Trabajo Fin de Grado
Ingeniería de Telecomunicación

Aplicación de contactos compartida

Autor: Alejandro Milán Rodríguez
Tutor: Jose Manuel Fornés Rumbao

Dpto. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Grado
Ingeniería de Telecomunicación

Aplicación de contactos compartida

Autor:
Alejandro Milán Rodríguez

Tutor:
Jose Manuel Fornés Rumbao
Profesor titular

Dep. de Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Aplicación de contactos compartida

Autor: Alejandro Milán Rodríguez

Tutor: Jose Manuel Fornés Rumbao

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mi familia

A mis maestros

Resumen

El trabajo de fin de grado contendrá un proyecto basado en una aplicación que consultará las llamadas entrantes en la base de datos a través de un servicio web del servidor para reemplazar número desconocido por el nombre y los apellidos de la persona llamante. También contendrá un servicio para recibir notificaciones del administrador.

Abstract

The final degree project will contain a project based on an application that will consult the incoming calls in the database through a server web service to replace the unknown number with the name and surnames of the calling person. It will also contain a service to receive notifications from the administrator.

Índice

ÍNDICE DE FIGURAS

BLOQUE 1. INTRODUCCION Y ALCANCE

1.1 Introducción

Se plantea un proyecto con el fin de simplificar y facilitar la conexión entre diferentes usuarios sin necesidad de la interacción física entre ellos. Estos usuarios pertenecerán a un departamento y debido a la incompatibilidad de horarios, de altas y bajas, y de contacto entre ellos, este proyecto nace para conseguir una comunicación más fácil.

El objetivo final de este proyecto, es unificar los diferentes teléfonos de los contactos pertenecientes a un departamento determinado en una base de datos. Esta base de datos podrá ser luego consultada por la aplicación móvil cuando reciba una llamada entrante y conocer quien es el contacto origen (si es que existiese en la base de datos). De esta manera, se logra una accesibilidad mayor entre usuarios sin necesidad de la misma interacción entre ellos.

A su vez, el administrador del sistema, podrá notificar a los diferentes usuarios registrados en la aplicación móvil con mensajes notificatorios como reuniones, noticias o imágenes desde su panel de control.

Para abordar esta tarea, en este apartado analizaremos la situación actual, el estado de arte, los objetivos y el alcance de este proyecto.

1.2 Situación actual

Como hemos explicado anteriormente, la situación actual es la existencia de numerosos departamentos y estos a su vez lleno de usuarios: de altas, bajas, y con horarios dispares entre ellos. Se quiere conseguir así un acercamiento entre ellos de tal manera que la comunicación sea más realizable. El usuario podrá contactar con otro de su mismo departamento sin necesidad de conocer previamente su teléfono.

El administrador también quiere poder enviar a un usuario o a un conjunto de ellos información desde su panel de control, ubicado en el servidor, en forma de mensaje (al recibirlo en el destino el usuario recibirá además una notificación sonora) a fin de mantenerlos informados con noticias, hechos o reuniones. De esta forma se consigue una transmisión de información sin necesidad de estar en contacto personalmente con todos ellos o esperando que revisen el correo.

1.3 Objetivos

El objetivo principal de este proyecto es lograr una aplicación que funcione como una “agenda global” entre usuarios sin mezclar con la personal. Con esta herramienta, si un usuario llama a otro, la aplicación consultará en la base de datos local el nombre del llamante y lo mostrará por pantalla sin necesidad de tenerlo añadido a la agenda de contactos (se prefiere evitar esta opción). Recordemos que en un departamento no todos estarán fijos y siempre habrá altas o bajas que considerar.

Otro objetivo es recibir notificaciones enviadas desde el servidor por el administrador (mensajes, imágenes,

enlaces) a partir de su panel en el servidor con previa autenticación.

Esta herramienta será accedida sólo por profesores (excluyendo al administrador), por lo que será una herramienta cerrada, privada y securizalizada y que cumplirá una serie de requisitos de seguridad (pantalla de autenticación, comprobación de existencia en la base de datos…) para poder ser lanzada.

1.4 Alcance

Este proyecto va a ser divido en 7 bloques, donde se realizará:

- Una **introducción** al mismo y definición de su alcance, así como los objetivos a cumplir y el estado del arte, en este primer bloque.
- En el **segundo** bloque, se realizará un estudio de la situación actual, desde el escenario donde se desea implementar el uso de la herramienta hasta las herramientas que se utilizarán.
- Un **tercer** bloque, donde se detallará la información acerca del desarrollo y se añadirán diagramas para explicar su funcionamiento.
- Un **cuarto** bloque donde se expondrá una línea futura y las conclusiones.
- Un **quinto** bloque donde se detallará un pliego de condiciones que deberá cumplir el escenario donde se implemente el proyecto.
- Finalizaremos con **dos bloques**, a saber un **sexto** bloque, en el que incluiremos la bibliografía; y un **séptimo** y último bloque destinado a los diferentes anexos, donde se incluirán desde el código fuente completo de la herramienta, hasta diferentes manuales o páginas de interés relacionadas con el mismo.

1.5 Estado del arte

Actualmente, existen multitud de aplicaciones móviles para compartir contactos entre usuarios (las conocidas whassap o telegram) o consultar llamadas de desconocidos (trapcall, truecaller, whoscall). Sin embargo, se necesita conocer antes el teléfono destino.

Para el envío y recepción de mensajes, entre las mencionadas arriba, también encontramos firebase cloud messaging, un servicio de google que se puede implementar en proyectos de manera no muy complicada. Esta herramienta permite enviar mensaje a los usuarios desde un servidor web o una aplicación móvil, por ejemplo.

La aplicación móvil estará diseñada en Android Studio al igual que muchas aplicaciones que se crean diaramente. Como el objetivo final es para móviles Android, es una de las mejores herramientas con la que codificar nuestro código. Para la recepción de mensajes usaremos firebase cloud messaging de google. Una herramienta fácil de implementar y pensada para el envio y recepción de mensajes entre otras utilidades.

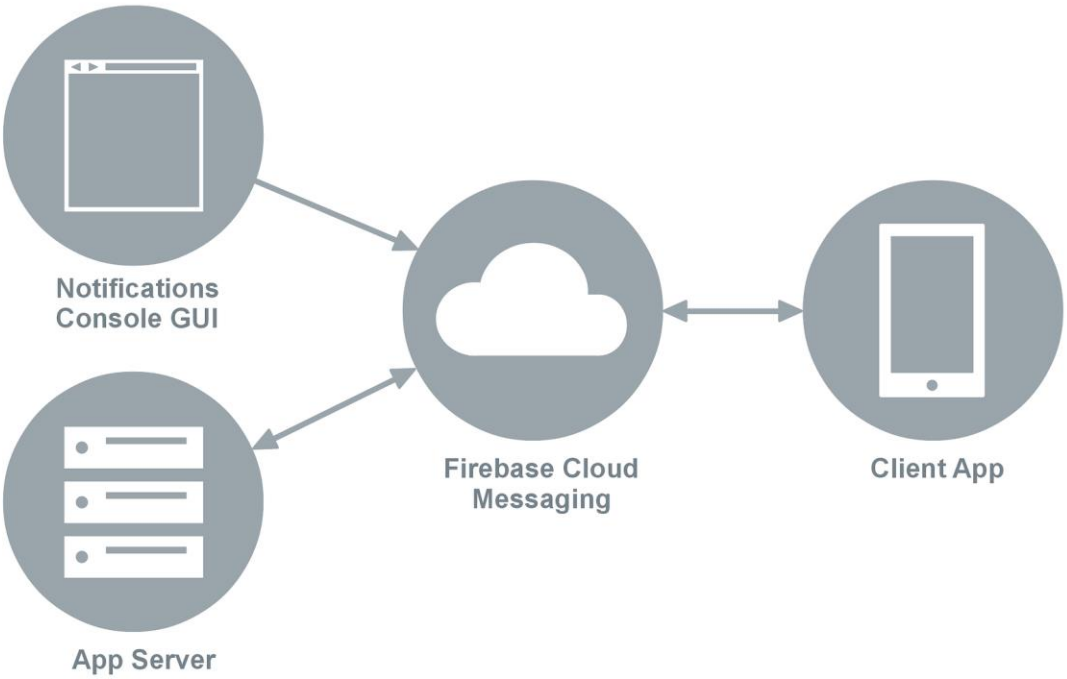

Figura 1 Ejemplo FCM

La gestión del servidor estará basada en php. Para ello se usará apache y para el código se usará Yii [1]. Yii es un framework basado en php pensado para trabajar con base de datos. Las operaciones CRUD (Create, Read, Update, Delete) son así mas fáciles de gestionar. El servidor, además, contendrá lenguaje curl para el envío de notificaciones.

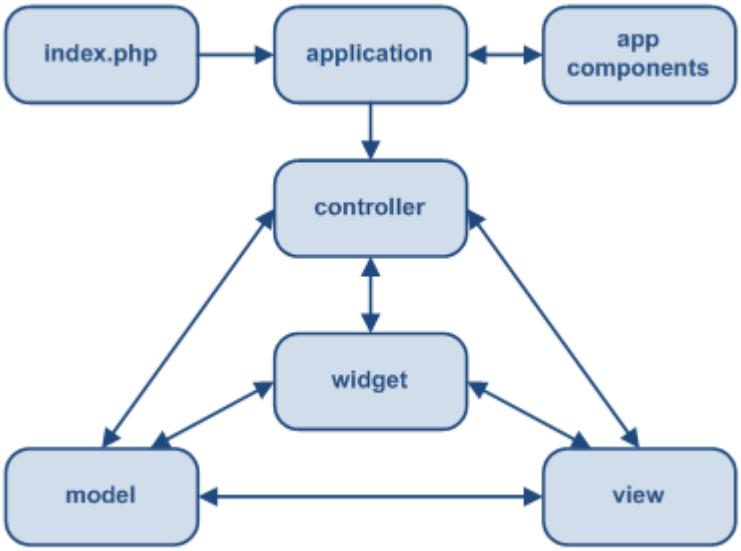

Figura 2 Funcionamiento de Yii

La base de datos, entre las diferentes opciones existentes (sqlite, mariadb, amazon relational database service…) estará diseñada en myqsl para una mejor implementación con la aplicación móvil y el servidor.

BLOQUE 2. ESCENARIO Y SOFTWARE

En este bloque se expondrá el escenario que se quiere realizar, así como el software utilizado para el desarrollo de las diferentes herramientas que lo componen. Se detallará de esta forma el por qué de la elección o decisión de tales herramientas para el desarrollo de este proyecto.

2.1 Escenario

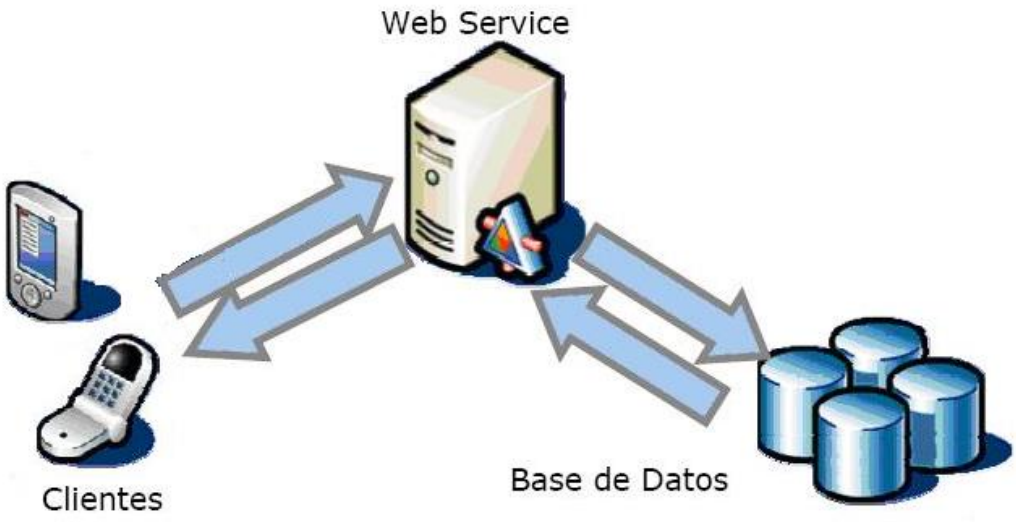

Figura 3 Escenario

El escenario como se puede observar será un servidor con varios servicios webs que consultará en la base de datos y responderá a las peticiones realizadas por los clientes desde su aplicación.

Absolutamente todas las peticiones realizadas por el móvil pasarán por el servidor y usarán sus servicios para obtener respuesta alguna. Para ello el usuario deberá de autenticarse y se comprobará que el usuario esté dado de alta y tiene permisos para poder realizar ciertas acciones.

2.1.1 Servidor

A primera instancia se necesitará un servidor donde se tendrá:

- **Base de datos**: contendrá varias **tablas**:
    1. Tabla **contactos**: Nombre, apellidos, teléfono y token asignado al teléfono de los diferentes contactos.
    2. Tabla **login**: formada por usuario, contraseña para poder acceder a la aplicación móvil.
    3. Tabla **admin**: formada por la cuenta del administrador (usuario, contraseña) para gestionar tanto el servicio web como la base de datos.

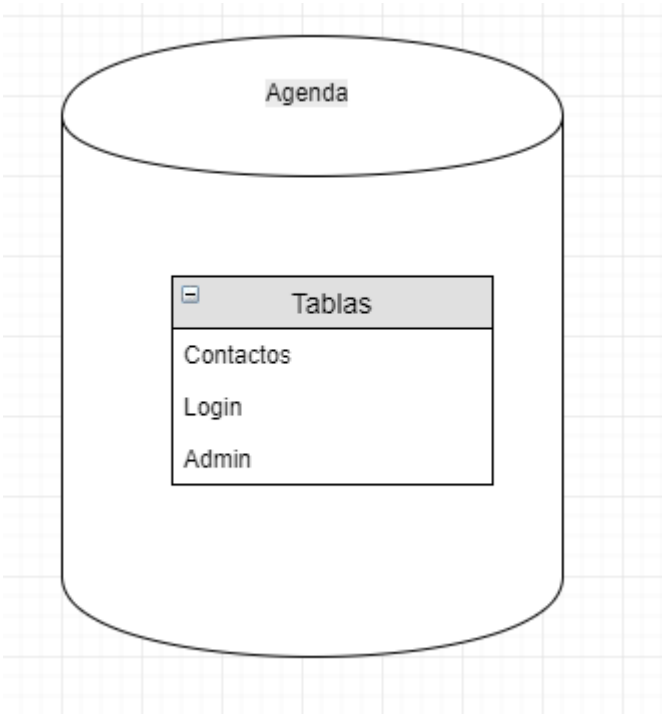

Figura 4 Esquema básico tablas

- **Servidor web**: basado en yii, lenguaje basado en php y optimizado para base de datos. Ejecutará el servicio web que permitirá:
    1. Consultar teléfonos a través de la aplicación móvil.
    2. Administración de usuarios:
        2.1 Agregar contactos.
        2.2 Dar de alta usuarios para la aplicación móvil.
        2.3 Consultar datos de todos los contactos.
        2.4 Enviar notificaciones a los usuarios.
    3. Sincronizar la base de datos.
    4. Autenticar al usuario en el móvil.
    5. Registrar el token del móvil.
    6. Gestionar la autenticación del administrador del sistema.

Figura 5 Panel del administrador en el servidor

2.1.2 Cliente

El cliente de este proyecto solo se hará a nivel de aplicación móvil (Android).

Este cliente tendrá una pantalla de autenticación donde podrá:

- Registrar el token de su teléfono en la base de datos
- Sincronizar la base de datos local con la remota
- Consultar la base de datos

Cuando reciba una llamada entrante se consultará la base de datos para consultar el llamante y se mostrará por pantalla en vez de **número desconocido.** Tendrá integrado un servicio para recibir notificaciones y se mostrarán detalladamente.

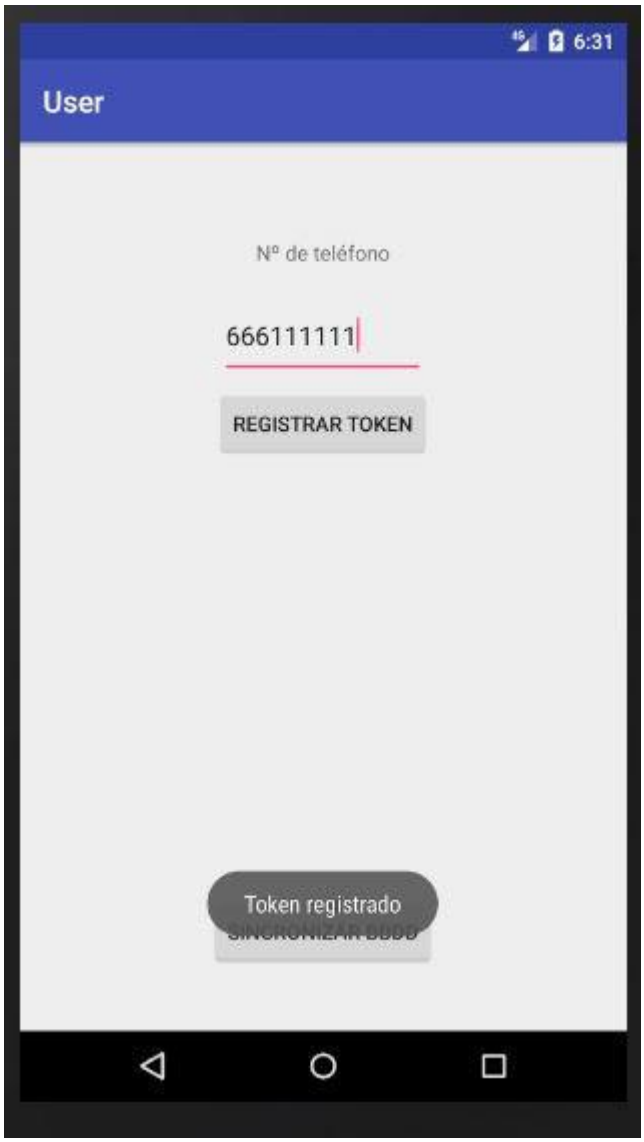

Figura 6 Aplicación móvil. Cliente correctamente registrado

2.2 Software

En este apartado se detallará más información acerca de las herramientas de trabajo que se usarán.

2.2.1 VMware

La primera herramienta que se usará para el servidor será **VMware** [2]. Se utilizará para la virtualización del servidor.

Vmware es un programa que proporciona un software de virtualización para ordenadores compatibles x86. Este programa jugará un aspecto importante ya que en él se simulará el servidor web así como las herramientas necesarias y la base de datos, para una posterior implementación en uno real. Se conseguirá así adaptar la maquina a las necesidades del servidor.

La máquina virtual correrá en una imagen de Linux Mint 18.1 de 64 bits [3]. Las características que poseerá

serán las siguientes.

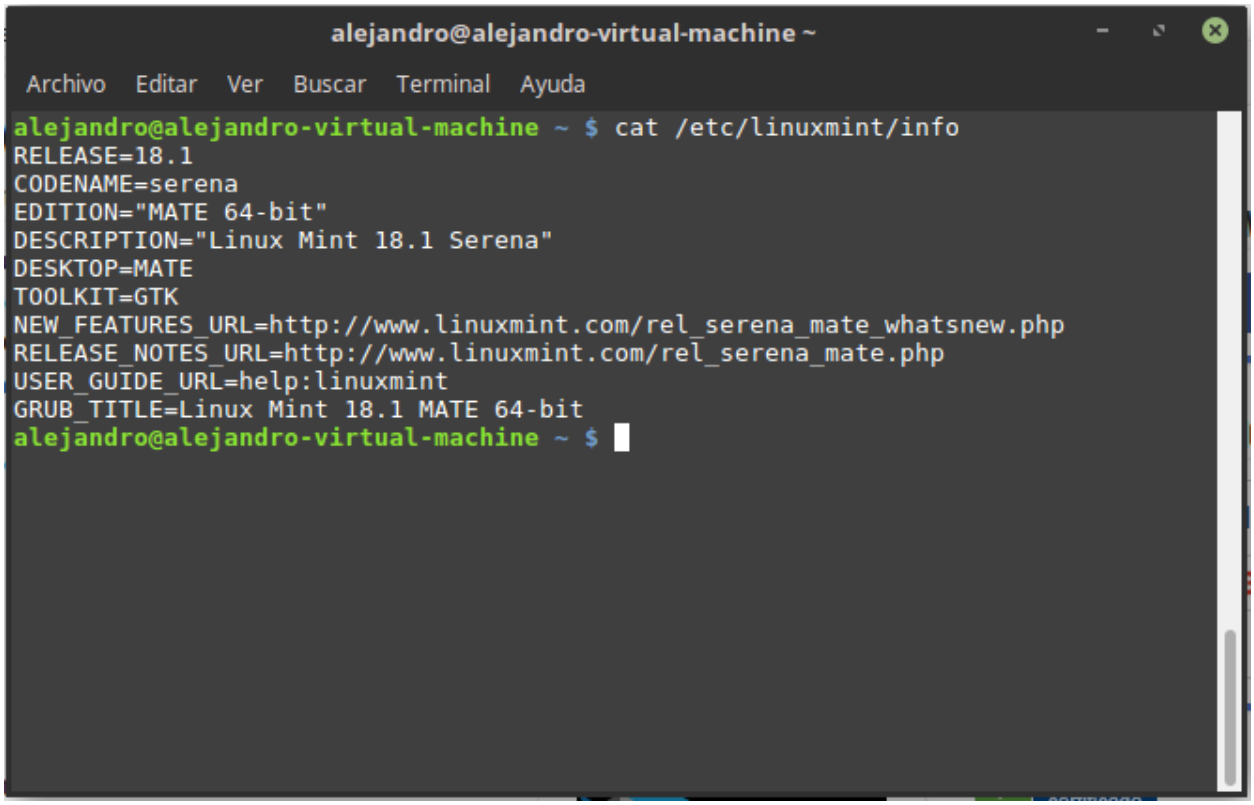

Figura 7 Características servidor emulado

2.2.2 Base de datos

Para la base de datos será necesario trabajar con mysql [4]. Mysql es un sistema de gestión de bases de datos relacional y es una de las más populares. Permite trabajar con un gran volúmenes de datos y no por ello no es rápida.

Será el escenario un entorno donde la modificación de datos será baja y la lectura sin embargo intensive. Esto hará que mysql sea una de las grandes cantidatas. Además su excasa complejidad para ser implementadas en servicios web o aplicaciones móviles la hace una de las mejores opciones.

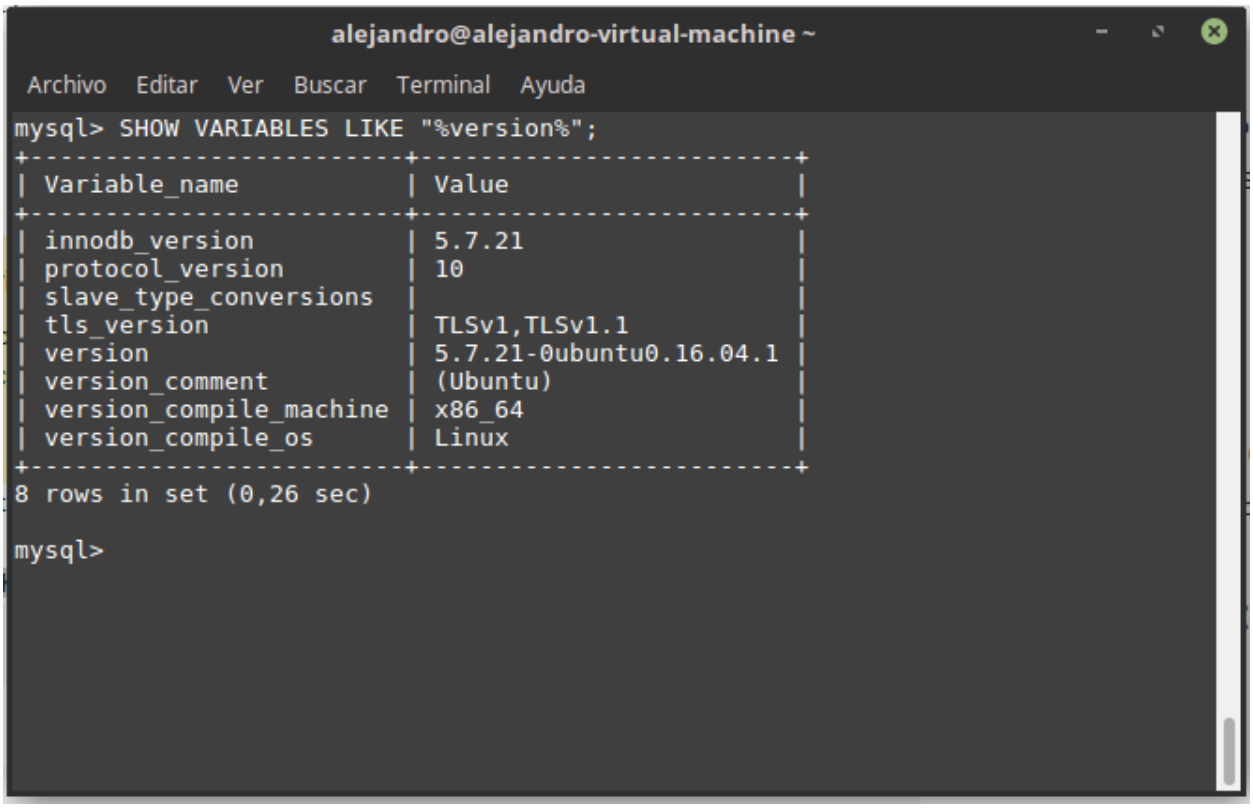

Figura 8 Versión mysql

2.2.3 Servidor

2.2.3.1 Yii

El servidor estará alojado en un sistema linux. Para ello habrá que implementar el servidor **apache** (si no lo tiene ya) y **curl**, ya que serán necesarios para la herramienta que se utilizarán como servidor web.

El framework que se usará para trabajar en el servidor web será Yii. Yii está orientado a objetos, de alto rendimiento basado en componentes, PHP y framework para aplicaciones web.

Entre las funciones de Yii que contiene, destacamos:

- Patrón de diseño Modelo Vista Controlador (MVC) [5].
- Database Access Objects (DAO), constructor de consultas query, Active Record y migración de base de datos.
- Entradas de formulario y validación del mismo.
- Soporte de autenticación incorporado.
- El manejo de errors y logging. Los errores son manejados y personalizados, y los log de los mensajes pueden ser categorizados, filtrados y movidos a diferentes destinos.
- Las medidas de seguridad incluyen la prevención cross-site scripting (XSS), prevención cross-site request forgery (CSRF), prevención de la manipulación de cookies, etc.
- Generación automática de código para el esqueleto de la aplicación, aplicaciones CRUD, etc.
- Cuidadosamente diseñado para trabajar bien con código de terceros.
- Monitorización en tiempo real de los recursos del servidor.

Para el apartado de notificaciones como hemos dicho arriba, será necesario instalar **curl** en el servidor.

Para este proyecto se usará la version de Yii 2.0.

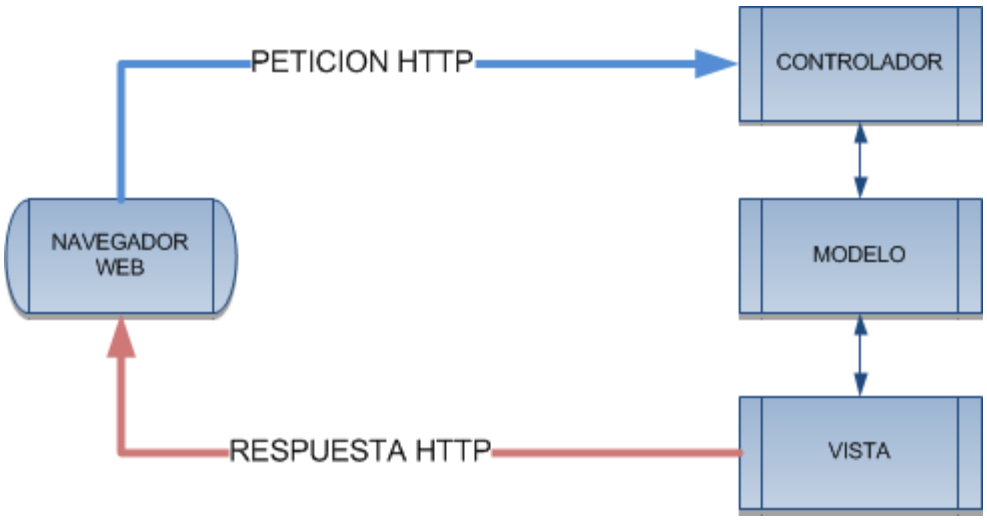

Figura 9 Ilustración modelo vista controlador

2.2.3.2 Firebase Cloud Messaging

Firebase [6] es un servicio de Google que permite la mensajería multiplataforma que permite enviar mensajes de forma segura y gratuita.

Con FCM se puede notificar a una app que tiene datos disponibles.

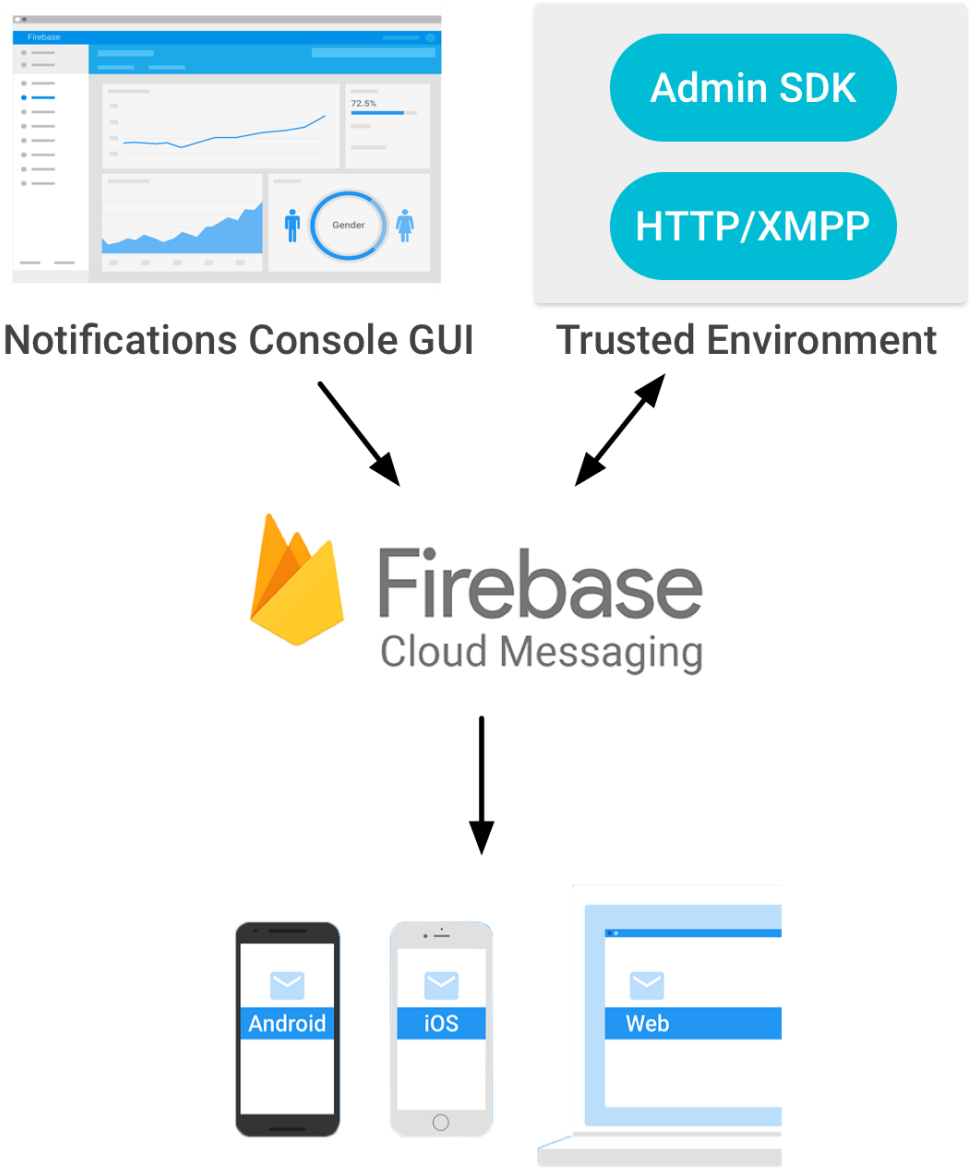

Figura 10 Funcionamiento Firebase

Una implementación de FCM incluirá dos componentes principales para enviar y recibir datos: [7]

- Un entorno de confianza como Cloud Functions para Firebase o un servidor de apps para generar, orientar y enviar mensajes.
- Una app cliente de iOs, Android o Web que reciba mensajes.

Para enviar mensajes también es posible usar el compositor de Notifications.

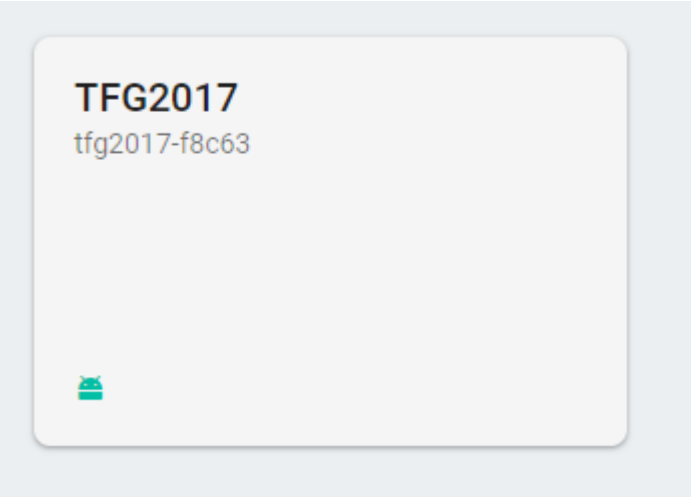

Figura 11Proyecto Firebase

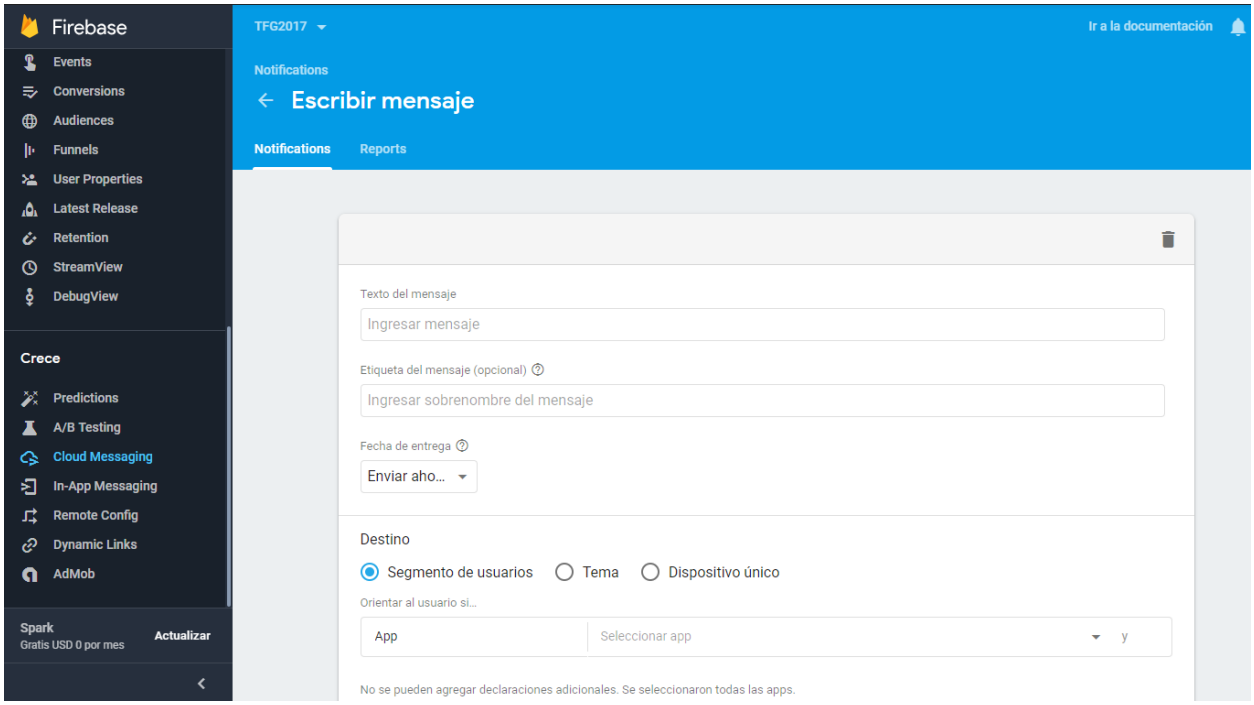

Figura 12 Panel de control Firebase web

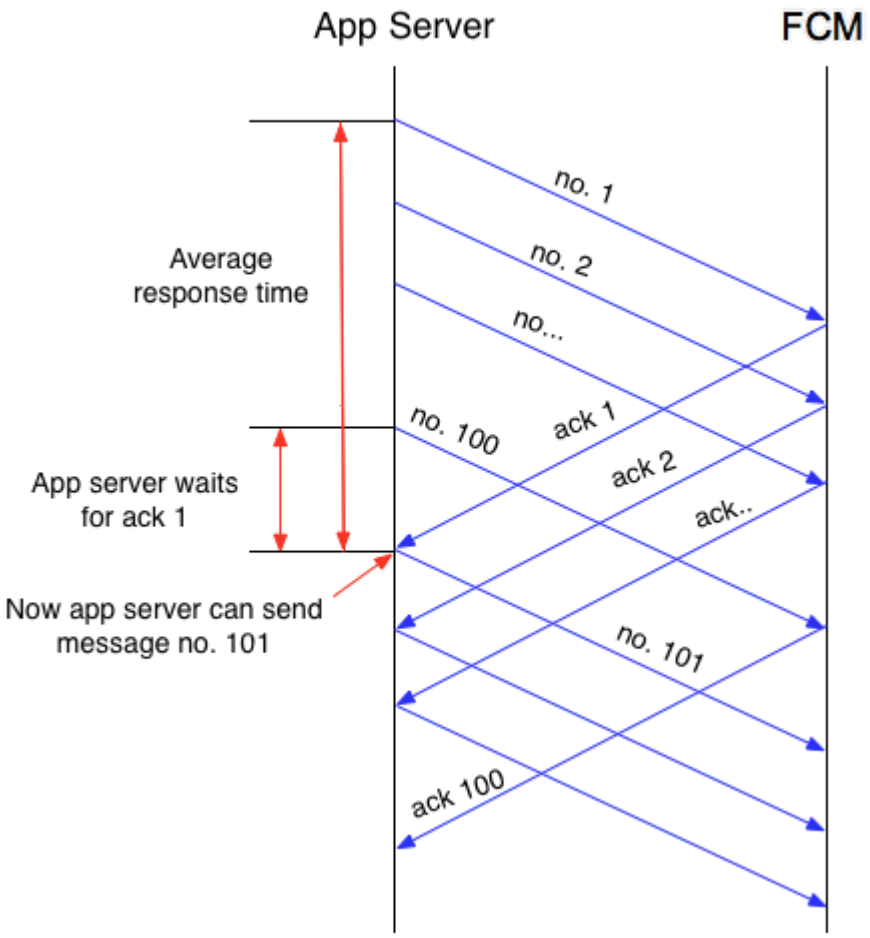

Figura 13 Paso de mensajes cliente servidor FCM

2.2.4 Aplicación móvil

Para la aplicación móvil se desarrollará en Android Studio [8]. Android Studio es un entorno integrado oficial para Android.

Entre las principales caracteristicas encontramos:

• Renderizado en tiempo real

• Consola de desarrollo

• Herramienta Lint para detectar problemas de rendimiento, usabilidad, compatiblidad de versiones y otros problemas.

• Dispositivo virtual de Android para ejecutar y probar aplicaciones.

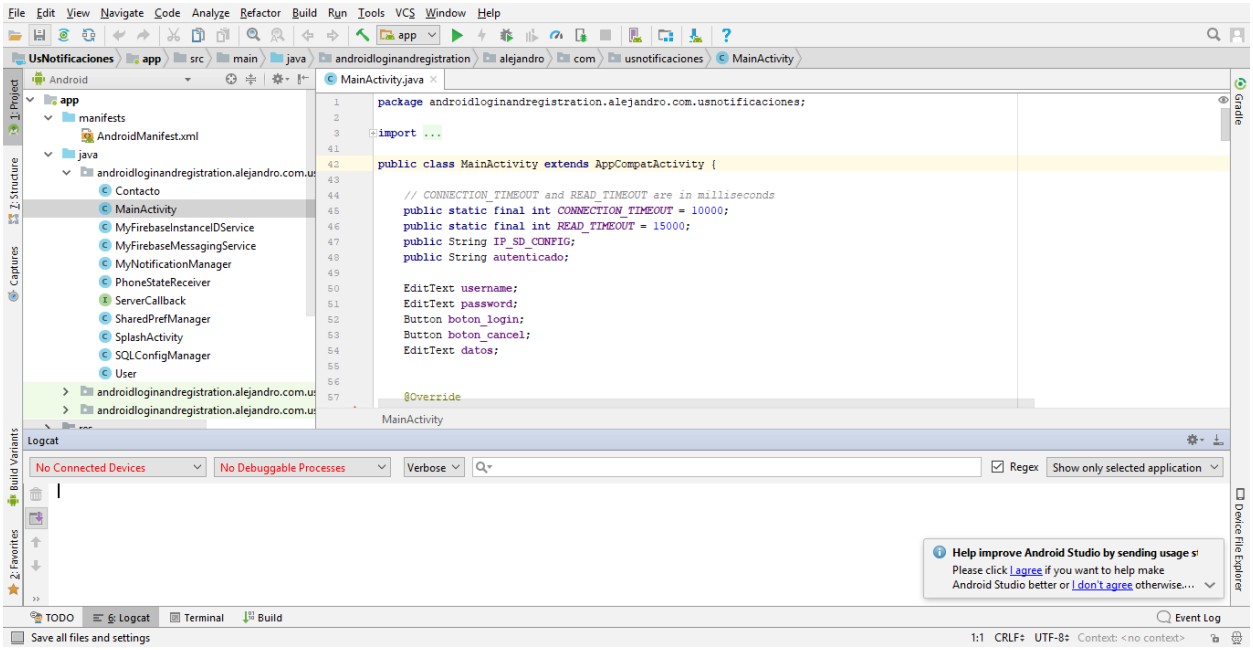

Figura 14 Android Studio

Figura 15 Máquina virtual Android Studio

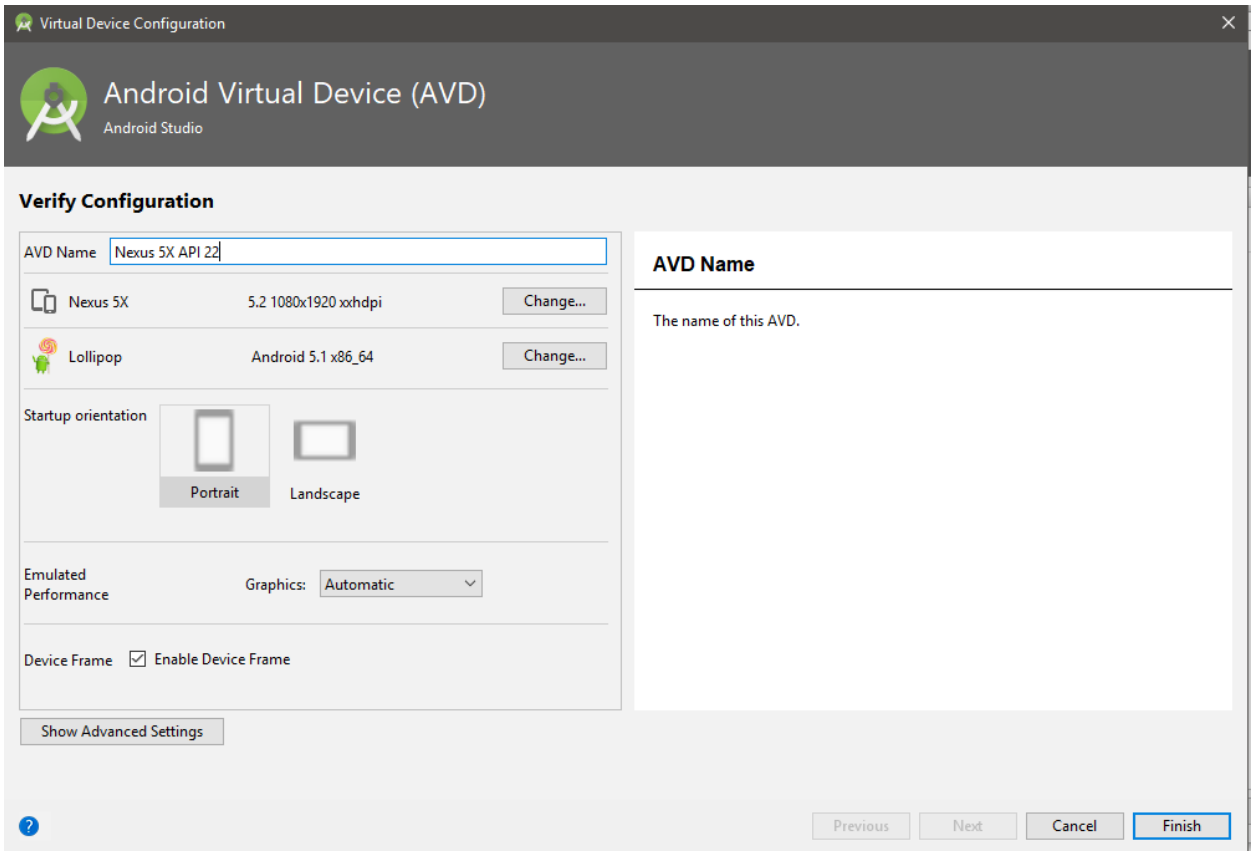

Figura 16 Características AVD

Resta decir que la aplicación siempre se desarrollará en la ultima plataforma existente y siempre actualizada.

BLOQUE 3. DESARROLLO

En este capítulo se describirá y detallará el diseño de los diferentes componentes del proyecto. Para ello, se acompañarán las descripciones con diagramas, casos de usos y algún gráfico para facilitar su explicación.

El ecosistema de trabajo será en una máquina virtual de WMware donde se pondrá en el siguiente apartado sus características, así como la de la del sistema móvil simulado por Android Studio.

3.1 VMware

La máquina virtual que simulará el servidor, será como se ha nombrado anteriormente, un escenario de simulación que intentará amoldarse al escenario real.

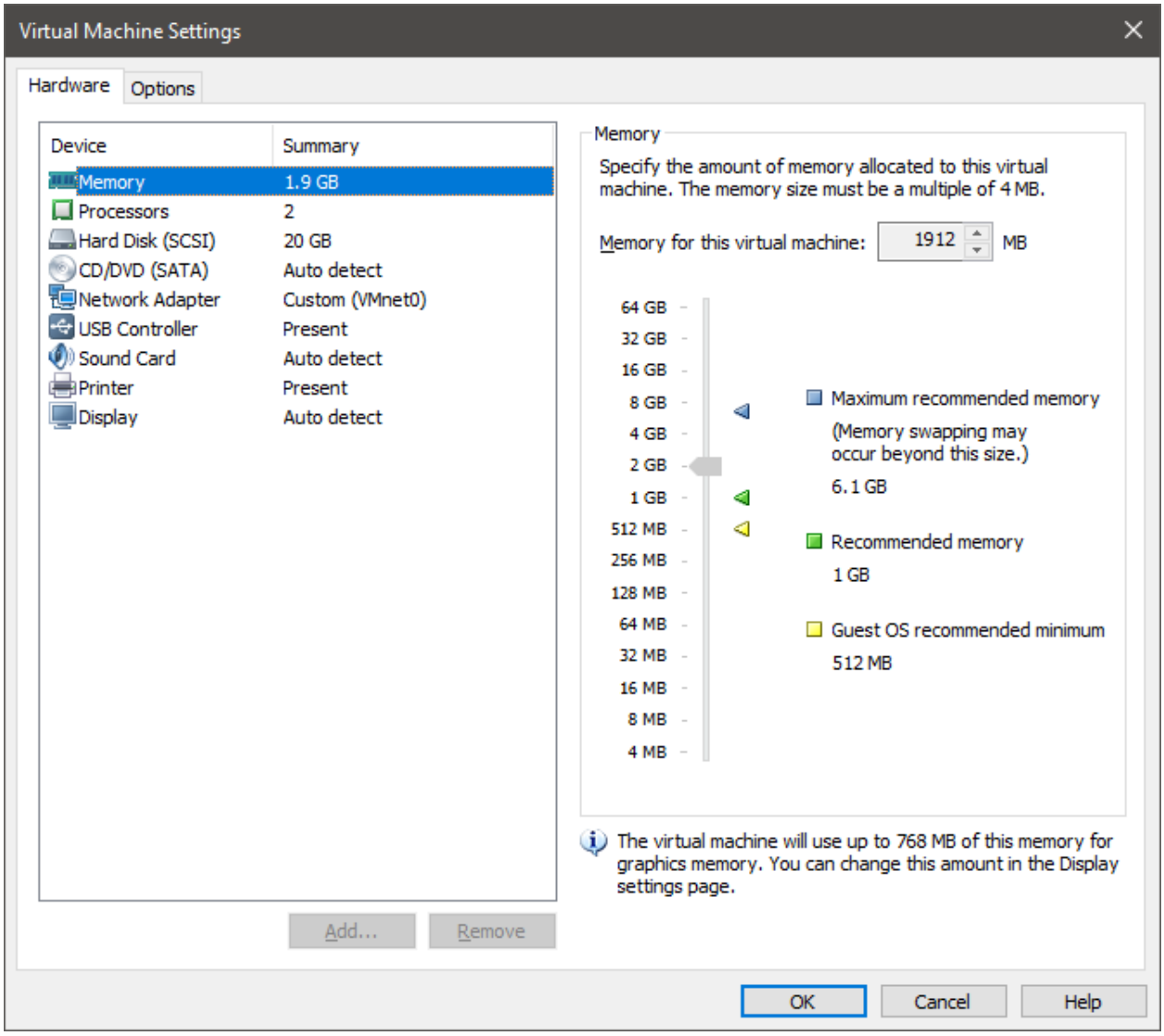

Figura 17 Características servidor VMware

Se comprobarán diferentes configuraciones, de núcleos de procesador y memoria RAM, entre las cuales se apreciará:

- **Memoriam RAM:** Se variará en rangos desde 512mb a 2048mb. Se ha observado que para el uso del servicio web desde la aplicación móvil con 512 mb son suficientes. Sin embargo, si se quiere acceder al panel de control del adminitrador vía navegador, se recomienda subir la memoria RAM hasta 1024 mb. Esto es porque el navegador puede consumir recursos de más y esta configuración evita la ralentizaciones con el sistema.

- **Núcleos del procesador:** en cuanto a los núcleos tampoco ha hecho falta probar muchas configuraciones. Se puede variar entre 1 o 2 núcleos pero se recomienda este último en la configuración.

- **Disco duro:** se ha dejado por defecto la configuración de 20 gb recomendada por la máquina virtual.

3.2 Base de datos

La base de datos se diseñará en mysql como se ha explicado anteriormente, por facilidades a la hora de implementarlo tanto en el servidor [9] [10] como en la aplicación móvil.

La base de datos tendrá de nombre "Agenda" y contendrá 3 tablas (una por cada gestión): [11]

- **Contactos**: que tendrá los campos *Nombre* (String (20)), *Apellidos* (String (20)), *Teléfono* (String (9)) y *Token* (String).
- **Login**: estará formada por *user* (String (20)) y *passwd* (String (20)) la cuál debera ser de mínimo 6 caracteres y será cifrada con md5 por lo que su valor real no será mostrado.
- **Admin**: formada por por *username* (String (20)) y *password* (String (20)). Ya que el administrador es el único que tendrá acceso y Yii, esta contraseña no estará cifrada.

El diseño que se realizará se detalla a continuación con una imagen. Estará creada con el software DBDesigner 4 [12] de la asignatura Base de datos. Esta aplicación nos dará una vez diseñado, los scripts para crear, optimizar y eliminar las diferentes tablas, que se detallarán en los anexos.

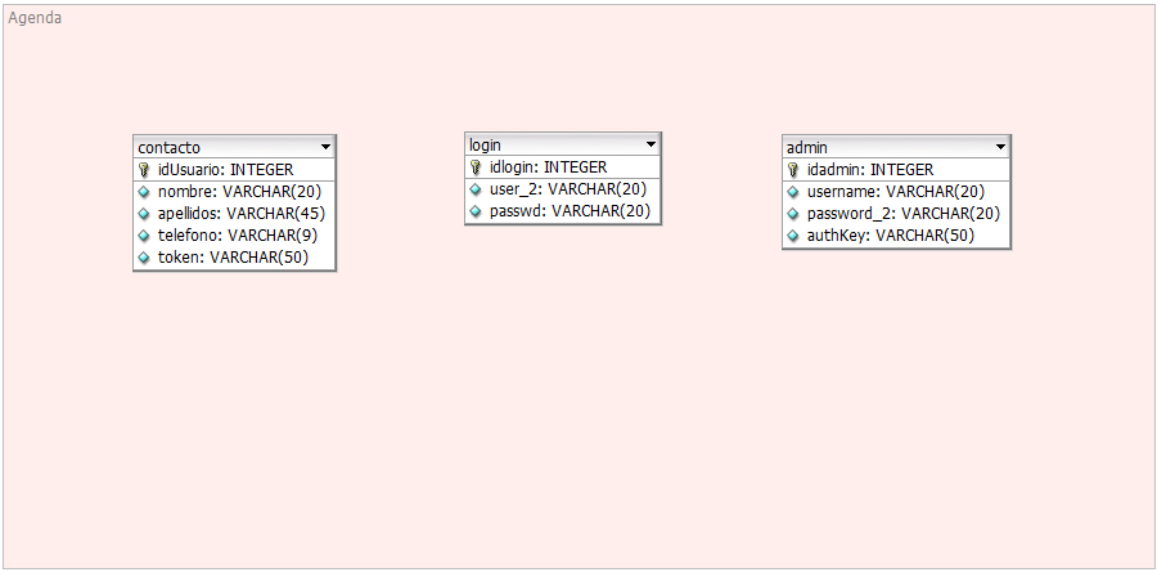

Figura 18 Esquema base de datos.

Y su implementación en el servidor.

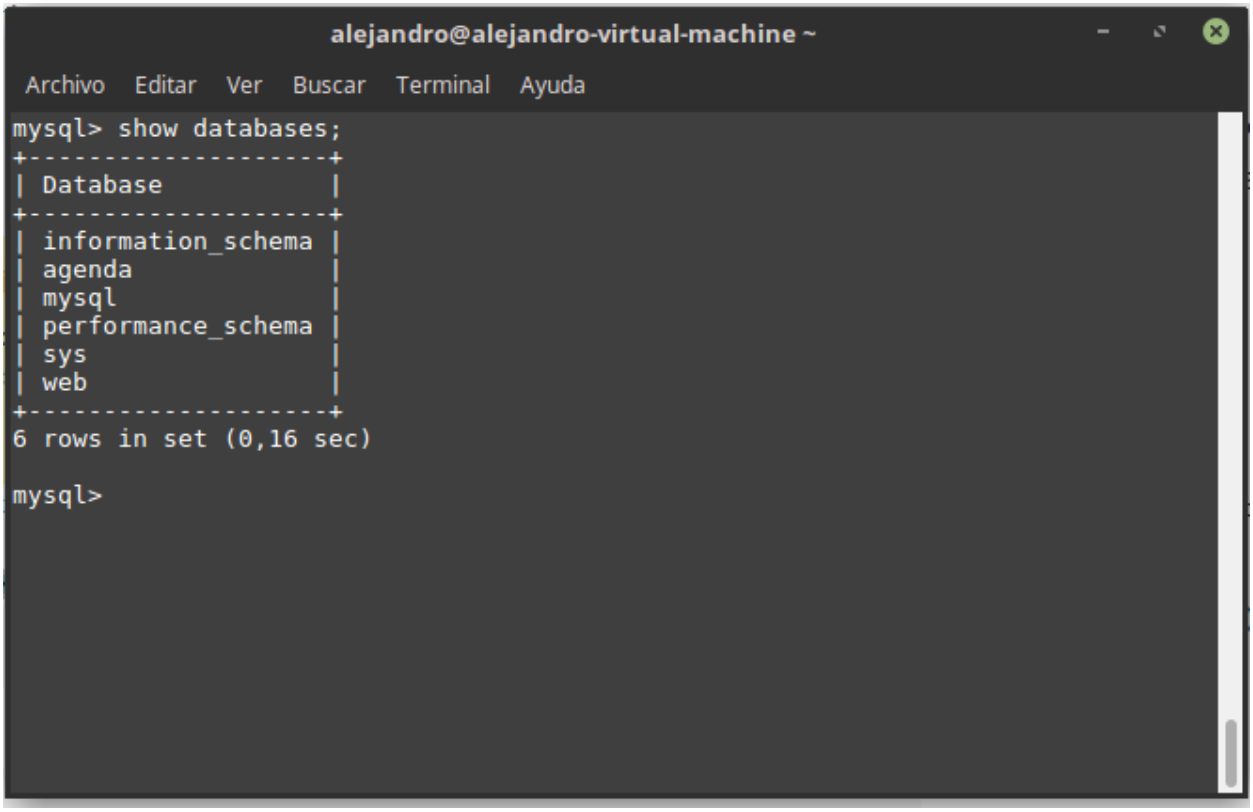

Figura 19 Base de datos en el servidor

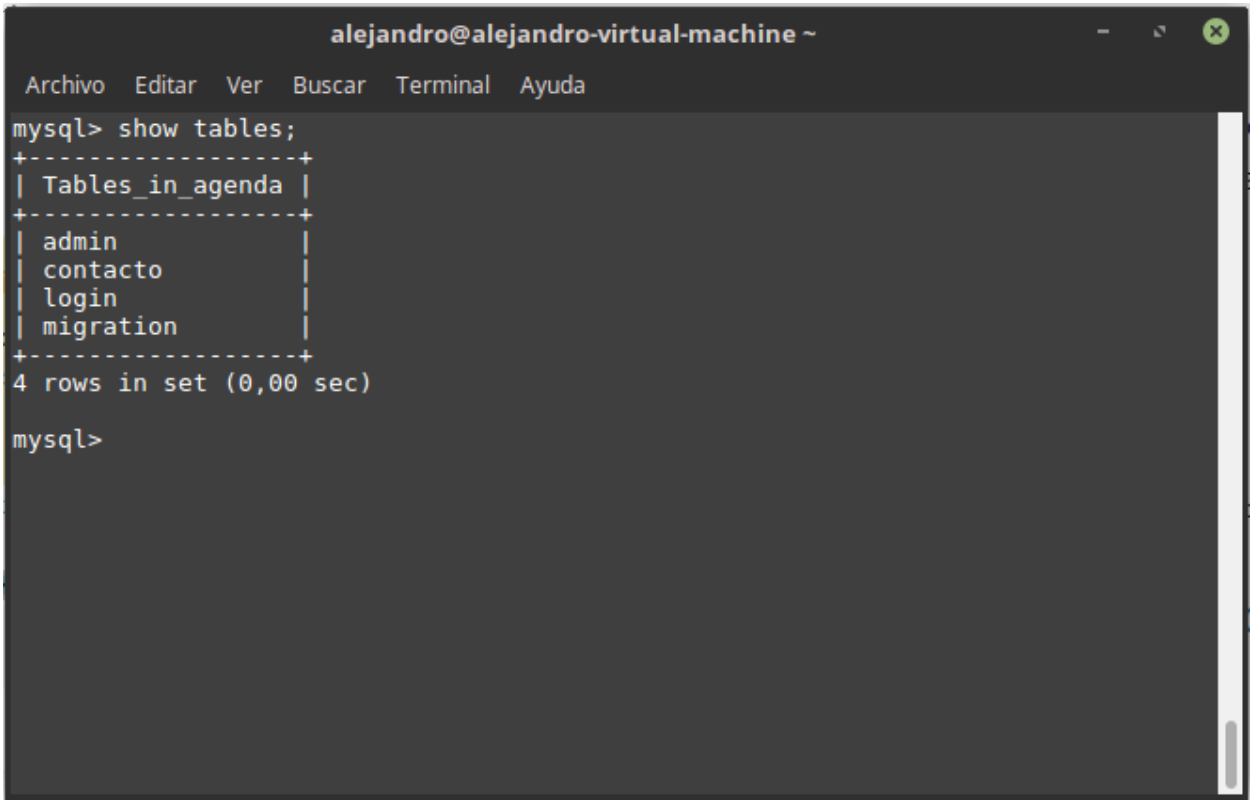

Figura 20 Tablas

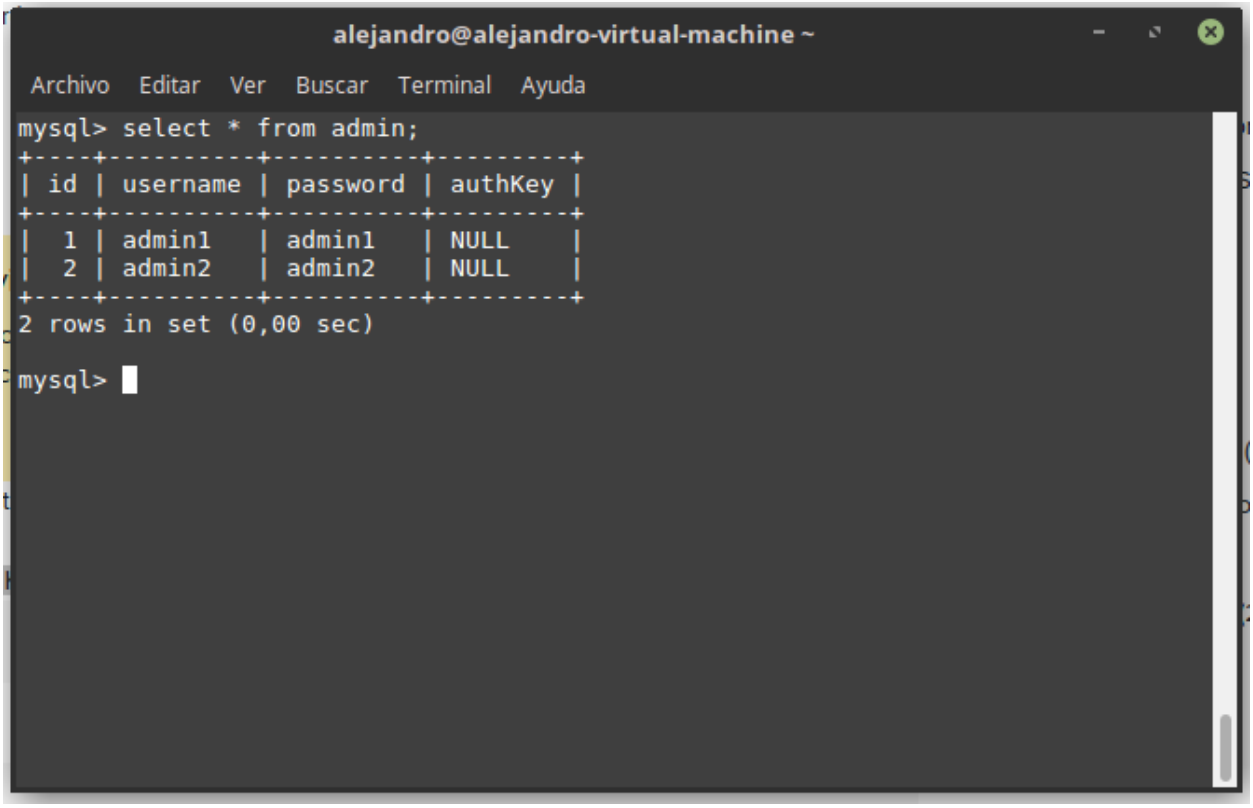

Figura 21 Tabla admin

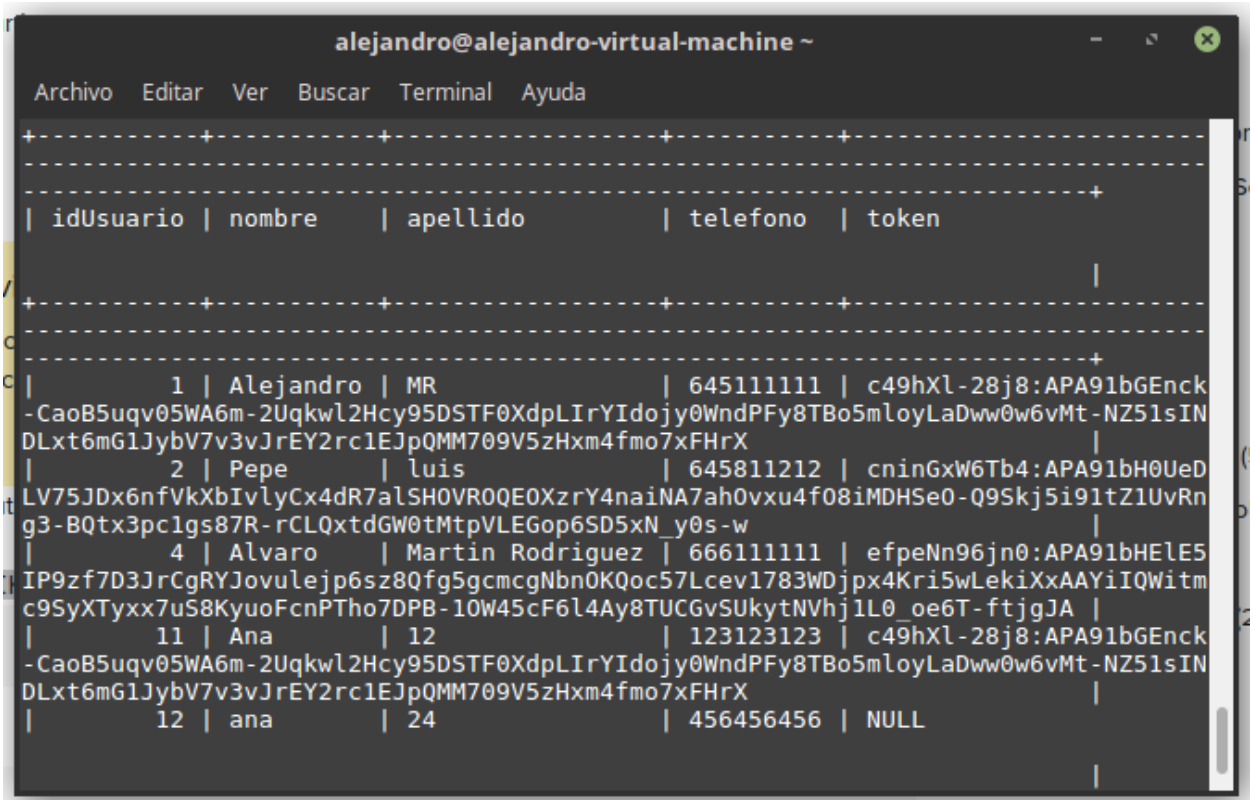

Figura 22 Tabla contactos

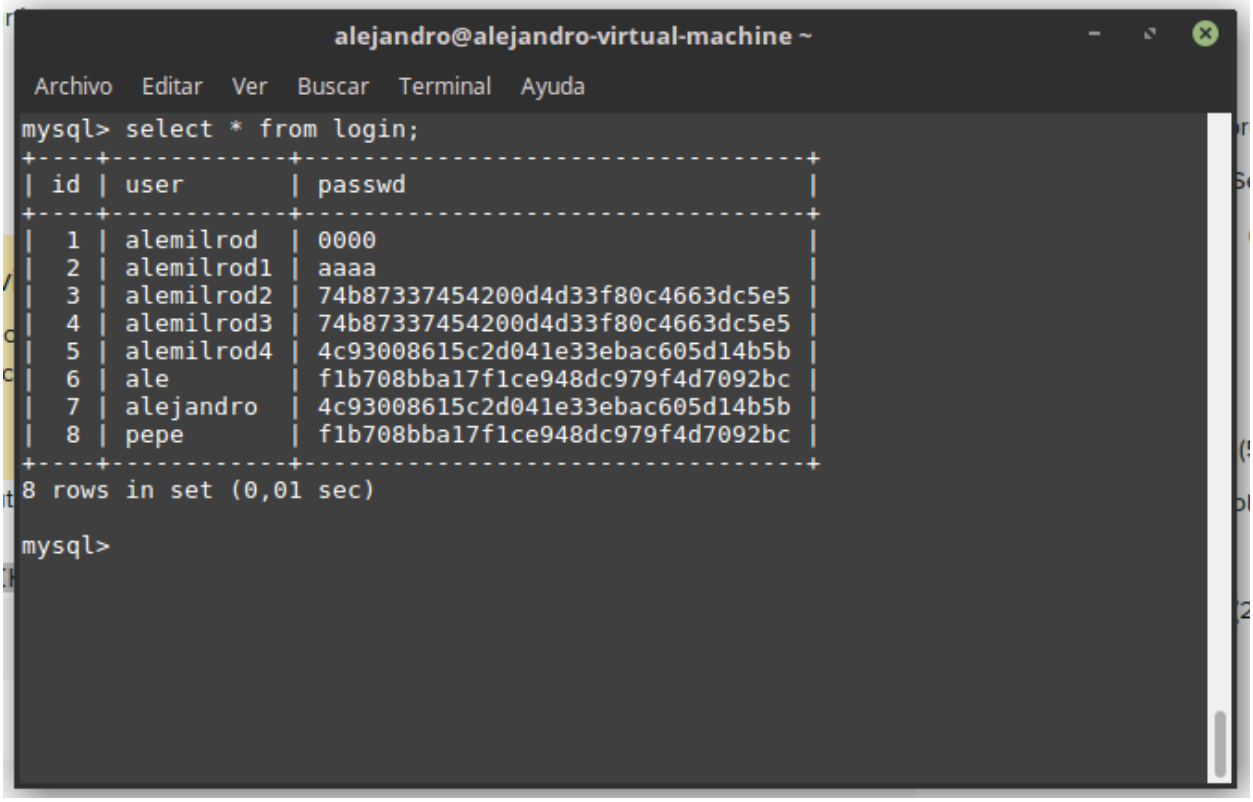

Figura 23 Tabla login

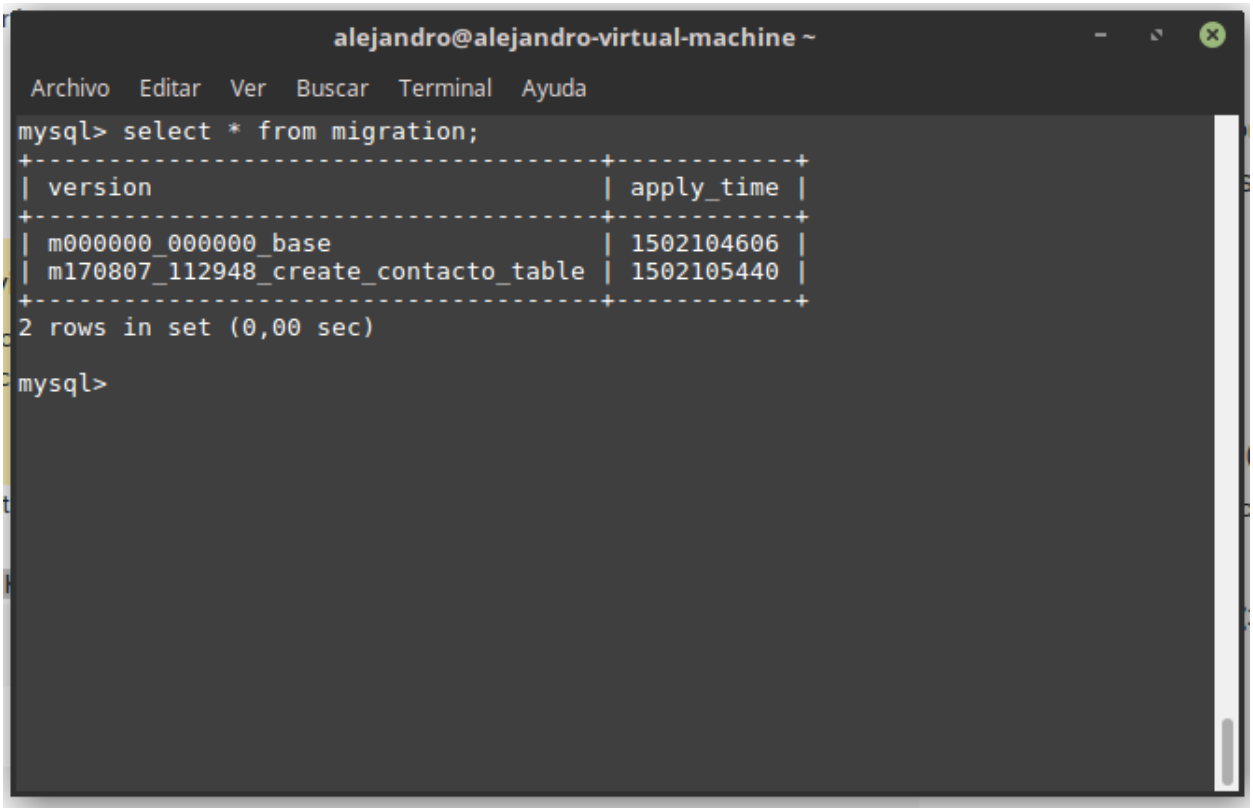

Figura 24 Tabla migration

3.3 Servidor

El servicio de apache ya vendrá instalado así que solo habrá que cambiar de la configuración de la carpeta web Apache a exportar e iniciar / reiniciar el servicio. La carpeta (tfg) se colocará en carpeta personal.

El servidor contendrá la capacidad de:

1 **Autenticación del administrador**: si no estuviera autenticado solo se permitirá la vista de la página por defecto, así como el contacto con el adminitrador o el apartado "sobre" de la página. No podrá acceder al panel del administrador.

2 **Registrar token del teléfono**: se enviará un token y un teléfono asignado y se comprobará que el teléfono existe en la tabla antes de añadirlo.

3 **Consultar llamadas y devolver el nombre y apellido del contacto**: Para ello desde la aplicación móvil se enviará el token el teléfono y se comprobará que exista. De ser así se devolverá el resultado de la búsqueda.

4 **Sincronizar la base de datos local con la del móvil**: Se recibirá un token y se comprobará que existe, y devolverá primero si se acepta o deniega la petición y si se hace, devolverá la tabla de "Contactos" sin el campo token.

Además el panel del administrador contará con las opciones de:

1 **Añadir un contacto**: Necesitará completar todos sus campos y además cumplir una serie de requisitos para añadirlos (nombre, apellidos y un número de teléfono válido de 9 dígitos).

2 **Dar de alta un usuario**: De cara a la aplicación móvil. Necesitará completar todos sus campos y además deberá cumplir una serie de requisitos para añadirlos (nombre de usuario y contraseña de mínimo 6 caracteres alfanuméricos).

3 **Consultar usuarios**: Mostrará la tabla de los contactos y desde ahí se podrá gestionar (añadir, borrar, editar).

4 **Enviar notificaciones**: Página donde a partir del token se podrá enviar un mensaje (con imagen y url) a un destinatario. Se mostrará además la estructura del mensaje enviado y la respuesta del mismo.

Lo primero a configurar será una cookie aleatoria necesaria en el fichero **/config/web.php** para hacer funcionar el servidor.

Una vez realizado, se incluirá la opción **urlManager** que permitirá tener direcciones url más limpias y accesibles a la hora de acceder al servidor.

Finalmente, se añadirá la tabla con la que trabajará Yii por defecto, para poder crear los formularios y las funciones CRUD.

Este fichero se encuentra en ***/migrations/m170807_112948_create_contacto_table.***

```php
class m170807_112948_create_contacto_table extends Migration
{
    /**
     * @inheritdoc
     */
    public function up()
    {
        $this->createTable('contacto', [
            'idUsuario' => $this->primaryKey(),
            'nombre' => $this->string()->notNull(),
            'apellido' => $this->string()->notNull(),
            'telefono' => $this->string()->notNull(),
            'token' => $this->string()

        ]);
    }
```

Figura 25 Función migration

Se configurará también el fichero ***/config/db.php*** y ***/config/db2.php*** que contendrán la configuración de la base de datos con la que conectarán y el usuario y contraseña de **root**.

A continuación se detallará cada una de las funciones:

3.3.1 Autenticación del administrador

Para autenticarse a partir de la tabla creada en la base de datos, habrá que configurar el fichero ***/models/User.php***.

Se añadirá unas nuevas funciones a la plantilla de Yii que consultarán en la base de datos, la tabla **admin** y devolverán el resultado de la comprobación.

```php
public static function findIdentity($id){
  return static::findOne($id);
}

public static function findIdentityByAccessToken($token, $type = null){
  throw new NotSupportedException();
}

public function getId(){
  return $this->id;
}

public function getAuthKey(){
  return $this->authKey;
}

public function validateAuthKey($authKey){
  return $this->authKey === $authKey;
}
public static function findByUsername($username){
  return self::findOne(['username'=>$username]);
}

public function validatePassword($password){
  return $this->password === $password;
}
```

Figura 26 Funciones User.php

Se protegerá el panel del administrador de tal manera que si se intetara acceder anónimamente, Yii devolverá la vista por defecto de sin autenticación.

Para ello se modificará la función actionIndex donde se realizará previamente una comprobación de la variable **user**. Si valiera null significará que el usuario no se ha podido indentificar.

```php
public function actionIndex()
{
// necesitamos que el usuario se autentique. Para poder autenticarse tendra que superar la autentificacion frente a la base de datos. mientras
// el usuario valdra null. Asi que no pararemos de mostar la misma vista
    if(($identity = Yii::$app->user->identity) != null)
    return $this->render('index');
    else
    return $this->render('index2');
}
```

Figura 27 Función Index servidor

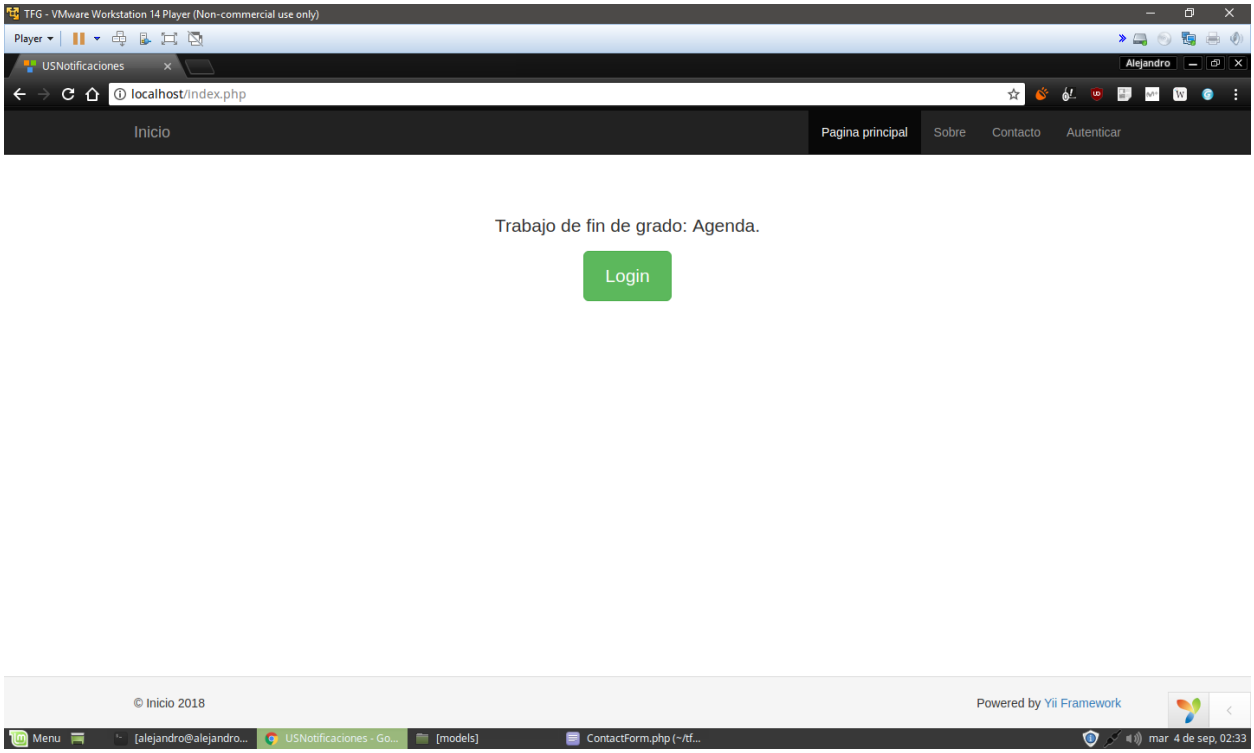

Figura 28 Servidor sin autenticar

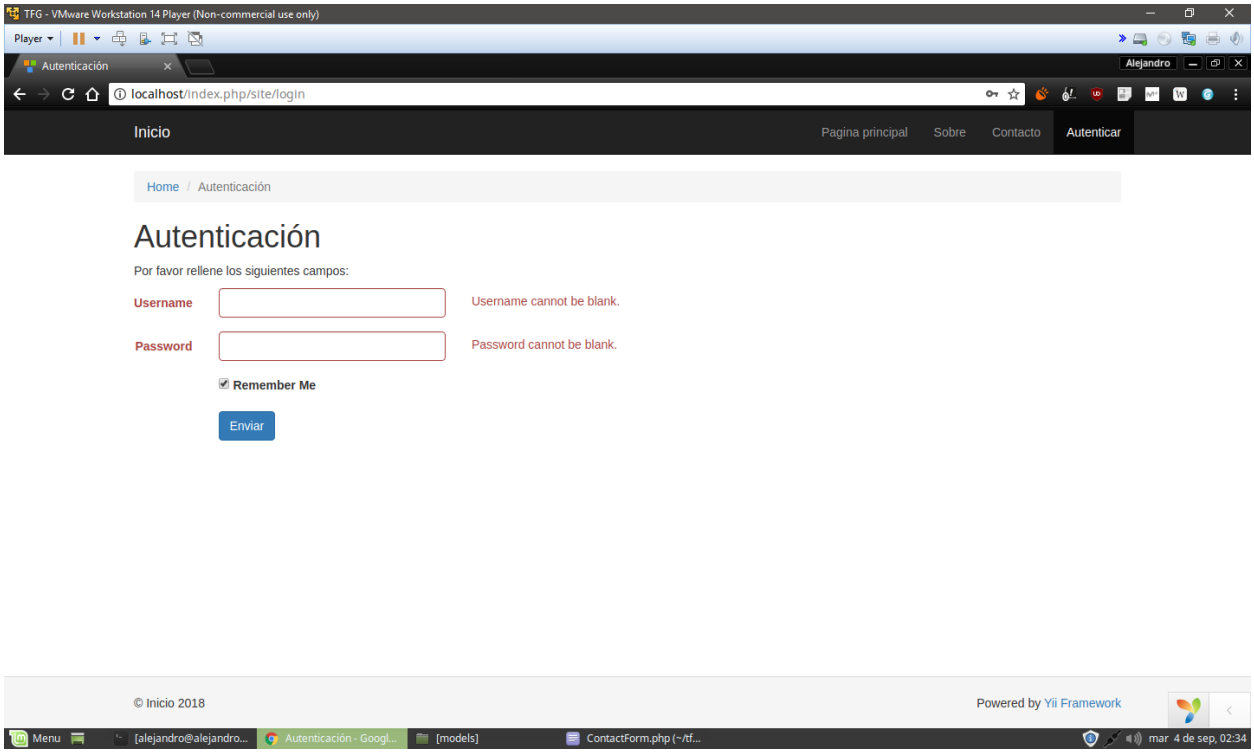

Figura 29 Servidor pantalla autenticación

3.3.2 Registrar token del teléfono

Este servicio se encargará de enlazar el token que dará Firebase a un determinado terminal a la base de datos con su número correspondiente. Este token será necesario para poder realizar las consultas siguientes.

Para poder registrar el token de un teléfono en particular, se comprobará el método de la petición. La petición se realizará en POST pero implementará tambien el método GET o PUT.

```php
$request = Yii::$app->request;
$result="false";

if ($request->isGet)  {
  $numero = $request->get('numero');
  $token = $request->get('token');
}

if ($request->isPost) {
  $post = $request->post();
  $numero = $request->post('numero');
  $token = $request->post('token');
}

if ($numero == NULL && $token == NULL){
  $params = $request->bodyParams;
  $numero = $request->getBodyParam('numero');
  $token = $request->getBodyParam('token');
}
```

Figura 30 Función registrar token 1

Se comprobará que el número exista en la base de datos y que se podrá almacenar su valor en la tabla.

El controlador devolverá un resultado dependiendo de si se ha encontrado o no el número y si se ha podido guardar el token.

Para ello el controlador **TokenController.php** realizará una petición sql a la base de datos preguntando por la tabla **Contactos**. Posteriormente, se hará un bucle recorriendo las entradas y se buscará aquella con la que coincida el número de teléfono.

```php
if($numero != null && $token != null){

  $sql = "SELECT * FROM contacto WHERE ((telefono = $numero))";
  $model = \app\models\Agenda::findBySql($sql)->all();   //modelo donde se encuentra la clase Login que devuelve la tabla


  foreach($model as $contacto)
  {
      //recorremos la tabla buscando que usuario y numero coincidan. Si es asi ponemos a 1 y validamos.
      $result=true;
      $contacto->token = NULL;
      $contacto->token = $token;
      $contacto->save();
  }
}
echo $result;
```

Figura 31 Función registrar token 2

3.3.3 Consultar llamadas y devolver el nombre y apellido del contacto

El servidor tendrá un servicio que responderá cuando la aplicación móvil reciba una llamada entrante. Se le realizará una petición donde se enviará el número de teléfono.

El controlador encargado responderá con el nombre y apellidos del teléfono encontrado.

La petición se realizará con el método POST.

```
if ($numero != NULL ){
  $contactos = \app\models\Agenda::find()->all(); //buscamos en la tabla
  foreach($contactos as $contacto)
  {
    if($contacto->telefono == $numero){ //recorremos cada contacto buscando el numero
    //imprimimos el valor encontrado para probar que funciona
    echo $contacto->nombre . " " . $contacto->apellido;
```

Figura 32 Función consultar llamadas

Se buscará mediante una petición sql a la table correspondiente, si coincide el número con algun contacto, y de ser así, devolverá el nombre y los apellidos.

Para ello se utilizará el controlador **ConsultaController.php**.

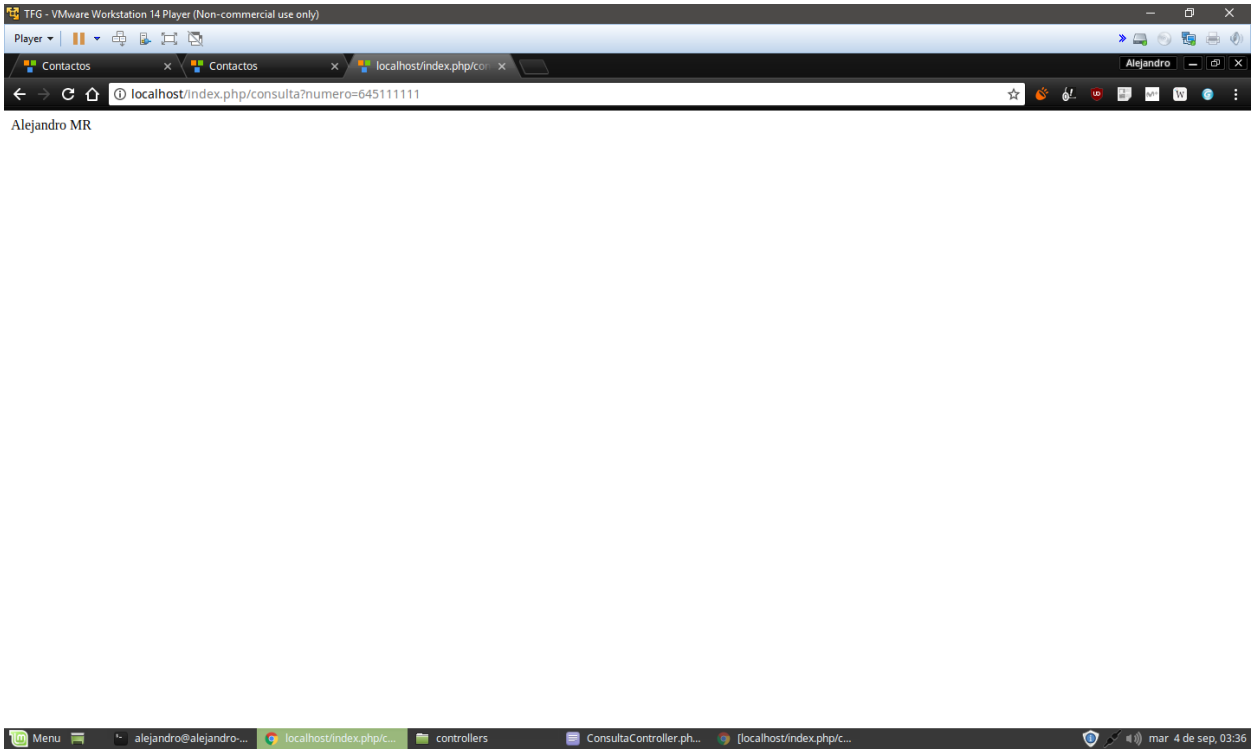

Figura 33 Petición get consultar llamada

3.3.4 Sincronizar la base de datos local con la del móvil

El servidor devolverá nombre, apellidos, y teléfono de cada uno de los contactos de la base de datos local. Para ello el servidor recibirá un token que comprobará de su existencia y si así fuera responderá la petición.

El mensaje de respuesta se comprondrá con una respuesta de correcto o incorrecto y si fuera correcto los campos anteriormente mencionados. Este trabajo lo realizará el controlador **SincronizarController.php**. Separará el mensaje de respuesta (nombre, apellidos, teléfono, salto de línea) de OK o KO en otra línea.

```php
if($token != null){

  $sql = "SELECT * FROM contacto WHERE ((token = '$token'))";
  $model = \app\models\Agenda::findBySql($sql)->all();   //modelo donde se encuentra la clase Login que devuelve la tabla

  foreach($model as $consulta)
  {
    if($consulta != null){
       $result=true;
    }
  }
}
echo $result . "<br>";

if($result=== true){
  $sql = "SELECT * FROM contacto";
  $model = \app\models\Agenda::findBySql($sql)->all();
  foreach($model as $contacto)
  {
    echo $contacto->nombre . "<br>";
    echo $contacto->apellido . "<br>";
    echo $contacto->telefono . "<br>";
  }
}
```

Figura 34 Función sincronizar base de datos

3.4 Panel de control del administrador

El panel de control o vista que mostrará Yii una vez la autenticación sea correcta será el siguiente, la vista **index.php**

Esta vista contará con 4 botones que harán referencia a un servicio diferente cada uno y evitará tener que acceder de forma manual.

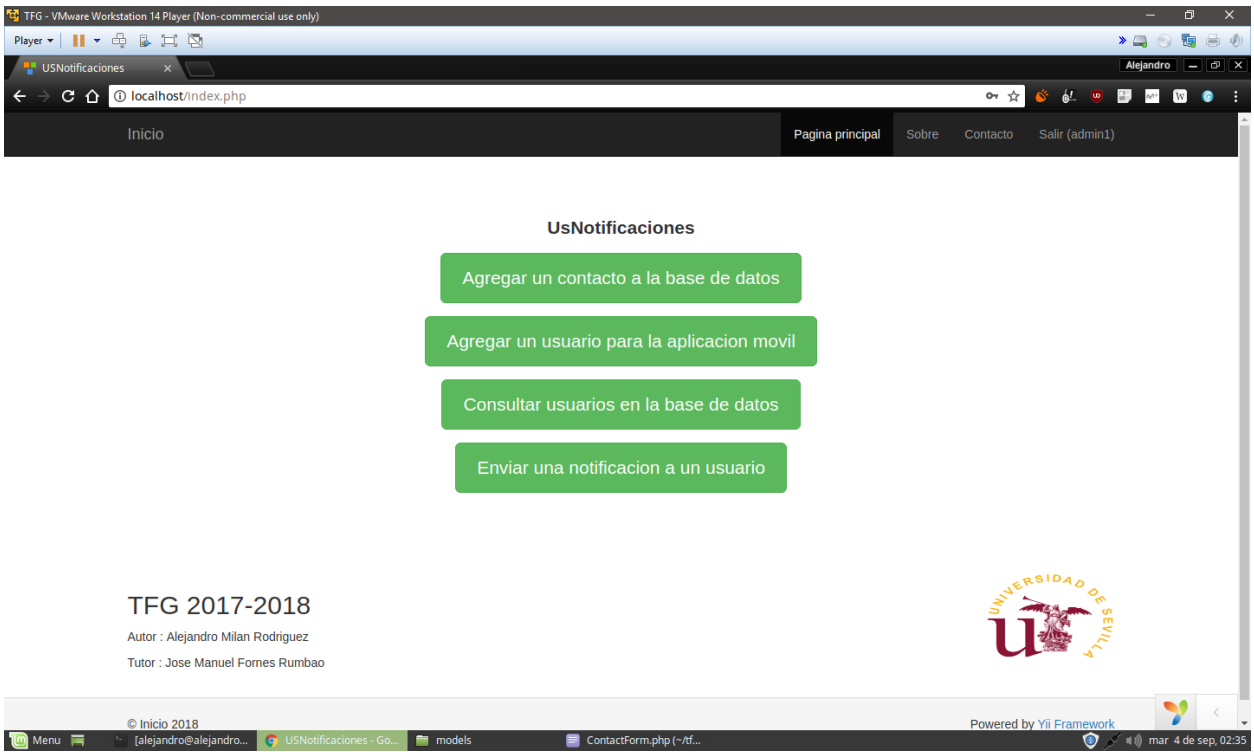

Figura 35 Panel de control administrador

3.4.1 Añadir un contacto

Añadir un contacto permitirá al administrador completando los campos correspondientes y cumpliendo los requisitos, añadir un nuevo contacto completo a la base de datos. Para ello **AgendaController.php** devolverá el modelo **Agenda.php** y renderizará la vista correspondiente del formulario.

Este modelo comprobará que se cumple los requisitos anteriormente mencionados.

```php
/**
 * Creates a new Agenda model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new Agenda();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->idUsuario]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}
```

Figura 36 Función añadir contacto 1

```php
public static function tableName()
{
    return "contacto";
}

public function rules()     //para validar el formulario
{
    return [
        [["nombre", "apellido", "telefono"], "required"],   //exigimos todos los campos
        [["nombre"], "string"],
        [["apellido"], "string"],
        [["telefono"], "string", "max" => 9, "min" => 9],       //maximo numero 9 digitos
    ];
}

public function attributeLabels()
{

    return[
        "id" => "Id",
        "nombre" => "Nombre",
        "apellido" => "Apellidos",
        "telefono" => "Telefono"
        ];

}
```

Figura 37 Función añadir contacto 2

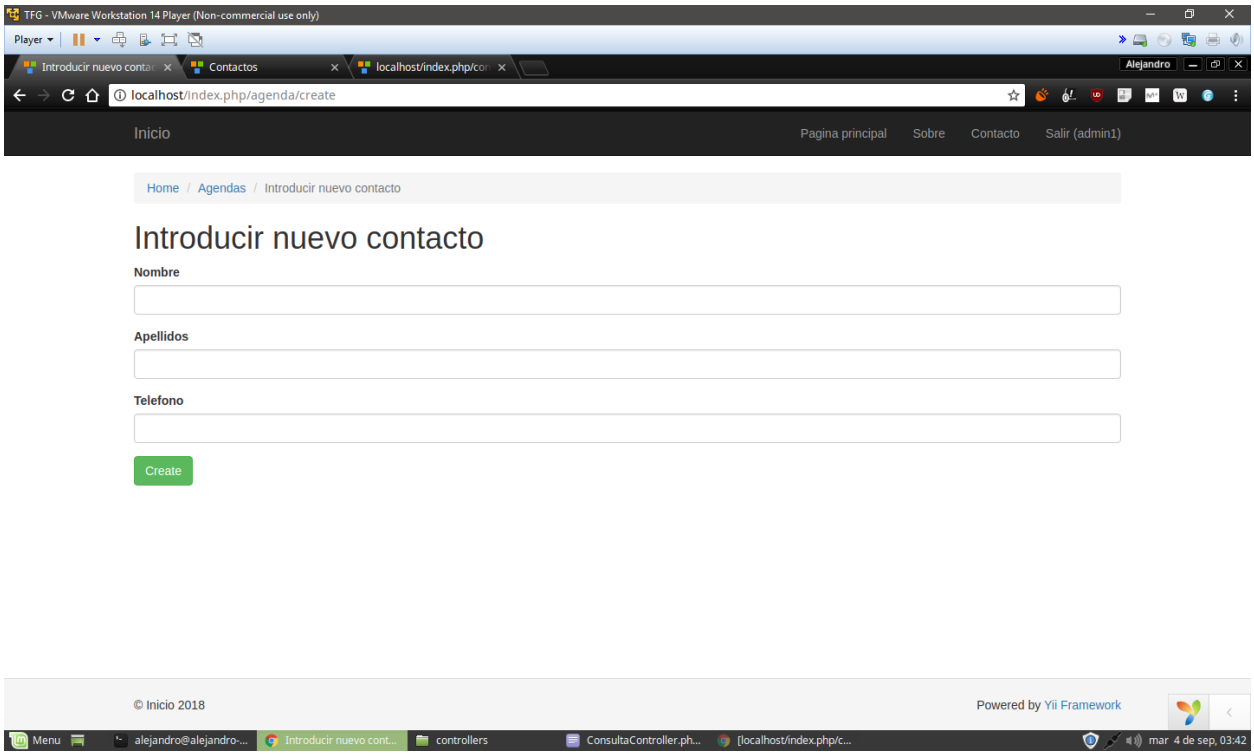

Figura 38 Añadir contacto nuevo

Se comprobará que los campos no estarán vacios y además que el teléfono contendrá 9 dígitos.

3.4.2 Dar de alta a un usuario

Esta función permitirá al administrador añadir un usuario nuevo a la tabla **Login**. Recordaremos que esa tabla corresponderá con la autenticación desde la aplicación móvil. El controlador que nos facilitará la vista de este formulario será **LoginController.php**. [13]

Recordemos que el controlador no devolverá ninguna vista si el usuario no se ha autenticado antes en la tabla de admin.

Se comprobará una serie de requisitos para introducirlo como que no estarán en blanco o que la longitud de la contraseña será mayor de 6 caracteres. Se hará una conexión con la base de datos si los datos no contienen errores, y se almacenarán, evitando al administrador tener que añadir los usuarios a través del programa mysql. La contraseña se codificará usando la función md5 que contiene php.

```php
// registro de usuario
if (isset($_POST['reg_user'])) {
  $username = mysqli_real_escape_string($db, $_POST['username']);
  $password_1 = mysqli_real_escape_string($db, $_POST['password_1']);
  $password_2 = mysqli_real_escape_string($db, $_POST['password_2']);

  // validamos el formulario
  if (empty($username)) { array_push($errors, "Un nombre de usuario es necesario"); }
  if (empty($password_1)) { array_push($errors, "Una contraseña es necesaria"); }
  if ($password_1 != $password_2) {
  array_push($errors, "Las dos contraseñas no coinciden");
  }
  if (strlen($password_1) < 6){
    array_push($errors, "La contraseña tiene que tener 6 caracteres minimo");
  }

  // Comprobamos que el usuario no exista en la base de datos
  $user_check_query = "SELECT * FROM login WHERE user='$username' LIMIT 1";
  $result = mysqli_query($db, $user_check_query);
  $user = mysqli_fetch_assoc($result);

  if ($user) { // si existe
      array_push($errors, "Error. El usuario ya existe");
  }

  // Registramos al usuario si no hay errores en el formulario
  if (count($errors) == 0) {
    $password = md5($password_1); //usamos md5 para encriptar la contraseña

    $query = "INSERT INTO login (user, passwd)
          VALUES('$username', '$password')";
    mysqli_query($db, $query);
    $_SESSION['username'] = $username;
    $_SESSION['success'] = "You are now logged in";
    echo "Usuario ". $username . " correctamente registrado.";
    header('location: index.php');
  }
```

Figura 39 Funciñon añadir usuario aplicación móvil

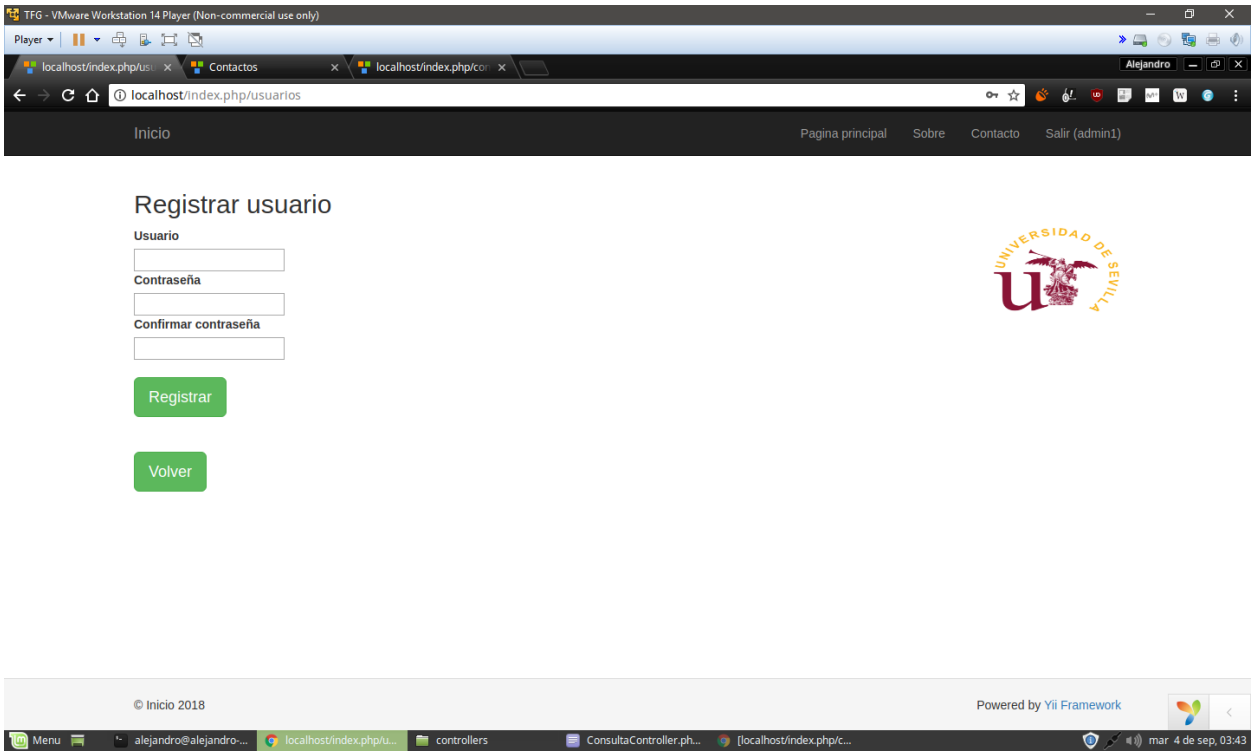

Figura 40 Añadir usuario a base de datos

3.4.3 Consultar usuarios

El administrador podrá consultar los usuarios de la tabla **Contactos** vía web sin necesidad de acceder mediante mysql. Para ello el controlador **AgendaController.php** renderizará su propia vista **index.php**.

Desde ahí el adminsitrador podrá modificar, visualizar, editar los diferentes contactos o añadir contactos nuevos a la base de datos.

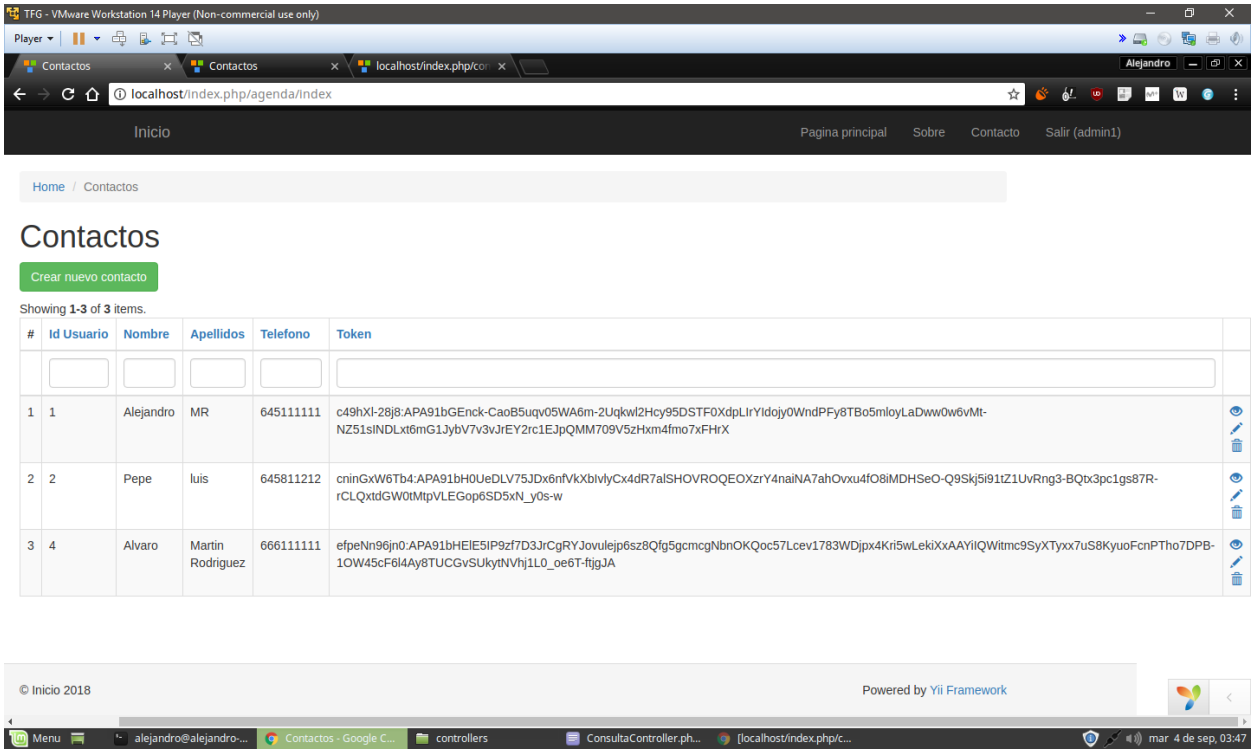

Figura 41 Consultar contactos

3.4.4 Enviar notificaciones

Para el uso de esta función, se necesitará tener instalado en el servidor curl. Curl es una biblioteca y un interprete de comandos orientado a la transferencia de archivos.

Desde este panel, el administrador podrá enviar notificaciones en forma de mensajes Json que el usuario recibirá en su aplicación. Para ello hará uso de **Notifications.php** para formar la clase. [14] [15]

Se deberá completar los campos token, titulo y mensaje. El token corresponderá al número de teléfono enlazado al que se querrá enviar.

Habrá campos opcionales también como incluir una imagen o incluir una acción. La acción a realizar será añadir una url que se podrá abrir desde el cliente móvil. La imagen reemplazará a la imagen por defecto de la aplicación. La respuesta del servidor de Firebase se recogerá y mostrará por pantalla, así como el mensaje Json creado.

```php
<?php
if(isset($_POST['title'])){

  require_once __DIR__ . '/notification.php';
  $notification = new Notification();

  $title = $_POST['title'];
  $message = isset($_POST['message'])?$_POST['message']:'';
  $imageUrl = isset($_POST['image_url'])?$_POST['image_url']:'';
  $action = isset($_POST['action'])?$_POST['action']:'';

  $actionDestination = isset($_POST['action_destination'])?$_POST['action_destination']:'';

  if($actionDestination ==''){
    $action = '';
  }
  $notification->setTitle($title);
  $notification->setMessage($message);
  $notification->setImage($imageUrl);
  $notification->setAction($action);
  $notification->setActionDestination($actionDestination);

  $firebase_token = $_POST['firebase_token'];
  $firebase_api = 'AIzaSyCz2GsFZ3UObZINzNeqWV7188r4ChntiK0';

  $requestData = $notification->getNotificatin();

  $fields = array(
      'to' => $firebase_token,
      'data' => $requestData,
  );


  // Asignamos los valores al mensaje POST
  $url = "https://fcm.googleapis.com/fcm/send";

  $headers = array(
    'Authorization: key=' . $firebase_api,
    'Content-Type: application/json'
  );

  $notification = array('title' =>$title , 'body' => $message,'sound' => 'default', 'badge' => '1');
  $data = array('title' =>$title , 'body' => $message, 'message' => $message, 'imageUrl' => $imageUrl, 'Url' => $actionDestination);
```

Figura 42 Función Firebase 1

```php
// Asignamos los valores al mensaje POST
$url = "https://fcm.googleapis.com/fcm/send";

$headers = array(
  'Authorization: key=' . $firebase_api,
  'Content-Type: application/json'
);

$notification = array('title' =>$title , 'body' => $message,'sound' => 'default', 'badge' => '1');
$data = array('title' =>$title , 'body' => $message, 'message' => $message, 'imageUrl' => $imageUrl, 'Url' => $actionDestination);
$arrayToSend = array('to' => $firebase_token, 'notification' => $notification,'priority'=>'high', 'data'=> $data);

$json = json_encode($arrayToSend);


// Codigo para abrir conexion y enviar mensaje
$ch = curl_init();

curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST,"POST");
curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
curl_setopt($ch, CURLOPT_HTTPHEADER,$headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

//Enviamos las peticion
$result = curl_exec($ch);

if ($result === FALSE) {
die('FCM Send Error: ' . curl_error($ch));
}
curl_close($ch);

echo '<h2>Resultado</h2><hr/><h3>Peticion </h3><p><pre>';
echo json_encode($fields,JSON_PRETTY_PRINT);
echo '</pre></p><h3>Respuesta </h3><p><pre>';
echo $result;
echo '</pre></p>';
```

Figura 43 Función Firebase 2

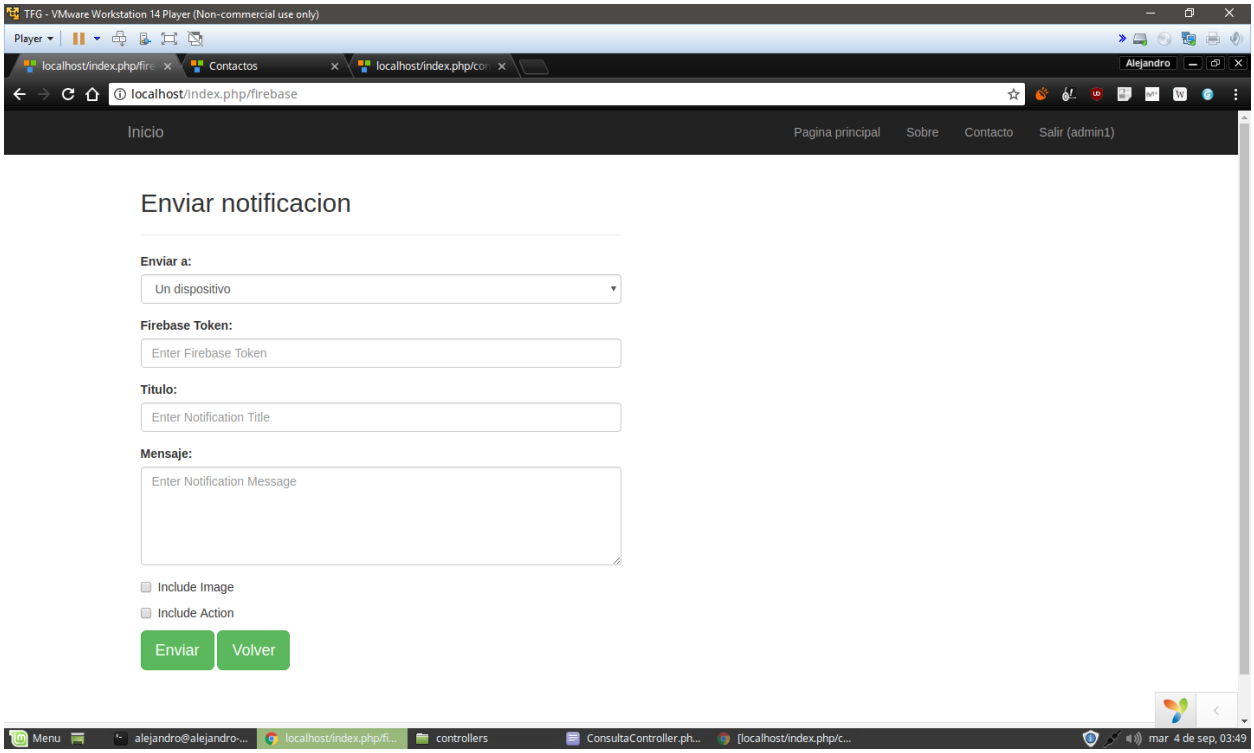

Figura 44 Panel de notitificación

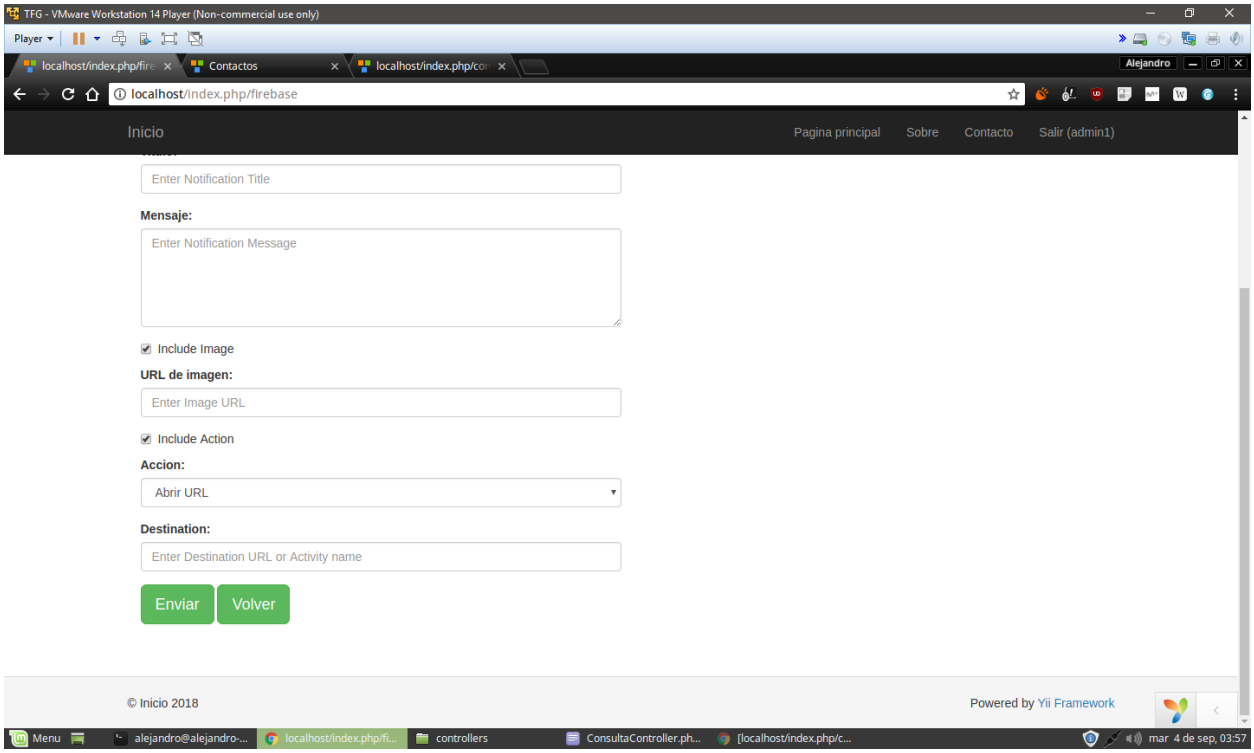

Figura 45 Panel de notificación otras opciones

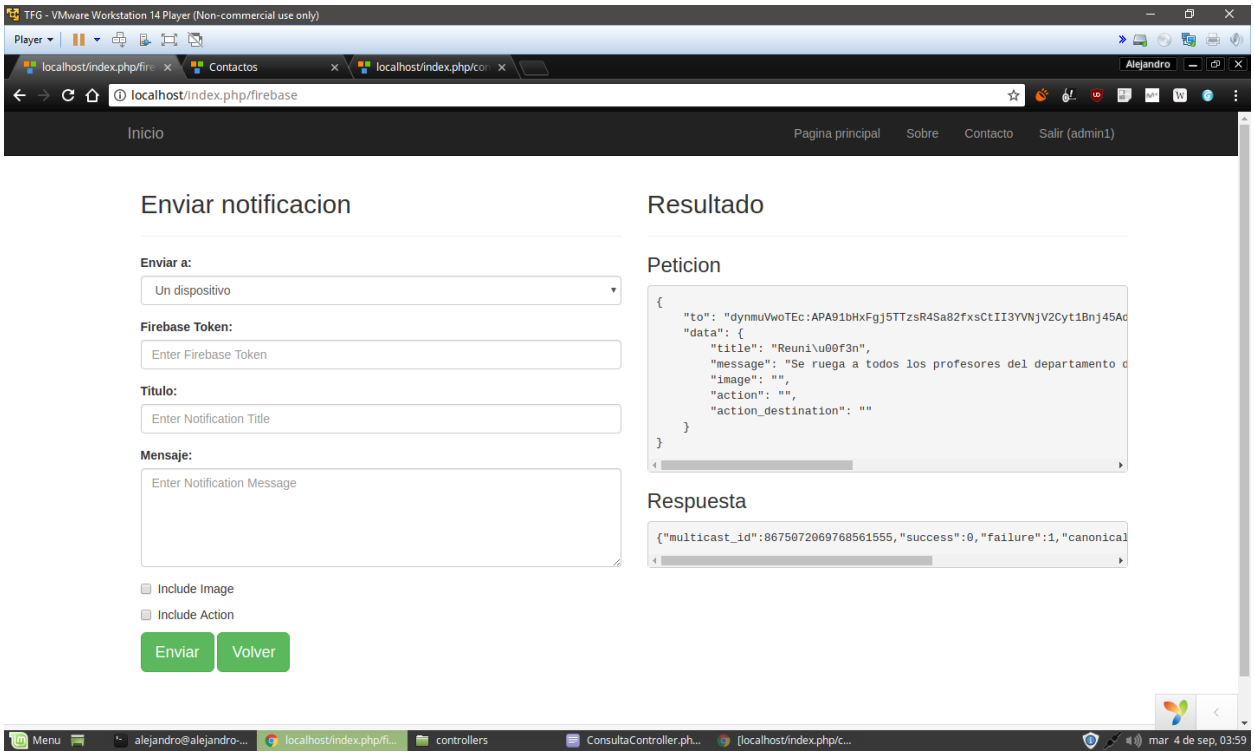

Figura 46 Panel de notitificación. Enviado

En este ejemplo, la respuesta ha sido **failure** puesto que ese token no corresponde a nadie de la base de datos.

En la carpeta **/views/firebase/server.php** habrá que configurar el usuario y contraseña de root y el nombre de la base de datos.

3.5 Aplicación móvil

La aplicación móvil contará con una pantalla de autenticación en el MainActivity que el usuario rellenerá con sus credenciales [16] [17] [18]. Si la autenticación con el servidor es correcta, se pasará a la actividad User donde se mostrará una pantalla que permite registrar el token del teléfono en el servidor, sincronizar y consultar la base de datos local.

Contendrá dos servicios que correrán en segundo plano:

- Un servicio para las llamadas entrantes. Este servicio buscará en la base de datos local y si no encuentra nada, en la remota.
- Un servicio para la recepción de mensajes. Al pulsar en ella, llevará a una actividad nueva Splash que mostrará el contenido del mensaje.

La aplicación necesitará de los siguientes permisos.

- Acceso a la red/Internet: para poder realizar la conexion con el servidor y sincronizar los datos.
- Leer estado del teléfono/Procesar llamadas: correspondiente al servicio PhoneStateReceiver para conocer cuando se reciberá una llamada entrante y sus datos. [19] [20]
- Leer / Escribir en la memoria externa: se leerá el fichero de configuración de la IP del servidor.

- Leer/Escribir contactos: para las funciones que añadirán contacto / eliminarán contacto.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

Figura 47 Permisos Android

Para que la aplicación funcione, el usuario deberá bajarse un archivo "**config**" y se almacenará en las descargas. Éste contendrá la IP donde se alojará el servidor.

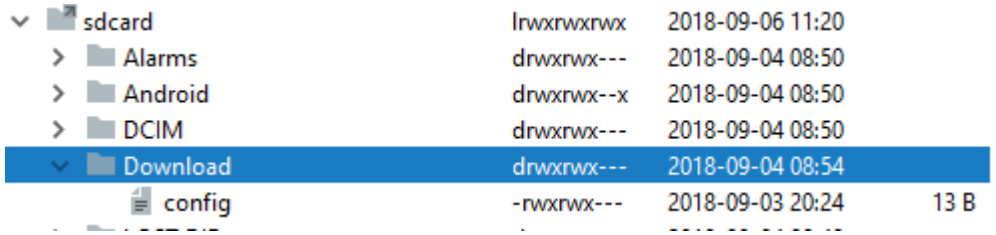

Figura 48 Directorio archivo configuración

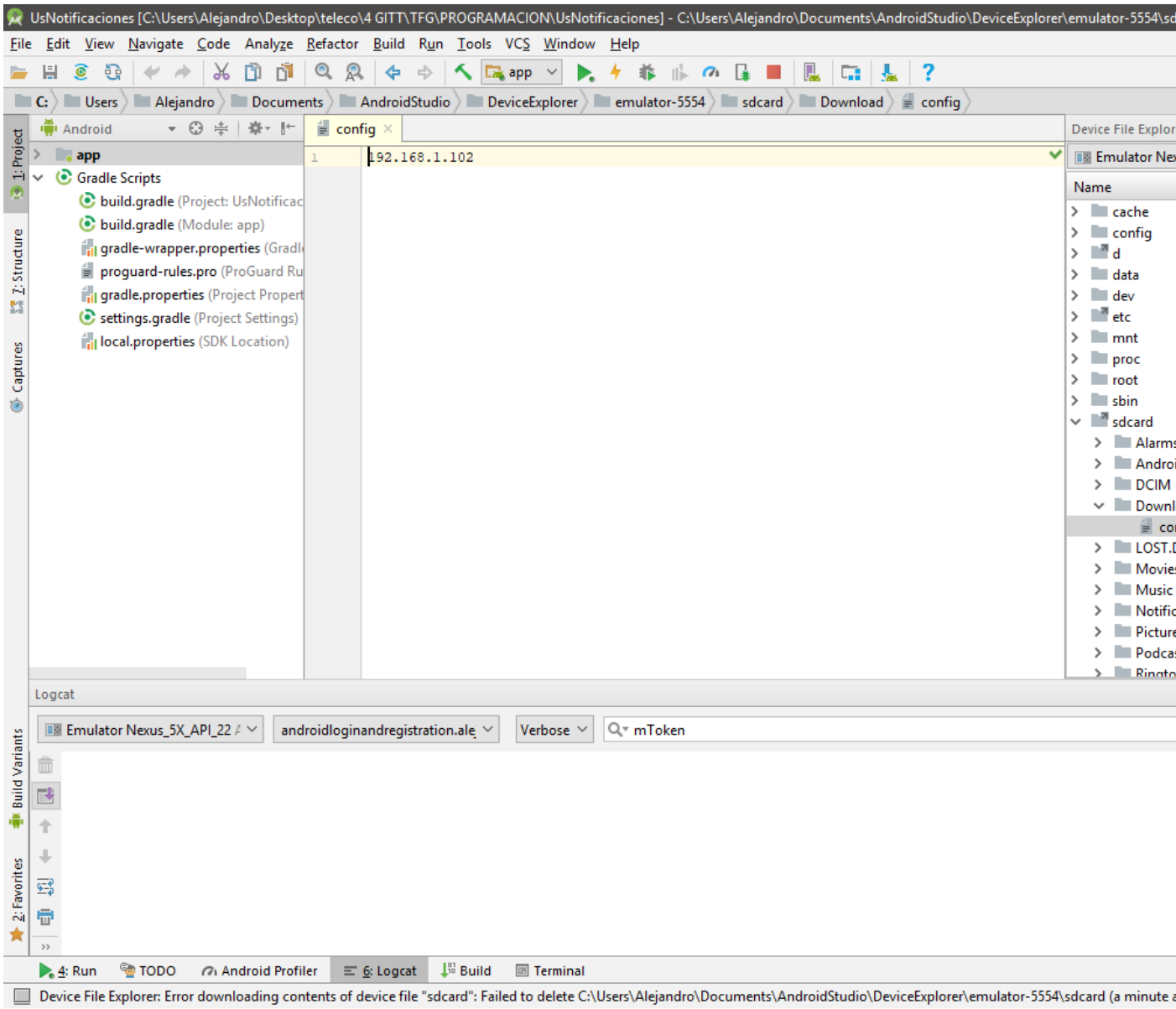

Figura 49 Contenido arhivo de configuración

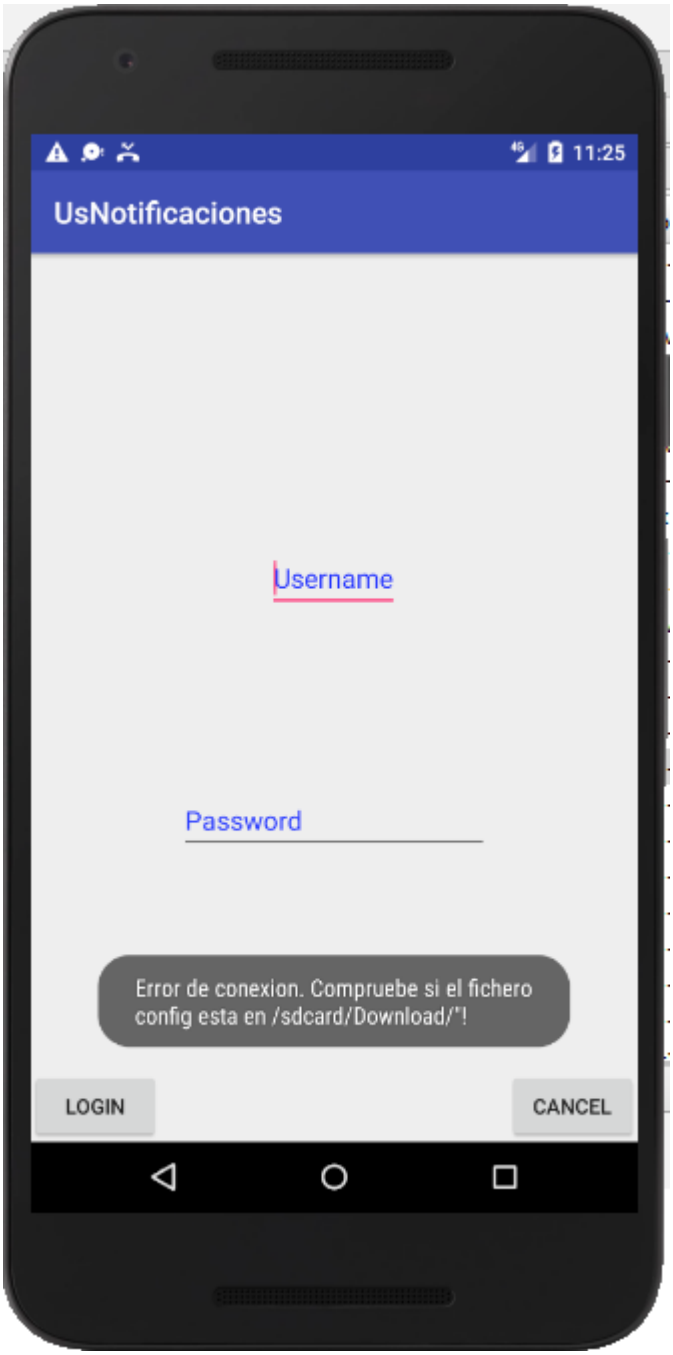

Figura 50 Error archivo de configuración

La actividad **MainActivity.class** constará de varias funciones:

- **read_from_sd**: función encargada que leerá al iniciar la actividad el archivo config ubicado en ***storage/sdcard/Download/*** . Tendrá que leer la línea que compone el fichero que contiene la dirección IP del servidor. En el caso de que la IP fuera errónea, no existiera o no estuviera bien ubicado, se le mostrará un mensaje Toast al usuario especificando *"Error de conexión"*.
- **checkLogin**: función que será llamada cuando se pulse en el boton Login y obtendrá el usuario y contraseña puestos. Encriptará la contraseña con md5 antes de enviarla y posteriormente llamará a AsyncLogin.
- **AsyncLogin**: función que mostrará un diálogo de **Autenticando…** mientras realiza la conexión con el servidor. Enviará los parámetros usuario y contraseña pasados por checkLogin y obtendrá la respuesta

del servidor.

Finalmente mostrará por pantalla el resultado de la operación en forma de mensaje toast.

- **Si el usuario logra autenticarse**: Pasará a la siguiente actividad **User.class**. Se le pasará con un *putExtra* la IP del servidor.
- **Si el usuario no logra autenticarse**: Mostrará un toast con el mensaje "Usuario o contraseña incorrecta".
- **Si no logra conexión con el servidor**: Mostrará un toast con el mensaje "Error de conexión".

- **MD5_Hash:** función que se encargará de encriptar la contraseña del usuario.

El usuario se autenticará con los datos registrados por el administrador. De no ser así se le mostrará un mensaje de error de usuario/contraseña incorrectos.

Si el usuario se autentica correctamente (la base de datos le da el OK), se le mostrará al usuario la siguiente actividad, User.

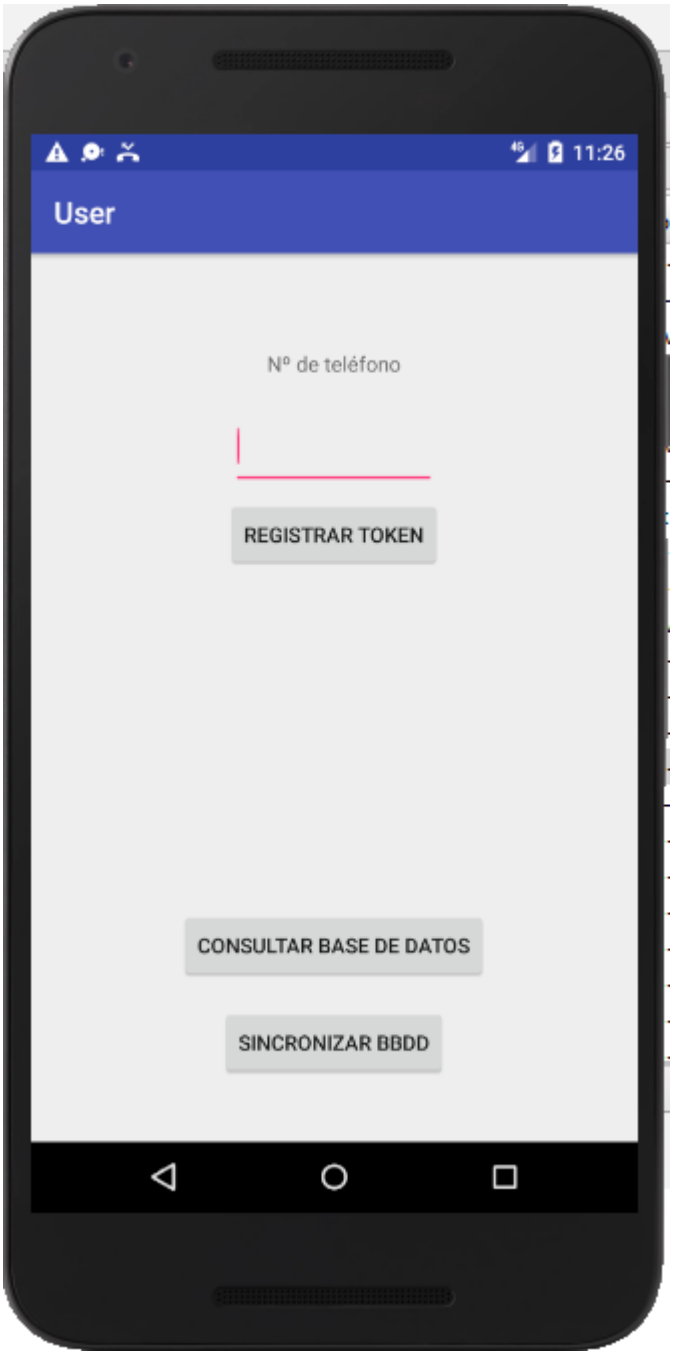

Figura 51 Panel de usuario aplicación móvil

User será la actividad donde el usuario podrá hacer todas sus tareas.

Se compone de las siguientes funciones:

- **registrar_token:** función encargada de obtener el número que se registrará y el token del teléfono que nos proporcionará el servicio de firebase. Pasará estos parametros a la función AsyncToken.
- **eliminar_base_datos:** esta función nada más entrar a la actividad borrará la base de datos actual.
- **sincronizar_bbdd_manual:** función que será llamada cuando se pulse en el botón ***"Sincronizar BBDD"***. Llamará a la función AsyncToken. Sincronizará la base de datos local con la remota.
- **sincronizar_bbdd_automatica:** función que se llamará al principio de la actividad después de eliminar_base_datos para sincronizarla nada más entrar. Llamará a la función AsyncToken.

Sincronizará la base de datos local con la remota.

- **consultar_base_de_datos:** función que se llamará cuando se pulse en el botón ***"Consultar BBDD"***. Abrirá una nueva actividad **ConsultarContactos.class** que listará los contactos existentes en la base de datos local.
- **AsyncToken**: función que recibirá un token y un número y los enviará a la base de datos a **TokenController.php**

  Obtendrá una respuesta y la mostrará en forma de toast.

  - **Si el token se ha registrado correctamente**: Mostrará un toast con el mensaje *"Token registrado"*.
  - **Si el token no se ha podido registrar**: Mostrará un toast con el mensaje *"Token o teléfono incorrecto"*.
  - **Si no hay conexión con el servidor**: Se mostrará el mismo mensaje que en **MainActivity**. Un toast que ponga *"Error de conexión"*.

En este actividad nada mas entrar automaticamente se intentará siempre sincronizar la base de datos. Sin embargo, la primera vez, el usuario recibirá un mensaje de error diciendo que el token es incorrecto.

Para ello el usuario deberá registrar su número de teléfono en el servidor.

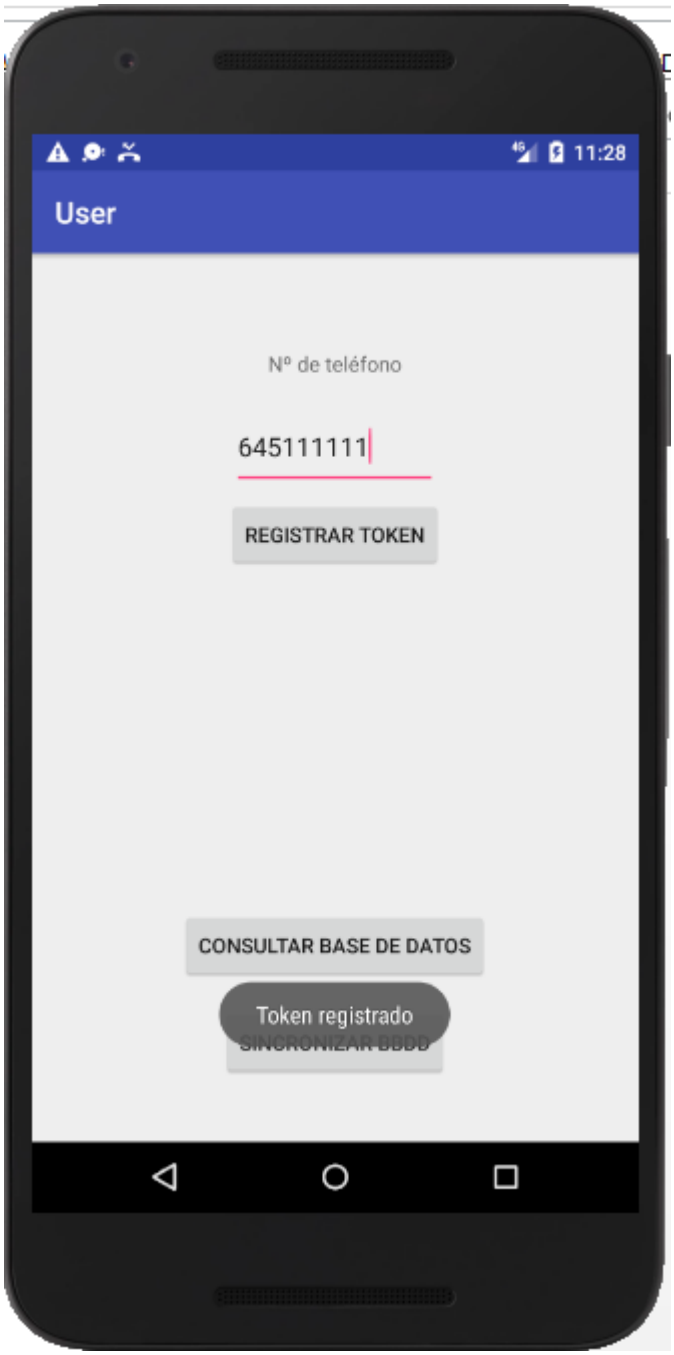

Figura 52 Registro de token desde el móvil

```
//funcion para registrar el token a partir del numero de telefono
public void registrar_token(View arg0){
    //El servicio de Firebase nos dara el token
    FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new OnSuccessListener<InstanceIdResult>() {
        EditText number = (EditText)findViewById(R.id.editText);
        @Override
        public void onSuccess(InstanceIdResult instanceIdResult) {
            String mToken = instanceIdResult.getToken();
            new AsyncLogin().execute(number.getText().toString(), mToken);
        }
    });
}
```

Esto permitirá al usuario utilizar todas las funciones y servicios posibles de la aplicación. Si por un casual el usuario no encontrara su teléfono registrado en la base de datos, se le devolverá un mensaje de error.

El token de registro podrá cambiar en las siguientes situaciones:

- La app borra un ID de instancia.
- La app se restablece en un dispositivo nuevo.
- El usuario desinstala y vuelve a instalar la app.
- El usuario borra los datos de la app.

Cuando se necesite recuperar el token actual, se llamará a ***FirebaseInstanceId.getInstance().getToken().*** Este método muestra el valor null si el token aún no se genera.

Una vez registrado el token correspondiente al teléfono móvil, el usuario podrá sincronizar manualmente la base de datos.

Para ello basta con pulsar al botón **"Sincronizar BBDD"** que hará una sincronización con los datos del servidor.

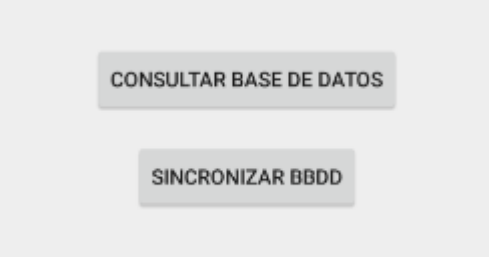

Figura 53 Opciones usuario aplicación móvil

Al pulsar el botón **"Consultar base de datos"** el usuario abrirá otra actividad, **ContactosActivity.class** donde podrá consultar la base de datos que ha descargado y que tiene como local.

Esta actividad no contendrá ninguna función.

```
// Leemos todos los contactos
List<Contacto> contacto = db.getAllContacts();
sustituir.setText("Nombre              Apellido              Telefono");
sustituir.append("\n");
for (Contacto auxiliar : contacto) {
        sustituir.append(auxiliar.getNOMBRE());
        sustituir.append("\t\t\t\t");
        sustituir.append(auxiliar.getAPELLIDOS());
        sustituir.append("\t\t\t\t");
        sustituir.append(auxiliar.getTELEFONO());
        sustituir.append("\n");

    }
}
```

Figura 54 Función devolver contactos

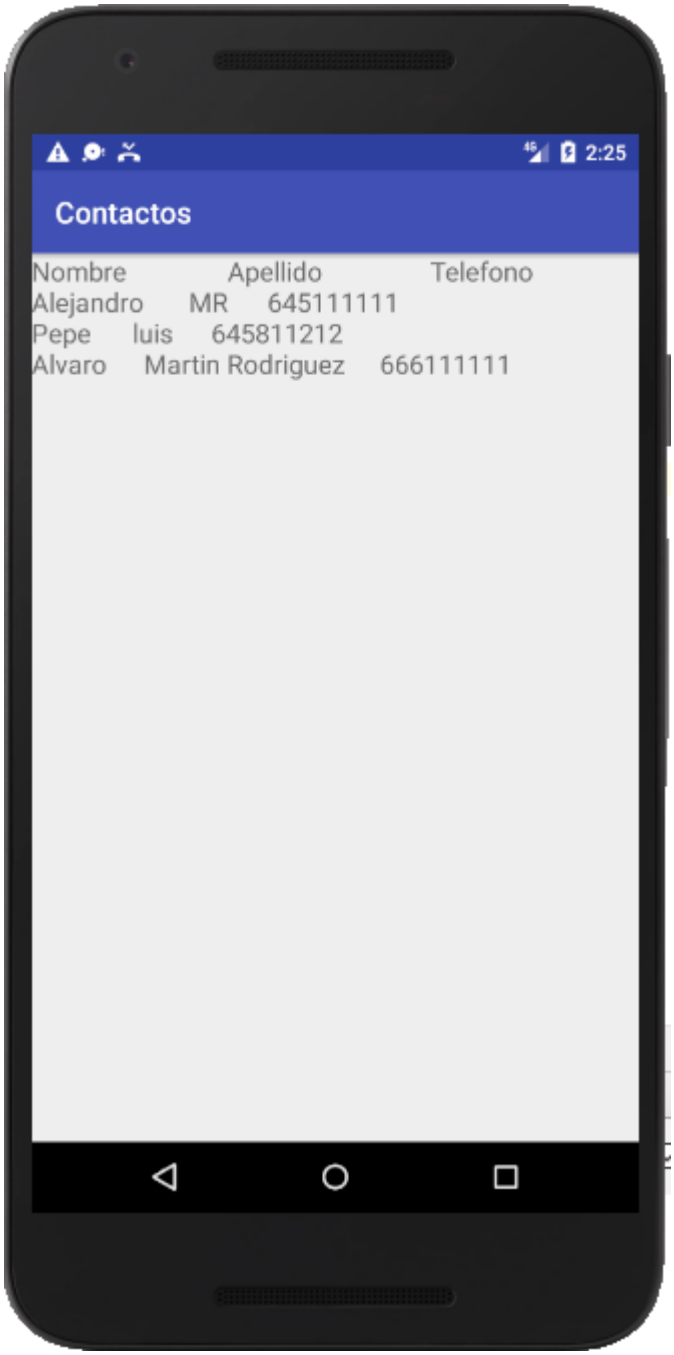

Figura 55 Lista contactos base de datos

Hasta aquí las interacciones que podrá tener el usuario directamente con la aplicación. Ahora explicaremos los servicios que se desarrollarán.

3.5.1 Servicio de búsqueda de número de teléfono

Este servicio responderá cuando el usuario reciba una llamada entrante (***state.equals(TelephonyManager.EXTRA_STATE_RINGING)***).

El servicio se encargará de buscar ese número desconocido en la base de datos local, y, si obtiene respuesta, sustituirá el teléfono en pantalla por el nombre. Para ello, usaremos el servicio **PhoneStateReciver.java** que usará la clase **BroadcastReceiver**.

Utilizará las siguientes funciones:

- **read_from_sd**: función encargada que leerá al iniciar la actividad el archivo config ubicado en ***storage/sdcard/Download/*.** Tendrá que leer la línea que compone el fichero que contiene la dirección IP del servidor.
- **SendNumbertoServer**: función que será llamada en el caso que no se encuentre en la base de datos local el teléfono de la llamada entrante.

  Se encargará de realizar una petición POST al servidor enviando el número de teléfono entrante mediante una petición **Volley**, esperará la respuesta y añadirá el contacto activando una bandera si lo encuentra. Posteriormente, eliminará el contacto si la bandera se ha activado. Pondrá en cola la petición realizada.
- **addContact**: recibirá un nombre y un teléfono y lo añadirá a la agenda local del móvil mediante un *OperationList*.
- **deleteContact**: recibirá un nombre y un teléfono y lo borrará de la agenda local del móvil. Es necesario que esta petición se realice justo después de añadir el contacto, porque al salir de la función la variable *nombre* perderá su valor. Se recorrerá mediante un cursor la agenda personal buscando los valores y posteriormente eliminándolo.

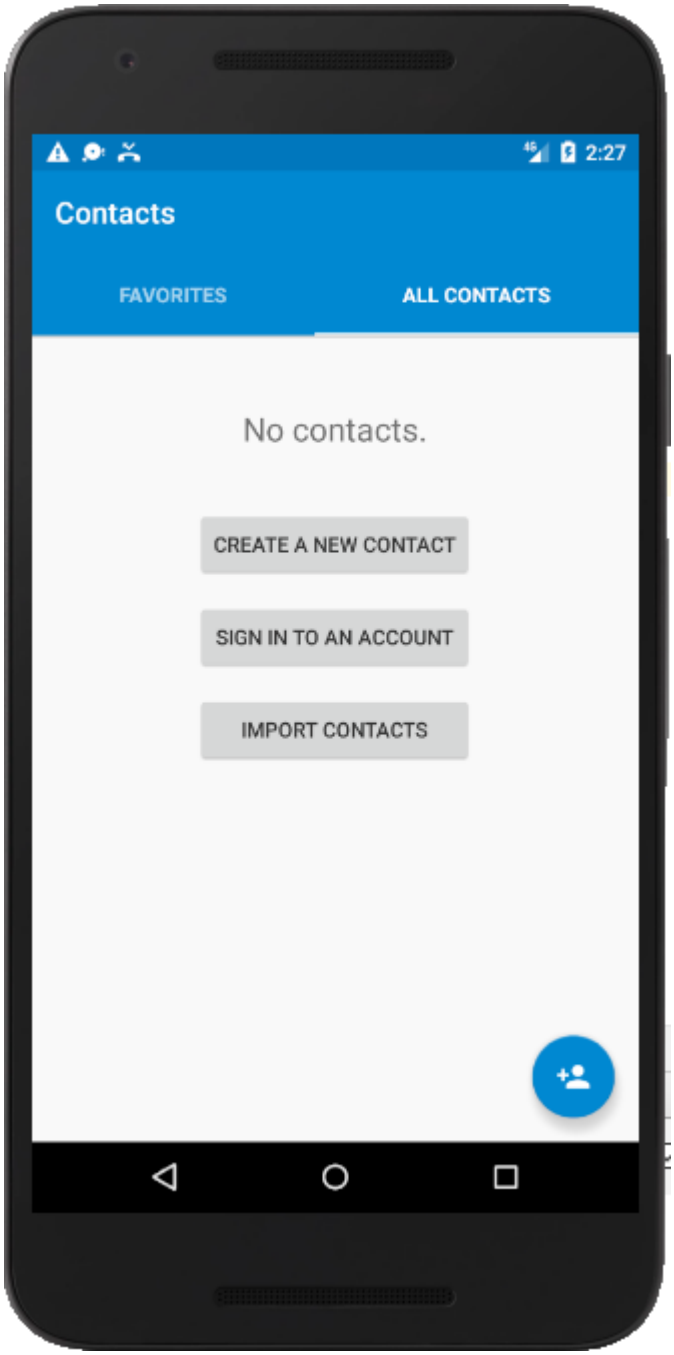

Figura 56 Agenda móvil

Figura 57 Llamada entrante

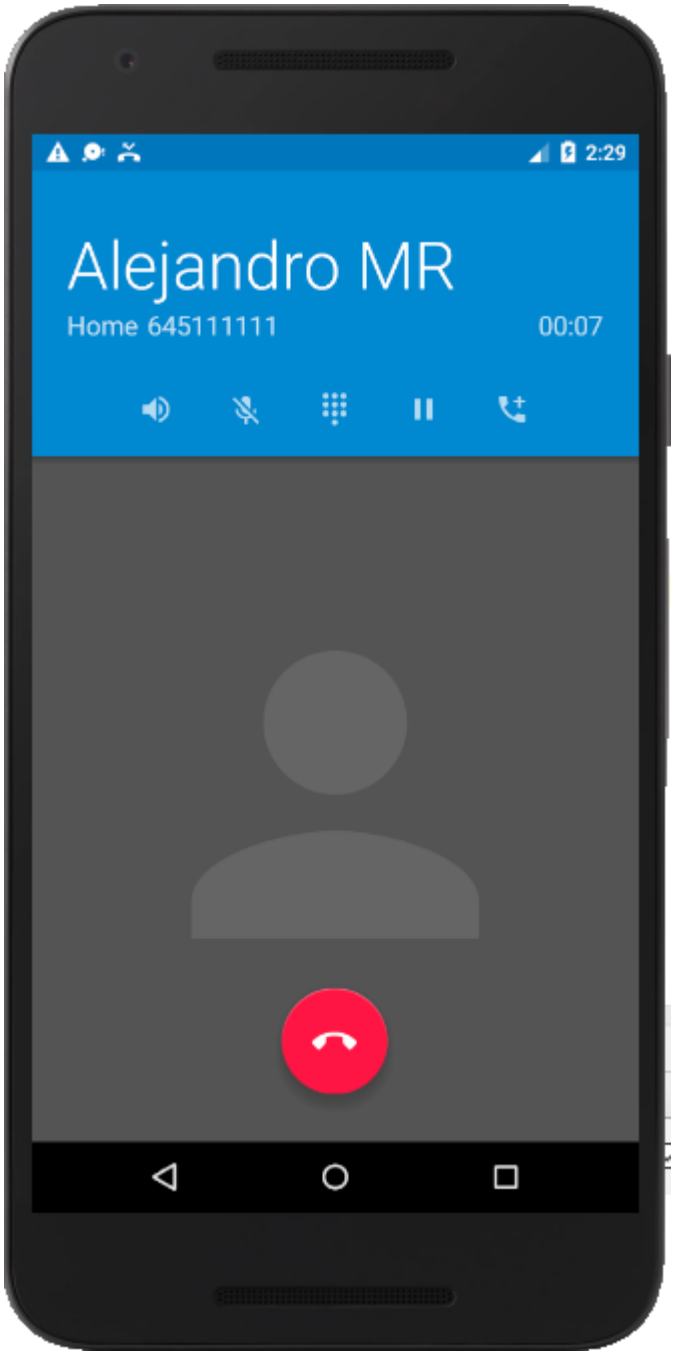

Figura 58 Descolgar

Si por casualidad, el teléfono no se encontrase en la base de datos local, el servicio buscará en el servidor, a ver si obtiene alguna respuesta.

```
        if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) { //llaman
            SQLConfigManager db = new SQLConfigManager(context);
            resultado = db.recuperarN_CONTACTO(incomingNumber); //comprobamos en la base de datos local que exista
            if (resultado.length() > 1) {   //si se ha encontrado algo
                    //añadimos el contacto a los contactos del movil y lo eliminamos, tiempo suficiente
                    // para que el call ui cambie
                    addContact(context, resultado, incomingNumber);
                    deleteContact(context, incomingNumber, resultado);
            } else {
                sendNumberToServer(incomingNumber, context);    //si no enviamos al servidor
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 59 Consultar en el servidor

Para ello se realizará todo en una petición volley y con el método POST. Se enviará el número a consultar y se esperará una respuesta inmediata del nombre.

Si no se encontrase ningún nombre relacionado con el teléfono asociado, no se mostrará nada por pantalla.

Para mostrar los nombres por pantalla, Android buscará en la agenda de contactos el nombre, ya que la variable **DISPLAY_NAME** de una llamada es solo de lectura.

Para poder mostrar el nombre de la respuesta habrá que usar dos funciones. Una que añadirá el usuario a la agenda de contactos y otra que la eliminará. El motivo principal por el que no se guardará el contacto es porque no se permite tocar la información del usuario. Si así fuera el usuario tendria sus contactos mezclados con los de la aplicación y al final será todo muy engorroso. Para ello, se añadirá el contacto y se eliminará, dando el tiempo justo a Android para buscarlo.

```
//codigo para añadir el contacto temporalmente, funciona
private void addContact(Context context, String nombre, String telefono) {
    ArrayList<ContentProviderOperation> operationList = new ArrayList<~>();
    operationList.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_URI)
            .withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, null)
            .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, null)
            .build());

    // first and last names
    operationList.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
            .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, previousResult: 0)
            .withValue(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
            .withValue(ContactsContract.CommonDataKinds.StructuredName.GIVEN_NAME, nombre)
            .build());

    operationList.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
            .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, previousResult: 0)
            .withValue(ContactsContract.Data.MIMETYPE, ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
            .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, telefono)
            .withValue(ContactsContract.CommonDataKinds.Phone.TYPE, ContactsContract.CommonDataKinds.Phone.TYPE_HOME)
            .build());
    try {
        ContentProviderResult[] results = context.getContentResolver().applyBatch(ContactsContract.AUTHORITY, operationList);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 60 Función añadir contacto

```
//codigo para borrar el contacto despues de la llamada
public static boolean deleteContact(Context ctx, String phone, String name) {
    Uri contactUri = Uri.withAppendedPath(ContactsContract.PhoneLookup.CONTENT_FILTER_URI, Uri.encode(phone));
    Cursor cur = ctx.getContentResolver().query(contactUri, projection: null, selection: null, selectionArgs: null, sortOrder: null);
    try {
        if (cur.moveToFirst()) {
            do {
                if (cur.getString(cur.getColumnIndex(ContactsContract.PhoneLookup.DISPLAY_NAME)).equalsIgnoreCase(name)) {
                    String lookupKey = cur.getString(cur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));
                    Uri uri = Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
                    ctx.getContentResolver().delete(uri, where: null, selectionArgs: null);
                    return true;
                }

            } while (cur.moveToNext());
        }

    } catch (Exception e) {
        System.out.println(e.getStackTrace());
    }
    return false;
}
```

Figura 61 Función eliminar contacto

3.5.2 Servicio de notificaciones de Firebase Cloud Messasing

Este servicio se implementará con el objetivo de que el usuario reciba los mensajes enviados por el servidor. Se hará uso de la clase **MyFirebaseMessagingService.class** que utiliza **FirebaseMessagingService**. [21] [22] [23] [24]

Esta clase contiene varias funciones reseñables:

- **onMessageReceived**: esta función recibirá un *RemoteMessage* y lo desglozará internamente en Logs. Recibirá un *JsonObject* y lo convertirá a *String* y se lo pasará a **sendPushNotification**.
- **sendPushNotification:** recibirá un JSONObject y un RemoteMessage. Se encargará de realizar la notificación al sistema.

  Esta función dividirá el mensaje en varias partes:

  - Título
  - Cuerpo
  - ImagenUrl
  - Url

  Cogerá si existe campo *ImageUrl* y la convertirá a un *Bitmap*. Iniciará un nuevo *Intent* para poder mostrarlo todo en **SplashActivity.class.** Para ello hará uso de *PendingIntent.*Creará un NotificationCompact.Builder para añadirle los diferentes campos a la notificación.

```
        .setContentTitle(title)
        .setContentText(message)
        .setContentText(imageUrl)
        .setContentText(Url)
        .setStyle(new NotificationCompat.BigPictureStyle()
                .bigPicture(image))/*Notification with Image*/
        .setLargeIcon(image)/*Notification icon image*/
        .setAutoCancel(true)
        .setSound(defaultSoundUri)
        .setContentIntent(pendingIntent);
NotificationManager notificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
```

Figura 62 Contenido notificación Firebase

- **getBitmapfromUrl**: esta función se encargará de a partir de una url, de obtener la imagen y la convertirá en bitmap para posteriormente devolverla.

Para entrar en la actividad **SplashActivty.class** se necesitará recibir una notificación. Sin ella será imposible acceder.

La pantalla por defecto será la siguiente.

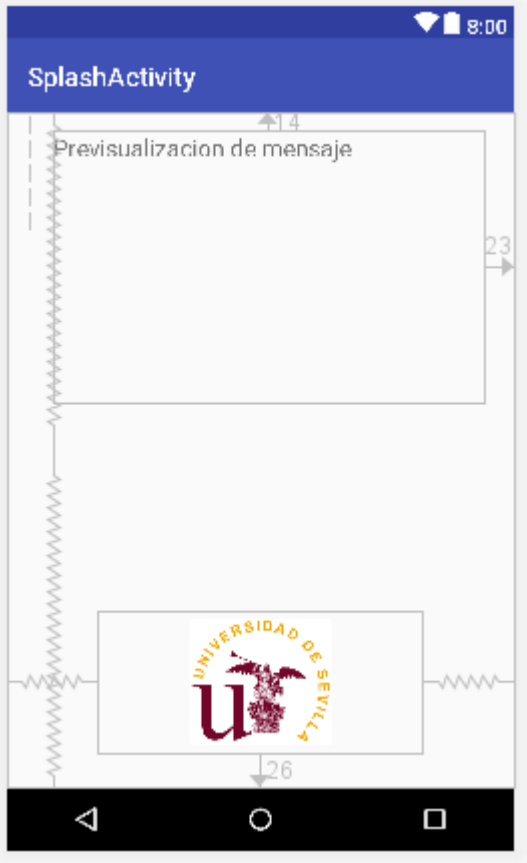

Figura 63 Firebase SplashActivity

Se mostrará una descripción más detallada del mensaje, con su titulo y cuerpo, una opción de enlace, invisible si no hay nada, y una imagen por defecto, si no se envía ninguna.

Figura 64 Recibo de notificación desde el servidor

Con la posibilidad de apertura del enlace.

Figura 65 Apertura mensaje notificatorio desde servidor

Donde enlace será sustituido por la url de información. Para ello se comprobará que el enlace empiece por **http://** o **https://** ya que de lo contrario, provocará un error interno en Android.

```
public void Open_Browser (View arg0) {
    if(Url!=null){
        //Comprobamos que la url empiece con http o https si no dara error al intentar abrirlo
        if (!Url.startsWith("http://") && !Url.startsWith("https://"))
            Url = "http://" + Url;
        Intent browser = new Intent(Intent.ACTION_VIEW , Uri.parse(Url));
        startActivity(browser);
    }
```

Figura 66 Abrir url

3.6 Diagramas UML

3.6.1 Casos de uso

A continuación se detallarán los casos de uso del usuario con la aplicación móvil y del administrador con su panel de control.

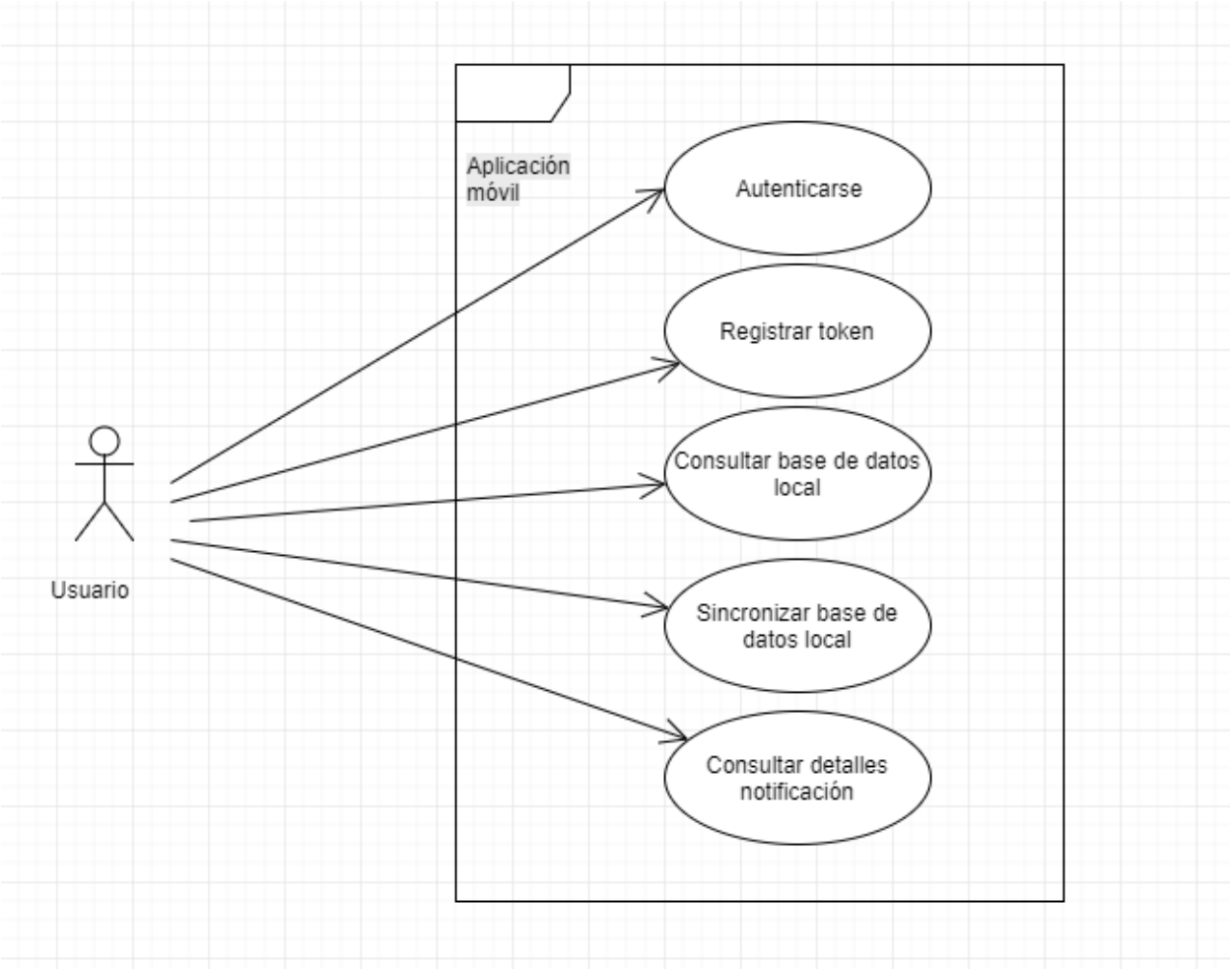

Figura 67 Caso de uso usuario aplicación

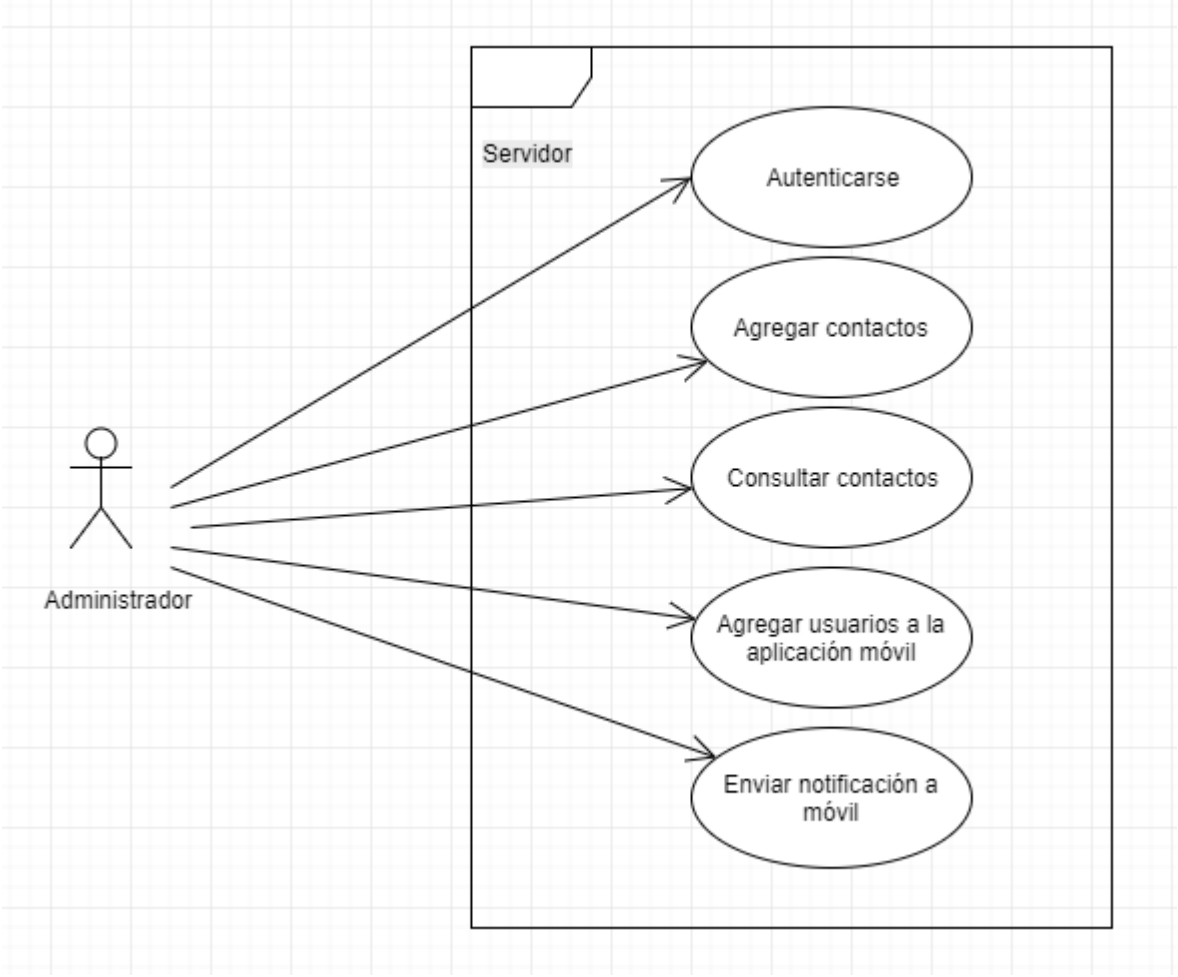

Figura 68 Caso de uso administrador

3.6.2 Diagramas de mensaje

En este apartado se expondrá un diagrama de la autenticación del usuario en la aplicación móvil, un diagrama al sincronizar la base de datos local y uno del envío de notificación por el administrador.

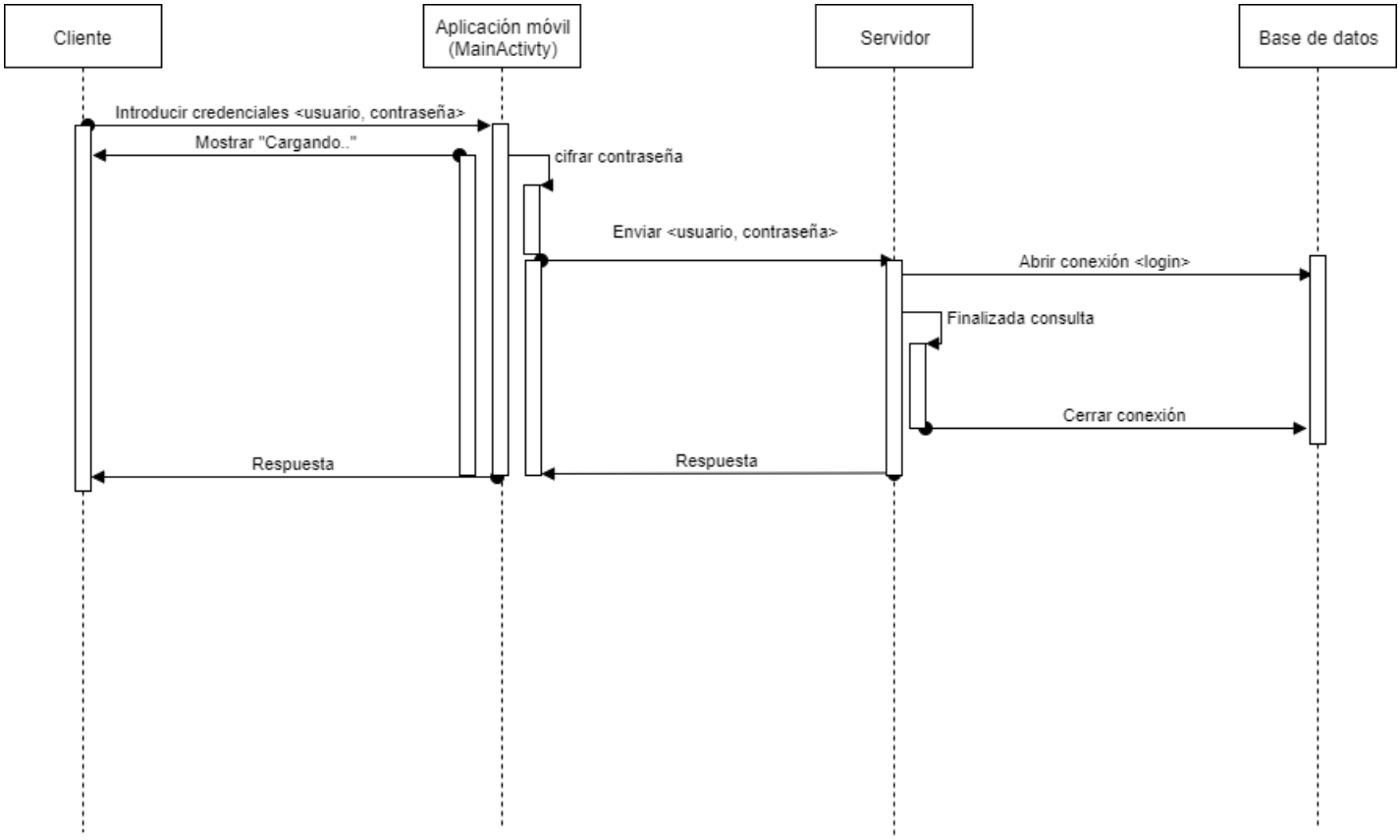

Figura 69 Diagrama de mensaje autenticación

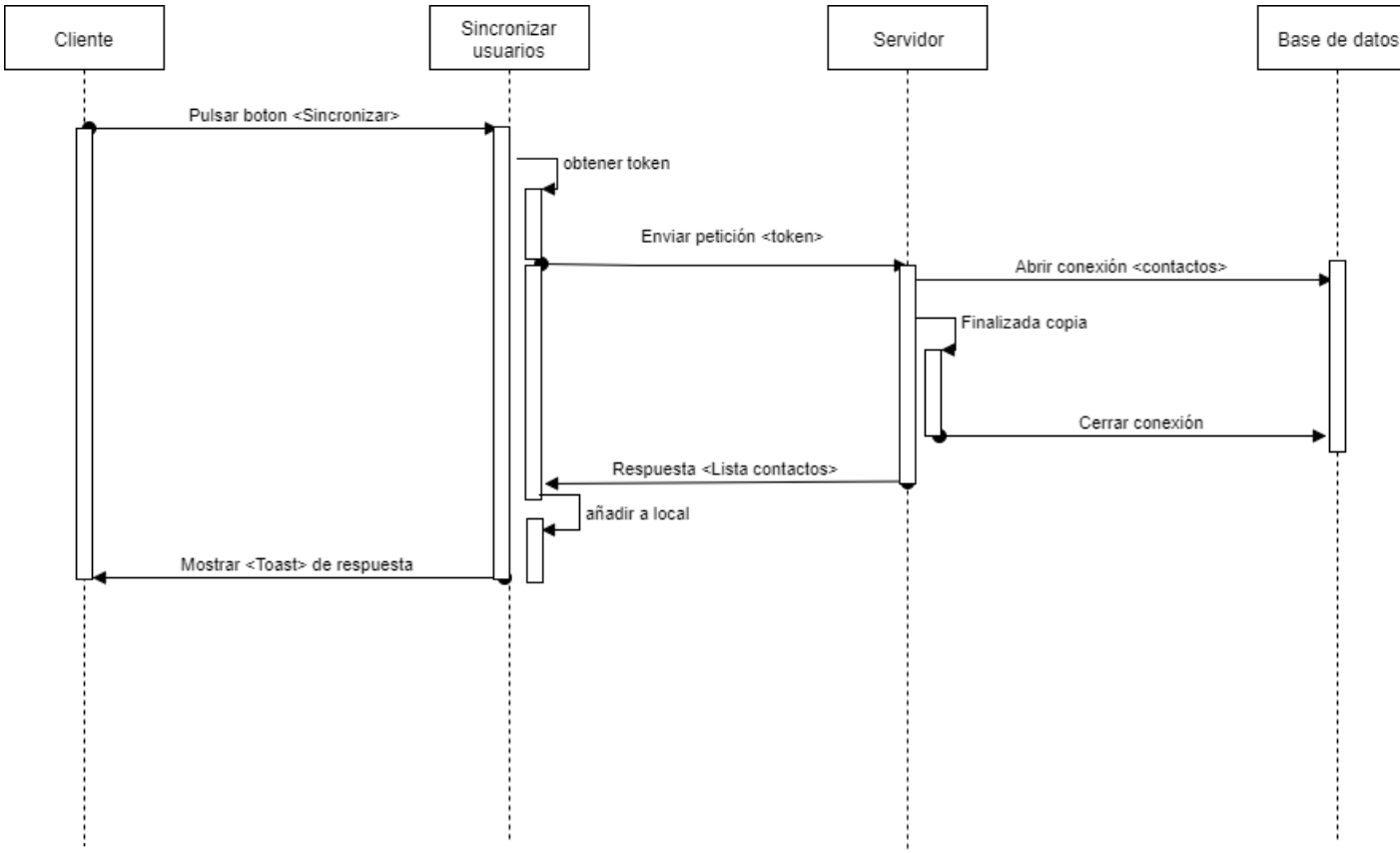

Figura 70 Diagrama de mensaje sincronización

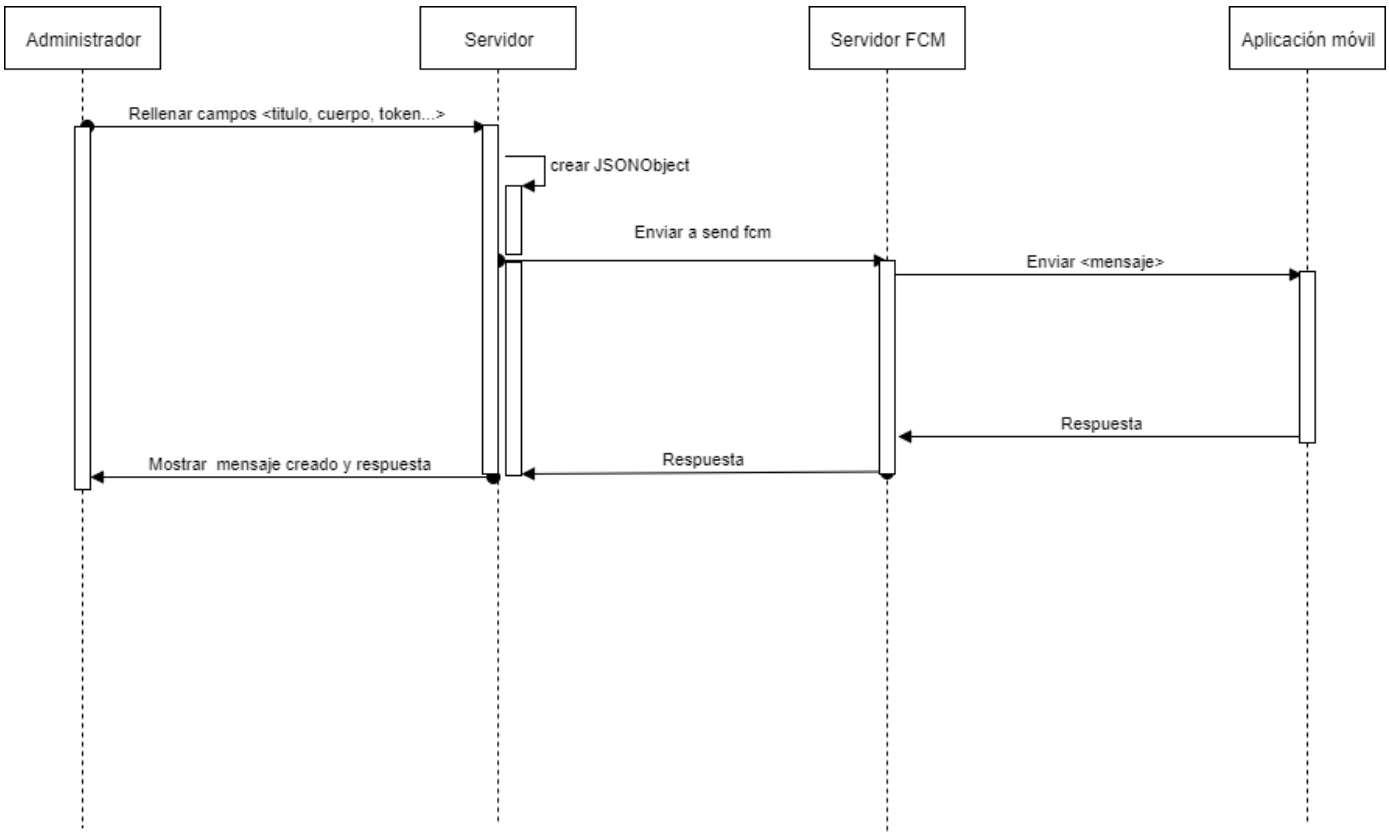

Figura 71 Diagrama de mensaje envío de notificación

BLOQUE 4. PLIEGO DE NECESIDADES

En este bloque se abarcará todos los requisitos necesarios a la hora de implementar este proyecto en otro destino que no sea aquel donde se ha desarrollado originalmente y al cual está adaptado. Para ello se diferenceriará las tres herramientas que lo componen y se recapitularán sus requisitos.

Recordar que el sistema original en el que se desarrollará será en una máquina virtual en VMware, en un sistema Linux por lo que se el ecosistema deberá ser similar para un correcto funcionamiento.

Las características de esta máquina virtual son:

- En este proyecto ha sido Linux Mint 19 versión x64 bits.
- La máquina virtual ha tenido dos núcleos del procesador dedicado.
- 2048 gb de ram y 20 gb de disco duro.
- Conectividad en puente con la máquina host.

4.1 Base de datos

Para la base de datos se necesitará tener instalado mysql. Por defecto debería debir instalado en la imagen de Linux Mint 19. Si no estuviera disponible se adjuntará la versión utilizada durante el desarrollo.

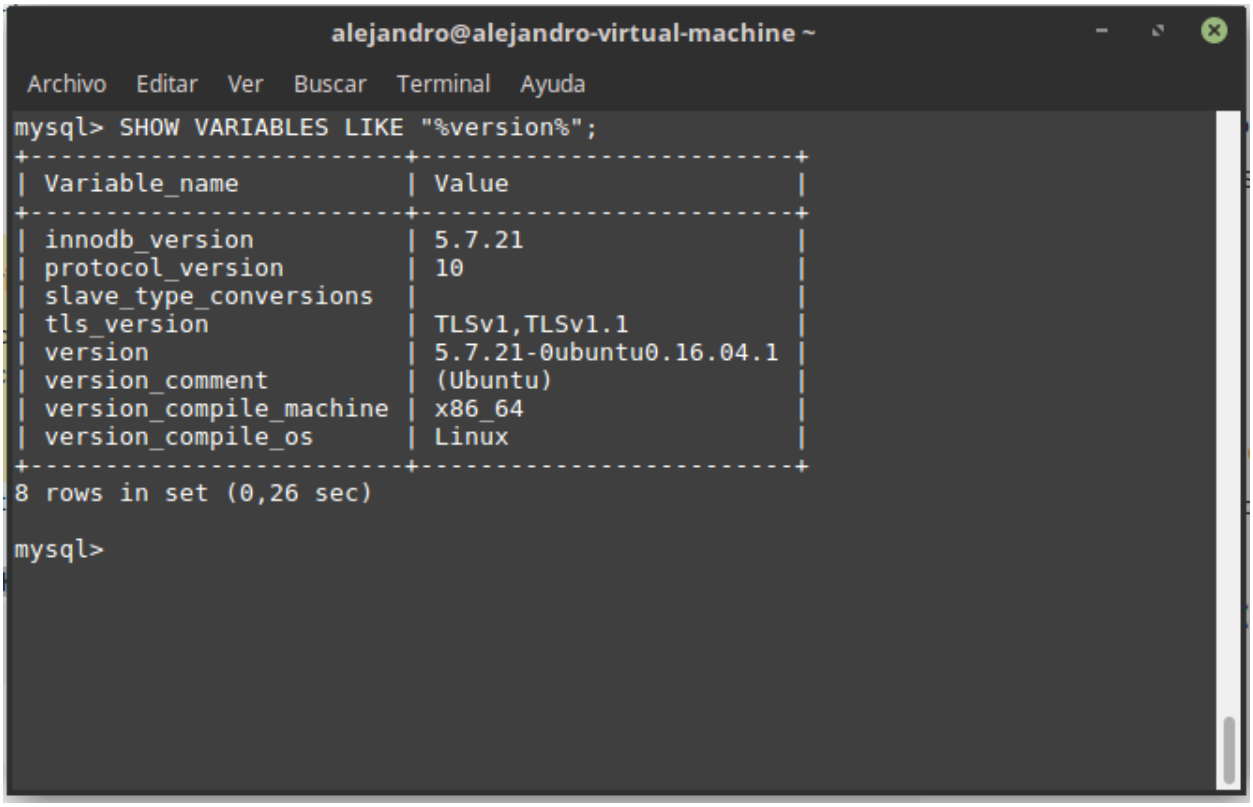

Figura 72 Características Linux Mint

Además, hará falta crear una base de datos de nombre: "***Agenda***" cuyas tablas sean:

-Contactos.

-Login.

-Admin.

-Migration. Esta última tabla es la que utilizará Yii para la concordancia con el servidor y que puede inyectar en la base de datos.

- Script para crear las tablas:

```
CREATE TABLE admin (
  idadmin INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  username VARCHAR(20) NOT NULL,
  password VARCHAR(20) NOT NULL,
  authKey VARCHAR(50) NULL,
  PRIMARY KEY(idadmin)
);

CREATE TABLE contacto (
  idUsuario INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
  nombre VARCHAR(20) NOT NULL,
  apellidos VARCHAR(45) NOT NULL,
  telefono VARCHAR(9) NOT NULL,
  token VARCHAR(50) NULL,
  PRIMARY KEY(idUsuario)
);

CREATE TABLE login (
  idlogin INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  user VARCHAR(20) NULL,
  passwd VARCHAR(20) NULL,
  PRIMARY KEY(idlogin)
);
```

- Script para eliminar las tablas:

```
DROP TABLE login;

DROP TABLE contacto;

DROP TABLE admin;
```

- Script para optimizar las tablas:

```
DROP TABLE login;

DROP TABLE contacto;

DROP TABLE admin;
```

- Script para reparar las tablas:

```
OPTIMIZE TABLE admin;

OPTIMIZE TABLE contacto;

OPTIMIZE TABLE login;
```

4.2 Servidor Web

Para el servidor web (Yii) hará falta una serie de configuraciones y adaptaciones que enumeramos a continuación:

- Instalación y configuración del servidor apache para exportar la carpeta.
- Instalación de curl para el funcionamiento del servicio de notificaciones firebase.
- Modificación de los ficheros siguientes ficheros :

  - Config/db.php y config/db2.php donde deberá ponerse en la siguiente línea el nombre de la base de datos, el usuario y contraseña de root.

```
'dsn' => 'mysql:host=localhost;dbname=',
'username' => 'root',
'password' => '',
```

  - Config/web.php: una key válida para Yii (aleatoria) para poder ser iniciado.

```
*'cookieValidationKey' => 'asdsadsadsadsaa',
```

  - En migrations/ estará la estructura de la tabla principal de contacto. Si quisiera cambiarse alguno de sus valores habría que editarlo aquí y volver a realizar la migración.

```
public function up()
  {
    $this->createTable('contacto', [
      'idUsuario' => $this->primaryKey(),
```

```
            'nombre' => $this->string()->notNull(),
            'apellido' => $this->string()->notNull(),
                'telefono' => $this->string()->notNull(),
                'token' => $this->string()

            ]);
    }
```

- Views/usuarios/server.php : se deberá cambiar la parte de conexión con la base de datos.

```
$db = mysqli_connect('localhost', 'root', ' ', 'agenda');
```

Permisos:

```
drwxrwxrwx  2 alejandro alejandro    4096 jun  5  2017 assets
drwxr-xr-x  2 alejandro alejandro    4096 sep  4 02:22 backend
-rwxrwxrwx  1 alejandro alejandro     800 jun  5  2017 codeception.yml
drwxrwxrwx  2 alejandro alejandro    4096 sep  1 14:21 commands
drwxr-xr-x  2 alejandro alejandro    4096 ago 12 13:16 components
-rwxrwxrwx  1 alejandro alejandro    1782 jun  5  2017 composer.json
-rwxrwxrwx  1 alejandro alejandro  109291 jun  5  2017 composer.lock
-rwxr-xr-x  1 alejandro alejandro 1875611 ago 29 11:51 composer.phar
-rw-r--r--  1 alejandro alejandro  305459 ago 29 11:50 composer-setup.php
drwxrwxrwx  2 alejandro alejandro    4096 ago  1 11:42 config
drwxrwxrwx  2 alejandro alejandro    4096 sep  4 05:16 controllers
-rwxrwxrwx  1 alejandro alejandro    1622 jun  5  2017 LICENSE.md
drwxrwxrwx  3 alejandro alejandro    4096 jun  5  2017 mail
drwxrwxrwx  2 alejandro alejandro    4096 feb 23  2018 migrations
drwxrwxrwx  2 alejandro alejandro    4096 sep  1 14:38 models
-rwxrwxrwx  1 alejandro alejandro    6579 jun  5  2017 README.md
-rwxrwxrwx  1 alejandro alejandro    5141 jun  5  2017 requirements.php
drwxrwxrwx  6 alejandro alejandro    4096 jul 28  2017 runtime
drwxrwxrwx  9 alejandro alejandro    4096 jun  5  2017 tests
drwxrwxrwx 23 alejandro alejandro    4096 jun  5  2017 vendor
drwxrwxrwx  7 alejandro alejandro    4096 sep  1 12:47 views
drwxrwxrwx  4 alejandro alejandro    4096 jun  5  2017 web
-rwxrwxrwx  1 alejandro alejandro     556 jun  5  2017 yii
```

-rwxrwxrwx 1 alejandro alejandro 515 jun 5 2017 yii.bat

Config

-rwxrwxrwx 1 alejandro alejandro 990 jun 5 2017 console.php
-rwxrwxrwx 1 alejandro alejandro 188 ago 7 2017 db2.php
-rwxrwxrwx 1 alejandro alejandro 188 ago 7 2017 db.php
-rwxrwxrwx 1 alejandro alejandro 60 jun 5 2017 params.php
-rwxrwxrwx 1 alejandro alejandro 199 jun 5 2017 test_db.php
-rwxrwxrwx 1 alejandro alejandro 1001 jun 5 2017 test.php
-rwxrwxrwx 1 alejandro alejandro 2531 ago 1 11:42 web.php

Controllers

-rwxrwxrwx 1 alejandro alejandro 3149 feb 23 2018 AgendaController.php
-rwxrwxrwx 1 alejandro alejandro 1740 sep 4 03:32 ConsultaController.php
-rw-r--r-- 1 alejandro alejandro 700 sep 1 14:27 FirebaseController.php
-rwxrwxrwx 1 alejandro alejandro 1776 sep 1 14:29 LoginController.php
-rwxrwxrwx 1 alejandro alejandro 2015 sep 1 14:37 SincronizarController.php
-rwxrwxrwx 1 alejandro alejandro 3030 sep 1 14:33 SiteController.php
-rwxrwxrwx 1 alejandro alejandro 1760 sep 4 05:16 TokenController.php
-rw-r--r-- 1 alejandro alejandro 588 sep 1 14:36 UsuariosController.php

Migrations

-rw-r--r-- 1 alejandro alejandro 763 feb 23 2018 m170807_112948_create_contacto_table.php

Views

drwxrwxrwx 2 www-data www-data 4096 sep 1 14:43 agenda
drwxr-xr-x 2 alejandro alejandro 4096 sep 4 04:05 firebase
drwxrwxrwx 2 alejandro alejandro 4096 sep 1 15:05 layouts
drwxrwxrwx 2 alejandro alejandro 4096 sep 1 15:09 site
drwxr-xr-x 2 alejandro alejandro 4096 sep 4 06:37 usuarios

4.3 Aplicación móvil

Para la aplicación móvil el usuario deberá descargarse un fichero “config” sin extensión y que deberá situar en la carpeta por defecto de las descargas: ***/storage/sdcard/Downloads***.

Dicho esto, el usuario ya tendrá acceso a la aplicación móvil si se ha dado de alta en la base de datos, tanto el usuario de autenticación como sus datos.

Una vez hecho esto, el usuario sólo deberá registrar el token correspondido a su teléfono móvil para poder utilizar los servicios del servidor web.

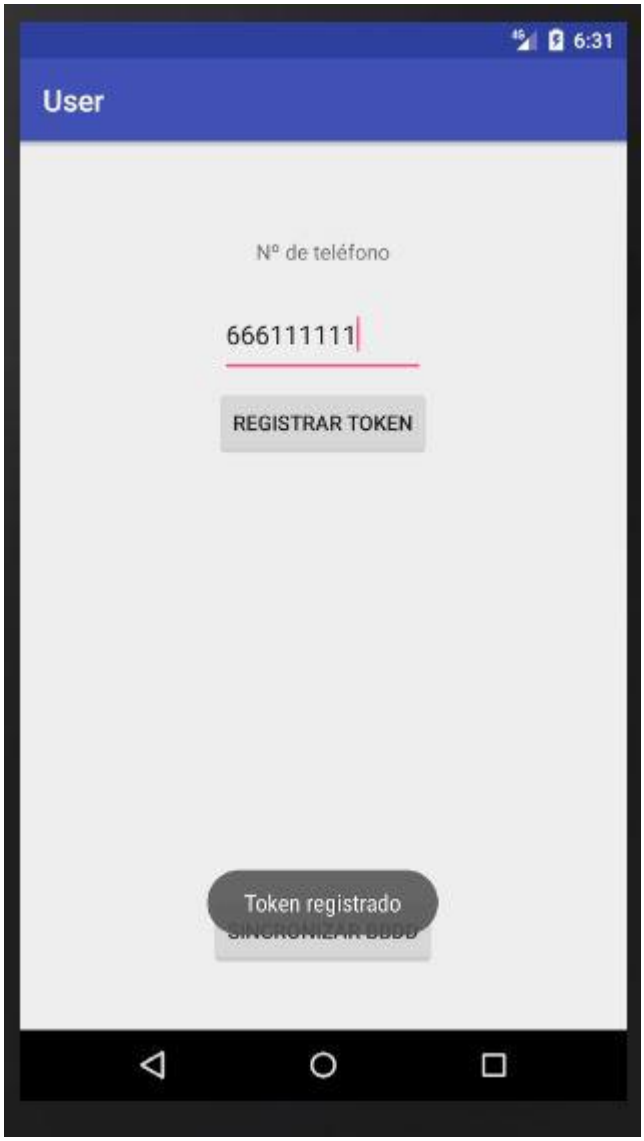

Figura 73 Contacto correctamente registrado.

BLOQUE 5. CONCLUSIONES Y LINEA FUTURA

En este bloque se hablará de las conclusiones sacadas del proyecto y de un trabajo futuro que se podría realizar para mejorarlo, así como la aportación de este trabajo.

5.1 Conclusiones

A raíz del problema planteado, la idea de este proyecto ha sido una apuesta inteligente por parte del personal docente. Este proyecto conlleva muchos conceptos, ideas y novedades con el fin de facilitar a los profesores la comunicación entre ellos incluyendo una respuesta rápida.

Con este proyecto se consigue:

- Facilidad a la hora de comunicarse con los demás usuarios de un departamento ya que no necesitan saber en ningún momento su teléfono ni interaccionar entre ellos si les hiciera falta para obtenerlo.
- Una centralización de contactos que cualquier usuario dado de alta puede consultar.
- Un centro de información por parte del administrador que será capaz de comunicar, informar y notificar a los usuarios que le interese detalles, noticias, imágenes o cualquier otra cosa digna de ser nombrada.
- Ahorro de tiempo por parte tanto del usuario como del adminitrador a la hora de contactar o buscar contactar con otro usuario.

5.2 Línea futura

Una vez visto las capacidades de las que dispone este proyecto, podemos proponer las siguientes líneas de mejoras, siendo estas líneas las más inmediatas.

Basadas en la aplicación móvil:

- Búsqueda de contactos en la base de datos
- Posibilidad de elegir y añadir contactos a la agenda personal.
- Envio de notificaciones entre usuarios con firebase.
- Capacidad de suscripción a temas o grupos de departamentos.
- Cambio de contraseña desde la aplicación móvil.
- Registrarse desde la aplicación móvil y que el administrador sólo tenga que aceptar la petición.
- Simultaneidad de aplicaciones por parte de un mismo usuario en diferentes dipositivos.

- Uso del correo para recepción de notificaciones.
- Recibir notificaciones en primer plano.
- Buscar un método alternativo para mostrar el nombre encontrado.

Basadas en la base datos:

- Un campo “correo” por si se querría hacer envios personales.
- Posibilidad que un contacto tenga multiples teléfonos asignados.
- Posibilidad de que un mismo contacto posea dos teléfonos y solo necesite un token para ambos.

Basadas en el servidor:

- Envio de multiples token
- Envio a temas suscritos por los usuarios.
- Envio de una copia al correo electrónico.
- Integración del protocolo https.
- Incluir más funciones para el panel del administrador.
- Gestionar usuarios de aplicación móvil desde el panel de control.

BLOQUE 6. BIBLIOGRAFIA

[1] [En línea]. Available: https://www.yiiframework.com/.

[2] [En línea]. Available: https://www.vmware.com/es.html.

[3] [En línea]. Available: https://linuxmint.com/.

[4] [En línea]. Available: https://www.mysql.com/.

[5] [En línea]. Available: https://www.yiiframework.com/doc/guide/1.1/es/basics.mvc.

[6] [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/?hl=es-419.

[7] [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/?hl=es-419#how_does_it_work.

[8] [En línea]. Available: https://developer.android.com/studio/.

[9] [En línea]. Available: https://www.yiiframework.com/doc/guide/2.0/en/start-databases.

[10] [En línea]. Available: https://code.tutsplus.com/tutorials/how-to-program-with-yii2-working-with-the-database-and-active-record--cms-22768.

[11] [En línea]. Available: https://forum.yiiframework.com/t/using-more-than-one-database-in-the-application/74346.

[12] [En línea]. Available: http://fabforce.eu/dbdesigner4/.

[13] [En línea]. Available: http://codewithawa.com/posts/complete-user-registration-system-using-php-and-mysql-database.

[14] [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/server#implementing-http-connection-server-protocol.

[15] [En línea]. Available: http://www.androiddeft.com/2017/11/18/push-notification-android-firebase-php/.

[16] [En línea]. Available: https://stackoverflow.com/questions/8278497/android-login-authentication-to-remote-mysql-database.

[17] [En línea]. Available: https://www.simplifiedcoding.net/android-login-and-registration-tutorial/#Connecting-to-the-Database.

[18] [En línea]. Available: http://androidcss.com/android/android-php-mysql-login-tutorial/.

[19] [En línea]. Available: http://jonsegador.com/2011/11/android-obtener-numero-telefono-desde-aplicacion/.

[20] [En línea]. Available: http://www.theappguruz.com/blog/detecting-incoming-phone-calls-in-android.

[21] [En línea]. Available: https://platzi.com/blog/notificaciones-firebase/.

[22] [En línea]. Available: https://www.codementor.io/flame3/send-push-notifications-to-android-with-firebase-du10860kb.

[23] [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/http-server-ref?hl=es-419.

[24] [En línea]. Available: https://www.simplifiedcoding.net/firebase-cloud-messaging-android/.

BLOQUE 7. ANEXOS

Por último, se añadirán todos los anexos correspondientes a este proyecto así como el código programado para ello.

7.1 Herramientas

7.1.1 Base de datos

Create.sql

```
CREATE TABLE admin (
 idadmin INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
 username VARCHAR(20) NOT NULL,
 password VARCHAR(20) NOT NULL,
 authKey VARCHAR(50) NULL,
 PRIMARY KEY(idadmin)
);

CREATE TABLE contacto (
 idUsuario INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
 nombre VARCHAR(20) NOT NULL,
 apellidos VARCHAR(45) NOT NULL,
 telefono VARCHAR(9) NOT NULL,
 token VARCHAR(50) NULL,
 PRIMARY KEY(idUsuario)
);

CREATE TABLE login (
 idlogin INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
 user VARCHAR(20) NULL,
 passwd VARCHAR(20) NULL,
 PRIMARY KEY(idlogin)
);
```

Drop.sql

```
DROP TABLE login;

DROP TABLE contacto;

DROP TABLE admin;
```

Repair.sql

```
REPAIR TABLE admin;

REPAIR TABLE contacto;

REPAIR TABLE login;
```

Optimize.sql

```
OPTIMIZE TABLE admin;

OPTIMIZE TABLE contacto;

OPTIMIZE TABLE login;
```

7.1.2 Servidor Web

Para poder hacer funcionar el servidor primero habrá que realizar los siguientes comandos.

```
sudo apt-get install php-curl
sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
sudo service apache2 restart

Y desde la carpeta de Yii:
./yii migrate/create create_agenda_table
```

- **Directorio config**

Console.php

```
<?php

$params = require(__DIR__ . '/params.php');
$db = require(__DIR__ . '/db.php');

$config = [
 'id' => 'basic-console',
 'basePath' => dirname(__DIR__),
 'bootstrap' => ['log'],
 'controllerNamespace' => 'app\commands',
 'components' => [
  'cache' => [
   'class' => 'yii\caching\FileCache',
  ],
  'log' => [
   'targets' => [
    [
     'class' => 'yii\log\FileTarget',
     'levels' => ['error', 'warning'],
    ],
   ],
  ],
  'db' => $db,
 ],
 'params' => $params,
 /*
 'controllerMap' => [
  'fixture' => [ // Fixture generation command line.
   'class' => 'yii\faker\FixtureController',
  ],
```

```
  ],
  */
];

if (YII_ENV_DEV) {
  // configuration adjustments for 'dev' environment
  $config['bootstrap'][] = 'gii';
  $config['modules']['gii'] = [
    'class' => 'yii\gii\Module',
  ];
}

return $config;
```

db.php

```
<?php

return [
  'class' => 'yii\db\Connection',
  'dsn' => 'mysql:host=localhost;dbname=agenda',
  'username' => 'root',
  'password' => 'alemilrod',
  'charset' => 'utf8',
];
```

db2.php

```
<?php

return [
  'class' => 'yii\db\Connection',
  'dsn' => 'mysql:host=localhost;dbname=agenda',
  'username' => 'root',
  'password' => 'alemilrod',
  'charset' => 'utf8',
];
```

params.php

```
<?php
```

```
return [
  'adminEmail' => 'admin@example.com',
];
```

test.php

```
<?php
$params = require(__DIR__ . '/params.php');
$db = require(__DIR__ . '/test_db.php');

/**
 * Application configuration shared by all test types
 */
return [
  'id' => 'basic-tests',
  'basePath' => dirname(__DIR__),
  'language' => 'en-US',
  'components' => [
    'db' => $db,
    'mailer' => [
      'useFileTransport' => true,
    ],
    'assetManager' => [
      'basePath' => __DIR__ . '/../web/assets',
    ],
    'urlManager' => [
      'showScriptName' => true,
    ],
    'user' => [
      'identityClass' => 'app\models\User',
    ],
    'request' => [
      'cookieValidationKey' => 'test',
      'enableCsrfValidation' => false,
      // but if you absolutely need it set cookie domain to localhost
      /*
      'csrfCookie' => [
```

```
        'domain' => 'localhost',
      ],
      */
    ],
  ],
  'params' => $params,
];
```

test_db.php

```
<?php
$db = require(__DIR__ . '/db.php');
// test database! Important not to run tests on production or development databases
$db['dsn'] = 'mysql:host=localhost;dbname=yii2_basic_tests';

return $db;
```

web.php

```
<?php

$params = require(__DIR__ . '/params.php');
$db = require(__DIR__ . '/db.php');

$config = [
  'id' => 'basic',
  'basePath' => dirname(__DIR__),
  'bootstrap' => ['log'],
  'components' => [
    'request' => [
      // !!! insert a secret key in the following (if it is empty) - this is required by cookie validation
      'cookieValidationKey' => 'asdsadsadsadsaa',
    ],
    'cache' => [
      'class' => 'yii\caching\FileCache',
    ],
    'user' => [
      'identityClass' => 'app\models\BackendUser', // backenduser es nuestra nueva clase que implementara los metodos de la nueva base de datos (admin)
      'enableAutoLogin' => true,
      'enableSession' => true,
```

```
    ],
    'errorHandler' => [
      'errorAction' => 'site/error',
    ],
    'urlManager' => [  //es necesario para la barra de navegacion tenga una estructura mas limpia
      'class' => "yii\web\UrlManager",
      'showScriptName' => true,
      'enablePrettyUrl' => true,
      ],

    'mailer' => [
      'class' => 'yii\swiftmailer\Mailer',
      // send all mails to a file by default. You have to set
      // 'useFileTransport' to false and configure a transport
      // for the mailer to send real emails.
      'useFileTransport' => true,
    ],
    'log' => [
      'traceLevel' => YII_DEBUG ? 3 : 0,
      'targets' => [
        [
          'class' => 'yii\log\FileTarget',
          'levels' => ['error', 'warning'],
        ],
      ],
    ],
    'db' => $db,  //db=bbdd para agenda
    /*
    'urlManager' => [
      'enablePrettyUrl' => false,
      'showScriptName' => true,
      'rules' => [
      ],
    ],*/

  ],
  'params' => $params,
];
```

```
if (YII_ENV_DEV) {
  // configuration adjustments for 'dev' environment
  $config['bootstrap'][] = 'debug';
  $config['modules']['debug'] = [
    'class' => 'yii\debug\Module',
    // uncomment the following to add your IP if you are not connecting from localhost.
    //'allowedIPs' => ['127.0.0.1', '::1'],
  ];

  $config['bootstrap'][] = 'gii';
  $config['modules']['gii'] = [
    'class' => 'yii\gii\Module',
    // uncomment the following to add your IP if you are not connecting from localhost.
    //'allowedIPs' => ['127.0.0.1', '::1'],
  ];
}

return $config;
```

- **Directorio Controllers**

AgendaController.php

```
<?php

namespace app\controllers;

use Yii;
use app\models\Agenda;
use app\models\AgendaSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

/**
 * AgendaController implements the CRUD actions for Agenda model.
```

```
 */
class AgendaController extends Controller //generado por gii
{
 /**
  * @inheritdoc
  */
 public function behaviors()
 {
  return [
    'verbs' => [
     'class' => VerbFilter::className(),
     'actions' => [
       'delete' => ['POST'],
     ],
    ],
  ];
 }

 /**
  * Lists all Agenda models.
  * @return mixed
  */
 public function actionIndex()
 {
  $searchModel = new AgendaSearch();
  $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

  return $this->render('index', [
    'searchModel' => $searchModel,
    'dataProvider' => $dataProvider,
  ]);
 }

 /**
  * Displays a single Agenda model.
  * @param integer $id
  * @return mixed
  */
```

```
public function actionView($id)
{
  return $this->render('view', [
    'model' => $this->findModel($id),
  ]);
}

/**
 * Creates a new Agenda model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
  $model = new Agenda();

  if ($model->load(Yii::$app->request->post()) && $model->save()) {
    return $this->redirect(['view', 'id' => $model->idUsuario]);
  } else {
    return $this->render('create', [
      'model' => $model,
    ]);
  }
}

/**
 * Updates an existing Agenda model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
  $model = $this->findModel($id);

  if ($model->load(Yii::$app->request->post()) && $model->save()) {
    return $this->redirect(['view', 'id' => $model->idUsuario]);
```

```
    } else {
      return $this->render('update', [
        'model' => $model,
      ]);
    }
  }

  /**
   * Deletes an existing Agenda model.
   * If deletion is successful, the browser will be redirected to the 'index' page.
   * @param integer $id
   * @return mixed
   */
  public function actionDelete($id)
  {
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
  }

  /**
   * Finds the Agenda model based on its primary key value.
   * If the model is not found, a 404 HTTP exception will be thrown.
   * @param integer $id
   * @return Agenda the loaded model
   * @throws NotFoundHttpException if the model cannot be found
   */
  protected function findModel($id)
  {
    if (($model = Agenda::findOne($id)) !== null) {
      return $model;
    } else {
      throw new NotFoundHttpException('The requested page does not exist.');
    }
  }
}
```

ConsultaController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;
//controlador que sirve para buscar el nombre y apellido del numero de telefono que se le pasa
//este controlador sera usada por la aplicacion de android
class ConsultaController extends Controller //el controlador es el que da lo que muestra //localhost/index.php/consulta de la clase
{
 public function beforeAction($action) { //para las peticiones POST
 $this->enableCsrfValidation = false;
 return parent::beforeAction($action);
 }

 public function actionIndex()
 {
  $request = Yii::$app->request;

  if ($request->isGet) {
   $numero = $request->get('numero');
  }



  if ($request->isPost) {
   // devuelve todos los parametros
   $post = $request->post();
   // equivalente a: $post = $_POST;
   //$numero = isset($_POST['numero']) ? $_POST['numero'] : null;
   $numero = $request->post('numero');
  }

  if ($numero == NULL){
   // devuelve todos los parametros
   $params = $request->bodyParams;
   // devuelve el parametro "numero"
   $numero = $request->getBodyParam('numero');
  }
```

```
   if ($numero != NULL ){
    $contactos = \app\models\Agenda::find()->all(); //buscamos en la tabla
    foreach($contactos as $contacto)
    {
     if($contacto->telefono == $numero){ //recorremos cada contacto buscando el numero
     //imprimimos el valor encontrado para probar que funciona
     echo $contacto->nombre . " " . $contacto->apellido;
     //http://localhost/index.php/consulta?numero=xxxxxxxx
     }
    }
   }
 }
}
```

FirebaseController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;
//controlador que sirve para devolver la vista de la gestion de notificaciones

class FirebaseController extends Controller //el controlador es el que da lo que muestra
//localhost/index.php/firebase
{
 public function beforeAction($action) {
  $this->enableCsrfValidation = false;
  return parent::beforeAction($action);
  }



  public function actionIndex()
  {
   if(($identity = Yii::$app->user->identity) != null) {

    $contactos = \app\models\Agenda::find()->all();

    return $this->render('./firebase');
```

```
   }
   else
   return $this->render('../index2');
  }
}
```

LoginController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;
// controlador usado por la aplicacion movil para login con la bbdd

class LoginController extends Controller //localhost/index.php/login
{
 public function beforeAction($action) {  //codigo para que funcione el metodo post
 $this->enableCsrfValidation = false;
 return parent::beforeAction($action);
 }

 public function actionIndex()
 {
  $request = Yii::$app->request;
  $result="false";

 // metodo get para coger usuario y contraseña del login de android no es el que vamos a usar
 if ($request->isGet) {
  $username = $request->get('username');
  // equivalent to: $get = $_GET;
  $password = $request->get('password');

  }

 if($request->isPost){
  // returns all parameters
  $post = $request->post();
  // equivalent to: $post = $_POST;
```

```
    $username = $request->post('username');
    $password = $request->post('password');
   }

   if($username != null && $password != null){

    $sql = "SELECT * FROM login WHERE ((user = user) AND (passwd = passwd))";
    $model = \app\models\Login::findBySql($sql)->all();  //modelo donde se encuentra la clase Login que devuelve la tabla

    foreach($model as $usuario)
    {
     if(($usuario->user === $username) && ($usuario->passwd === $password)){
     $result="true";      //recorremos la tabla buscando que el usuario y contraseña coincidan. Si es asi ponemos a 1 y validamos.
     }
    }
   }

   echo $result;

  //  http://localhost/index.php/login?username=alemilrod&password=0000 ejemplo url login con get

 };//fin function

}
```

SincronizarController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;
// controlador usado por la aplicacion movil para obtener un token
// este token es necesario para poder enviar notificaciones
/*Ejemplo URL
http://localhost/index.php/sincronizar?token=cninGxW6Tb4:APA91bH0UeDLV75JDx6nfVkXbIvlyCx4dR7alSHOVROQEOXzrY4naiNA7ahOvxu4fO8iMDHSeO-Q9Skj5i91tZ1UvRng3-BQtx3pc1gs87R-rCLQxtdGW0tMtpVLEGop6SD5xN_y0s-w */
```

```
class SincronizarController extends Controller //el controlador es el que da lo que muestra //localhost/index.php/sincronizar
{
 public function beforeAction($action) {
 $this->enableCsrfValidation = false;
 return parent::beforeAction($action);
 }

 public function actionIndex()
 {

  $request = Yii::$app->request;
  $result="false";

  if ($request->isGet) {
   $token = $request->get('token');
  }

  if ($request->isPost) {
   $post = $request->post();
   $token = $request->post('token');
  }

  if ($token == NULL){
   $params = $request->bodyParams;
   $token = $request->getBodyParam('token');
  }




  if($token != null){

   $sql = "SELECT * FROM contacto WHERE ((token = '$token'))";
   $model = \app\models\Agenda::findBySql($sql)->all();  //modelo donde se encuentra la clase Login que devuelve la tabla

   foreach($model as $consulta)
   {
```

```
      if($consulta != null){
        $result=true;
      }
     }
    }
    echo $result . "<br>";

    if($result=== true){
     $sql = "SELECT * FROM contacto";
     $model = \app\models\Agenda::findBySql($sql)->all();
     foreach($model as $contacto)
     {
      echo $contacto->nombre . "<br>";
      echo $contacto->apellido . "<br>";
      echo $contacto->telefono . "<br>";
     }
    }
  }
}
```

SiteController.php

```
<?php

namespace app\controllers;

use Yii;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\web\Response;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models\ContactForm;

class SiteController extends Controller
{
  /**
   * @inheritdoc
   */
```

```
public function behaviors()
{
 return [
  'access' => [
   'class' => AccessControl::className(),
   'only' => ['logout'],
   'rules' => [
    [
     'actions' => ['logout'],
     'allow' => true,
     'roles' => ['@'],
    ],
   ],
  ],
  'verbs' => [
   'class' => VerbFilter::className(),
   'actions' => [
    'logout' => ['post'],
   ],
  ],
 ];
}

/**
 * @inheritdoc
 */
public function actions()
{
 return [
  'error' => [
   'class' => 'yii\web\ErrorAction',
  ],
  'captcha' => [
   'class' => 'yii\captcha\CaptchaAction',
   'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
  ],
 ];
```

```
  }

  /**
   * Displays homepage.
   *
   * @return string
   */
  public function actionIndex()
  {
  // necesitamos que el usuario se autentique. Para poder autenticarse tendra que superar la autentificacion frente a la base de datos. mientras tanto
  // el usuario valdra null. Asi que no pararemos de mostar la misma vista
    if(($identity = Yii::$app->user->identity) != null)
    return $this->render('index');
    else
    return $this->render('index2');
  }

  /**
   * Login action.
   *
   * @return Response|string
   */
  public function actionLogin()
  {
    if (!Yii::$app->user->isGuest) {
      return $this->goHome();
    }

    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
      return $this->goBack();
    }
    return $this->render('login', [
      'model' => $model,
    ]);
  }
```

```
/**
 * Logout action.
 *
 * @return Response
 */
public function actionLogout()
{
  Yii::$app->user->logout();

  return $this->goHome();
}

/**
 * Displays contact page.
 *
 * @return Response|string
 */
public function actionContact()
{
  $model = new ContactForm();
  if ($model->load(Yii::$app->request->post()) && $model->contact(Yii::$app->params['adminEmail'])) {
    Yii::$app->session->setFlash('contactFormSubmitted');

    return $this->refresh();
  }
  return $this->render('contact', [
    'model' => $model,
  ]);
}

/**
 * Displays about page.
 *
 * @return string
 */
public function actionAbout()
{
```

```
    return $this->render('about');
  }
}
```

TokenController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;
// controlador usado por la aplicacion movil para obtener un token
// este token es necesario para poder enviar notificaciones
class TokenController extends Controller //el controlador es el que da lo que muestra //localhost/index.php/token
{
  public function beforeAction($action) {
  $this->enableCsrfValidation = false;
  return parent::beforeAction($action);
  }

  public function actionIndex()
  {

    $request = Yii::$app->request;
    $result="false";

    if ($request->isGet) {
     $numero = $request->get('numero');
     $token = $request->get('token');
    }

    if ($request->isPost) {
     $post = $request->post();
     $numero = $request->post('numero');
     $token = $request->post('token');
    }

    if ($numero == NULL && $token == NULL){
     $params = $request->bodyParams;
```

```
    $numero = $request->getBodyParam('numero');
    $token = $request->getBodyParam('token');
   }




   if($numero != null && $token != null){


    $sql = "SELECT * FROM contacto WHERE ((telefono = $numero))";
    $model = \app\models\Agenda::findBySql($sql)->all();  //modelo donde se encuentra la clase Login que devuelve la tabla



    foreach($model as $contacto)
    {
     //recorremos la tabla buscando que usuario y numero coincidan. Si es asi ponemos a 1 y validamos.
     $result=true;
     $contacto->token = NULL;
     $contacto->token = $token;
     $contacto->save();
    }
   }
   echo $result;
  }
}
```

UsuariosController.php

```
<?php

namespace app\controllers;
use yii\web\Controller;
use Yii;

class UsuariosController extends Controller //el controlador es el que da lo que muestra
{
 public function beforeAction($action) {
  $this->enableCsrfValidation = false;
  return parent::beforeAction($action);
```

```
  }


 public function actionIndex()
 {
  if(($identity = Yii::$app->user->identity) != null) {

   $contactos = \app\models\Agenda::find()->all();

   return $this->render('./usuarios');
   }
   else
   return $this->render('../index2');
 }
}
```

- Directorio migrations

m170807_112948_create_contacto_table.php

```
<?php

use yii\db\Migration;

/**
 * Handles the creation of table `contacto`.
 */
 /** archivo de migracion usado para el tfg.
 Se le pasa la tabla a crear para que lo haga yii con su script
 *
 */

class m170807_112948_create_contacto_table extends Migration
{
  /**
   * @inheritdoc
   */
  public function up()
```

```
        {
          $this->createTable('contacto', [
            'idUsuario' => $this->primaryKey(),
            'nombre' => $this->string()->notNull(),
            'apellido' => $this->string()->notNull(),
            'telefono' => $this->string()->notNull(),
            'token' => $this->string()

          ]);
        }

        /**
         * @inheritdoc
         */
        public function down()
        {
          $this->dropTable('contacto');
        }
      }
```

- **Directorio models**

Agenda.php

```
<?php


namespace app\models;

class Agenda extends \yii\db\ActiveRecord
{

 public static function tableName()
 {
   return "contacto";
 }

  public function rules()   //para validar el formulario
```

```
 {
  return [
   [["nombre", "apellido", "telefono"], "required"],  //exigimos todos los campos
   [["nombre"], "string"],
   [["apellido"], "string"],
   [["telefono"], "string", "max" => 9, "min" => 9],    //maximo numero 9 digitos
  ];
 }

 public function attributeLabels()
 {

  return[
   "id" => "Id",
   "nombre" => "Nombre",
   "apellido" => "Apellidos",
   "telefono" => "Telefono"
   ];

 }

}
```

AgendaSearch.php

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;
use yii\data\ActiveDataProvider;
use app\models\Agenda;

/**
 * AgendaSearch represents the model behind the search form about `app\models\Agenda`.
 */
class AgendaSearch extends Agenda
{
```

```
/**
 * @inheritdoc
 */
public function rules()
{
  return [
    [['idUsuario'], 'integer'],
    [['nombre', 'apellido', 'telefono', 'token'], 'safe'],
  ];
}

/**
 * @inheritdoc
 */
public function scenarios()
{
  // bypass scenarios() implementation in the parent class
  return Model::scenarios();
}

/**
 * Creates data provider instance with search query applied
 *
 * @param array $params
 *
 * @return ActiveDataProvider
 */
public function search($params)
{
  $query = Agenda::find();

  // add conditions that should always apply here

  $dataProvider = new ActiveDataProvider([
    'query' => $query,
  ]);
```

```
    $this->load($params);

    if (!$this->validate()) {
      // uncomment the following line if you do not want to return any records when validation fails
      // $query->where('0=1');
      return $dataProvider;
    }

    // grid filtering conditions
    $query->andFilterWhere([
      'idUsuario' => $this->idUsuario,
    ]);

    $query->andFilterWhere(['like', 'nombre', $this->nombre])
      ->andFilterWhere(['like', 'apellido', $this->apellido])
      ->andFilterWhere(['like', 'telefono', $this->telefono])
      ->andFilterWhere(['like', 'token', $this->token]);

    return $dataProvider;
  }
}
```

BackendUser.php

```
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "admin".
 *
 * @property integer $id
 * @property string $username
 * @property string $password
 * @property string $authKey
 */
class BackendUser extends \yii\db\ActiveRecord implements \yii\web\IdentityInterface //generamos a partir de gii y con la tabla "admin"
```

```
{
 /**
  * @inheritdoc
  */
 public static function tableName()
 {
  return 'admin';
 }

 /**
  * @inheritdoc
  */
 public function rules()
 {
  return [
   [['username', 'password'], 'required'],
   [['username', 'password', 'authKey'], 'string', 'max' => 50],
  ];
 }

 /**
  * @inheritdoc
  */
 public function attributeLabels()
 {
  return [
   'id' => 'ID',
   'username' => 'Username',
   'password' => 'Password',
   'authKey' => 'Auth Key',
  ];
 }



                                                                    pub
lic static function findIdentity($id){

                                                                    retu
```

```
rn static::findOne($id);
                                                                                }

                                                                                pub
lic static function findIdentityByAccessToken($token, $type = null){

                                                                                thr
ow new NotSupportedException();
                                                                                }

                                                                                pub
lic function getId(){

                                                                                retu
rn $this->id;
                                                                                }

                                                                                pub
lic function getAuthKey(){

                                                                                retu
rn $this->authKey;
                                                                                }

                                                                                pub
lic function validateAuthKey($authKey){

                                                                                retu
rn $this->authKey === $authKey;
                                                                                }
                                                                                pub
lic static function findByUsername($username){

                                                                                retu
rn self::findOne(['username'=>$username]);
                                                                                }

                                                                                pub
lic function validatePassword($password){

                                                                                retu
rn $this->password === $password;
                                                                                }
```

```
}
```

ContactForm.php

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;

/**
 * ContactForm is the model behind the contact form.
 */
class ContactForm extends Model
{
  public $name;
  public $email;
  public $subject;
  public $body;
  public $verifyCode;


  /**
   * @return array the validation rules.
   */
  public function rules()
  {
    return [
      // name, email, subject and body are required
      [['name', 'email', 'subject', 'body'], 'required'],
      // email has to be a valid email address
      ['email', 'email'],
      // verifyCode needs to be entered correctly
      ['verifyCode', 'captcha'],
    ];
  }

  /**
```

```
   * @return array customized attribute labels
   */
  public function attributeLabels()
  {
    return [
      'verifyCode' => 'Verification Code',
    ];
  }

  /**
   * Sends an email to the specified email address using the information collected by this model.
   * @param string $email the target email address
   * @return bool whether the model passes validation
   */
  public function contact($email)
  {
    if ($this->validate()) {
      Yii::$app->mailer->compose()
        ->setTo($email)
        ->setFrom([$this->email => $this->name])
        ->setSubject($this->subject)
        ->setTextBody($this->body)
        ->send();

      return true;
    }
    return false;
  }
}
```

Login.php

```
<?php



namespace app\models;

class Login extends \yii\db\ActiveRecord
{
```

```
 public static function tableName()
 {
   return "login";
 }

 public function rules()   //para validar el formulario
 {
   return [
     [["user", "passwd"], "required"],  //exigimos todos los campos
     [["user"], "string"],
     [["passwd"], "string"],
   ];
 }

 public function attributeLabels()
 {

   return[
     "id" => "Id",
     "user" => "user",
     "passwd" => "passwd"
     ];

 }

}
```

LoginForm.php

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;

/**
 * LoginForm is the model behind the login form.
```

```
 *
 * @property User|null $user This property is read-only.
 *
 */
class LoginForm extends Model
{
  public $username;
  public $password;
  public $rememberMe = true;

  private $_user = false;



  /**
   * @return array the validation rules.
   */
  public function rules()
  {
    return [
      // username and password are both required
      [['username', 'password'], 'required'],
      // rememberMe must be a boolean value
      ['rememberMe', 'boolean'],
      // password is validated by validatePassword()
      ['password', 'validatePassword'],
    ];
  }

  /**
   * Validates the password.
   * This method serves as the inline validation for password.
   *
   * @param string $attribute the attribute currently being validated
   * @param array $params the additional name-value pairs given in the rule
   */
  public function validatePassword($attribute, $params)
  {
    if (!$this->hasErrors()) {
```

```
    $user = $this->getUser();

    if (!$user || !$user->validatePassword($this->password)) {
      $this->addError($attribute, 'Incorrect username or password.');
    }
  }
}

/**
 * Logs in a user using the provided username and password.
 * @return bool whether the user is logged in successfully
 */
public function login()
{
  if ($this->validate()) {
    return Yii::$app->user->login($this->getUser(), $this->rememberMe ? 3600*24*30 : 0);
  }
  return false;
}

/**
 * Finds user by [[username]]
 *
 * @return User|null
 */
public function getUser()
{
  if ($this->_user === false) {
    $this->_user = BackendUser::findByUsername($this->username);
  }

  return $this->_user;
}
}
```

User.php

```
<?php
```

```
namespace app\models;

class User extends \yii\base\Object implements \yii\web\IdentityInterface
{
  public $id;
  public $username;
  public $password;
  public $authKey;
  public $accessToken;

  private static $users = [
  ];


  /**
   * @inheritdoc
   */
  public static function findIdentity($id)
  {
    return isset(self::$users[$id]) ? new static(self::$users[$id]) : null;
  }

  /**
   * @inheritdoc
   */
  public static function findIdentityByAccessToken($token, $type = null)
  {
    foreach (self::$users as $user) {
      if ($user['accessToken'] === $token) {
        return new static($user);
      }
    }

    return null;
  }

  /**
   * Finds user by username
```

```
     *
     * @param string $username
     * @return static|null
     */
    public static function findByUsername($username)
    {
        foreach (self::$users as $user) {
            if (strcasecmp($user['username'], $username) === 0) {
                return new static($user);
            }
        }

        return null;
    }

    /**
     * @inheritdoc
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * @inheritdoc
     */
    public function getAuthKey()
    {
        return $this->authKey;
    }

    /**
     * @inheritdoc
     */
    public function validateAuthKey($authKey)
    {
        return $this->authKey === $authKey;
```

```
  }

  /**
   * Validates password
   *
   * @param string $password password to validate
   * @return bool if password provided is valid for current user
   */
  public function validatePassword($password)
  {
    return $this->password === $password;
  }
}
```

- **Directorio views/agenda**

Create.php

```
<?php

use yii\helpers\Html;



/* @var $this yii\web\View */
/* @var $model app\models\Agenda */

$this->title = 'Introducir nuevo contacto';
$this->params['breadcrumbs'][] = ['label' => 'Agendas', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="agenda-create">

  <h1><?= Html::encode($this->title) ?></h1>

  <?= $this->render('_form', [
    'model' => $model,
  ]) ?>

</div>
```

_form.php

```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $model app\models\Agenda */
/* @var $form yii\widgets\ActiveForm */
?>

<div class="agenda-form">

  <?php $form = ActiveForm::begin(); ?>

  <?= $form->field($model, 'nombre')->textInput(['maxlength' => true]) ?>

  <?= $form->field($model, 'apellido')->textInput(['maxlength' => true]) ?>

  <?= $form->field($model, 'telefono')->textInput(['maxlength' => true]) ?>

  <div class="form-group">
    <?= Html::submitButton($model->isNewRecord ? 'Create' : 'Update', ['class' => $model->isNewRecord ? 'btn btn-success' : 'btn btn-primary']) ?>
  </div>

  <?php ActiveForm::end(); ?>

</div>
```

Index.php

```
<?php

use yii\helpers\Html;
use yii\grid\GridView;

/* @var $this yii\web\View */
/* @var $searchModel app\models\AgendaSearch */
```

```
/* @var $dataProvider yii\data\ActiveDataProvider */

$this->title = 'Contactos';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="agenda-index">

  <h1><?= Html::encode($this->title) ?></h1>
  <?php // echo $this->render('_search', ['model' => $searchModel]); ?>

  <p>
    <?= Html::a('Crear nuevo contacto', ['create'], ['class' => 'btn btn-success']) ?>
  </p>
  <?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
      ['class' => 'yii\grid\SerialColumn'],

      'idUsuario',
      'nombre',
      'apellido',
      'telefono',
      'token',

      ['class' => 'yii\grid\ActionColumn'],
    ],
  ]); ?>
</div>
```

Index2.php

```
<?php

/* @var $this yii\web\View */

$this->title = 'USNotificaciones';
?>
<div class="site-index">
```

```
 <div class="jumbotron">

  <p class="lead">Trabajo de fin de grado 2017-2018.</p>

  <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/site/login">Login</a></p>
 </div>

</div>
```

_search.php

```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $model app\models\AgendaSearch */
/* @var $form yii\widgets\ActiveForm */
?>

<div class="agenda-search">

 <?php $form = ActiveForm::begin([
  'action' => ['index'],
  'method' => 'get',
 ]); ?>

 <?= $form->field($model, 'idUsuario') ?>

 <?= $form->field($model, 'nombre') ?>

 <?= $form->field($model, 'apellido') ?>

 <?= $form->field($model, 'telefono') ?>

 <?= $form->field($model, 'token') ?>
```

```
  <div class="form-group">
    <?= Html::submitButton('Search', ['class' => 'btn btn-primary']) ?>
    <?= Html::resetButton('Reset', ['class' => 'btn btn-default']) ?>
  </div>

  <?php ActiveForm::end(); ?>

</div>
```

Update.php

```
<?php

use yii\helpers\Html;

/* @var $this yii\web\View */
/* @var $model app\models\Agenda */

$this->title = 'Update Agenda: ' . $model->idUsuario;
$this->params['breadcrumbs'][] = ['label' => 'Agendas', 'url' => ['index']];
$this->params['breadcrumbs'][] = ['label' => $model->idUsuario, 'url' => ['view', 'id' => $model->idUsuario]];
$this->params['breadcrumbs'][] = 'Update';
?>
<div class="agenda-update">

  <h1><?= Html::encode($this->title) ?></h1>

  <?= $this->render('_form', [
    'model' => $model,
  ]) ?>

</div>
```

View.php

```
<?php

use yii\helpers\Html;
use yii\widgets\DetailView;

/* @var $this yii\web\View */
```

```
/* @var $model app\models\Agenda */

$this->title = $model->idUsuario;
$this->params['breadcrumbs'][] = ['label' => 'Agendas', 'url' => ['index']];
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="agenda-view">

  <h1><?= Html::encode($this->title) ?></h1>

  <p>
    <?= Html::a('Update', ['update', 'id' => $model->idUsuario], ['class' => 'btn btn-primary']) ?>
    <?= Html::a('Delete', ['delete', 'id' => $model->idUsuario], [
      'class' => 'btn btn-danger',
      'data' => [
        'confirm' => 'Are you sure you want to delete this item?',
        'method' => 'post',
      ],
    ]) ?>
  </p>

  <?= DetailView::widget([
    'model' => $model,
    'attributes' => [
      'idUsuario',
      'nombre',
      'apellido',
      'telefono',
      'token',
    ],
  ]) ?>

</div>
```

Directorio view/firebase

Firebase.php

```
<!DOCTYPE html>
```

```
<html lang="es">
<head><title>Gestion de mensajes firebase cloud messaging</title>
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
 </head>
 <body>


 <div class="container">
  <div class="row">
  <div class="col-lg-6">
   <h2>Enviar notificacion</h2>
   <hr />
   <form action="" method="post">
   <div class="form-group">
    <label for="send_to">Enviar a:</label>
    <select name="send_to" id="send_to" class="form-control">
    <option value="sngle">Un dispositivo</option>
    </select>
   </div>


   <div class="form-group" id="firebase_token_group">
    <label for="firebase_token">Firebase Token:</label>
    <input type="text" required="" class="form-control" id="firebase_token" placeholder="Enter Firebase
Token" name="firebase_token">
   </div>
   <div class="form-group">
    <label for="title">Titulo:</label>
    <input type="text" required="" class="form-control" id="title" placeholder="Enter Notification Title"
name="title">
   </div>
   <div class="form-group">
    <label for="message">Mensaje:</label>
    <textarea required="" class="form-control" rows="5" id="message" placeholder="Enter Notification
Message" name="message"></textarea>
   </div>
   <div class="checkbox">
```

```
    <label><input type="checkbox"id="include_image" name="include_image">Include Image</label>
   </div>
   <div class="form-group" style="display: none" id="image_url_group">
    <label for="image_url">URL de imagen:</label>
   <input type="url" class="form-control" id="image_url" placeholder="Enter Image URL"
name="image_url">
   </div>
   <div class="checkbox">
    <label><input type="checkbox" id="include_action" name="include_action">Include Action</label>
   </div>
   <div class="form-group" style="display: none" id="action_group">
    <label for="action">Accion:</label>
    <select name="action" id="action" class="form-control">
    <option value="url">Abrir URL</option>
    </select>
   </div>
   <div class="form-group" style="display: none" id="action_destination_group">
    <label for="action_destination">Destination:</label>
   <input type="text" class="form-control" id="action_destination" placeholder="Enter Destination URL or
Activity name" name="action_destination">
   </div>


   <button type="submit" class="btn btn-lg btn-success">Enviar</button>
   <button class="btn btn-lg btn-success" href="http://localhost/index.php">Volver</button>
   </form>
  <br /></br>


  </div>
  <div class="col-lg-6">
   <?php
   if(isset($_POST['title'])){

   require_once __DIR__ . '/notification.php';
   $notification = new Notification();

   $title = $_POST['title'];
   $message = isset($_POST['message'])?$_POST['message']:'';
   $imageUrl = isset($_POST['image_url'])?$_POST['image_url']:'';
```

```
$action = isset($_POST['action'])?$_POST['action']:'';

$actionDestination = isset($_POST['action_destination'])?$_POST['action_destination']:'';

if($actionDestination ==''){
 $action = '';
}
$notification->setTitle($title);
$notification->setMessage($message);
$notification->setImage($imageUrl);
$notification->setAction($action);
$notification->setActionDestination($actionDestination);

$firebase_token = $_POST['firebase_token'];
$firebase_api = 'AIzaSyCz2GsFZ3UObZINzNeqWV7188r4ChntiK0';

$requestData = $notification->getNotificatin();

$fields = array(
 'to' => $firebase_token,
 'data' => $requestData,
);


// Asignamos los valores al mensaje POST
   $url = "https://fcm.googleapis.com/fcm/send";

$headers = array(
 'Authorization: key=' . $firebase_api,
 'Content-Type: application/json'
);

   $notification = array('title' =>$title , 'body' => $message,'sound' => 'default', 'badge' => '1');
   $data = array('title' =>$title , 'body' => $message, 'message' => $message, 'imageUrl' => $imageUrl,
'Url' => $actionDestination);
   $arrayToSend = array('to' => $firebase_token, 'notification' => $notification,'priority'=>'high', 'data'=>
$data);
```

```
    $json = json_encode($arrayToSend);



  // Codigo para abrir conexion y enviar mensaje
  $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_CUSTOMREQUEST,"POST");
    curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
    curl_setopt($ch, CURLOPT_HTTPHEADER,$headers);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    //Enviamos las peticion
    $result = curl_exec($ch);

    if ($result === FALSE) {
    die('FCM Send Error: ' . curl_error($ch));
    }
    curl_close($ch);

  echo '<h2>Resultado</h2><hr/><h3>Peticion </h3><p><pre>';
  echo json_encode($fields,JSON_PRETTY_PRINT);
  echo '</pre></p><h3>Respuesta </h3><p><pre>';
  echo $result;
  echo '</pre></p>';
  }
  ?>

 </div>
 </div>
</div>

<script>
 $('#include_image').change(function(e){
  if($(this).prop("checked")==true){
  $('#image_url_group').show();
```

```
  $("#image_url").prop('required',true);
  }else{
  $('#image_url_group').hide();
  $("#image_url").prop('required',false);



  }
 });
 $('#include_action').change(function(e){
  if($(this).prop("checked")==true){
  $('#action_group').show();
  $('#action_destination_group').show();
  $("#action_destination").prop('required',true);
  }else{
  $('#action_group').hide();
  $('#action_destination_group').hide();
  $("#action_destination").prop('required',false);



  }
 });

 $('#send_to').change(function(e){
  var selectedVal = $("#send_to option:selected").val();
  if(selectedVal=='topic'){
  $('#topic_group').show();
  $("#topic").prop('required',true);
  $('#firebase_token_group').hide();
  $("#firebase_token").prop('required',false);
  }else{
  $('#topic_group').hide();
  $("#topic").prop('required',false);
  $('#firebase_token_group').show();
  $("#firebase_token").prop('required',true);
  }
 });
</script>
</body>
```

```
</html>
```

Notification.php

```
<?php
 class Notification{
 private $title;
 private $message;
 private $image_url;
 private $action;
 private $action_destination;
 private $data;

 function __construct(){

 }

 public function setTitle($title){
 $this->title = $title;
 }

 public function setMessage($message){
 $this->message = $message;
 }

 public function setImage($imageUrl){
 $this->image_url = $imageUrl;
 }

 public function setAction($action){
 $this->action = $action;
 }

 public function setActionDestination($actionDestination){
 $this->action_destination = $actionDestination;
 }

 public function setPayload($data){
 $this->data = $data;
```

```
}

public function getNotificatin(){
$notification = array();
$notification['title'] = $this->title;
$notification['message'] = $this->message;
$notification['image'] = $this->image_url;
$notification['action'] = $this->action;
$notification['action_destination'] = $this->action_destination;
return $notification;
}
}
?>
```

- **Directorio views/site**

About.php

```
<?php

/* @var $this yii\web\View */

use yii\helpers\Html;

$this->title = 'Sobre';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-about">
  <h1><?= Html::encode($this->title) ?></h1>

  <p>Este TFG ha sido realizado por:</p>
  <p>Alumno: Alejandro Milán Rodríguez</p>
  <p>Tutor: José Manuel Fornés Rumbao</p>
</div>
```

Consulta.php

```
<?php

namespace app\models;
```

```
class Token extends \yii\db\ActiveRecord
{
 public static function tablename()
 {
  return "contactos";
 }



}
```

Contact.php

```
<?php

/* @var $this yii\web\View */
/* @var $form yii\bootstrap\ActiveForm */
/* @var $model app\models\ContactForm */

use yii\helpers\Html;
use yii\bootstrap\ActiveForm;
use yii\captcha\Captcha;

$this->title = 'Contacto';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-contact">
  <h1><?= Html::encode($this->title) ?></h1>

  <?php if (Yii::$app->session->hasFlash('contactFormSubmitted')): ?>

    <div class="alert alert-success">
      Gracias por contactarnos. Le responderemos lo antes posible.
    </div>

    <p>
      Note that if you turn on the Yii debugger, you should be able
      to view the mail message on the mail panel of the debugger.
      <?php if (Yii::$app->mailer->useFileTransport): ?>
```

```
    Because the application is in development mode, the email is not sent but saved as
    a file under <code><?= Yii::getAlias(Yii::$app->mailer->fileTransportPath) ?></code>.
    Please configure the <code>useFileTransport</code> property of the <code>mail</code>
    application component to be false to enable email sending.
  <?php endif; ?>
 </p>

<?php else: ?>

 <p>
  Si tiene alguna duda o preguntas rellene el siguiente formulario para contactarnos. Gracias.
 </p>

 <div class="row">
  <div class="col-lg-5">

   <?php $form = ActiveForm::begin(['id' => 'contact-form']); ?>

    <?= $form->field($model, 'name')->textInput(['autofocus' => true]) ?>

    <?= $form->field($model, 'email') ?>

    <?= $form->field($model, 'subject') ?>

    <?= $form->field($model, 'body')->textarea(['rows' => 6]) ?>

    <?= $form->field($model, 'verifyCode')->widget(Captcha::className(), [
      'template' => '<div class="row"><div class="col-lg-3">{image}</div><div class="col-lg-
6">{input}</div></div>',
    ]) ?>

    <div class="form-group">
     <?= Html::submitButton('Enviar', ['class' => 'btn btn-primary', 'name' => 'contact-button']) ?>
    </div>

   <?php ActiveForm::end(); ?>

  </div>
```

```
    </div>

  <?php endif; ?>
</div>
```

Error.php

```
<?php

/* @var $this yii\web\View */
/* @var $name string */
/* @var $message string */
/* @var $exception Exception */

use yii\helpers\Html;

$this->title = $name;
?>
<div class="site-error">

  <h1><?= Html::encode($this->title) ?></h1>

  <div class="alert alert-danger">
    <?= nl2br(Html::encode($message)) ?>
  </div>

  <p>
    Ha ocurrido un error mientras el servidor web estaba procesando su petición.
  </p>
  <p>
    Por favor, contáctenos si piensa que es un error de servidor. Gracias.
  </p>

</div>
```

Index.php

```
<?php

/* @var $this yii\web\View */
```

```
$this->title = 'USNotificaciones';
?>
<div class="site-index">


  <div class="jumbotron">


    <p class="lead"><b>UsNotificaciones</b></p>


    <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/agenda/create">Agregar un contacto a la base de datos</a></p>
    <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/usuarios">Agregar un usuario para la aplicacion movil</a></p>
    <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/agenda/index">Consultar usuarios en la base de datos</a></p>
    <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/firebase">Enviar una notificacion a un usuario</a></p>


  </div>


  <div class="body-content">


    <div class="row">
      <div class="col-lg-4">
        <h2>TFG 2017-2018</h2>


        <p>Autor : Alejandro Milan Rodriguez</p>
        <p>Tutor : Jose Manuel Fornes Rumbao</p>
      </div>
      <div class="col-lg-4">
      </div>
      <div class="col-lg-4">
       <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/4/43/Emblema_Universidad_de_Sevilla.png/245px-Emblema_Universidad_de_Sevilla.png" width="150" height="100" alt="Icono Universidad de Sevilla" align=right>
      </div>
    </div>


  </div>
</div>
```

Index2.php

```
<?php

/* @var $this yii\web\View */

$this->title = 'USNotificaciones';
?>
<div class="site-index">

  <div class="jumbotron">

    <p class="lead">Trabajo de fin de grado: Agenda.</p>

    <p><a class="btn btn-lg btn-success" href="http://localhost/index.php/site/login">Login</a></p>
  </div>

</div>
```

Login.php

```
<?php

/* @var $this yii\web\View */
/* @var $form yii\bootstrap\ActiveForm */
/* @var $model app\models\LoginForm */

use yii\helpers\Html;
use yii\bootstrap\ActiveForm;

$this->title = 'Autenticación';
$this->params['breadcrumbs'][] = $this->title;
?>
<div class="site-login">
  <h1><?= Html::encode($this->title) ?></h1>

  <p>Por favor rellene los siguientes campos:</p>

  <?php $form = ActiveForm::begin([
    'id' => 'login-form',
```

```
    'layout' => 'horizontal',
    'fieldConfig' => [
      'template' => "{label}\n<div class=\"col-lg-3\">{input}</div>\n<div class=\"col-lg-8\">{error}</div>",
      'labelOptions' => ['class' => 'col-lg-1 control-label'],
    ],
  ]); ?>

    <?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>

    <?= $form->field($model, 'password')->passwordInput() ?>

    <?= $form->field($model, 'rememberMe')->checkbox([
      'template' => "<div class=\"col-lg-offset-1 col-lg-3\">{input} {label}</div>\n<div class=\"col-lg-8\">{error}</div>",
    ]) ?>

    <div class="form-group">
      <div class="col-lg-offset-1 col-lg-11">
        <?= Html::submitButton('Enviar', ['class' => 'btn btn-primary', 'name' => 'login-button']) ?>
      </div>
    </div>

  <?php ActiveForm::end(); ?>

<!--  <div class="col-lg-offset-1" style="color:#999;">
    You may login with <strong>admin/admin</strong> or <strong>demo/demo</strong>.<br>
    To modify the username/password, please check out the code <code>app\models\User::$users</code>.
  </div>
  -->
</div>
```

Sincronizar.php

```
<?php

namespace app\models;

class Sincronizar extends \yii\db\ActiveRecord
{
 public static function tablename()
```

```
 {
  return "contactos";
 }



}
```

Token.php

```
<?php

namespace app\models;

class Token extends \yii\db\ActiveRecord
{
 public static function tablename()
 {
  return "contactos";
 }



}
```

Directorio views/usuarios

Errors.php

```
<?php if (count($errors) > 0) : ?>
 <div class="error">

                                                                                <?p
hp foreach ($errors as $error) : ?>

<p><?php echo $error ?></p>

                                                                                <?p
hp endforeach ?>
 </div>
<?php endif ?>
```

Server.php

```
<?php
```

```
$username = "";
$email  = "";
$errors = array();

// conexion a la base de datos
$db = mysqli_connect('localhost', 'root', 'alemilrod', 'agenda');

// registro de usuario
if (isset($_POST['reg_user'])) {
 $username = mysqli_real_escape_string($db, $_POST['username']);
 $password_1 = mysqli_real_escape_string($db, $_POST['password_1']);
 $password_2 = mysqli_real_escape_string($db, $_POST['password_2']);

 // validamos el formulario
 if (empty($username)) { array_push($errors, "Un nombre de usuario es necesario"); }
 if (empty($password_1)) { array_push($errors, "Una contraseña es necesaria"); }
 if ($password_1 != $password_2) {
                                                                                arra
y_push($errors, "Las dos contraseñas no coinciden");
 }
 if (strlen($password_1) < 6){

                                                                                arra
y_push($errors, "La contraseña tiene que tener 6 caracteres minimo");
 }

 // Comprobamos que el usuario no exista en la base de datos
 $user_check_query = "SELECT * FROM login WHERE user='$username' LIMIT 1";
 $result = mysqli_query($db, $user_check_query);
 $user = mysqli_fetch_assoc($result);

 if ($user) { // si existe
   array_push($errors, "Error. El usuario ya existe");
 }

 // Registramos al usuario si no hay errores en el formulario
 if (count($errors) == 0) {

                                                                                $pa
```

```
ssword = md5($password_1); //usamos md5 para encriptar la contraseña

                                                                    $qu
ery = "INSERT INTO login (user, passwd)

VALUES('$username', '$password')";

                                                                    my
sqli_query($db, $query);

                                                                    $_S
ESSION['username'] = $username;

                                                                    $_S
ESSION['success'] = "You are now logged in";

                                                                    ech
o "Usuario ". $username . " correctamente registrado.";

                                                                    hea
der('location: index.php');
 }
}
```

Style.css

```
* {
 margin: 0px;
 padding: 0px;
}
body {
 font-size: 120%;
 background: #F8F8FF;
}

.header {
 width: 30%;
 margin: 50px auto 0px;
 color: white;
 background: #5F9EA0;
 text-align: center;
```

```
 border: 1px solid #B0C4DE;
 border-bottom: none;
 border-radius: 10px 10px 0px 0px;
 padding: 20px;
}
form, .content {
 width: 30%;
 margin: 0px auto;
 padding: 20px;
 border: 1px solid #B0C4DE;
 background: white;
 border-radius: 0px 0px 10px 10px;
}
.input-group {
 margin: 10px 0px 10px 0px;
}
.input-group label {
 display: block;
 text-align: left;
 margin: 3px;
}
.input-group input {
 height: 30px;
 width: 93%;
 padding: 5px 10px;
 font-size: 16px;
 border-radius: 5px;
 border: 1px solid gray;
}
.btn {
 padding: 10px;
 font-size: 15px;
 color: white;
 background: #5F9EA0;
 border: none;
 border-radius: 5px;
}
.error {
```

```
 width: 92%;
 margin: 0px auto;
 padding: 10px;
 border: 1px solid #a94442;
 color: #a94442;
 background: #f2dede;
 border-radius: 5px;
 text-align: left;
}
.success {
 color: #3c763d;
 background: #dff0d8;
 border: 1px solid #3c763d;
 margin-bottom: 20px;
}
```

Usuarios.php

```
<?php include('server.php') ?>
<!DOCTYPE html>
<html>
<head>
 <title>Registro en la aplicacion</title>
 <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
 <div class="header">

                                                                                   <h2
>Registrar usuario</h2>

                                                                                   <i
mg
src="https://upload.wikimedia.org/wikipedia/commons/thumb/4/43/Emblema_Universidad_de_Sevilla.png/
245px-Emblema_Universidad_de_Sevilla.png" width="150" height="100" alt="Icono Universidad de
Sevilla" align=right>
 </div>


 <form method="post" action="usuarios">

                                                                                   <?p
hp include('errors.php'); ?>
```

```
                                                                    <di
v class="input-group">

<label>Usuario<br /></label>
</br>

<input type="text" name="username" value="<?php echo $username; ?>">

                                                                    </d
iv>

                                                                    <di
v class="input-group">

<label>Contraseña<br/></label>
</br>

<input type="password" name="password_1">

                                                                    </d
iv>

                                                                    <di
v class="input-group">

<label>Confirmar contraseña<br/></label>
</br>

<input type="password" name="password_2">

<br/></br>

                                                                    </d
iv>

                                                                    <di
v class="input-group">

<button type="submit" class="btn btn-lg btn-success" name="reg_user">Registrar</button>

                                                                    </d
iv>
                                                                    <br
/></br>

<p><a class="btn btn-lg btn-success" href="http://localhost/index.php">Volver</a></p>
```

```
</form>
</body>
</html>
```

7.1.3 Aplicación móvil

- **App/manifests**

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="androidloginandregistration.alejandro.com.usnotificaciones">

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".User"
    android:label="User">
    <intent-filter>
        <action android:name="androidloginandregistration.alejandro.com.usnotificaciones.User" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

<activity
    android:name=".ContactosActivity"
    android:label="Contactos">
    <intent-filter>
        <action android:name="androidloginandregistration.alejandro.com.usnotificaciones.Contactos" />
    </intent-filter>
</activity>


<activity android:name=".SplashActivity" android:label="SplashActivity"/>
<service
    android:name=".MyFirebaseMessagingService">
    <intent-filter>

        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
<receiver android:name=".PhoneStateReceiver">
    <intent-filter>
        <action android:name="android.intent.action.PHONE_STATE" />
    </intent-filter>
</receiver>

</application>
```

```
</manifest>
```

- **App/Java**

Contacto.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

//Clase contacto que nos servira para obtener los datos de la tabla
public class Contacto {

   private int id;
   private String nombre;
   private String apellidos;
   private String telefono;

   public Contacto(int id, String nombre, String apellidos, String telefono) {
      this.id = id;
      this.nombre = nombre;
      this.apellidos = apellidos;
      this.telefono = telefono;
   }

   public int getID() { return id; }
   public void setID(int id) { this.id = id; }


   public String getNOMBRE() { return nombre; }
   public void setNOMBRE(String nombre) { this.nombre = nombre; }


   public String getAPELLIDOS() {
      return apellidos;
   }
   public void setAPELLIDOS(String apellidos) {
      this.apellidos = apellidos;
   }
```

```
    public String  getTELEFONO() { return telefono; }
    public void setTELEFONO(String telefono) { this.telefono = telefono; }


}
```

ContactosActivity.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.database.Cursor;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.EditText;
import android.widget.TextView;

import java.util.List;

public class ContactosActivity  extends AppCompatActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_contactos);
        TextView sustituir = findViewById(R.id.contacto_reemplazo);
        SQLConfigManager db = new SQLConfigManager(this);

        // Leemos todos los contactos
        List<Contacto> contacto = db.getAllContacts();
        sustituir.setText("Nombre              Apellido              Telefono");
        sustituir.append("\n");
        for (Contacto auxiliar : contacto) {
                sustituir.append(auxiliar.getNOMBRE());
                sustituir.append("\t\t\t\t");
                sustituir.append(auxiliar.getAPELLIDOS());
                sustituir.append("\t\t\t\t");
                sustituir.append(auxiliar.getTELEFONO());
                sustituir.append("\n");
```

```
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_user, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

MainActivity.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.app.Notification;
import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.content.BroadcastReceiver;
import android.content.ContentProviderOperation;
import android.content.ContentProviderResult;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
```

```
import android.content.IntentFilter;
import android.content.OperationApplicationException;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Environment;
import android.os.RemoteException;
import android.provider.ContactsContract;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.NotificationCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.content.LocalBroadcastManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.math.BigInteger;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

```
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    // CONNECTION_TIMEOUT y READ_TIMEOUT son milisegundos. Para la conexion con el servidor
    public static final int CONNECTION_TIMEOUT = 10000;
    public static final int READ_TIMEOUT = 15000;
    public String IP_SD_CONFIG;            //para leer el fichero donde esta almacenada la IP
    public int autenticado;          //variable para no tener que re auntenticarse otra vez

    EditText username;
    EditText password;
    Button boton_login;
    Button boton_cancel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        username = findViewById(R.id.editText);
        password = findViewById(R.id.editText2);
        boton_cancel = findViewById(R.id.button2);
        boton_login = findViewById(R.id.button);

        //comprobamos al "crear" el main que no hemos recibido una notificacion
        setNotificationData(getIntent().getExtras());

        //si el usuario ha pulsado cancelar, terminamos la actividad
        boton_cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
```

```
        IP_SD_CONFIG=read_from_sd();
    }

    // funcion para comprobar que no hemos recibido una notificacion
    private void setNotificationData(Bundle extras) {
        if (extras == null)
            return;
        //comprobamos que la notificacion contenga alguna "key"
        if (extras.containsKey("title") || extras.containsKey("message") || extras.containsKey("imageUrl") || extras.containsKey("Url")){
            Intent intent = new Intent(MainActivity.this, SplashActivity.class);
            String titulo=null;
            String mensaje = null;
            String imageUrl=null;
            String Url=null;

            if(extras.containsKey("title"))
            titulo= extras.get("title").toString();

            if(extras.containsKey("message"))
            mensaje = extras.get("message").toString();

            if(extras.containsKey("imageUrl"))
            imageUrl = extras.get("imageUrl").toString();

            if(extras.containsKey("Url"))
            Url = extras.get("Url").toString();

            intent.putExtra("title", titulo);
            intent.putExtra("message", mensaje);
            intent.putExtra("imageUrl", imageUrl);
            intent.putExtra("Url", Url);

            startActivity(intent);
            MainActivity.this.finish();
        }
```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_login, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

// Leemos desde el directorio /sdcard/Downloads el archivo config para saber la IP del servidor
public String read_from_sd(){
    String IP=null;

    // Directorio  /storage/emulated/0
    File externalDir = Environment.getExternalStorageDirectory();

    // Directorio /storage/emulated/0/Downloads
    File downloadPublicDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);

    //Elegimos el fichero
    File file = new File(downloadPublicDir,"config");
```

```
    StringBuilder text = new StringBuilder();

    try {
        BufferedReader br = new BufferedReader(new FileReader(file));
        String line;
        while ((line = br.readLine()) != null) {
            IP=line.toString();
            text.append(line);
            text.append('\n');
        }
        br.close();
    }
    catch (IOException e) {

    }
    return IP;
}

// Accion de cuando pulsemos en Login
public void checkLogin(View arg0) {

    final String usuario = username.getText().toString();
    //ciframos con md5 la contraseña obtenida para no enviarla "al desnudo"
    final String passwd = MD5_Hash(password.getText().toString());
    //Llamamos a la funcion que sincroniza con el servidor
    new AsyncLogin().execute(usuario, passwd);

}

//Asyntask encargado de autenticar al usuario
private class AsyncLogin extends AsyncTask<String, String, String> {
    ProgressDialog pdLoading = new ProgressDialog(MainActivity.this);
    HttpURLConnection conn;
    URL url = null;
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
```

```
    //Este mensaje correra en el hilo principal
    pdLoading.setMessage("\tAutenticando...");
    pdLoading.setCancelable(false);
    pdLoading.show();

}

@Override
protected String doInBackground(String... params) {
    try {

        // URL del servidor
        url = new URL("http://"+IP_SD_CONFIG+"/index.php/login");

    } catch (MalformedURLException e) {
        e.printStackTrace();
        return "exception";
    }
    try {
        // Clase HttpURLConnection para enviar y recibir datos de php y mysql
        conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(READ_TIMEOUT);
        conn.setConnectTimeout(CONNECTION_TIMEOUT);
        conn.setRequestMethod("POST");

        conn.setDoInput(true);
        conn.setDoOutput(true);

        // Parametros a enviar a la URL
        Uri.Builder builder = new Uri.Builder()
                .appendQueryParameter("username", params[0])
                .appendQueryParameter("password", params[1]);
        String query = builder.build().getEncodedQuery();

        // Realiamos la conexion y empezamos a enviar datos
        OutputStream os = conn.getOutputStream();
```

```
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(os, "UTF-8"));
        writer.write(query);
        writer.flush();
        writer.close();
        os.close();
        conn.connect();

    } catch (IOException e1) {
        e1.printStackTrace();
        return "exception";
    }

    try {

        int response_code = conn.getResponseCode();

        // Conexion correcta
        if (response_code == HttpURLConnection.HTTP_OK) {

            // Leemos la respuesta de lservidor
            InputStream input = conn.getInputStream();
            BufferedReader reader = new BufferedReader(new InputStreamReader(input));
            StringBuilder result = new StringBuilder();
            String line;

            while ((line = reader.readLine()) != null) {
                result.append(line);
            }

            // Lo pasamos al postexecute
            return (result.toString());

        } else {
            return ("unsuccessful");
        }

    } catch (IOException e) {
```

```
            e.printStackTrace();
            return "exception";
        } finally {
            conn.disconnect();
        }


    }

    @Override
    protected void onPostExecute(String result) {

        //Este método va a correr en el hilo principal asi que vendra bien para cambiar valores y demás

        pdLoading.dismiss();

        // Comprobamos que el resultado ha sido true o que nos hemos autenticado antes
        // Si es asi pasamos a la actividad User
        if (result.equalsIgnoreCase("true")|| autenticado==1) {

            Intent intent = new Intent(MainActivity.this, User.class);
            // Le pasamos a la actividad la direccion IP
            intent.putExtra("IP_SD", IP_SD_CONFIG);
            autenticado = 1;
            startActivity(intent);
            MainActivity.this.finish();



        } else if (result.equalsIgnoreCase("false")) {

            // If username and password does not match display a error message
            Toast.makeText(MainActivity.this, "Usuario o contraseña incorrecta",
Toast.LENGTH_LONG).show();

        } else if (result.equalsIgnoreCase("exception") || result.equalsIgnoreCase("unsuccessful")) {

            Toast.makeText(MainActivity.this, "Error de conexion. Compruebe si el fichero config esta en
/sdcard/Download/\"!", Toast.LENGTH_LONG).show();
```

```
        }
    }

}

//funcion para codificar la contraseña con md5
public static String MD5_Hash(String s) {
    MessageDigest m = null;

    try {
        m = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }

    m.update(s.getBytes(),0,s.length());
    String hash = new BigInteger(1, m.digest()).toString(16);
    return hash;
}
}
```

MyFirebaseMessagingService.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.net.Uri;
import android.support.v4.app.NotificationCompat;
import android.util.Log;

import com.google.firebase.messaging.FirebaseMessagingService;
```

```
import com.google.firebase.messaging.RemoteMessage;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Map;
//Clase para recibir mensajes y notificarlo
public class MyFirebaseMessagingService extends FirebaseMessagingService {

   private static final String TAG = "MyFirebaseMsgService";

   @Override
   public void onMessageReceived(RemoteMessage remoteMessage) {
      final Context ctx = this;

      if (remoteMessage.getData().size() > 0) {
         Log.e(TAG, "Data Payload: " + remoteMessage.getData().toString());
         try {
            JSONObject json = new JSONObject(remoteMessage.getData().toString());

            for (Map.Entry<String, String> entry : remoteMessage.getData().entrySet()) {
               Log.e("Test", "Key = " + entry.getKey() + ", Value = " + entry.getValue() );
               Log.e(TAG, "collapsekey: " + remoteMessage.getCollapseKey());
               Log.e(TAG, "from: " + remoteMessage.getFrom());
               Log.e(TAG, "message id: " + remoteMessage.getMessageId());
               Log.e(TAG, "message type:: " + remoteMessage.getMessageType());
               Log.e(TAG, "to: " + remoteMessage.getTo());
               Log.e(TAG, "send time: " + remoteMessage.getSentTime());
               Log.e(TAG, "ttl: " + remoteMessage.getTtl());
               Log.e(TAG, "title: " + remoteMessage.getNotification().getTitle());
               Log.e(TAG, "body: " + remoteMessage.getNotification().getBody());
               Log.e(TAG, "click action: " + remoteMessage.getNotification().getClickAction());
               Log.e(TAG, "color: " + remoteMessage.getNotification().getColor());
               Log.e(TAG, "icon: " + remoteMessage.getNotification().getIcon());
```

```
            }
            sendPushNotification(json, remoteMessage);
        } catch (Exception e) {
            Log.e(TAG, "Exception: " + e.getMessage());
        }
    }
}

//Este metodo mostrara la notificacion
//firebase cloud messaging
private void sendPushNotification(JSONObject json, RemoteMessage notification) {
    Log.e(TAG, "Notification JSON " + json.toString());
    try {
        JSONObject data = json.getJSONObject("data");

        String title = data.getString("title");
        String message = data.getString("message");
        String imageUrl = data.getString("imageUrl");
        String Url = data.getString("Url");
        //Convertimos url en imagen
        Bitmap image = getBitmapfromUrl(imageUrl);

        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
            PendingIntent.FLAG_ONE_SHOT);
        Uri                                                                    defaultSoundUri=
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)
            .setContentTitle(title)
            .setContentText(message)
            .setContentText(imageUrl)
            .setContentText(Url)
            .setStyle(new NotificationCompat.BigPictureStyle()
                .bigPicture(image))/*Notification with Image*/
            .setLargeIcon(image)/*Notification icon image*/
            .setAutoCancel(true)
            .setSound(defaultSoundUri)
```

```
                .setContentIntent(pendingIntent);
        NotificationManager notificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());

    } catch (JSONException e) {
        Log.e(TAG, "Json Exception: " + e.getMessage());
    } catch (Exception e) {
        Log.e(TAG, "Exception: " + e.getMessage());
    }
}

public Bitmap getBitmapfromUrl(String imageUrl) {
    try {
        URL url = new URL(imageUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setDoInput(true);
        connection.connect();
        InputStream input = connection.getInputStream();
        Bitmap bitmap = BitmapFactory.decodeStream(input);
        return bitmap;

    } catch (Exception e) {
        e.printStackTrace();
        return null;

    }
}

//Nuevo metodo para obtener el token
@Override
public void onNewToken(String mToken) {
    super.onNewToken(mToken);
    Log.e("TOKEN",mToken);
}
}
```

PhoneStateReceiver.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.content.BroadcastReceiver;
import android.content.ContentProviderOperation;
import android.content.ContentProviderResult;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.Environment;
import android.provider.ContactsContract;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.VolleyLog;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.iid.FirebaseInstanceId;
import com.google.firebase.iid.InstanceIdResult;



import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.HashMap;
import java.util.Map;

import static android.content.ContentValues.TAG;

/**
 * Created by Alejandro on 17/08/2017.
 */
//Clase encargada de gestionar una llamada entrante
public class PhoneStateReceiver extends BroadcastReceiver {
   public String resultado;
   String incomingNumber = null;
   public String IP_SD_CONFIG;
   public int flag=0;

   @Override
   public void onReceive(Context context, Intent intent) {

      try {
         IP_SD_CONFIG = read_from_sd();

         String state = intent.getStringExtra(TelephonyManager.EXTRA_STATE);
         String incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER);

         if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)) { //llaman
            SQLConfigManager db = new SQLConfigManager(context);
            resultado = db.recuperarN_CONTACTO(incomingNumber); //comprobamos en la base de datos local que exista
            if (resultado.length() > 1) {   //si se ha encontrado algo
                  //añadimos el contacto a los contactos del movil y lo eliminamos, tiempo suficiente
                  // para que el call ui cambie
                  addContact(context, resultado, incomingNumber);
                  deleteContact(context, incomingNumber, resultado);
            } else {
               sendNumberToServer(incomingNumber, context);    //si no enviamos al servidor
            }
         }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //funcion para leer la ip del servidor
    public String read_from_sd() {
        String IP = null;

        File externalDir = Environment.getExternalStorageDirectory();
        File downloadPublicDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
        File file = new File(downloadPublicDir, "config");

        StringBuilder text = new StringBuilder();
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            String line;
            while ((line = br.readLine()) != null) {
                IP = line.toString();
                text.append(line);
                text.append('\n');
            }
            br.close();
        } catch (IOException e) {
        }
        return IP;
    }

    //funcion volley para buscar el servidor en la base de datos
    private void sendNumberToServer(final String number, final Context context) {
        Log.d(TAG, "Dentro de la funcion");
        if (number != null) {
            String URL_REGISTER_DEVICE = "http://" + IP_SD_CONFIG + "/index.php/consulta";
            StringRequest stringRequest = new StringRequest(Request.Method.POST, URL_REGISTER_DEVICE,
                    new Response.Listener<String>() {
                        @Override
```

```
                public void onResponse(String response) {
                    VolleyLog.v("Response:%n %s", response);
                        resultado = response.toString();

                    if (resultado != null){
                        if (!resultado.isEmpty()) {
                            //Toast.makeText(context, "Usuario no registrado en la base de datos local.\nSincronicela por favor.", Toast.LENGTH_SHORT).show();
                            addContact(context, resultado, number);
                            flag=1;
                        }
                    }

                    Log.d(TAG, "Response: " + resultado);
                    //while (resultado == null); //esperamos a que de respuesta o acabe el tiempo de peticion

                    if(flag==1) //eliminamos en caso de que hayamos añadido el contacto
                        deleteContact(context, number, resultado);
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    VolleyLog.d(TAG, "Error: " + error.getMessage());
                }
            }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("numero", number);
            return params;
        }
    };
    RequestQueue requestQueue = Volley.newRequestQueue(context.getApplicationContext());
    requestQueue.add(stringRequest);
  }
}
```

```
   //codigo para añadir el contacto temporalmente, funciona
   private void addContact(Context context, String nombre, String telefono) {
      ArrayList<ContentProviderOperation> operationList = new ArrayList<ContentProviderOperation>();

operationList.add(ContentProviderOperation.newInsert(ContactsContract.RawContacts.CONTENT_URI)
            .withValue(ContactsContract.RawContacts.ACCOUNT_TYPE, null)
            .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, null)
            .build());


      // first and last names
      operationList.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
            .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
            .withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
            .withValue(ContactsContract.CommonDataKinds.StructuredName.GIVEN_NAME, nombre)
            .build());


      operationList.add(ContentProviderOperation.newInsert(ContactsContract.Data.CONTENT_URI)
            .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
            .withValue(ContactsContract.Data.MIMETYPE,
ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
            .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER, telefono)
            .withValue(ContactsContract.CommonDataKinds.Phone.TYPE,
ContactsContract.CommonDataKinds.Phone.TYPE_HOME)
            .build());
      try {
         ContentProviderResult[] results =
context.getContentResolver().applyBatch(ContactsContract.AUTHORITY, operationList);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   //codigo para borrar el contacto despues de la llamada
   public static boolean deleteContact(Context ctx, String phone, String name) {
      Uri contactUri = Uri.withAppendedPath(ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
Uri.encode(phone));
      Cursor cur = ctx.getContentResolver().query(contactUri, null, null, null, null);
      try {
         if (cur.moveToFirst()) {
```

```
            do {
                if
(cur.getString(cur.getColumnIndex(ContactsContract.PhoneLookup.DISPLAY_NAME)).equalsIgnoreCas
e(name)) {
                    String                                    lookupKey                                    =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));
                    Uri  uri  =  Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_LOOKUP_URI,
lookupKey);
                    ctx.getContentResolver().delete(uri, null, null);
                    return true;
                }

            } while (cur.moveToNext());
        }

    } catch (Exception e) {
        System.out.println(e.getStackTrace());
    }
    return false;
  }
}
```

SplashActivity.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.Image;
import android.net.Uri;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
```

```
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
//Actividad cuando se recibe una notificacion. Tendra un diseño por defecto se reciba lo que se reciba
public class SplashActivity extends AppCompatActivity {

   String titulo = null;
   String mensaje = null;
   String imageUrl = null;
   String Url = null;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_splash);
      TextView texto = findViewById(R.id.textView);
      TextView url = findViewById(R.id.editText1);
      //obtenemos del main los keys de las notificaciones
      if (getIntent().getExtras() != null) {
         titulo = getIntent().getExtras().getString("title");
         mensaje = getIntent().getExtras().getString("message");
         imageUrl = getIntent().getExtras().getString("imageUrl");
         Url = getIntent().getExtras().getString("Url");

         if(Url != null)
            url.setText(Url);

         texto.setText("\nDescripción del mensaje:");
         texto.append("\n" + "-Titulo: ");
```

```
        if(titulo!= null)
            texto.append(titulo);

        texto.append("\n" + "-Mensaje: ");

        if(mensaje!=null)
            texto.append(mensaje);

        if(imageUrl!=null)
            new UpdateTask().execute(imageUrl);

    }
}

//Asyntask para recuperar la imagen recibida y modificarla por la de defecto
private class UpdateTask extends AsyncTask<String, String, Bitmap> {
    protected Bitmap doInBackground(String... urls) {
        Bitmap bitmap=null;
        if (imageUrl != null) {
            try {
                bitmap = BitmapFactory.decodeStream((InputStream) new URL(imageUrl).getContent());
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return bitmap;
    }
    @Override
    protected void onPostExecute(Bitmap result) {
        if (result!=null){
            ImageView imagen = findViewById(R.id.imageView2);
            imagen.setImageBitmap(result);
        }
    }
}
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_user, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    //Si el usuario hace click en la url abriremos el navegador
    public void Open_Browser (View arg0) {
        if(Url!=null){
            //Comprobamos que la url empiece con http o https si no dara error al intentar abrirlo
            if (!Url.startsWith("http://") && !Url.startsWith("https://"))
                Url = "http://" + Url;
            Intent browser = new Intent(Intent.ACTION_VIEW , Uri.parse(Url));
            startActivity(browser);
        }
        else
            Toast.makeText(getApplicationContext().getApplicationContext(), "El mensaje recibido no contenia
URL", Toast.LENGTH_LONG).show();
    }
}
```

SQLConfigManager.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

import static android.content.ContentValues.TAG;

/**
 * Created by Alejandro on 07/10/2017.
 */
//Clase para la base de datos local
public class SQLConfigManager extends SQLiteOpenHelper {
    private static final int VERSION_BASEDATOS = 1;
    private static final String NOMBRE_BASEDATOS = "mibasedatos.db";
    private static final String TABLA_CONTACTOS = "CREATE TABLE IF NOT EXISTS contactos " +
            " (_id INTEGER PRIMARY KEY, nombre TEXT, apellidos TEXT, telefono TEXT)";

     private static final String SQL_DROP_CONFIG = "drop table if exists contactos";

    public SQLConfigManager(Context context) {
        super(context, NOMBRE_BASEDATOS, null, VERSION_BASEDATOS);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(TABLA_CONTACTOS);
    }

    @Override
```

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

public boolean insertarCONTACTO(int id, String nombre, String apellidos, String telefono) {
    long salida = 0;
    SQLiteDatabase db = getWritableDatabase();
    if (db != null) {
        ContentValues valores = new ContentValues();
            valores.put("_id", id);
            valores.put("nombre", nombre);
            valores.put("apellidos", apellidos);
            valores.put("telefono", telefono);
            salida = db.insert("contactos", null, valores);
    }
    db.close();
    return (salida > 0);
}

public String recuperarN_CONTACTO(String telefono) {
    SQLiteDatabase db = getReadableDatabase();
    int flag=0;
    String nombre="";
    String apellidos="";
    String resultado;
    String[] valores_recuperar = {"_id", "nombre", "apellidos", "telefono"};
    Cursor c = db.query("contactos", valores_recuperar, null, null, null, null, null, null);
    c.moveToFirst();
    do {
        Contacto contacto_buscado = new Contacto(c.getInt(0), c.getString(1), c.getString(2), c.getString(3));

        if (contacto_buscado.getTELEFONO().equals(telefono)) {
            nombre = contacto_buscado.getNOMBRE();
            apellidos = contacto_buscado.getAPELLIDOS();
        }
    } while (c.moveToNext());
    db.close();
```

```
    c.close();
    resultado = nombre+ " " +apellidos;
    return resultado;
}

public SQLiteDatabase bdmi() {
    SQLiteDatabase db = getReadableDatabase();
    return db;
}

public List<Contacto> getAllContacts() {
    List<Contacto> contactList = new ArrayList<Contacto>();
    // Select All Query
    String selectQuery = "SELECT  * FROM contactos";

    SQLiteDatabase db = this.getWritableDatabase();
    Cursor cursor = db.rawQuery(selectQuery, null);

    // looping through all rows and adding to list
    if (cursor.moveToFirst()) {
        int i=0;
        do {
            Contacto contact = new Contacto(i, "1", "1", "1");
            contact.setID(Integer.parseInt(cursor.getString(0)));
            contact.setNOMBRE(cursor.getString(1));
            contact.setAPELLIDOS(cursor.getString(2));
            contact.setTELEFONO(cursor.getString(3));
            // Adding contact to list
            contactList.add(contact);
            i++;
        } while (cursor.moveToNext());
    }

    // return contact list
    return contactList;
}
```

```
}
```

User.java

```
package androidloginandregistration.alejandro.com.usnotificaciones;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.os.AsyncTask;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.iid.FirebaseInstanceId;
import com.google.firebase.iid.InstanceIdResult;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
```

```
import static android.content.ContentValues.TAG;

public class User extends AppCompatActivity {

   //para registrar el nº de telefono
   //URL para RegisterDevice.php
   public static final int CONNECTION_TIMEOUT = 10000;
   public static final int READ_TIMEOUT = 15000;
   public String IP_SD_CONFIG;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_user);

      //Button boton_register_token = findViewById(R.id.button);
      //Button boton_sincronizar = findViewById(R.id.button3);
      IP_SD_CONFIG = getIntent().getExtras().getString("IP_SD");

      eliminar_base_datos();
      sincronizar_bbdd_automatica();

   }

   //funcion para registrar el token a partir del numero de telefono
   public void registrar_token(View arg0){
      //El servicio de Firebase nos dara el token
      FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new
OnSuccessListener<InstanceIdResult>() {
         EditText number = (EditText)findViewById(R.id.editText);
         @Override
         public void onSuccess(InstanceIdResult instanceIdResult) {
            String mToken = instanceIdResult.getToken();
            new AsyncToken().execute(number.getText().toString(), mToken);
         }
      });
   }
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_user, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

// Al entrar eliminamos al base de datos
public void eliminar_base_datos(){

    SQLConfigManager MDB;
    MDB = new SQLConfigManager(getApplicationContext());
    SQLiteDatabase db = MDB.getWritableDatabase();
    MDB.onUpgrade(db, 1,1);
    MDB.close();

}

//misma funcion para sincronizar con el servidor que en el main
private class AsyncToken extends AsyncTask<String, String, String> {
    HttpURLConnection conn;
    URL url = null;
```

```
@Override
protected void onPreExecute() {
    super.onPreExecute();
}

@Override
protected String doInBackground(String... params) {
    try {

        // Enter URL address where your php file resides
        url = new URL("http://"+IP_SD_CONFIG+"/index.php/token");

    } catch (MalformedURLException e) {
        e.printStackTrace();
        return "exception";
    }
    try {
        conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(READ_TIMEOUT);
        conn.setConnectTimeout(CONNECTION_TIMEOUT);
        conn.setRequestMethod("POST");

        conn.setDoInput(true);
        conn.setDoOutput(true);

        Uri.Builder builder = new Uri.Builder()
                .appendQueryParameter("numero", params[0])
                .appendQueryParameter("token", params[1]);
        String query = builder.build().getEncodedQuery();

        OutputStream os = conn.getOutputStream();
        BufferedWriter writer = new BufferedWriter(
                new OutputStreamWriter(os, "UTF-8"));
        writer.write(query);
        writer.flush();
        writer.close();
```

```
            os.close();
            conn.connect();

        } catch (IOException e1) {
            e1.printStackTrace();
            return "exception";
        }

        try {

            int response_code = conn.getResponseCode();

            if (response_code == HttpURLConnection.HTTP_OK) {
                InputStream input = conn.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(input));
                StringBuilder result = new StringBuilder();
                String line;

                while ((line = reader.readLine()) != null) {
                    result.append(line);
                }

                return (result.toString());

            } else {
                return ("unsuccessful");
            }
        } catch (IOException e) {
            e.printStackTrace();
            return "exception";
        } finally {
            conn.disconnect();
        }
    }

    @Override
    protected void onPostExecute(String result) {
      if (result.equalsIgnoreCase("false")) {
```

```
            Toast.makeText(getApplicationContext().getApplicationContext(), "Token o numero de telefono
incorrecto", Toast.LENGTH_LONG).show();
        } else if (result.equalsIgnoreCase("exception") || result.equalsIgnoreCase("unsuccessful")) {


            Toast.makeText(getApplicationContext().getApplicationContext(), "Error de conexion.
Compruebe si el fichero config esta en /sdcard/Download/\"!", Toast.LENGTH_LONG).show();


        } else {
            Toast.makeText(getApplicationContext().getApplicationContext(), "Token registrado",
Toast.LENGTH_LONG).show();
        }
      }
   }


   // Despues de eliminar la base de datos antigua, la sincronizamos con esta funcion
   public void sincronizar_bbdd_automatica(){
      FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new
OnSuccessListener<InstanceIdResult>() {
         @Override
         public void onSuccess(InstanceIdResult instanceIdResult) {
            String mToken = instanceIdResult.getToken();
            SQLConfigManager MDB;
            MDB = new SQLConfigManager(getApplicationContext());
            new AsyncDDBB().execute(mToken);
         }
      });
   };


   // funcion por si el usuario quiere sincronizar la base de datos manualmente
   public void sincronizar_bbdd_manual(View arg0) {
      FirebaseInstanceId.getInstance().getInstanceId().addOnSuccessListener(new
OnSuccessListener<InstanceIdResult>() {
         @Override
         public void onSuccess(InstanceIdResult instanceIdResult) {
            String mToken = instanceIdResult.getToken();
            SQLConfigManager MDB;
            MDB = new SQLConfigManager(getApplicationContext());
            new AsyncDDBB().execute(mToken);
         }
```

```
        });
    }

    //Asyntask para sincronizar los usuarios
    private class AsyncDDBB extends AsyncTask<String, String, String> {
        HttpURLConnection conn;
        URL url = null;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
        }

        @Override
        protected String doInBackground(String... params) {
            try {
                url = new URL("http://"+IP_SD_CONFIG+"/index.php/sincronizar");

            } catch (MalformedURLException e) {
                e.printStackTrace();
                return "exception";
            }
            try {
                conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(READ_TIMEOUT);
                conn.setConnectTimeout(CONNECTION_TIMEOUT);
                conn.setRequestMethod("POST");

                conn.setDoInput(true);
                conn.setDoOutput(true);

                Uri.Builder builder = new Uri.Builder()
                        .appendQueryParameter("token", params[0]);
                String query = builder.build().getEncodedQuery();

                OutputStream os = conn.getOutputStream();
                BufferedWriter writer = new BufferedWriter(
                        new OutputStreamWriter(os, "UTF-8"));
```

```
            writer.write(query);
            writer.flush();
            writer.close();
            os.close();
            conn.connect();

        } catch (IOException e1) {
            e1.printStackTrace();
            return "exception";
        }

        try {
            int response_code = conn.getResponseCode();

            if (response_code == HttpURLConnection.HTTP_OK) {

                InputStream input = conn.getInputStream();
                BufferedReader reader = new BufferedReader(new InputStreamReader(input));
                StringBuilder result = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null) {
                    result.append(line);
                    Log.d(TAG, "Response: " +line);
                }
                // Leemos los datos que devuelve el servidor separados por un intro
                String[] strings = result.toString().split("<br>");
                SQLConfigManager MDB;
                MDB = new SQLConfigManager(getApplicationContext());    //y los vamos introduciendo en la
tabla local
                if(strings[0].equals("1")){
                    int indice =1;
                    for(int tabla =1; tabla< strings.length; tabla=tabla+3) {
                        MDB.insertarCONTACTO(indice,    strings[tabla].toString(),    strings[tabla+1].toString(),
strings[tabla+2].toString());
                        indice++;
                    }
                }
                MDB.close();
```

```
                //devolvemos el string0 que es el que tiene si ha sido la respuesta del servidor correcta o no
                return (strings[0]);
            } else {
                return ("unsuccessful");
            }
        } catch (IOException e) {
            e.printStackTrace();
            return "exception";
        } finally {
            conn.disconnect();
        }
    }

    @Override
    protected void onPostExecute(String result) {
        if(result.equals("1")){
            Toast.makeText(getApplicationContext().getApplicationContext(), "Base de datos sincronizada", Toast.LENGTH_LONG).show();
        }
        else if (result.equalsIgnoreCase("false")) {
            Toast.makeText(getApplicationContext().getApplicationContext(), "Token incorrecto", Toast.LENGTH_LONG).show();
        } else if(result.equalsIgnoreCase("exception") || result.equalsIgnoreCase("unsuccessful")) {
            Toast.makeText(getApplicationContext().getApplicationContext(), "Error de conexion. Compruebe si el fichero config esta en /sdcard/Download/\"!", Toast.LENGTH_LONG).show();
        }
    }
  }

  public void consultar_base_de_datos(View arg0){
    Intent intent = new Intent(User.this, ContactosActivity.class);
    // Le pasamos a la actividad la direccion IP
    startActivity(intent);
  }

}
```

- **Res/layout**

Activity_contactos.xml

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

   <TextView
      android:id="@+id/contacto_reemplazo"
      android:layout_width="354dp"
      android:layout_height="477dp"
      android:text="TextView"
      android:textSize="18sp"
      tools:layout_editor_absoluteX="16dp"
      tools:layout_editor_absoluteY="16dp" />

</RelativeLayout>
```

Activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".SplashActivity">

   <TextView
      android:id="@+id/textView"
      android:layout_width="326dp"
      android:layout_height="207dp"
      android:layout_alignParentEnd="true"
      android:layout_alignParentTop="true"
      android:layout_marginEnd="23dp"
      android:layout_marginTop="14dp"
      android:text="Previsualizacion de mensaje"
```

```
    android:textSize="18dp" />

  <ImageView
    android:id="@+id/imageView2"
    android:layout_width="246dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="26dp"
    app:srcCompat="@mipmap/ic_launcher_foreground_us" />

  <TextView
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignStart="@+id/textView"
    android:layout_centerVertical="true"
    android:onClick="Open_Browser" />

</RelativeLayout>
```

Actvity_user.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".User">

  <Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="43dp"
    android:text="Sincronizar BBDD"
```

```
    android:onClick="sincronizar_bbdd_manual"
    />

<Button
    android:id="@+id/button4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="167dp"
    android:onClick="registrar_token"
    android:text="Registrar token" />

<EditText
    android:id="@+id/editText"
    android:layout_width="139dp"
    android:layout_height="57dp"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="103dp" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="65dp"
    android:text="Nº de teléfono" />

<Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="109dp"
```

```
    android:onClick="consultar_base_de_datos"
    android:text="Consultar base de datos" />

</RelativeLayout>
```

Actvity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity"
  >

  <EditText
    android:id="@+id/editText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="198dp"
    android:focusable="true"
    android:hint="Username"
    android:textColorHighlight="#ff7eff15"
    android:textColorHint="#2537ff"
    tools:layout_editor_absoluteX="130dp"
    tools:layout_editor_absoluteY="210dp" />


  <EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="195dp"
    android:ems="10"
```

```
    android:hint="Password"
    android:inputType="textPassword"
    android:textColorHint="#293eff"
    tools:layout_editor_absoluteX="85dp"
    tools:layout_editor_absoluteY="278dp" />

    <Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:text="login"
    android:onClick="checkLogin"
    tools:layout_editor_absoluteX="16dp"
    tools:layout_editor_absoluteY="437dp"
    />

  <Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentEnd="true"
    android:text="Cancel"
    tools:layout_editor_absoluteX="266dp"
    tools:layout_editor_absoluteY="437dp" />

</RelativeLayout>
```

- **Res/values**

Colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="colorPrimary">#3F51B5</color>
  <color name="colorPrimaryDark">#303F9F</color>
```

```
    <color name="colorAccent">#FF4081</color>
    <color name="red">#d6d11010</color>
</resources>
```

Ids.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="action_settings" type="id" />
</resources>
```

Strings.xml

```
<resources>
    <string name="app_name">UsNotificaciones</string>
    <string name="login">Login</string>
    <string name="enter_name">Enter Name</string>
    <string name="Username">login</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_user">User</string>
    <string name="password_text">Password</string>
    <string name="attempts_text">Attempts :</string>
    <string name="button_login">Login</string>
    <string name="Secon_page_text">Welocome to the Second Page</string>
</resources>
```

Styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

Gradle Scripts

Build.gradle(UsNotificaciones)

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.1.4'
        classpath 'com.google.gms:google-services:4.0.0'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        maven {
            url "https://maven.google.com" // Google's Maven repository
        }
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Build.gradle(module:app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "androidloginandregistration.alejandro.com.usnotificaciones"
```

```
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
    implementation fileTree(dir: 'libs', include: '*.jar')
    implementation 'com.mcxiaoke.volley:library-aar:1.0.0'
    implementation 'com.google.firebase:firebase-core:16.0.1'
    implementation 'com.google.firebase:firebase-messaging:17.1.0'
}

apply plugin: 'com.google.gms.google-services'
```