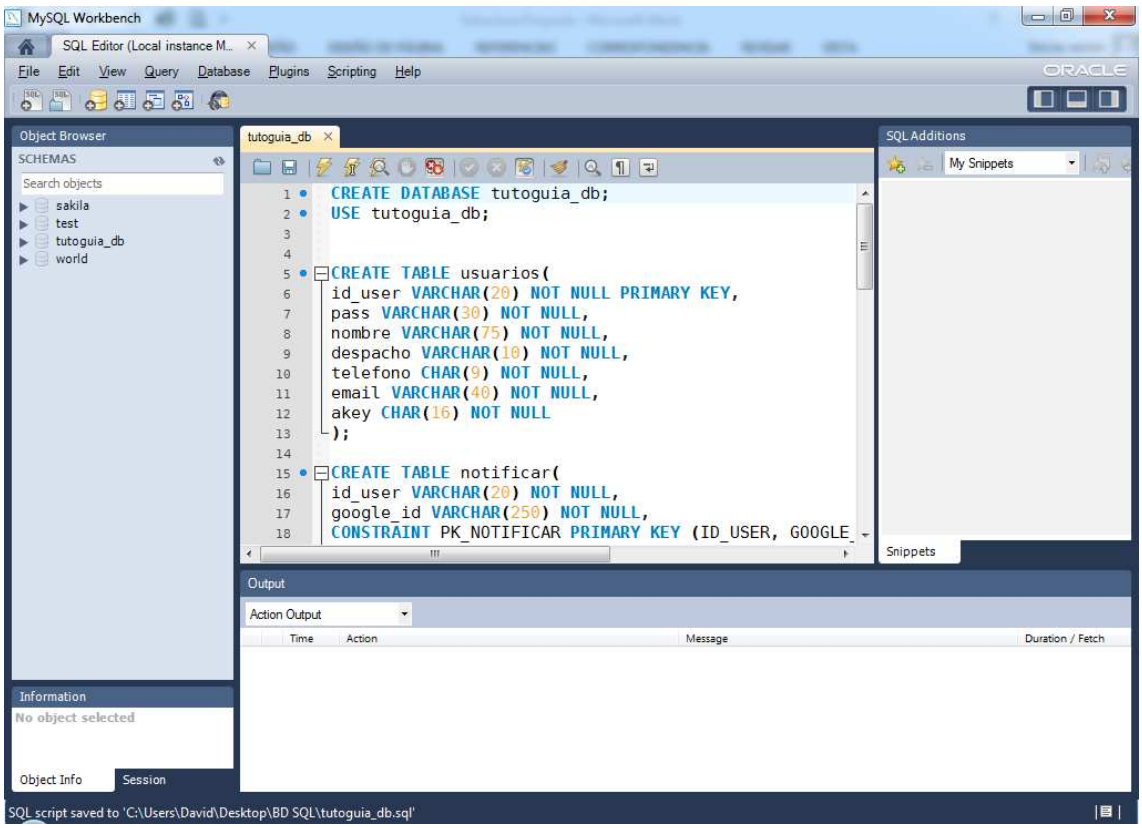


Figura 37. Interfaz MySQL Workbench

Hecho esto, ya está el servidor y la base de datos preparados para funcionar.

La documentación *javadoc* se encuentra en el CD en la carpeta: DOCUMENTACIÓN/Javadoc Server/index.html

2.2 Anexo II: Manual de usuario de la aplicación

Una vez instalada la aplicación, lo primero que preguntará al usuario es el modo en que se va a ejecutar:

- Modo estudiante: con la interfaz necesaria para visualizar avisos y tutorías.
- Modo profesor: con la interfaz necesaria para la gestión de avisos y tutorías.

La misma aplicación sirve, tanto para los profesores, como para los estudiantes, marcando un checkbox en 'Opciones' se cambia entre un modo y otro, aunque para hacer uso de las acciones de los profesores hay que estar debidamente registrado.

En la siguiente imagen se muestra la forma en que se pregunta al usuario el modo en que se ejecuta la aplicación y el menú 'Opciones'.

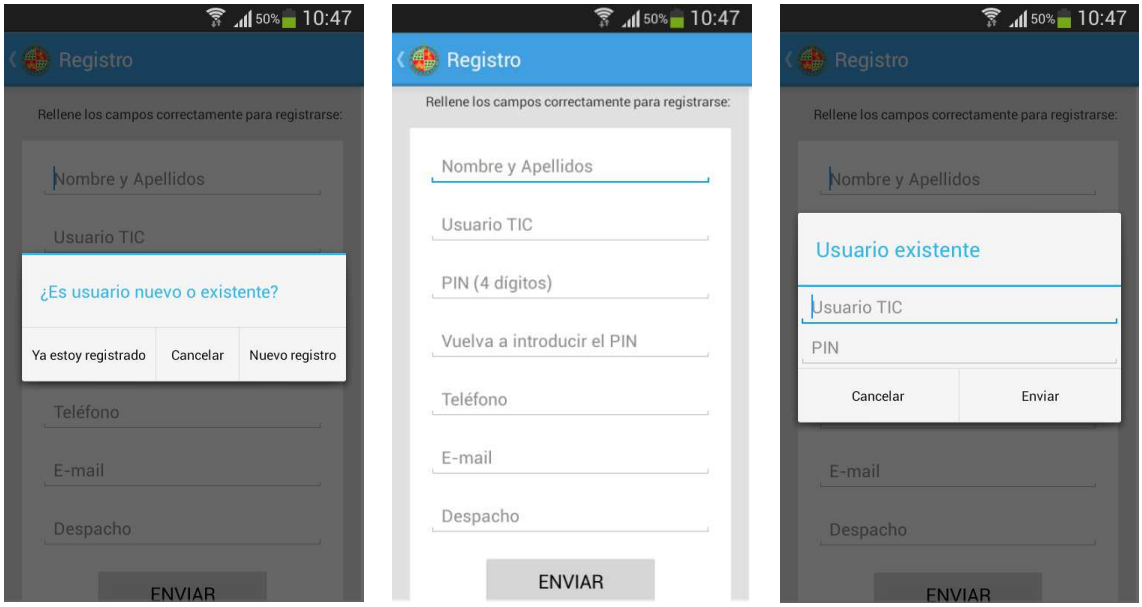
a) b)

Figura 38. Modo aplicación a). Opciones de la aplicación b)

2.2.1 Modo profesor

Cuando se selecciona el modo Profesor, automáticamente preguntará si se quiere registrar con nuestra cuenta TIC ahora (es necesaria conexión a Internet). En caso de no hacerlo, no se podrá hacer nada en la aplicación y se cerrará. Si se elige 'Sí', hay dos formas de autentificarse registrándose como nuevo usuario (introduciendo

todos los datos necesarios), o como usuario ya registrado (introduciendo sólo usuario TIC y PIN).



a) b) c)

Figura 39. Registro usuario. Pregunta tipo usuario a), Usuario nuevo b), Usuario antiguo c)

Una vez introducidos los datos, de una forma u otra, si los datos son correctos, se habrá autentificado correctamente, y se podrá usar la aplicación.

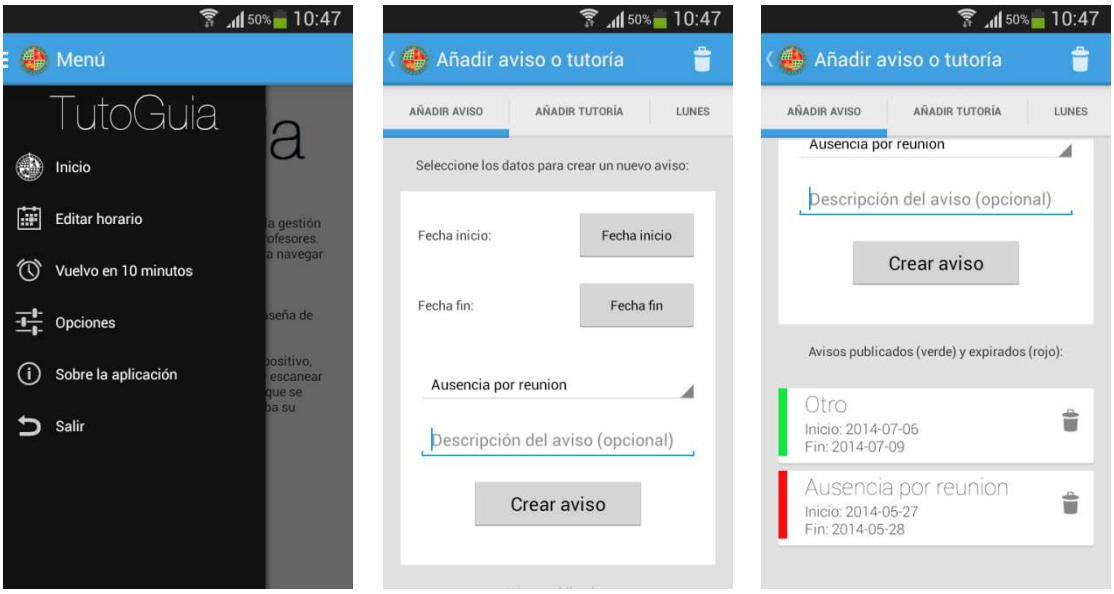
2.2.1.1 Insertar avisos

La primera pestaña que aparece cuando se pulsa en 'Editar horario', la que posiblemente más se use es el de 'Añadir aviso'.

Simplemente se introducen los datos necesarios, como fecha de inicio, fecha de fin, motivo del aviso y descripción (opcional). Pulsando en enviar, la aplicación se comunicará con el servidor, y se insertará en la base de datos.

Pulsando el icono del cubo de basura de cada aviso se elimina.

Además se indica con color verde o rojo si el aviso está activo o ha expirado.

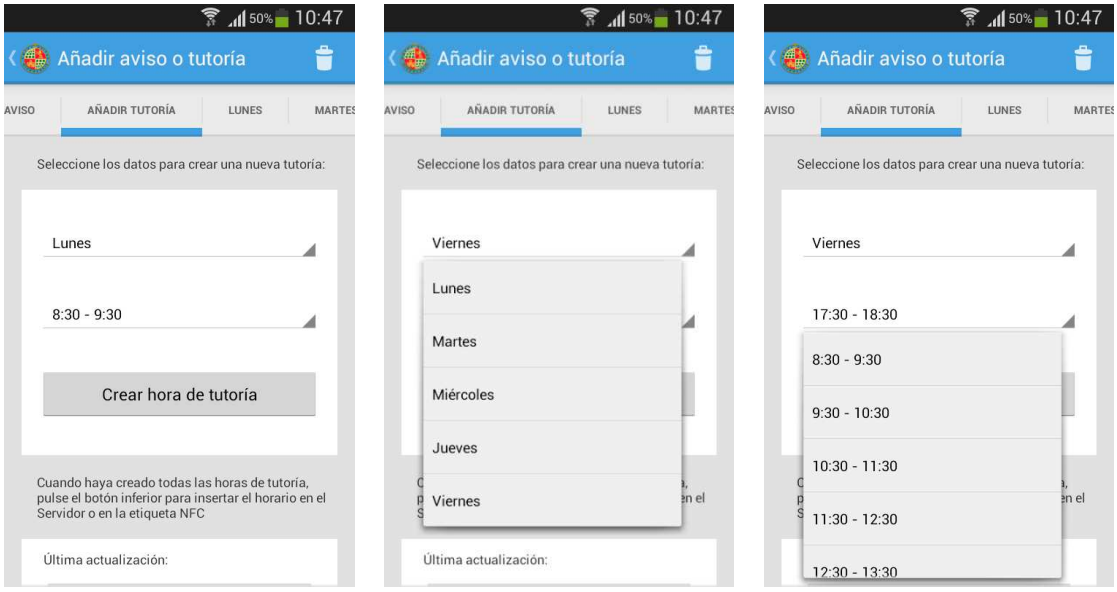


a) b) c)

Figura 40. Menú lateral a), Insertar aviso b), Avisos creados c)

2.2.1.2 Insertar tutorías

La segunda pestaña que aparece cuando se pulsa en 'Editar horario' es la de 'Añadir tutoría'. Para añadir una hora de tutoría se debe seleccionar el día y el tramo horario de la tutoría. Una vez elegidos, se pincha el botón 'Crear hora de tutoría' y se guarda localmente en el dispositivo.



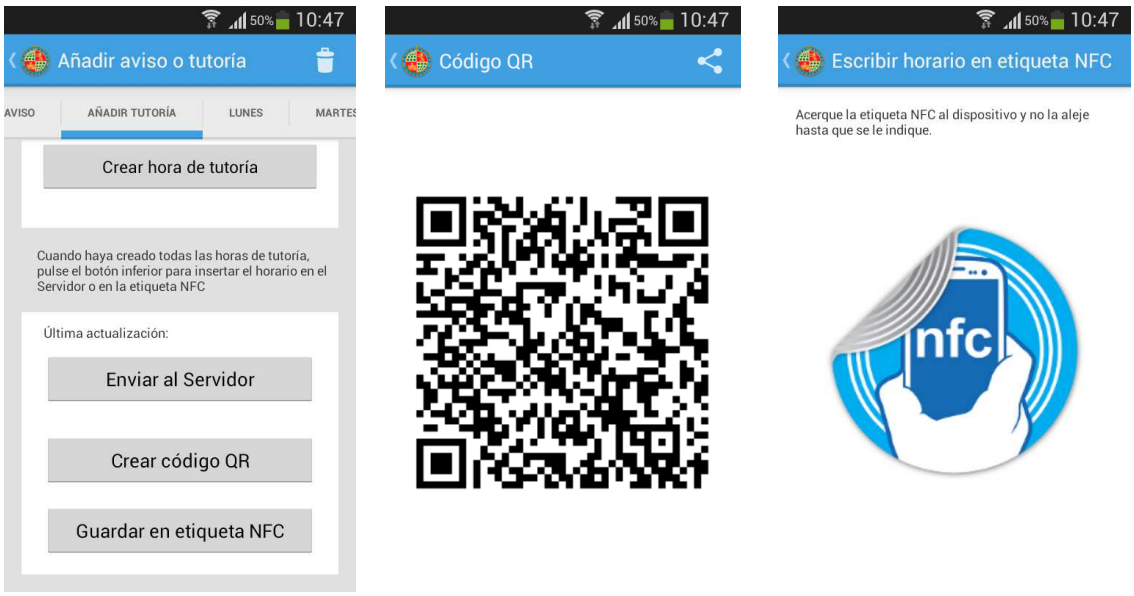
a) b) c)

Figura 41. Pantalla creación tutorías a), Selección día b), Selección tramo c)

Para guardar o compartir el horario de tutorías, una vez esté completo con todos los tramos horarios de los correspondientes días, se pueden utilizar tres formas:

- Enviar al Servidor: el horario se envía al servidor y se almacena en la base de datos. Este paso es necesario para la actualización 'online' para que los alumnos obtengan el horario automáticamente una vez actualizaad.
- Crear código QR: el horario junto con los datos del profesor se escriben en un código QR. Posteriormente este código puede compartirse por correo, guardarse, imprimirse, etc.
- Guardar en etiqueta NFC: al igual que con el código QR, pero la información se guarda en una etiqueta NFC.

Para hacer pública la información a través del código QR y/o la etiqueta NFC (se puede usar un tipo de etiqueta NFC adhesiva universal), se recomienda utilizar un documento similar a la plantilla que se muestra en la Figura.
Cada vez que se actualice el horario de tutorías, debería actualizarse la información impresa del documento, así como el código QR y la etiqueta NFC (pueden reescribirse).



a) b) c)

Figura 42. Opciones publicación horario a), Código QR generado b), Escribir etiqueta NFC c)

Figura 43. Etiqueta NFC adhesiva

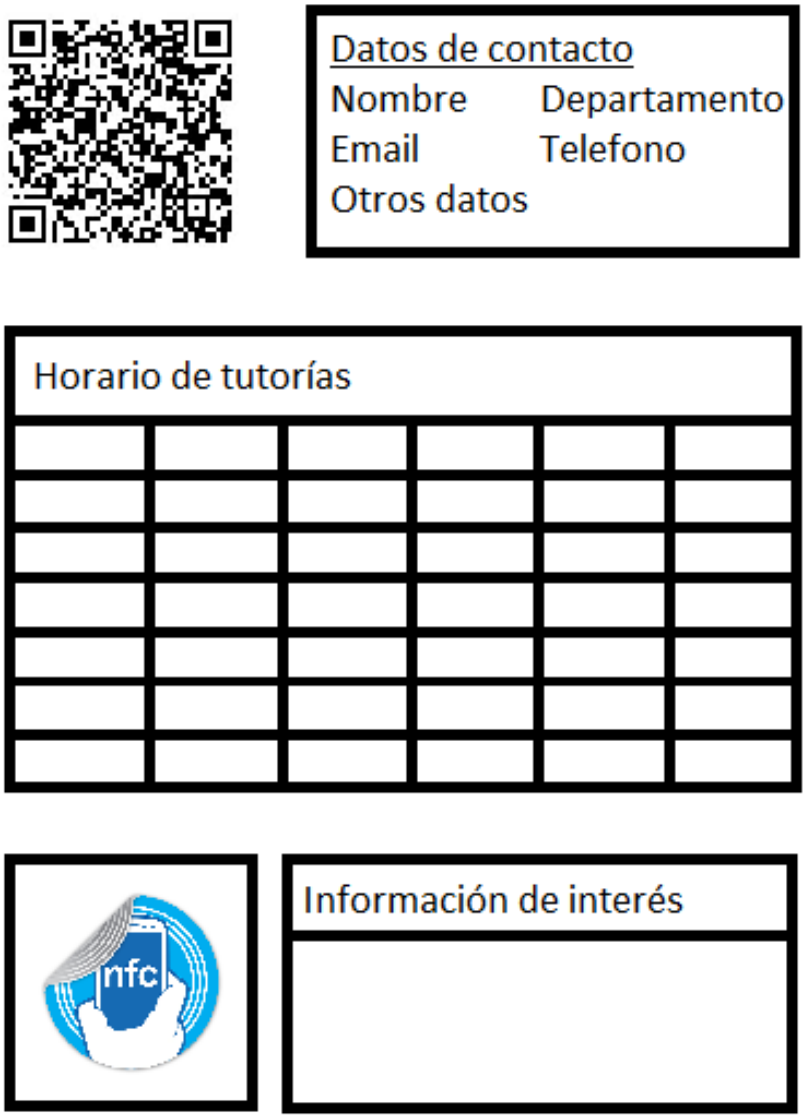


Figura 44. Plantilla documento informativo puerta despacho

En la imagen a la derecha se muestra un ejemplo de tutorías registradas el Lunes.

Se puede desplazar por los distintos días de la semana deslizando las pantallas.

En cada tutoría se muestra un icono de un cubo de basura, pinchando en él se elimina esa hora de tutoría.

Si se pretende eliminar todas las horas de tutorías de todos los días, se puede hacer en un solo paso, pinchando en el icono que aparece en la barra superior azul. Al pincharlo se da la opción de qué borrar, las todas las tutorías o todos los avisos.

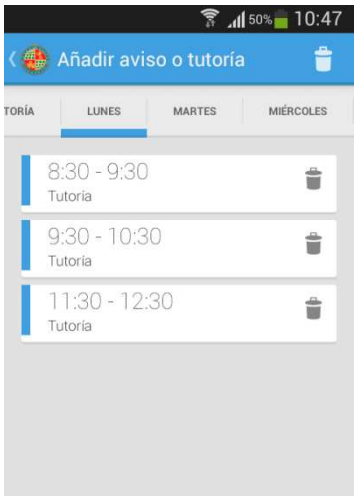
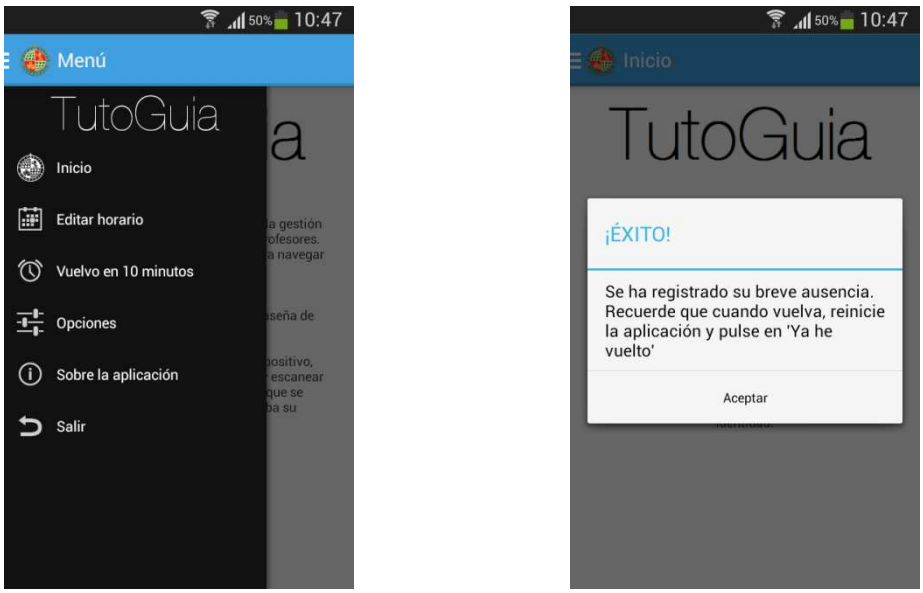


Figura 45. Ejemplo visualización tutoría

2.2.1.3 Menú 'Vuelvo en 10 minutos'

La opción 'Vuelvo en 10 minutos' del menú lateral es útil cuando un profesor debe ausentarse brevemente cuando está en hora de tutoría. Para ello, activando esta opción, se registrará en el servidor y automáticamente notificará a los alumnos que 'sigan' al profesor la ausencia activada. Cuando se desactive, se volverá a notificar que el profesor está de nuevo en su despacho.



a) b)

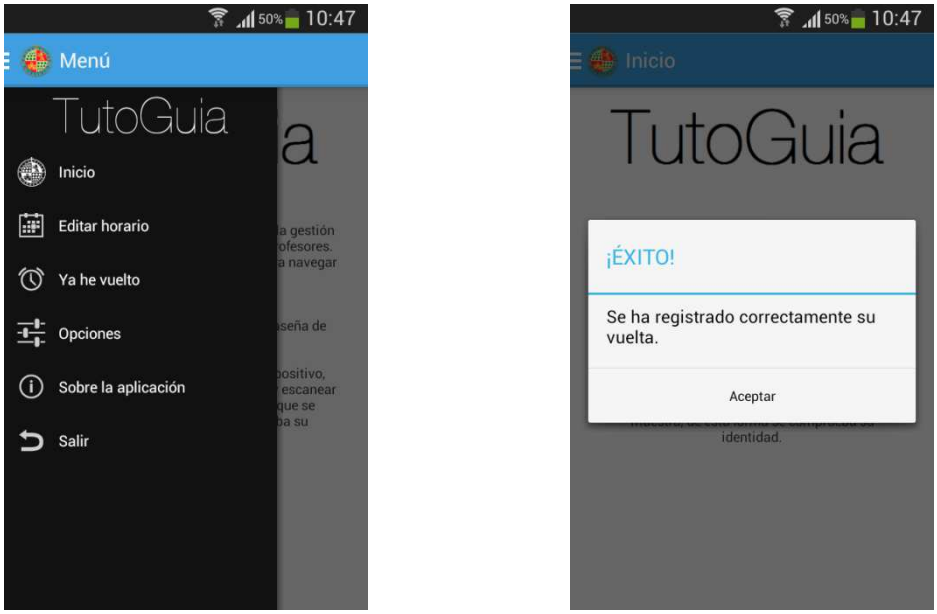
Figura 46. Menú lateral a), Registro 'Vuelvo en 10 minutos' b

Este modo sólo puede activarse si hay registrada una tutoría en el día y hora en que se quiera activar el modo, en caso contrario, mostrará uno de los dos mensajes inferiores:

a) b)

Figura 47. Mensaje error 'Vuelvo en 10 minutos'. No está en hora de tutoría a), Fin de semana b)

Para desactivar el modo, se debe pulsar la opción ‘Ya he vuelto’.



a) b)

Figura 48. Menú lateral 'Ya he vuelto' a), Mensaje éxito 'Ya he vuelto'

2.2.2 Modo estudiante

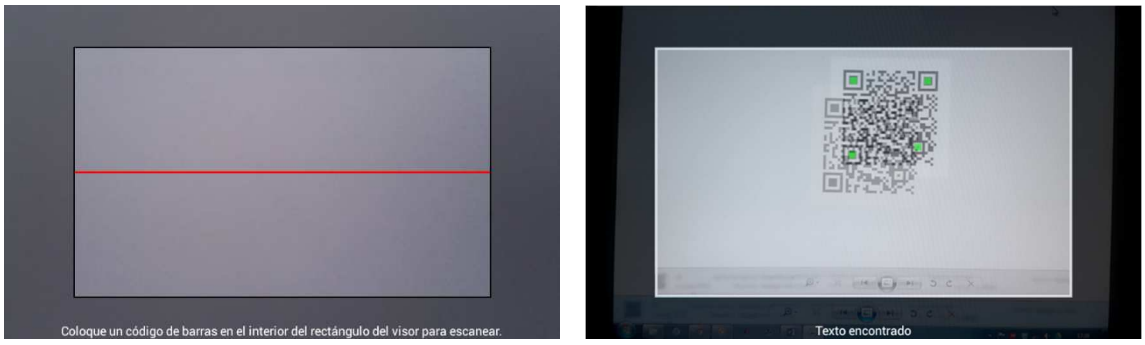
El modo estudiante sirve para 'que los alumnos consulten los avisos y horas de tutorías que publiquen los profesores que elijan. Existen dos formas parar obtener los datos de los profesores: escaneando un código QR o escaneando una etiqueta NFC (depende de cada terminal, si no lo permite se indica en la pantalla de Inicio).

a) b) c)

Figura 49. Pantalla bienvenida terminal con NFC a), Pantalla bienvenida terminal sin NFC b), Menú lateral c)

2.2.2.1 Capturar código QR

Al pulsar esta opción del menú se activará el escáner de códigos QR:



a) b)

Figura 50. Pantalla captura código QR a), Código QR capturado b)

Basta con colocar el visor en el código QR del profesor y automáticamente guardará los datos en la base de datos de la aplicación. En el código QR contiene información sobre el profesor y el horario de tutorías. Si ya se tenía al profesor guardado en la base de datos y se escanea un QR con un horario más antiguo se mostrará un mensaje que lo indica.

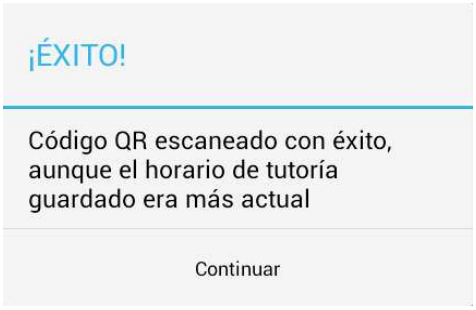


Figura 51. Mensaje éxito código QR capturado

Además, se preguntará al alumno si desea 'seguir' al profesor, es decir, registrarse en la base de datos para notificarle cualquier cambio que haga el profesor (si no lo ha hecho ya).

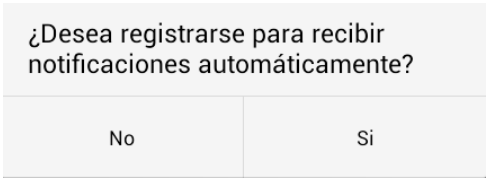


Figura 52. Pregunta registro notificaciones

2.2.2.2 Escanear etiqueta NFC

Para escanear una etiqueta NFC, basta con acercar el terminal a dicha etiqueta y automáticamente se guardará los datos que contenga. Por lo demás funciona exactamente igual a como si se escanease un código QR.

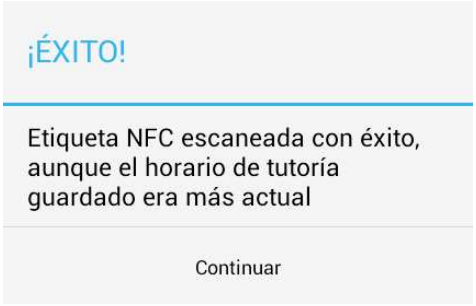
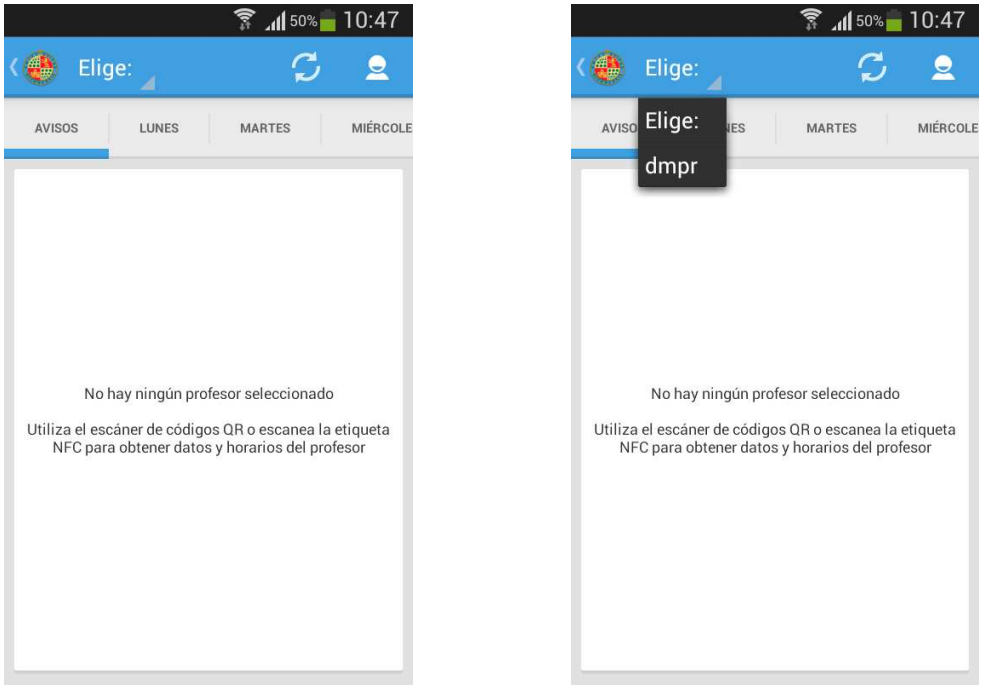


Figura 53. Mensaje éxito NFC escaneada

2.2.2.3 Horarios de tutorías, avisos, lista de profesores y personalización

Para visualizar los horarios de tutorías de los profesores guardados debemos seleccionarlo desde el desplegable de la barra azul. Por defecto viene marcado 'Elige:'

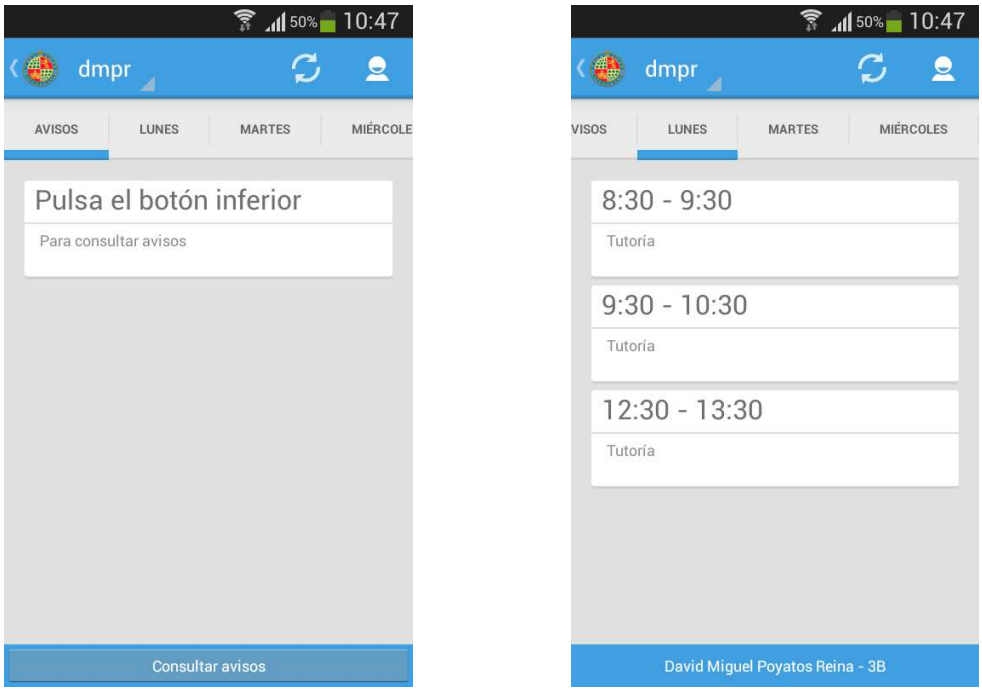
Como no hay ningún profesor seleccionado se indica en la visualización de cada una de las pestañas de avisos y días de la semana.



a) b)

Figura 54. Pantalla visualización sin profesor seleccionado a), Desplegable selector de profesores b)

Al seleccionar un profesor, la visualización de las pestañas cambia y será algo parecido a las imágenes inferiores. En la de la izquierda se mostrarán los avisos (para consultarlos hay que pulsar el botón inferior 'Consultar avisos' y en la de la derecha las horas de tutoría para un día de la semana, en este caso el Lunes (en la parte inferior se muestra el nombre completo del profesor y su despacho).



a) b)

Figura 55. Visualización avisos a), Visualización tutorías b)

Para añadir funcionalidad, existe un menú que permite:

- Actualizar el horario de tutorías online (no es necesario escanear QR ni NFC).
- Consultar más datos del profesor y envíar correo electrónico
- Editar la lista de profesores guardados y activar/desactivar notificaciones.

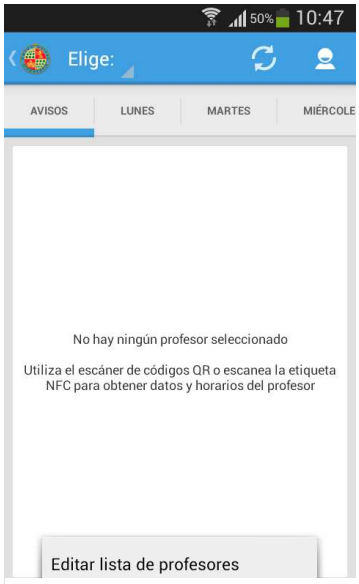


Figura 56. Menú 'Editar lista de profesores'

Dependiendo del tamaño del dispositivo, los elementos del menú se mostrarán en la barra azul superior o de forma tradicional con un menú desplegable inferior (pulsando tecla MENU del terminal).

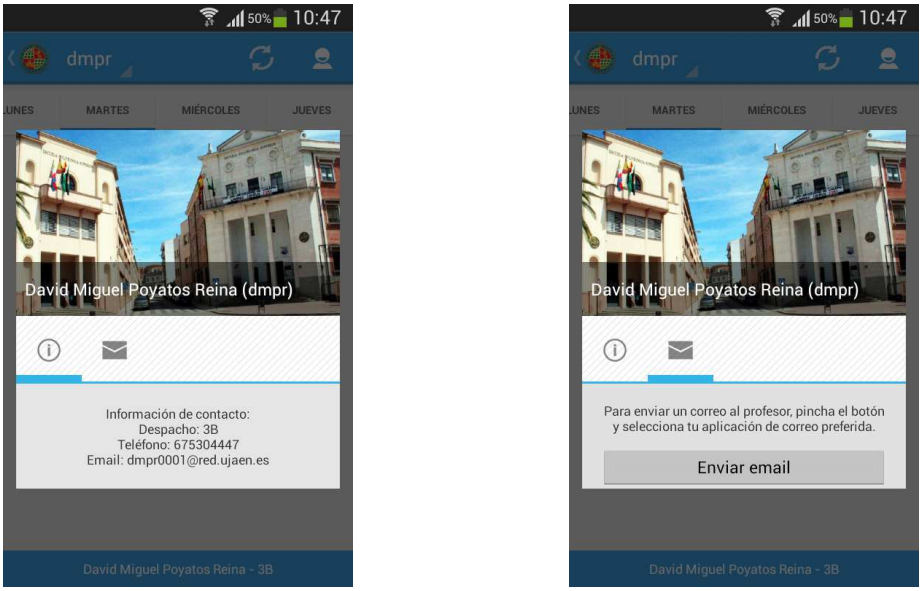
El primer icono (flechas) es para actualizar el horario de tutorías.

El segundo icono (persona) es para consultar los datos del profesor y enviar un correo electrónico.

La tercera opción del menú se muestra en el menú desplegable inferior.

Al pulsar el icono de la persona se mostrará una ventana emergente con información de contacto del profesor: despacho, teléfono y correo

electrónico. Además se ofrece la posibilidad de enviar un correo directamente (no es necesario escribir la dirección de correo del profesor, se pasa directamente a la aplicación de correo que el usuario seleccione).



a) b)

Figura 57. Datos de contacto profesor a), Enviar email a profesor b)

Para eliminar un profesor que ya no se quiere seguir o se quiere activar/desactivar las notificaciones, se puede hacer seleccionando la tercera opción del menú 'Editar lista de profesores'.

En el desplegable se selecciona el usuario TIC del profesor y en función de si las notificaciones están o no activadas, el botón de la derecha mostrará la opción contraria.

En la parte inferior se muestran en forma de 'cards' todos los profesores guardados con su usuario TIC y su nombre completo.

Para añadir personalización, en el menú 'Opciones' puede configurarse si se desea que el dispositivo vibre cuando se recibe una notificación.

Figura 58. Editar lista de profesores

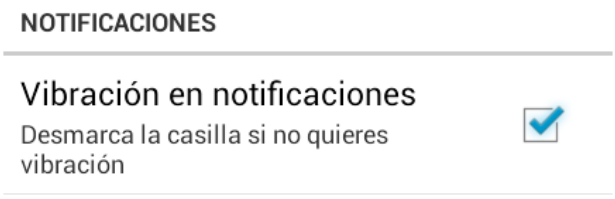


Figura 59. Configuración vibración notificaciones

2.3 Anexo III: Configuración Google Cloud Messaging

Google Cloud Messaging (GCM) para Android es un servicio gratuito que permite enviar información desde un servidor de aplicación a un dispositivo Android. El servicio GCM controla el almacenamiento en cola de los mensajes y la entrega a las aplicaciones.

Características básicas:

- Para que una aplicación reciba mensajes, no es necesario que ésta se encuentre ejecutándose, sino que el sistema la despertará cuando el mensaje llegue.
- GCM pasa la información recibida directamente a la aplicación, la cual tiene control total sobre ella.
- Requiere dispositivos con Android 2.2 o superior que tengan instalada la aplicación de Google Play Store.
- Hace uso de una conexión existente a los servicios de Google. Es necesario que los usuarios tengan configurada su cuenta de Google en el dispositivo.

Pasos para activar el servicio:

1. Crear un nuevo proyecto en la consola de Google APIs (https://code.google.com/apis/console) pulsando en 'Create Project'.

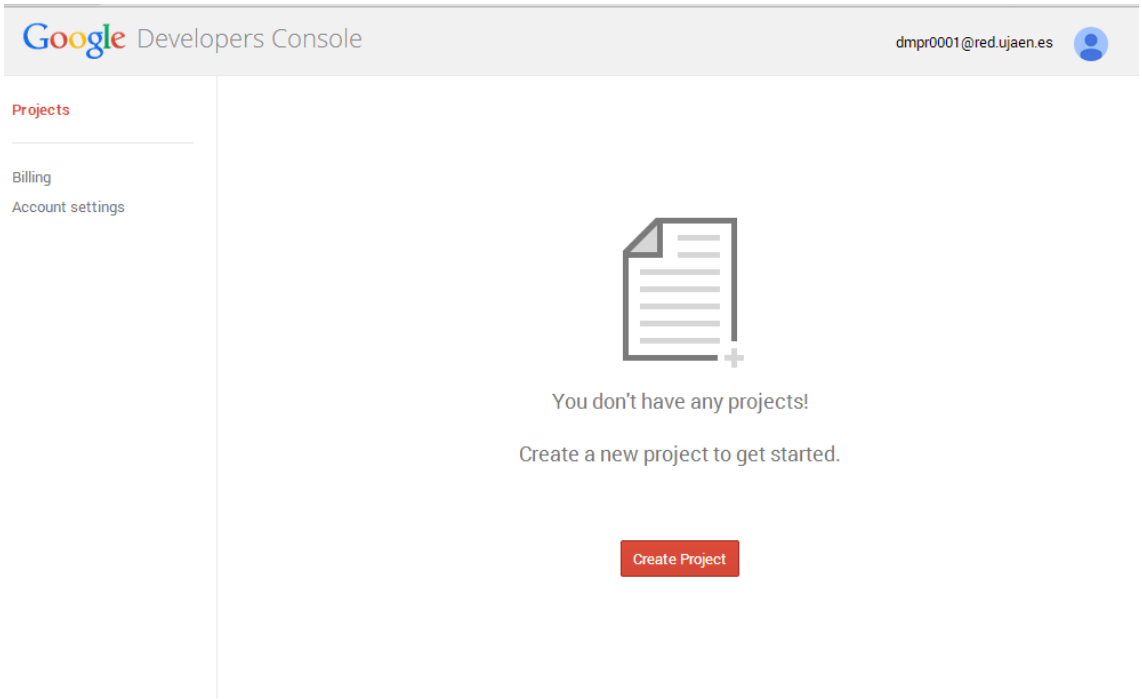


Figura 60. Creación Proyecto en Google Developers Console

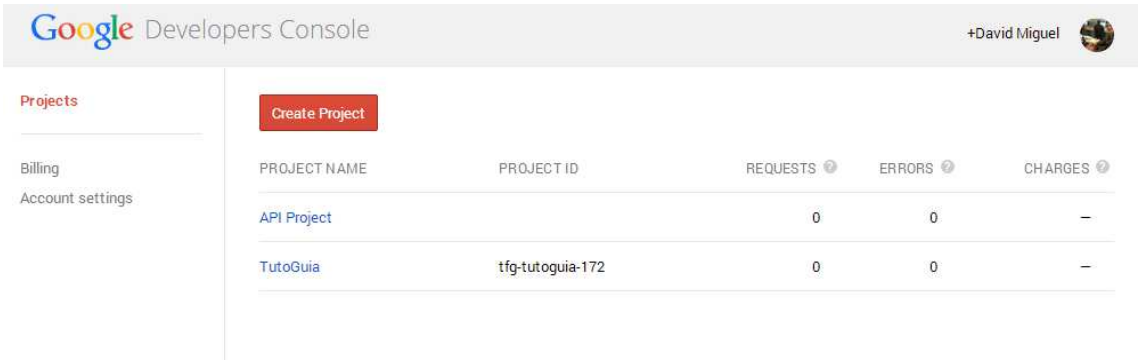


Figura 61. Proyecto TutoGuia creado

2. Una vez creado el proyecto se pincha en él, y en el menú 'APIS & AUTH' / 'Credentials', se obtiene el valor de 'SENDER ID' y se crea la Key ('Create new Key') para 'browser aplications' y se obtiene la 'API Key'.

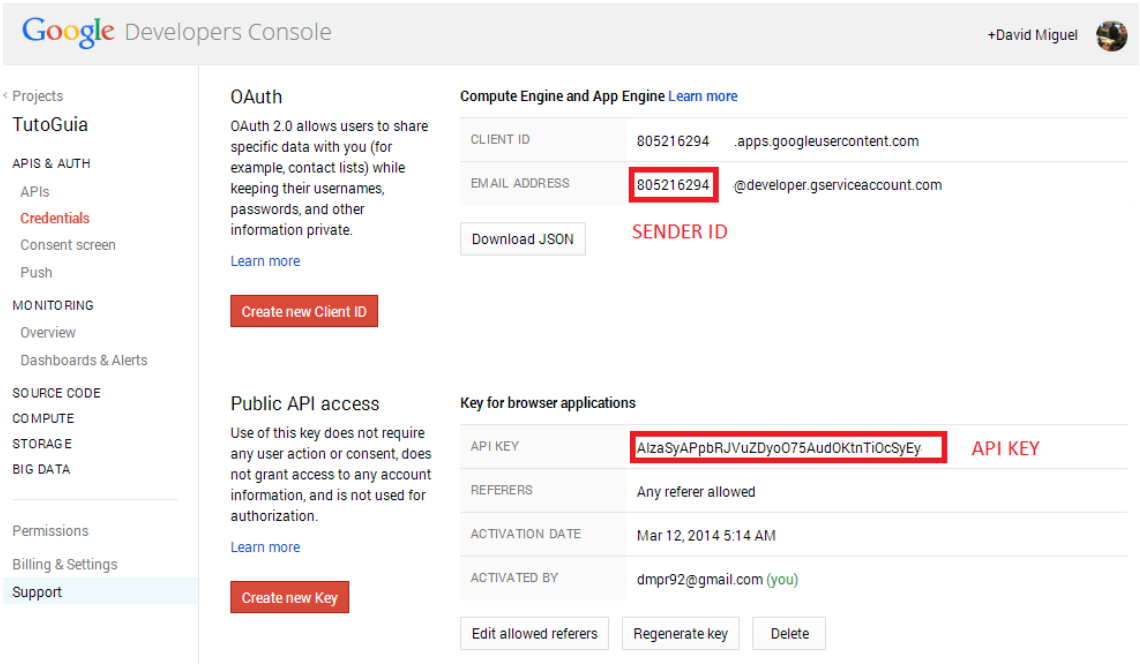


Figura 62. Obtención SENDER ID y API Key

3. En el menú 'APIS & AUTH' / 'APIs' se debe activar la API para 'Google Cloud Messaging for Android'.

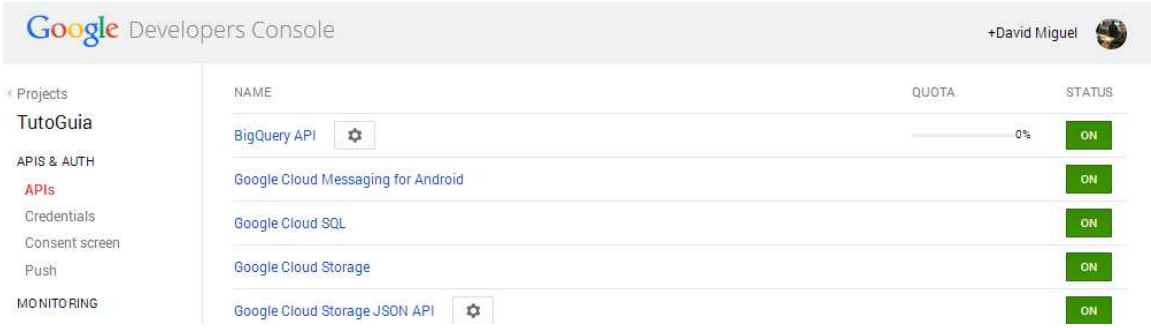


Figura 63. Activación API GCM

Una vez terminado este proceso ya está activado el servicio, el siguiente paso será implementar las funciones necesarias en la aplicación cliente y en la aplicación servidor.

2.3.1 Configuración de la aplicación

1. Instalar la librería de GCM. Paquete “Google Play services”.

   Para instalar la librería, en primer lugar hay que importar al espacio de trabajo dicha librería, antes debe estar descargada mediante el SDK de Android. Para ello, abrir el programa ‘SDK Manager’ y en la sección ‘Extras’ instalar el paquete ‘Google Play services’.

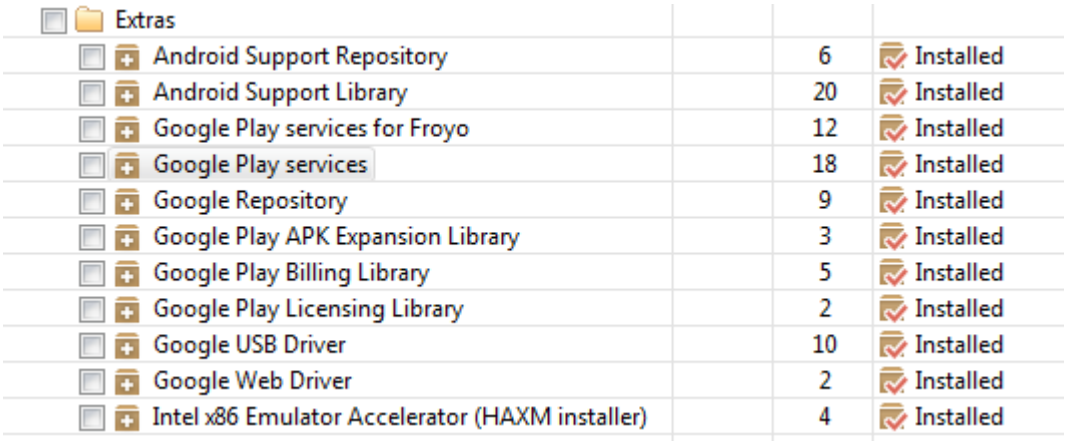


Figura 64. Descarga librería 'Google Play services'

Una vez que el paquete está instalado, se puede importar al espacio de trabajo del entorno de desarrollo.

File → Import → General → Existing Projects into Workspace → Next → Select root directory (Path SDK/extras/google/google_play_services) → Marcar ‘Copy projects into workspace’ → Finish

Ahora hay que importar la librería al proyecto. Para ello se pulsa con el botón derecho del ratón sobre la carpeta del proyecto → Properties (Alt + Enter).

Seleccionando el segundo elemento de la tabla de la izquierda 'Android', se añaden las librerías de las que depende el proyecto.

En la parte inferior, en el apartado 'Library' aparecen las librerías ya añadidas y pulsando el botón 'Add' se pueden seleccionar las librerías disponibles para importar al proyecto.

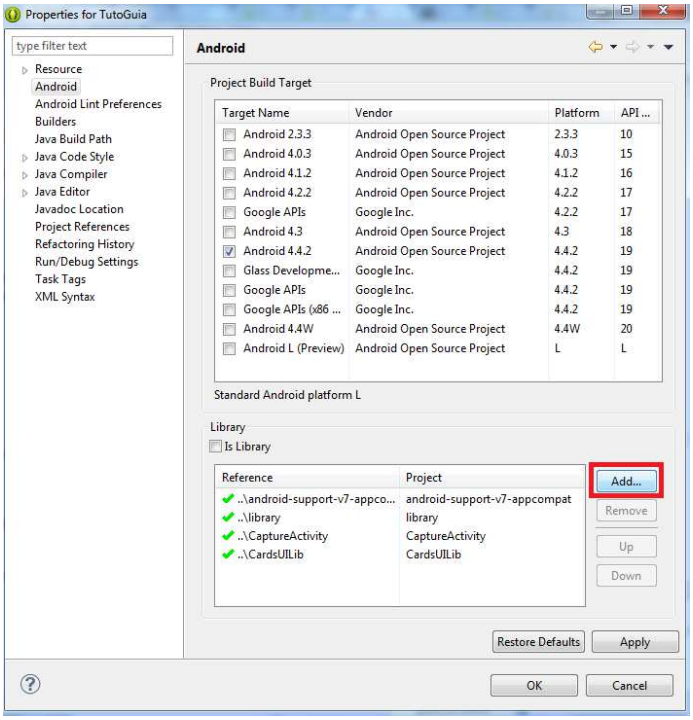


Figura 65. Librerías del proyecto

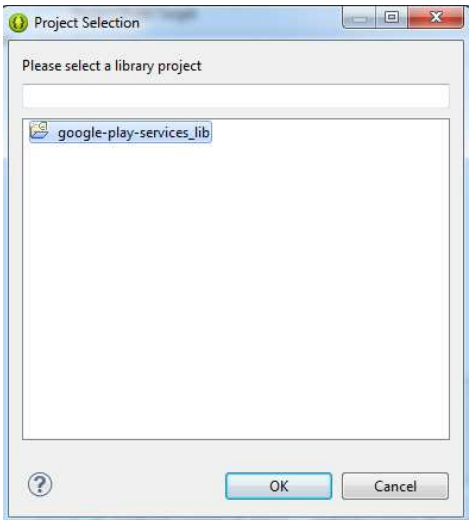


Figura 66. Añadir librería 'Google Play services'

Hecho esto ya está la librería importada en el proyecto.

2. Añadir los permisos necesarios al archivo “AndroidManifest.xml”.

Se añaden permisos para evitar que otras aplicaciones se registren y reciban los mensajes de nuestra aplicación. Se debe especificar el mismo nombre que el del paquete de la aplicación (que se encuentra también indicado en el propio *manifest*).

```
<permission android:name="com.tfg.tutoguia.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
<uses-permission android:name="com.tfg.tutoguia.permission.C2D_MESSAGE" />
```

Se añade el resto de permisos:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

“GET_ACCOUNTS” permite acceder a la cuenta de Google del dispositivo.

Dentro de la etiqueta “application” se añade el receptor o cliente de los mensajes. Son 2 *intents* los que pueden recibirse del GCM: “REGISTRATION” y “RECEIVE”. Gracias a que se está usando la librería de GCM para Android, no hay que implementar ninguna clase propia cliente, sino que se usa la que la librería proporciona *com.google.android.gcm.GCMBroadcastReceiver*.

```
<receiver
    android:name="com.google.android.gcm.GCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.tfg.tutoguia" />
    </intent-filter>
</receiver>
```

Se debe declarar de esta forma, y no mediante programación, para que los mensajes sean recibidos incluso si la aplicación no se está ejecutando.

El permiso “SEND” se incluye para que tan sólo se reciban *intents* provenientes de los servidores GCM.

Por último, se añade un servicio cuyo nombre debe ser “GCMIntentService”. Este nombre es al que se llama desde el receptor anterior. Este servicio sí hay que implementarlo con acciones propias.

```
<service android:name=".GCMIntentService" />
```

3. Implementar el código para registrase al servicio.

En la actividad de la aplicación, en el método 'onCreate' hay que añadir el código para que la aplicación se registre en los servidores de GCM y así obtener el identificador necesario para el envío de notificaciones.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ctx=this;

    // Check device for Play Services APK.
    if(isOnline()==true){
        if (checkPlayServices()) {
            // If this check succeeds, proceed with normal processing.
            // Otherwise, prompt user to get valid Play Services APK.

        try{
           GCMRegistrar.checkDevice(this);
           GCMRegistrar.checkManifest(this);

           final String regId = GCMRegistrar.getRegistrationId(this);
           if (regId.equals("")) {
              GCMRegistrar.register(this, SENDER_ID);
           }
        } catch (UnsupportedOperationException e) {
           Toast.makeText(this, "El dispositivo no soporta GCM.", Toast.LENGTH_LONG).show();
        } catch (IllegalStateException e) {
            Toast.makeText(this, "El manifest no está bien configurado.", Toast.LENGTH_LONG).show();
        }
        }
    }
```

4. Implementar el código del servicio que va a recibir los *intents.*

Cada vez que el dispositivo reciba un mensaje de los servidores de GCM, lo hará a través de un servicio que se ejecuta aunque la aplicación no lo esté haciendo. Este servicio consta de cuatro métodos que se encargan de procesar cuatro tipos de mensajes: error, mensaje recibido, registro correcto, baja correcta.

```
import com.google.android.gcm.GCMBaseIntentService;

public class GCMIntentService extends GCMBaseIntentService {

   @Override
   protected void onError(Context context, String errorId){
      // Error en el registro: tratamiento del error
   }

   @Override
   protected void onMessage(Context context, Intent intent){
      // Notificación recibida: informo al usuario u otra acción
   }

   @Override
   protected void onRegistered(Context context, String regId){
      // Registro correcto: envío el regId a mi servidor
   }

   @Override
   protected void onUnregistered(Context context, String regId){
      // Borrado correcto: informo a mi servidor
   }
}
```

2.3.2 Configuración del servidor

En este caso la configuración es más simple, ya que no hay que hacer uso de librerías, ni crear servicios. Tan solo hay que crear una clase con los métodos adecuados para enviar al servidor de GCM los mensajes que se reenviarán a los teléfonos.

Para más detalle de esta clase (Notify.java) véase la documentación *javadoc* incluida en el CD en la carpeta: DOCUMENTACIÓN/Javadoc Server/index.html

3 Estado de las mediciones

En este apartado se definen y determinan las unidades de cada partida que configuran la totalidad del Trabajo Fin de Grado.

Código	Resumen	Unidades
CAPÍTULO 1 – RECURSOS MATERIALES		
	Hardware y software utilizado.	
1.01	**Ordenador**	**1**
1.02	**Microsoft Windows 7**	**1**
1.03	**Microsoft Office 2013**	**1**
1.04	**Android Developer Tools**	**1**
1.05	**MySQL Workbench**	**1**
1.06	**Apache Tomcat 7**	**1**
1.07	**Java (JRE + JDK)**	**1**
1.08	**Samsung Galaxy Ace 3**	**1**
1.09	**Etiquetas NFC**	**2**
CAPÍTULO 2 – RECURSOS HUMANOS		
	Horas empleadas para cada una de las fases de desarrollo.	
2.01	**Análisis del entorno**	**20**
2.02	**Análisis del sistema**	**20**
2.03	**Diseño del sistema**	**40**
2.04	**Implementación**	**150**
2.05	**Pruebas**	**20**
2.06	**Documentación**	**50**
CAPÍTULO 3 – COSTESS INDIRECTOS		
	Costes indirectos aplicados al trabajo.	
3.01	**Conexión a Internet**	
3.02	**Luz**	

4 Presupuesto

En este documento se determinará el coste económico de cada capítulo y del Trabajo Fin de Grado en su totalidad.

4.1 Capítulo 1. Recursos materiales

Unidades	Concepto	Coste (€)	Tiempo de uso	Período de amortización	Subtotal (€)
1	Ordenador	500,00	8 meses	5 años	66,66
1	Microsoft Windows 7	0,00	8 meses		0,00
1	Microsoft Office 2013	0,00	2 meses		0,00
1	Android Developer Tools	0,00	8 meses		0,00
1	MySQL Workbench	0,00	8 meses		0,00
1	Apache Tomcat 7	0,00	8 meses		0,00
1	Java (JRE + JDK)	0,00	8 meses		0,00
1	Samsung Galaxy Ace 3	200,00	8 meses	2 años	66,66
2	Etiquetas NFC	2,50	3 meses		5,00
Total					**138,32**

Tabla 1. Presupuesto 'Recursos materiales'

4.2 Capítulo 2 – Recursos humanos

Concepto	Unidades	Precio/hora(€)	Subtotal (€)
Análisis del entorno	20	25,00	500,00
Análisis del sistema	20	25,00	500,00
Diseño del sistema	40	25,00	1.000,00
Implementación	150	25,00	3.750,00
Pruebas	20	25,00	500,00
Documentación	50	25,00	1.200,00
Total	**300**		**7.450,00**

Tabla 2. Presupuesto 'Recursos humanos'

4.3 Capítulo 3 – Costes indirectos

Concepto	Precio (€)/mes	Tiempo de uso	Subtotal (€)
Conexión a Internet	6,00	8	48,00
Luz	60,00	8	480,00
Total			**528,00**

Tabla 3. Presupuesto 'Costes indirectos'

SUBTOTAL = CAPÍTULO 1 + CAPÍTULO 2 + CAPÍTULO 3 **SUBTOTAL: 8.116,32 €**

IVA (21%): 1.704,42 €

TOTAL: 9.820,74 €

Asciende el presente Presupuesto de Trabajo Fin de Grado “Aplicación de atención personalizada dinámica para interiores basada en terminales Android” a la cantidad de nueve mil ochocientos veinte euros y setenta y cuatro céntimos.

En Linares, a 2 de septiembre de 2014

Fdo.: David Miguel Poyatos Reina
Graduado en Ingeniería Telemática
Precolegiado nº 14354