Máster universitario de Desarrollo de aplicaciones para dispositivos móviles

Trabajo Fin de Máster

Solución móvil para gestor de colas

Laura García González

Tutor/a
Francesc D'Assís Giralt Queralt

Girona, enero de 2018

Ficha del trabajo final

Título del trabajo:	Solución móvil para gestor de colas
Nombre del autor:	Laura García González
Nombre del consultor:	Francesc D'Assís Giralt Queralt
Entrega:	Enero 2018
Área del TFM:	Desarrollo de aplicaciones sobre dispositivos móviles

Resumen:

Este Trabajo Final de Máster consiste en el diseño e implementación de una aplicación móvil integrada en el gestor de colas `X` funcionando actualmente en instituciones o ayuntamientos. La aplicación permitirá al usuario que vaya a realizar un trámite administrativo, obtener la información y el estado de su cola en su smartphone y ser notificado de su turno.
Actualmente existen unas pantallas en las salas de espera que informan del estado de las diferentes colas, con esta aplicación se evitará que el usuario que vaya a realizar un trámite administrativo quede anclado en la sala de espera y así aprovechar su tiempo.
La aplicación estará disponible para todas las plataformas móviles.

Abstract:

This Final Master's Project consists of the design and implementation of a mobile application integrated in the queue manager `X` currently operating in institutions or town halls. The application will allow the user to go through an administrative process, obtain the information and the status of their queue on their smartphone and be notified of their turn.
Currently there are screens in the waiting rooms that report the status of the different queues, with this application will prevent the user who is going to perform an administrative process be anchored in the waiting room and thus take advantage of their free time.
The application will be available for all mobile platforms.

Palabras clave:

gestor de colas, optimizar, tiempo, multiplataforma, ionic, administración

Índice

Índice de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Internet ha causado junto con los avances en la telefonía móvil una repercusión considerable en nuestra vida y en cómo nos comunicamos. Solo dos avances tecnológicos han impactado a la sociedad, la invención de la imprenta y la revolución industrial en los siglos XVI y XVIII respectivamente, como lo ha hecho hoy en día el Internet. Lo que antes podían parecer dos realidades separadas, actualmente se complementan, y de hecho nos parecería extraño, incluso, pensar en un móvil sin conexión a Internet en cualquiera de sus formas.

Con el uso de Internet la demanda sobre los teléfonos móviles ha ido creciendo a un ritmo alarmante y los fabricantes, en su empeño por no quedarse atrás, han ido invirtiendo más esfuerzo y dinero a mejorarlos, hasta lo que hoy en día se conoce como *smartphones* (teléfono inteligente). De esta forma en ofrecerse un dispositivo con mejores prestaciones, surgen a la vez sistemas operativos más capaces y eficientes, y gradualmente aparecen multitud de aplicaciones que desean participar ofreciendo su granito de arena. Pensando en la movilidad que ofrece un dispositivo de este calibre y en la posibilidad de conectarlo a Internet, surgen ideas tan variadas como herramientas para la salud, mensajería que sustituye al tradicional mensaje de texto *sms*, aplicaciones de gestión, en resumen aplicaciones que ayudan a simplificar y mejorar la vida del usuario.

Una parte de este proyecto trabaja este último aspecto, ayudar a mejorar la vida del usuario. El tiempo de las personas es valioso y cada vez, con las exigencias del día a día, se dispone de menos y se ha de recortar para llegar a las 24 horas del día. El objetivo en consecuencia de este proyecto es crear una aplicación que ayude a recortar un trocito de este tiempo en la espera para hacer gestiones, pudiendo así aprovecharlo en otros asuntos.

En la empresa donde trabajo actualmente disponen de un gestor de colas funcionando desde hace 7 años en oficinas de atención ciudadana y ayuntamientos. El gestor de colas se llama `X` [**?**].

Sin entrar en muchos detalles de todas las funcionalidades de `X`, un usuario que se presenta en una oficina de atención a la ciudadanía integrada con el sistema `X` para realizar un trámite, debe realizar el ciclo que se muestra en la Figura 20 y que se enumera a continuación:

1. Aproximarse a la recepción o tótem (pantalla táctil con impresora) y recibir un tique con su turno
2. Mirar en la pantalla de información de su mesa el estado de la cola
3. Esperar indefinidamente su turno
4. Cuando es su turno, realizar el trámite

En este proyecto se quiere realizar una solución móvil para que el usuario evite las esperas indefinidas. Para conseguirlo, el usuario tendrá la información de su cola en su *smartphone* y será notificado mediante notificaciones, así no tendrá

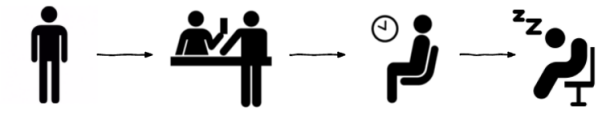


Figura 1: Ciclo que realiza el usuario al realizar un trámite con el gestor de colas `X`

que quedarse anclado en la sala de espera y podrá aprovechar el tiempo en otros quehaceres. En la Figura 2 muestra el nuevo ciclo de actividad del usuario.

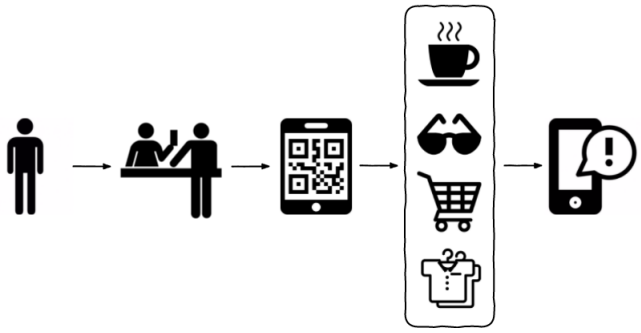


Figura 2: Ciclo que realizaría el usuario con una solución multiplataforma con el gestor de colas `X`

1.2. Análisis

1.2.1. Contexto de uso

Dado que el proyecto se basa en el desarrollo de una aplicación móvil para smartphones, es necesario tener en cuenta ciertas características sobre el contexto de uso de la aplicación. El contexto de uso se puede definir mediante tres parámetros básicos: entorno, situación y modo de uso.

El contexto de uso del producto está condicionado básicamente por la conectividad, si el usuario no tiene conexión a la red no podrá ser notificado. También presenta un entorno variable mayoritariamente público. El usuario no tendrá una gran interacción con la aplicación, la principal es la de registrar el tique.

1.2.2. Usuarios

El producto a estudiar, al estar sujeto al producto `X` (gestor de colas), es fácil saber quiénes serán sus usuarios finales. Se ha hablado con algunos usuarios actuales de `X` para investigar las necesidades de los usuarios finales, es decir, se ha hablado con los funcionarios que atienden a ciudadanos para encontrar un perfil general del ciudadano. Los principales usuarios de la aplicación serán personas que van regularmente a hacer trámites, como gestoras.

Para el uso de la aplicación móvil de presente proyecto, no se requiere que los usuarios posean un conocimiento experto en Internet, ni usuarios expertos en aplicaciones móviles, con una familiaridad básica con las aplicaciones móviles e Internet es suficiente.

Se identifica un único perfil: Cliente, que engloba a todo ciudadano que va a realizar un trámite.

1.2.3. Escenarios de uso

A continuación, se detallan una serie de posibles escenarios de uso de la aplicación:

Escenario 1

Nombre:	Pau Armengol
Lugar de residencia:	Olot
Edad:	40
S.O. de móvil:	Android
Ocupación:	Agricultor
Nivel de relación esperado con la app:	
Ocasionalmente para solicitar ayudas y subvenciones PAC	
Resumen:	
Pau trabaja duro en su pequeña finca de Olot, tiene que levantarse pronto para ordeñar las vacas y dar-las de comer. Cuidar de a su pequeña explotación ramadera y unos terrenos adyacentes no es todo su trabajo, además hace quesos y mató artesanales. Al mediodía tiene un descanso para comer y estar en casa con su familia, pero por la tarde tiene que volver a ordeñar las vacas.	
Escenario:	
Por su tamaño e interés para la comunidad la empresa de Pau recibe diversas subvenciones que le obligan a presentarse varias veces al año ante las administraciones pertinentes. Disponiendo únicamente del mediodía para hacer el tramite Pau tiene la suerte de disponer de la aplicación móvil `X`. Se presenta para hacer los trámites en la administración y en el caso de que la cola sea exageradamente larga puede aprovechar para hacer recados sabiendo que no perderá el turno.	

Escenario 2

Nombre:	Blanca Bello
Lugar de residencia:	Girona
Edad:	27
S.O. de móvil:	Android
Ocupación:	Abogada
Nivel de relación esperado con la app:	
Semanalmente para hacer tramites en nombre de sus clientes.	
Resumen:	
Blanca es abogada y trabaja en una gestora. Se dedica a realizar trámites varios para sus clientes. Durante la semana prepara la diferente documentación y una o dos veces por semana le toca presentarse personalmente para realizar los trámites.	
Escenario:	
Blanca tiene que personarse ante el Ayuntamiento de Girona. Hoy parece que esta muy frecuentado. Al coger el tique y escanear el código en el móvil ve que tiene muchos turnos por delante. La biblioteca esta al lado del ayuntamiento así que decide acabar de contestar correos desde la biblioteca hasta que recibe la notificación de su turno.	

1.2.4. Estudio del mercado

Se podría ver como principal competidor el tique convencional, pero el objetivo de la aplicación no es sustituir el tique convencional, si no convivir con él. Se han buscado productos actuales de aplicaciones o soluciones que eviten colas. Los principales vistos se citaran a continuación:

El producto SinCola [11], como `X`, también se basa en la gestión de colas y ofrece al cliente la notificación de su turno mediante mensajes de texto SMS. SinCola presenta su producto por diferentes categorias: banca y seguros, salud, restaurantes... No tiene aplicación móvil. Lo interesante de esta solución es poder enviar SMS, así no se condiciona la conexión Internet.

Figura 3: SMS que envía el producto SinCola

Otro producto encontrado es Orderella [8], este a diferencia del anterior sí que tiene una aplicación móvil. El eslogan que utilizan es *Let your phone do the queueing*, es decir, "deja que tu telefono haga la cola". Esta dirigida principalmente para la restauración, permite a los usuarios de la app pedir desde el smartphone y así evitar las colas en las barras o esperas en la mesa.

1.3. Objetivos del Trabajo

El principal objetivo de este trabajo es desarrollar el análisis, diseño e implementación de una aplicación multiplataforma que permita a un usuario del sistema de gestor de colas `X` estar informado y notificado del estado de la cola de espera a partir de su *smartphone*. Además de afianzar los conocimientos adquiridos en el máster y aprovechar esté TFM para adquirir y poner en práctica otros conocimientos que no se han tratado hasta ahora.

1.3.1. Requerimientos funcionales y no funcionales

Una vez el usuario ha ido a la institución y ha obtenido su tique es posible que, aunque haya pedido cita previa para una hora concreta, tenga que esperar considerando que quizás los responsables del servicio se han retrasado. Por eso el tique tendrá un código QR y un código alfanumérico. El usuario podrá introducir o escanear este código para ver el estado de su cola en su dispositivo móvil y ser notificado cuando tenga un usuario por delante suyo o ya sea su turno. Es en estos casos cuando la aplicación es útil, puesto que permite al usuario aprovechar su tiempo en otros quehaceres.

A continuación se identifican los servicios que proveerá el sistema, es decir, los requerimientos funcionales de la aplicación:

- Acceso a Internet
- Reconocimiento de el código QR del tique mediante la cámara del *smartphone*
- Reconocimiento de el código alfanumérico del tique introducido manualmente
- Visualización de la información de la cola una vez capturado el tique
- Notificaciones de los diferentes eventos de la cola

Los requerimientos no funcionales son los que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste. En la siguiente lista se citan los principales:

- Interfaz sencilla y intuitiva
- Accesible
- Ha de proporcionar un tiempo de respuesta rápido
- Deberá funcionar correctamente en dispositivos móviles con sistemas Android e iOS
- Fácil de descargar e instalar
- Smartphone con cámara (opcional)

1.4. Elección tecnológica

Una vez detectado el objetivo del proyecto, se baraja la opción de implementar la aplicación nativa para las principales plataformas del mercado, Android y iOS. A partir de un estudio de la aplicación se llega a la conclusión de que el producto a presentar no es complejo y que los principales requisitos de la aplicación (acceso a cámara, lector QR y notificación PUSH) también los ofrece IONIC framework. Finalmente, se opta por la implementación multiplataforma.

Por el lado del servidor, no hubo lugar a elección puesto que se trata de una solución que añade funcionalidad a un entorno de trabajo ya implantado. La única elección fue el uso de JSON para la comunicación de la aplicación con el servicio web, ante XML, los motivos fueron por experiencia personal y mayores ventajas.

1.5. Enfoque y método elegido

En este proyecto se desarrollará un producto nuevo, ya que está hecho a medida para que se pueda integrar en el sistema `X`. El producto se realizará con el framework Ionic puesto que con él se puede conseguir que el sistema sea multiplataforma, además se tiene un conocimiento básico al trabajarlo en el máster.

La aplicación móvil se comunicará a partir de un servicio web para la obtención de los datos. Para implementar esta capa el servicio web se utilizará:

- IIS como a servidor de aplicaciones
- El lenguaje C# de la plataforma .NET
- Módulo WCF para crear y consumir servicios web

1.6. Planificación

El trabajo se inicia a mediados de setiembre del 2017 y tiene como previsión de entrega la primera semana de enero del 2018. En este período transcurren 15 semanas, programando una dedicación de 20 horas semanales da como resultado un total de 300 horas. Las horas dedicadas a este trabajo se repartirán en diferentes fases. Considerando que cada fase del trabajo coincide con las PAC planificadas por la asignatura, la distribución de las horas es la siguiente:

Nombre	Inicio	Fin	Horas
PEC 1	15/09/2017	15/10/2017	**30**
Elección de proyecto	15/09/2017	01/10/2017	10
Elaboración del plan de trabajo	01/10/2017	15/10/2017	20
PEC 2	15/10/2017	01/11/2017	**50**
Análisis y arquitectura	15/10/2017	21/10/2017	20
Revisión del plan de trabajo	21/10/2017	29/10/2017	20
Preparación del entorno	29/10/2017	01/11/2017	10
PEC 3	01/11/2017	15/12/2017	**160**
Implementación	01/11/2017	30/11/2017	100
Pruebas	30/11/2017	10/12/2017	40
Memoria	10/12/2017	15/12/2017	20
PEC 4	15/12/2017	03/01/2018	**60**
Últimos retoques de implementación	15/12/2017	20/12/2017	20
Memoria	20/12/2017	03/01/2017	20
Presentación	20/12/2017	03/01/2017	20

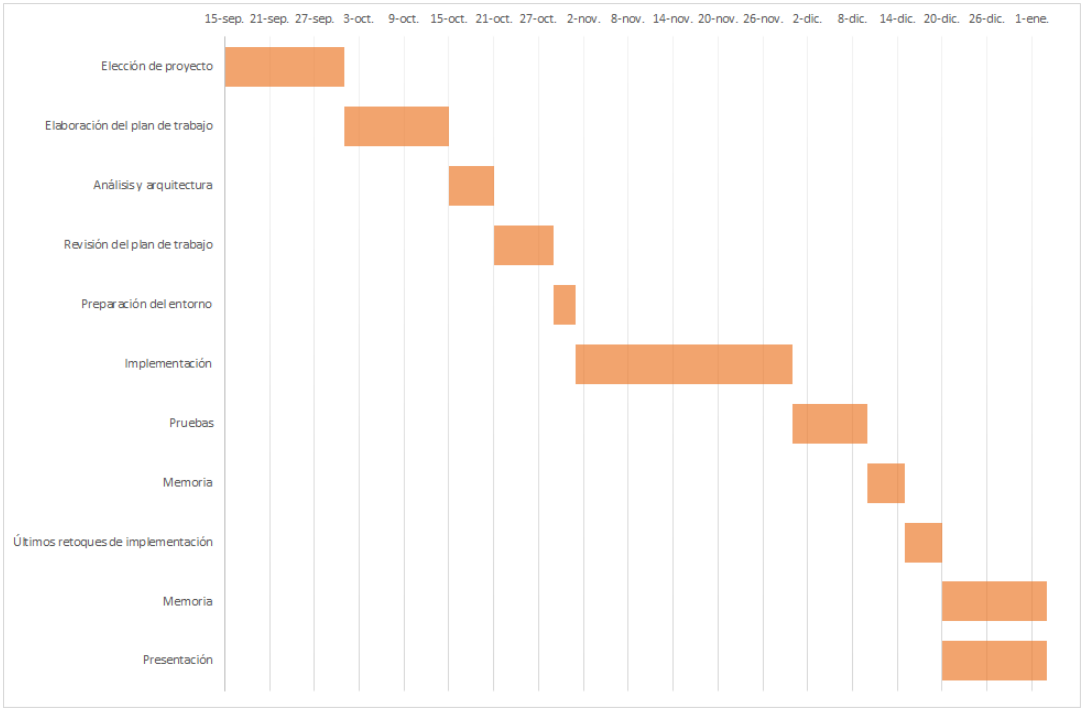


Figura 4: Diagrama de Gantt

1.7. Metodología

La metodología que se ha seguido para desarrollar este proyecto a estado el **modelo en cascada con retroalimentación** [13].

El modelo tradicional también llamado modelo en cascada es un enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal manera que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. De la misma forma que una cascada, estas fases están dispuestas de tal manera que avanzar en el desarrollo es rápido y fácil, mientras que en el caso de que se necesitara corregir algo y " volver atrás " sería muy costoso. En esta analogía, a medida que se va avanzando cada vez más, volver a las primeras etapas supone un coste que se incrementa casi de forma exponencial, ya que por ejemplo, modificar el análisis implicaría seguramente modificar el diseño y el resto de tareas que lo convierten.

Para evitar costes innecesarios se realizará una revisión después de finalizar cada etapa con el fin de comprobar y confirmar que se puede pasar a la siguiente fase. Se ha elegido este modelo porque es adecuado para proyectos con un objetivo claro y donde se conocen los detalles de cómo será la solución.

Aun siendo un proyecto pequeño, incluso el software más nimio no se libra de estar sujeto a fallos o variaciones, y presumir que todo el proceso seguirá la plena rigidez y no volatilidad de los requisitos y que cada etapa estará exenta de errores, es quizás demasiado optimista. Es por esta razón por la que el modelo en cascada " puro " rara vez se usa, optando por alguna de sus variantes.

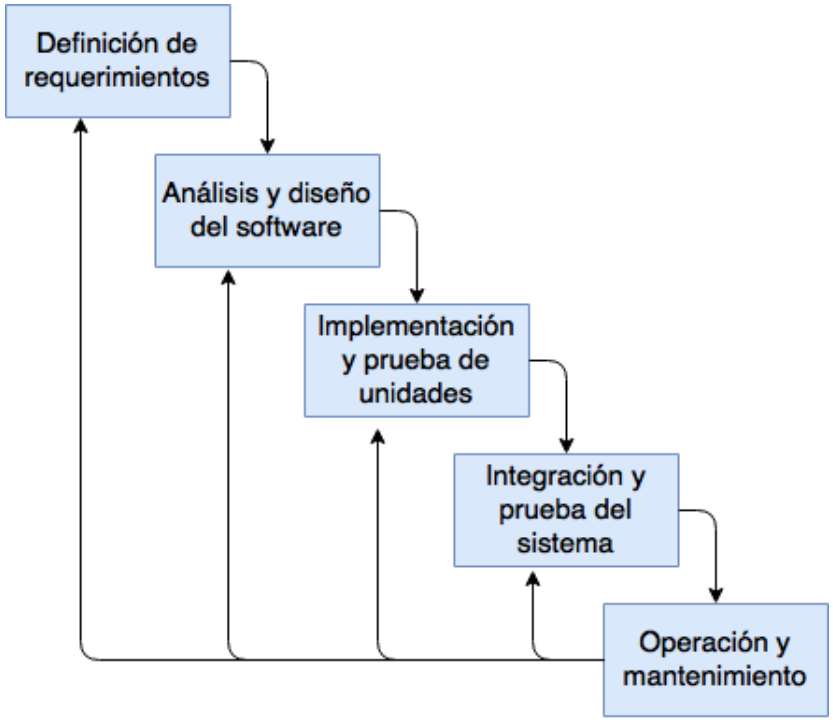


Figura 5: Modelo en cascada con retroalimentación

Es lógico pensar que podría ser interesante introducir retroalimentación entre

ciertas etapas para poder, por ejemplo, modificar o añadir requisitos aunque se haya entrado en la fase de diseño. De esta forma se permite retroceder a etapas anteriores si es necesario.

Para un proyecto como el que se presenta en este TFM, resulta casi ideal este tipo de modelo ya que se trata de un proyecto de alta rigidez, con requisitos correctamente especificados y muy claros.

En la Figura 5 se muestra un esquema de las diferentes etapas que se explican a continuación:

1. **Análisis de requisitos**
   Con lo que pide el cliente y con las necesidades de los usuarios finales se determinan cuáles son los objetivos a cumplir y especifican plasmados en forma de requisitos.
2. **Disseño del sistema**
   Es en esta fase donde se realiza la transición entre el especificado en el análisis (requisitos) y el diseño del sistema (diseño de alto nivel o diseño de la arquitectura, y diseño detallado).
   El diseño de la arquitectura se centra en definir la estructura de la solución, identificar los módulos y relacionarlos. El diseño detallado, por su parte, es más bajo nivel (algoritmos, definir la organización del código, etc.).
3. **Implementación**
   Es en esta fase donde se realiza la transición entre el especificado en el análisis (requisitos) y el diseño del sistema (diseño de alto nivel o diseño de la arquitectura, y diseño detallado).
   El diseño de la arquitectura se centra en definir la estructura de la solución, identificar los módulos y relacionarlos. El diseño detallado, por su parte, es más bajo nivel (algoritmos, definir la organización del código, etc.).
4. **Pruebas**
   Después de la implementación, se realizarían pruebas unitarias y se integrarán todas las unidades probándose en conjunto, y verificando como es obvio que cumpla con la demanda del cliente.
5. **Mantenimiento**
   Una vez ya se ha entregado el producto final al cliente, se empieza esta última fase que suele ser generalmente más larga (se acuerda con el cliente), en la que se realizan correcciones de errores descubiertos, posibles mejoras sobre la implementación, nuevos requisitos, etc.

1.8. Breve sumario de productos obtenidos

Para poder obtener los objetivos de este trabajo se harán los siguientes entregables:

- La memoria del proyecto.
- La aplicación multiplataforma implementada con el framework Ionic.
- El servicio web que se comunicará para la obtención de datos y el envío de notificaciones *push*.
- Un vídeo presentando el funcionamiento del producto.

2. Marco de trabajo y conceptos previos

En esta sección se especifican algunos detalles y matices que ayudaran a enfocar el marco de trabajo y la temática general que engloba este Proyecto final de Máster.

Como ya se ha comentado, este proyecto está formado por una aplicación multiplataforma y un servicio web que conecta con el producto `X`. El desarrollo se realizará sobre un MacBook Pro cuyas características se describen en la Figura 6. Las pruebas en terminales físicos se realizaran sobre un Motorola G5 y un iPhone 6. Por lo tanto las herramientas base son:

- MacBook Pro
- Motorola G5 con sistema operativo Android 7.0
- iPhone 6 versión 10.2

Figura 6: Características del MacBook Pro para desarrollo

A continuación se explica las herramientas que se utilizaran para desarrollar la aplicación móvil y se citará brevemente como será el servicio web y su comunicación con el mismo.

2.1. Entorno de implementación para aplicación multiplataforma

Para crear la aplicación multiplataforma se usará el Ionic Framework basado en Cordova. Se usará la versión de Ionic 3.0.

El IDE utilizado para la realización de la aplicación es WebStorm de JetBrains. Se ha considerado también el Visual Studio Code aunque este se caracteriza por

ser un editor y no un IDE. Visual Studio Code es open source y su uso es gratuito, al contrario de WebStorm. Se ha optado por WebStorm por familiaridad de uso, concede más funcionalidades y por disponer de una licencia de uso concedido por la UOC.

Para la elaboración del prototipado se han utilizado dos herramientas:

- NinjaMock [7]
  Herramienta online para prototipar que sirve tanto para diseño de páginas web como para aplicaciones Android, iOS o Windows Phone
- Ionic Creator [6]
  Software en linea para prototipar y que luego permite exportar el proyecto a código fuente.

2.2. Servicio web

En este proyecto no se centra en detalles del servicio web, por lo tanto sólo se nombrará la tecnología usada y en la Subsección 3.5 Servicio web se explicará la parte del servidor que incumbe a la aplicación, es decir, la comunicación.

La tecnología usada para el servicio web es WCF con arquitectura REST.

2.3. Memoria

A continuación se enumeraran las herramientas usadas para la elaboración de esta memoria:

- Texmaker [12]
  Texmaker es un editor gratuito distribuido bajo la licencia GPL para escribir documentos de texto, multiplataforma, que integra muchas herramientas necesarias para desarrollar documentos con LaTeX, en una sola aplicación
- draw.io [5]
  Es un software de diagrama en línea gratuito para hacer diagramas de flujo, diagramas de proceso, organigramas, UML, ER y diagramas de red.

3. Diseño conceptual

Dentro del diseño conceptual existen los casos de uso. Los casos de uso son una herramienta destinada al análisis de un proyecto software. Más concretamente, los casos de uso son una técnica de escenarios incorporada en UML que describe la interacción entre actores y sistemas.

Cada uno de los casos de uso, especifica una serie de precondiciones que deben cumplirse, una serie de postcondiciones producidas después de la ejecución del mismo, un escenario principal y un escenario alternativo. El escenario principal trata la secuencia común de interacciones, mientras que el alternativo describe la ejecución en caso de error o en caso de caminos de decisiones diferentes a la habitual.

3.1. Casos de uso

En este apartado se muestran los casos de uso detectados en la aplicación, como se puede comprobar en el diagrama de la Figura 7. El producto solo tiene un actor, el usuario, y dos casos de uso destacables.

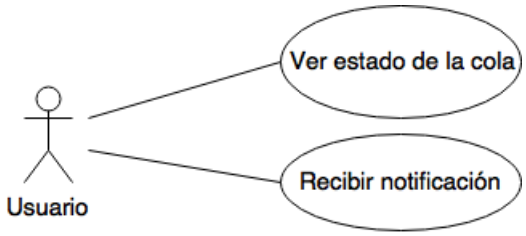


Figura 7: Casos de uso

3.2. Fichas de los casos de uso

En las siguiente fichas se muestran los casos de uso que se han comentado y que aparecen en la Figura 7.

Caso de uso:	Ver el estado de la cola
Descripción:	Muestra el estado de la cola en el que el usuario está asignado
Actores:	Usuario
Pre-condición:	El usuario tiene la aplicación en el dispositivo
Flujo principal:	1. El cliente entra el identificador del turno 2. El sistema muestra el estado de la cola
Flujo alternativo:	Si el identificador es incorrecto 2. Muestra un mensaje de error
Post-condición:	Se muestra el estado de la cola

Caso de uso:	Recibir notificación
Descripción:	El usuario recibe una notificación en su smartphone alertándolo de su turno
Actores:	Usuario
Pre-condición:	El usuario tiene la aplicación en el dispositivo y ha registrado su turno
Flujo principal:	1. El usuario recibe una notificación de su turno
Flujo alternativo:	Si el usuario no tiene conexión a Internet 2. Muestra un mensaje de alerta para que busque cobertura
Post-condición:	Se muestra el estado de la cola

El objetivo de este apartado es la de modelar el sistema basándose en la salida de la fase anterior (análisis), para que posteriormente el personal encargado de programación, pueda implementar el proyecto. La etapa de diseño permite también tener una idea más clara de cómo será el sistema, ya que saca parte de abstracción de la etapa anterior.

3.3. Diagrama de flujo

El diagrama de flujo sirve para representar de forma gráfica el proceso del sistema, es decir, mediante símbolos con significados definidos e interconectados con flechas se marcan los pasos del sistema. Este diagrama se caracteriza por tener un único punto de inicio y un único punto de finalización.

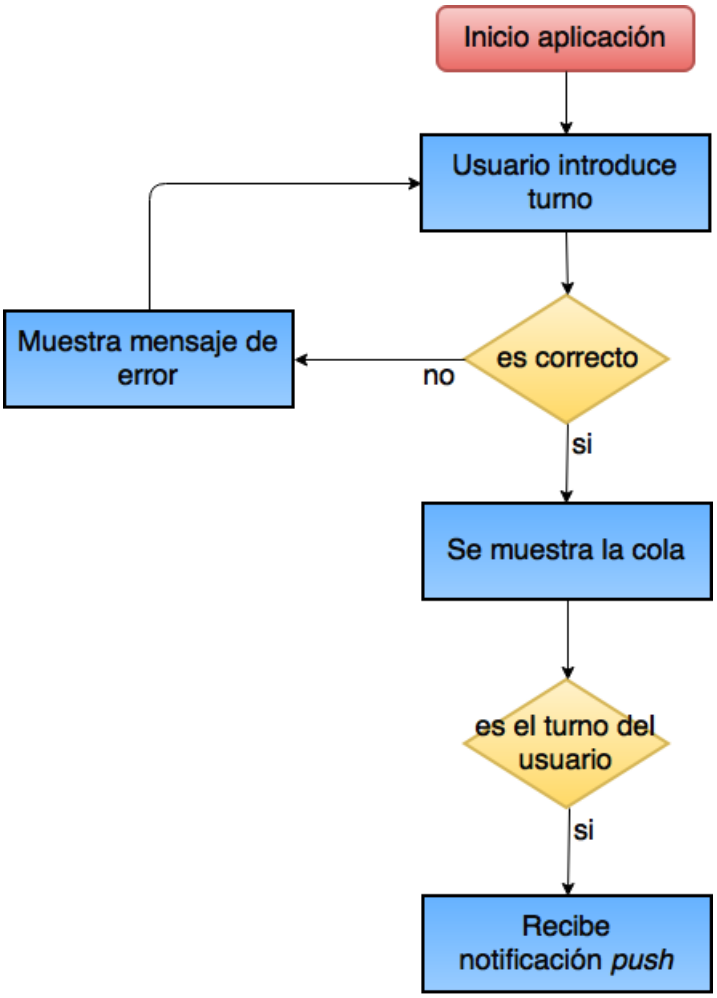


Figura 8: Diagrama de flujo

En la Figura 8 se observa el diagrama de flujo del funcionamiento general de la aplicación multiplataforma. Se inicia la aplicación y se pide registrar el código (de manera manual o escaneando el código QR), el usuario introduce el código, se comprueba que el código es correcto y devuelve el estado de la cola donde está asignado. El servidor envía al dispositivo una notificación cuando sea su turno.

3.4. Arquitectura MVC

Como se ha comentado anteriormente la aplicación será multiplataforma. Se implementará en Ionic, por lo tanto seguiremos su arquitectura.

Ionic, al estar basado en Angular, utiliza el patrón conocido como Vista-Controlador (View-Controller) que fue popularizado por frameworks como Cocoa Touch. En este tipo de patrón las diferentes secciones de la interfaz se pueden dividir en distintas vistas hijas o incluso podrían ser vistas hijas que contengan a su vez otras vistas hijas. Los controladores están asociados a estas vistas y se encargan de proporcionar los datos necesarios y la funcionalidad de los diferentes elementos. En la Figura 9 podemos ver un esquema de como funciona el patrón de arquitectura MVC.

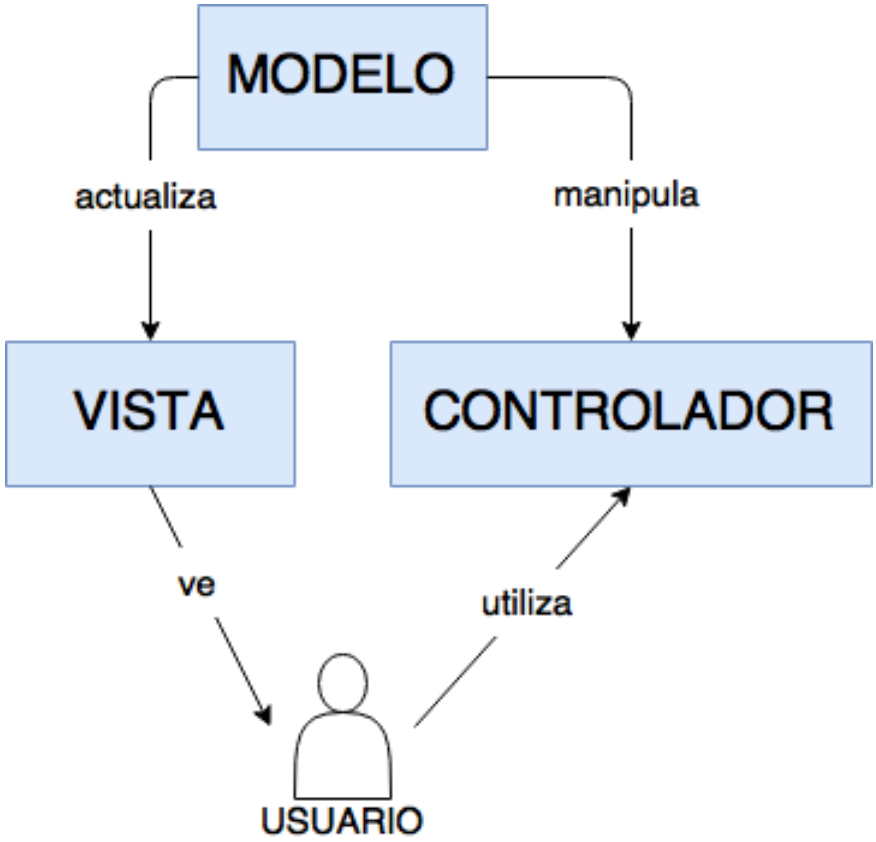


Figura 9: Arquitectura MVC

Nuestra aplicación se basará en tres controladores principales:

1. **Home**
   La aplicación se inicia con este controlador. Presenta al usuario las dos opciones para registrar su tique: introduciendo el código manualmente o escaneando el código QR.
2. **Scanner**
   Este controlador es el encargado de iniciar la cámara con el lector de QR.
3. **Queue**

Si el usuario a registrado correctamente su tique podrá ver el estado de la cola y este controlador será el encargado de mostrarlo.

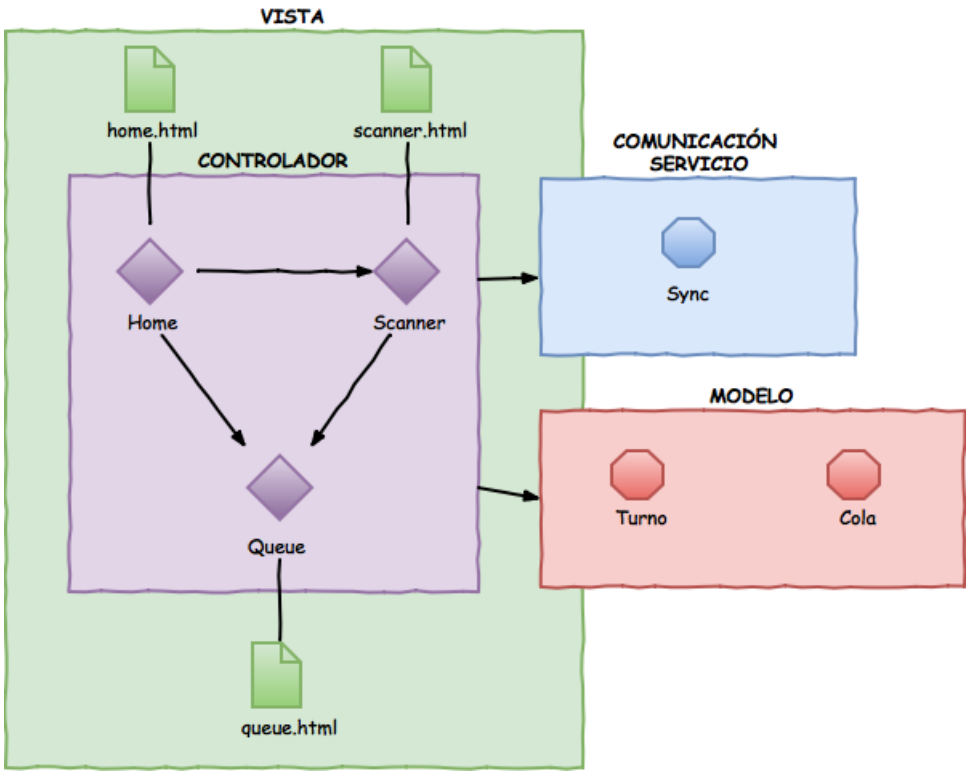


Figura 10: Diagrama del proyecto

Cada uno de los controlador tiene su respectiva vista. Además para obtener y analizar la comunicación con el servicio web, los tres controladores se comunicarán con la clase Sync. En la Figura 10 queda esquematizado en un diagrama la organización del proyecto en Ionic.

3.5. Servicio web

El servicio web será el responsable de la comunicación entre el producto `X` y nuestra aplicación, la Figura 11 muestra el esquema de comunicación.

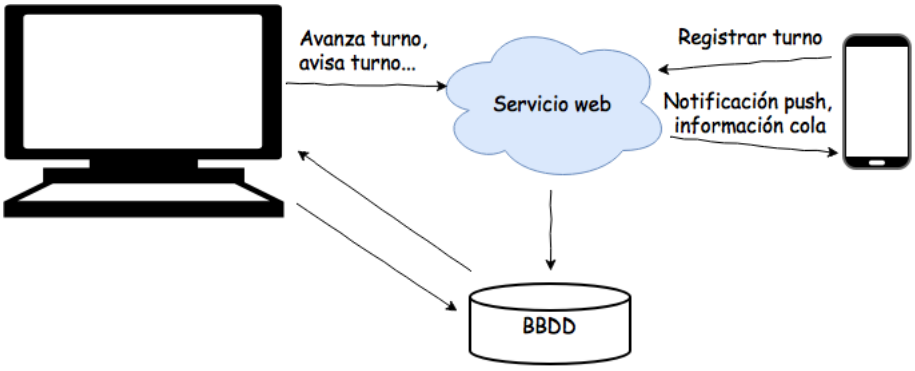


Figura 11: Esquema de comunicación del servicio web

Sin entrar en detalle, `X` es un gestor de colas compuesto por programas de escritorio y web. El que se comunica con nuestro servicio web es el de Atención al Usuario, es decir, sería el empleado que atiende a los trámites de los ciudadanos. La pantalla principal del Atención al Usuario se puede ver en la imagen de la Figura 12. Los dos botones subrayados son los que se comunicarían con el servicio web para notificar al usuario, si fuera el caso.

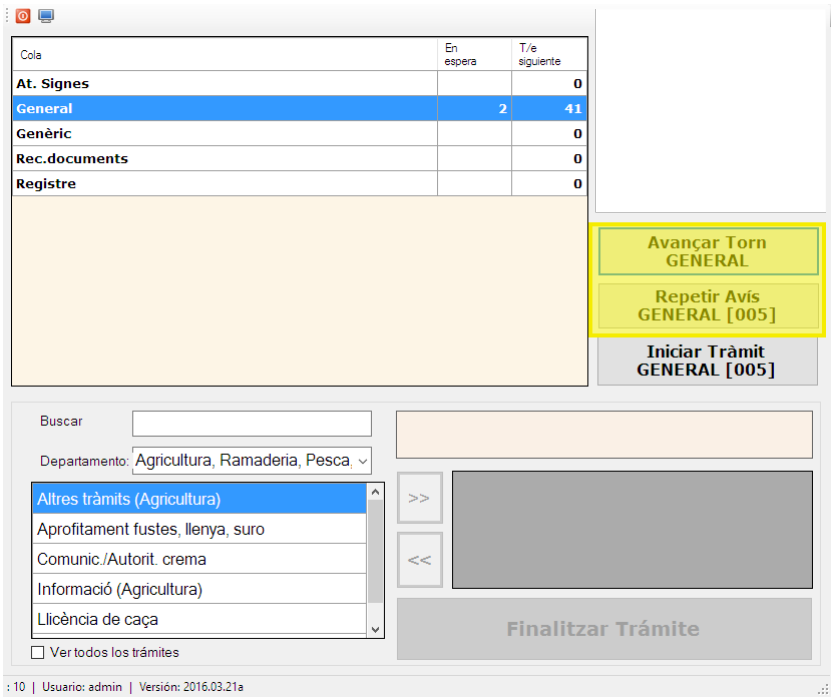


Figura 12: Pantalla principal del programa Atención al Usuario

3.5.1. Definición del servicio web

En esta sección se definen las llamadas del servicio web. Se distinguen dos tipos de llamadas. Las que hace la aplicación al servicio web y las que hace la aplicación `X` al servicio web y acaban repercutiendo en notificación a la aplicación.

Comunicación aplicación a servicio web:

- `RegistroFCM(idTurno,token)`: con el identificador del turno entrado manualmente por el usuario y el token FCM para notificaciones PUSH, el usuario registra su turno y así poder recibir los diferentes eventos de la cola en el móvil.

Comunicación `X` a servicio web a aplicación:

- `AvisarTurno(idTurno, mensaje)`: envía una notificación push al turno con un mensaje específico.
- `EstadoCola(idTurno)`: envía el estado de la cola con la lista de turnos en formato JSON.

3.6. Modelo de datos

El primero y más importante componente dentro de la arquitectura MVC utilizada en la capa del servidor es el modelo de datos. En este componente se le establecen las entidades necesarias para que el sistema funcione correctamente en toda su totalidad. Al hablar de entidad, se refiere a una unidad mayor que comprende una tabla en la base de datos y una relación con las demás.

Una sobrecarga de entidades o una incorrecta distribución de relaciones entre ellas en un modelo de datos puede provocar que el sistema se vuelva lento e inestable cuando el volumen de consultas y escrituras en la base de datos sea muy elevado. Es necesario que este sea el más óptimo posible, de manera que el acceso a un registro y a sus relaciones no suponga recorrer demasiadas tablas.

`X` ya dispone de una base de datos, para la realización de este proyecto se ha tenido que modificar la para añadir la nueva funcionalidad. A continuación se lista las nuevas entidades:

- TurnoPush: tabla relacionada con la tabla Turno. Principalmente contiene la información del dispositivo al que hay que enviar las notificaciones Push, entre otros.

4. Prototipado

4.1. Diseño de la interfície de usuario

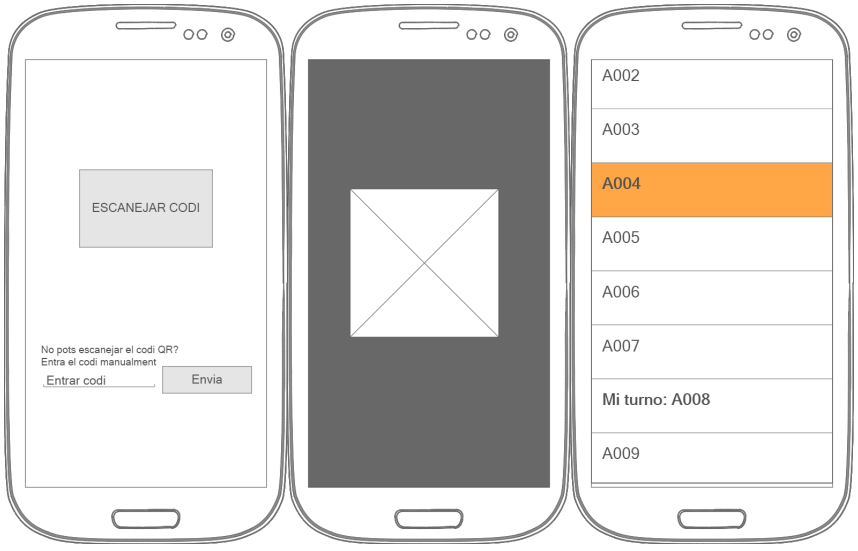
En esta sección se explica con detalle la interfaz funcional de usuario para el producto. El aspecto gráfico final puede variar de este borrador inicial.

La pantalla inicial se buscaba que fuera sencilla e intuitiva. Por lo tanto, se dejan claro cuáles son los dos métodos de introducir el código del turno: escaneando el código QR con la cámara o introduciendo manualmente el código impreso en el turno. Así que el primer prototipo diseñado se muestra en la Figura 13.

El escaneo del código no tiene ningún misterio, el diseño ocupa la imagen de la cámara en toda la pantalla en un color más oscuro y en medio un cuadrado donde se debe enfocar al código.

Una vez introducido el código (tanto manualmente como escaneando ello) se accede a una pantalla que muestra el orden de la cola. La celda subrayada indica el turno actual. El turno del usuario se distingue de los demás por el hecho que pone: *Mi Turno: XXXX*.

Hay que decir que no es necesario que el usuario esté registrado para la utilización de esta aplicación.



(a) Pàgina principal (b) Captura del codi QR (c) Informació de la cua

Figura 13: Diseño en alto nivel de las diferentes pantallas de la aplicación

Ionic ofrece la herramienta Ionic Creator para crear prototipos. Desde este enlace [9] se puede acceder al prototipo.

Para la elección de paleta de colores y diseño en general se usará de referencia los productos ya existentes de `X`, como por ejemplo el de la imagen 14. Como resultado obtenemos la *Moodboard* de la figura 15.

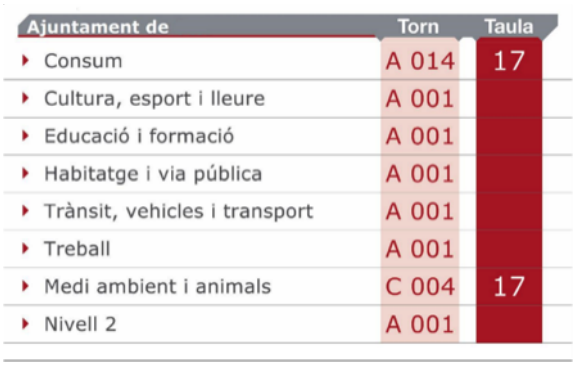


Figura 14: Pantalla con información de colas de `X` para las colas

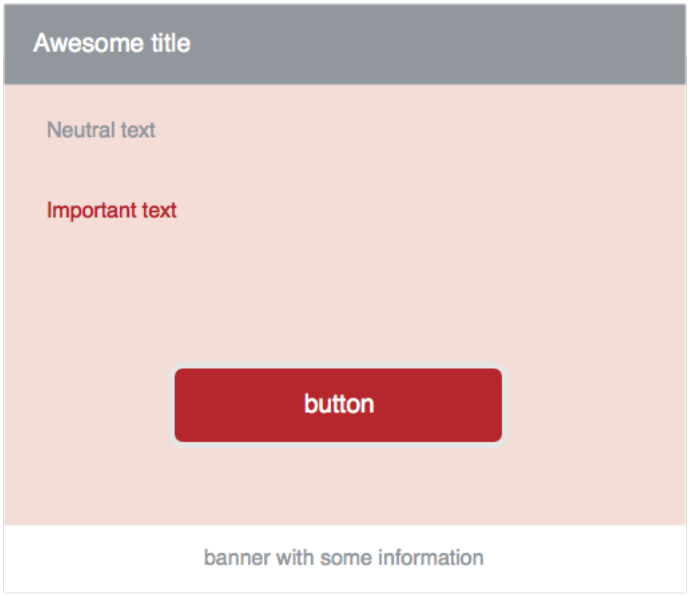


Figura 15: Moodboard con color y tipografía

5. Implementación

La fase de implementación ha consistido en la construcción de la aplicación móvil y su integración con el producto `X`.

Desde el punto de vista tecnológico, el proyecto se ha dividido en dos partes, por un lado, el desarrollo del servicio web, que proporciona una API Rest, y el desarrollo de la aplicación multiplataforma.

5.1. Aplicación móvil

Las características técnicas de la aplicación móvil son las siguientes:

- **Lenguaje de programación:**
  El proyecto está basado en framework IONIC 3.0, por lo tanto el lenguaje de programación es TypeScript.

- **Entorno de desarollo**
  El entorno de desarrollo es WebStorm, también comentado en el apartado 2.1. Es necesaria la instalación de IONIC. En el siguiente punto 5.2.1 se explican los pasos seguidos para la configuración de IONIC.
- **Librerías requeridas**
  - QRScanner. Plataformas soportadas: Android, iOS, Browser, Windows
  - Camera. Plataformas soportadas: Android, BlackBerry 10, Browser, Firefox OS, iOS, Ubuntu, Windows, Windows Phone 8
  - FCM. Plataformas soportadas: Android, iOS
- **Entorno de ejecución y pruebas**
  Se han realizado las pruebas funcionales mediante dispositivos físicos Android y iOS (características en el punto 2).

5.2. Preparación del entorno

Para iniciar el proyecto es necesario preparar el entorno previamente. En nuestro caso ya tenemos instalados tanto el IDE WebStorm (desde la misma página web [4]) como el Ionic Framework.

5.2.1. Configuración Ionic

Para la instalación de Ionic se hace desde el mismo terminal de WebStorm con el siguiente comando:

```
$ npm install -g cordova ionic
```

Creamos el proyecto:

```
$ ionic start proyecto blank
```

Esto creará una carpeta llamada proyecto en el directorio donde se ejecutó el comando. Luego, tenemos que decir a Ionic que queremos habilitar las plataformas iOS y Android.

```
$ ionic cordova platform add ios
$ ionic cordova platform add android
```

Instalamos los plugins necesarios para satisfacer los requerimientos de la aplicación, en nuestro caso: la cámara, el escaner QR, notificaciones FCM y almacenamiento local:

```
$ ionic cordova plugin add cordova-plugin-qrscanner
$ npm install --save @ionic-native/qr-scanner

$ ionic cordova plugin add cordova-plugin-camera
$ npm install --save @ionic-native/camera

$ ionic cordova plugin add cordova-plugin-fcm
$ npm install --save @ionic-native/fcm

$ ionic cordova plugin add cordova-sqlite-storage
$ npm install --save @ionic/storage
```

Para probar el código en un dispositivo físico, tanto iOS como Android solo se tiene que conectar al ordenador con un USB y ejecutar el siguiente comando:

```
$ ionic cordova run ios
$ ionic cordova run android
```

5.2.2. Configuración Firebase

Para la configuración de las notificaciones FCM hay que acceder a la consola en linea de *Firebase* [1] y crear un proyecto nuevo. Una vez creado el proyecto hay que añadir una nueva aplicación para cada plataforma, Android y iOS. Para cada aplicación hay que introducir el identificador de la aplicación, este identificador lo podemos obtener del archivo `config.xml` dentro del proyecto Ionic, en la etiqueta `widget` y atributo `id`.

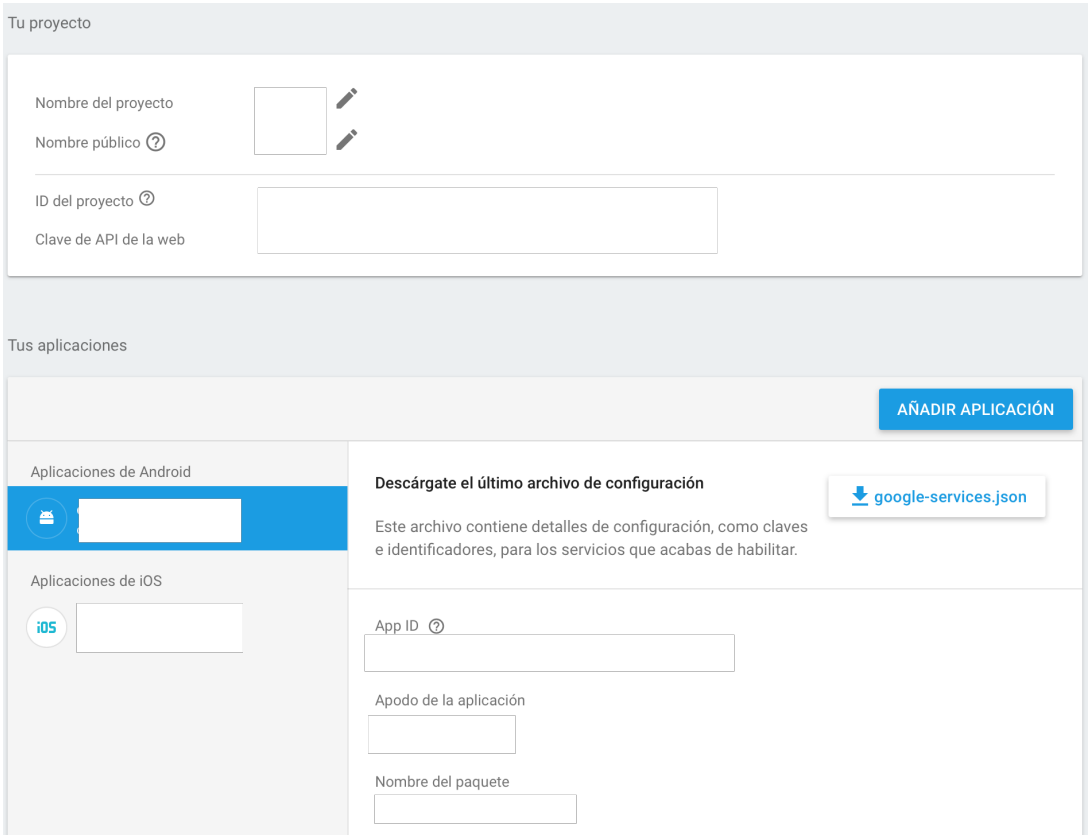


Figura 16: Consola firebase con las aplicaciones creadas para el proyecto

Al crear la aplicación Android y iOS en Firebase se crea el archivo de configuración `GoogleService-Info.plist` y `gogle-services.json` que hay que añadir en la raíz del proyecto Ionic.

5.3. Estructura de la aplicación

Dentro de la carpeta de nuestro proyecto, se encuentra una estructura de proyecto típica de Cordova donde se pueden instalar *plugins* nativos y crear archivos específicos de plataforma. A continuación se citaran las principales carpetas y archivos del proyecto:

.\node_modules La carpeta `node_modules` se genera automáticamente al instalar las dependencias npm con "npm install". Este comando explora el archivo

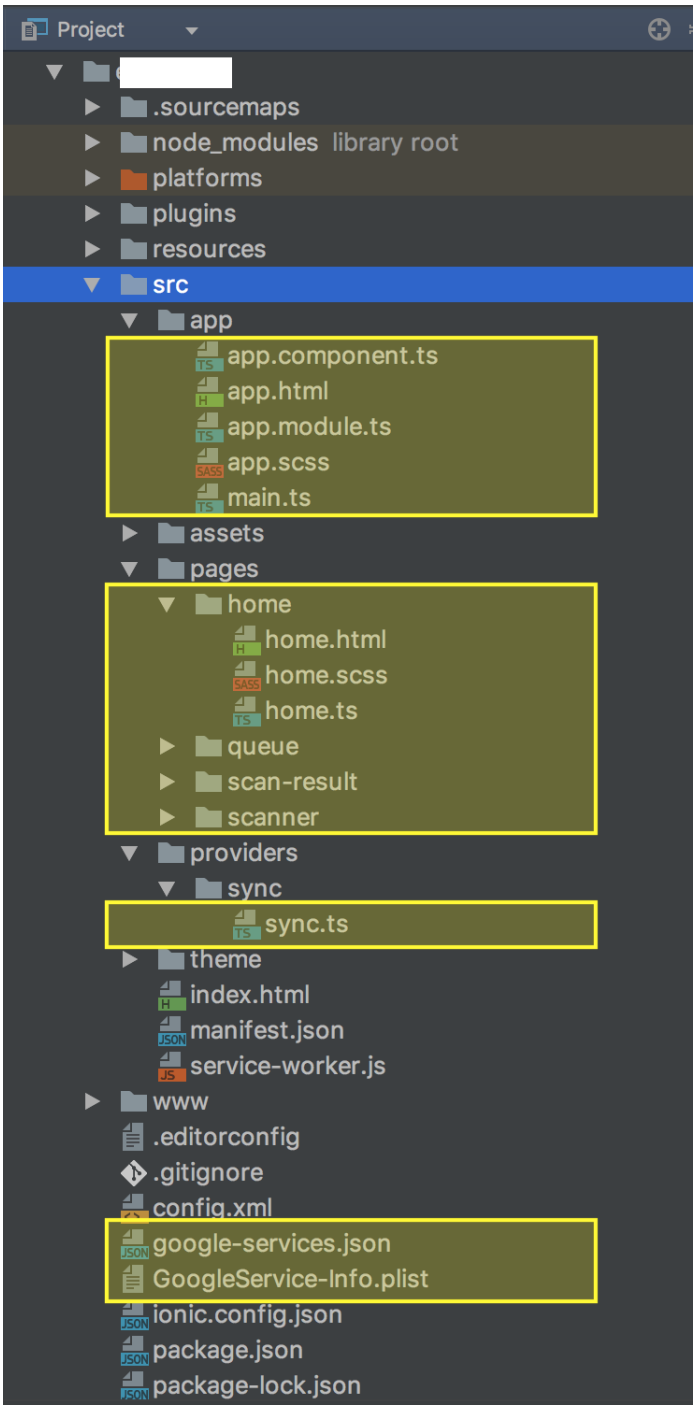


Figura 17: Estructura de la aplicación

`package.json` para todos los paquetes que necesitan ser instalados.

.\platforms En esta carpeta se generará los proyectos nativos para cada plataforma que se haya añadido previamente. En este caso, al añadir la plataforma IOS y Android se creará una carpeta llamada `ios` y otra llamada `android` y dentro tendrán los archivos y carpetas con la estructura de un proyecto nativo.

.\plugins Contiene los *plugins* de Cordova que se hayan instalado. Se crean automáticamente en esta carpeta al instalar un *plugin*. No se debe modificar nada de esta carpeta.

.\resources Contiene el icono y la pantalla de presentación de la aplicación con la que después se puede crear automáticamente todas las imágenes en todos los tamaños necesarios para cada plataforma, lo que ahorra mucho tiempo al no tener que generar a mano todos los tamaños de imagen necesarios del icono y la pantalla de presentación.

.\src Esta es la carpeta más importante y donde se realiza la mayor parte del trabajo. Aquí es donde están los archivos con el contenido de la aplicación, donde se definen las pantallas, el estilo y el comportamiento que tendrá la aplicación.

.\www Esta carpeta se genera automáticamente y contiene la versión actual del código cada vez que efectuamos un cambio. No se debe cambiar nada aquí ya que todo lo que se cambie en esta carpeta se machacará con cada cambio que se realice en la carpeta `src`.

.\editorconfig y .\gitignore Son dos archivos ocultos, están relacionados con la configuración del editor de código y Git.

.\config.xml El archivo `config.xml` contiene parámetros que se utilizan cuando se construye un proyecto nativo a partir de un proyecto ionic. Aquí deberemos indicar los permisos especiales que necesite la aplicación y otras configuraciones que puedan ser necesarias.

.\ionic.config.json Contiene información básica sobre la configuración del proyecto, se utiliza para subir la aplicación a la plataforma Ionic.io.

.\package.json Contiene paquetes y dependencias.

.\tsconfig.json y .\tslint.json Son archivos que contienen información necesaria a la hora de compilar TypeScript.

5.3.1. Carpeta de código fuente "src"

El directorio `src` contiene la carpeta `pages`, en está carpeta es donde se van a alojar todas la páginas que contenga la aplicación, *grosso modo*, una página será como una vista o un pantalla de nuestra aplicación.

Una página esta formada por tres archivos:

- Un archivo `.html`: contiene la plantilla html de la página
- Un archivo `.scss`: contiene el archivo sass donde podremos modificar el estilo de los componentes de la página
- Un archivo `.ts`: es el archivo TypeScript que contiene el controlador de la página, donde se define el comportamiento de la misma, como por ejemplo la función con la lógica a ejecutarse cuando se pulse sobre un botón de la página.

La aplicación esta compuesta por tres páginas:

- HomePage: página principal, permite acceder a la página de captura de código QR o introducir manualmente el código de texto
- ScannerPage: página para capturar el código QR
- QueuePage: página con la información de la cola

Para añadir una página al proyecto IONIC se hace a partir del terminal. A continuación se muestra los tres comandos para crear las páginas del proyecto:

```
$ ionic generate page Home
$ ionic generate page Scanner
$ ionic generate page Queue
```

La carpeta `providers` contiene los servicios encargados de manipular los datos como conexiones con API Rest, LocalStorage, SQLite... En el proyecto solo existe un servicio que se comunica con el servicio web: `Sync`. Los `providers` también se crean a partir de un comando:

```
$ ionic generate provider Sync
```

5.4. Pruebas funcionales

Se han realizado pruebas de integración sobre el dispositivo físico Android. Con un resultado satisfactorio de las pruebas se puede considerar que el resultado es un producto capaz de brindar unas funciones y una calidad mínima exigible.

5.4.1. Definición de casos de prueba

ID	Nombre	Pasos del caso de prueba
1.1	Registro tique manual	1. En la pantalla inicio, introducir código en el formulario de introducción manual 2. Pulsar el botón Registrar 3. Ver la pantalla con el estado de la cola **Caminos alternativos:** ▪ Introducir código con formato incorrecto ▪ Introducir código inexistente ▪ Introducir código ya registrado, token FCM diferente ▪ Introducir código caducado
1.2	Registro tique QR	1. En la pantalla inicio, pulsar el botón Escanear Tique 2. Centrar el código QR en la cámara 3. Ver la pantalla con el estado de la cola **Caminos alternativos:** ▪ No se reconoce el código ▪ Código ya registrado, token FCM diferente ▪ Código caducado
2	Desregistrar tique	1. En la pantalla Información de cola 2. Pulsar el botón de la barra de navegación Eliminar tique

5.4.2. Resultado de la ejecución casos de prueba

ID	Caso	Resultado esperado	Resultado
1.1	Registro tique manual	Se muestra una alerta informando que se ha registrado correctamente el tique y la pantalla con la información de la cola	OK
1.1	**Caminos alternativos:** ▪ Introducir código con formato incorrecto ▪ Introducir código inexistente ▪ Introducir código ya registrado, token FCM diferente ▪ Introducir código caducado	Aparece un mensaje de error identificativo del mismo	OK
1.2	Registro tique QR	Se muestra una alerta informando que se ha registrado correctamente el tique y la pantalla con la información de la cola	OK
1.2	**Caminos alternativos:** ▪ No se reconoce el código ▪ Código ya registrado, token FCM diferente ▪ Código caducado	Aparece un mensaje de error identificativo del mismo	OK
2	Desregistrar tique	Se muestra una ventana de confirmación para desregistrar el tique y si confirma aparece la pantalla inicial	OK

5.5. Diseño final

Finalmente, en este apartado se mostrará un conjunto de imágenes con el resultado final de la aplicación visto desde el emulador de Ionic para Android (izquierda) y iOS (derecha).

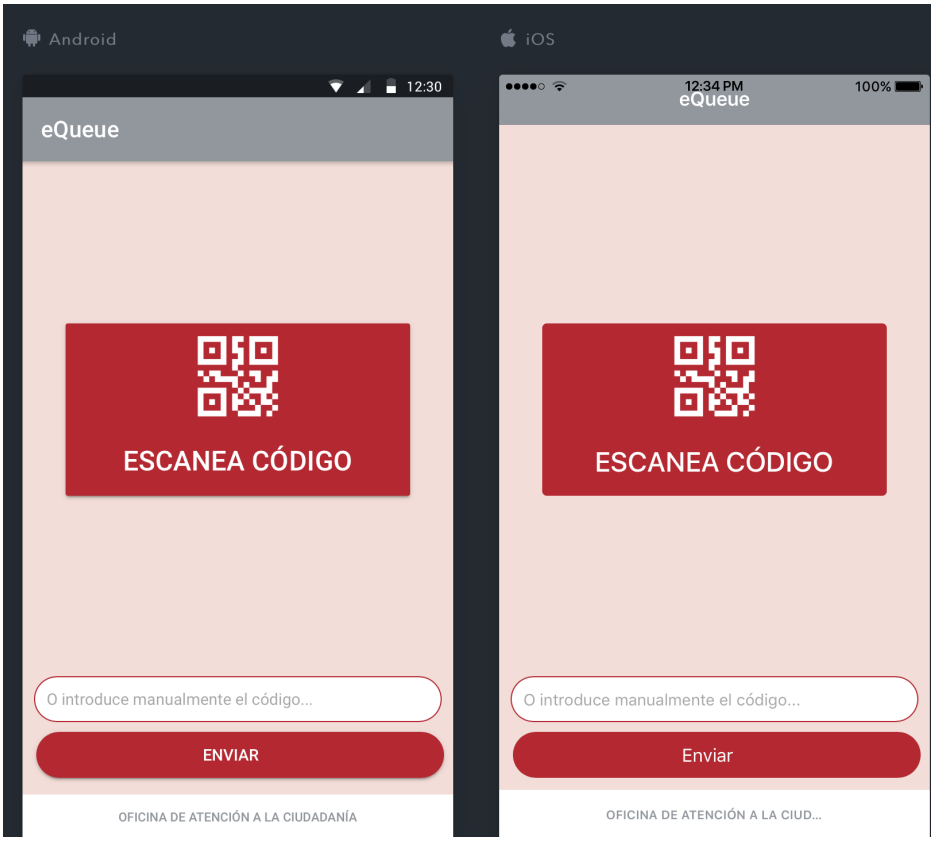


Figura 18: Pantalla inicial para Android y iOS

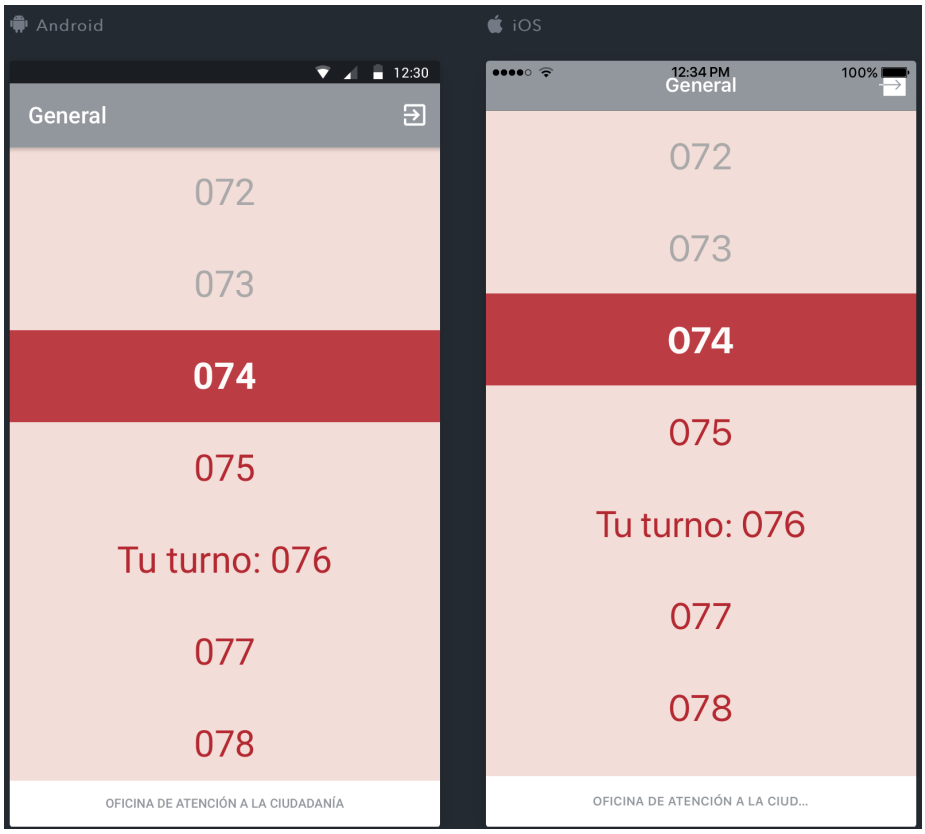


Figura 19: Pantalla con información de la cola para Android y iOS

6. Conclusiones

6.1. Resultado final

El resultado final del proyecto en términos metodológicos, ha sido el análisis, diseño e implementación de una aplicación móvil para añadir la funcionalidad al gestor integral de colas `X`, con el objetivo de optimizar el tiempo de los usuarios que van a hacer trámites a una oficina de atención a la ciudadanía.

En términos generales, la experiencia en el desarrollo con el framework de IONIC ha sido satisfactoria, gracias a toda la documentación disponible en la red es fácil resolver dudas y encontrar ejemplos. Inicialmente no se tenían demasiadas expectativas en el resultado con IONIC y ha sorprendido gratamente, IONIC es un framework potente y que cada vez dispone de más funcionalidades nativas. Además de profundizar en el estudio y evaluar la capacidad y potencial de IONIC, se han consolidado conocimientos previamente iniciados en el máster de programación TypeScript y html.

La fase de análisis es escasa puesto que los requisitos y objetivos del producto estaban previamente definidos por la empresa y los clientes que actualmente trabajan con `X`.

Respecto a la planificación, aunque se ha podido finalizar el trabajo

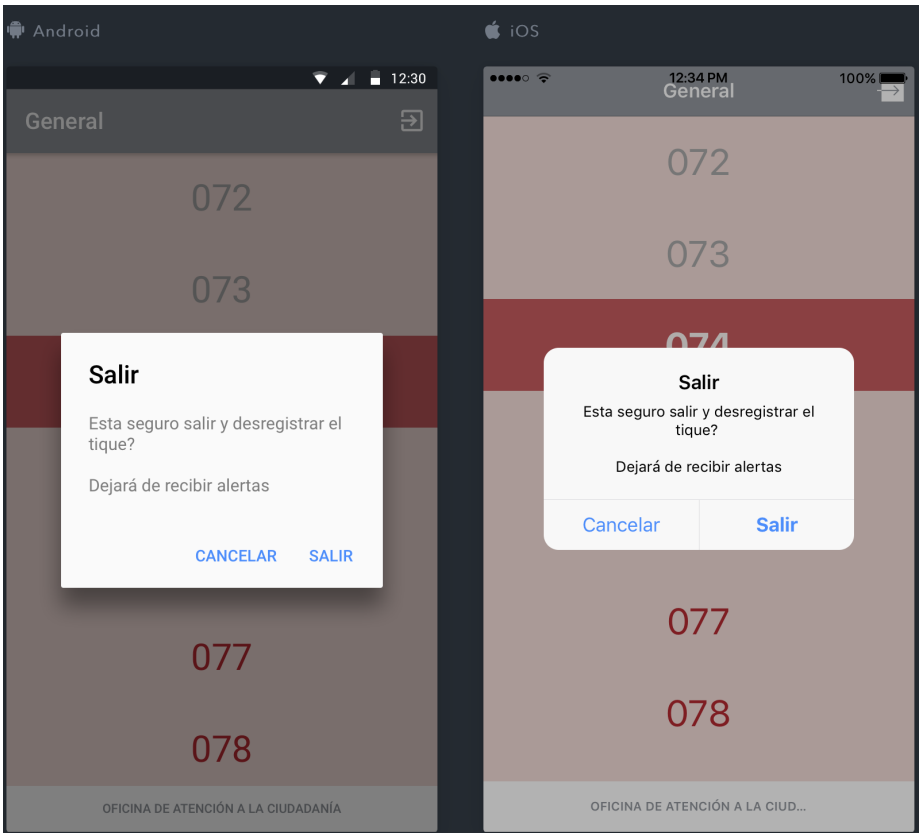


Figura 20: Pantalla alerta para desregistrar tique para Android y iOS

satisfactoriamente, no se han respetado los plazos para cada fase establecidos en la planificación, esto a provocado una descompensación de horas de trabajo en la etapa final.

6.2. Posibles trabajos futuros

La primera versión de aplicación ha finalizado con resultados satisfactorios, pero todavía no se puede poner a producción. Primero hay que hablar con los clientes que disponen del producto `X` y ofrecer esta nueva funcionalidad, que lo prueben en un entorno de pre-producción. Además, los clientes pueden corporizar la aplicación y eso podría suponer un cambio de diseño.

Para proyectos futuros en relación a esta aplicación se podría añadir más funcionalidades como poder pedir cita previa o poder coger un turno sin estar presente en la oficina.

Referencias

[1] Consola de firebase. https://console.firebase.google.com/.

[2] Cordova tools. https://marketplace.visualstudio.com/items?itemName=vsmobile.cordova-tools.

[3] Cómo inicializar proyecto ionic en vs code. https://blogs.msdn.microsoft.com/visualstudio/2016/03/30/build-ionic-apps-in-minutes-with-vs-code/.

[4] Descarga webstorm para mac. https://www.jetbrains.com/webstorm/download/#section=mac.

[5] draw.io. https://www.draw.io.

[6] Ionic creator. https://creator.ionic.io/.

[7] Ninjamock. https://ninjamock.com.

[8] Orderella. http://www.orderella.co.uk.

[9] Prototipo con ionic creator. https://creator.ionic.io/share/350d81315899.

[10] Página de descarga de visual studio code. https://code.visualstudio.com/.

[11] Sincola. http://www.sin-cola.com.

[12] Texmaker. http://www.xm1math.net/texmaker/.

[13] INTECO. *Ingeniería del software: metodologías y ciclos de vida.* Laboratorio Nacional de Calidad del Software de INTECO, 2009.

A. Compilar app IONIC

Para abrir nuestro proyecto en un navegador para poder ver y depurar lo que vamos haciendo, simplemente tenemos que ejecutar el siguiente comando en un terminal en la raíz del proyecto:

```
$ ionic serve
```

Emular o instalar en un dispositivo real Para emular un proyecto simplemente tendríamos que escribir en un terminal:

```
$ ionic emulate <PLATFORM>
```

Donde `PLATFORM` es la plataforma para la que queremos emular. Por ejemplo para emular en Android escribiríamos:

```
$ ionic emulate android
```

Y para instalar y ejecutar en un dispositivo real se hace de igual forma pero usando el comando run:

```
$ ionic run <PLATFORM>
```

En ambos casos podemos añadir la opción `--livereload` o `-l`, de esta forma tras cada cambio que hagamos en el código se recargará la emulación o la aplicación en el dispositivo. Por ejemplo:

Por último mencionar que si no queremos ejecutar el proyecto y solamente queremos compilarlo y generar el código destino de la plataforma podemos usar la opción build indicando también la plataforma de compilación:

```
$ ionic build <PLATFORM>
```

Si no indicamos la plataforma este comando compilaría el proyecto para todas las plataformas que tengamos instaladas.

IMPORTANTE Dado que los turnos caducan al finalizar el dia, no se puede provar la aplicación con un turno si no se crea el mismo día. Por lo tanto he creado una variable debug en el código HomePage y QueuePage que si está a true muestra información y no utiliza el servicio web.