Trabajo de Fin de Grado

Turn - Time

Turn - Time

Martín Chinea, Kevin

La Laguna, 4 de septiembre de 2016

D. **Alejandro Pérez Nava**, con N.I.F. 43821179-S profesor Asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Fernando Pérez Nava**, con N.I.F. 42091420-V profesor Titular de Universidad adscrito al Departamento de Estadística, Investigación Operativa y Computación de la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

"*Turn - Time*"

ha sido realizada bajo su dirección por D. **Kevin Martín Chinea**, con N.I.F. 42418683-J.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de septiembre de 2016

Agradecimientos

A Alejandro Pérez Nava y a Fernando Pérez Nava, por darme la oportunidad de desarrollar este proyecto y por su continúo asesoramiento y dirección.

A mi familia, por su apoyo y que gracias a ellos he podido estudiar algo que me gusta y disfruto.

Resumen

El objetivo de este proyecto trata de realizar una aplicación en la cual se facilite la gestión de turnos en cualquier establecimiento. Aprovechando el alto crecimiento de las aplicaciones móviles en estos últimos años, se pretende que el proyecto este orientado a los dispositivos móviles, permitiendo que los usuarios obtengan sus tickets y se les notifique mediante mensaje. Además, complementado con una página web, los empleados puedan gestionar cada una de las colas que tienen, haciendo que estos reciban una mayor retroalimentación del estado de la gestión en su establecimiento a la vez que se desarrolla la tarea de una manera más cómoda y sencilla.

Palabras clave: Turn-Time, turnos, colas, colas de espera, tickets, gestión de colas.

Abstract

The goal of this project is to develop an app to facilitate turn management in establishments such as supermarkets. Considering that apps have become life blood today's smartphone, this project aims to reduce queues allowing users to obtain their tickets and to be notified by means of a message wherever they are. More over, the app is complemented by a web page. This web page allows employees to manage the queues, making this work more comfortable and easier.

Keywords: *Turn-Time, shifts, queues, tickets, management of queues.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En sus principios, la gestión de los turnos en cualquier sistema de administración o tienda venían determinados por un dispensador de tickets, en el cual el cliente coge un ticket y espera a su turno de manera presencial. Actualmente, las empresas han desarrollado sistemas para facilitar a estos usuarios la gestión de este trabajo. Partiendo de este punto, surge el propósito de este proyecto en el que se busca estudiar el mercado existente y la mejor manera de enfrentarse a una posible aplicación que facilite el trabajo, tanto a los dependientes que gestionan la cola, como a los clientes que tienen que esperan a ser atendidos.

1.1. Antecedentes

A día de hoy, la mayoría de empresas o tiendas tienen su gestión de turnos determinadas por el común dispensador de turnos mecánico. En este dispositivo existe un rollo de tickets, en el que el usuario cogería su ticket con el número que le toca y mediante una pantalla los empleados que gestión esa cola muestran el turno al que le toca atender.

La evolución de este tipo de dispensadores ha derivado en diferentes tipos siendo algún ejemplo como los dispensadores de turnos electrónico en atril o el de mesa multiservicio. Estos presentan una similitud con el explicado anteriormente, sólo varían en el método de obtener el ticket del turno, ya que trata de una pantalla táctil en dónde el cliente escoge el turno en la cola de la gestión que desee realizar. En el caso de ciertas empresas como podrían ser algunas oficinas de BBVA[1] o Movistar[2], usan estos dispensadores nombrados en último lugar, y no sólo muestran el turno a atender por pantalla, sino que, también avisan al usuario mediante un mensaje de texto en su dispositivo móvil o smartphone. Estos mensajes, serán enviados con antelación al número de turno pertinente al usuario que, generalmente, serán tres turnos previos. Por último, las mayores novedades en este tema, a pesar de no ser un mercado muy explotado,

[1]Página del banco BBVA: https://www.bbva.es/particulares/index.jsp

[2]Página de la empresa telefónica Movistar: http://www.movistar.es/

son aplicaciones móviles como las instaladas en Carrefour. Esta funcionalidad esta implementada en la aplicación oficial y permiten coger el número del turno cuando el cliente se encuentra en un centro Carrefour y además, se le notificará tres turnos previos al turno del cliente.

1.2. Introducción al problema

El objetivo es desarrollar un proyecto similar al que tenga acceso cualquier comercio, desde pequeños a grandes, permitiéndoles agilizar y simplificar esta administración de las colas realizando toda esta gestión en la nube, tanto en el trabajo que tiene que realizar un empleado en la gestión de las colas como a nivel del cliente que espera.

En concreto, se pretende que el cliente tenga la posibilidad de:

- Acceder a través del código QR a las colas del establecimiento.
- Poder ver en cualquier momento el estado de las colas en las que espera.
- No realizar la espera de manera presencial y pueda realizar otras gestiones, gracias a la notificación mediante un mensaje al móvil de la proximidad de su turno.
- En el caso de no poder realizar este trámite vía móvil, por cualquier motivo, dar posibilidad a realizar la espera de su turno de forma presencial mediante un ticket de papel.

Por otro lado se pretende que el empleado pueda:

- Gestionar cada una de sus colas (crear colas nuevas, eliminar las existentes o avanzar en el turno específico de la cola).
- Observar el saturamiento de las citas en determinadas horas del día.
- Notificar de la proximidad de su turno a los siguientes clientes que esperan.

Permitiendo, finalmente, que este producto se haga un hueco en el mercado existente.

Capítulo 2

Objetivo

Como se ha especificado en el punto 1.1 Antecedentes y el punto 1.2 Introducción al problema, en el problema que se quiere abarcar, el proyecto tendrá dos orientaciones bien definidas. Por un lado se definirá el sistema a usar por la empresa para la gestión de las colas, y por otro, el implantado en el lado del cliente que tendrá como fin acomodar la tarea de espera de su turno a los clientes.

A continuación, se describirá el objetivo de cada uno de de un modo más detallado.

2.1. Desarrollo para la empresa

En el lado de la empresa, como se ha definido anteriormente la gestión se realiza en la nube, por lo que se ha orientado el desarrollo hacia una página web, desde la cual tendrán tendrán acceso los empleados mediante una previa autenticación. Para permitir que cualquier negocio pueda gestionar sus colas mediante esta aplicación, el único requisito es que previamente se hayan registrado mediante un formulario en el cual se guardarán los datos de su negocio en la base de datos. Dentro de esta página, tendrán acceso a diversas vistas en las que podrán ver aspectos como podrían ser los datos de la empresa, los cuales se pueden ser modificados en cualquier momento. Y, entrando más en la problemática a solucionar, tendrán un listado de colas que han creado previamente, pudiendo realizar acciones como crear nuevas, eliminar cualquiera de las existentes o gestionar el turno específico de cada una de las colas. En el caso de esta última acción, el empleado además tendrá la posibilidad de ver la saturación de citas en cada momento del día gracias al listado ordenado por la hora en la que cogieron el turno todos los clientes que esperan o han estado esperando en la cola, dándole con esto una mayor información de la gestión y permitiéndole realizar acciones para solventarlo como podría ser contratar más empleados o añadir nuevas colas en el establecimiento.

2.2. Desarrollo para el cliente

Por el lado del cliente y para facilitarle su espera, se ha desarrollado una aplicación móvil en la que el usuario, mediante un identificador propio al dispositivo. Éste podrá acceder acceder a las colas de cada entidad mediante la captura de su código QR[1], poniéndose así a la cola. Además de esto, podrá ver en que colas se encuentra en espera, mostrando los datos correspondientes como podrían ser:

- El turno por el que va la cola.
- El turno que le toca al cliente.
- La fecha en la que se obtuvo el turno.

Una de las funcionalidades fundamentales implementadas para facilitar la espera al cliente son las notificaciones. El usuario podrá definir cuantos turnos previos al suyo quiere que se le notifique, mandándole la notificación en ese turno determinado. El objetivo de esta funcionalidad es proporcionar al cliente la posibilidad de que la espera de la cola no se haga de manera presencial, y que en caso de que el cliente tenga que realizar cualquier otra actividad pueda realizarla sin miedo a perder el turno ya que se les avisará previamente.

Para los usuarios que no tengan acceso a está aplicación, o a cualquiera de sus funcionalidades, se ha pensado en el desarrollo de una vista, en la parte web del proyecto. En ella cualquier persona podrá acceder y ver todas las colas disponibles, dándole la opción de coger turno en formato papel, permitiéndole realizar la cola en modo presencial.

[1]Tipo de código de barras de matriz (o código de barras bidimensional) legible por máquina que contiene información sobre el elemento al que está unido.

Capítulo 3

Herramientas

En este apartado se trata, de manera previa a las fases llevadas a cabo para realizar el proyecto, las herramientas y lenguajes de programación que se han usado para obtener el resultado definido.

3.1. MAMP

MAMP[1] hace referencia al conjunto de programas software usados para el desarrollo de sitios web dinámicos. En concreto, éste se centra en sistemas operativos Apple. Sus siglas corresponden con:

- Mac: Sistema operativo (Mac OS[2], Windows[3], Linux[4]).
- Apache: Servidor Web[5].
- MySQL: Sistema Gestor de Bases de Datos.
- PHP, Perl ó Python: lenguaje de programación usado en el sitio web.

En la Figura 3.1 se puede ver el logo de la herramienta MAMP.

3.2. MySQL

Sistema multihilo y multiusuario para la gestión de bases de datos relacionales. Este sistema almacena los datos en tablas diversas, proporcionando flexibilidad y velocidad en su tratamiento, y otorgando la posibilidad de relacionar estas

[1]Página oficial: https://www.mamp.info/en/

[2]Sistemas operativos desarrollados por Apple Inc.

[3]Sistemas operativos gráficos desarrollados, comercializados y vendidos por Microsoft.

[4]Sistema operativo Unix montado bajo el modelo de desarrollo de software libre y de código y distribución abierto

[5]Sistema informático que procesa las solicitudes a través de HTTP, el protocolo de red básica que se utiliza para distribuir información en la World Wide Web.

Figura 3.1: Logo de Mamp.

tablas con el fin de realizar consultas de una combinación de datos. Está desarrollado mediante una licencia dual GPL/Licencia Comercial y es open source.

En la Figura 3.2 se puede ver el logo de MySQL.

Figura 3.2: Logo de MySQL.

3.3. PHP

PHP (Hypertext Preprocessor), se trata de un lenguaje de código abierto, adecuado para el desarrollo web, el cual se puede ser incrustado en HTML. Lo que destaca de este lenguaje de programación es que es un lenguaje en el lado del servidor, generá el HTML[6] y se lo envía al cliente, dónde el navegador procesa esta información y muestra la página definida.

Al estar enfocada a la programación en el lado del servidor, se puede realizar cualquier interfaz que se puede hacer en otro programa CGI (Common Gateway Interface)[7]. Algunos ejemplos pueden ser formularios, generar páginas con contenidos dinámicos, o enviar o recibir cookies, por lo que todo esto se

[6]Lenguaje de marcado para la elaboración de páginas y aplicaciones web. Con Cascading Style Sheets (CSS) y JavaScript, forman la base para la World Wide Web. https://en.wikipedia.org/wiki/HTML

[7]Estándar para transferir datos entre el cliente (navegador web) y un programa ejecutado en un servidor web.

ha implementado para generar las funcionalidades y aspectos necesarios en la funcionalidades web.

PHP no se limita a generar HTML, también se incluyen la creación de imágenes, ficheros de PDF, etc. Estas características se han aprovechado mediante ciertas librerías como pueden ser *PHP QR Code* para generar los códigos QR o *FPDF*, que permite crear archivos en formato PDF.

Por último, una de las características más potentes y destacables de PHP, es la sencillez y fácilidad de conexión de este lenguaje con un amplio abanico de bases de datos (siempre que admitan el estándar de Conexión Abierta a Bases de Datos).

En la Figura 3.3 se puede ver el logo del lenguaje PHP.

Figura 3.3: Logo de MySQL.

3.4. Bootstrap

Bootstrap[8] es un framework, que permite dar forma a una página web mediante liberías CSS en donde existen elementos como tipográficas, botones, cuadros, menús, etc. Es una herramienta que ayuda en el desarrollo de crear interfaces de usuario limpias y completamente adaptables a cualquier dispositivo, independientemente del tamaño de la pantalla.

En la Figura 3.4 se puede ver el logo del Framework Bootstrap.

Figura 3.4: Logo de Bootstrap.

[8]Página oficial: http://getbootstrap.com/

3.5. Android Studio

Entorno de desarrollo integrado para la plataforma Android[9], actualmente es su IDE oficial, usa una licencia de software libre Apache 2.0[10], está programado en Java[11] y es multiplataforma[12]. Otras de las muchas características que posee es que:

- Posee soporte para programar aplicaciones para Android Wear[13].
- Herramienta Gradle[14] para gestionar y automizar la construcción de proyectos.
- Posibilidad del control de versiones.
- Vista previa en distintos dispositivos y resoluciones.
- Editor de diseño.

En la Figura 3.5 se puede ver el logo del IDE Android Studio.

Figura 3.5: Logo de Android Studio.

3.6. Java

Lenguaje de programación de próposito general, concurrente, imperativo y que se centra en el paradigma orientado a objetos. Se trata de un lenguaje diseñado para permitir la ejecución de un mismo programa en múltiples sistemas operativos, lo que significa que programas desarrollados en este lenguaje pueden

[9]Página oficial: https://www.android.com/

[10]Licencia de software libre permisiva creada por la Apache Software Foundation. Esta licencia requiere la conservación del aviso de derecho de autor y el descargo de responsabilidad, pero no es una licencia copyleft, ya que no requiere la redistribución del código fuente cuando se distribuyen versiones modificadas.

[11]Lenguaje de programación.

[12]Programas informáticos o métodos y conceptos de cómputo que son implementados e interoperan en múltiples plataformas informáticas.

[13]Versión del sistema operativo Android de Google diseñado para smartwatches y otros elementos vestibles.

[14]Herramienta para automatizar todo el proceso de construcción de un proyecto (compilar, testing,empaquetado, ..).

ejecutarse en cualquier tipo de hardware. Java ha tenido un gran crecimiento y amplitud en distintos ámbitos de la industria de la informática, desde dispositivos móviles y sistemas embebidos, aplicaciones de escritorio, sistemas de servidor o incluso en el propio navegador web mediante Applets[15].

En la Figura 3.6 se puede ver el logo del lenguaje de programación Java.

Figura 3.6: Logo de Java.

3.7. Firebase

Firebase[16] es un servicio de google que funciona como nube y back-end. Esta plataforma proporciona funcionalidades orientadas al desarrollo de software tanto para aplicaciones móviles cómo web. El producto principal es una base de datos en tiempo real con una API en dónde los desarrolladores pueden almacenar y sincronizar datos a través de múltiples clientes. Otras funcionalidades implementadas en esta plataformas de las muchas que poseen pueden ser:

- Autenticación.
- Notificaciones.
- Analíticas.
- Links dinámicos.

En la Figura 3.7 se puede ver el logo de la herramienta Firebase.

Figura 3.7: Logo de Firebase.

[15]Programas escritos en Java enfocados para su ejecución en un navegador web
[16]Página oficial: https://www.firebase.com/

Capítulo 4

Fases y desarrollo del proyecto

En este apartado se tratan los pasos que se han realizado para el desarrollo de este proyecto, intentando que sea lo más cercano a lo desarrollado, convirtiéndose en una guía para que se pueda repetir la experiencia. En la Figura 4.1, se encuentra el logo desarrollado para la aplicación.

Figura 4.1: Logo de la aplicación.

4.1. Página Web

El desarrollo de la página web, debido a su futura implementación en un servidor dónde se conectará una base de datos y tendrán acceso todos los usuarios de la aplicación, es necesario montarla en un servidor o en su defecto, algún tipo de herramienta que proporcione está funcionalidad. Para esto, recurrimos a la herramienta nombrada en el apartado anterior, *Mamp*, esta herramienta proporciona dos versiones, la normal y la PRO. Para el desarrollo de este proyecto la versión normal sirve, ya que nos permitirá crear un servidor propio en local. La instalación de esta herramienta es simple, lo primero que hay que realizar es descargar el binario para la instalación, el programa nos solicitará aspectos

como aceptar la licencia y una ubicación con el fin de realizar completamente la instalación. Una vez instalado, el programa se encuentra en la carpeta Aplicaciones, al abrirlo, se muestra una interfaz bastante simple. Este servidor, como se ha nombrado, incluye Apache y MySQL, en la opción de *Abrir página de inicio*, se puede ver la página de inicio de Mamp, en dónde proporciona el puerto en el que esta trabajando el Mamp, por defecto el 8888. En la ventana principal del Mamp, e iniciando los servidores, se puede acceder a la página web a través del navegador en la dirección localhost:8888. Otro de los accesos que proporciona la interfaz, *Preferencias*, nos indican características como pueden ser:

- Iniciar los servidores.
- La versión de PHP.
- El puerto en el que se ejecuta el servidor.
- La dirección *htdocs* para guardar los ficheros.

Esta dirección nombrada es un aparatado básico para el desarrollo de la web, dentro de estas preferencias en Apache, vemos la dirección para poder modificar lo que el cliente va a tener en su ventana del navegador, desarrollando y creando los ficheros en esta carpeta *htdocs* se consigue cambiar lo que el servidor muestra en el puerto definido.

Para desarrollar la página web, lo primero que hay que realizar es una vista dónde los establecimientos puedan darse de alta, Figura 4.2. Para ello lo que hay que realizar es un formulario, en el que se le solicita al usuario ciertos detalles de la empresa, estos datos son:

- El nombre de la empresa.
- La contraseña.
- Un número de contacto.
- La dirección.
- Una breve descripción de la empresa.
- La URL de la imagen para la foto de perfil.

Estos serán los atributos, que unidos a un identificador generado automáticamente formen la primera tabla de la base de datos, *Entitys*.

Ya definida la vista para las creaciones de cuentas, se pasa a la vista principal, en donde la entidad se loguea y entra en la aplicación, Figura 4.3. Está funcionalidad se basa en realizar de forma similar al anterior un formulario, comprobando en el lado del servidor, cuando se realice la petición POST, que el

nombre de usuario existe y la contraseña introducida coincide con la definida en la tabla. En el caso de que sea correcto, posteriormente se iniciaría la la variable $_SESSION, y, comprobando previamente, que esta variable está inicializada en el resto de vistas nos aseguramos de que el usuario esta autenticado.

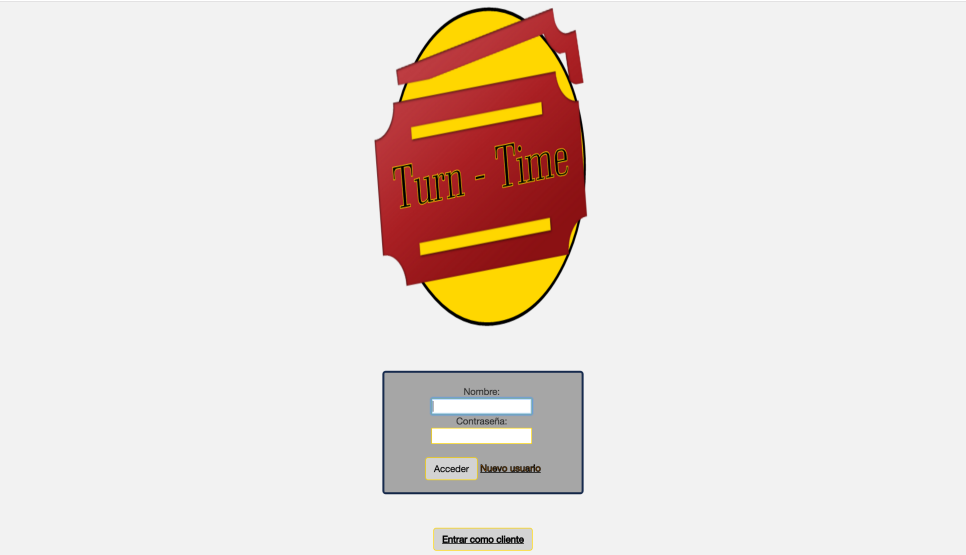


Figura 4.2: Menú de logueado.

Función para comprobar si un usuario que accede a la página esta autenticado:

```
function isAuth(){
        if (!isset($_SESSION) ){
                session_start();
        }

        if (isset($_SESSION['user'])){
        return true;
        } else {
        return false;
        }
}
```

Para el desarrollo del estilo de la aplicación se ha añadido el framework Bootstrap, introduciendo los elementos necesarios que se han creído adecuados para facilitar la interfaz de usuario. De igual forma se ha implementando un estilo propio, mediante un fichero con formato CSS para proporcionarle un estilo más personal al ya definido en Bootstrap y, también página web en general. Todo esto se ha desarrollado en cada una de las vistas de la web para así tener una temática común.

Links de los recursos de bootstrap:

```
        <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/bootstrap
            /3.3.6/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js
        "></script>
    <script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.
        js"></script>
```

4.2. Gestión de las colas

Para la gestión de las colas es necesario proporcionar a este empleado, que se ha logueado de manera previa en la página web, un listado de las colas que tiene creadas, mediante la tabla *Queues*, que tiene como atributos:

- Un identificador propio.
- El id de la entidad logueada.
- El nombre.
- La dirección.
- El turno por el que va en la cola.

Con esta tabla se realiza la consulta a través de PHP y se muestra en html. En el caso de que este usuario quiera crear una nueva cola lo único que hay que hacer es añadirle un acceso en el que pueda definir un nombre, y haciendo la correspondiente inserción en la base de datos crearla. Para definir el identificador de esta cola se generá un número aleatorio comprobando previamente que no exista ya en la tabla una misma cola con dicho identificador.

Uno de los puntos más importante para la creación de las colas es generar un código QR, en el cual guardaremos el identificador de la cola para posteriormente mostrarlo y que los clientes puedan acceder a este, capturarlo y acceder a la cola de espera. Esta funcionalidad la desarrollamos con PHPQRCode[1], se trata de una pequeña librería de software libre, para crear códigos QR. Para llevar a cabo esto simplemente hay que descargar los ficheros, en concreto el fichero necesario es *phpqrcode.php*, con el cual se generá un fichero en formato png determinando el tamaño y los datos que guardará este código.

En el caso de que se quiera eliminar una determinada cola, con un simple enlace que realice un delete de la cola específica en la base de datos y eliminando el fichero que representa el código QR bastaría.

Si un empleado quisierá trabajar con una cola determinada, el propio elemento que muestra la cola es un enlace a sus datos. Datos como pueden ser:

[1]Página oficial: http://phpqrcode.sourceforge.net/

Figura 4.3: Gestión de las colas.

- Su código QR.
- El identificador.
- El turno por el que va la cola..

Además, en esta vista de los detalles de la cola obtenemos el listado de usuarios de la cola con la ayuda de una nueva tabla, llamada *UsersQueue*. Esta nueva tabla esta formada por los atributos:

- La posición del usuario.
- El identificador de la cola a la que pertenece.
- El identificador del usuario del turno.
- La fecha en la que ha sido creado el turno.
- Booleano que define si se ha atendido al usuario o no.

Para gestionar estos turnos simplemente se realizan dos botones que funcionan como enlaces. Estos enlaces redirigirán a la misma página incrementando el valor del atributo del turno a atender en la cola y notificando a los siguientes usuarios (se entrará más en detalle a continuación). La diferencia entre un botón y otro viene reflejado en que mediante el envío de un valor booleano, dependiendo de cuál sea el botón, cambiará el estado del usuario del turno actual a atendido o no atendido dependiendo de si se realiza la gestión o no. Estos cambios se verán reflejados en la parte baja de la vista, en el listado completo de usuarios que esperan o han esperado en la cola. Esta vista la podemos contemplar en las Figuras 4.5 y 4.6.

Figura 4.4: Ejemplo código QR.

Figura 4.5: Listado de usuarios de las colas.

4.3. App móvil

Para el desarrollo de la app móvil, simplemente es necesario realizar un *layout*[2] por cada *Activity* [3]. En los *Activitys* se desarrolla la asignación de los *layouts* en cada una de las vistas de la aplicación y proporcionan funcionalidad a los elementos que contengan.

La primera vista a desarrollar, la que podemos ver en la Figura 4.7, será el menú de inicio. Esta está desarrollada con el logo en el centro y un botón en la parte inferior, en dónde el *activity* llamado *MainView* le proporciona la funcionalidad de cambiar de *activity* al *activity MainMenu*. En este nuevo *activity*

[2]https://developer.android.com/guide/topics/ui/declaring-layout.html
[3]https://developer.android.com/reference/android/app/Activity.html

Figura 4.6: Listado de usuarios de las colas.

se encuentra el menú dónde tenemos tres nuevos botones, que dependiendo de cuál sea se le cambiará a una nueva vista con su correspondiente funcionalidad, implementada con su correspondiente *activity*. El primer botón establecido se trata de la captura del código QR, el siguiente, la vista que muestra cada uno de los tickets en los que el usuario tiene y está esperando a su turno, y por último, la configuración para las notificaciones. Esta vista se puede observar en la Figura 4.8. En la Figura 4.9 se puede contemplar el flujo de estos *activitys*, más adelante se describirán en mayor profundidad cada funcionalidad correspondiendo a cada uno de sus apartados.

Para desarrollar el estilo de la aplicación móvil en primer lugar se tiene que cambiar los iconos por el logo personalizado en la carpeta *minmap*. Se sustituye cada uno de los archivos por defecto de Android por el logo, estas nuevas imágenes tienen que tener las dimensiones determinas del archivo que se va a sustituir. Posteriormente, en el fichero *app/res/values/colors.xml* se define el color de fondo, colores primarios, y cualquier otra variables necesarias a usar en la temática. Del mismo modo, en la carpeta *app/res/drawable/* creamos un fichero *personalButton.xml*, que se tratará del recurso que le proporcionará el estilo a los botones dándole un gradiente y un borde haciendo que los botones estén personalizados.

Código estilo del botón:

```
    <shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
<gradient android:startColor="#FFF"
    android:endColor="#D3D3D3"
    android:angle="270" />
<corners android:radius="3dp" />
<stroke android:width="2px" android:color="#FFD700" />
    </shape>
```

Figura 4.7: Menú de inicial.

Con esto se consigue aplicar a la app un estilo similar a la temática web.

4.4. Conexión App - BDD

Para el desarrollo de la conexión de la App a la base de datos, desarrollamos una clase a la que llamaremos, *Database*, y que, en primer lugar, tendrá tres constantes, la ruta del servidor, y el nombre de los dos ficheros con los que interactuarán, el fichero con el que accederemos al listado de colas del usuario, y otra, que nos insertará al usuario en nuestra tabla de la base de datos *UsersQueue* haciendo que este a la espera.

Esta clase estará formada por dos clases internas que difieren en su uso según de la función de la app que el usuario este ejecutando, el diagrama UML de estas clases esta representado en la Figura 4.10. Estas dos clases internas herendan de la clase *AsyncTask*, clase que desarrolla la actividad en segundo plano mediante multihilo, de manera similar que la clase *Thread* pero de una forma más optima y sencilla a la hora de codificar.

Figura 4.8: Menú de opciones de la app.

La primera clase privada desarrollada es la clase *PostRequest*. Es usada en la lectura de los códigos QR. Al sobreescribir el método *doInBackground*, se guardan en una variable todas las variables que se enviarán al servidor, variables necesarias como son:

- El identificador de la cola.
- El id del dispositivo que ejecuta la aplicación.

Y otras que se irán añadiendo más adelante en cuando se avance el proyecto. Una vez realizada la conexión con el servidor, desde este, lo único que habrá que hacer es comprobar que los datos son correctos (por ejemplo que la cola capturada existe) y en el caso de que todo sea correcto añadir a un nuevo usuario a la espera de su turno en dicha cola. El servidor nos responderá con un código indicándonos si la inserción del usuario a sido realizada con éxito o no, para poder notificar al usuario mediante un *Toast*[4] del estado de la inserción. Todo

[4]Notificación temporal que le aparece al usuario en la aplicación.

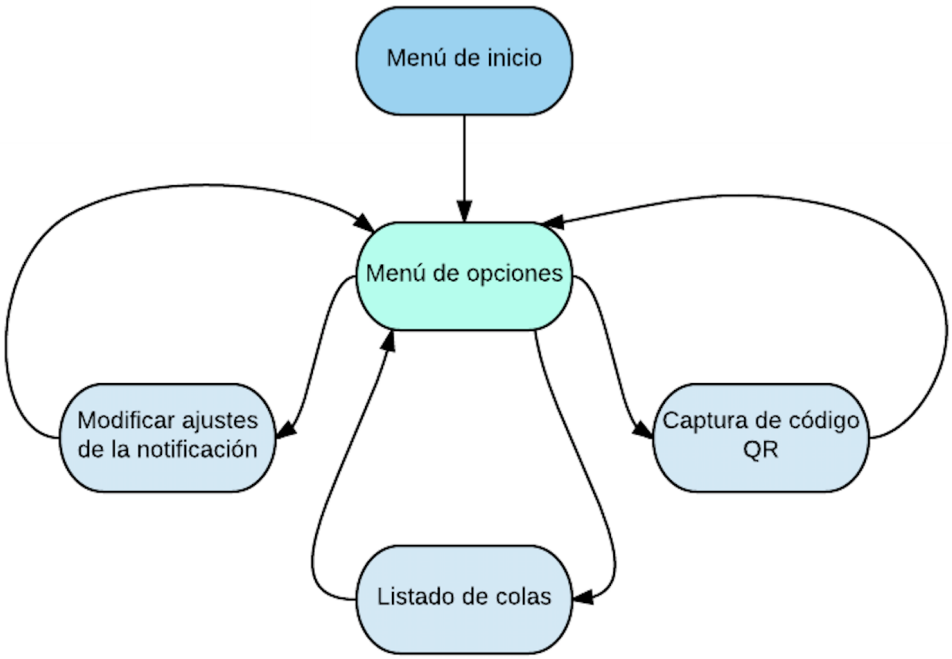


Figura 4.9: Diagrama del flujo de las actividades.

esto es comprobado en el método *onPostExecute*, método que se ejecuta cuando la tarea en segundo plano es finalizada.

La segunda clase interna, definida como *GetRequest*, es usada para mostrar en el *layout* correspondiente, todas las colas en las que el usuario se encuentra en espera. Para ello, de similar forma a la clase anterior se sobrescribe el método *doInBackground*. En este caso, el único parámetro que tenemos que enviar al servidor es el identificador propio al dispositivo, dónde a nivel del servidor y haciendo una consulta a la base de datos, se seleccionan todas las t-uplas que correspondan al identificador del usuario en las que no haya sido atendido, guardándolas en un array. Posteriormente parseado en formato json, este será el elemento devuelto a la aplicación móvil. Ya desde la app móvil, se tratará este json creando un array en el que guardamos cada uno de los items, pasándolo a nuestro adaptador y generando el contenido del elemento de Android *Listview* mostrándolo por la pantalla al usuario móvil.

El primer paso para realizar el adaptador, es el de crear un *layout* que representa el estilo que tendrá cada item del *ListView*. Se desarrollará mostrando mediante *TextView* los atributos del ticket:

- Nombre de la entidad.
- Nombre de la cola.
- Posición de la cola.
- Turno del usuario.

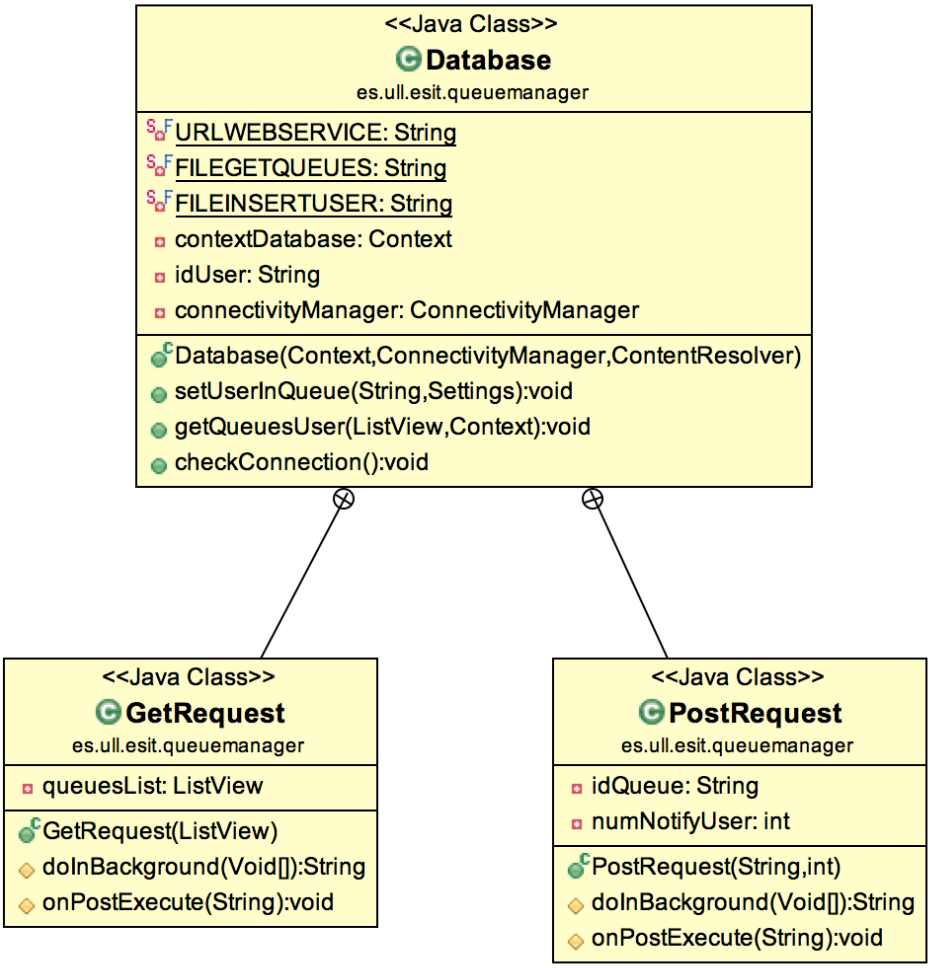


Figura 4.10: Estructura de la conexión a la base de datos.

- Fecha de obtención del ticket.

También se puede añadir cualquier otro elemento, como podrían ser imágenes, para así obtener cada item del *ListView* más personalizado.

De igual forma, se desarrollará una clase *Ticket*, Figura 4.11, que representará estos elementos. Esta clase está formada por los atributos nombrados en el *Layout* anterior y sus correspondientes Setters y Getters.

Esta nueva clase *Ticket* es la que usará el adaptador *TicketAdapter*. Este adaptador será el encargado de visualizar los elementos del *ListView*. Lo más importante de esta clase es el método *getView* encargado de personalizar la lista mediante el uso de la clase *Ticket*. Podemos contemplar la clase *TicketAdapter* en la Figura 4.12.

Ahora, simplemente, usando el adaptador con el array de items parseado del json anterior, mostramos todos los elementos como se muestra a continuación:

```
TicketAdapter adapter = new TicketAdapter((Activity) getContext(),
    listTickets);
```

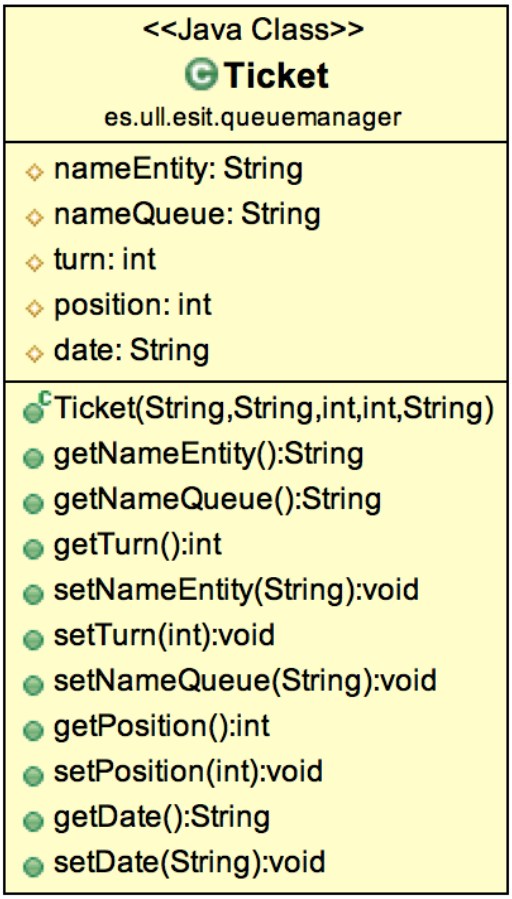


Figura 4.11: Clase Ticket.

```
    getQueuesList().setAdapter(adapter);
```

El resultado final del listado de tickets lo podemos contemplar en la Figura 4.13.

Para ampliar la funcionalidad de la lista de tickets y permitiendo que un usuario interactúe con ellos es posible añadirle un listener. Este nuevo listener da la funcionalidad de activar un nuevo *Activity*, al cuál, se le pasa el objeto *Ticket* clickado[5]. Este nuevo *Activity*, llamado *SimpleTicket* muestra por pantalla los mismos datos que se pueden ver en un item del *ListView*, lo que usando un nuevo *layout* propio a ese activity, el resultado se puede ver en la Figura 4.14.

Código correspondiente al listener:

```
        getQueuesList().setOnItemClickListener(new AdapterView.
            OnItemClickListener() {
                @Override
```

[5]Para pasar el elemento de un Activity a otro es necesario que el objeto que se pase implemente la clase *Serializable*. Esta clase permite que el programa en Java pueda permitir al objeto convertirse en un conjunto de bytes los cuales posteriormente se pueden recuperar y obtener dicho objeto.

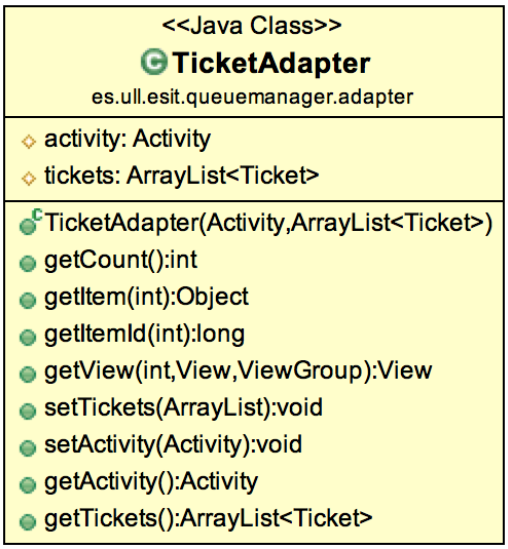


Figura 4.12: Clase adaptador.

```
            public void onItemClick(AdapterView<?> parent, View view, int
                position, long id) {
                    Intent intent = new Intent((Activity) getContext(),
                        ShowSimpleTicket.class);
                    intent.putExtra("ticket", listTickets.get(position));
                    getContext().startActivity(intent);
            }
        });
```

4.5. Lectura código QR

Para desarrollar este apartado se recurre a la librería *ZXing*[6], librería de procesamiento de imágenes de código de barras. A partir de esta librería lo que hay que añadir en la app móvil es la nueva clase *IntentIntegrator*. Con esta nueva instancia de la clase se inicia el escaner al clickar el botón que nos interese mediante la función *initiateScan()*, en este caso el bóton *Capturar QR*. Cuando se llama a esta función se inicia una nueva Activity, por lo que redefiniendo la función *onActivityResult* e instanciando la clase *IntentResult* se parsea el intent que se recibe. Para que así, mediante esta clase poder obtener el resultado del código QR y el formato, posteriormente se procesan estos datos, enviándolos al servidor y añadiendo al usuario a la base de datos, mediante la inserción correspondiente en la base de datos. Previamente a la inserción se comprueba que el código escaneado es válido y existe una cola que corresponda con este.

Esta clase *IntentResult*, se trata de la clase que encapsula el resultado de la captura a la exploración de un código de barras de la clase *IntentIntegrator*, en dónde esta última clase se encarga de crear un nuevo *Intent*, iniciando el escáner

[6]https://github.com/zxing/zxing

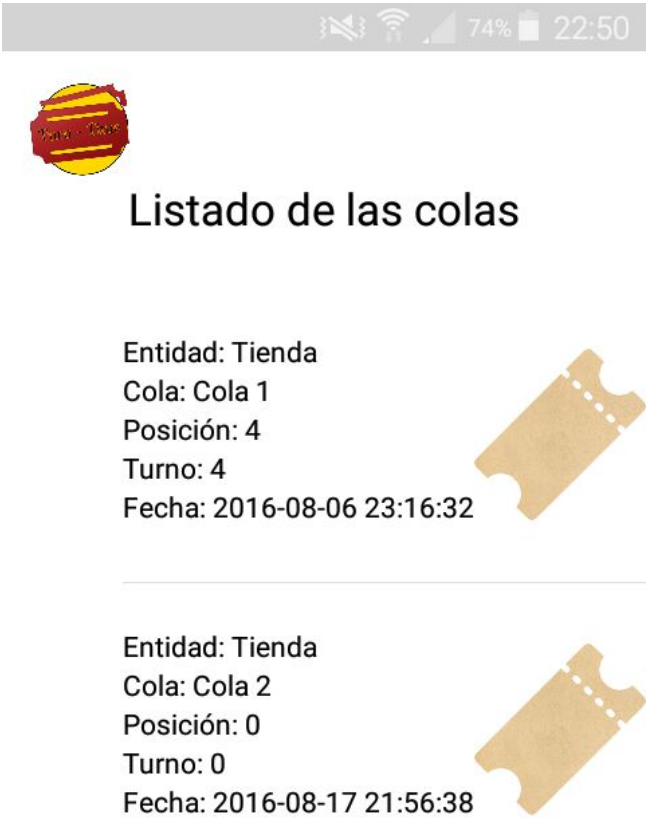


Figura 4.13: Vista del listado de colas en las que espera un usuario.

con la cámara y que hasta no capturar correctamente un código adecuado no finaliza dicho *Intent*, en caso de que se capture el código se parsea y devuelve el resultado obtenido mediante la clase ya nombrada, *IntentResult*, para tratar estos datos como hemos dicho con anterioridad.

4.6. Tickets en pdf

Pensando en los usuarios que por cualquier motivo no tengan un teléfono móvil a mano, o no dispongan de acceso a Internet se ha pensado una vista, como se puede ver en la Figura 4.15. Esta nueva vista es accesible desde la ventana principal de la aplicación web sin necesidad de autentificarse. Para crearla, desde el servidor se trata la consulta a la base de datos que da como resultado el listado de todas las colas existentes (con el respectivo nombre de la entidad que la crea, consiguiendo con esto diferenciar mejor una cola de otra).

Para generar está nueva posición en la cola, se realiza de forma similar al paso descrito en el apartado *Lectura código QR*, cambiando simplemente atributos

Figura 4.14: Ticket concreto del listado de usuarios.

como pueden ser el id del usuario, y la posición en la que se quiere que se le avise por un valor por defecto a la hora de añadir un usuario a la espera en la cola. La parte que difiere es la identificación del turno del cliente frente al empleado, ya que para esto se recurré a la librería FPDF[7], la cuál proporciona scripts PHP con los que generar archivos PDF, creando así un ticket personalidado con los datos del turno y cualquier otro detalle cómo podría ser el logo de la aplicación o la fecha en la que se creo. El resultado final lo podemos ver en la Figura 4.16.

4.7. Firebase y notificaciones push

El primer paso para poder realizar mensajes push, es crear una cuenta en Firebase, herramienta descrita brevemente en el apartado de herramientas. Una vez en el Dashboard de la página, podemos ver nuestros proyectos y crear otros nuevos. Se creará un nuevo proyecto para usarlo para la gestión de las

[7]http://www.fpdf.org/

Figura 4.15: Vista del listado público de colas.

notificaciones.

Ya creado, añadimos el proyecto a una aplicación Android como se puede ver en la Figura 4.17, solicitando con esto que se defina el nombre del paquete asignado a la aplicación.

Esto creará un archivo en formato json que se tiene que guardar en la carpeta app de nuestro proyecto en Android, para posteriormente configurar la aplicación móvil añadiendole las dependencias y plugin necesarios para usar esta funcionalidad.

El primer archivo para añadir las dependencias es en el archivo build.gradle a nivel de root. Se añaden las dependencia como se muestra a continuación.

```
        dependencies {
        classpath 'com.android.tools.build:gradle:2.1.0'

        classpath 'com.google.gms:google-services:3.0.0'
    }
```

A nivel aplicación también añadimos nuevas dependencias, en el fichero con el mismo nombre, build.gradle, añadimos la dependencia de Firebase y el plugin de los servicios de google.

```
        dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        testCompile 'junit:junit:4.12'
        compile 'com.google.android.gms:play-services-ads:9.0.2'
```

Figura 4.16: Ejemplo de ticket.

Figura 4.17: Añadir Firebase a la app.

```
    compile 'com.google.firebase:firebase-messaging:9.0.0'
    compile 'com.squareup.okhttp3:okhttp:3.2.0'
    }

    apply plugin: 'com.google.gms.google-services'
```

Estos pasos vienen explicados a la hora de añadir una aplicación android en el proyecto en la página web de Firebase.

Lo siguiente que hay que desarrollar son los *Services*[8], para ello lo primero que se necesita es añadirlos en el Manifiest, haciendo que estos se ejecuten en segundo plano cuando se inicia la aplicación.

[8]https://developer.android.com/reference/android/app/Service.html

Services activos al iniciar la aplicación definidos en el fichero *AndroidManifiest.xml*:

```
<service android:name=".notification.NotificationMessage">
        <intent-filter>
                <action android:name="com.google.firebase.
                    MESSAGING_EVENT" />
        </intent-filter>
</service>

<service android:name=".notification.InstanceIDService">
        <intent-filter>
                <action android:name="com.google.firebase.
                    INSTANCE_ID_EVENT" />
        </intent-filter>
</service>
```

El primer *service* se puede ver representado por su diagrama UML en la Figura 4.18 y ha sido definido con el nombre *NotificationMessage*. Hereda de la clase *FirebaseMessagingService*, en dónde, sobreescribiendo el método *onMessageReceived*, al cual se le pasa un objeto *RemoteMessage*, es usado para obtener el mensaje recibido y que usando los atributos que reciben en una función propia, se crea la notificación para el usuario añadiéndole aspectos como pueden ser la vibración o el sonido[9].

Código para crear la notificación:

```
NotificationCompat.Builder notificationBuilder = new NotificationCompat.
    Builder(this)
        .setAutoCancel(true)
        .setContentTitle(getTitle())
        .setContentText(getMessage())
        .setSmallIcon(R.mipmap.ic_launcher)
        .setContentIntent(pendingIntent);

        if(((Settings) this.getApplication()).isVibrate()){
        notificationBuilder.setVibrate(PATTERN);
}

if(((Settings) this.getApplication()).isSound()){
        notificationBuilder.setSound(defaultSoundUri);
}

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

[9]El *Intent* relacionado con está notificación se trata del *SimpleTicket*, al cuál se le pasa un objeto *Ticket* creado a partir del array de la notificación de Firebase.

```
notificationManager.notify(0, notificationBuilder.build());
```

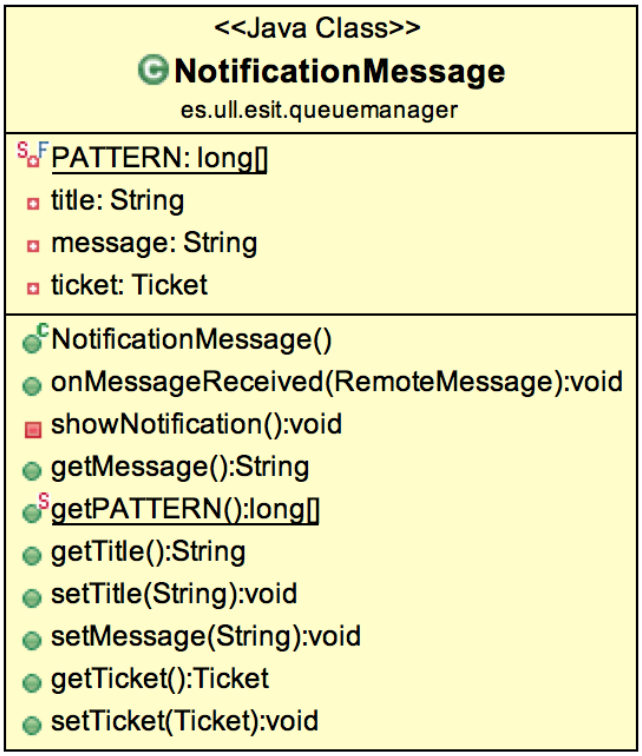


Figura 4.18: Clase notificación de mensaje.

El segundo *service*, representado por su diagrama UML en la Figura 4.19 y definido con el nombre *InstanceIDService*. Clase que hereda de *FirebaseInstanceIdService*. En esta se sobrescribe la función *onTokenRefresh*, para guardar en una variable propia el identificador que usa Firebase para distinguir cada uno de los dispositivos con los que comunicarse.

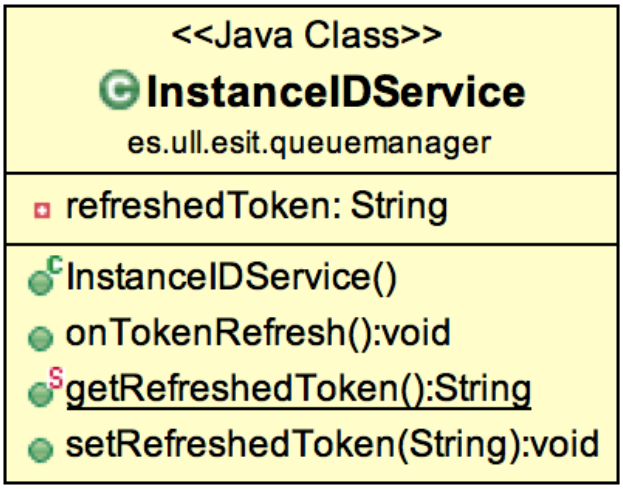


Figura 4.19: Clase identificador para la notificación del mensaje.

El siguiente paso a trabajar es a la hora de capturar un código QR de una cola, para esto se le añade un nuevo atributo a los atributos que se mandan al servidor. Este nuevo atributo hace referencia a este último identificador propio a cada dispositivo que usa Firebase para los mensajes. Realizando en el servidor una comprobación de si existe el usuario actual debido a un uso previo de esta aplicación actualizamos su *token* único y poder notificarlo exclusivamente de otros dispositivos, en el caso de que no exista en esa tabla ningún usuario, se crea una nueva fila con dicho identificador y el*token*, que como hemos dicho es identificativo para Firebase.

De igual forma, se sube y guarda un atributo nuevo en la fila del usuario que espera en la cola. Este atributo reflejará el puesto en el que el usuario quiere que se le notifique. La asignación de este valor lo podemos ver en el *layout* que se muestra en la Figura 4.20. Con esta vista de la app modificamos los valores a la clase *Settings*, clase que se inicializa al abrir la aplicación y usa los valores en determinadas funcionalidades, el valor del turno de notificación en este caso, y más adelante, a la hora de crear una notificación comprobará si el usuario tiene activada la vibración y el sonido.

El último paso para realizar las notificaciones, es realizar una consulta de los siguientes usuarios en la cola cuando se realiza un cambio de turno. Independientemente de si se atiende o no al cliente, se comprueba cada uno de estos usuarios que continúan a la espera, en el caso de que el turno actual de la cola sea igual al turno en el que se le quiere notificar y este usuario específico no haya sido atendido se le notificará. Para estas notificicaciones se realiza una conexión con la URL de Firebase [10], a la cuál se le pasa como cabecera un identificador[11] único extraído del proyecto creado en Firebase y un array compuesto con los datos del ticket actual de la cola. Este array que se le pasa no sólo esta formado por el mensaje de la notificación si no por todos los datos que forman la clase *Ticket*, permitiendo crear una instancia de esta y que la notificación sea un acceso a la actividad *SimpleTicket.java*, de forma similar al subapartado cuatro.

[10]Dirección de la herramienta Firebase: https://android.googleapis.com/gcm/send

[11]Este identificador se puede encontrar en la configuración del proyecto, Figura 4.20.

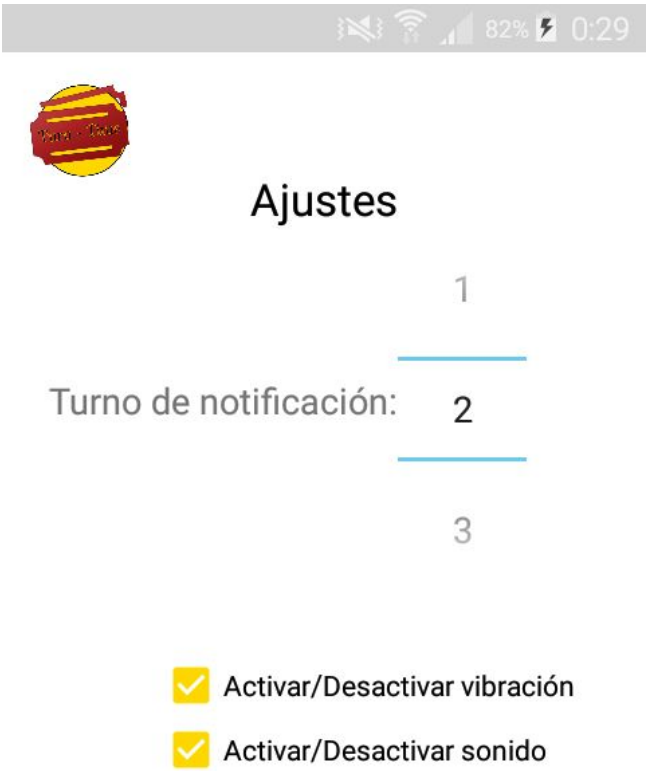


Figura 4.20: Layout clase Settings.

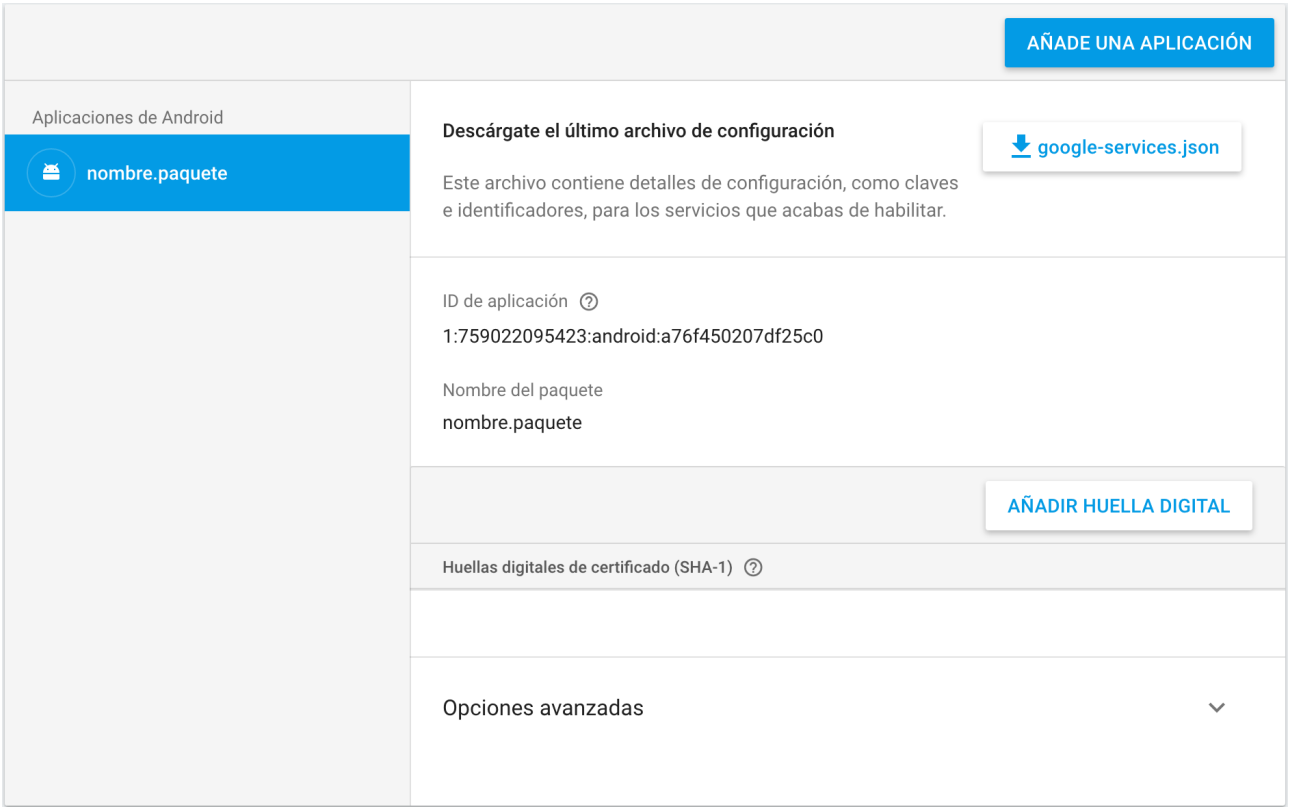


Figura 4.21: Ejemplo de configuración de un proyecto en Firebase.

Capítulo 5

Conclusiones y líneas futuras

En este Trabajo Final de Grado se ha afrontado el desarrollo de un proyecto con el fin de integrarlo en la sociedad. Se han desarrollado distintas funcionalidades implementandas en cada parte del proyecto mediante el uso de amplias tecnologías. Esto a proporcionado a nivel personal la oportunidad de trabajar con estas herramientas, adentrándonos con diversa profundidad en cada una ellas, y combinándolas con el fin de alcanzar del objetivo final.

En concreto, el tema del proyecto ha aportado la posibilidad de realizar un estudio de la sociedad y del mercado, afrontando desde un comienzo la funcionalidad y necesidad que requiere el objetivo de la aplicación. Trabajando en cada una de sus partes de como se comportaría un usuario que estuviera usando nuestra aplicación en cada momento, para así entender las necesidades y problemas que pueden llegar a tener.

El estado actual del proyecto, se puede enfocar en la incorporación de está aplicación a los establecimientos, ya que actualmente tiene desarrollado las funcionalidades para que cualquier establecimiento pueda crear sus cuentas, gestionar sus colas y que sus clientes puedan acceder a cada una de estas, tanto a nivel de aplicación móvil como a nivel de un ticket de papel. A partir de esto, sería necesario ver cómo se adapta a los usuarios tanto en las empresas como los clientes de estas empresas. En el caso de que su aceptación sea positiva, el desarrollo de la aplicación se podría ampliar incluyéndole la posibilidad de que los clientes una vez tengan su turno puedan realizar su pedido de manera no presencial a través de la app, agilizando así la cola y facilitando la solicitud de su encargo al cliente. Por otro lado, se puede ampliar añadiéndole estadísticas, estas estadísticas mostrarían aspectos como el tiempo medio de duración de un turno, el número de clientes que se tiene que atender en una cola, etc. Todo con el fin de dar a la empresa que tenga establecido está aplicación una mayor información de su negocio, para que estos realicen posteriormente cualquier modificación adaptándose a las necesidades de su negocio y de sus clientes.

Capítulo 6

Summary and Conclusions

In this Final Work the aim has been to develop and create a society oriented project through the use of different functionalities and a wide range of technologies, in every part of the project. Besides, this work gave me the opportunity to investigate deeply these tools and combine them for the development of the whole project.

From the beginning the topic of the project has been focused on a market analysis as well as a survey of the society. To achieve our target we developed the app functionalities, in each of its parts, as if we were the final user; in order to understand and fix every single problem may arise.

The project is developed in order to make possible the incorporation of the app to any establishment. So that each of them could create its accounts, manage its queues and let the client enter each queue through a mobile app or a paper ticket. The project is now at a stage in which it can be incorporated in mordern society, and then see how, it adapts to the user as well as the companies and the clients of these companies. In case of possitive response, the app development may be extended as to include statistics that show the average waiting time of a turn, the number of clients who have been attended in a single queue depending on the hours, etc. Everything in order to give the company that has incorporated the app more data about his business, provide future changes and generate a better organization of costumers flow. Moreover, the app could be extended to enable clients can do your orders through the mobile.

Capítulo 7

Presupuesto

Este proyecto ha sido desarrollado mediante el uso de herramientas gratuitas, por lo que el coste del proyecto se basa en el sueldo del desarrollador y en la línea de Internet requerida en una parte del proyecto.

A continuación se muestran los costes correspondientes a cada apartado en la ejecución del proyecto.

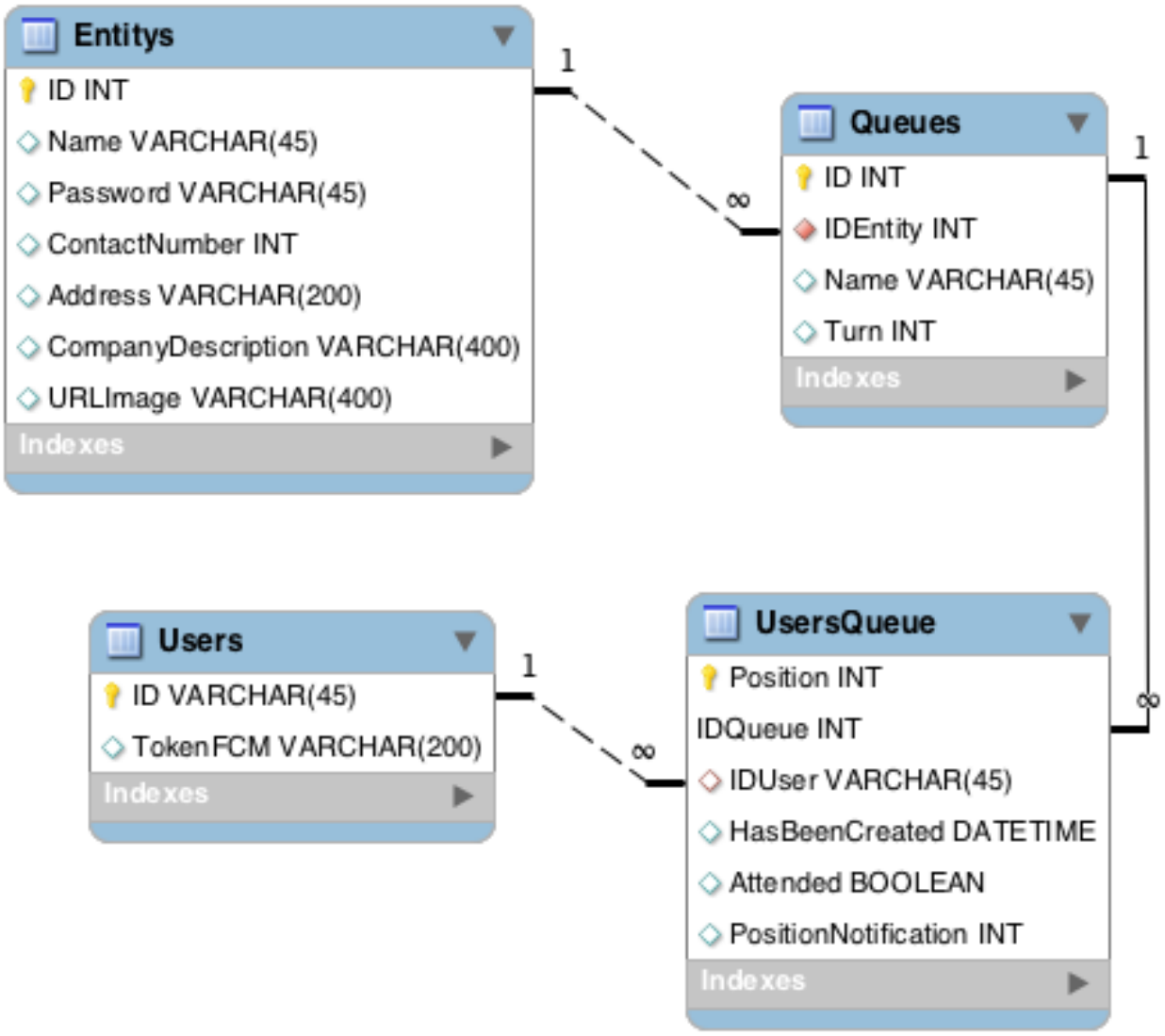
Objeto, función	**euro/hora**	**días**	**total**
Análisis del problema	60	10	2400
Análisis de las herramientas	60	10	2400
Desarrollo web	36	30	4320
Desarrollo móvil	36	40	5760
Total			14880

Tabla 7.1: Presupuesto aproximado

Apéndice A

Base de datos

A.1. Esquema



Bibliografía

[1] *Bootstrap 3 Tutorial*. http://www.w3schools.com/bootstrap/.

[2] *Configurar Firebase Cloud Messaging una app cliente en Android*. https://firebase.google.com/docs/cloud-messaging/android/client?hl=es

[3] *FPDF Library*. http://www.fpdf.org/.

[4] *MAMP*. https://www.mamp.info/en/.

[5] *Manual de PHP*. https://secure.php.net/manual/es/index.php.

[6] *PHP QR CODE*. http://phpqrcode.sourceforge.net/.

[7] Vikrant A Agaskar, Surjit H Singh, Srujan S Chaudhari, and Keyur P Rajyaguru. To automate entire placement and training cell for the college using android application with cloud computing.

[8] Ken Arnold, James Gosling, David Holmes, and David Holmes. *The Java programming language*, volume 2. Addison-wesley Reading, 2000.

[9] Kris M Bell, DarrylÑ Bleau, and Jeffrey T Davey. Push notification service, November 22 2011. US Patent 8,064,896.

[10] Ángel Cobo. *PHP y MySQL: Tecnología para el desarrollo de aplicaciones web*. Ediciones Díaz de Santos, 2005.

[11] Android Developers. What is android, 2011.

[12] Daniel Guzmán Reyes. Bases de datos distribuidas con una solución lamp (linux, apache, mysql y php). 2006.

[13] José Manuel Huidobro. Código qr. *Bit, dic.-ene*, 172:47–49, 2009.

[14] Douglas Lea. *Concurrent programming in Java: design principles and patterns*. Addison-Wesley Professional, 2000.

[15] AB MySQL. Mysql, 2001.

[16] AB MySQL. Mysql: the world's most popular open source database, 2005.

[17] Sylvain HÉBUTERNE-Sébastien PÉROCHON. *Android: Guía de desarrollo de aplicaciones para Smartphones y Tabletas (2a edición)*. Ediciones ENI, 2014.

[18] Juan Antonio Robledo Matas. Creació i integració d'una web per a la gestió de l'assignatura sim. 2008.

[19] Luke Welling and Laura Thomson. *Desarrollo web con PHP y MySQL*. 2005.