Escuela Politécnica Superior de Jaén
Máster Universitario en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

Trabajo Fin de Máster

ESTUDIO Y DESARROLLO DE UN PROTOTIPO WEB PARA LA GAMIFICACIÓN DE RECORRIDOS CULTURALES

Alumno/a: **Jiménez De la Paz, José**

Tutor/a: Víctor Manuel Rivas Santos
Dpto.: Informática

Septiembre, 2021

Universidad de Jaén

Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Víctor Manuel Rivas Santos, tutor del Trabajo Fin de Máster titulado: ESTUDIO Y DESARROLLO DE UN PROTOTIPO WEB PARA LA GAMIFICACIÓN DE RECORRIDOS CULTURALES, que presenta JOSÉ JIMÉNEZ DE LA PAZ, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2021

El alumno: Los tutores:

José Jiménez De la Paz Víctor Manuel Rivas Santos

Índice de Contenido

Índice de ilustraciones

1. Introducción

Para finalizar el Máster de Ingeniería Informática he decidido basar mi Trabajo Fin de Máster en la realización de una aplicación móvil para gamificar itinerarios mediante la creación de rutas turísticas.

El objetivo fundamental de este proyecto es trasladar la mecánica del juego al ámbito cultural, fomentando el interés turístico de ese lugar para ello utilizamos el sistema de recompensa una vez completada cada ruta, y el sistema de competición implementando un ranking de los mejores usuarios, es decir, aquellos que más rutas haya completado.

Esta aplicación está enfocada desde varios puntos de vista:

En primer lugar, el perfil de usuario, que desde su propio dispositivo móvil se conectará con su cuenta a la aplicación y podrá acceder a distintas rutas disponibles en cada provincia. Además, en su perfil aparecerán las rutas ya completadas y las que está realizando en ese momento. La idea es que el usuario descubra las distintas rutas que ofrece cada provincia, cada ruta está formada por varios objetivos y mediante la captura de códigos QR el usuario va registrando los lugares visitados en el itinerario de la ruta hasta completarla.

En segundo lugar, el perfil de administrador, cuya función es la de crear rutas y objetivos de éstas, modificarlas, añadiendo o quitando objetivos o, incluso rectificando o ampliando la información cultural y fotografías, o eliminarlas.

Motivación:

Actualmente estudios como el de HMD Global junto con Fly Research demuestran que el uso de teléfonos móviles revela datos muy interesantes, como que dependemos tanto de nuestros teléfonos que los usamos una media de 142 veces al día, estando con ellos en uso 18 horas y 12 minutos a la semana. Además, revelan que España es el país que más tiempo pasa con el móvil. [1]

Los teléfonos móviles se han convertido en una herramienta imprescindible para nuestra vida, con ellos podemos comprar, gestionar nuestras finanzas, ayudarnos para hacer ejercicio, consumir contenido multimedia, etc.

Desde esta visión no es extraño usar el teléfono móvil en nuestros viajes, tanto para orientarnos con el uso de GPS, como para descubrir los lugares destacables del patrimonio cultural, gastronómico o natural de nuestro destino.

Reseñable es que el turismo en nuestro país aporta el 11,7 del PIB nacional y el 12,2% de la población activa tienen empleos que dependen del turismo, por ende es un potente motor económico. Con anterioridad a la crisis sanitaria, que recientemente hemos sufrido, ya se encontraba en revisión este sector, en busca de un desarrollo más sostenible. Así, apreciamos que, la Covid-19 ha acelerado esta transformación dado que los turistas buscan destinos ricos en patrimonio natural y cultural no masificados. [2]

Por todo lo anterior, se ha considerado buena idea desarrollar un prototipo de aplicación móvil para fomentar el turismo apoyándonos en la tecnología.

Si bien, debemos partir de que existe un gran mercado de teléfonos móviles con sus respectivos sistemas operativos, entre lo que destacan: Android e iOS, entre otros. Esta diversidad de sistemas operativos provoca desarrollos específicos dedicados para cada uno, y esto hay que tenerlo en cuenta si se quiere hace una aplicación para ser compatible con el mayor número de SO.

Una aplicación móvil híbrida multiplataforma, tiene como ventaja realizar un solo desarrollo de la aplicación y generar la aplicación con compatibilidad con distintos sistemas operativos. Por ello se decidió basar la aplicación en un framework que nos permitiera esto.

La realización de este proyecto me permite poner en práctica los conocimientos aprendidos durante mi formación académica.

1.2. Objetivos

El objetivo principal del proyecto es realizar una app móvil que sea compatible para los 2 sistemas operativos de móvil más usados en la actualidad, Android o Apple iOS, y sea capaz de proporcionar a los usuarios una forma cómoda y divertida de descubrir los puntos culturales de sus viajes. Se realizará con el Framework Ionic que está basado en Angular y el servidor con Spring Boot.

El proyecto implicará un estudio previo de soluciones existentes, además de análisis, diseño e implementación.

1.3. Objetivos específicos

De forma detallada, para la realización de este TFM se establecieron los siguientes objetivos, cuyo alcance y consecución se presenta a lo largo de esta memoria:

Realizar un estudio de necesidades y soluciones existentes en el contexto seleccionado. Diseñar un sistema informático especialmente orientado a la utilización de arquitecturas, principios de diseño y metodologías de desarrollo web.

Seleccionar un entorno de desarrollo web e implementar un prototipo de aplicación web que satisfaga las necesidades que se hayan determinado.

1.3. Metodologías

Para desarrollar este trabajo, se consideró aplicar una metodología ágil porque es capaz de soportar de forma óptima los cambios durante el desarrollo. Esta metodología nos permite ir viendo las funcionalidades intervalos para que el producto final sea una suma de entregables que nos proporcione un mejor seguimiento, y conocer realmente cuales son las tareas innecesarias para poder centrar los esfuerzos en las verdaderamente importantes. Se comentan a continuación algunas ideas sobre este tipo de metodologías.

Kent Beck y otros expertos en la industria del software se reunieron Utah, EEUU en 2001 [3]. Firmaron el 'Manifiesto por el desarrollo ágil del software' [4], en el que destacaron los siguientes principios:

Individuos e interacciones sobre procesos y herramientas.

- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio, mejor que acogerse a un plan.

Dando importancia a los elementos de la derecha, aunque con más importancia aún a los de la izquierda, surgió un nuevo paradigma.

Principios derivados del manifiesto. [5]

- Satisfacción del cliente como prioridad, mediante entregas frecuentes y en etapas tempranas.
- Los cambios son bienvenidos, aun en una etapa avanzada del desarrollo.
- Entrega frecuente de software que funcione, de 2 semanas a un par de meses lo más pronto que se pueda.
- El personal de la empresa y los desarrolladores deben trabajar juntos y mejorar la comunicación.
- Motivación del equipo, con los trabajadores motivados se aumentará la productividad.
- Comunicación cara a cara como método de comunicación con los integrantes del equipo principal.
- El software que funciona es la mejor medida de progreso.
- Ritmo de desarrollo sostenible, máximo 40 horas semanales.
- Atención continua a la calidad técnica y al buen diseño.
- Maximizar el trabajo no hecho, no desarrollar más de lo acordado.
- Equipos auto-organizados, mantener los equipos de un proyecto a otro.
- Autoevaluación, el equipo se reúne regularmente para hacer autocrítica y plantear posibles mejoras.

El proceso ágil sigue haciendo las mismas actividades del proceso de desarrollo de software, aunque difiere en la forma de hacerlo. [6]

Como se observa en las ilustraciones 1 y 2 se puede observar que las fases de desarrollo en el proceso ágil son las mismas, pero varía en la forma de hacerlo.

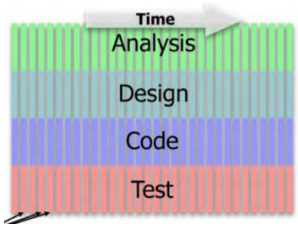


Ilustración 2 Proceso ágil

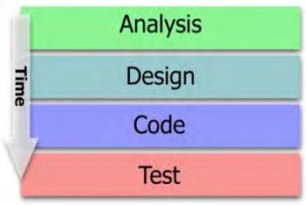


Ilustración 1 Proceso tradicional

Pero, aunque tengan las mismas actividades tiene una consecuencia importante a la hora de realizar el proyecto. En el enfoque tradicional se fijan los requisitos y a partir de ellos se estiman el tiempo y coste para poder cumplir los requisitos. Sin embargo, con el enfoque ágil se fijan el tiempo y el coste y a partir de estos dos valores se estima los requisitos [7]. Como se observa en las ilustraciones 3 y 4.

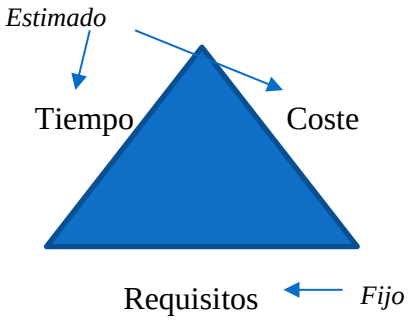


Ilustración 4 Enfoque tradicional

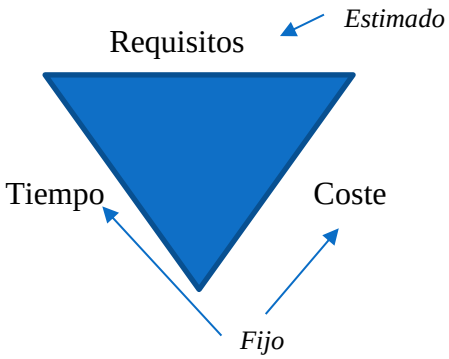


Ilustración 3 Enfoque ágil

El enfoque ágil "suaviza" el costo de cambio (véase la Ilustración 6) consiguiendo la satisfacción de los clientes. El proceso ágil contempla la entrega incremental además de pruebas unitarias continuas, disminuyendo el coste de cambios.

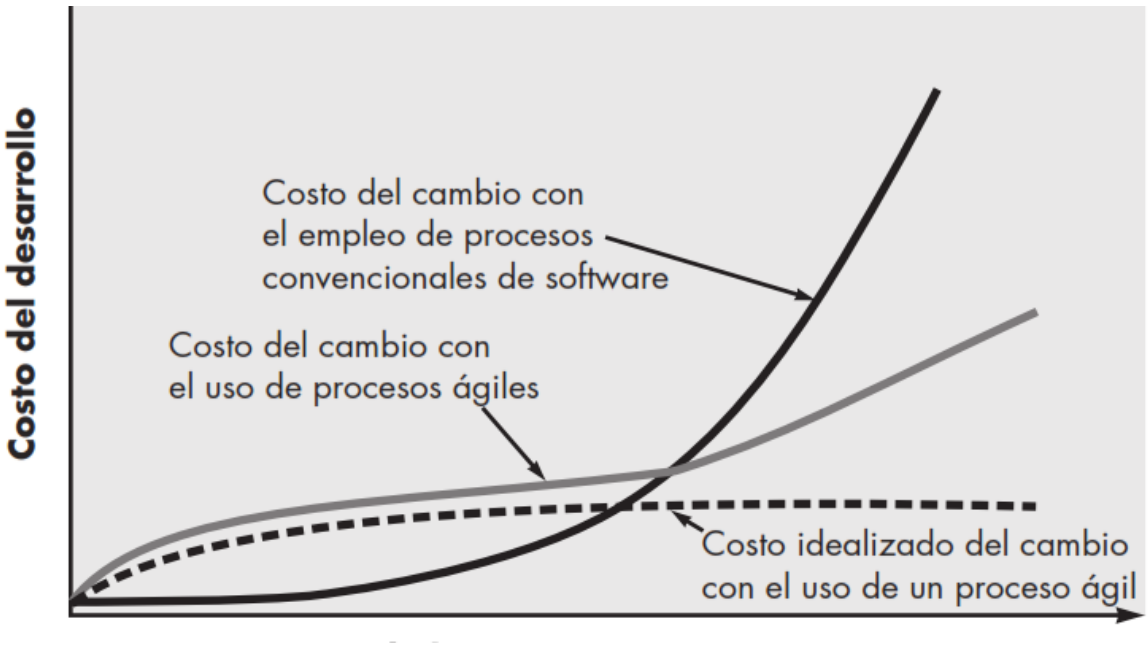


Ilustración 5 Costo del cambio

Finalmente debido a que el proyecto tiene una alta probabilidad de que tenga requisitos cambiantes, predispuestos al cambio y modificaciones es uno de los motivos que me ha hecho decidirme a este tipo de metodología. Concretamente la metodología elegida es SCRUM.

1.4. SCRUM

SCRUM es un proceso ágil adaptable, iterativo, rápido, flexible y eficaz que está diseñado para entregar valor al cliente durante el desarrollo del proyecto. [8]

SCRUM usa patrones de desarrollo que son eficaces para proyecto con plazos de entrega cortos y requisitos cambiantes. Con intervalos de 2 o 4 semanas se puede ver el software funcionando y decidir si seguir mejorándolo o entregarlo.

El objetivo principal es satisfacer las necesidades del cliente, favoreciendo la comunicación con el cliente y el equipo de desarrolladores.

Ciclo de trabajo en SCRUM:

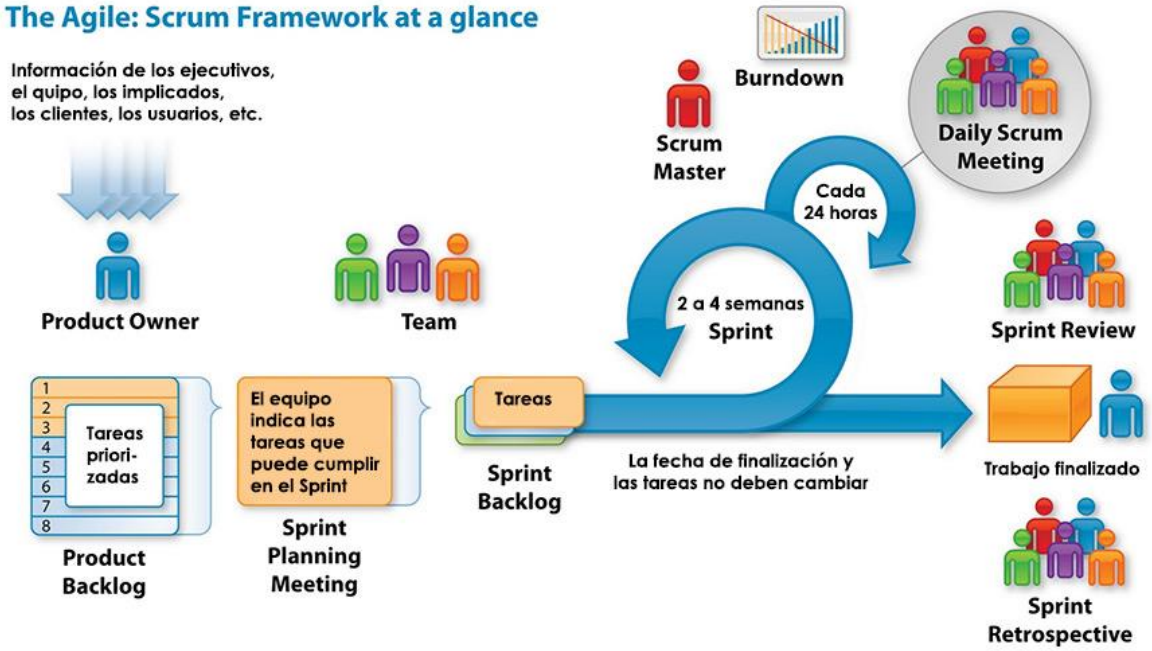


Ilustración 6 Ciclo de trabajo SCRUM

1. Toma de requisitos al cliente, para cada requisito principal se crea una historia de usuario.
2. El cliente ordena la lista priorizada del producto (*backlog*), que contiene la lista de los requisitos en orden de prioridad para el negocio en forma de historias de usuario.
3. El equipo de trabajo toma un grupo de historias del *backlog* y las incluyen en la planificación del Sprint divididas en sub tareas más sencillas. El equipo trabaja en la realización de los entregables durante 2-6 semanas. En el Sprint no se admiten cambios para que el equipo trabaje de forma estable y a corto plazo.
4. Durante el Sprint se realizan reuniones diarias y se debe responder a 3 preguntas clave:
    a. ¿Qué hiciste ayer?
    b. ¿Qué harás hoy?
    c. ¿Hay obstáculos en tu camino?

Al final de cada Spring se realiza una reunión de retrospectiva del Spring donde se analizan los aspectos que mejorar y los problemas ocurridos durante el mismo.

Una vez finalizado el Spring se entrega al cliente el resultado del trabajo y se vuelve al punto 2 hasta acabar con las historias.

1.5.1. Adaptación de SCRUM al proyecto

En mi caso, para adaptar esta metodología ágil he tenido que adaptar algunos roles y uso de reuniones:

- ***Product Owner*:** sí se mantiene, es el responsable del producto. Define las funcionalidades principales del producto, estima fechas, prioriza las funcionalidades, además de evaluar los entregables. Este rol lo adoptaría yo y mis familiares cercanos, que les iré enseñando la aplicación.
- ***Scrum Master y Team*:** El responsable del funcionamiento de SCRUM y equipo de desarrollo (analista y programador) lo haría individualmente.
- **Reuniones de equipo:** Puesto que yo adoptaré el perfil de equipo, no tiene sentido hacer las reuniones, aunque sí es útil hacer un seguimiento del trabajo realizado, anotando los problemas, lo que se hará en el futuro y lo que hice el día anterior.

1.6. Estructura del documento

El documento se ha organizado de la siguiente forma: en el apartado 2 llamado "Análisis" se describirá el sistema del que se parte y se verán sus desventajas además de ver otras aplicaciones similares y la propuesta de solución, incluyendo historias de usuario, planificación y viabilidad.

En el apartado 3 de diseño se mostrarán los detalles que se han tenido en cuenta a la hora de organizar la información que gestionará la aplicación, así como la estructura y funcionamiento del software a desarrollar, para los que se incluyen: el diagrama entidad relación de la base de datos, el diagrama de clases de la aplicación, diagramas de secuencia de los aspectos más relevantes y diseño de la interfaz mediante un programa informático.

En apartado 4 de implementación se detallará la arquitectura de la aplicación y la relación entre los elementos, además de los detalles sobre la implementación donde se menciona técnicas y tecnologías empleadas.

Después en el apartado 5 llamado “Pruebas”, se mostrará cómo se ha comprobado el correcto funcionamiento de la aplicación y si se ha obtenido los resultados esperados.

En el apartado 6 se explica las conclusiones del proyecto y trabajos futuros para ampliar o mejorar el proyecto.

Finalmente se añaden tres anexos: manual de instalación del sistema, manual de usuario y manual de contenidos suministrados.

2. Análisis

La etapa de análisis debe estar completa antes de pasar al diseño. El análisis proporciona al diseño toda la información necesaria para crear los modelos del diseño [1].

2.1 Análisis preliminar

Para el análisis previo podemos partir de la situación turística de no hace mucho tiempo. Si nos remontamos a hace unos años podemos ver que era imprescindible para viajar contactar con distintas empresas externas como agencias de viajes para organizar nuestro viaje.

Si bien, con la irrupción de internet y el uso de app en nuestros dispositivos móviles resulta mucho mas cómodo e incluso económico, dado que podemos consultar distintas webs y ofertas.

Actualmente, tras pasar la pandemia del Covid-19 podemos apreciar la tendencia a realizar turismo a lugares menos masificados y menos conocidos popularmente. Estos lugares presentan varias desventajas para atraer a los turistas:

- No es popularmente conocido.
- No está en las primeras posiciones de los buscadores al realizar búsquedas genéricas del estilo: “qué visitar en Andalucía”

- No está ubicado geográficamente en un punto de transito de viajeros.

Para solventar estos incovenientes que pueden hacer tedioso la búsqueda de lugares nuevos y desconocidos, se ha pensando en la app propuesta.

Con esta aplicación Hay varias alternativas para hacer una aplicación web para el móvil: aplicación nativa, aplicación híbrida o una web.

Ilustración 7 Comparativa tipo de apps

La principal desventaja de hacer una aplicación nativa es que no sería multiplataforma y solo nos serviría para el sistema operativo en que se base. Una aplicación nativa sería un inconveniente para el proyecto.

Basar la aplicación en una web, tendría inconvenientes, como que no nos proporcionaría una interfaz fluida e intuitiva, no se podría subir a las tiendas de aplicaciones móviles y tendría poco rendimiento.

Una aplicación híbrida es una combinación *de HTML, CSS y JavaScript*, que se ejecuta dentro de un contenedor nativo. Permite usar funcionalidades del dispositivo como el GPS o la cámara entre otras.

Además, una aplicación híbrida proporciona una interfaz fluida e intuitiva, multiplataforma, posibilidad de subir la aplicación a las tiendas de aplicaciones móviles entre otras ventajas.

Una vez realizada la comparación con las tres opciones, nos hemos decantado por realizar una aplicación híbrida. Puesto que sus características son ideales para el proyecto.

2.2 Estado del arte

A continuación, se comentarán algunas aplicaciones similares a la que vamos a desarrollar para descubrir recorridos culturales. Esto servirá para coger ideas para nuestra aplicación y evitar los posibles fallos.

Minube [1]

Esta aplicación es la más completa, ofrece diversas funcionalidades, como descubrir las zonas destacables de los lugares, comentarios, imágenes etc.

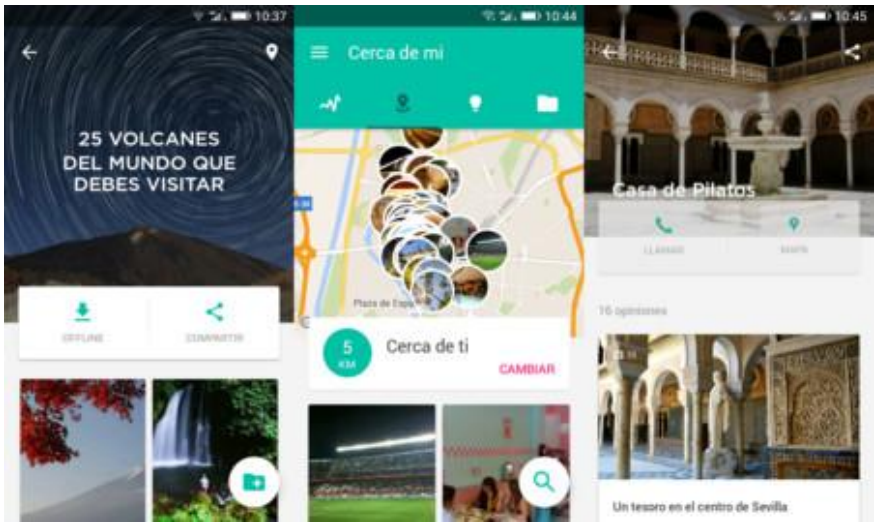


Ilustración 8 Capturas de Minube

Ventajas:

[1] https://www.minube.com/movil

- Recomendación de restaurantes
- Recomendación de hoteles.
- Sitios interesantes desde la ubicación actual.
- Sistema de comentarios.
- Sistema de favoritos y planificación

Inconvenientes:

- No dispone de un sistema de recompensa por visitar los lugares,
- No tiene la funcionalidad de ranking de los mejores usuarios que más sitios han visitado.
- No agrupa los objetivos por rutas interesantes.

Yelp [2]

Esta aplicación nos ayuda a descubrir lugares interesantes a través de las reseñas de su gran comunidad de usuarios.

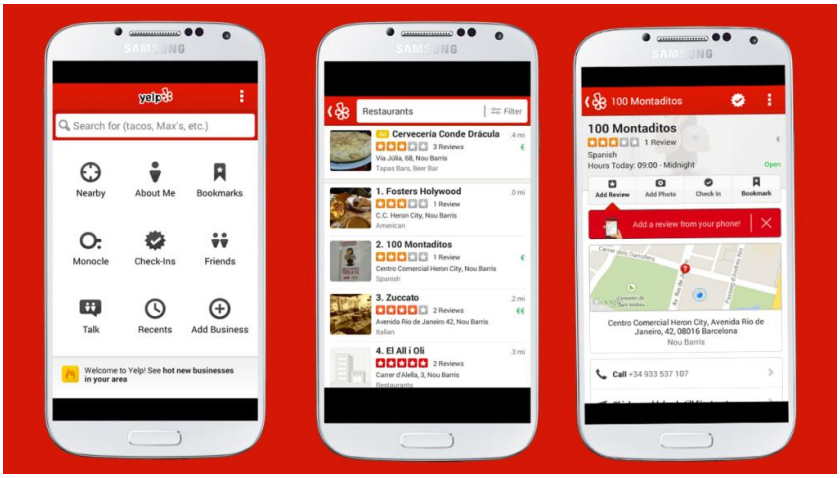


Ilustración 9 Capturas de Yelp

Ventajas

- ofrecer recomendaciones sencillas a los usuarios fomentando a la vez su participación.
- En Yelp encontraremos recomendaciones geolocalizadas sobre todo tipo de lugares

[2] https://www.yelp.es/madrid

- Es internacional

Inconvenientes

- La interfaz es menos amigable.
- No tiene sistema de rutas
- No tiene sistema de recompensas.

2.3 Propuesta de solución

Con este análisis previo, se ha quedado más claro el sistema que se quiere realizar. El objetivo es hacer un prototipo de una aplicación web móvil híbrida instalable y multiplataforma.

El sistema estará dividido en dos aplicaciones, la del cliente y la del servidor.

Cliente

El framework elegido para hacer la aplicación web híbrida es Ionic que es un SDK (software development kit) que provee servicios y herramientas para desarrollar aplicaciones híbridas. Está basado en Angular y Apache Cordova , el objetivo es simplificar el front-end. Ionic es gratuito, con foro para dudas y un gran manual online donde explica con ejemplos todo su potencial.

Servidor

El servidor deberá proporcionar una API RESTFUL para la comunicación el cliente, además de proporcionar un sistema de identificación por token y acceso a la base de datos. La mejor opción es utilizar Spring Boot .

Spring boot es un framework basado en Java, nos permite hacer Servidores de aplicaciones embebidos (Tomcat, Jetty, Undertow), POMs con dependencias y plug-ins para Maven, uso extensivo de anotaciones que realizan funciones de configuración, inyección, etc.

Utilizaremos JPA para mapear las clases de persistencia (que queramos guardar en la base de datos) haciendo uso de DAOs para acceder a la base de datos. Como base de datos se utilizará MySQL.

La combinación de Ionic y Spring Boot nos dará una interfaz atractiva para el cliente y fluida y un servidor estable que proporciona todo lo necesario para el cliente.

2.4 Historias de usuario

Puesto que se ha decidido usar una metodología ágil, concretamente SCRUM, detallaremos más adelante las historias de usuario, estas se centran en el valor del cliente con una fuerte intención de fomentar la comunicación. Y obtendremos así los requisitos.

Una historia de usuario es la unidad de trabajo más pequeña en un marco ágil. Es un objetivo final, no una función, expresado desde la perspectiva del usuario del software.

Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente. [9]

Cada historia de usuario es enumerada con un número entero que sirve de identificador, además tiene criterios de satisfacción. Los criterios de satisfacción describen las acciones que hay que realizar una vez acabada la historia de usuario para que se válida. Más adelante se mostrará una tabla con esta información.

Con la ayuda del *Product Owner*, en este caso es el cliente, he realizado las siguientes historias de usuario que recoge las funcionalidades principales de la aplicación.

Haré una aclaración de los roles que se usan en las historias de usuario:

- **Usuario**: usuario que usa la aplicación.
- **Administrador**: rol encargado de la administración.

ID	Título	Descripción
1	Registro en el sistema	Como cliente, quiero que los usuarios puedan registrarse en el sistema para acceder a las funcionalidades que requieran autenticación, rellenando los datos del formulario correspondiente.
2	Identificación en el sistema	Como cliente quiero que los usuarios puedan acceder al sistema introduciendo sus credenciales.
3	Listado de provincias	Como cliente quiero que los usuarios accedan a la lista de provincias disponibles,, una vez seleccione una, le aparezca una vista con las rutas de dicha provincia.
4	Lista de rutas	Como cliente, quiero que los usuarios puedas ver la lista de rutas perteneciente a una provincia.
5	Información de la ruta	Como cliente , quiero que los usuarios puedan acceder a la información de la ruta. Mostrando la descripción, imagen, objetivos que incluye y ubicación de cada objetivo en el mapa.
6	Información del objetivo	Como cliente quiero que los usuarios puedan acceder a la información del objetivo, mostrando descripción, imagen, url a Wikipedia para más información.
7	Navegación GPS	Como cliente, quiero que los usuarios desde la ubicación de cada objetivo en el mapa, se inicie su navegador gps y cómo destino la ubicación del objetivo seleccionado.
8	Captura códigos QR.	Como cliente, quiero que los usuarios puedan capturar el código QR del objetivo, registrando esta captura.
9	Perfil de usuario	Como cliente, quiero que los usuarios puedan acceder a su perfil de usuario, para poder cerrar sesión y ver las rutas en curso con la información de las capturas realizadas.
10	Nueva Ruta	Como cliente, quiero que el administrador, pueda crear una nueva ruta, introduciendo los datos necesarios como, nombre, descripción, provincia a la que pertenece y objetivos que incluye.

11	Nuevo objetivo	Como cliente, quiero que el administrador pueda crar un nuevo objetivo, introduciento el nombre, descripción, imagen, coordenadas, url wiki.
12	Ranking de usuarios	Como cliente quiero, que los usuarios puedan acceder al ránking de los usuarios con mayor número de capturas.
13	Ver premio de la ruta	Como cliente, quiero que el usuario pueda ver el premio que se le daría al completar la ruta.
14	Asignar premio a la ruta	Como cliente quiero, que el administrador pueda asignar el premio a la ruta.
15	Bonificación por ruta	Como cliente quiero, que el usuario pueda obtener la bonificación tras haber completado la ruta, mediante un correo electrónico.
16	Sistema de comentarios	Como cliente quiero, que los usuarios puedan hacer comentarios sobre las rutas y sobre cada objetivo.
17	Sistema de favoritos	Como cliente, quiero que los usuarios puedan asignar a favoritos un objetivo, para poder verlo más tarde desde la vista de favoritos.

Ilustración 10 Tabla de casos de uso

2.5 Planificación temporal

En este apartado se hará la planificación temporal para realizar las historias de usuario, esto conlleva a la estimación, priorización y cálculo de la velocidad.

2.5.1. Estimación de las historias de usuario y definición de las tareas.

A continuación, se estiman las historias de usuario para hacer una idea de la dificultad que llevaría hacer cada una, he utilizado la serie de *Fibonacci*: 1,2,3,5,8 cuanto mayor sea la nota mayor dificultad conlleva realizarla.

En primer lugar, he buscado una historia para que sirva como referencia en la estimación de las demás, esta historia no es ni la más fácil ni la más difícil: “Información del objetivo” con 5 puntos asignados.

En base a esta historia hemos estimado las demás, siendo la más difícil “Registro en el sistema” con 8 puntos, porque conlleva el estudio de cómo tratar la se seguridad en el servidor y tratamiento del token en el front.

Cabe destacar que en la tabla siguiente se descompone cada historia de usuario en tareas. Esta descomposición se debería realizar al sacar la historia de usuario del *Backlog*. Además de reflejar las horas de trabajo realizadas con cada perfil.

ID	HU / tareas	Estimación (puntos de historia) / Asignación
1	**Registro en el sistema**	**8 PH**
1.1	Estudio de Spring Security y JWT, back	Analista | 5 horas
1.2	Análisis, diseño del servicio, back	Analista | 5 horas
1.3	Implementación del servicio ,back	Programador | 5 horas
1.4	Diseño de la vista, front	Programador | 3 horas
1.5	Estudio de almacenamiento en Storage del dispositivo, front	Programador | 2 horas
1.6	Implementación funcionalidades ,front	Programador | 2 horas
1.7	Pruebas	Programador | 1 hora
2	**Identificación en el sistema**	**8 PH**
2.1	Análisis, diseño del servicio back	Analista | 6 horas
2.2	Implementación del servicio back	Programador | 5 horas
2.3	Diseño de la vista front	Programador | 2 horas
2.4	Implementación funcionalidades front	Programador | 5 horas
2.5	Pruebas	Programador | 2 horas
3	**Listado de provincias**	**2 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 horas
1.2	Implementación del servicio back	Programador | 3 horas
1.3	Diseño de la vista front	Programador | 3 horas
1.4	Implementación funcionalidades front	Programador | 3 horas
1.5	Pruebas	Programador | 1 hora
4	**Lista de rutas**	**2 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 horas
1.2	Implementación del servicio	Programador | 3 horas

	back	
1.3	Diseño de la vista front	Programador | 3 horas
1.4	Implementación funcionalidades front	Programador | 3 horas
1.5	Pruebas	Programador | 1 hora
5	**Información de la ruta**	**8 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 horas
1.2	Implementación del servicio back	Programador | 4 horas
1.3	Diseño de la vista front	Programador | 4 horas
1.4	Implementación funcionalidades front	Programador | 8 horas
1.5	Pruebas	Programador | 1 horas
6	**Información del objetivo**	**5 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 hora
1.2	Implementación del servicio back	Programador | 3 horas
1.3	Diseño de la vista front	Programador |6 horas
1.4	Implementación funcionalidades front	Programador | 4 horas
1.5	Pruebas	Programador | 1 horas
7	**Navegación GPS**	**5 PH**
1.3	Diseño de la vista front	Programador | 3 horas
1.4	Implementación funcionalidades front	Programador | 4 horas
1.5	Pruebas	Programador | 2 horas
8	**Capturas códigos QR**	**5 PH**
1.3	Diseño de la vista front	3 horas
1.4	Implementación funcionalidades front	6 horas
1.5	Pruebas	1 hora
9	**Perfil de usuario**	**5 PH**
1.1	Análisis, diseño del servicio back	Analista | 5 horas
1.2	Implementación del servicio back	Programador | 6 horas
1.3	Diseño de la vista front	Programador | 4 horas
1.4	Implementación funcionalidades front	Programador | 4 horas
1.5	Pruebas	Programador | 2 horas
10	**Nueva ruta**	**3 PH**
1.1	Análisis, diseño del servicio back	Analista | 5 horas
1.2	Implementación del servicio back	Programador | 3 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación	Programador | 3 horas

	funcionalidades front	
1.5	Pruebas	Programador | 2 horas
11	**Nuevo objetivo**	**5 PH**
1.1	Análisis, diseño del servicio back	Analista | 3 horas
1.2	Implementación del servicio back	Programador | 6 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación funcionalidades front	Programador | 3 horas
1.5	Pruebas	Programador | 2 horas
12	**Ranking de usuarios**	**3 PH**
1.1	Análisis, diseño del servicio back	Analista | 3 horas
1.2	Implementación del servicio back	Programador | 2 horas
1.3	Diseño de la vista front	Programador | 3 horas
1.4	Implementación funcionalidades front	Programador | 2 horas
1.5	Pruebas	Programador | 2 horas
13	**Ver premio de la ruta**	**3 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 horas
1.2	Implementación del servicio back	Programador | 2 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación funcionalidades front	Programador | 2 horas
1.5	Pruebas	Programador | 2 horas
14	**Asignar premio a la ruta**	**5 PH**
1.1	Análisis, diseño del servicio back	Analista | 1 hora
1.2	Implementación del servicio back	Programador | 4 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación funcionalidades front	Programador | 3 horas
1.5	Pruebas	Programador | 1 hora
15	**Bonificación por ruta**	**8 PH**
1.1	Análisis, diseño del servicio back	Analista | 4 horas
1.2	Implementación del servicio back	Programador | 6 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación funcionalidades front	Programador | 4 horas
1.5	Pruebas	Programador | 3 horas
16	**Sistema de comentarios**	**8 PH**
1.1	Análisis, diseño del servicio	Analista | 8 horas

	back	
1.2	Implementación del servicio back	Programador | 4 horas
1.3	Diseño de la vista front	Programador | 5 horas
1.4	Implementación funcionalidades front	Programador | 5 horas
1.5	Pruebas	Programador | 2 horas
17	**Sistema de favoritos**	**5 PH**
1.1	Análisis, diseño del servicio back	Analista | 2 horas
1.2	Implementación del servicio back	Programador | 3 horas
1.3	Diseño de la vista front	Programador | 2 horas
1.4	Implementación funcionalidades front	Programador | 3 horas
1.5	Pruebas	Programador | 1 hora

2.5.2. Priorización de las historias de usuario

La priorización proporciona una ordenación de las historias de usuario según su importancia en la aplicación. Esto nos ayudará para determinar qué funcionalidades podrán desarrollarse con los recursos y tiempo con los que se dispongan.

Para priorizar las historias de usuario hemos utilizado el sistema de priorización *MoSCow* [10]. Se basa en agrupar las historias de usuarios en grupos llamados: *Must, Should, Should y Would Not*. A continuación, se detallará cada grupo y las historias de usuario que los forman.

En la siguiente tabla se agrupan las historias de usuario del grupo *Must*, que son los requisitos que deben tener la solución, son los requisitos mínimos.

Must		
ID	Nombre	Puntos de historia
1	Registro en el sistema	8
2	Identificación en el sistema	8
3	Listado de provincias	2
4	Lista de rutas	2
5	Información de la ruta	8
6	Información del objetivo	5

8	Capturas códigos QR	5
10	Nueva ruta	3
11	Nuevo objetivo	5

A continuación, se agrupan las historias de usuario del grupo *Should*, que son los requisitos que se deberían implementar, pero no son obligatorios necesarios.

Should		
ID	Nombre	Puntos de historia
7	Navegación GPS	5
9	Perfil de usuario	5

Los del grupo *Could* son requisitos que podrían incluir la solución.

Could		
ID	Nombre	Puntos de historia
12	Ranking de usuarios	3
13	Ver premio de la ruta	3
14	Asignar premio a la ruta	5

Finalmente, los del grupo *Would not* son requisitos que en un futuro podrían hacerse.

Would not		
ID	Nombre	Puntos de historia
15	Bonificación por ruta	8
16	Sistema de comentarios	8
17	Sistema de favoritos	5

2.6 Estimación de velocidad

Para calcular la duración de la planificación desde los puntos de historia, se utiliza el concepto de velocidad, la velocidad es una medida para estimar el avance del equipo de desarrollo.

Por temas de tiempo se decide descartar las HU del grupo *would not,* e incluir incluso las del grupo *could* para intentar desarrollarlas. La suma de los puntos de historia es 70.

La fecha de comienzo del proyecto será el 22 de marzo de 2021 y la de finalización el 21 de junio, en este periodo de tiempo disponemos de 16 semanas.

La primera semana del 22 de marzo se dedica a la instalación de SW y diseño. Instalación de programas como IntelliJ, MySql, Postman, diagrama de clases, etc.

La última semana será dedicada para terminar el desarrollo de la documentación , guía de usuario y conclusiones.

En total, sin contar las semanas mencionadas anteriormente tenemos 6 semanas para el desarrollo. Por lo cual se ha decidido abordarlo en 6 *sprints*.

Velocidad: 67 ph / 6= 11 ph por sprint.

Con los datos anteriores se realiza la siguiente planificación.

Nº de Sprint	Fecha Inicio	Fecha Fin	Puntos de historia	Velocidad	Historias de usuario
0	22 marzo	28 marzo	Instalación SW y diseño		
1	29 marzo	11 abril	10	90%	1,3
2	12 abril	25 abril	10	90%	2,4
3	26 abril	9 mayo	8	72%	5
4	10 mayo	23 mayo	10	90%	6,7
5	24 mayo	6 junio	13	120%	8,9,10
6	7 junio	20 junio	16	140%	11,12,13,14
7	21 junio	27 junio	Documentación		

Como se aprecia en la tabla anterior, la primera historia de usuario se empieza a hacer el 29 de marzo y la última el 7 de junio. También se asocia una velocidad a cada *Sprint*, concretamente hay varias superiores al 100%, el *Sprint* 5 y 6. Esto quiere decir que en esas semanas hay que trabajar más horas que las planificadas para poder alcanzar el objetivo planificado de terminar esas historias de usuario.

2.7 Evolución de las estimaciones

El *Sprint* 1,2,3,4, como sus velocidades son por debajo del 100% reflejan que habrá más tiempo para terminar las historias de usuario asignadas.

Al abordar el sprint 5, no se pudo finalizar a tiempo todas las historias de usuario que este definía. Se desarrollo la historia de usuario 8 y 9, sin embargo la 10 (3 PH) no se pudo abordar.

Es por ello que se decidió ajustar la velocidad para terminar la parte faltante y el sprint siguiente de desarrollo, para poder terminar a tiempo.

Para ajustar la velocidad se decidió no abordar las historias de usuario del grupo *Could*.

Could		
ID	Nombre	Puntos de historia
12	Ranking de usuarios	3
13	Ver premio de la ruta	3
14	Asignar premio a la ruta	5

En el Sprint 6, se aborda los PH que faltaron del sprint 5 (3 PH), más los puntos de historia del sprint 11 (5).

Nº de Sprint	Fecha Inicio	Fecha Fin	Puntos de historia	Velocidad	Historias de usuario
0	22 marzo	28 marzo	Instalación SW y diseño		
1	29 marzo	11 abril	10	90%	1,3

2	12 abril	25 abril	10	90%	2,4
3	26 abril	9 mayo	8	72%	5
4	10 mayo	23 mayo	10	90%	6,7
5	24 mayo	6 junio	13	120%	8,9
6	7 junio	20 junio	5+3	72%	10,11
7	21 junio	27 junio	Documentación		

2.8 Estimación de costes

Este punto está dedicado a la estimación económica para el desarrollo de este proyecto. Para calcular el coste económico se tendrá en cuenta el software, hardware y coste del personal para el desarrollo. Habrá un incremento al coste total del 17% para tener un margen de beneficio.

El presupuesto dedicado a software es bastante reducido, ya que se ha intentado utilizar software con licencias gratuitas.

Programas	Precio	Tiempo de uso	Precio de uso
IntelliJ IDEA 2021 1.1	0€ Licencia UJA	5 meses	0€
Microsoft Office 365 Personal	7€/mes	5 meses	35€
Visual Studio Code 1.56	0€	5 meses	0€
Postman 7.36	0€	5 meses	0€
Visual Paradigm Community Edition	0€ Licencia UJA	5 meses	0€
MAMP	0€	5 meses	0€
TOTAL:			35€

Ilustración 11 Estimación de costes software

A continuación se calcula el coste hardware, para realizar el cálculo se ha tenido en cuenta la vida útil del componente y la estimación del tiempo de uso que se le ha dado. El coste económico total es calculado por estas dos variables.

Hardware	Precio	Vida estimada	Precio de uso
MacBook Pro Retina 13"	1400€	10 años	80€
Poco X3 NFC	200€	4 años	70€
TOTAL:			150€

Ilustración 12 Estimación costes hardware

Respecto al coste salarial, para el cálculo de un analista programador junior , se ha obtenido como referencia el BOE 13 mayo de 2020 [11]

El total de horas de trabajo ha sido calculado de la siguiente forma, los días de trabajo realizados con el rol correspondiente en este caso Analista y programador multiplicado por las horas al día de trabajo en este caso 5 horas.

Y finalmente las horas de trabajo multiplicadas por el salario por hora de cada rol nos da como resultado el coste total por cada uno.

Cargo	Salario Mensual	Salario por hora	Horas al día	Total de horas	Precio
Programador	1249,08€	7,60€	5	269	2044,4€
Analista	1406,90€	8,78€	5	83	728,74€
Total	2773,14€				

Ilustración 13 Estimación de coste salarial

Ya calculados los tres costes anteriores, se procede a la suma de ellos y a la aplicación del beneficio, quedando como resultado 3460,94€.

Costes	Precio
Coste hardware	150,00€
Coste software	35,00€
Coste personal	2773,14€
Coste total	2958,14€
Beneficio- 17% del total	502,8€
Total:	3460,94€

Ilustración 14 Coste final

2.9 Modelo de dominio

A continuación, a partir de los conceptos anteriores de análisis se intenta hacer una aproximación a la solución, donde podemos intuir las clases conceptuales que se crearán en el proyecto.

La clase usuario contiene la información del usuario además de la información necesaria para realizar la identificación con la plataforma. Está relacionada con la tabla rol para identificar si es de tipo cliente o administrador.

La tabla captura hace referencia a cuando u usuario captura un código QR con el móvil, en ese momento se crearía un nuevo registro en esta tabla con la información de la fecha en la que se produjo.

La captura está relacionada con el objetivo, y este a su vez con ruta y provincia.

A partir de este modelo se centrará el estudio del diseño donde posiblemente habrá cambio con respecto a esta estructura.

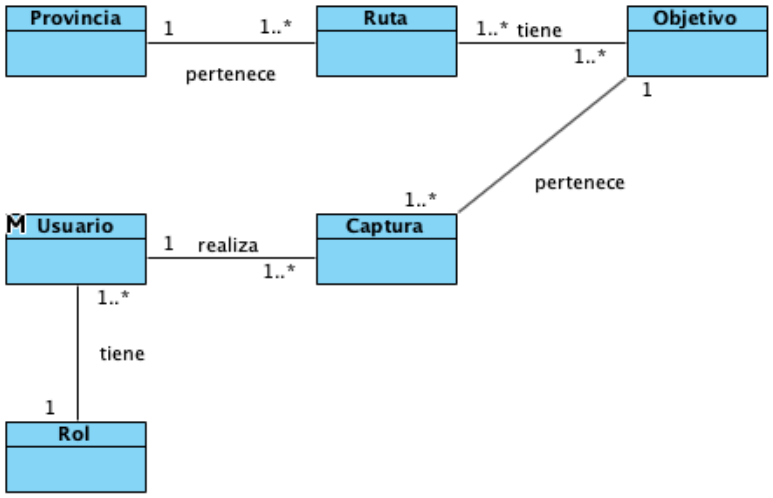


Ilustración 15 Modelo del dominio

3. Diseño

A partir de la definición sobre el diseño '... proceso de aplicar distintas técnicas y principios para definir un dispositivo, proceso o sistema con suficientes detalles como para permitir su realización física ' [12], desde el punto de vista del dominio de la solución, anteriormente definido, podemos construir el sistema.

3.1 Persistencia

En nuestra aplicación usaremos una base de datos de tipo relacional, cómo lo es MySql, apoyándonos de JPA. Esta API nos ofrecerá facilidad para el acceso a los datos, evitando el uso de queries nativas y mapeos de entidades automáticos.

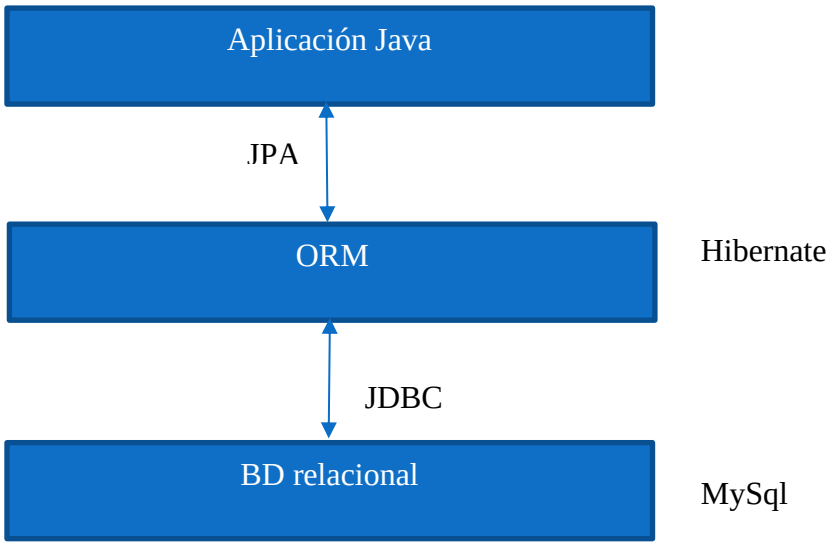


Ilustración 16 Esquema de comunicación a BBDD

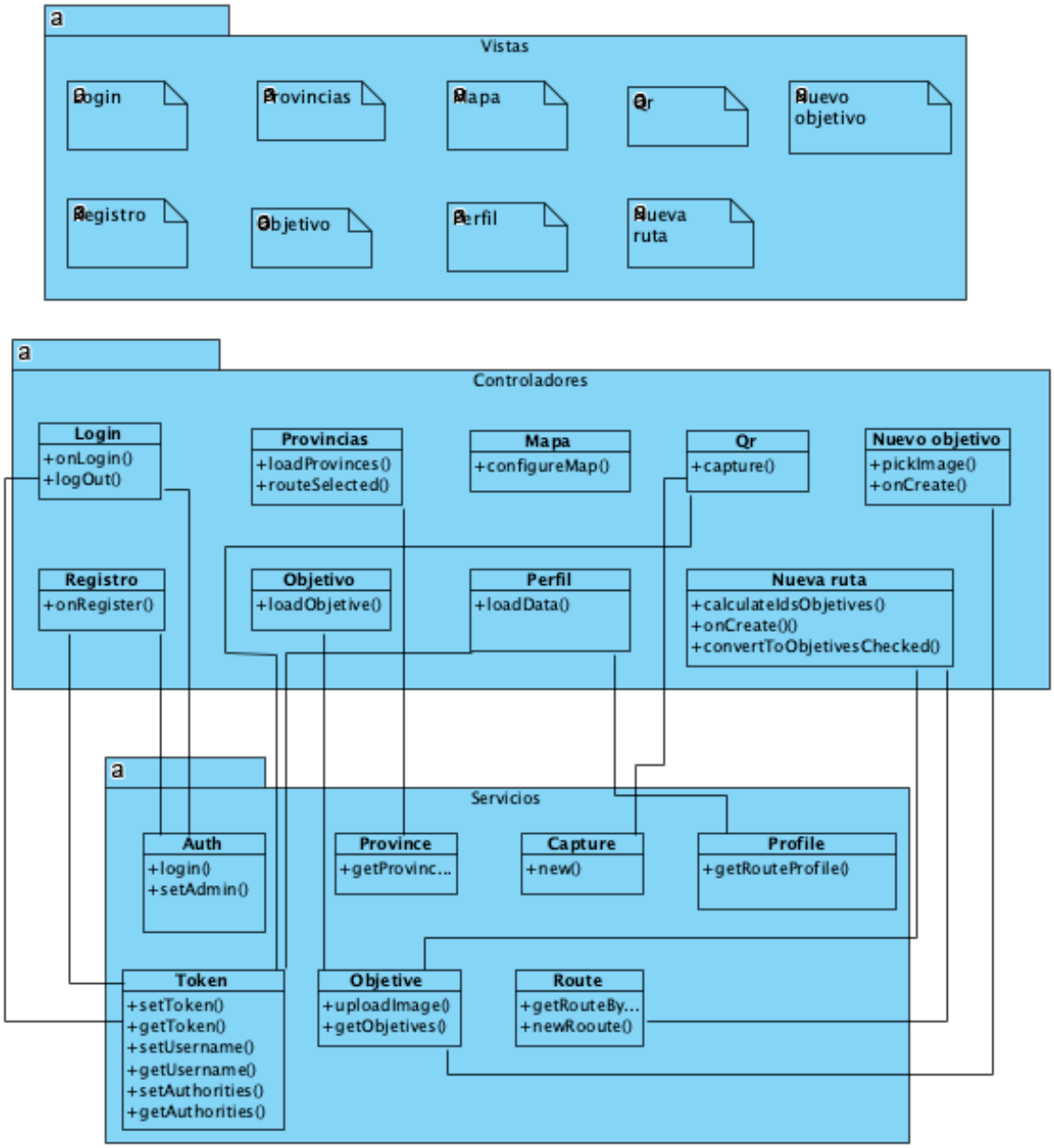
Spring es el encargado de construir este esquema a partir de las entidades anotadas como persistentes y de sus relaciones, indicadas con anotaciones.

3.2 Diagrama de clases

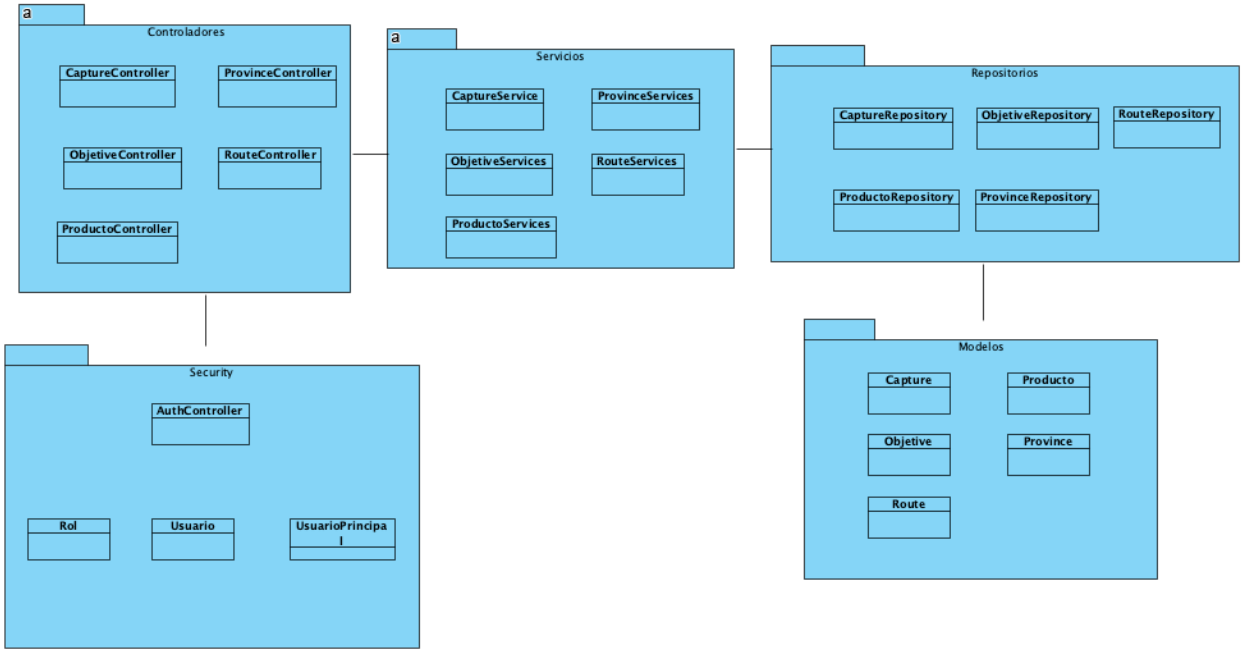
A continuación, se estudiará el diagrama de clases de la aplicación, donde se podrá ver la estructura del sistema compuesta en paquetes y clases, con sus atributos, operaciones o métodos y las relaciones.

Cabe destacar que se estudiará el diagrama de clases del cliente y el diagrama de clases del servidor por separado. Debido a que son aplicaciones que se ejecutan en equipos distintos y con lenguajes diferentes.

Aplicación Front



Aplicación Back



3.3 Diagramas de secuencia

Los diagramas de secuencia nos permiten observar la interacción de los objetos a través del tiempo. Es una manera clara de ver la comunicación entre los objetos para realizar una funcionalidad.

A continuación se detallará los diagramas de secuencia con más complejidad, desde el punto de vista del front, aplicación en la que interacciona el usuario y del back que es el que proporciona el API REST.

Seguridad en el lado backend

Cada petición a la API REST va acompañada del token del usuario. Este token es enviado en el encabezado http Authorization cada vez que la aplicación hace una solicitud HTTP.

Inicialmente JWTFilter intercepta la petición y capta el token, solicitando a TokenProvider si es válido el token. Este extrae la información de nick del token y

mediante UsuarioDAO se accede a la BD y se comprueba si el usuario existe en el sistema.

Finalmente, si es válido se permite acceder a la petición del controlador de recurso o si es inválido se devuelve un código de error.

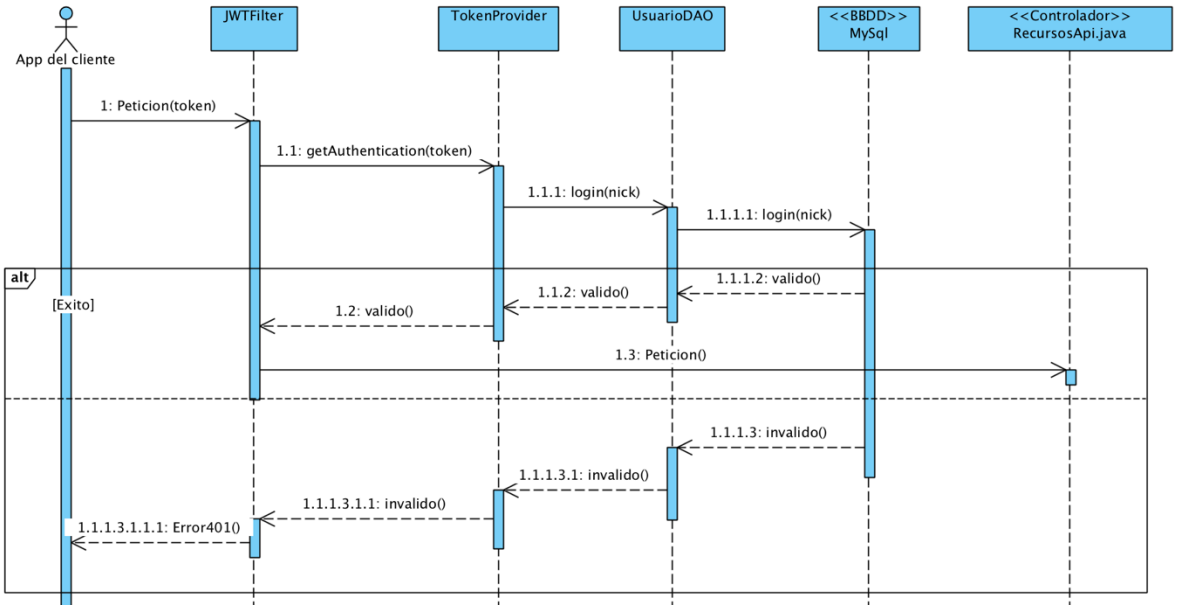


Ilustración 17 Diagrama de secuencia seguridad back

Identificación de usuario front

Cuando el usuario realice la petición de login, se solicita al controlador de login la vista y este se la proporciona. Una vez que el usuario rellene los datos, los envía al controlador de login y este se los manda al servicio que se encarga de la comunicación mediante una petición POST al servidor para realizar la autentificación.

El servicio envía los datos a la API a la espera de una respuesta. Si la respuesta es válida, se devuelve el token al usuario y se redirige a la ventana de home.

El token es una secuencia de cadena de caracteres, que en nuestra aplicación tiene un significado coherente. Está basado en el estándar JWT [13], compuesto en tres partes: encabezado, carga útil y firma.

El token se enviará al servidor para cada petición que requiera validación, para comprobar si el cliente está identificado en el sistema y tiene los permisos

requeridos. El token es una secuencia de cadena de caracteres, que en nuestra aplicación tiene un significado coherente. Está basado en el estándar JWT, compuesto en tres partes: encabezado, carga útil y firma.

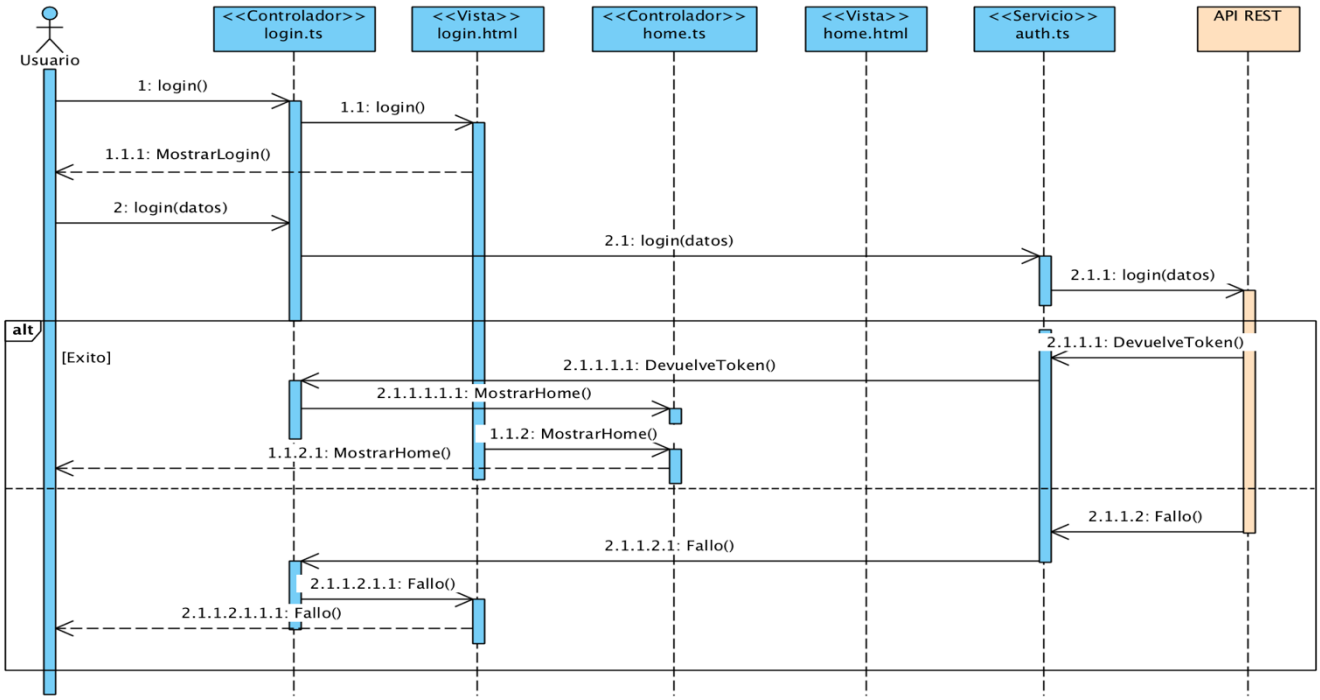


Ilustración 18 Diagrama de secuencia de login, front

En el servidor, cuando recibe una petición con token, puede obtener desde el token el Nick del usuario, esto nos será útil para saber en cada momento qué usuario está realizando cada funcionalidad.

En el servidor, se recibe los datos de *login* en el controlador de la API, y este hace uso del *DAO* de usuarios para que este acceda a la base de datos para comprobar que el usuario es válido.

Finalmente, si el usuario es válido se devuelve el *token* como hemos visto anteriormente, y si no se devuelve un error.

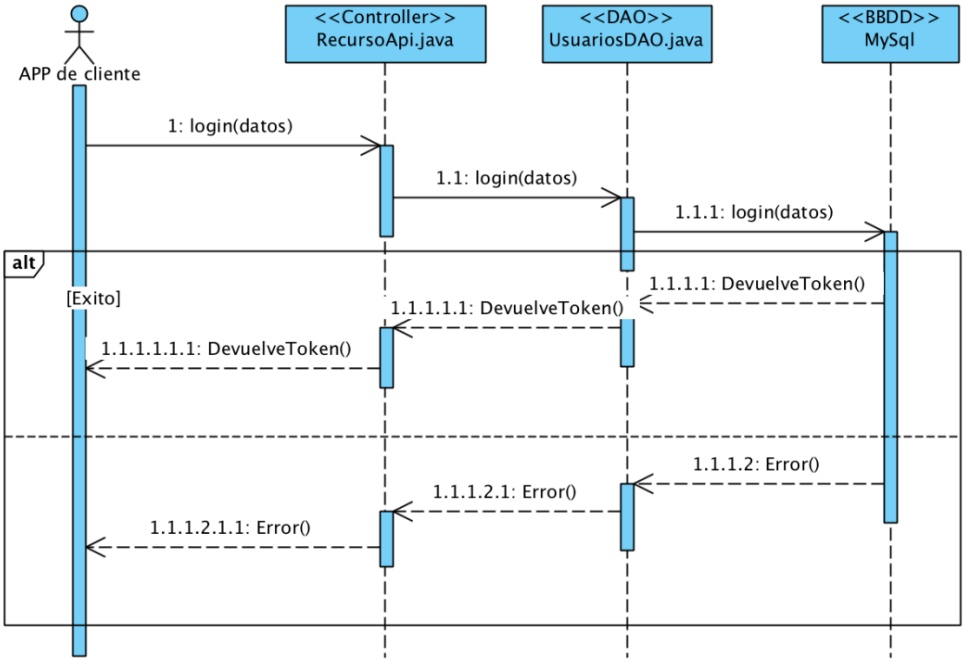


Ilustración 19 Diagrama de secuencia login, back

Registro de usuario

Inicialmente, el usuario solicita al controlador de login que le muestre la vista de login y este la recibe. Seguidamente, cuando pulse en el botón de registro, el controlador de login recibe la solicitud y le muestra al usuario el formulario de registro. Una vez que el usuario rellene los datos lo envía al controlador de (Signup) y este los envía al servicio llamado auth.ts que se comunica con la API.

El servidor devuelve una respuesta de éxito o fracaso y se muestra al cliente la respectiva vista.

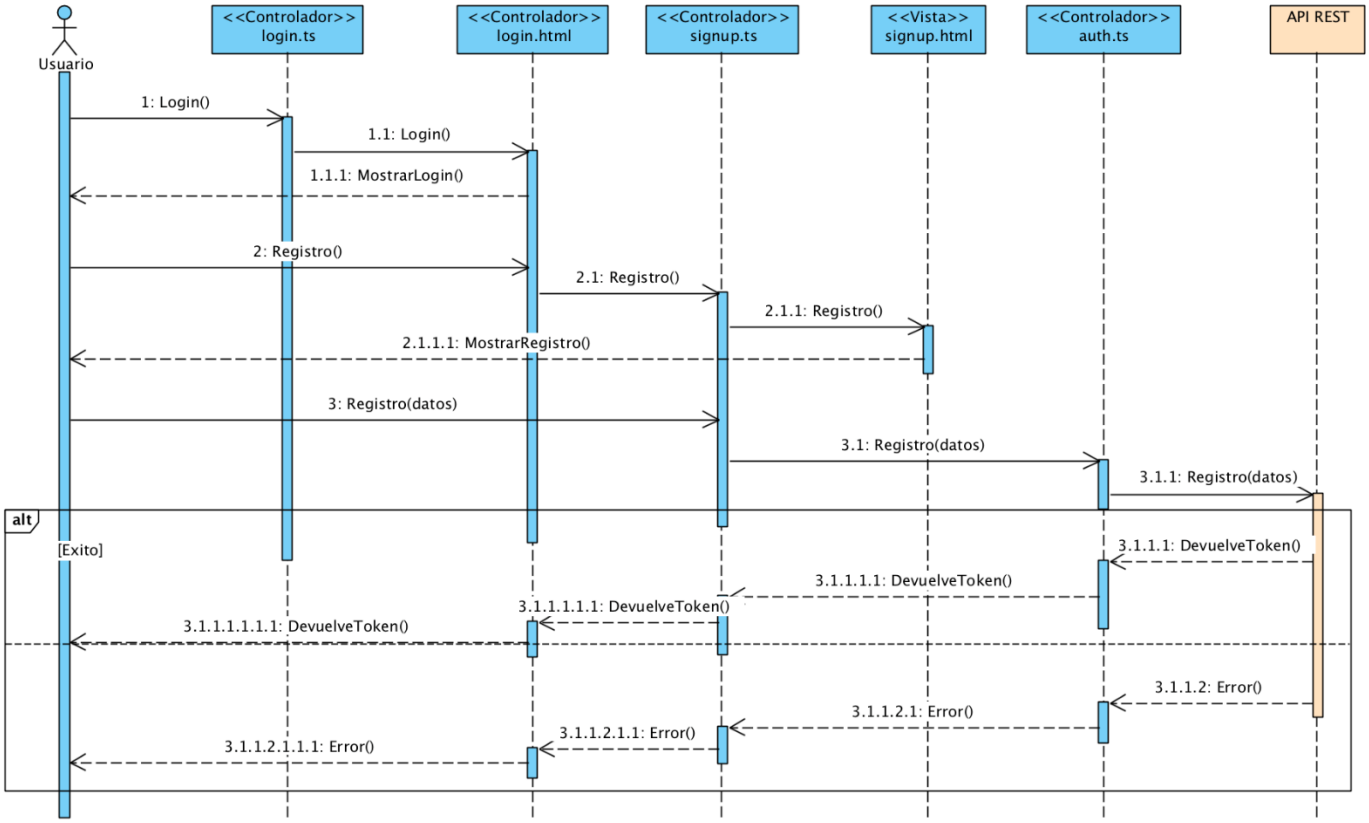


Ilustración 20 Registro en el sistema, front

El registro por parte del servidor comienza recibiendo del cliente los datos mediante la API, seguidamente utiliza el DAO de usuarios para comprobar que no existe ningún usuario con dicho *nick*, y devuelve el *token* si es válido y vacío si no lo es.

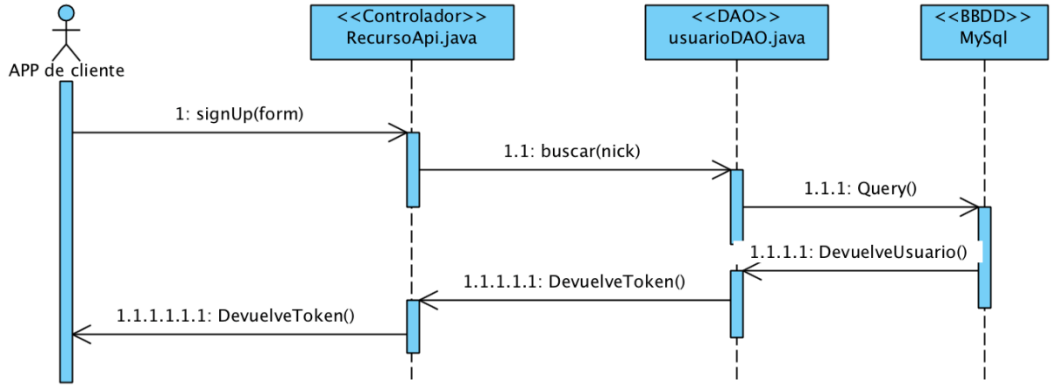


Ilustración 21 Diagrama de secuencia de registro sistema , back

Realizar captura QR

El usuario solicita al controlador de qr la vista, y este se la devuelve. El usuario interacciona intentando realizar la captura y es el controlador qr quien se encarga de comprobar si esta logeado el usuario para poder realizar esta funcionalidad mediante tokenService. Si está logeado, el controlador comunica al API la nueva captura del usuario, en caso contrario redirige al usuario a la ventana de login.

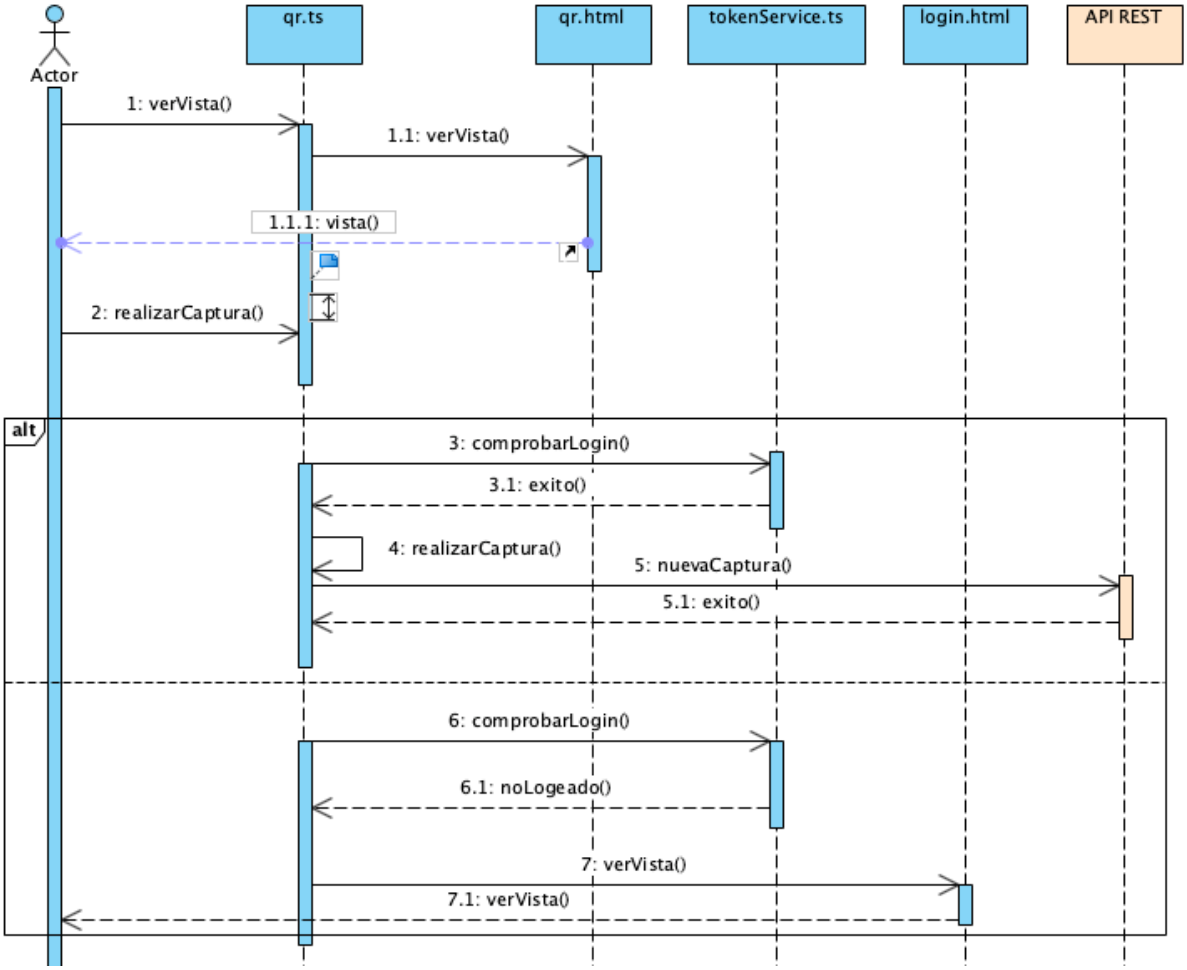


Ilustración 22 Realizar captura QR, front

El servidor, mediante el controlador recibe los datos de la petición, y llama al servicio donde se encuentra la lógica de negocio, éste a su vez llama al DAO para la inserción de la nueva captura en base de datos. Si la inserción ha ido bien devuelve un mensaje de éxito y en caso contrario de error.

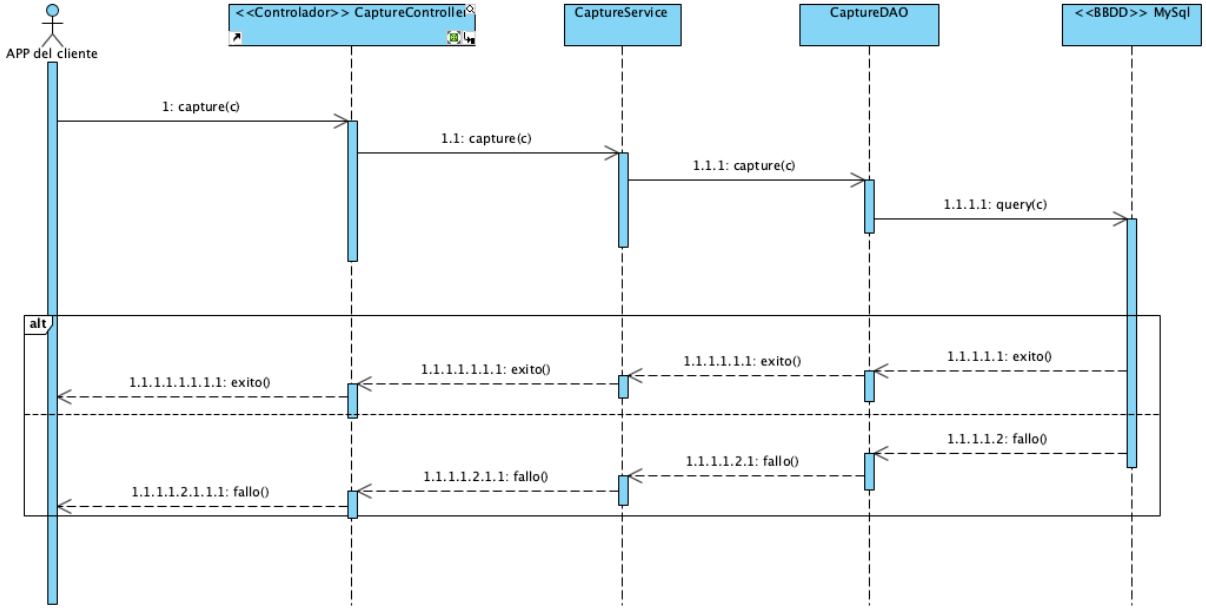


Ilustración 23 Realizar captura QR, back

3.4 Diseño de la interfaz

En este apartado mostraremos en primer lugar un diagrama de estados de la aplicación cliente, para observar los distintos estados y transiciones de la aplicación cliente. [14]

Además se mostrará los bocetos de las interfaces usuario y administrador.

3.4.1. Diagrama de estados de la aplicación cliente

El siguiente diagrama muestra el flujo de ejecución del cliente para facilitar su comprensión. Cada estado tiene un identificador, para ser usado en el siguiente apartado y ver que vista corresponde con cada estado.

El usuario puede acceder sin estar logeado a ver las provincias que existen para filtrar las rutas y la información de la ruta, en otro caso debe iniciar sesión. Con la sesión iniciada podrá capturar códigos qr, ver la información de su perfil, y ver la ubicación en el mapa de los objetivos.

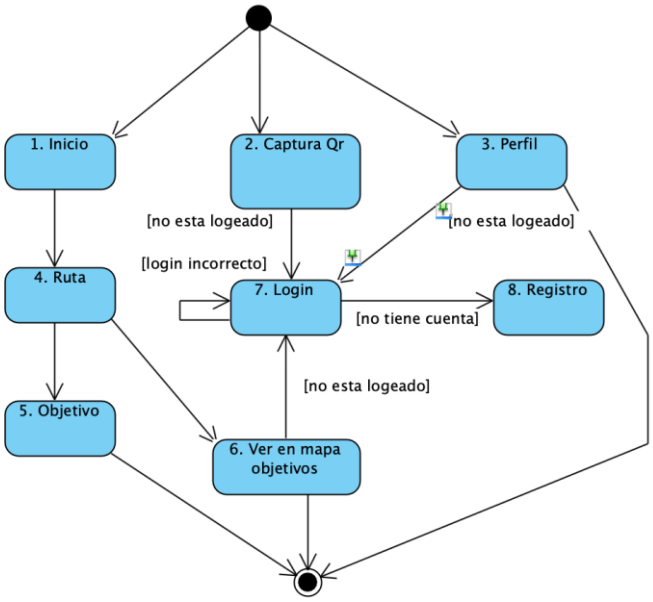


Ilustración 24 Diagrama de estados, cliente

El perfil de administrador, debe iniciar sesión para que mediante su rol pueda realizar las funcionalidades de nueva ruta y nuevo objetivo. Además, el administrador podrá navegar por las rutas y objetivos y ver las ubicaciones en el mapa.

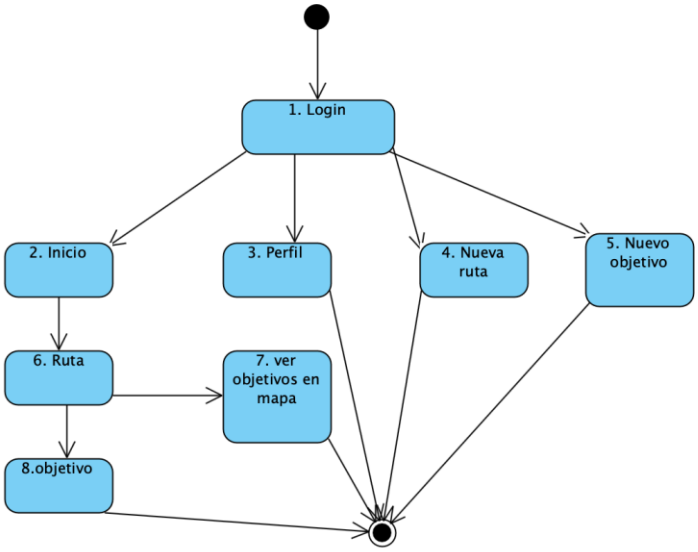


Ilustración 25 Diagrama de estados, admin

Interfaces de usuario

En el siguiente boceto correspondiente al ID 2 del diagrama de estados, el usuario puede filtrar las rutas por provincias y leer una pequeña información sobre ellas.

Ilustración 26 Interfaz de inicio

El siguiente boceto corresponde con la captura de código Qr.

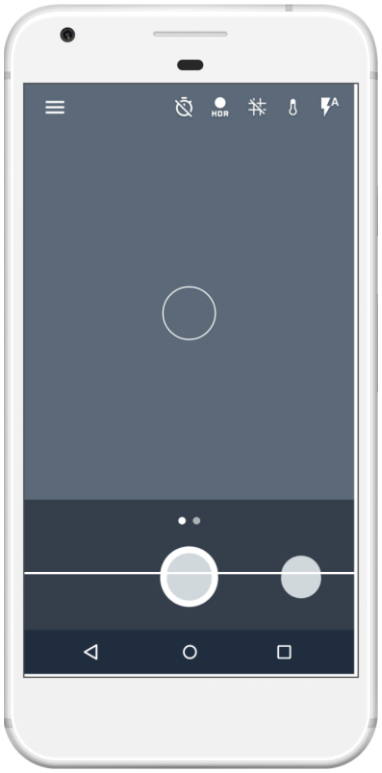


Ilustración 27 Interfaz captura QR

El siguiente boceto pertenece al ID 3 del diagrama de estado del cliente, se puede observar el histórico de capturas y la fecha en que se realizó.

Ilustración 28 Interfaz perfil de usuario

En el siguiente boceto se puede muestra la venta de ruta con los objetivos que contiene, descripción, imagen.

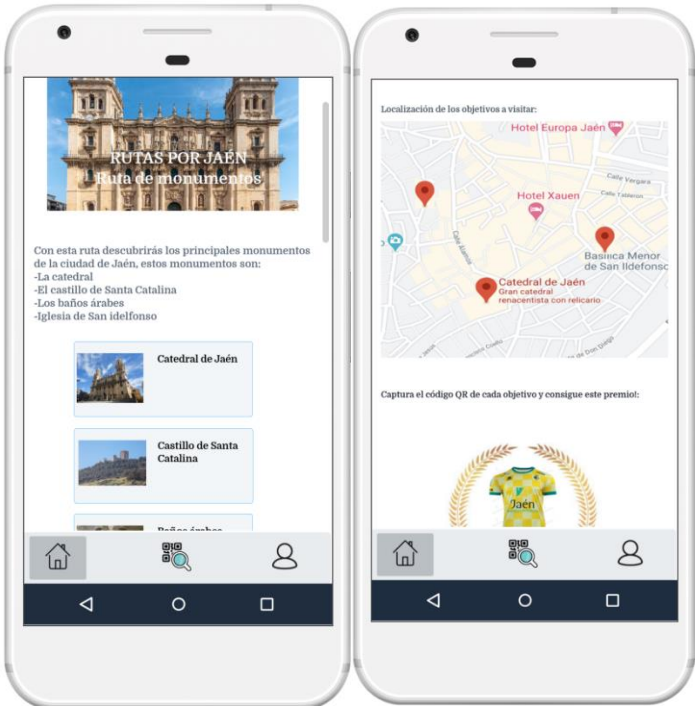


Ilustración 29 Interfaz de Información de ruta

En el siguiente boceto se puede observar la interfaz del detalle del objetivo, en el que se muestra el nombre, imagen, descripción, enlace a Wikipedia y ubicación.

Ilustración 30 Boceto detalle de objetivo

A continuación se muestran los bocetos de iniciar sesión y crear cuenta de usuario, correspondientes al ID 3 y 8 del diagrama de estados del cliente. Para poder realizar el login, debe introducir dos variables de texto,su *nick* y su contraseña. Si los datos introducidos son válidos, se accede al sistema con el rol correspondiente a la cuenta (usuario o administrador) y se mostrará el menú de navegación, correspondiente al a cada uno.

La interfaz de registro contiene varios campos a rellenar por el usuario, estos son: nombre, correo, *nick* y contraseña.

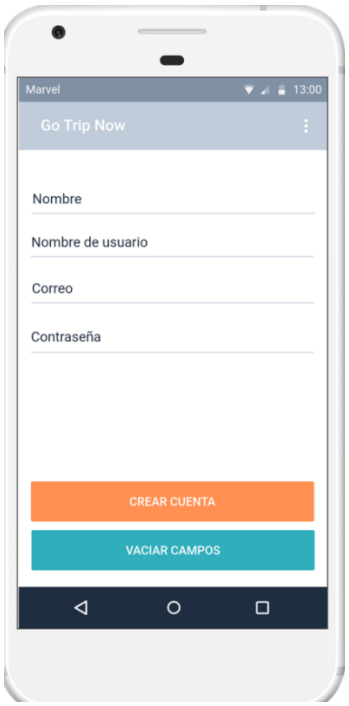


Ilustración 32 Boceto de iniciar sesión

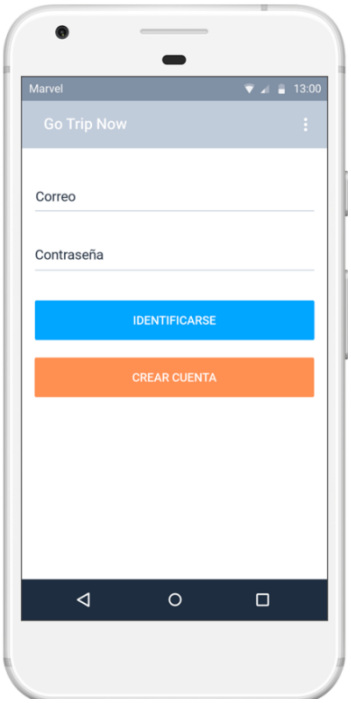


Ilustración 31 Boceto registro de usuario

Interfaces de administrador

El siguiente boceto corresponde con el diagrama de estados ID 4, en el cual, el administrador puede crear una nueva ruta, indicando nombre, descripción y los objetivos que incluye.

Ilustración 33 Interfaz nueva ruta

El siguiente boceto corresponde con el id 5 del diagrama de estados del administrador, en el que puede crear nuevos objetivos, indicando el nombre, imagen, descripción y ubicación.

Ilustración 34 Interfaz Nuevo objetivo

4. Implementación

En este punto se explicará la arquitectura de la aplicación y los componentes más relevantes que se han usado para llevar a cabo la aplicación ayudándonos de fragmentos de código.

4.1 Arquitectura de la aplicación

Nuestra aplicación usa una arquitectura cliente-servidor, el servidor proporciona recursos y estos son solicitados por el cliente. Es decir, el cliente realiza peticiones y estas son respondidas por el servidor.

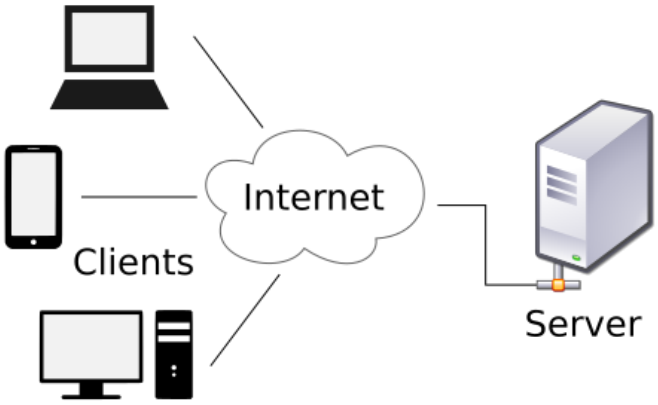


Ilustración 35 Diagrama cliente servidor

Nuestro servidor para atender las peticiones del cliente, proporciona una API [15].

Una API REST se puede definir como una interfaz entre sistemas que utilizan el protocolo HTTP [16] para obtener datos o generar operaciones sobre esos datos en distintos formatos como JSON o XML.

Por lo tanto, siguiendo esta arquitectura, la siguiente imagen muestra cómo estaría organizada la aplicación en base a los *frameworks* y tecnologías usados.

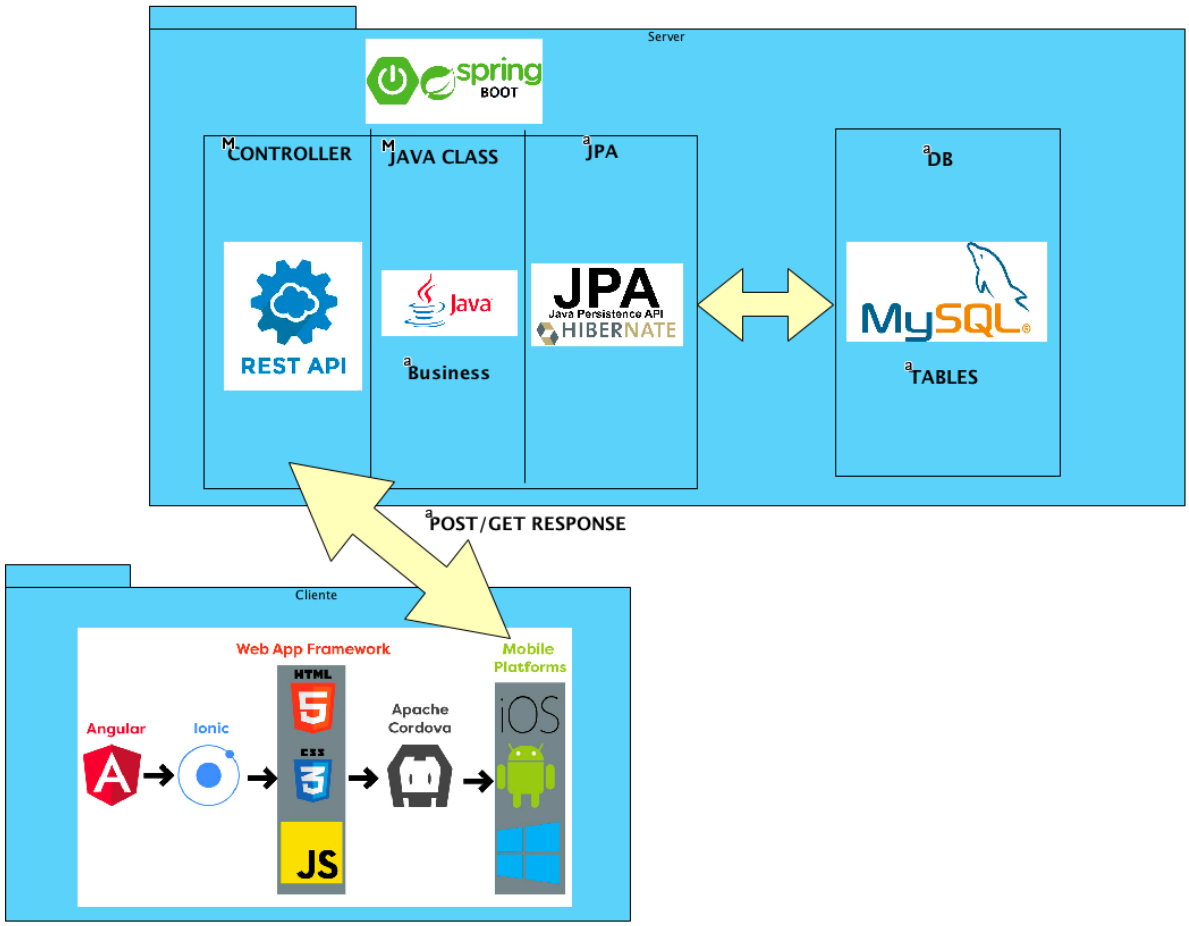


Ilustración 36 Diagrama arquitectónico del sistema

En el lado del cliente, se muestra como ionic está basado en Angular y como *Ionic* hace uso de HTML, CSS y JS. Después *Ionic* hace uso de Apache *Cordova* para la compatibilidad con el hardware . Éstas pueden ser *Android*, *iOS* o *Windows Phone*.

Con la aplicación móvil generada previamente con *Apache Cordova*[3], mediante peticiones POST/GET se comunica con la REST API proporcionada por el servidor.

Las peticiones recibidas en el servidor, hacen uso de clases de Java, que son del modelo de negocio, y éstas a su vez si necesitan acceder a la base de datos lo hacen mediante JPA.

[3] https://cordova.apache.org/

4.2 Detalles sobre la implementación

A continuación, se describirá una lista de las tecnologías empleadas durante la implementación:

- Ionic v6.16
- Angular core 12.0.1
- HTML 5
- CSS 3
- Apache Cordova
- GIT 2.6.2
- Spring Boot 2.0
- Postman 5.5.2
- MySql Server 8.0.25

4.3 Aplicación servidor

4.3.1. GIT

En el desarrollo de la aplicación back, se ha utilizado un sistema de control de versiones, concretamente GIT. De esta manera podemos recuperar versiones anteriores de código en caso de necesitarlo.

En la siguiente imagen se observa un extracto del histórico de commits.

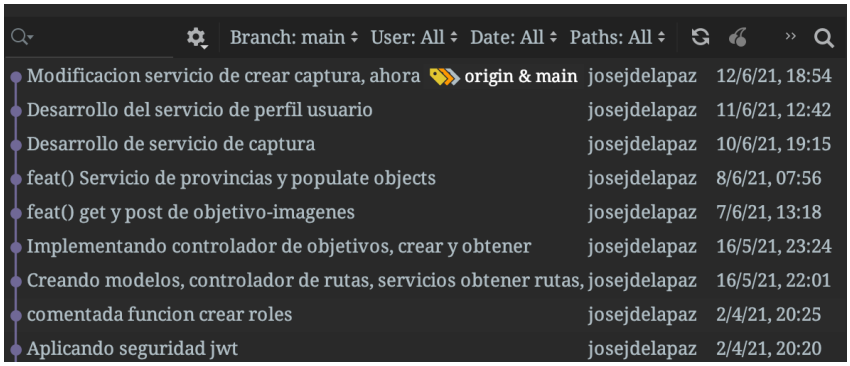


Ilustración 37 histórico commits back

Mapeo ORM de clase con anotaciones JPA

Dado que el mapeado de una clase a una tabla no es automático, tenemos que proporcionar a JPA una mínima información, mediante anotaciones en código Java.

En la siguiente imagen se puede observar un ejemplo del uso de anotaciones en la clase Categoría.

Con la anotación *@Entity* indicamos que la clase Objetive es una clase persistente, que en JPA se denomina entidad.

Con la anotación *@Id* marcamos el atributo que hace de clave primaria de la tabla, en este caso es autogenerado, marcado con la etiqueta *@GeneratedValue*.

Ilustración 38 Anotaciones JPA

Acceso a la BBDD desde DAO

Cuando un servicio necesita acceder a la base de datos, lo hace mediante el DAO correspondiente. Concretamente se ha utilizado la interfaz JPARepository [17] que nos proporciona operaciones básicas para realizar consultas evitando así tener que desarrollar queries nativas, ahorrando código y mejorando el entendimiento de éste.

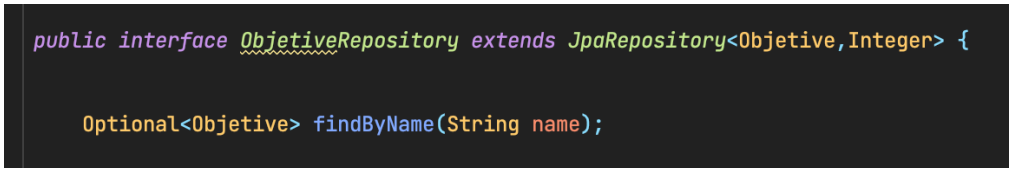


Ilustración 39 Acceso a BBDD

Recursos API

El servidor proporciona una API REST con distintos servicios para atender las peticiones del cliente.

Por ejemplo, en la siguiente imagen se puede ver la implementación de la función del controlador que realiza la lógica de un endpoint de tipo GET, que realiza la búsqueda de las provincias, devolviendo una lista de provincias.

```java
@RestController
@RequestMapping("/provinces")
@CrossOrigin(origins = "*")
public class ProvinceController {

    @Autowired
    private ProvinceService provinceService;

    @GetMapping()
    ResponseEntity<List<ApiProvince>> getProvinces(){
        return new ResponseEntity(ProvinceMapper.toApiProvince(provinceService.getProvinces()), HttpStatus.OK);
    }
}
```

Ilustración 40 Endpoint tipo GET

Tabla de end-point REST

En la siguiente tabla se detalla los endpoints del servidor, detallando la url, tipo y respuestas.

Tipo	Detalle	URL	RequestBody	Devuelve	Error
POST	Crear captura	http://192.168.151.146:8080/capture	ApiCapture	200{Message}	404
GET	Obtener objetivos	http://192.168.151.146:8080/objetives	Null	200{List<Objetive>}	404
POST	Crear objetivo	http://192.168.151.146:8080/objetives	Objetive	200 {Int}	404
POST	Crear imagen de un objetivo	http://192.168.151.146:8080/objetives/{id} /image/create	MultipartFile	200	404
GET	Obtener imagen de un objetivo	http://192.168.151.146:8080/objetives/{id} /image	Null	200 {Resource}	404
GET	Obtener provincias	http://192.168.151.146:8080/provinces	Null	200 {List<ApiProvinces>}	404
GET	Obtener rutas	http://192.168.151.146:8080/route	Null	200 {List<Route>}	404
GET	Obtener ruta	http://192.168.151.146:8080/route/{id}	Null	200 {Route}	404
POST	Crear ruta	http://192.168.151.146:8080/route	ApiRouteNew	200	404
GET	Obtener capturas de un usuario	http://192.168.151.146:8080/captures	Null	200	404
POST	Crear usuario	http://192.168.151.146:8080/auth/nuevo	NuevoUsuario	200	404
POST	Login de usuario	http://192.168.151.146:8080/auth/login	LoginUsuario	JwtDto	404

Ilustración 41 Tabla endpoints

Seguridad token JWT

Con la utilización del *token* en forma *jwt*, podemos extraer del *token* el *Nick* del cliente. Esto nos proporciona en cada petición la posibilidad de comprobar si tiene permiso de acceso, proporcionando seguridad a la aplicación.

JWT [18] (*Json Web Token*) es un estándar para crear *tokens* de acceso. El JWT está formado por 3 partes, encabezado, carga útil y una firma. Para crear el *token*, los datos necesarios para crear el *token* son: la palabra secreta proporcionada desde el archivo *applycation.yml*, el tiempo que va a ser válido el *token* y para la firma el algoritmo (HMAC con SHA-512)4.

```
public String generateToken(Authentication authentication){
    UsuarioPrincipal usuarioPrincipal = (UsuarioPrincipal) authentication.getPrincipal();
    return Jwts.builder().setSubject(usuarioPrincipal.getUsername())
            .setIssuedAt(new Date())
            .setExpiration(new Date(new Date().getTime() + expiration * 1000))
            .signWith(SignatureAlgorithm.HS512, secret)
            .compact();
}
```

Ilustración 42 generación de token

Además, se filtran las solicitudes entrantes usando el *token* creado y *Spring security*. En la siguiente imagen se puede observar cómo se comprueba la validación del *token* y si el usuario existe en la base de datos, además se devuelve una serie de excepciones cuando falla la validación.

```
@Override
protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain filterChain) throws ServletException, IOException {
    try {
        String token = getToken(req);
        if(token != null && jwtProvider.validateToken(token)){
            String nombreUsuario = jwtProvider.getNombreUsuarioFromToken(token);
            UserDetails userDetails = userDetailsService.loadUserByUsername(nombreUsuario);

            UsernamePasswordAuthenticationToken auth =
                    new UsernamePasswordAuthenticationToken(userDetails,  credentials: null, userDetails.getAuthorities());
            SecurityContextHolder.getContext().setAuthentication(auth);
        }
    } catch (Exception e){
        logger.error("fail en el método doFilter " + e.getMessage());
    }
    filterChain.doFilter(req, res);
}
```

Ilustración 43 Filtro de solicitudes entrantes

Las peticiones que necesitan token se configuran en la siguiente captura. Quedando excluidas las peticiones /aut/* y /provinces.

[4] https://es.wikipedia.org/wiki/HMAC

```java
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.cors().and().csrf().disable() HttpSecurity
            .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
            .antMatchers( ...antPatterns: "/auth/**","/provinces/**").permitAll()
            .anyRequest().authenticated()
            .and() HttpSecurity
            .exceptionHandling().authenticationEntryPoint(jwtEntryPoint) ExceptionHandlingConfigurer<HttpSecurity>
            .and() HttpSecurity
            .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    http.addFilterBefore(jwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);
}
```

Ilustración 44 Excluyendo de seguridad auth y provinces

4.4 Aplicación cliente

4.4.1. GIT

En el desarrollo de la aplicación front, se ha utilizado un sistema control de versiones, concretamente github [19]. Esto nos ha proporcionado un respaldo de seguridad del código desarrollado. Los comandos que más se han utilizado son push, commit y pull.

En la siguiente captura se muestra un histórico de commits realizados:

```
TERMINAL  PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  2: git

commit 0b2f563e95d1b71574535cee1913c91fad6b5151 (HEAD -> master)
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Sat Sep 4 17:43:12 2021 +0200

    Mejoras front

commit 7fdf3388fdaf07ec2a37306037e13506ac2c6b4b
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Wed Sep 1 19:47:42 2021 +0200

    Desarrollando pagina de info objetivo

commit 3ec6f9aa174404fa888aceb4ed7b9d48ad6544b3
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Sun Aug 29 18:45:44 2021 +0200

    completando crear ruta, provincia

commit 9d091afd39104fd90f3f88142a74618406aaff9e
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Sun Aug 29 12:55:56 2021 +0200

    Formulario nueva ruta, control de checks y mostrar toast

commit d58dff308e9af67fbc582dd70939d258fb4d05ae
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Wed Jun 16 19:09:41 2021 +0200

    Implementando servicio seleccionar foto

commit 59e8678d6ea1813d4a28d2798f24d6b46df0ff77
Author: josejdelapaz <josejdlp@gmail.com>
Date:   Sun Jun 13 18:53:15 2021 +0200

    implementando servicio de crear objetivo
```

Ilustración 45 Histórico de commits front

4.4.2. Estructura ionic

Un proyecto ionic está formado por una jerarquía de carpetas y archivos similar a un proyecto nativo de Angular. A continuación, veremos los aspectos más relevantes.

En la carpeta *src* están todos los archivos con el contenido de nuestra aplicación. *Src* contiene la carpeta de app que es el módulo principal. En la carpeta app se incluyen todos los ficheros de código relacionados a las páginas, componentes, servicios, modelos, estilos etc.

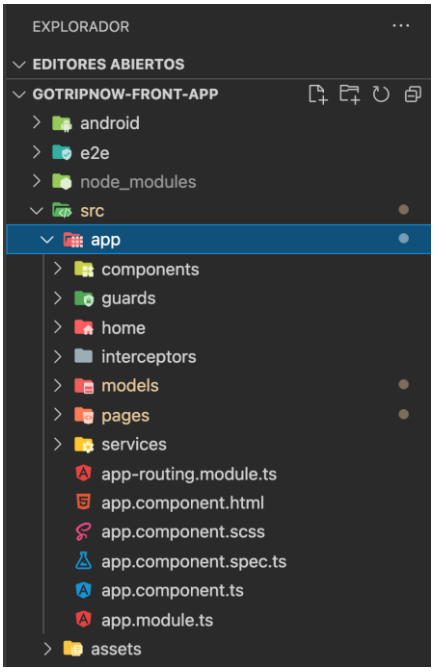


Ilustración 46 Estructura del proyecto ionic

En las siguientes capturas se muestra un ejemplo de una página, concretamente la de crear nueva ruta, está formada por el controlador RouteNewPage.ts y la vista RouteNewPage.html. Se puede observar la declaración y uso de funciones y cómo por ejemplo cuando una variable es modificada en la vista, simultáneamente se modifica en el controlador.

```
  styleUrls: ['./route-new.page.scss'],
})
export class RouteNewPage implements OnInit {

  objetives:Objetive[];
  name="";
  description="";
  provincia=0;
  objetivesChecked:ObjetivesChecked[]=new Array;
  objetivesToSend:number[]=new Array;
  provinces: Province[];

  constructor(private objetiveService: ObjetiveService
              private routeService:RouteService,
```

Ilustración 47 Controlador RouteNewPage.ts

```
<form #f="ngForm" (ngSubmit)="onCreate()" novalidate>
    <ion-list lines="full" class="ion-no-margin">
        <h5>Datos de la nueva ruta:</h5>
        <ion-item>
            <ion-label position="floating">Nombre de la ruta:
                <ion-text color="danger">*</ion-text>
            </ion-label>
            <ion-input name=name required type="text" [(ngModel)]="name" placeholder="...">
        </ion-item>
        <ion-item>
            <ion-label position="floating">Descripción de la ruta:
                <ion-text color="danger">*</ion-text>
            </ion-label>
            <ion-textarea name=description required type="text" [(ngModel)]="description" p
```

Ilustración 48 Vista RouteNewPage.html

4.4.3. Generar aplicación para varias plataformas

En este apartado veremos cómo se genera la aplicación [20] para Android o IOS.

Mediante el CLI [5]de IONIC, escribimos los siguientes comandos para agregar la plataforma Android o IOS al proyecto:

- ionic cordova platform add android
- ionic cordova platform add ios

Ahora para generar el archivo apk se escribe los siguientes comandos:

- ionic build android --release
- ionic build ios --release

El archivo apk[6] se encuentra en Plataformas \ Android \ Build \ Outputs \ apk

[5] https://ionicframework.com/docs/cli/

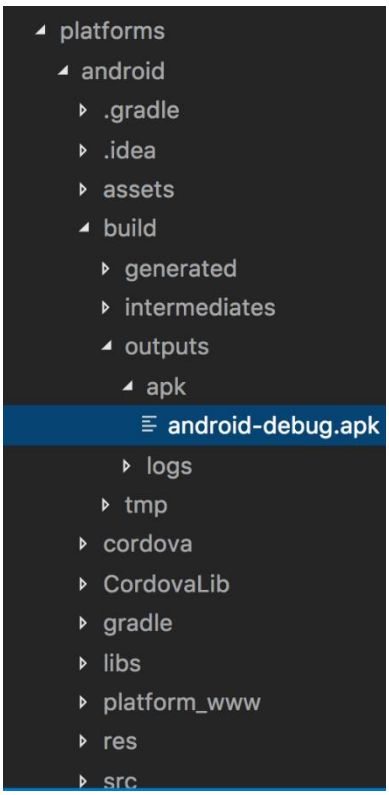


Ilustración 49 Directorio apk generado

Finalmente, para instalar el archivo .apk hay que seguir una serie de pasos [21]:

- Dar permiso al dispositivo móvil para poder instalar aplicaciones que no provengan de App Store. Se debe acceder Ajustes-Seguridad y habilitar la opción de “Orígenes desconocidos”.
- Conectar el dispositivo móvil mediante USB al ordenador y elegir en el dispositivo móvil la opción de “Transferir archivos”.
- Copiar el archivo desde el ordenador al dispositivo móvil.
- Finalmente extraer de forma segura el móvil y acceder desde el móvil a la carpeta donde ha sido copiada la aplicación y ejecutarla para instalarla.

[6] https://es.wikipedia.org/wiki/APK_(formato)

4.4.4. Ejemplo de un servicio

Los servicios de la aplicación cliente se encargan de comunicarse con el servidor mediante peticiones asíncronas. Estos servicios usan estructuras de datos para guardar la información, además de clases para el mapeo correcto de los objetos de la petición.

En la siguiente función se puede observar una petición GET y el resultado se envía como un Observable de tipo Route. Todo ello dentro de una promesa, para indicar que el objeto promesa contendrá un valor en el futuro que nos interesa y no podemos dejar que bloqueen la función.

```
export class RouteService {

  baseUrl='http://192.168.151.146:8080';
  //baseUrl= 'http://localhost:8080';

  routeUrl=this.baseUrl+'/route';

  constructor(private httpClient:HttpClient) {

  }

  public getRouteById(routeId:number):Observable<Route>{
    return this.httpClient.get<Route>(this.routeUrl+"/"+routeId);
  }
```

Ilustración 50 Servicio RouteService

4.4.5. Data Storage

Data Storage [22] nos permite almacenar información en el dispositivo, aunque se cierre la app, la información permanece. Unas de las características destacadas es que permite almacenar entre 5MB y 10MB, la información está almacenada en el dispositivo del cliente y no es enviada en cada petición al servidor.

Se ha empleado Local Storage para guardar en la memoria del dispositivo las variables necesarias para gestionar el sistema de autenticación, estas son el token, rol del usuario logeado y Nick del usuario logeado.

En la siguiente captura se puede observar, la ventana de Application-storage-localhost de la app cliente levantada en localhost, y el uso de la memoria de las tres variable mencionadas en el punto anterior.

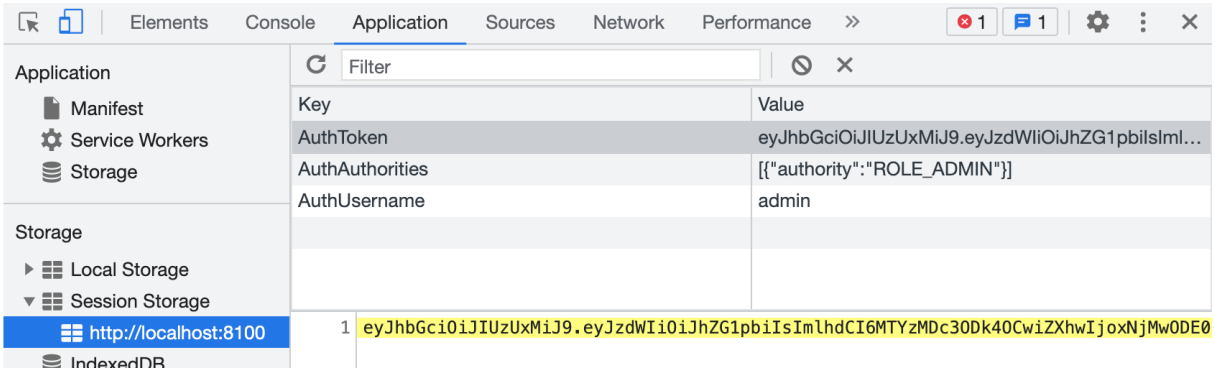


Ilustración 51 Local Storage

5. Pruebas

Se ha comprobado el correcto funcionamiento de las historias de usuario, después de cada sprint. Las historias de usuario fueron validadas realizando pruebas funcionales, realizando todo el flujo que haría el usuario con ese caso de uso y verificando en cada caso que se realiza la operación esperada y mostrando los mensajes de error en caso de ser necesarios.

Además, para comprobar el funcionamiento de la interfaz REST se ha utilizado "Postman"[7]. Enviando peticiones GET o POST al servidor con los datos requeridos analizando los resultados y comprobando si la respuesta del servidor era correcta.

Finalmente, surgieron ideas nuevas en las que se añadirían en futuras ampliaciones como que el cliente pueda pagar desde la aplicación con la tarjeta de crédito directamente. Estas ideas son expuestas en el apartado llamado "Trabajos futuros".

6. Conclusiones

En la actualidad podemos observar que uno de los motivos del desarrollo tecnológico es el de dar servicios a la sociedad. Automatizar procesos o realizar cálculos proporciona a la persona ahorro de tiempo, eficiencia y fiabilidad.

Además, año tras año, el uso de la tecnología para realizar tareas habituales es más común y esto abre la posibilidad de un avance tecnológico y nuevas aplicaciones orientadas a dar servicio a las personas.

[7] https://www.getpostman.com/

Es por ello que esta aplicación proporciona la posibilidad de gamificar distintos itinerarios culturales y turísticos creando diferentes rutas que hacen mas accesible la cultura a las personas, pudiendo añadir la información histórica o artística que deseemos a los monumentos de nuestras rutas.

En lo referente al desarrollo del proyecto, podemos concluir que se han alcanzado con éxito los objetivos inicialmente propuestos:

- Se ha desarrollado un estudio en el contexto donde se quiere instalar la aplicación, analizando su funcionamiento y extrayendo sus ventajas e inconvenientes.
- Se ha diseñado un sistema informático, definiendo la funcionalidad del sistema mediante historias de usuario y se ha estudiado la viabilidad temporal y económica del desarrollo. Siguiendo una metodología ágil, SCRUM.
- Se han realizado implementaciones utilizando tecnologías actuales como Angular, CSS, HTML5 (*Ionic)* en la aplicación del cliente; y Spring Boot para la aplicación del servidor.
- Y, por último, se han probado los resultados obtenidos siguiendo los criterios de satisfacción, adaptándose la planificación a los imprevistos surgidos.

Desde un punto de vista personal, al realizar este proyecto he podido ser consciente de la importancia del trabajo al que me dedicaré, debido a que muchos negocios, empresas y servicios públicos se basan en software y sus beneficios económicos pueden variar mucho dependiendo de la calidad de dicho software.

Al inicio del desarrollo del proyecto me enfrenté a una tecnología nueva para mí como es *Ionic*, no obstante, a las dificultades que encontré fui resolviéndolas a través de foros especializados en esta tecnología. Después, con el conocimiento adquirido pude avanzar con mayor eficacia en el proyecto.

Desde un punto de vista técnico, el uso de JPA para crear las tablas en la base de datos de forma automática me ha servido para conocer mejor su funcionamiento y las utilidades en lo que lo puedo emplear. Otro de los aspectos que he aprendido es

el de los beneficios que tiene las aplicaciones híbridas porque proporciona una aplicación multiplataforma, que puede usar funcionalidades nativas del móvil y con una interfaz muy conseguida.

También he valorado mucho el uso de sistema control de versiones GIT porque hubo veces que tuve que regresar a una versión del proyecto anterior para solventar errores y el empleo de esta tecnología me facilitó mucho la tarea.

Creo que este trabajo me servirá en el futuro porque sigue una arquitectura muy común en muchos proyectos y he usado tecnologías muy punteras en el mercado, aparte de que me ayudará en mi currículum profesional.

7. Trabajos futuros

En este apartado voy a comentar brevemente las historias de usuario que se descartaron en la planificación inicial y serian unas buenas candidatas para incluirlas en una futura planificación. Así como otras nuevas o mejoras de las existentes que han ido surgiendo a lo largo de la realización del proyecto.

Principalmente si se quisiera ampliar el proyecto, las historias de usuario esenciales son:

- Ranking de usuarios.
- Asignar premio a la ruta por el administrador.
- Ver premio de la ruta por el usuario.
- Bonificación por ruta conseguida.
- Sistema de comentarios en las rutas por los usuarios.
- Sistema de favoritos.

Por otro lado, las mejoras y nuevas funcionalidades podrían ser:

- Incluir un video explicativo de cada uno de los monumentos incluidos en la ruta, de este modo se podría dar una información histórica o artística de calidad al usuario de una forma amena.

- Mostrar notificaciones de las nuevas rutas creadas para así llamar la atención de los usuarios de la app.
- Valoración de las rutas por parte de los usuarios, esta mejora nos da la posibilidad de crear un ranking de las rutas mejores valoradas.

En un futuro si se implementara la aplicación en varios municipios, se podría añadir un sistema de geolocalización de rutas más cercanas.

Con todas estas ideas podría desarrollarse una aplicación muy completa y con mucha probabilidad de éxito en el mercado.

Con el diseño, arquitectura y tecnologías utilizadas en el proyecto, no sería difícil integrar estas mejoras, porque en los elementos de la aplicación se ha intentado que tengan alta cohesión y bajo acoplamiento; además de estar basada en una arquitectura modelo-vista-controlador.

8. Bibliografía

[1] «zonamovilidad.es,» [En línea]. Available: https://www.zonamovilidad.es/el-uso-del-telefono-movil-aumenta-un-90-en-los-ultimos-10-anos-a-nivel-mundial.

[2] «El Pais,» [En línea]. Available: https://elpais.com/espana/en-clave-de-bienestar/2020-10-22/el-turismo-del-futuro.html.

[3] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil.

[4] C. L. Gómez, « Scrum, Kanban, Lean,,» de *Métodos Ágiles*, Anaya, 2017.

[5] P. González, *Apuntes de la asignatura Desarrollo Ágil*, 2017.

[6] «Equipo Altran,» [En línea]. Available: http://equipo.altran.es/metodologias-agiles-vs-metodologias-tradicionales/.

[7] P. González, Apuntes de la asignatura Desarrollo Ágil, 2017.

[8] J. Sutherland, Scrum: El nuevo y revolucionario modelo organizativo que cambiará tu vida, 2015.

[9] «Atlassian,» [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories.

[10] «itdo,» [En línea]. Available: https://www.itdo.com/blog/moscow-que-es-y-como-priorizar-en-el-desarrollo-de-tu-aplicacion/.

[11] B. 1. d. m. d. 2020, «boe,» [En línea]. Available: https://www.boe.es/boe/dias/2020/05/13/pdfs/BOE-A-2020-5006.pdf.

[12] E.S.Taylor, «An Interim Report on Engineering Design,» *Massachusetts Institute of Technology*, 1959.

[13] «JWT,» [En línea]. Available: https://jwt.io/.

[14] M. CiIlero, «Manuel Cillero,» [En línea]. Available: https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-transicion-de-estados/.

[15] «Wikipedia, API rest,» [En línea]. Available: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional.

[16] «Wikipedia HTTP,» [En línea]. Available:

https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto.

[17] «Jpa Repository,» [En línea]. Available: https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html.

[18] «Jwt,» [En línea]. Available: https://jwt.io/.

[19] «Git hub,» [En línea]. Available: https://github.com/.

[20] «Generar app android e ios,» [En línea]. Available: https://ionicframework.com/docs/developing/ios.

[21] «Ionic. Pasos para instalar app disp. móvil,» [En línea]. Available: https://ionicframework.com/docs/developing/android.

[22] «Data Storage Ionic,» [En línea]. Available: https://ionicframework.com/docs/vue/storage.

[23] A. Á. G. R. d. l. H. d. D. Carmen Lasa Gómez, Métodos Ágiles. Scrum, Kanban, Lean, Anaya, 2017.

[24] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Manifiesto_%C3%A1gil..

[25] «Equipo Altran,» [En línea]. Available: http://equipo.altran.es/metodologias-agiles-vs-metodologias-tradicionales/.

[26] «Auth0,» [En línea]. Available: https://auth0.com/blog/alternatives-to-native-mobile-app-development/.

[27] M. Cohn, User stories applied: For Agile Software Development, Addison Wesley, 2004.

[28] «El laboratorio de las TI,» [En línea]. Available: http://www.laboratorioti.com/2016/09/26/tecnica-priorizacion-moscow/.

[29] S. M. y. R. F. S. Bennett, Análisis y diseño orientado a objetos de sistemas, 3a edición, McGraw-Hill, 2006.

[30] V. Mihalcea, High-Performance Java Persistence, 2016.

[31] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Diagrama_de_secuencia.

[32] «Esacademic,» [En línea]. Available: http://www.esacademic.com/dic.nsf/eswiki/1155905.

[33] «WikiUml,» [En línea]. Available: https://wikiuml.wikispaces.com/Diagrama+de+Estados.

[34] E. Amodeo, Principios de Diseño de APIs REST.

[35] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto.

[36] «W3schools,» [En línea]. Available: https://www.w3schools.com/js/js_json_intro.asp.

[37] «W3schools,» [En línea]. Available: https://www.w3schools.com/xml/.

[38] «Tablas salariales grupo Ilunion,» 2017. [En línea]. Available: https://www.ccoo-servicios.es/archivos/ilunioncontactcenter/TablasSalariales2017.pdf.

[39] E. Revilla, Desarrollo de aplicaciones móviles multiplataforma con Ionic desde cero, 2017.

[40] «Ionic Framework,» [En línea]. Available: https://ionicframework.com/docs/cli/commands.html.

[41] «El android libre,» [En línea]. Available: https://elandroidelibre.elespanol.com/2016/04/como-instalar-aplicaciones-apk-android.html.

[42] «Adictos al trabajo,» [En línea]. Available: https://www.adictosaltrabajo.com/tutoriales/html-5-almacenamiento-local/#04.

[43] A. Rueda, Apuntes de la asignatura Desarrollo de aplicaciones empresariales, 2017.

[44] «IETF-related tools, standalone or hosted on tools.ietf.org,» [En línea]. Available: https://tools.ietf.org/html/rfc7519.

9. APÉNDICE I: Manual de instalación del sistema

Aplicación del servidor

Inicialmente se requiere tener instalado MAMP e iniciar la aplicación, para que se inicie de MYSQL Server. Los parámetros de la BD son:

spring.datasource.url:jdbc:mysql://localhost:3306/jpa?useSSL=false&serverTimezone=UTC&useLegacyDateTimeCode=false

spring.datasource.username: root

spring.datasource.password: rootroot

Es necesario tener instalado Netbeans. El siguiente paso es pulsar sobre “Archivo”-“Importar proyecto” y abrir la carpeta del proyecto que contiene la aplicación del servidor.

Después pulsar sobre “Ejecutar proyecto”.

Aplicación móvil

Primero acceder con consola de comandos al directorio del proyecto de la parte front y ejecutar los siguientes comandos: *npm build* y *ionic serve.* Este comando abrirá el navegador con la aplicación.

Además, se incluye en la carpeta *Platforms* el proyecto compilado para Android o IOS.

10. APÉNDICE II: Manual de usuario

Inicialmente, el usuario cuando inicia la aplicación se le muestra la ventana de Inicio, en la que puede interaccionar entre las provincias y ver que rutas tiene cada una.

En caso de que quiera acceder a una ruta y no esté logeado, se le pedirá que realice el login. Además, se solicitará login para capturar código QR y acceder al perfil.

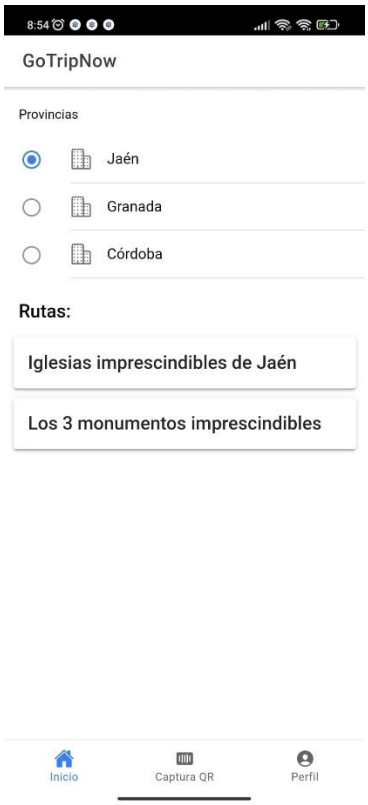


Ilustración 52 Inicio

En la siguiente captura se observa los objetivo que tiene una ruta, si se selecciona el botón de verlos en mapa, se un mapa con los puntos de los objetivos.

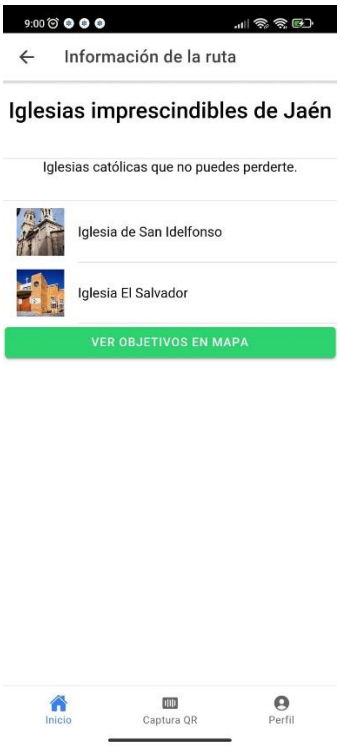


Ilustración 53 Información de ruta

En la siguiente captura si seleccionamos ‘iniciar indicacionies GPS’ y se abrirá la aplicación de navegación del dispositivo para iniciar la navegación desde la posición actual.

Ilustración 54 Objetivos en mapa

Para capturar un código QR, se seleccionará el botón de ‘capturar QR’ donde debemos apuntar con la cámara al código para capturar el objetivo, apareciendo como capturado en la ventana de ‘mi perfil’.

Ilustración 55 Capturar QR

En esta ventana podemos ver el histórico de objetivos capturados, señalando la fecha y hora de la captura y los objetivo que faltan para completar la ruta.

Ilustración 56 Información de perfil

En la siguiente captura observamos la ventana de login.
Si iniciamos sesión con una cuenta de administrador podremos acceder a las funcionalidades de gestión, dado acceso a crear un objetivo o crear una ruta.

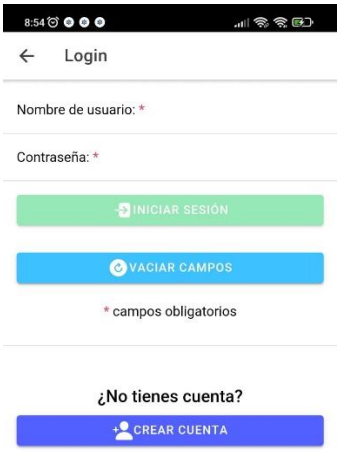


Ilustración 57 Inicio de sesión

La siguiente ventana contiene un formulario para crear un objetivo, se debe rellenar los datos, seleccionar una imagen e indicar la ubicación por coordenadas del objetivo nuevo (podemos seleccionar la ubicación actual)

Ilustración 58 Crear objetivo

En la siguiente captura podemos observar la ventana para crear una ruta nueva, debemos rellenar los datos y seleccionar los objetivos que forman la nueva ruta.

Ilustración 59 Crear nueva ruta

11. APÉNDICE III:Manual de contenidos suministrados

Se suministra en el archivo comprimido la carpeta correspondiente al código de la aplicación front y por otro lado la carpeta correspondiente al código de la aplicación back. Además de el presente documento de documentación.