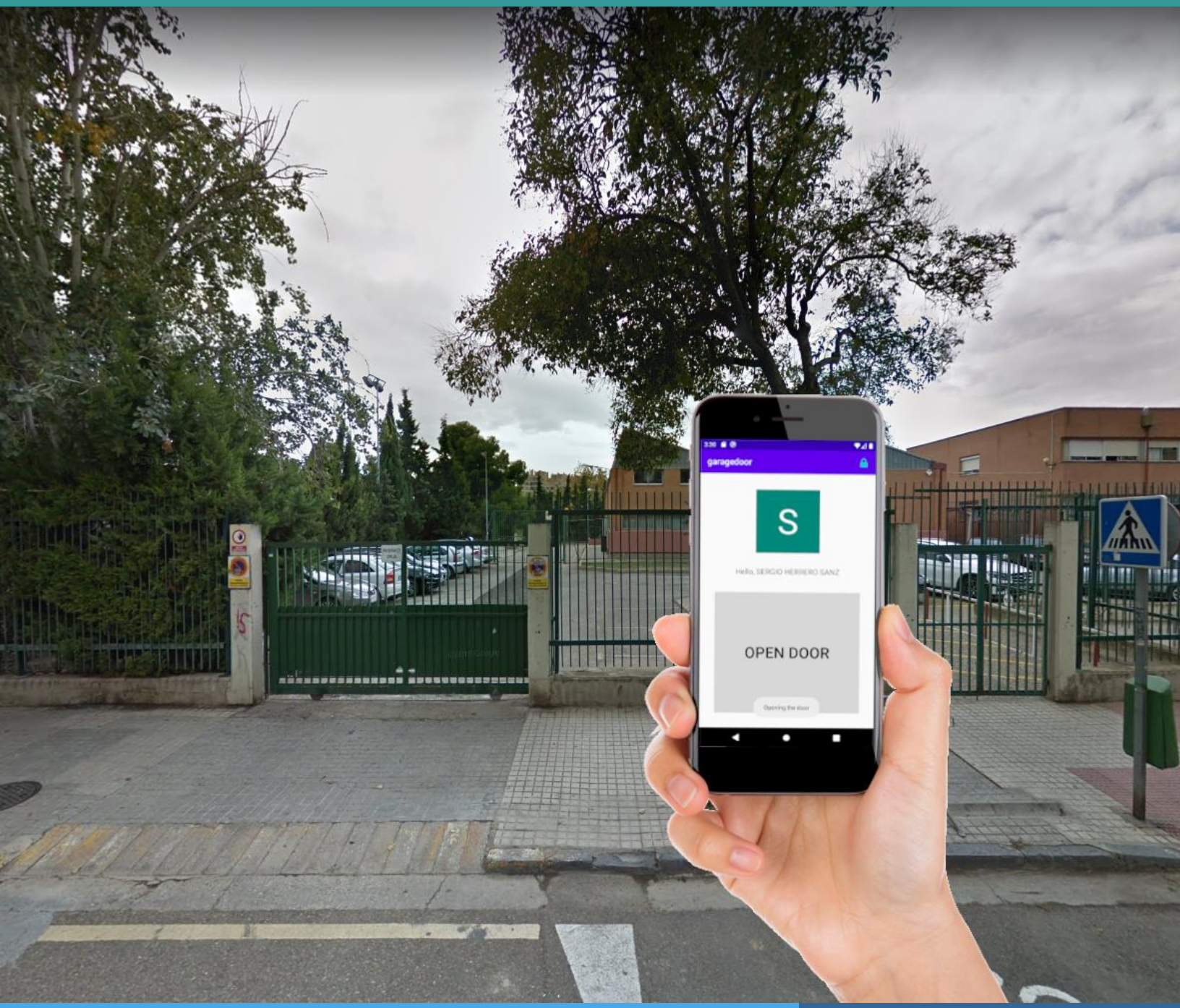
IES Pablo Serrano



Gestión de apertura de puerta de garaje mediante teléfono móvil

Sergio Herrero

2019/2020

INDICE

1. Descripción

1.1. Resumen e introducción

Este proyecto consiste en una aplicación Android que permite la apertura de la puerta de un garaje a través de un servidor conectado a internet, desarrollado también como parte del proyecto.

Mediante el entorno de desarrollo Android Studio se ha desarrollado una app que autentica a los usuarios utilizando las herramientas de Firebase, realiza peticiones HTTPS al servidor mediante el uso de la librería Volley y muestra al usuario si se ha abierto satisfactoriamente la puerta o ha habido algún tipo de problema.

El servidor HTTPS, codificado en JavaScript acepta peticiones POST en las que recibe un parámetro que debe de corresponder con el identificador único generado por Firebase de alguno de los usuarios autorizados. Al recibir el identificador, obtendrá el correo asociado y realizará una búsqueda en la base de dados para comprobar si el usuario está autorizado. Si el usuario está autorizado se abrirá la puerta y se añadirá un registro a la base de datos en el que se guarda el usuario, el día y la hora y si estaba autorizado a entrar. En caso de que el usuario no estuviera autorizado se introduce el registro indicándolo, y en caso de que no estuviese registrado en la base de datos se le dará de alta sin permisos.

1.2. Motivación para la realización del proyecto

Siempre he querido llevar a cabo un gran proyecto por mi cuenta, antes siquiera de comenzar este grado superior, pero siempre me he encontrado con el mismo problema: no me motiva la idea de destinar mi tiempo a desarrollar algo que no va a ser utilizado. Sin embargo, confío en poder ver la aplicación práctica de este proyecto.

Desde hace tiempo me ha atraído la tecnología y la domótica, hasta el punto de orientar mi vida hacia ello desde pequeño. Empecé pidiendo para mi cumpleaños y navidades juguetes como un Mecano, para empezar a interactuar con la tecnología y fui escalando poco a poco hasta desmontar cualquier tipo de aparato que caía en mis manos con la intención de averiguar cómo funcionaban por dentro. Finalmente he acabado aprendiendo de forma autodidacta con Arduino y Raspberry, con la esperanza de algún día conectar la tecnología de mi habitación a través de un servidor diseñado por mí mismo, y digo yo, ¿Qué mejor oportunidad para iniciarme en ello que con este proyecto?

Quiero que este proyecto sea publicado para que cualquiera que lo desee tenga acceso al código fuente para modificarlo libremente, y a una documentación o manual que le indique como implementarlo.

El sector de la domótica es excesivamente caro, costando un interruptor inteligente o una bombilla WiFi más de 25€, haciendo imposible adaptar una casa actual a un sistema 100% inteligente de forma asequible. Este proyecto es el comienzo de un proyecto personal mucho mayor con la intención de hacer accesible a todo el mundo la automatización de dispositivos y electrodomésticos ya existentes. Tomando el servidor y parte de la app Android desarrollados y creando pequeños dispositivos controlados por chips ESP8266 se puede crear una red inteligente a través de WiFi.

Hay proyectos similares en github. Destaca el proyecto Domoticz, que ofrece un servidor multiplataforma que comunica dispositivos IoT de diferentes empresas y ofrece una interfaz común para todos ellos.

1.3. Objetivos del proyecto

- Crear una aplicación Android que se comunique con una API.
- Autenticar usuarios utilizando Firebase.
- Crear un servidor web que ofrezca una API REST con capacidad de interacción con la puerta del garaje.
- Diseñar una base de datos que lleve el control de los usuarios autorizados y el registro de las interacciones con el servidor.
- Desarrollar un software que permita administrar la base de datos para el uso de los empleados de dirección.
- Redactar un manual de implementación, administración y uso para la publicación de este proyecto.
- Diseñar e imprimir en 3D una caja para proteger la electrónica.

2. Planificación del desarrollo del proyecto

1. Crear el servidor web [10h]:
   - Implementar seguridad utilizando HTTPS.
   - Utilizar JavaScript para la codificación y ejecutar en Node.
   - Servir una API REST.

2. Crear la aplicación Android [15h]:
   - Autenticar los usuarios utilizando las herramientas de Firebase.
   - Utilizar fragmentos distintos para el inicio de sesión y la interacción con el servidor.

3. Conectar Android y el servidor [2h]:
   - Realizar peticiones utilizando la librería Volley.
   - Manejar las peticiones utilizando Firebase Admin y JSON para intercambiar datos.

4. Crear la base de datos [3h]:
   - Utilizar MariaDB.
   - Manejar los permisos de cada usuario.
   - Crear una tabla para guardar un registro de las conexiones.

5. Conectar el servidor y la base de datos [2h]:
   - Comprobar que el usuario indicado por Android tenga permisos.
   - Dar de alta sin permisos a los usuarios que no estén registrados.
   - Insertar un registro en la base de datos cada vez que se recibe una petición.

6. Abrir la puerta [8h]:
   - Controlar los pines GPIO de la Raspberry para activar la electrónica.
   - Diseñar un circuito electrónico que utilice un relé y el mando RF de la puerta.

7. Adicionales:
   - Implementar una fecha de vencimiento opcional para cada usuario.
   - Comprobar diariamente los usuarios caducados y darlos de baja.
   - Implementar el circuito electrónico
   - Crear un software que permita controlar la base de datos sin necesidad de un SGBD.
   - Crear un manual de instalación y mantenimiento.
   - Diseñar una caja en 3D para almacenar la Raspberry y la electrónica.
   - Implementar el proyecto en un caso de uso real con la puerta de garaje del instituto.

3. Desarrollo del proyecto

3.1. Estructura del proyecto

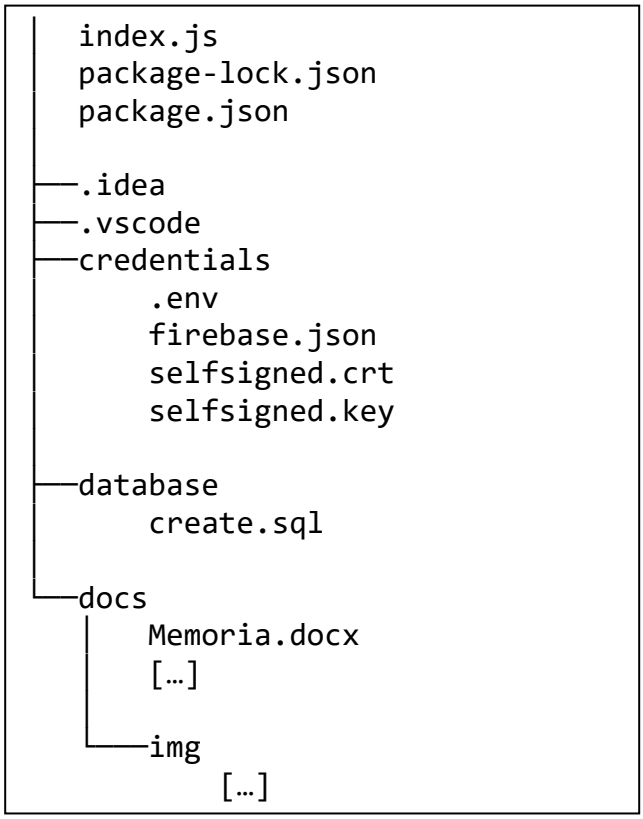
Se ha dividido el proyecto en dos piezas de software independientes, una para el servidor y la Raspberry y otra para Android.

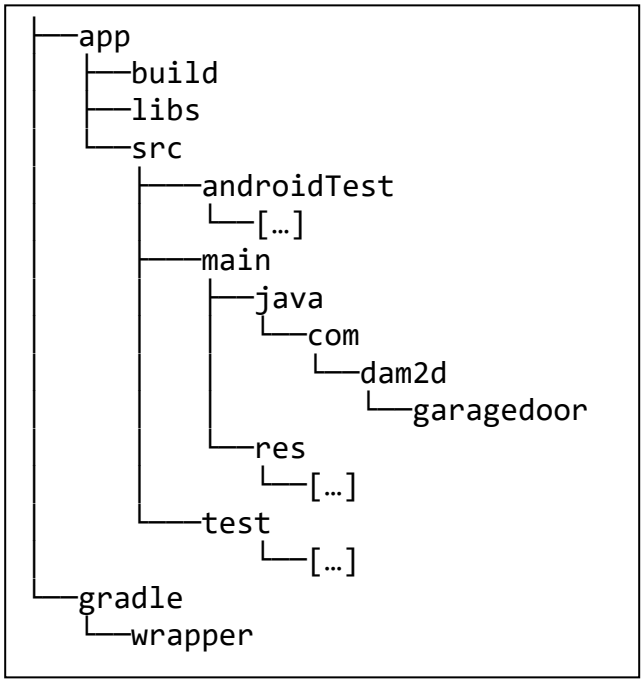
El servidor esta codificado en un único fichero (index.js) que necesita ciertos paquetes de terceros. El gestor de paquetes de Node genera una carpeta llamada node_modules en la que se encuentran las dependencias del proyecto, y los ficheros package.json y package-lock.json, donde se almacena la información del proyecto. La carpeta credentials contiene los ficheros de seguridad necesarios para el correcto funcionamiento:

- .env: Es un fichero del paquete dotenv, que permite almacenar las credenciales de acceso a la base de datos protegiéndolas de posibles ataques al servidor.
- firebase.json: Contiene las credenciales del proyecto de Firebase necesarias para acceder con permisos de administración.
- selfsigned.crt y selfsigned.key: Certificado HTTPS temporal, deberán de ser sustituidos por un certificado respaldado por una entidad autorizada de cara la implementación final del proyecto.

La carpeta database contiene el script de creación de la base de datos, y en la carpeta docs se encuentra todo lo relacionado con esta memoria y el manual.

Debido al funcionamiento de Android Studio todos los ficheros están ubicados en rutas predefinidas, situando el código en la carpeta app/src/main/java, dentro de su propio paquete, y los recursos en la carpeta app/main/res, con una carpeta destinada a cada tipo de recurso. El paquete elegido para este proyecto es com.dam2d.garagedoor. Además, Firebase necesita un archivo de credenciales situado en la carpeta app.

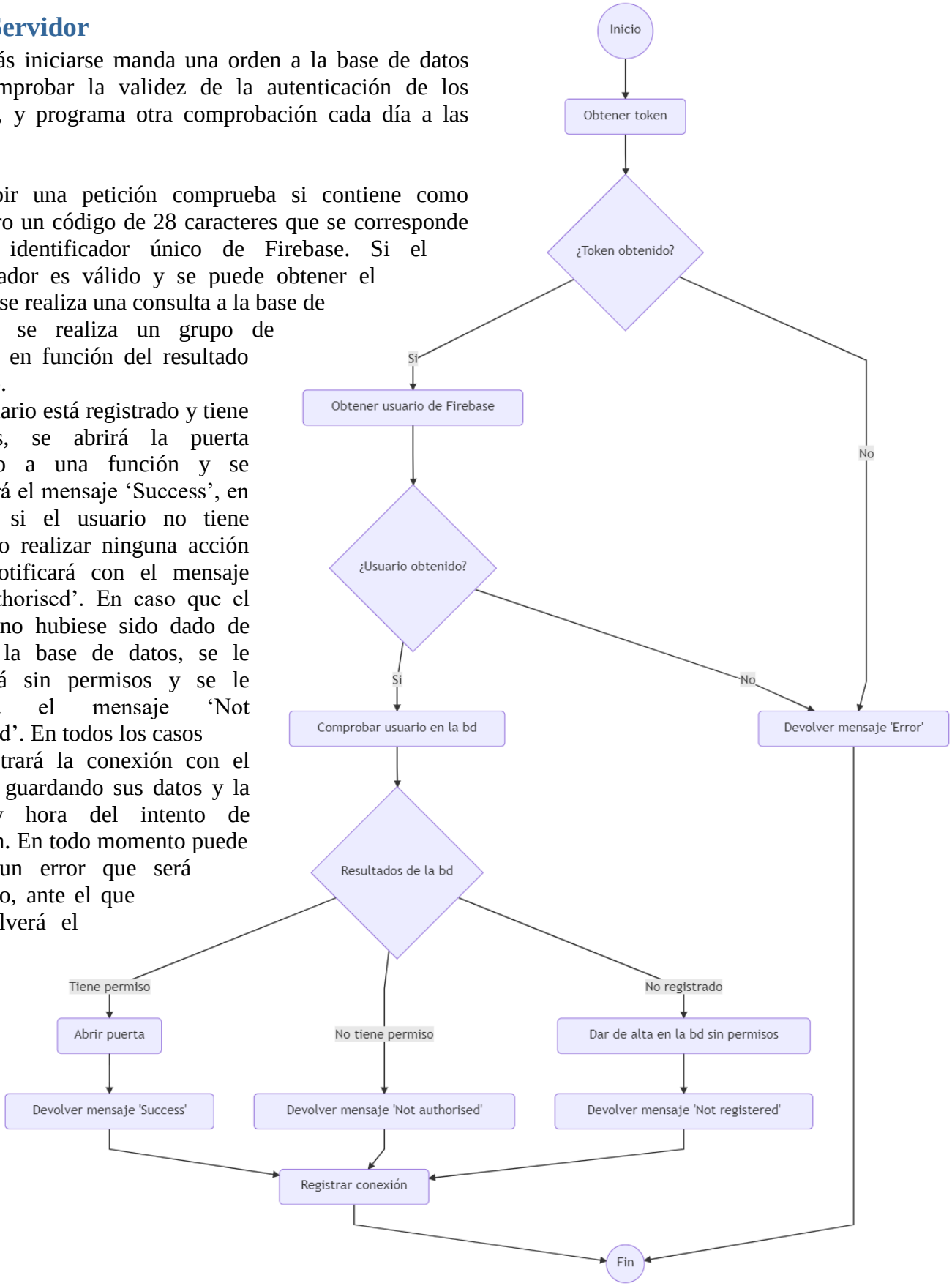




3.1.1. Servidor

Nada más iniciarse manda una orden a la base de datos para comprobar la validez de la autenticación de los usuarios, y programa otra comprobación cada día a las 23:59.

Al recibir una petición comprueba si contiene como parámetro un código de 28 caracteres que se corresponde con el identificador único de Firebase. Si el identificador es válido y se puede obtener el usuario, se realiza una consulta a la base de datos y se realiza un grupo de acciones en función del resultado obtenido.
Si el usuario está registrado y tiene permisos, se abrirá la puerta llamando a una función y se devolverá el mensaje 'Success', en cambio, si el usuario no tiene permitido realizar ninguna acción se le notificará con el mensaje 'Not authorised'. En caso que el usuario no hubiese sido dado de alta en la base de datos, se le registrará sin permisos y se le mostrará el mensaje 'Not registered'. En todos los casos
Se registrará la conexión con el servidor guardando sus datos y la fecha y hora del intento de conexión. En todo momento puede ocurrir un error que será capturado, ante el que se devolverá el mensaje 'Error'.

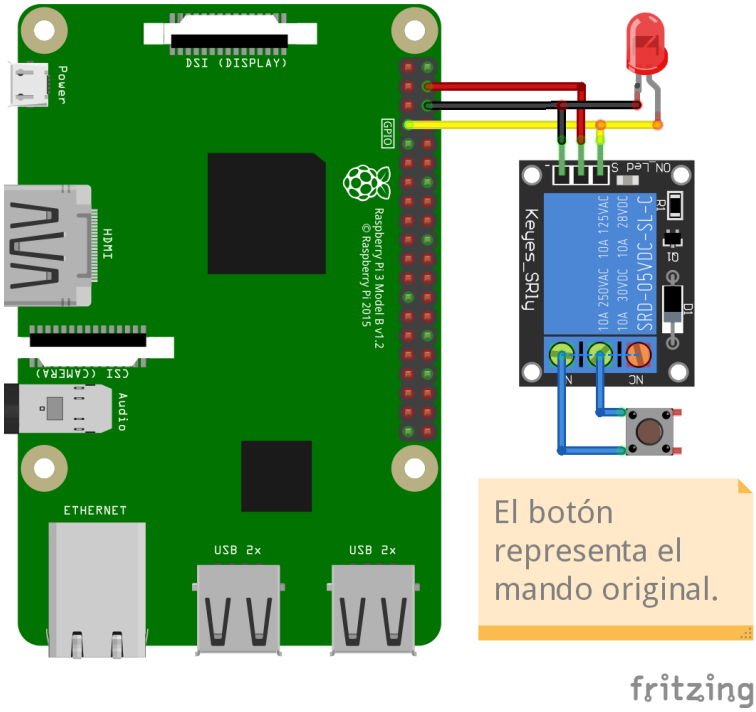


Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito con todo lo necesario para su uso integrado en una única placa. En este proyecto se hace uso de varios de los 40 pines programables de forma nativa y de su conexión a internet.

Este pequeño ordenador será el que actúe como servidor web, contenga la base de datos y controle la puerta de garaje.

En el proceso de abrir la puerta, el servidor utiliza la librería gpio para activar el pin 7 (GPIO 4). Conectando un led y un módulo de relé a este pin podemos simular una pulsación en el mando original de la puerta del garaje, iluminándose el led cada vez que se abra la puerta.

Un relé es un interruptor accionado eléctricamente que permite encender un circuito eléctrico, o, en este caso, cortocircuitar los dos pines del interruptor del mando de la puerta, simulando una pulsación y permitiendo el acceso a la persona que lo solicitó.

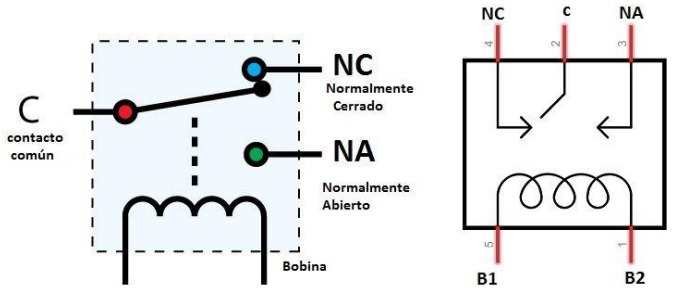


Se debe alimentar el módulo de relé con 5V, lo que le permite amplificar la señal proveniente de, en este caso, la Raspberry. Al tratarse de conexiones directas al microprocesador están limitadas a 20mAh y conectar el relé directamente a ellas sería catastrófico.

Como se ve reflejado en la imagen, se conecta un cable negro al uno de los pines negativos de la Raspberry (Pines 6, 9, 14, 25, 30, 34 y 39), un cable rojo a otro de los pines de 5V (Pines 2 y 4) y por último el cable amarillo de señal al pin 7.

Opcionalmente se conecta un led al pin 7 y a uno de los negativos, creando así un indicador visual, aunque dependiendo del módulo de relé elegido viene integrado en su placa. Debido al funcionamiento del relé se escucha un clic del interruptor interno cada vez que se activa o desactiva.

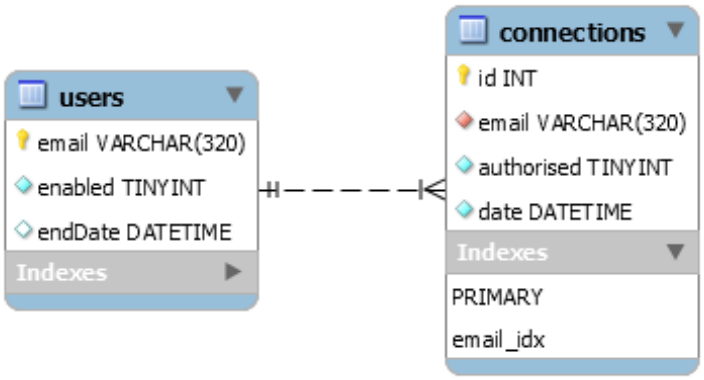
Los relés pueden actuar como interruptores normalmente cerrados o normalmente abiertos, por lo que tienen 3 conexiones de las que solo se usan dos. Al tratarse de un pulsador que solo se activa al ser pulsado, se conecta a los terminales C (Común) y NA (Normalmente Abierto). La conexión NC (Normalmente Cerrado) no se utiliza y se deja sin conectar.



Al meter corriente por la bobina los contactos abiertos se cierran y los cerrados se abren.

3.1.2. Base de datos

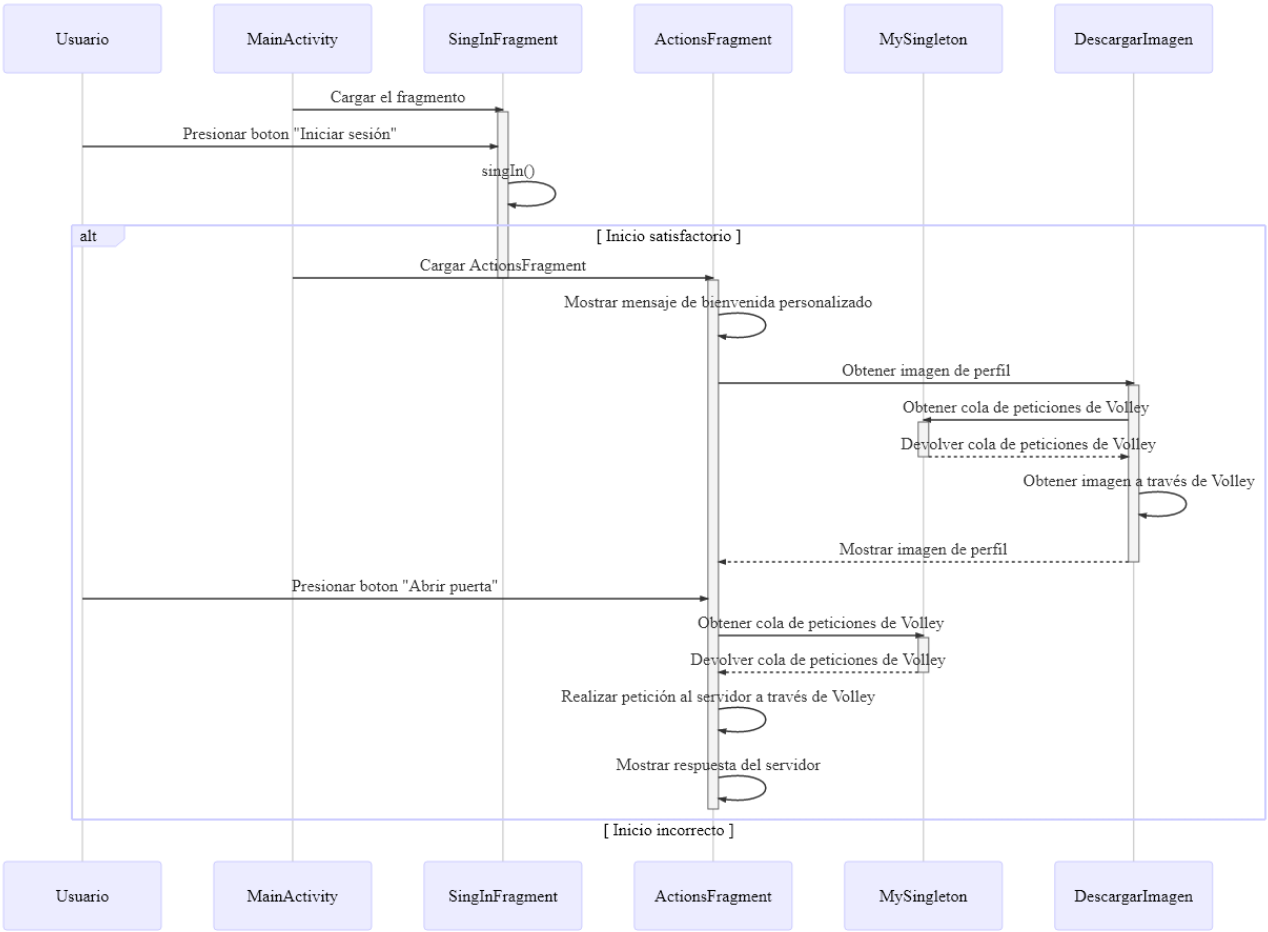
La base de datos contiene dos tablas que almacenan los usuarios y las interacciones realizadas con el servidor. La tabla users guarda el email (email) de los usuarios junto con un booleano (enabled) que representa si el usuario tiene permiso para abrir la puerta. Además se registra la fecha en la que el usuario debe de dejar de estar activo (endDate).



La tabla connections almacena todos y cada uno de los intentos de abrir la puerta, guardando el email del usuario (email), la fecha (date) y si estaba o no autorizado (authorised). Cada vez que se inicia el servidor y cada día a las 00:00 se ejecuta un script que actualiza el campo enabled de cada registro si se supera la fecha de vencimiento (campo endDate).

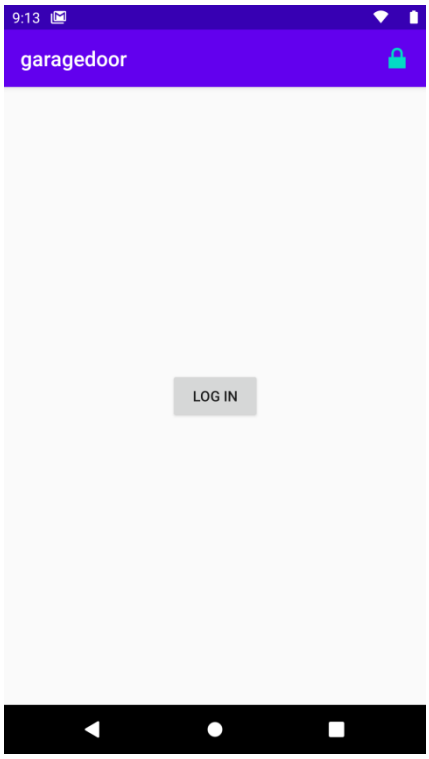
3.1.3. Android

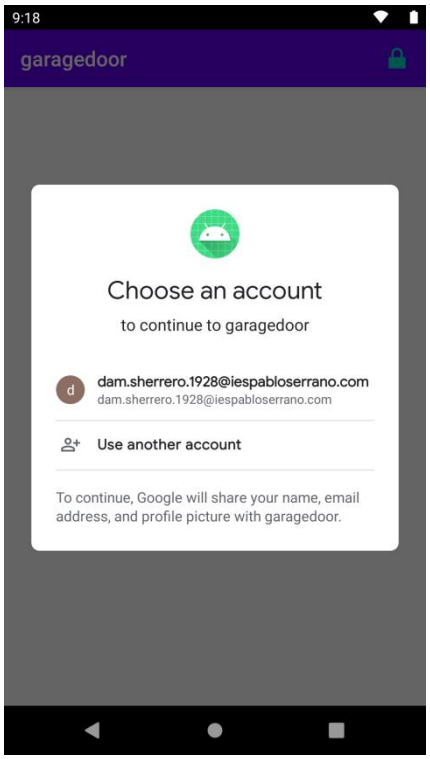
He decidido crear 5 clases y 3 layout, que permiten el siguiente funcionamiento:



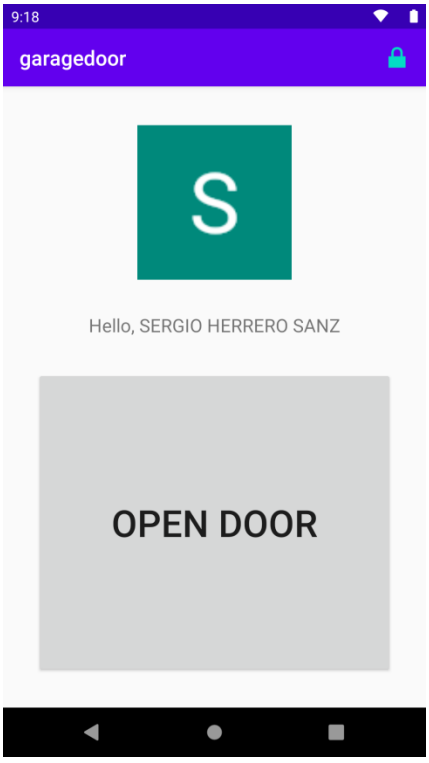
La actividad principal (MainActivity) contiene únicamente un contenedor para fragmentos que nada más iniciar la aplicación carga el fragmento SingInFragment. Además contiene el código necesario para implementar el cierre de sesión a través de un icono en la toolbar.

El fragmento SingInFragment permite iniciar sesión con una cuenta de Gmail para poder hacer uso de la aplicación. Al cargar el fragmento, comprueba si el usuario ha iniciado sesión y, en dicho caso, sustituye el fragmento por ActionsFragment.

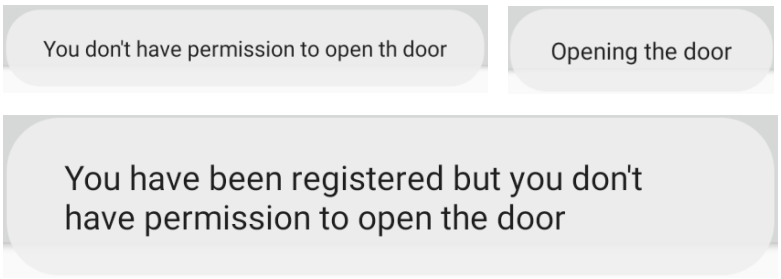




Cuando el usuario activa el botón destinado a iniciar sesión, se llama al método singIn() que comienza con el proceso de inicio de sesión. Cuando se haya completado satisfactoriamente cargará el fragmento ActionsFragment. Además define el método singOut() para cerrar sesión, que es llamado desde MainActivity.

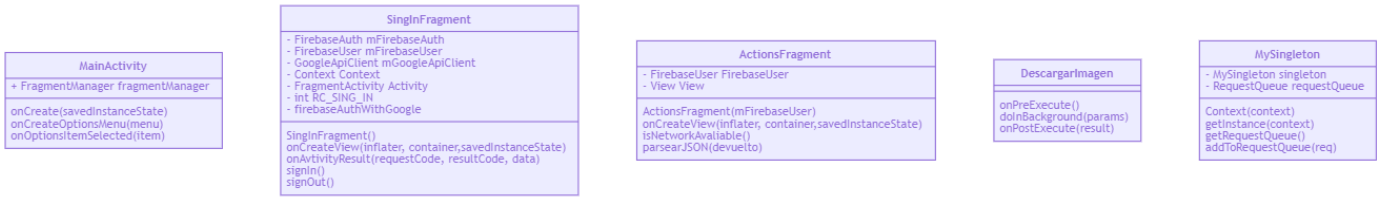


ActionsFragment contiene dos clases en un mismo fichero. Nada más cargar el fragmento utiliza la clase DescargarImagen para obtener la foto de perfil de la cuenta con la que se ha iniciado sesión, y muestra una frase de bienvenida. Al pulsar en el botón para abrir la puerta realiza una petición al servidor y muestra un mensaje con el resultado de dicha consulta.



La clase MySingleton utiliza el patrón Singleton para evitar crear más de una cola de peticiones de Volley. Se utiliza al realizar peticiones al servidor.

Podemos observar los métodos definidos en cada clase en el siguiente diagrama de clases:



Para la autenticación se utiliza Firebase, un conjunto de herramientas de Google compatibles con múltiples plataformas. Para este proyecto interesa únicamente el servicio de autenticación. En primer lugar se descarga el archivo de configuración para vincular el proyecto con la aplicación Android y se activa desde la consola del proyecto el inicio de sesión con cuentas de Google. A partir de ese momento se puede implementar el inicio de sesión. Cuando se registra un usuario se guardan sus datos y se les asocia un identificador único que se utiliza para comprobar su identidad en el servidor. A partir de ese identificador se puede acceder a los datos que el usuario haya hecho decidido compartir.

3.2. Temporización real

19/03/2020

Investigación de proyectos similares y planificación en base a lo encontrado

20/03/2020

Búsqueda de documentación y dependencias

23/03/2020

Creación del servidor https (servidor)
Búsqueda sobre certificados para servidores https (servidor)
Búsqueda de servidores web en Node (servidor)
Búsqueda de formas de autenticación (Android)

24/03/2020

Lectura de documentación de Firebase (Android)
Creación del servidor https y firma de certificados (servidor)

25/03/2020

Prueba de demos de Firebase (Android)
Lectura de documentación de Firebase y pruebas (Android)

26/03/2020

Lectura de documentación de Firebase y pruebas (Android)
Implementado inicio de sesión y envío de peticiones (Android)

27/03/2020

Lectura de documentación Firebase (servidor)
Parche de certificados y envío de parámetros mediante post (Android)
Recepción de parámetros post y creación de la estructura de rutas (servidor)

30/03/2020

Obtención de datos de Firebase (servidor)
Lectura de documentación Firebase y pruebas (servidor)

31/03/2020

Rehacer inicio de sesión y limpieza de código (Android)
Implementar cierre de sesión y fragmentos (Android)

01/04/2020

Instalación de Node y MariaDB y configuración del router (Raspberry)
Instalación y configuración del sistema operativo (Raspberry)
Mejora de la interfaz (Android)

02/04/2020

Implementación de conexión a la base de datos y consultas de usuarios (servidor)
Búsqueda sobre conexión y consultas a base de datos desde Node (Raspberry)
Crear base de datos (Raspberry)

03/4/2020

Extraer código a funciones y limpieza de código (servidor)

14/04/2020

Implementar control de pines de Raspberry con rpio (servidor)
Búsqueda de información sobre control de pines de la Raspberry (servidor)

15/04/2020

Implementar control de pines de Raspberry con gpio (servidor)
Solucionar problemas con git (servidor)

16/04/2020

Prueba real de todas las funciones en Raspberry

17/04/2020

Implementar registro de conexiones para cada usuario (servidor)
Añadir tabla connections y modificar campos existentes (Raspberry)

20/04/2020

Estructura (memoria)
Escribir motivación (memoria)
Escribir introducción (memoria)

21/04/2020

Escribir explicación Android (memoria)
Diagrama de actividad de Android (memoria)

22/04/2020

Escribir estructura del proyecto y diagramas de árbol (memoria)
Insertar imágenes de fragmentos Android (memoria)

27/04/2020

Crear diagramas de clases Android (memoria)

28/04/2020

Añadir mensajes personalizados (Android)
Cambiar imágenes, revisar texto y maquetar (memoria)

29/04/2020

Crear diagrama de clases servidor (memoria)
Escribir servidor (memoria)

30/04/2020

Texto funcionamiento rpi y relé (memoria)

01/05/2020

Instalar Fritzing
Crear diagrama electrónico (memoria)

04/05/2020

Explicar conexiones (memoria)
Explicar relé (memoria)

05/05/2020

Instalar Workbench
Explicar base de datos (memoria)

06/05/2020

Implementar llamada al procedure (servidor)
Añadir procedure para actualizar usuarios de la base de datos (Raspberry)
Crear diagrama de la base de datos y añadir imagen (memoria)

07/05/2020

Escribir casuística (memoria)

08/05/2020

Escribir bibliografía (memoria)
Escribir planificación (memoria)

11/05/2020

Escribir material empleado, hardware (memoria)

12/05/2020

Escribir material empleado, software para el desarrollo (memoria)

13/05/2020

Escribir material empleado, software parte del proyecto (memoria)

14/05/2020

Escribir propuesta de mejora (memoria)
Investigar y crear diagramas de propuesta de mejora (memoria)

15/05/2020

Buscar imágenes, maquetar y añadir enlaces a webgrafía (memoria)

18/05/2020

Escribir resultado (memoria)
Escribir temporización (memoria)
Revisar resumen e introducción y motivación (memoria)

3.3. Casuística del proyecto

Certificado HTTPS no aceptado por Android [SOLUCIONADO]

```
W/System.err: com.android.volley.NoConnectionError: javax.net.ssl.SSLHandshakeException: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
W/System.err:     at com.android.volley.toolbox.BasicNetwork.performRequest(BasicNetwork.java:181)
W/System.err:     at com.android.volley.NetworkDispatcher.processRequest(NetworkDispatcher.java:131)
W/System.err:     at com.android.volley.NetworkDispatcher.processRequest(NetworkDispatcher.java:111)
W/System.err:     at com.android.volley.NetworkDispatcher.run(NetworkDispatcher.java:90)
W/System.err: Caused by: javax.net.ssl.SSLHandshakeException: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
W/System.err:     at com.android.org.conscrypt.ConscryptFileDescriptorSocket.startHandshake(ConscryptFileDescriptorSocket.java:231)
W/System.err:     at com.android.okhttp.internal.io.RealConnection.connectTls(RealConnection.java:196)
W/System.err:     at com.android.okhttp.internal.io.RealConnection.connectSocket(RealConnection.java:153)
W/System.err:     at com.android.okhttp.internal.io.RealConnection.connect(RealConnection.java:116)
W/System.err:     at com.android.okhttp.internal.http.StreamAllocation.findConnection(StreamAllocation.java:186)
W/System.err:     at com.android.okhttp.internal.http.StreamAllocation.findHealthyConnection(StreamAllocation.java:128)
W/System.err:     at com.android.okhttp.internal.http.StreamAllocation.newStream(StreamAllocation.java:97)
W/System.err:     at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:289)
W/System.err:     at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:232)
W/System.err:     at com.android.okhttp.internal.huc.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:465)
W/System.err:     at com.android.okhttp.internal.huc.HttpURLConnectionImpl.connect(HttpURLConnectionImpl.java:131)
W/System.err:     at com.android.okhttp.internal.huc.HttpURLConnectionImpl.getOutputStream(HttpURLConnectionImpl.java:262)
W/System.err:     at com.android.okhttp.internal.huc.DelegatingHttpsURLConnection.getOutputStream(DelegatingHttpsURLConnection.java:219)
W/System.err:     at com.android.okhttp.internal.huc.HttpsURLConnectionImpl.getOutputStream(HttpsURLConnectionImpl.java:30)
W/System.err:     at com.android.volley.toolbox.HurlStack.addBody(HurlStack.java:292)
W/System.err:     at com.android.volley.toolbox.HurlStack.addBodyIfExists(HurlStack.java:277)
W/System.err:     at com.android.volley.toolbox.HurlStack.setConnectionParametersForRequest(HurlStack.java:249)
W/System.err:     at com.android.volley.toolbox.HurlStack.executeRequest(HurlStack.java:94)
W/System.err:     at com.android.volley.toolbox.BasicNetwork.performRequest(BasicNetwork.java:123)
W/System.err:  ... 3 more
W/System.err: Caused by: java.security.cert.CertificateException: java.security.cert.CertPathValidatorException: Trust anchor for certification path not found.
W/System.err:     at com.android.org.conscrypt.TrustManagerImpl.verifyChain(TrustManagerImpl.java:674)
W/System.err:     at com.android.org.conscrypt.TrustManagerImpl.checkTrustedRecursive(TrustManagerImpl.java:551)
```

Como se puede apreciar en esta imagen, Android detecta y bloquea los sitios web con certificados vencidos o auto firmados. La solución es cambiar el certificado por uno respaldado por una entidad fiable. Durante el desarrollo se ha implementado el método handleSSLHandshake(), que trata como si fuesen de confianza todos los servidores, clientes y certificados. Esta solución es extremadamente peligrosa y solo se debe utilizar durante el desarrollo.

Recepción de Parámetros mediante POST [SOLUCIONADO]

Al manejar una petición e intentar leer los parámetros como si se tratase de un JSON el valor de los parámetros es undefined. La solución es utilizar promesas para procesar el cuerpo de la petición.

No se puede modificar imagen de perfil

A la hora de mostrar la foto de perfil del usuario de google quiero modificarla para redondear los bordes o que se muestre con forma circular. A pesar de crear una capa para establecer como background que solo permita ver la forma deseada, la clase DescargarImagen restaura los atributos a los por defecto cuando se actualiza la imagen y se tal y como se obtiene de los servidores de Google.

Problemas de red en Ubuntu Server [SOLUCIONADO]

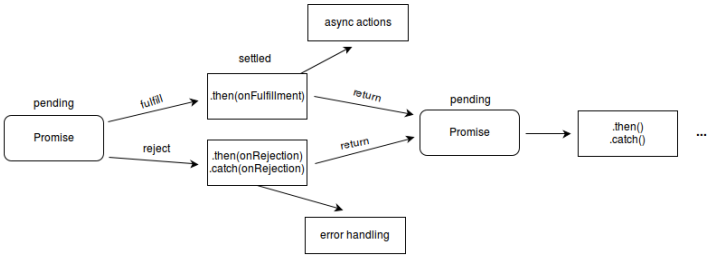
Al tratarse de un sistema operativo destinado a funcionar como servidor viene sin ninguna configuración previa. Ubuntu no detecta la red conectada mediante el cable Ethernet ni es capaz de crear una red local. No se puede acceder a la base de datos ni al servidor, ni desde internet, red local o localhost. La solución es cambiar el sistema operativo por la última versión de Raspbian.

Dotenv no es capaz de recuperar las credenciales de la base de datos [SOLUCIONADO]

El paquete dotenv lanza un error al intentar recuperar las credenciales de uno de los usuarios de la base de datos. El generador de contraseñas usado para los usuarios utiliza caracteres especiales no permitidos por dotenv. La solución es cambiar la contraseña por otra que no utilice los caracteres reservados.

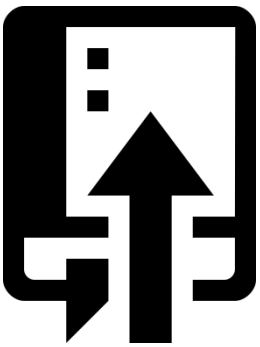
Promesas no extraídas a funciones

Al ejecutar el servidor después de refactorizar código a funciones, las promesas extraídas devuelven '<pending>' en lugar del valor de la variable. Sin resolver. La solución temporal es deshacer los cambios.



Librería rpio no permite iniciar el servidor

Una de las dependencias de la librería rpio tiene un problema de compatibilidad con la versión de Node o del sistema operativo. A pesar de la versatilidad de la librería y de haber implementado todas las funcionalidades necesarias la solución es usar la librería gpio en su lugar y adaptar el código.



No se puede hacer push a GitHub [SOLUCIONADO]

Tras hacer un push a GitHub y arreglar un pequeño error con un ammend el repositorio devuelve el mensaje 'Local repository not in sync' cuando se intenta actualizar. Resulta que al sobrescribir el último commit con el ammend GitHub detecta que la versión local y el servidor no contienen los mismos datos, puesto que el identificador del commit es el mismo pero el contenido no. Hay varias posibles soluciones:

- Forzar el push desde la consola. [ELEGIDA]
- Hacer un commit normal en lugar de un ammend.

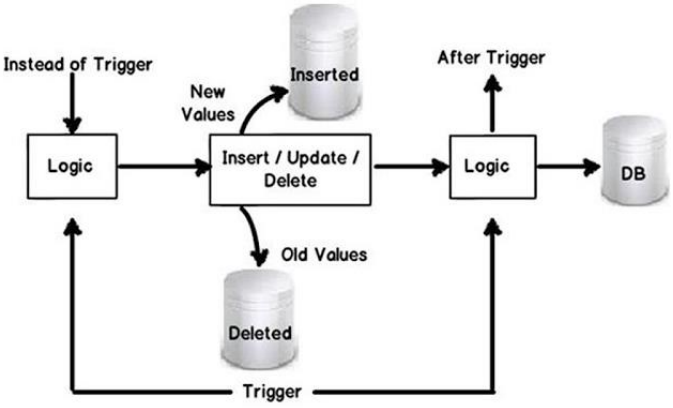
Cuenta Git no autorizada [SOLUCIONADO]

Al intentar hacer un push para solucionar el problema anterior, GitHub rechaza la conexión puesto que se está usando una cuenta no autorizada desde la que se inició sesión previamente. Hay dos posibles soluciones:

- Autorizar la cuenta desde GitHub.
- Borrar las credenciales de git desde el administrador de credenciales de Windows (Panel de control/Cuentas de usuario/Administrador de credenciales -> Credenciales de Windows -> git) e iniciar sesión de nuevo al hacer el push. [ELEGIDA]

No se puede comprobar la validez de un usuario en cada consulta [SOLUCIONADO]

Para comprobar que la autorización de un usuario no ha vencido se iba a implementar un trigger que se ejecutase antes de cada select. Como los triggers no se pueden ejecutar sobre select, la mejor opción es crear un procedimiento en la base de datos y llamarlo diariamente desde el servidor.



3.4. Material empleado

3.4.1. Hardware

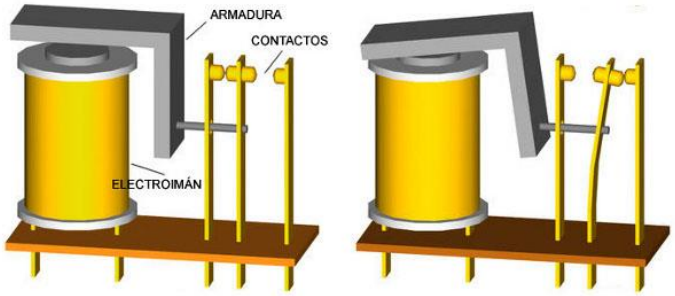
Raspberry Pi 3B+

Es un pequeño ordenador del tamaño de una tarjeta de crédito capaz de funcionar sin necesidad de componentes extra. Entre todas las versiones de tamaño completo disponibles (Ordenadas cronológicamente y por potencia: A, B, B+, 2B, 3A+, 3B, 3B+, 4B) se ha elegido la versión 3B+ para este proyecto por ser la más potente en el momento de compra, puesto que posteriormente se puso a la venta el modelo 4B. A pesar de ello, cualquier modelo posterior a la versión 3B debería de ser compatible y suficientemente potente para ejecutar el servidor. Todas las Raspberry compatibles integran WiFi, Bluetooth y un puerto Ethernet. Todos los modelos utilizan una tarjeta micro SD para almacenar el sistema operativo y los datos de usuario.

Versión	RAM	Procesador	USB	Ethernet	WiFi	Bluetooth	Precio
3 B	1 Gb	1.2 GHz 64-bit ARM Cortex A53	4 USB 2.0	10/100Mbps	802.11n	4.1	37€
3 B+	1 GB	1.4 GHz 64-bit ARM Cortex A53	4 USB 2.0	300Mbps/PoE	802.11ac	4.2	44€
4 B	4 GB	1.5 GHz 64-bit ARM Cortex A72	2 USB 2.0 2 USB 3.0	10/100/1000Mbps	802.11ac	5.0	64€

Módulo de Relé KY-019

Es un módulo que acepta una señal de bajo nivel y la amplifica para accionar un relé de 5V. Este módulo permite accionar un circuito de un máximo de 10A a 250V o 15A a 125V. En este proyecto no se necesita tanta potencia, sin embargo es compartir el potencial neutro (GND) necesario para accionar el circuito con un transistor dañaría en mando de garaje. Dando corriente al relé se genera un campo magnético que mueve un contacto interno, cerrando el circuito.

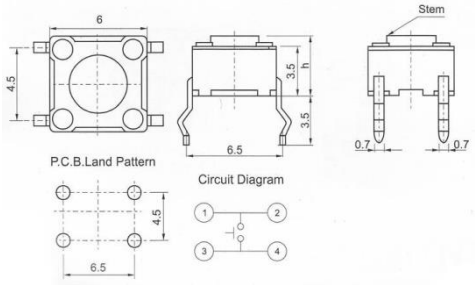


Led (opcional)

Un pequeño piloto luminoso que se ilumina al abrir la puerta de garaje. Para conectar un led a un circuito eléctrico hay que saber que es un componente con polaridad. A diferencia de las resistencias o los inductores depende de la orientación en la que se instala para su funcionamiento. Comúnmente los leds tienen una patilla más corta que indica en contacto negativo, sino se puede identificar este contacto mirando la cabeza. En su interior podemos observar que uno de los contactos, el ánodo, es más grande. En cualquier caso se debería de conectar una resistencia para evitar que se funda el led o uno de los pines de la Raspberry.

Mando de garaje

El mando de la puerta de garaje es un transmisor radiofrecuencia que emite un código para que un receptor en la puesta de garaje lo detecte. Si el código enviado es el correcto la puerta se abrirá. Para la realización de este proyecto se puede utilizar un mando original o un mando compatible programable. En caso de utilizar un mando programable habrá que configurar 8 interruptores DIP que establecen el código que se envía y, en algunos casos, ajustar un potenciómetro que controla la frecuencia de emisión antes de añadirlo al circuito conectado a la Raspberry.



Para simular una pulsación con el relé se debe de conectar dos cables al botón utilizado para abrir la puerta. Como podemos ver en el diagrama del botón, los cables se deben de conectar a las 2 patillas del mismo lado del botón, puesto que los dos extremos están comunicados externamente. En la imagen del mando indico que dos conexiones hay que soldar, usando el interruptor superior izquierdo como ejemplo.

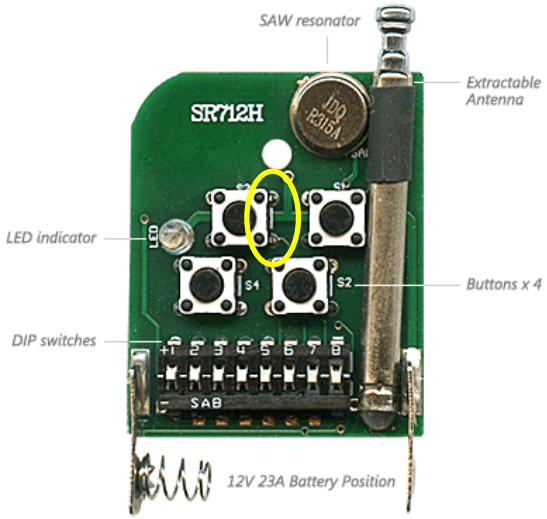


Figure 1. Front side of SR715 PCB

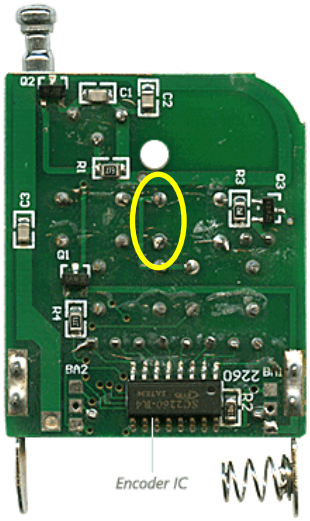


Figure 2. Back side of SR715 PCB

Como los cables se sueldan por la parte trasera hay que tener en cuenta que las conexiones están invertidas.

3.4.2. Software para el desarrollo

Android Studio

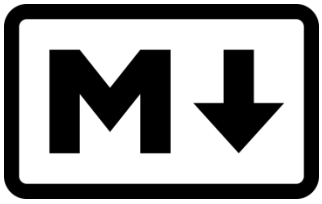
Es el IDE oficial para el desarrollo de apps Android basado en IntelliJ IDEA. Permite compilar las aplicaciones con Gradle y probarlas con un emulador integrado compatible con todas las versiones de Android. Integra GitHub, herramientas de Lint y plantillas de código.

WebStorm

Es el IDE de JetBrains para JavaScript. Incluye las comodidades de Android Studio, compartidas entre todos los IDEs basados en IntelliJ IDEA.

Visual Studio Code

Es un potente editor de texto orientado a la programación. Funciona a partir de módulos programados por la comunidad que añaden funcionalidades. Nativamente incluye IntelliSense, una herramienta que añade funcionalidades de resaltado de sintaxis y completación de código. A pesar de no ser un IDE permite debuguear desde el propio editor, siendo compatible con multitud de lenguajes y plataformas.



Markdown

Es un lenguaje de marcado para texto plano que permite aplicar estilos de forma sencilla sin necesitad de un editor específico. Fue creado para escribir páginas HTML de forma que fuese sencillo leer el texto fuente. Utilizado en un principio para escribir esta memoria.

Mermaid

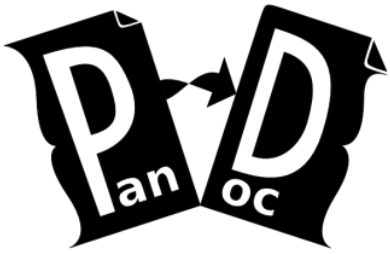
Es un script de JavaScript que permite dibujar diagramas a usando la sintaxis de Markdown. Permite generar diagramas de flujo, de secuencia, de Gantt, de clases, de estado, de tarta y más.

Mermaid Live Editor

Editor online de Mermaid utilizado para exportar los diagramas.

Pandoc

Es un conversor de texto libre y de código abierto capaz de convertir múltiples formatos entre sí, incluyendo Markdown a Word.



Kanban Tasker

Permite organizar el trabajo mediante metodologías ágiles. Funciona agregando tarjetas para cada tarea que se debe realizar y ubicándolas en una columna según su estado (en reserva, por hacer, en proceso, revisar y terminado) y según su prioridad (alta, normal y baja). De esta forma se puede ver el estado del proyecto de forma sencilla.

GitHub desktop

Versión de GitHub para escritorio. Permite gestionar el repositorio cómodamente, comparar versiones y actualizar el repositorio.

MySQL Workbench

Es una herramienta visual para el diseño gestión y mantenimiento de bases de datos MySQL. En este proyecto se ha utilizado para crear diagramas físicos de la mase de datos.

Real VNC

VNC significa Virtual Network Computing. Es un programa que permite controlar otro ordenador de forma remota a través de internet. Ha sido utilizado para acceder a la Raspberry sin necesidad de teclado ratón ni monitor. Consta de dos partes, la parte del servidor, instalada en la Raspberry, y la parte del cliente, instalada en el ordenador utilizado para el desarrollo.

3.4.3. Software parte del proyecto

Node.js

Es un entorno de ejecución que permite ejecutar código JavaScript sin necesidad de un navegador, haciendo posible crear aplicaciones independientes que corran como servidores. Node funciona de tal forma que es capaz de procesar múltiples peticiones desde un único subproceso que se llama nodo, apartando de la ejecución un evento cuando se bloquea y comenzando a procesar el siguiente de forma instantánea, para volver al anterior en cuanto sea posible. Su forma de trabajar hace que tenga muy poca sobrecarga, sea ligero y, al interpretar JavaScript, que sea muy sencillo de programar.

Raspbian

Es un sistema operativo basado en Debian optimizado para el hardware de la Raspberry. Viene en dos paquetes diferentes. Raspbian Pixel incluye un entorno gráfico y está diseñado para su uso como ordenador de escritorio. Raspbian Lite funciona exclusivamente por consola y está diseñado para usuarios avanzados y su uso como servidor. En este proyecto se ha utilizado la versión Pixel por comodidad, pero sería sencillo implementar el servidor y la base de datos en la versión Lite.

Noobs (New Out of Box Software)

Es el instalador de Raspbian. Permite su instalación a través de un asistente sencillo que se encarga de formatear la tarjeta SD y quemar la imagen ISO elegida con un solo clic.

MariaDB

Es un sistema gestor de bases de datos relacional basado en MySQL, que toma su código fuente original y lo mejora y libera como open source. Comparado con MySQL amplía los mecanismos de almacenamiento y la velocidad.

Dependencias del servidor (express, https, fs, Firebase admin, dotenv, MariaDB, gpio)

Como ya se ha explicado permiten el funcionamiento del servidor, otorgando seguridad, acceso y control a la base de datos, Firebase y los pines de la Raspberry.

Firebase

Es una plataforma de google que ofrece herramientas que facilitan el desarrollo de las aplicaciones, con la intención de aumentar la cantidad de usuarios y la cantidad de ingresos. Ofrece herramientas para el desarrollo, mejora de calidad, analíticas y crecimiento. Entre ellas las de autenticación, mensajería, control de fallos, bases de datos, almacenamiento y hosting. El uso de Firebase es gratuito con una cuota de usuarios limitada.

3.4.4. Bibliografía/Webgrafia

Proyectos similares:
https://www.youtube.com/watch?v=fCUJgj_pb4E
https://www.youtube.com/watch?v=wQlXgi7THG8
https://www.youtube.com/watch?v=NQINLrnciYM
https://circuitdigest.com/microcontroller-projects/iot-smart-garage-door-opener-using-raspberry-pi
Instalar de Node
https://www.instructables.com/id/Install-Nodejs-and-Npm-on-Raspberry-Pi/
https://openwebinars.net/blog/que-es-nodejs/
File system:
https://www.npmjs.com/package/file-system
Instalar express:
https://expressjs.com/es/starter/installing.html
Añadir seguridad a express:
https://stackoverflow.com/questions/11744975/enabling-https-on-express-js
https://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
https://www.npmjs.com/package/https
Firmar certificados con OpenSSL:
https://github.com/sagardere/set-up-SSL-in-nodejs
https://slproweb.com/products/Win32OpenSSL.html
Averiguar clave SHA1 de Android Studio
https://stackoverflow.com/questions/42663114/keystore-file-does-not-exist/46138449
Enviar parámetros con Volley:
https://www.itsalif.info/content/android-volley-tutorial-http-get-post-put
Descargar imagen en Android Studio
https://stackoverflow.com/questions/18210700/best-method-to-download-image-from-url-in-android
Redondear imagen en Android Studio
https://stackoverflow.com/questions/2459916/how-to-make-an-imageview-with-rounded-corners
MariaDB
https://howtoraspberrypi.com/mariadb-raspbian-raspberry-pi/
Raspberry
https://www.programoergosum.com/cursos-online/raspberry-pi/232-curso-de-introduccion-a-raspberry-pi/instalar-raspbian
https://www.raspberrypi.org/downloads/
https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/
Ejecutar servidor al inicio en Raspberry
https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/
Controlar pines Raspberry
https://www.npmjs.com/package/rpio
https://www.npmjs.com/package/gpio
Relé
https://como-funciona.co/un-rele-o-relevador/
https://app.emaze.com/@AZLOQQLQ
Mando de garaje
http://comprarmandogaraje.blogspot.com/2014/10/control-remoto-rf-tipica-diseno-de-pcb.html
Android Studio
https://developer.android.com/studio/intro

Visual Studio Code
https://blogs.itpro.es/eduardocloud/2016/08/22/visual-studio-code-que-es-y-que-no-es/
https://code.visualstudio.com/
Markdown
https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
https://daringfireball.net/projects/markdown/
Mermaid
https://mermaid-js.github.io/mermaid/#/
Mermaid Live Editor
https://mermaid-js.github.io/mermaid-live-editor/
GitHub Desktop
https://desktop.github.com/
MySQL Workbench
https://es.wikipedia.org/wiki/MySQL_Workbench
https://www.mysql.com/products/workbench/
Real VNC
https://www.realvnc.com/en/
https://es.wikipedia.org/wiki/RealVNC
Quick Settings Tiles
https://www.youtube.com/watch?v=xwvU6Pj3ATQ
Código QR
https://www.the-qrcode-generator.com/

3.4.5. Fuentes de imágenes

Portada: https://www.freepik.es/foto-gratis/mano-sujetando-smartphone-pantalla-blanco_987726.htm
Raspberry: https://es.wikipedia.org/wiki/Raspberry_Pi
Relé: https://createc3d.com/es/modulos-y-sensores/579-comprar-modulo-rele-5v-compatible-con-arduino-1-canal-precio-oferta.html
Leds: https://es.dhgate.com/product/5-colors-5mm-round-led-diode-light-bulb-super/449074219.html
Mando de garaje: http://comprarmandogaraje.blogspot.com/2014/10/control-remoto-rf-tipica-diseno-de-pcb.html
https://es.aliexpress.com/item/33020841416.html
Android Studio: https://commons.wikimedia.org/wiki/File:Android_Studio_icon.svg
Webstorm: https://logonoid.com/webstorm-logo/
Visual Studio Code: https://commons.wikimedia.org/wiki/File:Visual_Studio_Code_1.35_icon.svg
Markdown: https://es.wikipedia.org/wiki/Archivo:Markdown-mark.svg
Pandoc: https://groups.google.com/forum/#!topic/pandoc-discuss/1bKIuyBnWaQ/discussion%5B201-225%5D
GitHub desktop: https://www.boel.co.jp/tips/vol32_github_desktop.html
Git: https://blog.muktek.com/git-y-github-627858e07167
MySQL Workbench: https://www.pngocean.com/gratis-png-clipart-dlexd
Real VNC: https://apps.apple.com/es/app/vnc-viewer-remote-desktop/id352019548?l=en
Node: http://ediciones.openexpo.es/event/meetup-nodejs/
Debian: https://felfa.github.io/culturilla/2017/01/09/debian-logo.html
MariaDB: https://www.nicepng.com/ourpic/u2e6a9u2i1w7r5y3_mariadb-logo-png/

Firebase: https://www.elharrakfonts.com/2019/03/logo-firebase.html
Foto de perfil: https://pixabay.com/es/vectors/foto-de-perfil-en-blanco-973460/
Ubuntu: https://ca.wikipedia.org/wiki/Fitxer:Logo-ubuntu_cof-orange-hex.svg
JSON: https://es.wikipedia.org/wiki/JSON
Promesas: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promise
Git push: https://www.pngkit.com/bigpic/u2r5t4i1e6a9e6a9/
Triggers: https://www.youtube.com/watch?v=uTx7xd4ojkk

4. Resultado del proyecto

4.1. Grado de consecución del proyecto

Se han completado todas las características planificadas para este proyecto, y se han añadido algunas de las opcionales.

Android utiliza fragmentos para separar las diferentes pantallas y restringir las diferentes acciones que se pueden realizar. Implementa Firebase para la autenticación de usuarios con la cuenta de google y realiza peticiones al servidor a través de Volley, ante los que muestra mensajes personalizados para los idiomas español e inglés

El servidor web está escrito en JavaScript y se ejecuta sobre Node. Tiene seguridad HTTPS y sirve una API REST con dos rutas disponibles, una es utilizada por la aplicación Android y la otra muestra el estado del servidor. Además implementa interacción con el circuito electrónico externo y con la base de datos, realizando consultas de los usuarios dados de alta, registrando cada conexión y ejecutando el procedure que desactiva los usuarios vencidos diariamente.

La base de datos contiene dos tablas que almacenan las conexiones realizadas y los usuarios. La tabla que almacena los usuarios contiene un campo en el que se guarda la fecha de vencimiento de cada usuario (opcional), y se ha creado un procedimiento que comprueba todos los usuarios y da de baja aquellos que ya no deberían de ser válidos.

La Raspberry Pi corre el sistema operativo Raspbian Pixel aunque podría funcionar con la versión Lite. En él está instalado Node, MariaDB y la última versión del servidor. Además se ha diseñado un circuito para interactuar con el mando y se ha instalado una versión reducida para demostrar su funcionamiento. Para hacer posible las pruebas se ha configurado el router para reenviar puertos a la Raspberry.

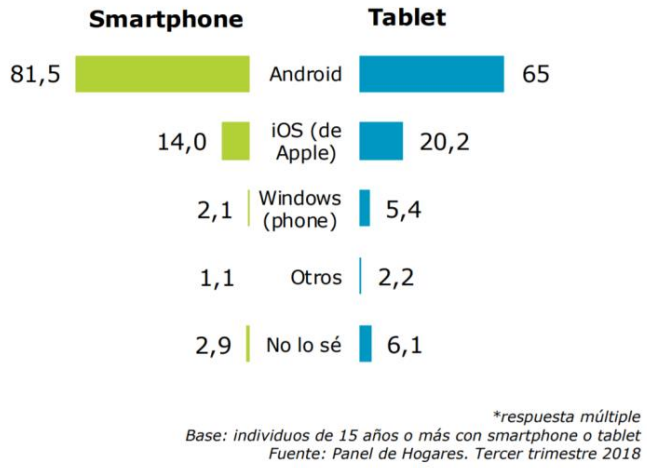
CARACTERÍSTICAS AÑADIDAS A LA PLANIFICACIÓN

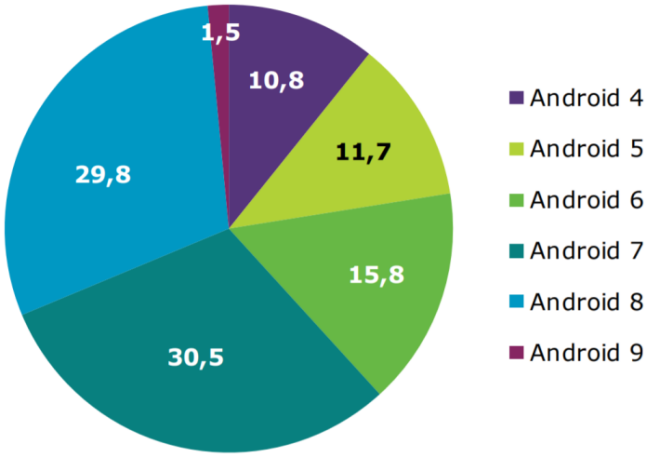
- Vencimiento de usuarios en la base de datos
- Comprobación diaria de dichos usuarios
- Mensajes personalizados en español e inglés en Android y servidor
- Configuración de router y redes
- Diseño de un circuito alternativo simplificado para la demostración
- Implementación de dicho circuito
- Diseño de diagramas

4.2. Propuesta de mejora

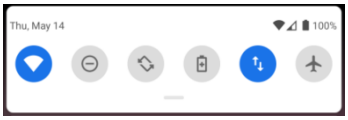
Una futura ampliación de este proyecto debería de tener como objetivo inmediato migrar la aplicación completa al sistema operativo IOS con la intención de dar acceso a todo el que lo desee a este servicio y asimismo eliminar los mandos a radiocontrol.

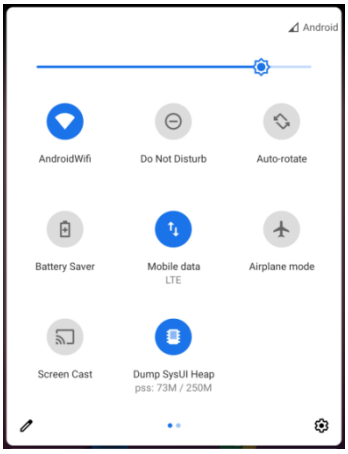
Según el informe 'LA SOCIEDAD EN RED. Transformación digital en España. Informe Anual 2018' elaborado por el equipo del ONTSI y editado por el Ministerio de Economía y Empresa, en 2018 el 81.5% del mercado de teléfonos inteligentes hace uso del sistema operativo Android y el 14% de IOS, permitiendo discriminar al 4.5% restante. De la misma forma indica que los usuarios Android siguen usando en un 10.8% la versión 4 del sistema operativo, siendo conveniente asegurar la funcionalidad de la aplicación para estos dispositivos.





Desechar los mandos existentes permitiría llevar el control de las plazas libres, y obligando a marcar cada plaza cada vez que se aparca permitiría mostrar un mapa del garaje con las plazas libres en la propia interfaz, de esta forma se podría recordar a cada usuario dónde aparcó. Sería conveniente añadir iconos en el launcher y para la barra de notificaciones (Quick Settings Tiles) que permitan abrir la puerta e indicar la plaza ocupada en una sola acción y widgets que muestren las plazas libres.





Para asegurar que los usuarios registren la plaza que ocupan se les debería de insistir con notificaciones, y se les debería de simplificar la tarea con la opción de escanear un código QR situado cercano a la plaza, introduciendo un código o seleccionando la plaza en la propia interfaz. Liberar la plaza debería de ser tan simple como pulsar un botón.

Convendría añadir además la opción de establecer una contraseña tipo patrón, pin o huella dactilar que permita bloquear la aplicación por motivos de seguridad.