



Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Iñigo Gascón Royo 685215
Rubén Moreno Jimeno 680882
Darío Sánchez Salvador 680239

Sistemas y Tecnologías Web

Añisclo's POI

Informe final del proyecto

8º cuatrimestre
Ingeniería Informática
22 de mayo de 2017,
Zaragoza

Contenido

1.	Introducción	4
1.1	Descripción del documento.....	4
1.2	Descripción del proyecto y del equipo.....	4
2.	Producto	5
2.1	Plan de producto	5
2.2	Propuestas similares	5
2.3	Planificación de lanzamientos.....	6
2.4	Análisis de riesgos.....	6
2.5	Estado de la aplicación final	8
2.6	Mapa de navegación	15
2.7	Arquitectura del sistema y aspectos interesantes sobre la implementación y el despliegue.....	16
2.7.1	Modelo de datos	16
2.7.2	Vista de componentes y conectores	18
2.7.3	Vista de módulos	20
2.7.4	Vista de distribución.....	22
2.8	Detalles de implementación del sistema	23
2.8.1	Front-end.....	23
2.8.2	Back-end	24
2.8.3	Persistencia y capa de datos	28
2.9	Descripción de las estadísticas implementadas.....	28
3.	Proceso	29
3.1	Estrategia y herramientas usadas para tests	29
3.2	Mantenibilidad y Quality Assurance	31
3.3	Definición de hecho	34
3.4	Estrategia para la construcción automática del software	35
3.5	Despliegue del sistema.....	35
3.6	Problemas encontrados durante el desarrollo.....	36
3.7	Análisis de problemas potenciales.....	36
3.8	Estrategia de control de versiones.....	37
3.9	Esfuerzos por persona y por actividad	38
3.10	Diagramas de trabajo	42
3.11	Velocidad del equipo	46
3.12	Planificación del proyecto	46
3.13	Resultados de las retrospectivas	47

3.14	Medidas que se llevarán a cabo para mejorar	49
4.	Conclusiones	49
4.1	Conclusiones del proyecto.....	49
4.2	Valoración personal del equipo.....	50
4.3	Valoración personal de cada integrante.....	51
5.	ANEXO I: API RESTful del sistema	53

1. Introducción

1.1 Descripción del documento

En este documento se describe el informe final del proyecto *Añisclo's POI*, desarrollado por el grupo "Añisclo" y recoge los resultados de sus sprints. En primer lugar, se va a realizar una descripción del proyecto junto con la descripción del equipo de desarrollo.

En segundo lugar, se va a hablar sobre el producto y su estado actual, enumerando así: el plan del producto, la planificación de los lanzamientos, un análisis de riesgos, estado actual de la aplicación y vistas de la arquitectura del sistema.

En tercer lugar se van a desarrollar los detalles del proceso de desarrollo de la aplicación, es decir: estrategia y herramientas usadas para tests y cobertura de los tests automáticos implementados, definiciones de hecho, estrategia para la construcción automática del software, estrategia de control de versiones que se sigue, esfuerzos por persona y por actividades, diagrama de trabajo completado, velocidad del equipo, planificación del proyecto, resultados de la última retrospectiva y cualquier otro aspecto relacionado con el proceso de trabajo y la gestión del proyecto, además de medidas que se llevarán a cabo para mejorar lo citado anteriormente.

En último lugar se va a realizar una conclusión del proyecto, además de una valoración personal a nivel de grupo y por persona de cada uno de los integrantes del mismo.

1.2 Descripción del proyecto y del equipo

Añisclo's POI es una aplicación web adaptativa centrada en poder gestionar puntos de interés y rutas en un mapa del mundo. Además de poder crear cualquier tipo de POI con determinada información, y poder utilizarlos posteriormente para formar rutas a la carta, también es dotada de muchas funcionalidades que la llevan a ser una red social. Además de poder seguir usuarios, ver los POIs que no son tuyos, poder valorarlos o darles favorito, puedes enviar rutas por correo para que las demás personas puedan reproducirlas a la perfección.

El equipo del proyecto está formado por cuatro miembros: Rubén Moreno Jimeno, Iñigo Gascón Royo, Sergio Martín Segura y Catalin Constantin Dumitrache. Para el desarrollo de la aplicación se ha seguido la metodología *Scrum*, distribuyendo así los roles en:

- **Dueño del producto:** Francisco Javier Fabra
- **Scrum master:** Rubén Moreno
- **Equipo de desarrollo:** Rubén Moreno, Iñigo Gascón y Darío Sánchez

Dado que es un equipo multidisciplinar en el que cada uno de los miembros domina lenguajes distintos de programación, se ha decidido que cada uno de ellos se encargue de unos temas en específico:

- **Rubén Moreno Jimeno:** Front-end Developer, Configuration Manager, Graphical Designer, Test Engineer, Quality Assurance.

- **Iñigo Gascón Royo:** Back-end Developer, Testing Engineer, Quality Assurance, Configuration Manager.
- **Darío Sánchez Salvador:** Front-end Developer, Back-end Developer.

2. Producto

2.1 Plan de producto

Añisclo's POI es una aplicación web adaptativa centrada en ofrecer una experiencia de gestión de rutas y POIs con otros usuarios. La aplicación cuenta con un mapa interactivo del mundo con el cual se puede interactuar para facilitar la creación edición y visualización tanto de POIs como de rutas.

Un usuario registrado dispone de las funcionalidades de ver, editar, duplicar, borrar y crear POIs (además de posteriormente poder realizar búsquedas sobre ellos) conteniendo una cantidad de información personalizable que va desde el título, un conjunto de tags, una descripción, una imagen y una url (la cual puede ser acortada). Una vez creados los POI, con ellos puede construir rutas que posteriormente se visualizan en el mapa junto con su información de gps de los pasos a seguir para completarla. Además, puede valorar los distintos POIs, seleccionarlos como favorito, seguir a otros usuarios, o mandar las rutas por correo para posteriormente poder reproducirlas exactamente. Por último, se le ofrece la capacidad de una sección de estadísticas de interés para su cuenta.

Un usuario de tipo administrador tiene las mismas funcionalidades del mapa que un usuario normal, pero además, dispone de la capacidad de listar todos los usuarios del sistema, pudiendo banearlos definitivamente o indefinidamente, editarles información de la cuenta, o activarles en caso de que la hubieran borrado anteriormente. También dispone la capacidad de visualizar estadísticas, pero en este caso en vez de a nivel de interés de una cuenta, a nivel de interés del sistema entero.

2.2 Propuestas similares

Existen actualmente algunas aplicaciones ya existentes que proporcionan parte de la funcionalidad de la página convirtiéndose así en potenciales competidores de la misma.

- **Google Maps:** La propia aplicación de Google maps, de la cual *Añisclo's POI* utiliza recursos, sirve para crear rutas y puntos de interés en los mapas, y poder hacer que otros usuarios los vean.
- **Quikmaps:** Dibuja y añade los puntos de interés y anotaciones en los mapas de Google.
- **CommunityWalk:** Aplicación para crear una comunidad de mapas profesionales con puntos de interés.
- **OpenStreetMap:** Aplicación que permite editar, ver y utilizar datos geográficos en un entorno de colaboración desde cualquier lugar.

Como se ha podido observar, existen varias aplicaciones similares, sin embargo, *Añisclo's POI* ofrece una visión mucho más enfocada a las redes sociales dotando de capacidades como enviar rutas por email, valorar y marcar como favorito a los POI, seguir a otros usuarios, y visualizar las estadísticas de tu cuenta en el sistema, lo cual le añade un gran

componente social que le distingue de otras aplicaciones que solo sirven para crear POIs, visualizar mapas, o como mucho poder compartir tus POI con otras personas.

2.3 Planificación de lanzamientos

- **Lanzamiento del proyecto (2017/03/ – 2017/03/30)**

Reunión de lanzamiento del proyecto en la que se decidirán los roles *Scrum* a seguir por el equipo de desarrollo, definición del producto y del primer Sprint.

- **Primer Sprint (2017/03/31- 2017/04/09)**

Realización del primer Sprint del primer lanzamiento del producto, incluyendo únicamente los requisitos de gestión de usuarios e integración de algunos servicios como comunicación segura vía https, definiciones de swagger, y tests de protractor y mocha-.

- **Segundo Sprint (2017/04/10 - 2017/04/23)**

Realización del segundo Sprint del primer lanzamiento del producto, incluyendo las funcionalidades del primer sprint y las siguientes: Interactuar con un mapa y poder visualizar POIs y gestionarlos (crear, borrar, duplicar, y editar), acortar urls, crear rutas de POIs y visualizarlas en el mapa, buscar POIs por tags, enviar rutas por correo, poder valorar POIs o marcarlos como favoritos, y seguir a otros usuarios.

- **Tercer Sprint (2017/04/24 - 2017/04/30)**

Realización del tercer Sprint del primer lanzamiento del producto, incluyendo las funcionalidades de los anteriores y las siguientes: Implementación de 11 estadísticas con valor para el usuario sin exceder la información que debería tener del sistema.

- **Cuarto Sprint (2017/05/01 - 2017/05/14)**

Realización del tercer Sprint del primer lanzamiento del producto, incluyendo las funcionalidades de los anteriores y las siguientes: Añadir el tipo de usuario administrador y todas sus funcionalidades asociadas, incluyendo gestionar usuarios (banear/desbanear permanente e indefinidamente, reactivarlos y editarlos), y estadísticas pertenecientes al estado del sistema. Además una mejora en las sesiones incluyendo json web tokens y la exposición de 3 servicios del sistema con XML y DTD

2.4 Análisis de riesgos

Esta sección realiza un inventario de los riesgos potenciales del proceso de desarrollo más importante. Anticipando lo que puede ocurrir, es posible mitigar el riesgo tomando las medidas oportunas. Los riesgos aplican directamente al proceso de desarrollo o tienen una consecuencia directa sobre el proceso de desarrollo. El registro y monitorización de estos riesgos continúa después de que se haya redactado este informe, ya que es un proceso continuo.

Los siguientes riesgos se han identificado para el proceso de desarrollo, con un total de 2 riesgos de nivel alto, y 2 de nivel medio:

Nº	Evento	Consecuencia	Impacto (1-5)	Probabilidad (1-5)	Exp. al riesgo	Medidas	Propietario (responsable de tomar medidas)
1	Falta de formación de los desarrolladores en el proceso.	Retraso en general en el proceso debido a una baja eficiencia de alguno de los miembros del equipo de desarrollo.	2	2	4	Establecer un periodo de aprendizaje para formar al equipo de desarrollo correctamente.	Todo el equipo de desarrollo.
2	Falta de uso, en la gestión del proceso, del uso de métricas e indicadores.	Falta del uso de métricas e indicadores para poder analizarlas y mejorar la gestión del proceso.	4	2	8	Realizar una recopilación y análisis de métricas del proceso durante la duración del mismo.	Dueño del producto.
3	Falta una herramienta de gestión de proyectos de software.	Falta del uso de herramientas específicas para la gestión del proyecto.	4	2	8	Buscar una herramienta específica para la gestión del proyecto adecuada al entorno actual.	Scrum Master.
4	Falta de formación del equipo en el uso de algunas herramientas.	Retraso en general en el proyecto debido a una baja eficiencia del equipo de desarrollo.	2	2	4	Establecer un periodo de aprendizaje para formar al equipo de desarrollo correctamente.	Todo el equipo de desarrollo.
5	Novedad de la tecnología a construir en la organización.	Posibilidad de no finalización del proyecto.	4	4	16	Estudiar con detenimiento el comportamiento de los algoritmos a implementar.	Todo el equipo de desarrollo.
6	Personas que trabajan en múltiples proyectos.	Personas que trabajan en múltiples proyectos.	5	1	5	Tratar de que el equipo de desarrollo no esté inmerso en múltiples proyectos o	Scrum Master.

						tenerlo en cuenta para las estimaciones.	
9	El equipo de desarrollo no ha recibido la capacitación necesaria.	Posibilidad de no finalización del proyecto.	4	4	16	Encargarse de que el equipo de desarrollo reciba la capacitación necesaria de parte de los docentes de la asignatura.	Scrum Master y todo el equipo de desarrollo.

El equipo de desarrollo, el Scrum Master, y el dueño del producto son conscientes de los riesgos y monitorizan las medidas adoptadas.

2.5 Estado de la aplicación final

Se pueden ver las funcionalidades que corresponden a cada una de las pantallas de la aplicación en el siguiente listado:

- Pantalla de iniciar sesión: Iniciar sesión, inicio de sesión con Google. (**Figura 1**)
- Pantalla de registro: Registrar usuario. (**Figura 2**)
- Pantalla de recuperar contraseña: Recuperar contraseña. (**Figura 3**)
- Pantalla de cambiar contraseña: Cambiar la contraseña tras el primer inicio de sesión. (**Figura 4**)
- Pantalla de mapa: Visualizar mapa; crear, buscar, listar y visualizar POIs; crear rutas con POIs o con ID. (**Figura 5**)
- Pantalla de modal de POI: Crear, editar, borrar, duplicar, y centrar en POI; acortar URL; valorar y marcar POI como favorito; seguir usuario. (**Figura 6**)
- Pantalla de visualizar ruta: Visualizar ruta sobre mapa y su información de gps; enviar ruta por correo. (**Figura 7**)
- Pantalla de ver favoritos: Ver POIs favoritos. (**Figura 8**)
- Pantalla de ver seguidores: Ver los POI de las personas a las que sigues. (**Figura 9**)
- Pantalla de estadísticas de usuario: Ver las estadísticas de usuarios. (**Figura 10**)
- Pantalla de ajustes de cuenta: Editar información de la cuenta; borrar cuenta. (**Figura 11**)
- Pantalla de gestionar usuarios: Gestionar usuarios como administrador pudiendo inhabilitarlos o habilitarlos, activarles la cuenta, y editarles la información de la misma. (**Figura 12**)
- Pantalla de estadísticas de administrador: Ver las estadísticas de administrador. (**Figura 13**)

Pirineo's POI [Iniciar Sesión](#) [Registrarse](#)

¡Inicia sesión en Pirineo's POI!

[Iniciar sesión](#)

Ha olvidado su contraseña


[Sign in with Google](#)

Figura 1. Pantalla de iniciar sesión.

Pirineo's POI [Iniciar Sesión](#) [Registrarse](#)

¡Regístrate en Pirineo's POI!

☐ No soy un robot

 reCAPTCHA

Privacidad - Condiciones

[Registrarse](#)

Figura 2. Pantalla de registro.

Introduce tu correo electrónico y te mandaremos una nueva contraseña

☐ No soy un robot reCAPTCHA
[Privacidad](#) - [Condiciones](#)[Recuperar contraseña](#)

Figura 3. Pantalla de recuperar contraseña.

Por favor, cambia tu contraseña

[Cambiar contraseña](#)

Figura 4. Pantalla de cambiar contraseña.

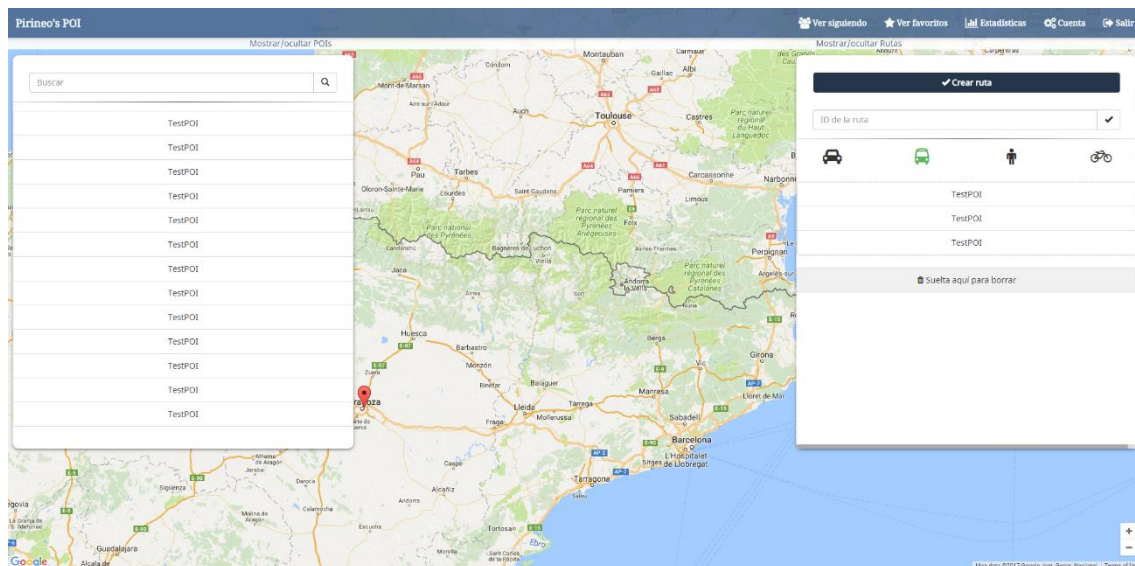


Figura 5. Pantalla de mapa.

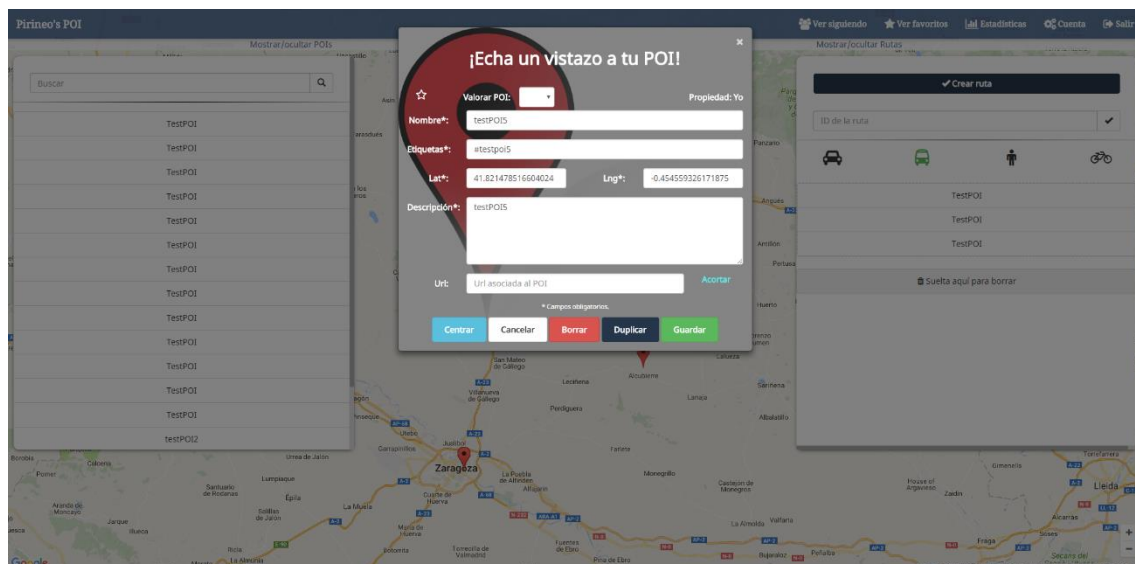


Figura 6. Pantalla de modal de POI.

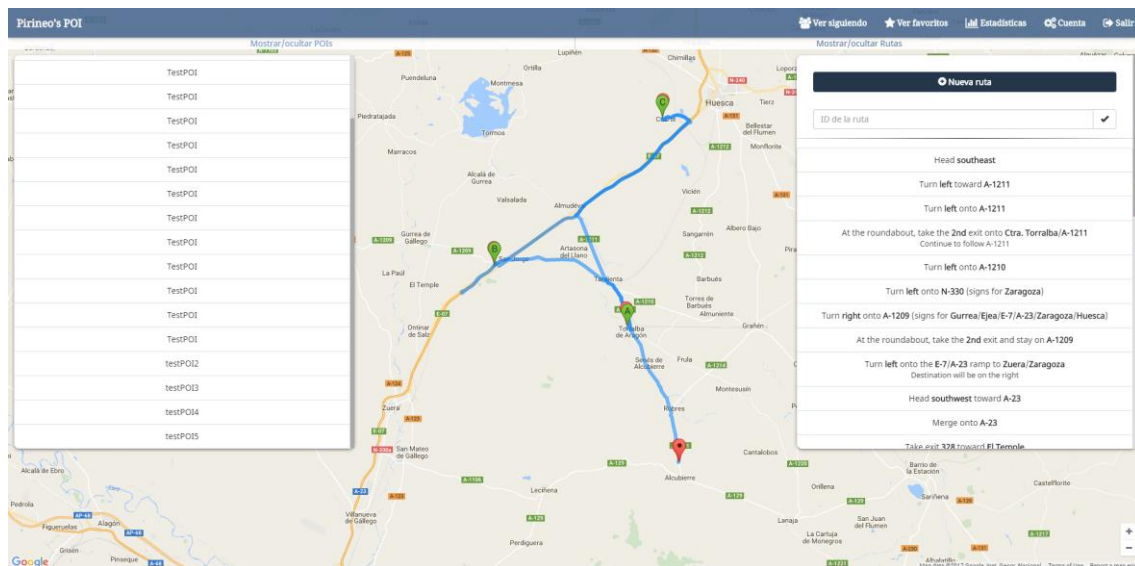


Figura 7. Pantalla de visualizar ruta.

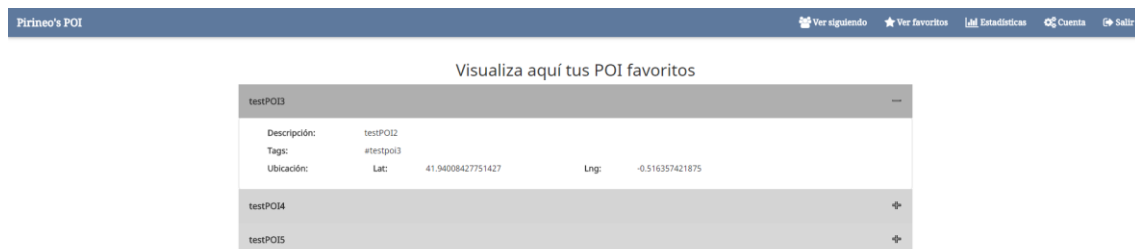


Figura 8. Pantalla de ver favoritos.

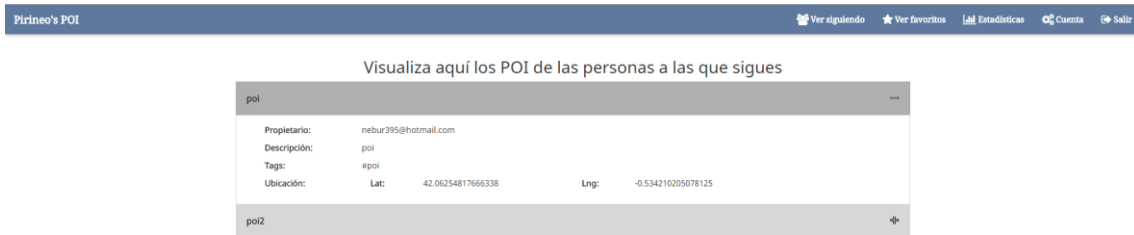


Figura 9. Pantalla de ver seguidores.



Figura 10. Pantalla de estadísticas de usuario.

Pirineo's POI

Ver siguiendo Ver favoritos Estadísticas Cuenta Salir

Bienvenido a tu cuenta Rubén

Apellido: Moreno

Correo electrónico: nebur-el-oscuro@hotmail.com

Introduce tu contraseña actual

Introduce tu nueva contraseña

Cambiar contraseña Borrar cuenta

Figura 11. Pantalla de ajustes de cuenta.

Pirineo's POI

Gestión de usuarios Estado del sistema Cuenta Salir

Gestiona los usuarios del sistema

nebur-el-oscuro@hotmail.com

Nombre: Rubén

Apellido/s: Moreno

Email: nebur-el-oscuro@hotmail.com

Tiempo: 0

Inhabilitar Activar Editar

Figura 12. Pantalla de gestionar usuarios.



Figura 13. Pantalla de estadísticas de administrador.

Todas las funcionalidades implementadas son plenamente usables y alcanzan un nivel de calidad suficiente para un lanzamiento al final del sprint. La navegación por la web se realiza de forma cómoda e intuitiva mejorando la experiencia de usuario gracias a una implementación del cliente de tipo SPA (Single Page Application).

En este momento no tiene constancia de ningún “bug” no resuelto. Las funcionalidades superan adecuadamente los tests automáticos.

2.6 Mapa de navegación

La aplicación cuenta con 3 flujos principales de mapas de navegación, dependiendo del estado de la misma. Si se encuentra el usuario sin iniciar sesión, si el usuario que ha iniciado sesión es administrador, o si el usuario que ha iniciado sesión es usuario normal.

Dado que cuenta con un número elevado de pantallas, se ha simplificado el mapa de navegación para que se puedan visualizar de una forma más legible en el formato actual del documento. Los nombres de las pantallas que aparecen, se refieren a las mismas que las del apartado anterior.

Existen algunas conexiones que no aparecen en el mapa de navegación, esto se ha hecho porque esas pantallas son accesibles a través del navbar en cualquier momento del estado de la aplicación únicamente dependiendo del flujo en el que se encuentre:

- **Sin iniciar sesión:** El navbar permite cambiar entre Registro e Iniciar sesión.
- **Usuario normal:** El navbar permite cambiar entre Mapa, Ver favoritos, Ver seguidores, Estadísticas de usuario, y Ajustes de la cuenta.
- **Usuario administrador:** El navbar permite cambiar entre Mapa, Ajustes de cuenta, Gestionar usuarios, y estadísticas del administrador.

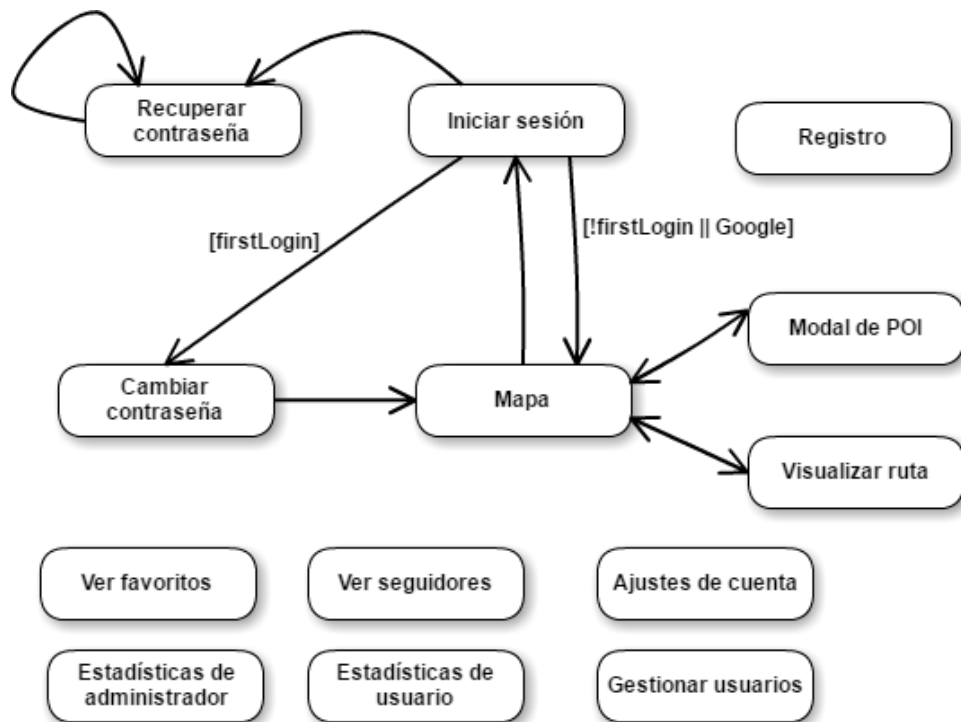


Figura 14. Mapa de navegación.

2.7 Arquitectura del sistema y aspectos interesantes sobre la implementación y el despliegue

2.7.1 Modelo de datos

Presentación primaria.

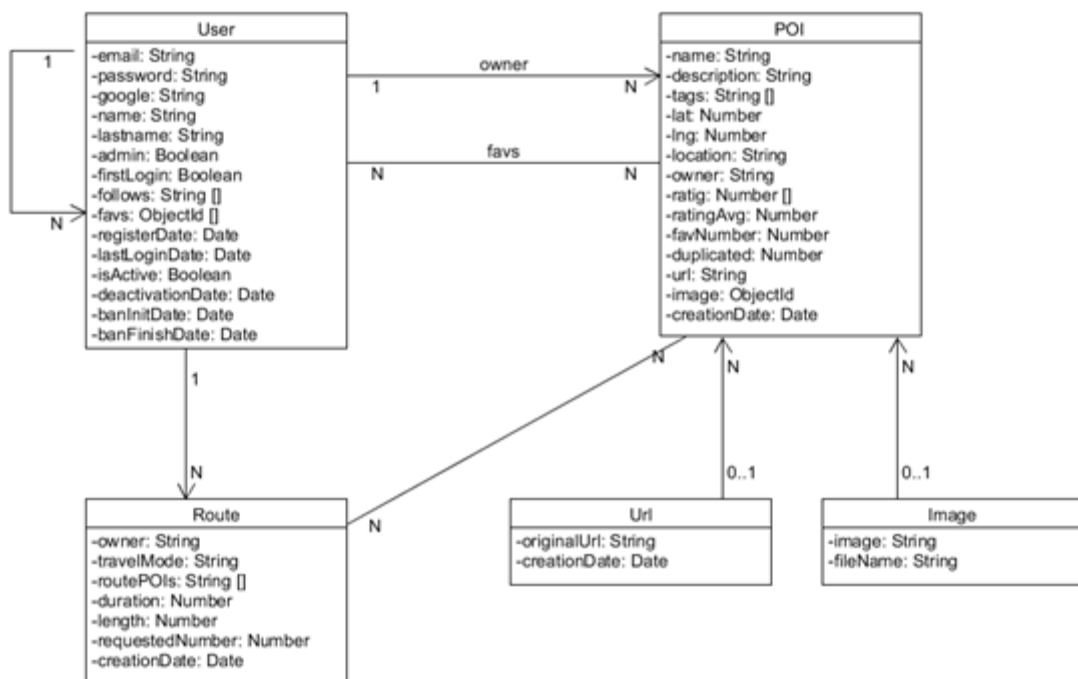


Figura 15. Diagrama de modelo de datos del sistema.

Catálogo de la vista

- **User:** Un usuario registrado en el sistema. Incluye los datos de registro del usuario, información para saber si es administrador, si es su primer inicio de sesión, si la cuenta está activa, los usuarios a los que sigue, sus POIs favoritos y fechas de registro, último acceso y posibles baneos.
- **POI:** Un punto de interés en un mapa en el sistema. Incluye toda la información relevante (nombre, coordenadas etc), quién lo ha creado, las valoraciones recibidas, el número de favoritos recibidos y de veces que se ha duplicado.
- **Route:** Una ruta creada en el sistema. Incluye una lista con los POIs que conforman la ruta, la duración y distancia de la misma, la fecha de creación, usuario que la ha creado y el número de veces que se ha solicitado para mostrarla.
- **Url:** Una URL acortada en el sistema. Incluye la fecha de creación y la URL original.
- **Image:** Una imagen adjuntada a un POI y almacenada en el sistema. Incluye la imagen en base 64 y el nombre con el que se ha guardado la imagen.

Exposición de razones

Se han tomado numerosas decisiones de diseño a lo largo del desarrollo del proyecto en lo referente al modelo de datos del mismo. En primer lugar, centrando la atención en la entidad User, se ha decidido almacenar los usuarios a los que se sigue en la propia entidad, en lugar de externalizarlo a otra colección distinta que relacionase a un usuario con todos sus seguidores. De igual manera, se ha decidido almacenar en la entidad User una lista con todos los POIs favoritos del usuario, también como alternativa a crear una colección específica para ello.

Por otro lado, pasando a la entidad POI, se ha decidido que cada entidad almacenaría una lista de todas las valoraciones recibidas, además de un campo que contendría la media de dichas valoraciones. La decisión de guardar la media, además de todos los valores (lo cual resulta en cierto modo redundante), se ha tomado debido a que es mucho más rápido calcular la media de valoraciones de un POI cada vez que se añada una valoración nueva, que tener que calcular la media de todos los POIs en una sola petición cuando se quieran mostrar estadísticas relacionadas con las valoraciones medias de los POIs. Por otro lado, también se almacenan el número de favoritos que ha recibido un POI (que también puede considerarse información redundante, dado que cada usuario guarda su lista de favoritos) y el número de veces que se ha duplicado un POI; de nuevo, son decisiones que se han tomado de cara a extraer estadísticas de forma más rápida y sencilla. Dado que estos posibles datos redundantes se emplean únicamente para fines estadísticos, no supondrían un problema crítico en caso de no ser consistentes en algún momento.

2.7.2 Vista de componentes y conectores

Presentación primaria

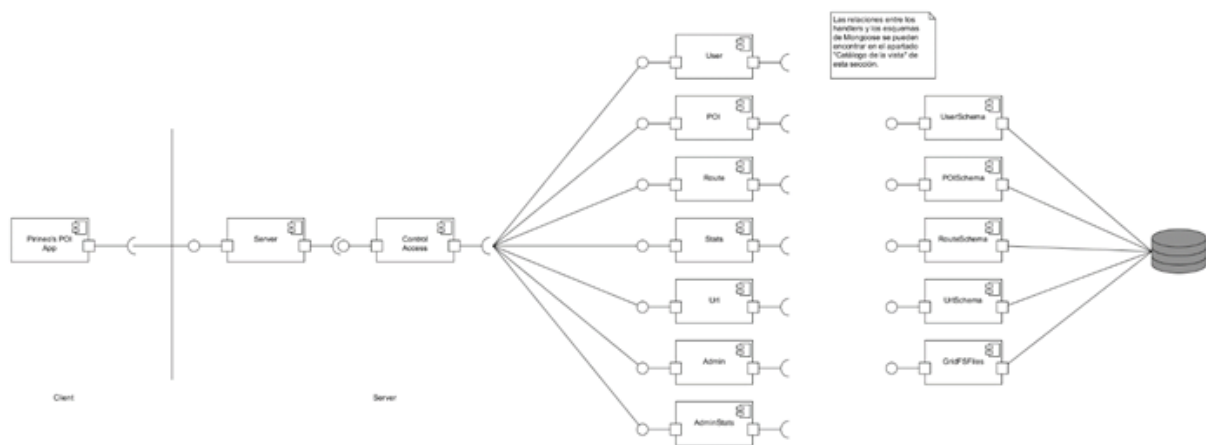


Figura 16. Diagrama de CyC.

Catálogo de la vista

En el diagrama presentado en el apartado anterior pueden diferenciarse las tres capas de la arquitectura Modelo-Vista-Controlador. A continuación, se van a especificar los componentes pertenecientes a cada una de las capas, así como sus conexiones entre ellos.

- **Vista:**

- **Pirineo's POI App:** Este componente representa la vista de la aplicación al completo. Requiere del componente Server dado que es quien realiza todas las peticiones al servidor.

- **Controlador:**

Las conexiones entre los componentes del controlador y los del modelo se encuentran representadas en la Tabla X.

- **Server:** Representación del server.js, que realiza todo el enrutamiento de los módulos que gestionan las peticiones, así como del módulo del control de acceso de usuarios. Requiere el componente Control Access dado que todas las peticiones pasan primero por el filtro para comprobar que la petición la ha realizado un usuario válido y que ha iniciado sesión.
- **Control Access:** Representa el filtro de JSON Web Tokens del servidor que comprueba que las peticiones realizadas son válidas con respecto a si el usuario existe y se ha logueado. Requiere de todos los componentes que gestionan las peticiones para redirigirlas en caso de que la petición sea válida.
- **User:** Componente que gestiona las peticiones relacionadas con las operaciones sobre un usuario del sistema.
- **POI:** Componente que gestiona las peticiones relacionadas con las operaciones sobre un POI del sistema.
- **Route:** Componente que gestiona las peticiones relacionadas con las operaciones sobre una ruta del sistema.

- **Stats:** Componente que gestiona las peticiones para obtener estadísticas del usuario.
- **URL:** Componente que gestiona peticiones sobre la creación y recuperación de URLs acortadas en el sistema.
- **Admin:** Componente que gestiona las peticiones sobre operaciones que puede realizar un administrador.
- **AdminStats:** Componente que gestiona las peticiones para obtener estadísticas para el administrador.
- **Modelo:**
 Todos los componentes del modelo están conectados directamente con la base de datos MongoDB mediante el driver Mongoose.
 - **UserSchema:** Componente que representa el modelo de datos de un usuario.
 - **POISchema:** Componente que representa el modelo de datos de un POI.
 - **RoutesSchema:** Componente que representa el modelo de datos de una ruta.
 - **URLShortener:** Componente que representa el modelo de datos de una URL acortada.
 - **GridFSFiles:** Componente que representa el modelo de datos de una imagen guardada en el sistema mediante la especificación GridFS de almacenamiento y recuperación de ficheros.

Componentes del controlador Componentes del modelo

User	UserSchema POISchema
POI	User Schema POISchema UrlSchema
Route Stats AdminStats	UserShema POISchema RouteSchema
Admin	UserSchema
Url	UrlSchema

Tabla 1: Conexiones entre controlador y modelo.

Exposición de razones

El framework de MEAN Stack lleva de forma casi natural a realizar una implementación siguiendo el patrón de diseño Modelo-Vista-Controlador, por lo que se ha decidido seguir dicho patrón para el desarrollo de la aplicación web.

Los controladores se han organizado según los recursos sobre los que trabajan, dado que se está siguiendo un estilo arquitectura REST, que se centra en los recursos del sistema. Todas las operaciones para usuarios, POIs, estadísticas etc están separadas y organizadas en sus propios ficheros y cuentan con su propia ruta para llamar a dichas funciones desde la vista.

2.7.3 Vista de módulos

Presentación primaria

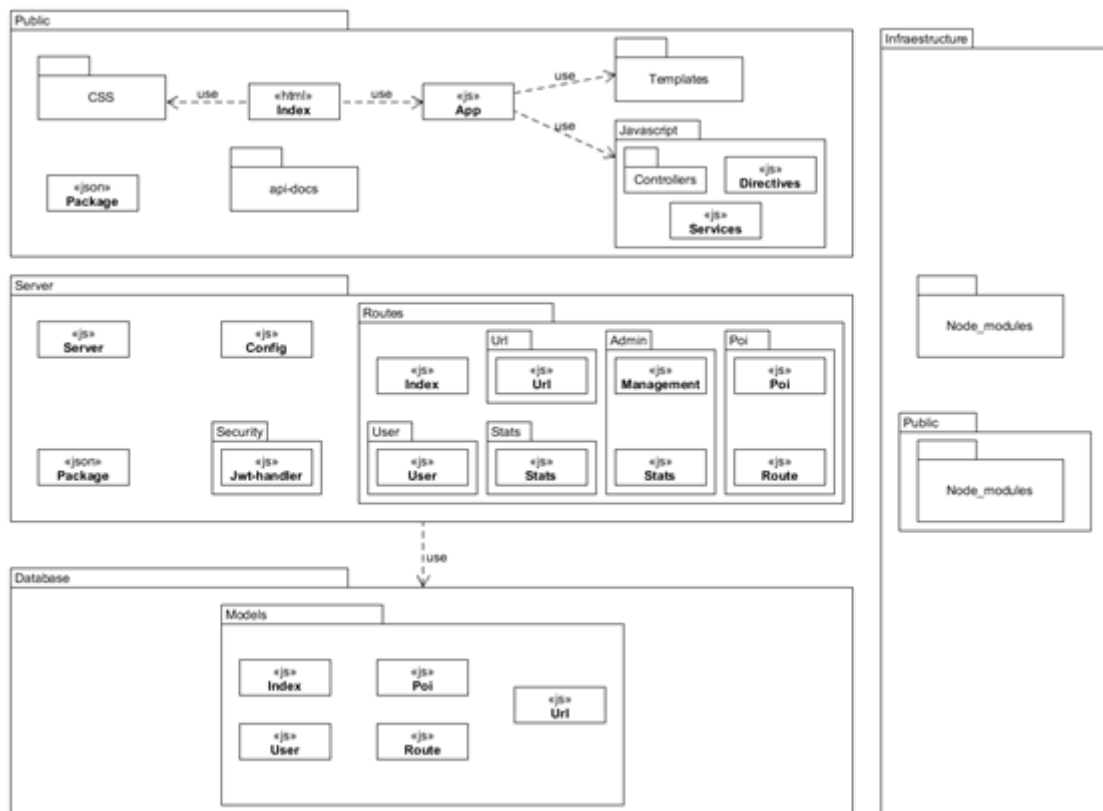


Figura 17. Diagrama de módulos del sistema.

Catálogo de la vista

El diagrama de módulos presentado en esta sección representa las cuatro capas del sistema desarrollado: vista, controlador, modelo e infraestructura. Se va a explicar a continuación el contenido de cada capa, sin entrar en demasiado detalle.

- **Módulo “Public”:**

Representa los módulos y ficheros que conforman la parte de la vista de la aplicación.

- **Index:** El fichero index.html de la aplicación web. Usa los módulos de CSS, así como el fichero app.js de la aplicación.
- **CSS:** Módulo que contiene todos los ficheros CSS necesarios para la vista de la aplicación.
- **App:** El fichero app.js de la vista de la aplicación. Usa las plantillas definidas en el módulo Templates y los ficheros Javascript definidos en el módulo Javascript.

- **Templates:** Módulo que contiene las plantillas HTML necesarias para la vista de la aplicación.
- **Javascript:** Módulo que contiene los ficheros Javascript que manejan la vista de la aplicación. Lo conforman los ficheros `services.js` y `directives.js`, además del módulo `Controllers` que contiene todos los controladores de la vista.
- **Api-docs:** Módulo que contiene los ficheros necesarios para la documentación de la API en Swagger.
- **Package:** Fichero `package.json` del módulo `Public` que contiene las dependencias de módulos de la vista de la aplicación.
- **Módulo “Server”:**
Representa los módulos y ficheros que conforman la parte del controlador de la aplicación.
 - **Server:** Fichero `server.js` de la aplicación, encargado de crear el servidor, la conexión con la base de datos y fijar todas las rutas y módulos globales necesarios.
 - **Config:** Fichero `config.js` de la aplicación. Contiene la clave secreta empleada para la firma de los JSON Web Tokens emitidos.
 - **Package:** Fichero `package.json` de la aplicación. Contiene dependencias, dependencias de desarrollo y la definición de algunos comandos para lanzar con `npm`.
 - **Security:** Módulo que contiene los ficheros encargados de la gestión de los JSON Web Tokens empleados en la aplicación.
 - **Routes:** Módulo que contiene todos los ficheros con los endpoints y la lógica de la aplicación.
 - **Index:** Fichero que contiene la definición de las rutas para cada uno de los ficheros del módulo.
 - **User:** Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre los usuarios del sistema.
 - **Url:** Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre las URLs acortadas del sistema.
 - **Stats:** Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre las estadísticas de los usuarios del sistema.
 - **Admin:** Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles para el administrador del sistema y las estadísticas que se muestran para éste.
 - **Poi:** Módulo que cuenta con los ficheros que contienen la lógica de la aplicación en lo que se refiere a las funciones disponibles sobre los POIs y las rutas del sistema.
- **Módulo “Database”:**
Representa los módulos y ficheros que conforman la parte del modelo de la aplicación.
 - **Models:** Módulo que contiene los ficheros de definición de modelos de datos de la aplicación.
 - **Index:** Fichero que exporta globalmente los modelos de datos de la aplicación.
 - **User:** Fichero que contiene el modelo de datos de un usuario en el sistema.
 - **Poi:** Fichero que contiene el modelo de datos de un POI en el sistema.

- **Route:** Fichero que contiene el modelo de datos de una ruta en el sistema.
- **Url:** Fichero que contiene el modelo de datos de una URL acortada en el sistema.
- **Módulo “Infraestructura”:**
Representa los módulos y ficheros que conforman la parte de infraestructura de la aplicación.
 - **Node_modules:** Módulo que contiene los módulos de npm empleados en la parte del controlador de la aplicación.
 - **Public/Node_modules:** Módulo que contiene los módulos de npm empleados en la parte de la vista de la aplicación.

Exposición de razones

Como ya se ha expuesto anteriormente, la aplicación sigue un patrón Modelo-Vista-Controlador, por lo que en el diagrama de módulos se han representado las distintas capas agrupando los módulos que pertenecen a cada una. No obstante, como se puede apreciar en dicho diagrama, la organización que se ha seguido en la aplicación es casi representativa de dicho patrón, agrupando los ficheros de cada capa en sus módulos correspondientes.

Tal y como dicta la arquitectura de capas que se ha representado, se ha respetado que tan sólo las capas superiores pueden utilizar las inferiores y nunca viceversa. La capa de infraestructura, aunque representada a un lateral, se sitúa en la parte más baja y todas las capas pueden acceder a ella.

2.7.4 Vista de distribución

Presentación primaria

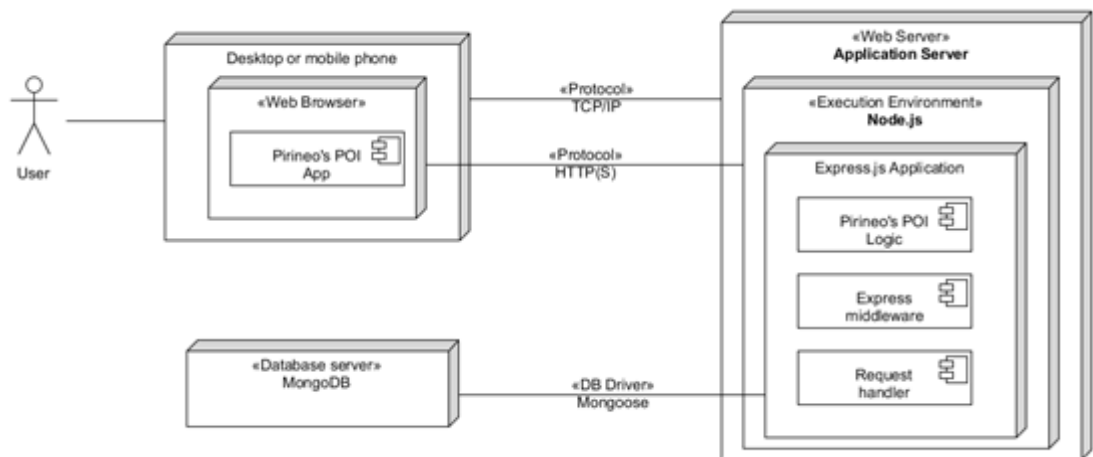


Figura 18. Diagrama de despliegue.

Catálogo de la vista

- **User:** Actor que interactúa con el sistema a través de un dispositivo.
- **Desktop or mobile phone:** Dispositivo sobre el que el actor realiza sus acciones para interactuar con el sistema y obtener resultados.
- **Web Browser:** Navegador del dispositivo que se esté empleando.
- **Pirineo's POI App:** La parte de la aplicación web que se ejecuta en el lado del cliente.
- **Application server:** Máquina en la cual se ejecuta el lado del servidor de la aplicación.
- **Node.js:** Entorno sobre el cual se ejecuta la aplicación en el servidor.
- **Express.js application:** Middleware que se despliega en el servidor y que contiene la lógica de la aplicación y gestiona las rutas de la misma.
- **MongoDB:** Servidor de base de datos conectado al servidor de la aplicación mediante el driver Mongoose.

Exposición de razones

Dado que la aplicación sigue el despliegue habitual para cualquier aplicación desarrollada con MEAN, no se han tomado decisiones relevantes en cuanto a la distribución del sistema.

En la parte del cliente sólo se ha planteado la ejecución de la aplicación desde un dispositivo de sobremesa o un dispositivo móvil a través de su navegador, descartando una aplicación específica para dispositivos móviles de ningún tipo.

2.8 Detalles de implementación del sistema

2.8.1 Front-end

En lo referente a los detalles de la implementación del Front-end realizado con AngularJS 1, se van a aclarar algunos detalles y nombrar las librerías de las que se ha hecho uso.

Primero hay que aclarar que se ha seguido uno de los estándares que marca AngularJS 1 para la organización del proyecto del cliente. De esta forma, se han agrupado todas las templates en un directorio, todos los controladores, directivas y servicios en otro, y en la raíz el index.html junto con el package.json y el app.js.

Se ha decidido usar el gestor de vistas de ui-router en vez del ngRouter puesto que además de ofrecer un mayor potencial y mejora de la flexibilidad gracias a su gestión de los estados, uno de los miembros del equipo ya tenía experiencia en ese gestor.

Las librerías que se han utilizado han sido las siguientes:

- **Angular-base64:** Librería que permite codificar y decodificar datos en base64, utilizado en la funcionalidad del login para realizar el estándar basic authorization.
- **Angular-char.js y chart.js:** Se han utilizado estas dos librerías para poder hacer uso de las gráficas que se requerían en la asignatura.
- **Angular-drag-and-drop-lists y drag-drop-touch-polyfill:** Se han utilizado estas dos librerías para conseguir el extra de poder crear rutas gracias al sistema de drag & drop. La segunda

es un añadido para permitir que dicho sistema funcione también desde dispositivos móviles.

- **Angular-google-maps, angular-simple-logger, y lodash:** se han utilizado estas librerías para poder implementar el mapa de Google Maps correctamente integrado con la aplicación de AngularJS.
- **Angular-jwt:** Librería utilizada para codificar y decodificar los JWT utilizados en el sistema.
- **Angular-recaptcha:** Librería utilizada para el correcto funcionamiento e integración del recaptcha de Google con la aplicación AngularJS.
- **Angular-sanitize:** Librería usada para renderizar texto plano en html. Este componente se ha utilizado para formatear correctamente la información gps devuelta de la API de Google Maps a la hora de crear una ruta.
- **Angular-ui-notification:** Librería para mostrar notificaciones de diversa naturaleza.
- **Satellizer:** Librería utilizada para gestionar el login con Google.
- **X2js:** Librería utilizada para gestionar el uso de traspaso y conversión de la información con formato XML.

Por último añadir que para el correcto funcionamiento del control de accesos en la aplicación, se han utilizado interceptores de peticiones http para incluir automáticamente el jwt si existe en cada una de las peticiones antes de ser enviada, o para comprobar si el servidor ha devuelto un código http 401 (debido a problemas con el jwt) realizar las acciones determinadas (comprobar que no hay errores en el jwt o si no existe, salir de la sesión) antes de que la página siga con su flujo de ejecución correspondiente.

2.8.2 Back-end

A continuación, se van a exponer algunos de los detalles y decisiones de implementación tomados a lo largo del desarrollo del proyecto en lo referente al back-end de la aplicación. No obstante, no se va a entrar en un gran detalle de cómo se ha implementado cada función, sino que se van a mencionar los módulos más interesantes que se han empleado, exponiendo el por qué se han decidido y usar y para qué finalidad. Se va a ir repasando el back-end completo fichero por fichero tal cual se ha organizado en el módulo “routes” del sistema.

user/user.js

Este fichero contiene todas las funciones relacionadas con los usuarios del sistema, tales como crear un nuevo usuario (un registro), iniciar sesión, cambiar la contraseña etc. Además, también se decidió añadir aquí la función de añadir un POI a la lista de favoritos de un usuario, dado que, al fin y al cabo, es el usuario el que guarda la lista de favoritos y es el usuario el que se modifica con la petición; por tanto, se consideró más natural y correcto situar esa función aquí en lugar de en el fichero con las funciones de los POIs.

Por otro lado, dado que se permite el inicio de sesión con una cuenta de Google, se tiene un endpoint distinto que sustituye a la vez al endpoint para registrar un usuario y para iniciar la sesión de un usuario del sistema. Dicho endpoint comprobará, pidiendo las credenciales del usuario a Google, si el usuario ya tiene una cuenta en el sistema o si es la

primera vez. En cualquier de los dos casos, iniciará la sesión al usuario en la aplicación y, si era la primera vez, guardará sus datos en una cuenta nueva en el sistema.

Además, como se ha visto en el modelo de datos, los usuarios cuentan con un campo que indica si la cuenta de usuario se encuentra activa o no. Esto se debe a que la función de borrar usuario no borra realmente ninguna cuenta, sino que la marca como inactiva. El administrador podrá volver a activar una cuenta.

En cuanto a los módulos externos empleados en las funciones relacionadas con el usuario, son los siguientes:

- **base-64**: Módulo que permite codificar y decodificar rstras de bytes o caracteres en o a base 64. Se emplea para decodificar la información del email y la contraseña del usuario enviados por el cliente en codificación base64 al realizar el inicio de sesión del usuario.
- **utf8**: La decodificación con base-64 da lugar a una ristra de bytes que más adelante hay que interpretar. Para ello, se usa el módulo utf8 para pasarlos a la codificación deseada.
- **randomstring**: Módulo cuya funcionalidad es crear una cadena de caracteres alfanuméricos del tamaño deseado. Se emplea para crear la contraseña aleatoria de 8 caracteres que se le envía al usuario tras confirmar su registro en el correo, o en caso de solicitar una nueva contraseña. Tras su primer inicio de sesión con esta contraseña se le obliga al usuario a cambiarla por una de su elección.
- **nodemailer**: Módulo que permite el envío de emails mediante SMTP u otro protocolo facilitándole un servicio junto con un email válido y sus credenciales (que usará para enviar el email que se le indique). En este caso, se le ha facilitado una cuenta de Gmail creada específicamente para esta aplicación. En las funciones de este fichero se envían emails tras registrarse, tras confirmar la creación de la cuenta y en caso de solicitar una nueva contraseña.
- **ip**: Módulo que facilita numerosas funciones para emplear sobre una dirección IP. En este caso, se emplea para extraer la IP donde se encuentra desplegado el sistema y, de esta forma, poder usarla para poner la dirección correcta en los emails de confirmación que se le envían a los usuarios.
- **request**: Módulo que permite el envío de peticiones HTTP. Se emplea para realizar las peticiones necesarias a los servicios externos de Google tanto para el inicio de sesión como para el ReCaptcha.
- **jsonwebtoken**: Módulo que ofrece varias funciones sobre JSON Web Tokens. Se emplea para firmar los tokens que se envían en el inicio de sesión de un usuario.

poi/poi.js

Este fichero contiene todas las funciones relacionadas con los puntos de interés del sistema: crear un nuevo POI, modificarlo, valorarlo, buscar por tags etc. En el caso de la función de eliminar un POI, se hace una comprobación para cerciorarse de si existen una URL acortada o una imagen adjuntas al POI y, de esta forma, borrarlos de la base de datos si ningún otro POI las está referenciando (que podría darse en caso de duplicar un POI, ya que no se duplican imágenes ni URLs acortadas, sino que se referencian los mismos). Y, en el caso de la función de crear un POI, se permite la entrada de datos en formato XML.

En lo referente a los módulos externos empleados en este fichero, son los siguientes:

- **gridfs-stream**: Módulo que facilita el uso de canales de lectura y escritura siguiendo la especificación GridFS de MongoDB. Se emplea para el almacenamiento y lectura de las imágenes adjuntas a los POIs del sistema. Se ha decidido usar GridFS para evitar el límite de tamaño de un documento estándar de MongoDB de 16MB y así poder subir imágenes grandes y de buena calidad, teniendo en cuenta que la restricción es una imagen por POI.
- **semaphore**: Módulo que permite emplear semáforos para operaciones de exclusión mutua. Tal y como se ha explicado, cuando se borra un POI se borran su imagen y URL adjuntas, si las tiene. Para evitar inconsistencias en la base de datos durante los borrados y dado que no existen transacciones en MongoDB, se han empleado los semáforos para realizar los borrados de POI en exclusión mutua.
- **async**: Módulo que facilita un conjunto de funciones para tratar con el asincronismo de Javascript y, en particular, de Node.js. Dado que no se recomienda el uso de `forEach` para iterar arrays, se han empleado las funciones `each` del módulo `async` para iterar listas de POIs. La función `each` de Javascript habría sido igualmente válida de no ser porque se ejecuta de forma asíncrona y, en estos casos, era necesario que la iteración finalizase antes de enviar una respuesta al cliente. La función `each` de `async` permite saber cuándo ha finalizado la iteración para realizar las acciones deseadas.
- **stream**: Módulo que facilita la creación y uso de streams. Se emplea para crear un stream con la imagen en base 64 que envía el cliente de cara a guardarla con `gridfs-stream`.
- **bs58**: Módulo que permite la codificación y decodificación de ristas de bytes a base 58. Se emplea para decodificar una URL acortada y hallar el id que identifica la URL original en la base de datos. Es necesario al buscar la URL de un POI que se va a borrar para comprobar si está acortada y hay que borrarla del sistema.
- **which-country**: Módulo que, dadas unas coordenadas, devuelve el país al que pertenecen. Se emplea para pasar el país obtenido al módulo `lookup`.
- **lookup**: Módulo que ofrece una serie de datos acerca de un país. Se emplea para conseguir el continente al que pertenece el país extraído a partir de las coordenadas del POI. El continente se emplea con fines estadísticos.

poi/route.js

Fichero que contiene las funciones relacionadas con las rutas del sistema: guardar una ruta, enviar una ruta por email y listar todos los POIs y el modo de transporte de una ruta para mostrar la información del GPS cuando se pide la ruta por su ID. La funcionalidad de enviar una ruta por email admite la entrada de los datos en XML.

Los módulos empleados ya se han expuesto en los anteriores ficheros, pero se va a especificar su uso en este caso:

- **async**: Empleado en este caso para iterar los POIs de una ruta y comprobar que existen, así como para calcular la distancia y duración total de todos los tramos de una ruta.
- **ip**: Al igual que en el caso del fichero `user.js`, se emplea para extraer la IP del anfitrión donde se encuentra desplegada la aplicación y así enviar una dirección correcta en el email con la ruta compartida.

- **nodemailer:** De nuevo, para enviar correos al email especificado.

admin/management.js

Fichero que contiene las funcionalidades del administrador sobre los usuarios del sistema tales como listar todos los usuarios existentes en el sistema, modificar la información de un usuario, banear a un usuario temporal o permanentemente, desbanear a un usuario y reactivar una cuenta de un usuario inactivo (“borrado”). La funcionalidad de modificar un usuario admite también la entrada de datos en XML.

El único módulo externo empleado es el siguiente:

- **async:** De nuevo, empleado para iterar la lista de usuarios del sistema y crear la respuesta que se envía al cliente.

admin/stats.js

Fichero que contiene las funciones que devuelven las distintas estadísticas que se le muestran al administrador del sistema, tales como número de usuarios totales, usuarios activos, baneados y no activos, número medio de POIs por usuario etc.

El único módulo externo empleado es el siguiente:

- **async:** Una vez más, empleado para iterar los diferentes resultados de la base de datos y crear las respuestas de las estadísticas para enviar al cliente.

stats/stats.js

Fichero que contiene las funciones que devuelven las distintas estadísticas que se le muestran a los usuarios, personalizadas para cada uno de ellos en función de sus datos, tales como sus POIs mejor valorados, los que más favoritos reciben o más se han duplicado, sus rutas más largas o duraderas etc

El único módulo externo empleado es el siguiente:

- **async:** Al igual que para las estadísticas del administrador, se emplea para iterar los diferentes resultados de la base de datos y crear las respuestas de las estadísticas para enviar al cliente.

url/urlShortener.js

Fichero que contiene las funciones encargadas de acortar URLs y redireccionar aquellas acortadas a la URL original. Tal como se ha expuesto en el modelo de datos, no se guarda la URL acortada sino que se crea a partir del ID generado al guardar la URL original en la base de datos y luego se decodifica para volver a buscar la original a partir del ID.

Los módulos externos empleados han sido los siguientes:

- **bs58:** Empleado para codificar la ID de la URL original almacenada en la base de datos en base 58 y crear así la URL acortada, que incluirá esa cadena de caracteres.

- **ip:** Al igual que en los otros ficheros, se emplea para conseguir la IP del anfitrión donde está desplegada la aplicación y así poder generar la URL acortada en base a esa dirección.

2.8.3 Persistencia y capa de datos

Aunque back-end y persistencia van muy unidos en el MEAN Stack, sí se pueden destacar algunos detalles de esta parte de la implementación.

El driver empleado para la comunicación con la base de datos MongoDB ha sido Mongoose, que, como se ha mencionado, ha facilitado la creación de esquemas y modelos para las colecciones en la base de datos. La conexión se crea en el fichero `server.js`, el primero en ejecutarse al lanzar la aplicación. Dicha conexión se crea con el puerto por defecto de MongoDB (27017) y con la base de datos “`aniscloDb`”. Además, también en el fichero `server.js`, se han asignado las rutas de los modelos en la aplicación con ‘`app.models`’. De esta forma, resulta mucho más sencillo acceder a ellos desde cualquier parte del back-end de la aplicación.

En el caso de los modelos para el usuario y para los POIs ha sido necesario modificar la variable “`Promise`” de Mongoose, fijándola a “`global.Promise`” dado que, de no hacerlo, se indicaba que las promesas que se estaban usando con ese modelo en algunas partes del código estaban deprecated.

Por último, tanto en los modelos para POIs como para las rutas se ha hecho uso de los métodos que se les pueden definir a los esquemas de Mongoose, creando en cada uno de ellos un método que devuelve un objeto de ese modelo, pero ya preparado para devolverlo al cliente, eliminando o modificando algunos de los campos no necesarios.

2.9 Descripción de las estadísticas implementadas

Se ha de tener en cuenta que hay dos tipos de estadísticas en el sistema: de usuario y de administrador.

Las de usuario se han diseñado teniendo en cuenta que el usuario no tiene por qué estar interesado en la información de los POIs de otros usuarios, por lo tanto, en las estadísticas de un usuario no administrador solo se muestran datos de los POIs y rutas creados por el mismo usuario.

Por el contrario, el administrador desea conocer información de los usuarios del sistema, por lo tanto se tienen en cuenta los datos de todos los usuarios para estas estadísticas.

Las estadísticas de usuario son:

- **Top 5 POIs de usuario (valoraciones):** El top 5 de los POIs creados por el usuario mejor valorados por los demás usuarios.
- **Top 5 POIs de usuario (favs):** El top 5 de los POIs creados por el usuario que más usuarios tienen marcados como favoritos.
- **Evolución de los POIs:** Cronograma de POIs creados a lo largo del último año, mes a mes.
- **Localización de los POIs:** Un diagrama de tarta que muestra la proporción de POIs creados en cada continente.

- **Top 5 POIs (duplicados):** El top 5 de los POIs creados por el usuario que más veces han sido duplicados por otros usuarios.
- **Top 5 rutas (longitud):** El top 5 de las rutas creadas por el usuario ordenadas por su longitud.
- **Top 5 rutas (duración):** El top 5 de las rutas creadas por el usuario ordenadas por su duración.
- **Número de rutas con la misma cantidad de POIs:** Se explica solo.
- **Tipo de transporte:** Un diagrama de rombo que muestra en proporción la cantidad de rutas creadas utilizando cada medio de transporte.
- **Top 5 rutas (recreadas):** El top 5 de las rutas creadas por el usuario que más han sido recreadas al utilizar el código enviado por email.

Las estadísticas del administrador son:

- **Número total de usuarios, POIs y rutas** actualmente almacenadas en el sistema.
- **Media de POIs y rutas por usuario.**
- **Estado de los usuarios:** Un diagrama de tarta que muestra la proporción del estado de los usuarios (baneado temporalmente, permanentemente, inactivo o activo).
- **Accesos a lo largo del tiempo:** Número de cuentas que han accedido a la aplicación a lo largo del último año, mes a mes.
- **Nuevos registros a lo largo del tiempo:** Número de registros a lo largo del último año, mes a mes.
- **Cuentas borradas a lo largo del tiempo:** Número de cuentas borradas a lo largo del último año, mes a mes.

Estas estadísticas están diseñadas con los datos que más puedan interesar a cada tipo de usuario en mente. En el caso del usuario, un elevado número de estadísticas acerca del impacto de sus POIs y rutas resulta interesante para poder medir su repercusión entre los distintos usuarios. En el caso del administrador, éste tiene otras prioridades como el control del número y del estado de usuarios. Por ello se considera que las estadísticas elegidas son relevantes y útiles.

3. Proceso

3.1 Estrategia y herramientas usadas para tests

Para la realización de la verificación y validación de la aplicación se han realizado tests unitarios, de sistema y de aceptación, todos ellos automáticos.

Detallando en primer lugar los tests unitarios implementados, se ha empleado la herramienta Mocha, un framework de testing para Node.js. Junto con Mocha, a fin de contar con aserciones para realizar los mencionados tests unitarios, se ha empleado también Chai, una librería de aserciones, también para Node.js, enfocado al desarrollo dirigido por pruebas o por comportamiento. Con la integración de ambas dos herramientas se han conseguido implementar varias suites de tests para las diferentes funcionalidades de la aplicación.

En segundo lugar, se han implementado los test de aceptación utilizando la herramienta Protractor, un framework de testing end-to-end para aplicaciones Angular y AngularJS. Se ha seleccionado esta herramienta para este tipo de tests ya que actualmente la parte del cliente está desarrollada con AngularJS, y esto hace que los test de Protractor se sincronicen con los ciclos de digest, lo que elimina los típicos problemas de tener que introducir esperas activas en este tipo de tests para que le dé tiempo al navegador web a realizar las acciones que se le han encargado.

Ya que este tipo de tests son más costosos en tiempo y complejos para probar todas las posibles combinaciones de acciones realizadas por un usuario, se ha decidido centrar los esfuerzos de estos tests en los caminos más críticos de la aplicación: Creación de POIs, gestión de sesión, gestión de perfil de usuario, gestión de usuarios del administrador. Como se puede ver en la **Figura 19**, se ha conseguido un total de 11 caminos críticos a probar.

Además, se ha utilizado uno de los patrones de diseño de test end-to-end más comunes: Page Objects. Esto ayuda a escribir tests más claros y limpios encapsulando la información acerca de los elementos de la página de la aplicación. De esa forma, los Page Objects pueden ser reusados a través de múltiples tests, y si el template de la aplicación cambia, sólo se tiene que actualizar el Page Object. Se han creado Page Objects para las páginas de la **Figura 1**, la **Figura 5**, la **Figura 6**, la **Figura 11**, la **Figura 12**, y el componente del navbar.

```
11 specs, 0 failures
Finished in 57.152 seconds

[19:23:42] I/launcher - 0 instance(s) of WebDriver still running
[19:23:42] I/launcher - chrome #01 passed
```

Figura 19: Cobertura con tests de Protractor.

En relación a la cobertura de los tests unitarios que se han mencionado, se ha decidido integrar en el proyecto una nueva herramienta que permite, mediante un análisis del código, mostrar en cada estado del proyecto la cobertura del código en ese momento. Dicha herramienta es Codecov, la cual se ha integrado con GitHub, dándole acceso al repositorio para realizar el análisis tras cada commit y pull request. De esta forma, Codecov muestra si la cobertura del código ha aumentado, descendido o si sigue igual tras añadir el contenido que se incluya en dicho commit o pull request. No obstante, dado que sólo comprueba los tests del servidor y no tests del sistema completo, Codecov no tiene en cuenta para el total de cobertura del proyecto los tests de Protractor, porque le es imposible saber cuántas líneas de código cubre al realizar cada petición.

Finalmente, se muestran las estadísticas de cobertura al momento de finalizar el proyecto, presentando así un 46% de cobertura de código tan sólo con los tests unitarios realizados con Mocha en un total de 100 tests.

Files					Coverage
models	40	31	0	9	77.50%
routes	1,200	521	0	679	43.41%
config.js	1	1	0	0	100.00%
security/jwt-handler.js	7	4	0	3	57.14%
server.js	33	30	0	3	90.90%
Project Totals (16 files)	1,281	587	0	694	45.82%

Figura 20: Cobertura del proyecto al finalizar el proyecto.

Como se puede observar, el directorio routes, que contiene los endpoints del backend cuenta con un 43% de cobertura. Ese porcentaje se debe a que los dos endpoints que muestran las estadísticas, tanto para el usuario como para el administrador, no cuentan con tests, una decisión que se ha tomado en base al esfuerzo que conllevaría realizarlos y el beneficio que se obtendría de ellos. Dado que son únicamente peticiones GET y que tan sólo devuelven una información que pretende ser ilustrativa y para nada resulta crítica para el sistema ni para su uso, se consideró que probar esa parte del sistema no era necesario ni reportaría grandes beneficios.

3.2 Mantenibilidad y Quality Assurance

Se ha decidido integrar la herramienta de Quality Assurance y análisis estático del código SonarQube con el proyecto, para mejorar la misma y la mantenibilidad del código. Para la configuración de la herramienta se ha hecho uso principalmente de dos módulos : Sonar way para los perfiles de calidad de Web, y Sonar way para los perfiles de calidad de JavaScript. Dado que ya se cuenta con una herramienta de análisis de cobertura de código, se ha desactivado esa sección de la herramienta.

Para el análisis del código se han activado las reglas que recomienda SonarQube en su página para cada uno de los módulos utilizados, estas se pueden ver en el siguiente enlace:

- **JavaScript:** https://sonarqube.com/coding_rules#qprofile=js-sonar-way-56838/activation=true
- **Web:** https://sonarqube.com/coding_rules#qprofile=web-sonar-way-50375/activation=true

Gracias a los estándares definidos internamente junto con las definiciones de hecho (ver apartado siguiente), el equipo ha intentado hacer un esfuerzo extra para conseguir la máxima calidad en mantenibilidad del código en el proyecto. Los análisis obtenidos son los siguientes:

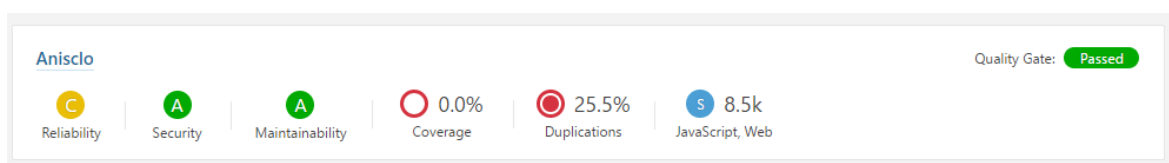


Figura 21. Vista general del análisis de SonarQube.

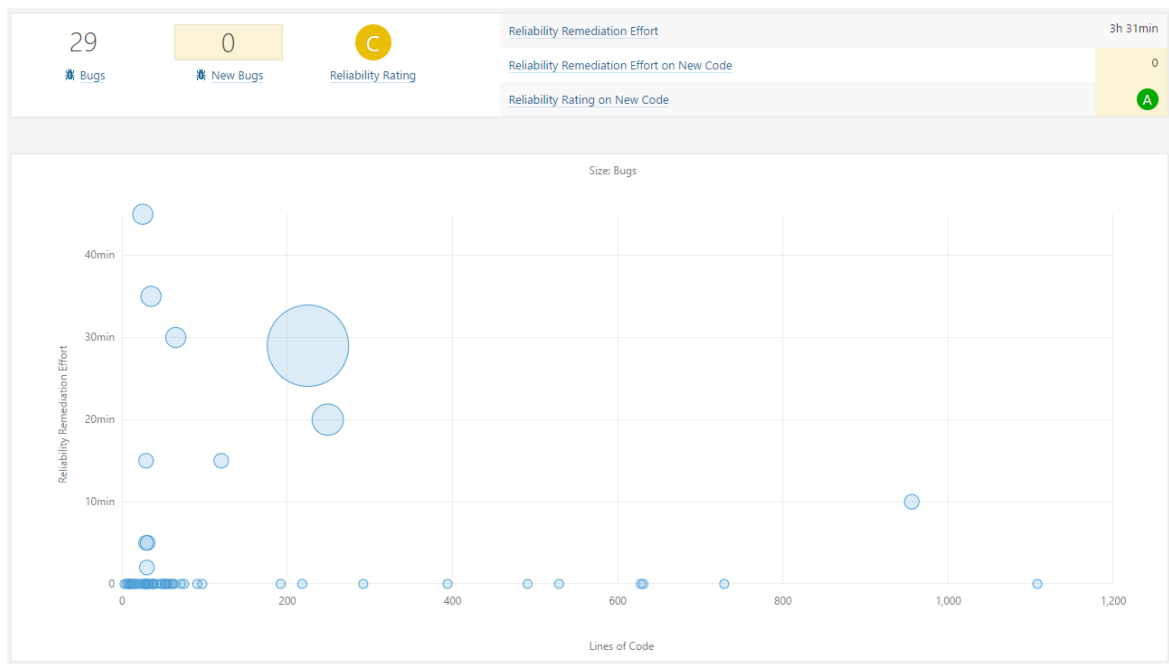


Figura 22. Análisis de SonarQube: Reliability.

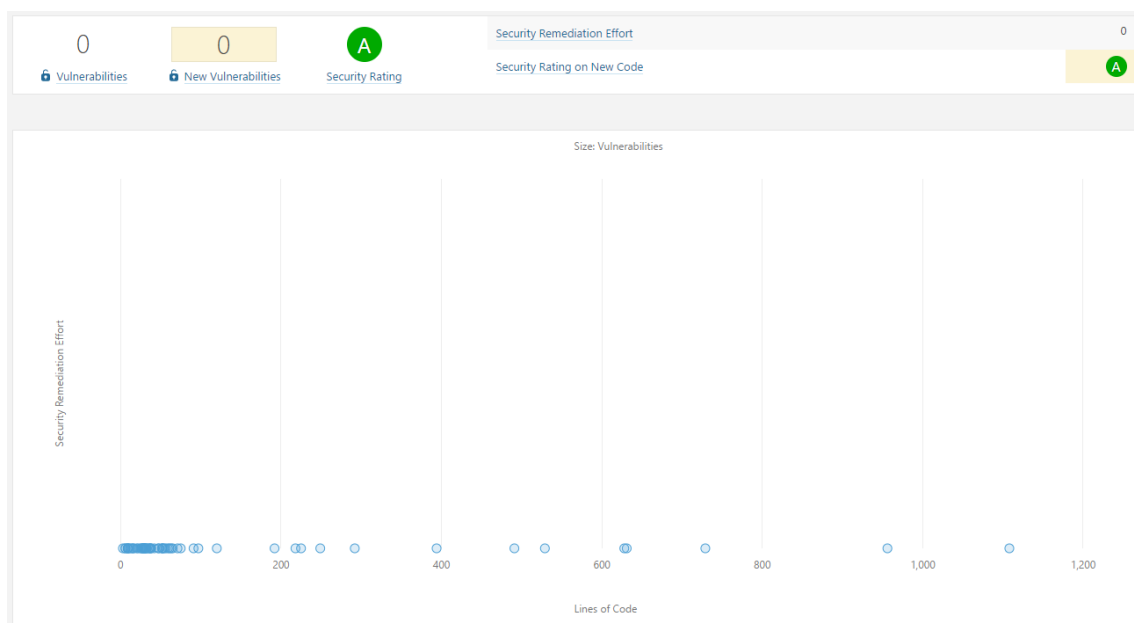


Figura 23. Análisis de SonarQube: Security.

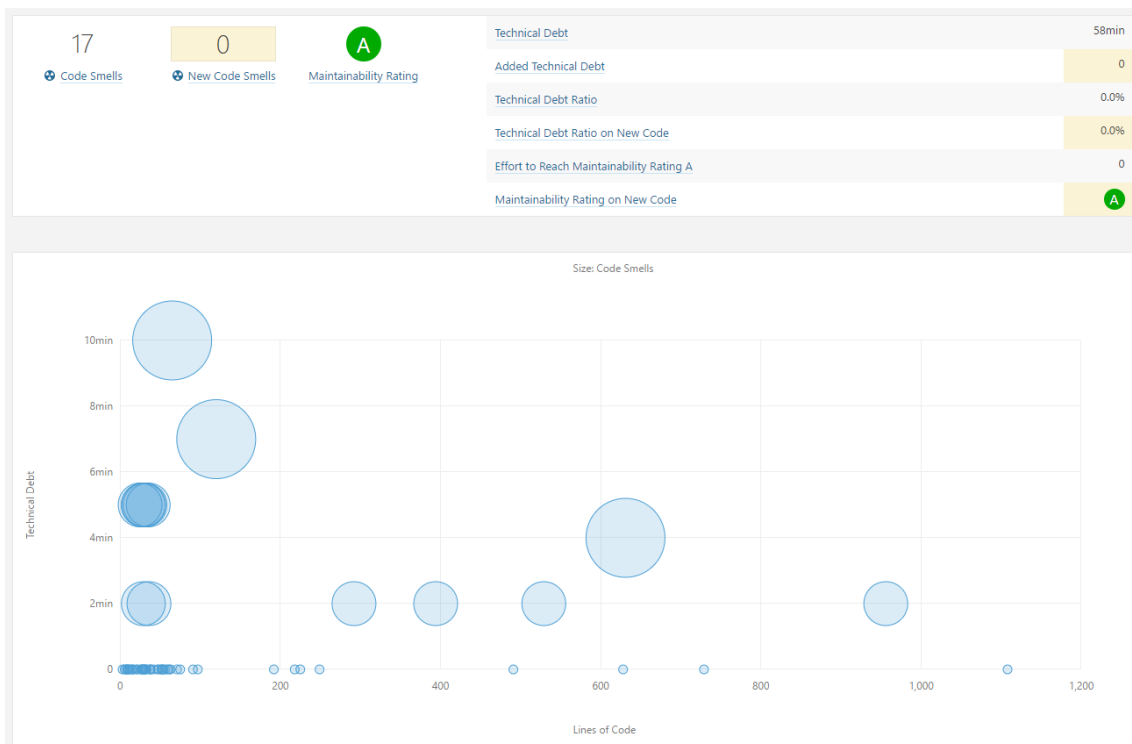


Figura 24. Análisis de SonarQube: Maintainability.

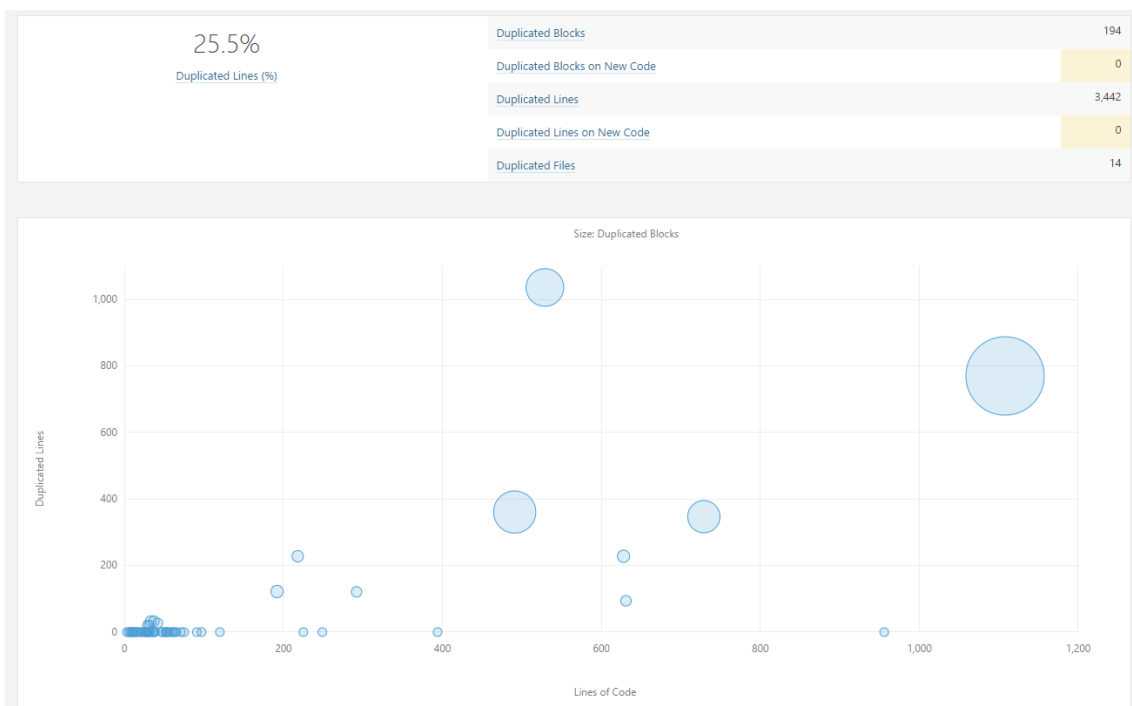


Figura 25. Análisis de SonarQube: Duplications.



Figura 26. Análisis de SonarQube: Size.

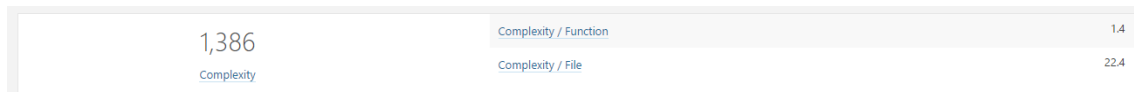


Figura 27. Análisis de SonarQube: Complexity.



Figura 28. Análisis de SonarQube: Issues.

Como se puede observar, el punto más flojo que tiene el proyecto es la métrica de confiabilidad con un nivel C respecto al nivel A del resto. Se cuenta actualmente con un total de 46 issues, las cuales 29 son de confiabilidad, y 17 de mantenibilidad. Además, el proyecto cuenta con una deuda técnica de unas 5h de resolución, para la cual según los análisis se habrían resuelto todas las issues correspondientes alcanzando un nivel A en todas las métricas.

3.3 Definición de hecho

- Todo el equipo de desarrollo pre-verifica los criterios de satisfacción (10 minutos).
- El producto ha sido incluido en la rama master del repositorio del owner y supera todos los test unitarios y de integración automáticos con TravisCI, siguiendo un *Integration-manager workflow*¹ (10 minutos).
- Tanto código de producto como su API están correctamente documentados siguiendo los estándares definidos en la wiki del repositorio: <https://github.com/nebur395/Anisclo/blob/master/docs/Estandares.md> (10 minutos).
- El código del producto sigue los estándares de codificación definidos en la wiki del repositorio: <https://github.com/nebur395/Anisclo/blob/master/docs/Estandares.md> (5 minutos).
- Los nombres de las variables, métodos y clases tienen que ser mínimamente descriptivos, así como guardar una consistencia semántica entre las distintas capas de implementación si se refieren al mismo concepto.

¹ <https://git-scm.com/book/es/v2/Distributed-Git-Distributed-Workflows>

3.4 Estrategia para la construcción automática del software

Dado que la aplicación se ha desarrollado empleando el framework MEAN, la única estrategia de construcción automática empleada ha sido el gestor de paquetes npm. Gracias a que se pueden configurar numerosos comandos en el fichero package.json del proyecto, npm facilita la ejecución de la aplicación con un solo comando: 'npm start' (que simplemente ejecutará el comando 'node' en el fichero principal que lanza el servidor, en este caso el fichero server.js).

Además, dado que su función principal es la gestión de paquetes y dependencias del proyecto, npm cuenta con el comando 'install' que descarga automáticamente todas las dependencias necesarias para el proyecto definidas en el package.json.

Se han configurado también en dicho fichero tres scripts que facilitan la construcción y despliegue de la aplicación: 'postinstall' e 'install', que se ejecutan después de que todos los paquetes de dependencias se hayan instalado tras ejecutar el comando 'install', y 'test', que se ejecuta con el comando 'npm test'. 'Postinstall' ejecuta el comando 'npm install' para el package.json del directorio public, para así instalar las dependencias de la vista de la aplicación. 'Install', por su parte, ejecuta el comando 'update' sobre el webdriver-manager que ejecuta los tests de Protractor, con el fin de mantenerlo actualizado. Por último, 'test' ejecuta los tests de mocha que se encuentren disponibles.

3.5 Despliegue del sistema

Para realizar el despliegue del sistema, asumiendo que el sistema anfitrión ya tiene instalados Node.js y MongoDB, hay que seguir los siguientes pasos en el orden especificado, situándonos de partida en la raíz del proyecto:

- Abrir una terminal de comandos y ejecutar '**npm install**'. Sólo será necesario la primera vez que se despliegue el sistema para descargar todas las dependencias.
- Lanzar un servicio de MongoDB con el siguiente comando en la terminal: '**sudo service mongod start**'. Si se prefiere, también puede crearse una carpeta 'db' en la raíz del directorio y ejecutar el comando '**mongod - -dbpath db**'.
- Añadir un usuario administrador en la base de datos. Para ello, ejecutar el fichero initialize.js mediante el comando '**node initialize.js**'. El usuario administrador se registra con el correo 'master@admin.com' y la contraseña 'pass', con lo que podrá iniciar sesión en la aplicación.
- Ejecutar en otra terminal el comando 'npm start' o, alternativamente, el comando 'node server.js'. El servidor se habrá desplegado cuando la consola indique que se encuentra escuchando en los puertos 8080 (HTTP) y 8443 (HTTPS).
- Para emplear la aplicación, abrir una ventana de un navegador (preferiblemente Mozilla Firefox o Google Chrome >55.X) e introducir la URL 'http://localhost:8080' o 'http://localhost:8443'.

3.6 Problemas encontrados durante el desarrollo

En esta sección se van a analizar los problemas que el equipo de desarrollo se ha encontrado en el proyecto durante su implementación. Cabe destacar que ninguno ha sido especialmente crítico o bloqueante como para ser un problema potencial para la aplicación:

- Durante el desarrollo de la aplicación se encontró un problema con el recaptcha de Google. Este problema viene de que la API de Google Captcha requiere de un dominio para validar tus peticiones y que funcione correctamente. A este dominio se le puede incluir "localhost" para comprobar el funcionamiento en modo desarrollo, y es lo que el equipo tiene actualmente. Sin embargo, esto hace que si se lanza el servidor en local, y se intenta acceder a las funciones del captcha desde otro dispositivo (ya sea móvil o desktop) conectándose a través de la ip de la máquina en local, la petición que procese Google ya no sea la de "localhost" si no otra. Esto hace que de un error y no deje validar el captcha. Para solucionarlo, es bastante simple, solo habría que conseguir un dominio y configurar la API de Google captcha con él. Actualmente este problema sigue existiendo porque el equipo no ha considerado necesario invertir el tiempo en buscar un dominio gratuito, pudiendo invertirlo en otras funcionalidades o no ha querido invertir dinero en comprar un dominio.
- El login de Google fue un problema durante los 2 primeros sprints. Estaba previsto terminarlo para el primer sprint, pero en ese momento, se intentó abordar la funcionalidad vía la API de Google directamente sin ningún intermediario. Esto hizo que surgieran problemas y se alargara su implementación tanto, que tuvo que dejarse fuera del sprint para cumplir los plazos. Durante el segundo sprint, había una carga alta de funcionalidades y no se pudo abordar dicha funcionalidad. Es pues, durante el tercer sprint, donde se pudo terminar dicha funcionalidad, ya que era un sprint un poco más holgado en tiempo. La solución vino de la mano de hacer uso de una librería externa que hace como intermediario realizando la gestión con el login de Google, llamada Satellizer.
- Otro de los problemas que se tuvieron fue con la funcionalidad extra de Drag&Drop. Hubo algunos problemas para conseguir que funcionara correctamente y a la vez fuera usable para los usuarios, haciendo que se pudieran borrar los POI una vez metidos a las rutas, o que consiguiera el funcionamiento esperado y no se reordenaran de la lista original, si no solo en la sección de rutas. Además, otro de los problemas vino de que dicha funcionalidad no funcionaba en los dispositivos móviles, lo cual se solucionó más tarde utilizando la librería de DragDropTouch.

3.7 Análisis de problemas potenciales

Como en cualquier aplicación con vistas a producción y producto real, hay que tener en cuenta algunos problemas potenciales que podría ser interesantes abordarlos en un futuro próximo.

- Al acortar o guardar una URL asociada a un POI, no se comprueba si la misma es alcanzable o no está disponible, lo cual hace que pueda ocasionar problemas a la hora de acceder a la misma posteriormente.

- También referente a las URL que se acortan en el sistema, existe otro problema que aunque tampoco supone un riesgo alto a corto plazo, se podría solucionar sin mucha complejidad. El problema en concreto es que las URLs actualmente se pueden acortar indefinidamente una vez tras otra, pudiendo acortar las que ya hay acortadas. Esto como ya se ha dicho, no supone un alto riesgo a corto plazo porque las redirecciones funcionan correctamente, sin embargo, podría llevar a obtener tiempos de latencia altos si se acorta de forma seguida un número muy elevado de veces. Una solución a esto podría ser comprobar si existe dicha URL acortada en el sistema, antes de volver a acortarla.
- Otra de las cosas que se debería abordar es el uso de colas de peticiones en los servidores. Esto corregiría dos posibles problemas que tiene la aplicación actualmente. El primero, es que se puede spamear con las funcionalidades que envían correos dentro de la aplicación, como por ejemplo enviar rutas por correo. El segundo, es parecido al mismo, pero con el sistema de autenticación, tanto en recuperar contraseña como en registrar usuario, se puede hacer un uso mal intencionado que acabe en spam de correos. Pese a que se tiene ya un sistema de control de bots como el recaptcha, sería interesante añadir otro nivel más de seguridad para evitar el spameo de las personas. Es por eso que un sistema de colas solucionaría los dos problemas, comprobando que si el número de peticiones que se realizan desde una misma ip a determinados servicios supera un número deseado de ejecuciones, restringirle temporalmente hasta que pueda volver a realizar las acciones correspondientes.
- Por último, otro de los problemas que puede tener la aplicación, es que actualmente cuando se borra cualquier POI, no se hace una comprobación de si ese pertenece a alguna ruta del sistema, lo que hace que pueda haber una fuente de errores si entre el momento en el que se crea una ruta, y se envía por correo, antes de recrearla por ID se borra algún POI de la misma. Esto ocasionaría que no se pudiera recrear y hubiera un error. No obstante, no es un error crítico ni hace que el resto de servicios dejen de funcionar, únicamente no se podría recrear esa ruta en particular. La solución, como se ha dicho al principio, pasaría por al borrar un POI comprobar si éste pertenece a alguna ruta en particular.

3.8 Estrategia de control de versiones

Dada la importancia de agilizar el proceso de desarrollo del proyecto, pero sin dejar de lado un buen control de versiones, se ha decidido usar la herramienta GitHub creando un repositorio llamado "Anisclo" y siguiendo un workflow tipo "Integration-manager workflow". De esta forma la organización posee un repositorio *upstream* al cual nadie puede hacer ningún tipo de *push* directo (salvo el owner, o en excepciones y situaciones extremas). A su vez, cada miembro de la organización realiza un *fork* del *upstream* para tener su propio repositorio local y remoto. Los miembros de la organización van realizando sus cambios en local y haciendo *push* a su remoto. Cuando quieran integrar algo lo suficientemente interesante y estable al *upstream* realizan una *pull request* desde su remoto. Una vez hecha, el owner del repositorio se encarga de revisarla, dar su aprobación, dejar *feedback*, o requerir cambios en la misma. Dicha *pull request* solo puede ser "mergeada" cuando el revisor ha dado su aprobación, se ha hecho *pull* del *upstream* en la rama remota y resuelto los conflictos, y se han pasado con éxito los tests automáticos de *TravisCI*.

De esta forma se consigue mantener una frecuencia de trabajo paralela y rápida entre los miembros del grupo, una rama *upstream* lo más estable posible, y un control mínimo de que todo el trabajo integrado en la misma ha pasado una revisión y posee un grado de calidad aceptable.

Para la organización de los documentos y de la gestión del proyecto se ha usado la plataforma de *Google Drive*, en la cual se ha mantenido un seguimiento de los documentos que se han ido actualizando. Se ha creado una organización en el directorio para que se pueda realizar una gestión adecuada del mismo siguiendo la siguiente estructura:

- 1_Gestión
- 2_Documentación producto
 - 2.01_Memorias
 - Z_Otros_docs
- 3_Documentación software
 - 3.01_Análisis y diseño
 - 3.01.01_Diagramas
- 4_Multimedia
- 5_Presentaciones
- Z_Cosas

3.9 Esfuerzos por persona y por actividad

Se ha llevado a cabo una recopilación de los esfuerzos realizados en horas en cada uno de los sprints para cada una de las tareas que se han realizado en ellos.

A continuación, vamos a ver gráficas que contienen el desgano de las horas por tareas por cada una de las entradas de la pila de producto. Esto se pueden observar en la **Figura 29** para el primer sprint, la **Figura 30** para el segundo sprint, la **Figura 31** para el tercer sprint, y la **Figura 32** para el cuarto sprint.

Además, se ha llevado a cabo una gráfica de los esfuerzos dedicados por persona al proyecto de *Añisclo* en la **Figura 33**, y un desglose de los esfuerzos globales dedicados para cada sección del proyecto en la **Figura 34**.

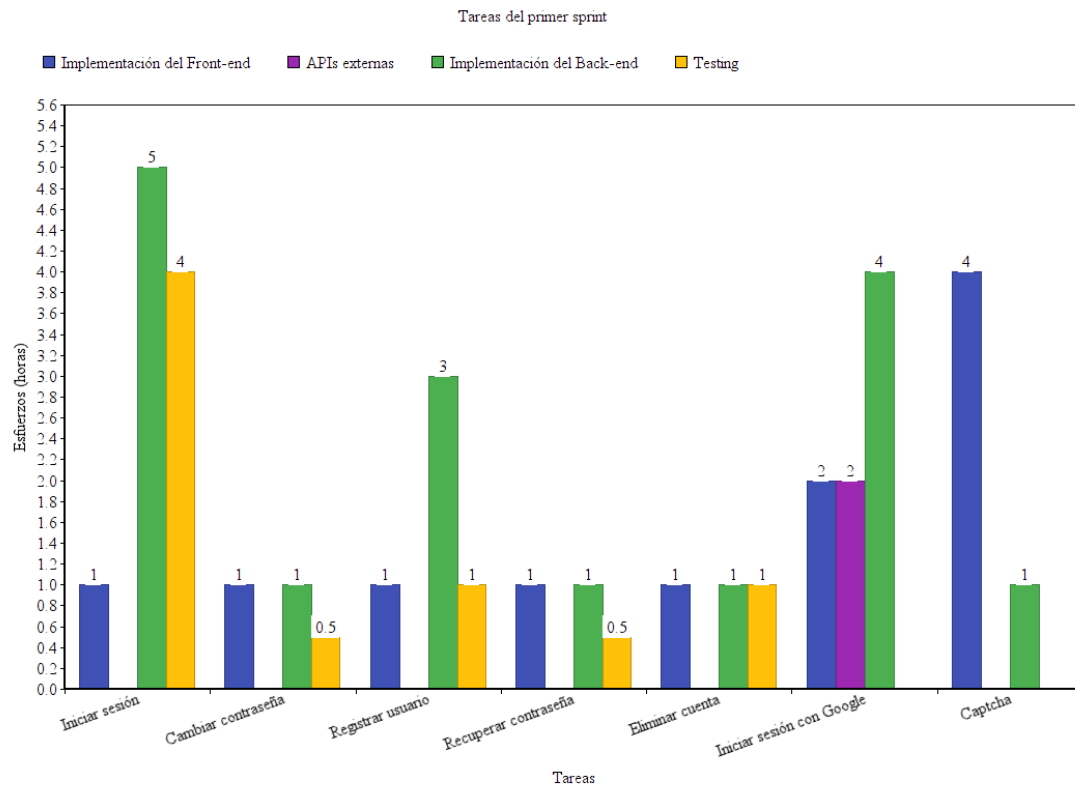


Figura 29. Esfuerzos dedicados por tarea y entrada de la pila en el primer sprint.

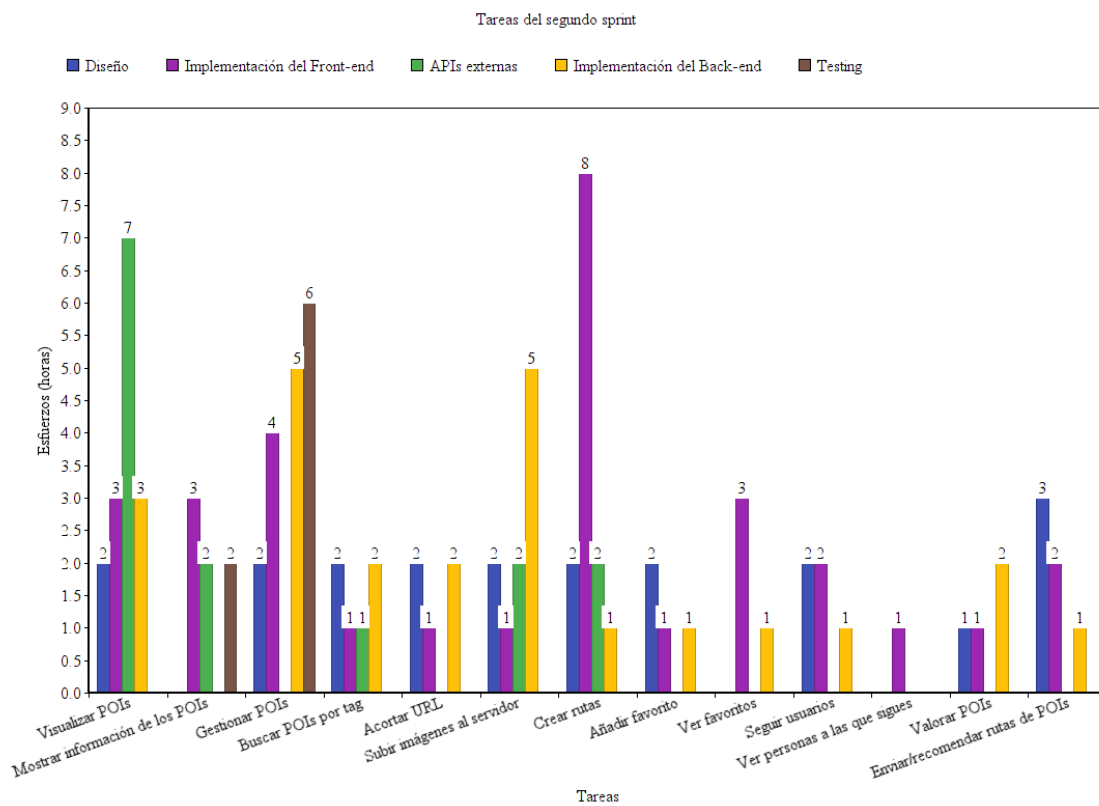


Figura 30. Esfuerzos dedicados por tarea y entrada de la pila en el segundo sprint.

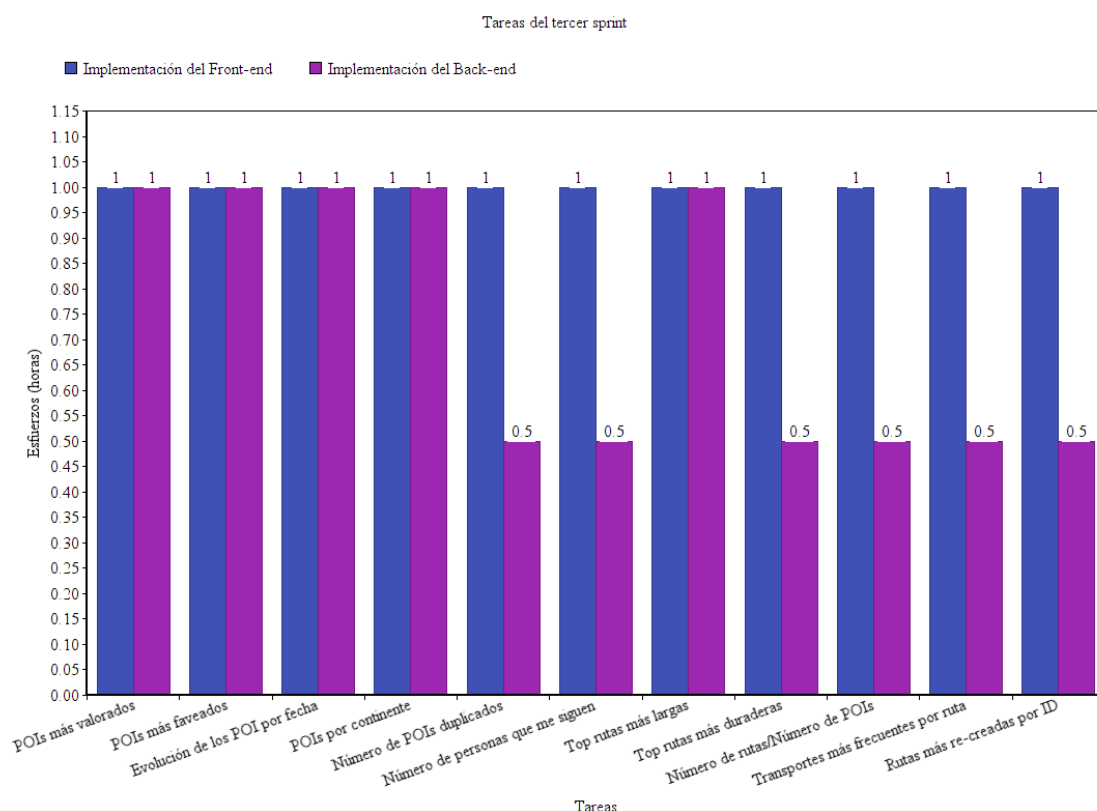


Figura 31. Esfuerzos dedicados por tarea y entrada de la pila en el tercer sprint.

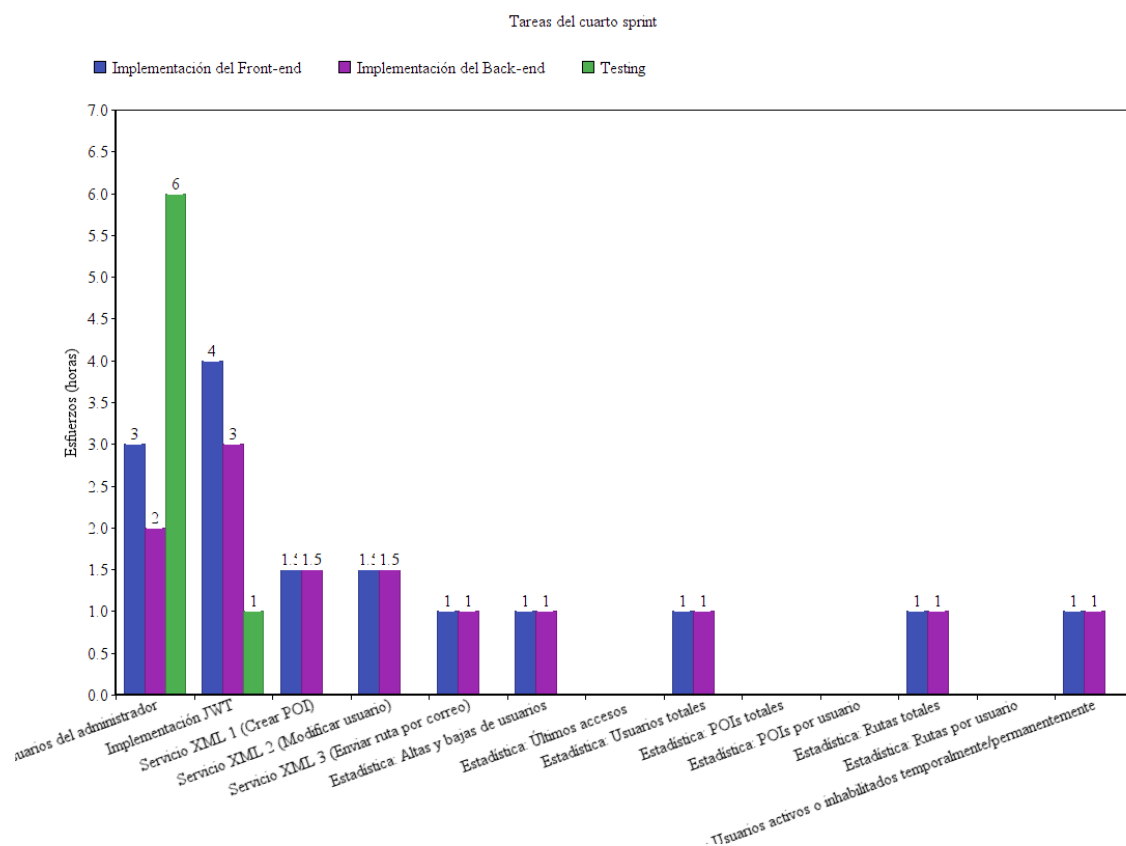


Figura 32. Esfuerzos dedicados por tarea y entrada de la pila en el cuarto sprint.

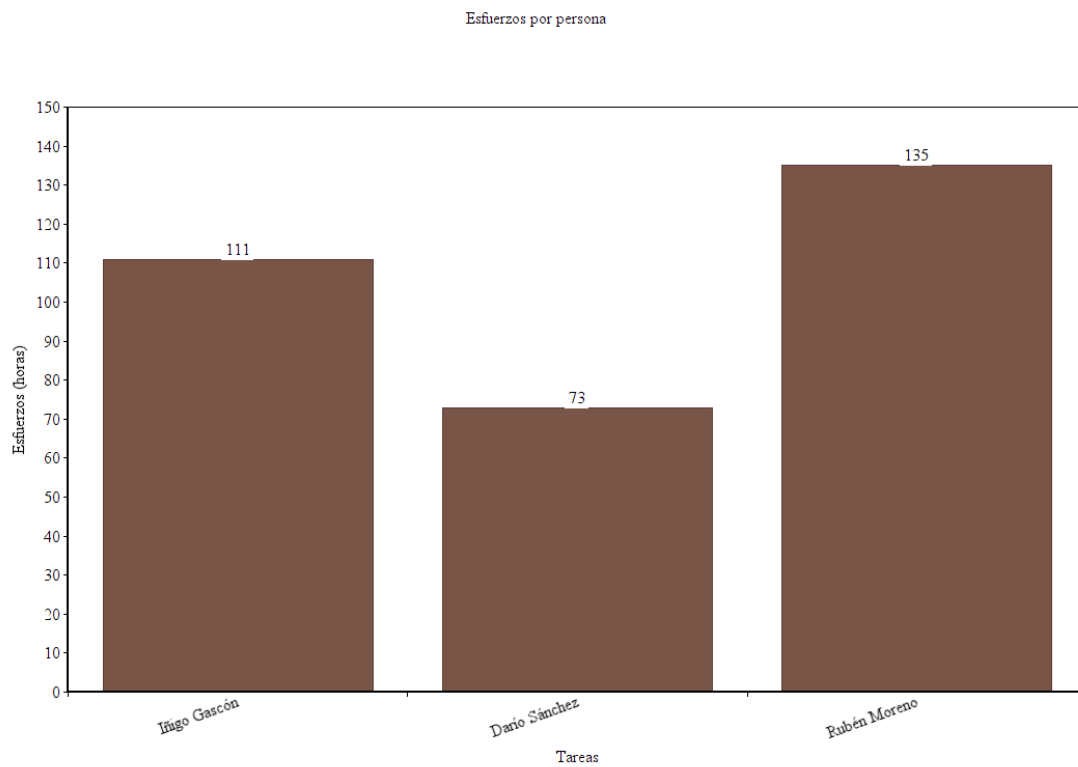


Figura 33. Esfuerzos dedicados por persona al proyecto.

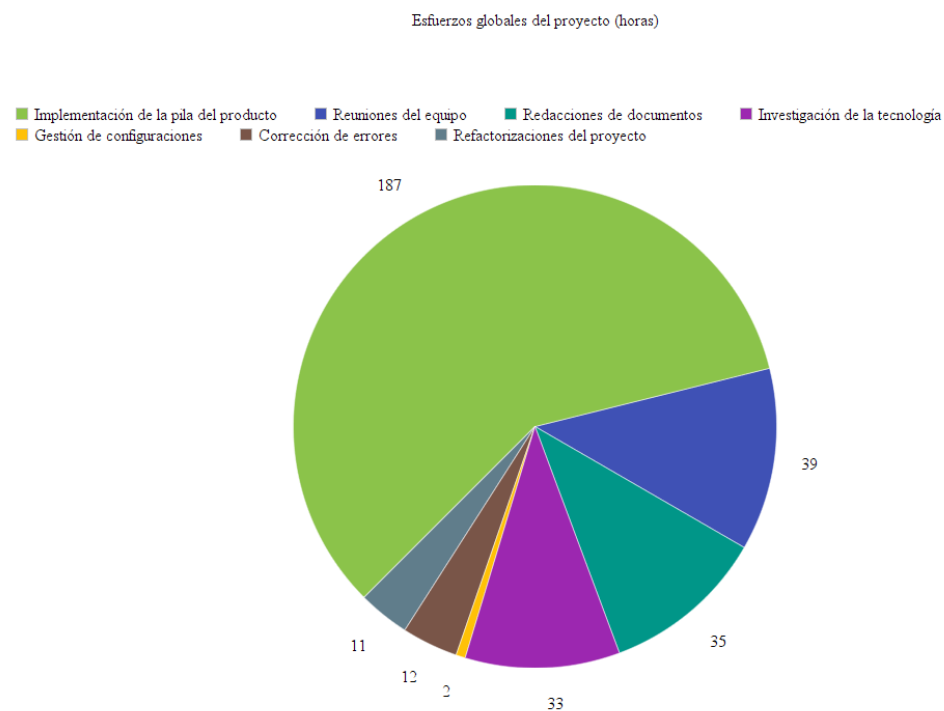


Figura 34. Desglose de esfuerzos globales del proyecto por actividades.

3.10 Diagramas de trabajo

Dado que se ha seguido una metodología Scrum a lo largo del proyecto para la organización y planificación del mismo, se van a presentar a continuación los siguientes diagramas de burnup que representan el trabajo realizado en cada uno de los 4 sprints en los que ha consistido el proyecto. Aunque dichos diagramas suelen representarse en puntos de historia por sprint, representando los sprints totales de un proyecto, en este documento se ha decidido realizar un diagrama de burnup por cada sprint, representando los puntos de historia de cada uno en función de las semanas que ha durado cada sprint. De esta forma, queda más claro y detallado el progreso a lo largo de todo el proyecto siendo que no es de duración muy larga y consiste en pocos sprints. Además, por supuesto, también se presentará el diagrama general del proyecto en puntos de historia por sprints totales, como se ha mencionado.

Comenzando con el primer sprint, puede observarse en la **Figura 35** que los puntos de historia (PH) planificados para este sprint suman un total de 22. No obstante, tan sólo llegaron a completarse 17 durante las 2 semanas en las que se alargó el sprint (no se cuentan semanas enteras, sino semanas del calendario, por lo que la primera semana incluye del 15 al 19 de marzo y la segunda del 20 al 24 del mismo mes). Los 5 PH restantes pertenecen a la entrada de la pila correspondiente con el Login con Google, el cual el equipo se vio obligado a retrasar a otro sprint por falta de tiempo.

Por otro lado, puede observarse que los 17 PH se completaron durante la segunda semana y ninguna durante la primera. Esto se debe a que todas las entradas de la pila contaban con sus correspondientes tests unitarios, y éstos se finalizaron durante la segunda semana, por lo que, de acuerdo a la definición de hecho acordada para este proyecto (y expuesta en este documento), no se dieron por finalizadas hasta entonces.

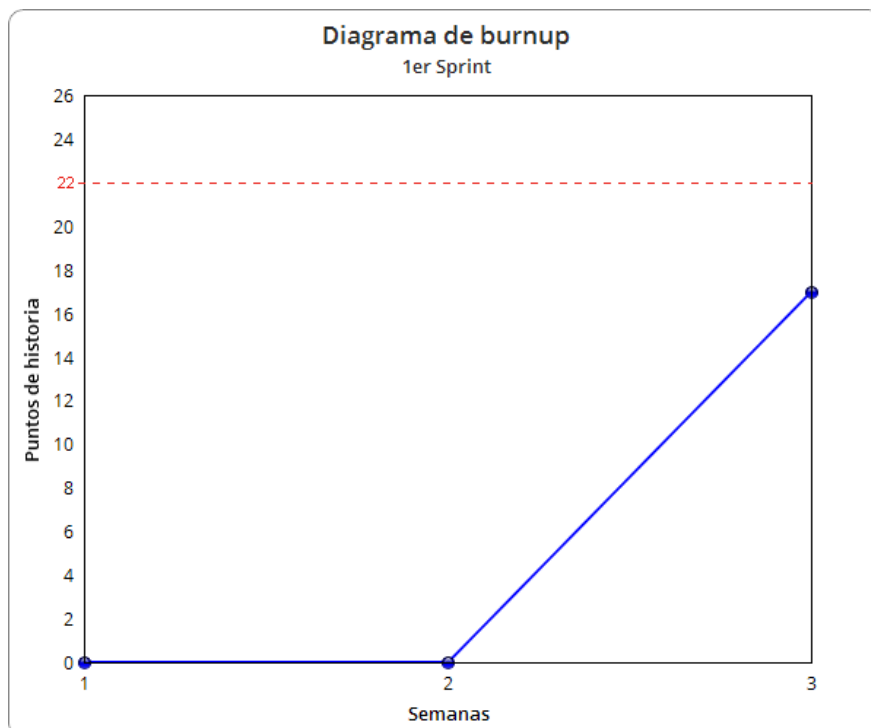


Figura 35: Diagrama de burnup del primer sprint.

Pasando a continuación al segundo sprint, tal y como se presenta en su correspondiente diagrama (**Figura 36**), el sprint cuenta con un total de 59 puntos de historia. El sprint se centraba en todas las funcionalidades referentes a los POIs y las rutas, por lo que fue el más largo e intenso del proyecto. Esto queda reflejado en el mencionado diagrama, que muestra que durante las 3 semanas de duración del sprint se fueron completando entradas de la pila de forma progresiva, finalizando primero 8 PH durante la primera semana, otros 26 más durante la segunda, y finalmente los 25 restantes durante la tercera semana, completando así los 59 puntos de historia planificados. En este caso las dos primeras semanas sí representan 14 días completos (del 3 al 9 de abril y del 10 al 16 de abril); sin embargo, la última semana representa únicamente 3 días (17-19 de abril). Por supuesto, no se finalizaron 25 PH en 3 días, sino que se fueron completando entradas de la pila que ya se habían empezado durante las semanas anteriores.

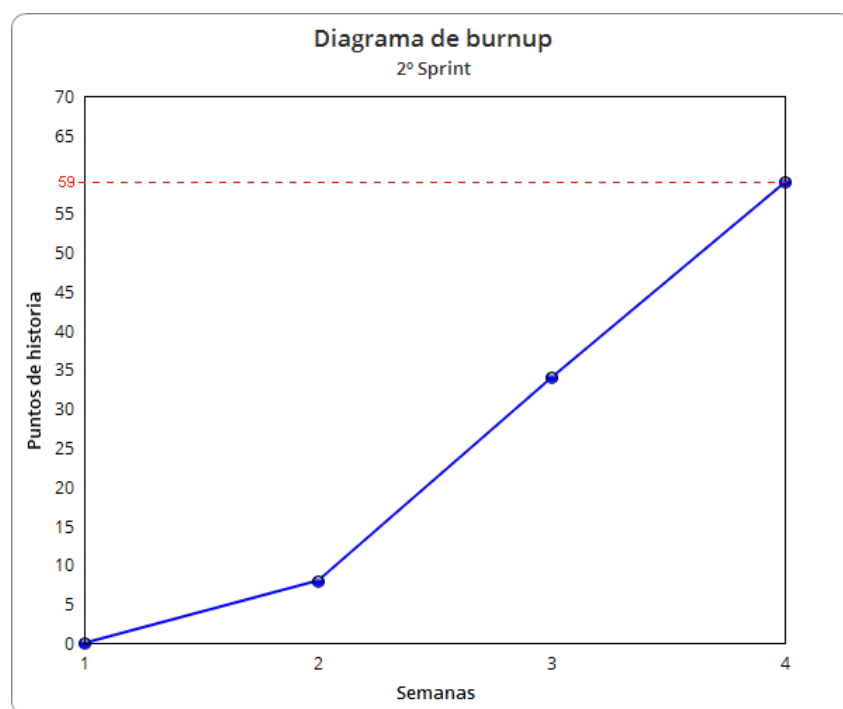


Figura 36: Diagrama de burnup del segundo sprint.

En lo referente al tercer sprint (**Figura 38**), que consta de 2 semanas cortas (19-23 abril y 24-26 abril), se completaron los 25 puntos de historia planificados para el sprint. En esta ocasión, de esos 25 PH, 5 eran los correspondientes la entrada de pila aplazada del primer sprint y que, por una alta carga de trabajo, tampoco pudo finalizarse durante el segundo sprint (no se incluyó en la planificación del segundo sabiendo que no podría completarse). De nuevo, todas las entradas se dieron por finalizadas durante la segunda semana, aunque se empezaron a desarrollar durante la semana anterior.

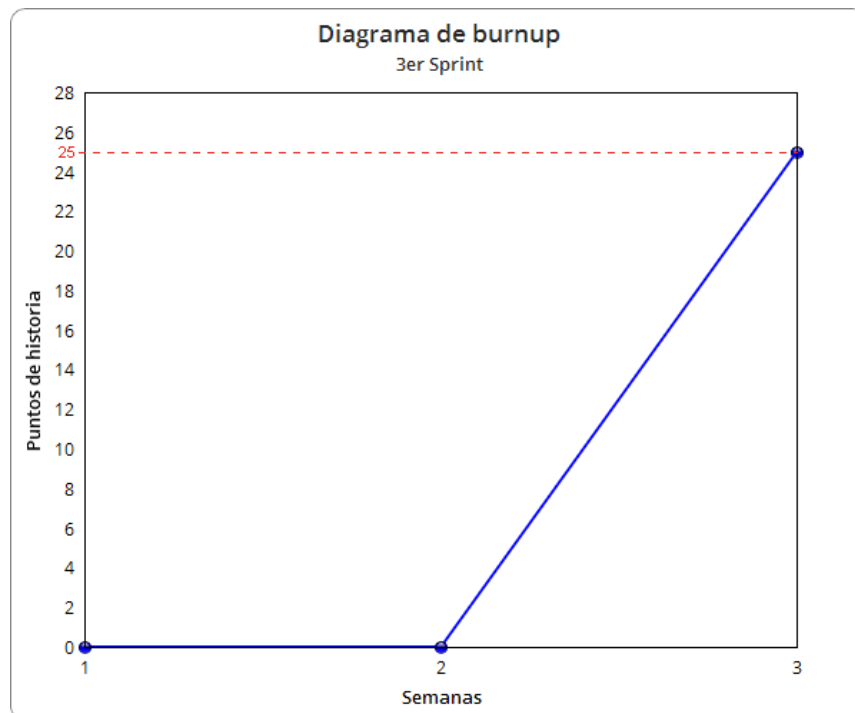


Figura 37: Diagrama de burnup del tercer sprint.

Y continuando con el cuarto y último sprint, el cual también cuenta con 25 PH, se completó a lo largo de 3 semanas, como puede observarse en la **¡Error! No se encuentra el origen de la referencia..** Durante la primera semana (27-30 abril) se comenzaron a desarrollar algunas de las entradas de la pila, pero no llegaron a completarse ninguna de ellas; durante la segunda (1-7 mayo), que sí era una semana entera de 7 días, se completaron todas las estadísticas del administrador, así como la implementación de los JSON Web Tokens; y, finalmente, durante la tercera semana (8-11 mayo) se completaron las funcionalidades del administrador para gestionar usuarios, que ya se habían implementado en las semanas anteriores pero estaban a la espera de completar los tests, y los 3 endpoints que debían contar con soporte XML.

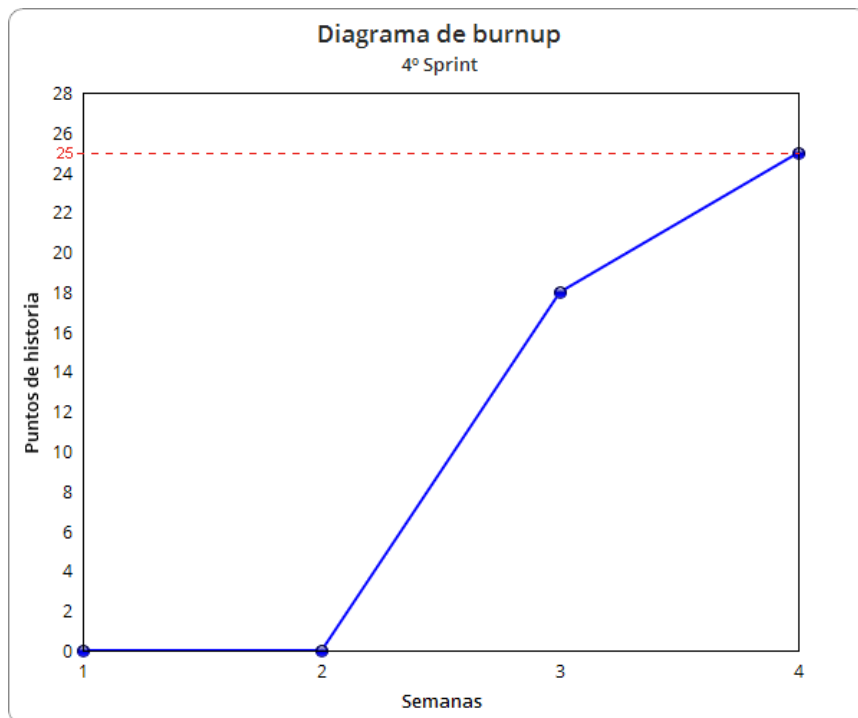


Figura 38: Diagrama de burnup del cuarto sprint.

Para finalizar, se presenta el diagrama de burnup del proyecto completo (**Figura 39**), el cual muestra los puntos de historia completados en cada sprint y cómo se ha ido avanzando de forma progresiva en cada uno (con una pendiente más marcada en el segundo sprint al ser el más complejo) hasta llegar a los 126 PH totales planificados.

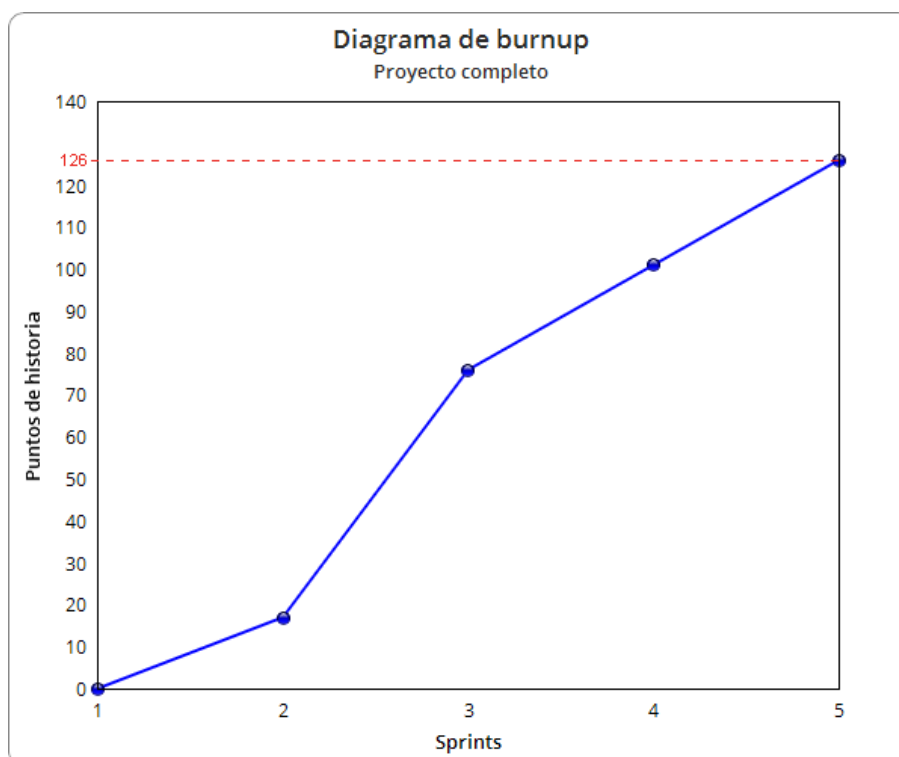


Figura 39: Diagrama de burnup del proyecto.

3.11 Velocidad del equipo

Habiendo realizado cuatro sprints a lo largo del proyecto, cada uno con un número de puntos de historia completados en un tiempo similar (de 2 a 3 semanas), es posible calcular la velocidad media del equipo en un sprint, representado en un rango de posibles puntos de historia a completar.

Para ello, se ha realizado la media de todos los puntos de historia completados en cada sprint, y a continuación se ha calculado la desviación típica en base a esos datos. Tal y como se ha expuesto en el punto anterior, los PH completados en cada sprint han sido 17, 59, 25 y 25, por lo que la media ha resultado en 31,5 y la desviación típica en 16,2. Una vez se conocen estos datos, se puede calcular una distribución normal al 95% de confianza para hallar el rango de puntos de historia en el que un equipo se moverá en cada sprint. Esta distribución se calcula como la media ya presentada \pm la desviación típica $\times 2$, con lo que obtenemos un rango de **[15,3, 47,7]** puntos de historia por sprint.

El motivo de que el rango haya resultado ser tan grande se debe a que uno de los cuatro sprints, concretamente el segundo, con 59 puntos de historia completados, se desvía bastante de la media. Podría tratarse como una excepción y emplear únicamente los datos de los otros 3 sprints, pero se ha considerado que era interesante incluirlo en este caso, dado que el rango obtenido es meramente informativo y no se va a tener en cuenta para futuros proyectos.

3.12 Planificación del proyecto

Debido a la clara diferenciación del proyecto en 4 partes (Usuarios, Funcionalidades principales, Estadísticas y Funcionalidades de administrador) se planteó desde el principio dividir el proyecto en 4 sprints, cada uno cubriendo dichas funcionalidades. Se realizó entonces la estimación de fechas. Debido a lo breve del tiempo del que se disponía, se planteó la división en sprints de 2 a 4 semanas, dejando un par de semanas al final para elaborar la documentación. Esta planificación resultó ser acertada y adecuada, y aunque en los primeros sprints fue un poco justo, los dos últimos se consiguieron realizar en la mitad del tiempo estimado. Cabe mencionar la excepción del Login de Google que, debido a un fracaso de implementación inicial, fue pospuesto hasta el tercer sprint, donde finalmente se consiguió una correcta implementación.



Figura 40: Diagrama de burnup del proyecto.

Leyenda:

- **Morado:** Primer sprint
- **Azul:** Segundo sprint
- **Verde:** Tercer Sprint
- **Naranja:** Cuarto Sprint
- **Rojo:** Desarrollo de la memoria final

3.13 Resultados de las retrospectivas

Teniendo en cuenta que la retrospectiva se ha realizado entre los tres miembros del equipo Scrum y, tomando como unidades de medida el 0, 1 y 2 (siendo el 0 un valor negativo y el 2 un valor positivo), se han valorado los siguientes aspectos del proyecto durante cada uno de los sprints:

- Se ha observado que el proyecto cuenta con la existencia de suficientes test unitarios automáticos. Desde un principio gran parte del código que se ha desarrollado ha sido respaldado por sus respectivos tests, logrando una cobertura actual del 45%. Aunque esta cobertura está bien, podría ser mejor, por lo tanto, el equipo decidió valorar esta directiva con un 5/6.
- El diseño de la aplicación se considera excelente e intuitivo en todo momento. Las experiencias de usuarios externos al equipo desarrollador han sido fluidas, por lo tanto, se valora este punto con un 6/6.
- La métrica de una correcta y eficiente refactorización ha obtenido la máxima puntuación (6/6) puesto que se considera que globalmente, debido a las múltiples iniciativas de refactorización tanto del código como de la documentación, se han conseguido resultados satisfactorios que de otra manera no habrían sido posibles.

- Aunque se ha realizado un esfuerzo por intentar integrar a los miembros del equipo en los distintos apartados y tecnologías del proyecto, no ha resultado demasiado exitoso, por lo tanto se ha valorado la propiedad colectiva con un 3/6.
- En cuanto a la calidad de la documentación del diseño arquitectural, se ha obtenido un 5/6 puesto que los diagramas se han diseñado adecuadamente, en paralelo al desarrollo del código tanto para un mejor entendimiento del código como para detectar fallos arquitecturales.
- La métrica que indica la existencia y calidad de scripts para automatización del build ha obtenido un 5/6. La disponibilidad de lanzamiento y configuración automáticas del servicio es casi completa y requiere una configuración mínima, por ello ha obtenido una puntuación elevada.
- El trabajo en control de versiones ha resultado satisfactorio y no ha habido ningún problema durante el desarrollo, por lo tanto, se ha obtenido un 6/6.
- Al estar trabajando con Scrum, una métrica muy interesante es la de calidad y claridad de las historias de usuario. Se es consciente de que inicialmente se desconocía el alcance del proyecto y a consecuencia de lo anterior esta métrica ha obtenido un 6/8.
- En cuanto a la estimación inicial de entradas de la pila, en los 4 sprints realizados resulta ser lo suficientemente acertada y cercana al esfuerzo final, se puntúa con un 5/6.
- Relacionada directamente con la métrica anterior, la estimación inicial de las tareas ha obtenido una puntuación de 5/6, por la misma razón que el punto anterior.
- La métrica referente a un ritmo de trabajo uniforme y repartido entre los integrantes ha obtenido una buena puntuación (4/6) puesto que se considera que el trabajo ha sido desarrollado de manera suficientemente uniforme, aunque en algunos casos algunos de los miembros del equipo se quedó ligeramente atrás. Aun así, se cumplen todas las fechas límite, incluso se han adelantado algunas. Esto se ve reflejado en la puntuación de la métrica.
- La visibilidad del proyecto para todos los miembros del mismo ha sido valorada positivamente, debido a la automatización y sencillez del proceso de despliegue, todos los miembros del equipo han sido capaces de seguir el estado de la aplicación en todo momento. Por ello se evalúa con un 6/6.
- En cuanto a la existencia, calidad y uso de un canal de comunicación, todos los miembros del equipo han concluido que tanto el grupo de Whatsapp del que disponía el equipo como el hecho de que los miembros del equipo coinciden físicamente todos los días en la universidad han hecho posible que la comunicación se lleve a cabo de una manera fácil y efectiva. No obstante, no se ha obtenido la máxima puntuación puesto que hubo ocasiones en las que las comunicaciones online han sido pasadas por alto u olvidadas. Por ello, se ha valorado con un 5/6.
- Por último, la existencia, calidad y uso de herramientas Scrum ha sido valorada con la máxima puntuación (6/6) ya que el proceso Scrum se ha seguido rigurosamente. Además, para todas las fases del mismo se ha empleado el proyecto de Github y se han realizado reuniones al comienzo de todos los sprints .

3.14 Medidas que se llevarán a cabo para mejorar

Analizando los resultados del apartado anterior, durante la reunión de retrospectiva se decidieron las siguientes medidas para mejorar la calidad del proyecto:

Propiedad colectiva:

Se es consciente de que los miembros están especializados cada uno en una parte del proyecto y, dada la naturaleza del mismo, donde lo más adecuado es el modelo que se está siguiendo, no se proponen mejoras para solucionarlo.

Ritmo de trabajo:

Aunque no ha sido un problema realmente acuciante debido al cumplimiento e incluso adelanto de las fechas, el ritmo de trabajo no ha sido realmente uniforme. La razón detrás de esta varianza de ritmos es la diferente cantidad de horas de clase y otros proyectos de los miembros del grupo, por lo tanto, no se propone ninguna mejora para ello, debido a que son en su mayoría causas externas y se considera que en un entorno profesional cerrado este ritmo sería correcto en su totalidad.

Comunicación:

La existencia únicamente del grupo de WhatsApp para la comunicación del equipo ha provocado que, en caso de caída, el equipo se queda parcialmente incomunicado. Además, si se realizaba un comunicado y posteriormente se discutía la existencia de algún problema, el comunicado se quedaba perdido en el limbo. Por ello se debería establecer otro canal de comunicación para comunicados importantes y charla seria, delegando el off-topic y consultas rápidas a WhatsApp. Opciones válidas para ello son Slack o Telegram.

4. Conclusiones

4.1 Conclusiones del proyecto

Se ha realizado el proyecto satisfactoriamente en el tiempo dado, incluyendo todas sus características, e incluso se han implementado la mayoría de las funcionalidades extra del guion, y el extra de la herramienta de SonarQube que no estaba incluida en el mismo. Por ello, el proyecto es considerado un éxito por parte del equipo.

Sin embargo, también se enfrentaron retos desde el principio. El principal de ellos ha sido trabajar con la API de Google, reto que fue mayormente solucionado al usar el módulo angular-google-maps, que permite realizar las operaciones de la API de Google directamente en el Front-end. Otro de los retos ha sido enfrentarse a algunas tecnologías nuevas que el equipo desconocía, como son Swagger, ChartJS y los frameworks de testing de Mocha y Protractor.

Se concluye, pues, que ha sido un proyecto que ha proyectado retos de diversas magnitudes, y que el equipo ha sabido solventarlos, todo esto siempre dentro del estrecho margen de tiempo del que se disponía.

4.2 Valoración personal del equipo

Como grupo estamos muy satisfechos con el trabajo realizado y con la aplicación resultante de dicho trabajo. Consideramos, como equipo en conjunto, que la propuesta de este trabajo tiene tanto puntos positivos como negativos, y otros que sería interesante añadir de cara a proyectos en años futuros. Vamos a comentar a continuación algunos de esos puntos, sin entrar en mucho detalle, pero para dejar constancia de nuestras opiniones que esperemos sean constructivas.

Empezando por aquello que hemos considerado interesante y positivo en el proyecto, nuestra opinión se centra principalmente en el uso del MEAN Stack para el desarrollo de la aplicación. Es un Stack que la mayoría no hemos tocado al completo en ningún momento de la carrera, o al menos no por requerimientos de ninguna asignatura, y que resulta muy interesante de aprender, principalmente por el hecho de que actualmente está en auge y la probabilidad de que a alguno nos toque programar con estas tecnologías, o con Javascript en general, es bastante alta. Además, aquellos que ya hemos tenido que programar aplicaciones web durante la carrera ha sido con tecnologías centradas en lo que ya conocíamos, es decir, los servlets de Java y, como abstracción de estos, el framework Spring. Si bien se nos daba libertad en algunas asignaturas para usar otras tecnologías, el trabajo que suponía aprenderlas no se veía recompensado en la nota final, pues el objetivo era construir una aplicación y no aprender nuevas tecnologías. Contar con una introducción básica al MEAN Stack durante las prácticas es un empujón muy bienvenido para empezar a aprender y realizar la aplicación con estas nuevas tecnologías para nosotros.

Pasando ahora a aquellas partes que nos han gustado menos o no nos ha parecido que aportasen cosas tan interesantes para el proyecto, son las siguientes. Principalmente, el problema lo hemos encontrado en que algunos de los conceptos planteados en el proyecto se solapan con otras asignaturas, especialmente de la especialidad de Ingeniería del Software. Un ejemplo de este solapamiento podrían ser los requisitos en los que se pide que se implementen varios conceptos de red social, como seguir a otros usuarios, marcar POIs como favoritos o valorarlos. Durante la asignatura de Arquitectura del Software se nos pidió realizar como proyecto una red social, por lo que hacerlo de nuevo en este proyecto sólo suponía aumentar innecesariamente las horas de dedicación necesarias al trabajo. Por tanto, dado que ya sabíamos implementar ese tipo de funcionalidades, no aprendíamos nada nuevo y, además, no aumentaba la complejidad del proyecto, tan sólo las horas de implementación que podríamos dedicar a aprender cosas nuevas más interesantes.

Por último, como cosas que consideramos que podrían ser interesantes para añadirlas de cara a proyectos en años futuros, creemos que hacer más énfasis en la parte de tests sería realmente bueno para todos los que cursen la asignatura. Si bien en Ingeniería de Software se tiene una asignatura específica para ello, ir probando diferentes entornos de testing resulta interesante y muy enriquecedor. Por supuesto, para todos aquellos que no conocen la parte de pruebas tan bien, aunque suponga un poco más de esfuerzo creemos que es algo que les resultará muy útil en su carrera. Además, valorar estos tests en función de la cobertura de código que proporcionan creemos que sería una métrica interesante a la hora de valorar los tests realizados. Obligar a realizar la documentación con Swagger también consideramos que es algo que podría resultar muy productivo para los equipos de trabajo. Tener una API bien

definida ayuda mucho a tener las ideas claras y desarrollar el proyecto, además de que el hecho de que te permita probar el back-end con peticiones que cumplen esa misma API es extremadamente útil (aunque en este apartado la herramienta sea un poco peculiar), ya que hay gente reacia a emplear otras herramientas como Postman o similares. Finalmente, plantear la introducción de herramientas como SonarQube, que introduce y facilita estadísticas de Quality Assurance, nos parece también una perspectiva muy interesante, ya que más allá de lo poco que se explica en algunas asignaturas por encima y en Metodologías Ágiles (que, además, es optativa), no se hace hincapié en esa característica del software que consideramos importante. Además, SonarQube tiene una buena integración con Docker, que sí se cuenta en la asignatura y podría encontrarse una forma de integrarla (o al menos darle una pincelada) en la práctica 9.

4.3 Valoración personal de cada integrante

Darío Sánchez:

Ha resultado un proyecto muy interesante y educativo por mi parte, ya que no había utilizado todavía tecnologías de stack MEAN. Mi equipo, conociendo esto, ha realizado un esfuerzo para tratar de integrarme un poco en cada sitio, por lo que he realizado tareas tanto de front-end como de back-end, lo cual me ha servido para aprender a manejar tanto NodeJS como AngularJS, además de tecnologías y varias cosas complementarias más. Por ello, valoro muy positivamente el proyecto, aunque quizá hubiera estado bien algo más de tiempo.

Iñigo Gascón

Coincidiendo con lo dicho por mí y mis compañeros en la valoración del equipo, ha sido muy interesante aprender las tecnologías que conforman un framework como el MEAN Stack de cara a un posible futuro profesional. El trabajo se ha hecho duro por el tiempo limitado con el que se contaba y por la exigencia de los requisitos que se pedían, pero una buena organización por parte del equipo ha hecho posible sacarlo adelante y con unos resultados muy positivos. Me llevo la experiencia de tener una base bastante sólida (aunque claramente ampliable, pues es una base) de tecnologías como Node.js, MongoDB y Express.js, además de haber programado una aplicación web íntegramente en Javascript.

Rubén Moreno

Me ha resultado interesante trabajar con el Stack MEAN, al cual le tenía bastante curiosidad y ganas. Aunque contaba con conocimientos previos de algunas tecnologías como AngularJS, me ha resultado interesante poder comparar los puntos de vista de algunos aspectos y/o ver en cuales coincidía y cuales estaba haciendo bien o mal (pues la experiencia que tenía era de haber seguido cursillos/tutoriales y experimentación propia).

Creo que tanto todo el equipo como yo nos sentimos bastante contentos con el resultado final de la aplicación, ya que ha llevado mucho tiempo implementarla, pero el resultado ha valido la pena. Aunque la implementación ha sido una completa carrera dado que en un primer momento se disponía prácticamente de un mes para implementarla, el resultado (gracias a la experiencia previa del equipo en este tipo de proyectos y una buena organización y planificación) se habría podido terminar incluso un poco antes de la fecha original de

entrega, con lo cual estamos bastante contentos tanto del proceso interno seguido como del resultado final.

Además, personalmente me ha sorprendido el hecho de que se valore un mínimo de usabilidad en la página como requisito de la misma (se pide que todo sea mínimamente responsive), ya que es algo que nunca se ha pedido en ninguna asignatura de la carrera, un mínimo de calidad final del producto de cara al usuario final. Me gusta mucho el desarrollo de Front-end, y siempre intento esforzarme en ese tipo de resultados finales, que luego en la vida real, son aspectos muy importantes y que pueden superar el éxito o fracaso de una aplicación, y sin embargo, nunca se ha valorado para el resto de trabajos o asignaturas en la carrera.

5. ANEXO I: API RESTful del sistema

POST

/users/

Crear usuario (Sign Up)

Creando un nuevo usuario en el sistema, comprueba que no se trate de un bot con el recaptcha de google y manda un correo electrónico al parámetro [email] con un link para confirmar la creación de la cuenta.

Try it out

Parameters

Name	Description
name <small>required</small> string (body)	Nombre del usuario.
lastname <small>required</small> string (body)	Apellido del usuario.
email <small>required</small> string (body)	Email del usuario que sirve como identificador.
captcha <small>required</small> string (body)	Key del recaptcha de google.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div>FeedbackMessage { description: Mensaje de feedback que se devuelve al usuario en caso de error o acierto en una determinada operación. success: boolean required: true True si la operación ha ido con éxito. False si ha habido algún error. message: string required: true Mensaje que describe el resultado de una operación. }</div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div>{ "success": true, "message": "string" }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div>{ "success": true, "message": "string" }</div>

POST

/users/google

Crear usuario con cuenta de Google

Crear un nuevo usuario en el sistema usando los datos de su cuenta de google

Try it out

Parameters

Name	Description
code * <small>required</small> string (body)	Codigo para pedir datos a google
clientId * <small>required</small> string ()	API key publico de nuestra app
redirectUri ()	Direccion de enrutamiento para la peticion a google (predefinido)

Responses

Response content type: application/json

Code	Description
200	<div>Informacion de perfil de usuario.</div> <div>Example ValueModel</div> <div><pre>{ "email": "string", "name": "string", "lastname": "string", "admin": true, "firstlogin": true, "favs": ["string"], "follows": ["string"]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/users/login

Iniciar sesión

End-point para iniciar sesión en el sistema. El usuario pasa los credenciales de la cuenta siguiendo el estándar de base64.

Parameters

Try it out

Name	Description
Authorization <small>* required</small> string <i>(header)</i>	Base64 estándar: Authorization: Basic + base64.encode(user:password).

Responses

Response content type: application/json

Code	Description
200	<div>Información de perfil del usuario metida dentro de un JSON Web Token.</div> <div>Example ValueModel</div> <div><pre>{ "token": { "email": "string", "name": "string", "lastname": "string", "admin": true, "firstLogin": true, "favs": ["string"], "follows": ["string"] }}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

PUT

/users/retrievePass Recuperar contraseña

Genera una nueva contraseña aleatoria para el usuario y la manda por correo electrónico para que el usuario pueda acceder al sistema si se ha olvidado de su contraseña anterior.

Parameters

Try it out

Name	Description
email <small>* required</small> string <small>(body)</small>	Email del usuario que sirve como identificador.
captcha <small>* required</small> string <small>(body)</small>	Key del recaptcha de google.

Responses

Response content type application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/users/confirm/{email} Confirmar contraseña

Confirma la cuenta de un usuario creando una nueva contraseña aleatoria y pasándosela al usuario por correo electrónico.

Parameters

Try it out

Name	Description
email <small>* required</small> string <small>(path)</small>	Email del usuario que sirve como identificador.

Responses

Response content type application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

PUT

/users/{email}/fav

Añadir POI a favoritos

Añade el POI indicado a la lista de POIs favoritos del usuario.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <i>(path)</i>	Email del usuario que sirve como identificador.
poId <small>required</small> string <i>(body)</i>	ID del POI que se desea añadir como favorito.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

PUT

/users/{email}/follow

Seguir a un usuario

Añade un nuevo usuario a la lista de usuarios a los que sigue.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <i>(path)</i>	Email del usuario que sirve como identificador.
userToFollowEmail <small>required</small> string <i>(body)</i>	Email del usuario al que se va a seguir que sirve como identificador.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

PUT

/users/{email} Cambiar contraseña

Cambia la contraseña de un usuario determinado.

Parameters

Try it out

Name	Description
Authorization * <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
email * <small>required</small> string <small>(path)</small>	Email del usuario que sirve como identificador.
current * <small>required</small> string <small>(body)</small>	Contraseña actual del usuario.
new * <small>required</small> string <small>(body)</small>	Contraseña nueva del usuario.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>

DELETE

/users/{email} Borrar usuario

Borra la cuenta de usuario.

Parameters

Try it out

Name	Description
Authorization * <i>required</i> string (header)	JWT estándar: Authorization: Bearer + JWT.
email * <i>required</i> string (path)	Email del usuario que sirve como identificador.
current string (body)	Contraseña actual del usuario.
google string (body)	Booleano que indica si es cuenta de google

Responses

Response content type application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>

GET
/users/{email}
Pedir usuario de google

Devuelve la cuenta de usuario registrado con google.

Parameters
Try it out

Name	Description
Authorization <small>required</small> string (header)	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string (path)	Email del usuario que sirve como identificador.
token <small>required</small> string (body)	Token identificador de google

Responses
Response content type
application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

GET
/pois/
Obtener POIs

Devuelve una lista con todos los POIs en el sistema.

Parameters
Try it out

Name	Description
Authorization <small>required</small> string (header)	JWT estándar: Authorization: Bearer + JWT.

Responses
Response content type
application/json

Code	Description
200	<div>Lista con todos los POIs del sistema.</div> <div>Example Value Model</div> <pre>{ "pois": [{ "_id": "string", "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "owner": "string", "image": "string", "url": "string" }] }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

POST

/pois/ Crear POI

Crea un nuevo POI en el sistema.

Try it out

Parameters

Name	Description
Authorization required string (header)	JWT estándar: Authorization: Bearer + JWT.
userEmail required string (body)	Email del usuario que sirve como identificador.
poi required (body)	Información del POI que se va a añadir <div> <div>Example Value</div> <div>Model</div> <div> <pre>{ "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "url": "string", "image": "string" }</pre> </div> </div> <div> <div>Parameter content type</div> <div>application/json</div> </div>

Responses

Response content type application/json

Code	Description
200	<div>El POI que se acaba de crear.</div> <div> <div>Example Value</div> <div>Model</div> <div> <pre>{ "poi": { "_id": "string", "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "owner": "string", "image": "string", "url": "string" } }</pre> </div> </div>
404	<div>Mensaje de feedback para el usuario.</div> <div> <div>Example Value</div> <div>Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div> </div>
500	<div>Mensaje de feedback para el usuario.</div>

GET

/pois/filter/

Buscar POIs por tags

Busca los POIs que contengan los tags que se han indicado y devuelve una lista con ellos.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
tags <small>required</small> string <small>(header)</small>	Conjunto de tags separados por un '#'.

Responses

Response content type: application/json

Code	Description
200	<div>Liste con todos los POIs del sistema.</div> <div>Example Value Model</div> <div><pre>{ "pois": [{ "id": "string", "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "owner": "string", "image": "string", "url": "string" }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

POST

/pois/{id} Duplicar un POI

Duplica un POI existente y lo añade a los POIs del usuario. Al POI duplicado se le añade al nombre + "_Duplicado" (p.ej., nombre -> nombre_Duplicado).

Parameters

Try it out

Name	Description
Authorization required string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
id required string <i>(path)</i>	ID del POI que se va a duplicar.
userEmail required string <i>(body)</i>	Email del usuario que va a obtener el duplicado del POI y que sirve como identificador.

Responses

Response content type application/json

Code	Description
200	<p><i>El POI que se acaba de duplicar.</i></p> <div>Example Value Model</div> <pre>{ "poi": { "_id": "string", "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "owner": "string", "image": "string", "url": "string" } }</pre>
404	<p><i>Mensaje de feedback para el usuario.</i></p> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<p><i>Mensaje de feedback para el usuario.</i></p> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

DELETE

/pois/{id} Eliminar un POI

Elimina un POI existente del usuario, incluida la imagen adjunta, si la hay, y la URL acortada, si se ha acortado.

Parameters

Try it out

Name	Description
Authorization required string (header)	JWT estándar: Authorization: Bearer + JWT.
id required string (path)	ID del POI que se va a eliminar.
userEmail required string (body)	Email del usuario propietario del POI que se va a eliminar.

Responses

Response content type application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

PUT

/pois/{id} Modificar POI

Modifica un POI existente del usuario con nueva información.

Parameters

Try it out

Name	Description
Authorization required string (header)	JWT estándar: Authorization: Bearer + JWT.
id required string (path)	ID del POI que se va a eliminar.
userEmail required string (body)	Email del usuario propietario del POI que se va a modificar.
poi required (body)	Información que se va a editar en el POI. <div><div>Example ValueModel</div><pre>{ "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "url": "string"} </pre><div>Parameter content type<div>application/json</div></div></div>

Responses

Response content type

application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div><div>Example ValueModel</div><pre>{ "success": true, "message": "string"} </pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div><div>Example ValueModel</div><pre>{ "success": true, "message": "string"} </pre></div>
500	<div>Mensaje de feedback para el usuario.</div>

GET

/pois/{id}

Obtener POI

Devuelve el POI identificado con la id dada.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content type: application/json

Code	Description
200	<div>Objeto POI.</div> <div>Example Value Model</div> <pre>{ "poi": { "name": "string", "description": "string", "tags": "string", "lat": 0, "long": 0, "url": "string", "image": "string" } }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

PUT

/pois/{id}/rate

Valorar un POI

Añade la valoración indicada a la lista de valoraciones del POI.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
id <small>required</small> string <small>(path)</small>	ID del POI al que se va a añadir la valoración.
rating <small>required</small> integer <small>(body)</small>	Valoración que se va a añadir al POI

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

POST

/routes/{id}/sendRoute/

Enviar email con un ruta

Envía un email a la dirección indicada con el ID de una ruta para poder introducirla y mostrarla en la app.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
id <small>required</small> string <i>(path)</i>	Id de la ruta.
ownerEmail <small>required</small> string <i>(body)</i>	Email del usuario que envía la ruta (y que la ha creado).
receiverEmail <small>required</small> string <i>(body)</i>	Email del usuario al que se le envía la ruta.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>

POST

/routes/ Guardar una ruta

Se guarda una nueva ruta creada por un usuario junto con toda la información relevante que se considere guardar con fines estadísticos.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(body)</i>	Email del usuario creador de la ruta.
travelMode <small>required</small> string <i>(body)</i>	Modo de transporte de la ruta. Valor discreto entre: 'DRIVING' OR 'WALKING' OR 'BICYCLING' OR 'TRANSIT'
routePOIs <small>required</small> array [string] <i>(body)</i>	Lista de los IDs de los POIs que componen la ruta para poder reproducirla.
routeInfo <small>required</small> array [object] <i>(body)</i>	Una lista que contiene información asociada a la ruta. Por cada tramo de la misma, su duración y distancia.

Responses

Response content type application/json

Code	Description
200	<div><div>ID de la ruta creada.</div><div>Example Value Model</div><div><pre>{ "routeID": "string"}</pre></div></div>
404	<div><div>Mensaje de feedback para el usuario.</div><div>Example Value Model</div><div><pre>{ "success": true, "message": "string"}</pre></div></div>
500	<div><div>Mensaje de feedback para el usuario.</div><div>Example Value Model</div><div><pre>{ "success": true, "message": "string"}</pre></div></div>

GET

/routes/{id}/

Listar POIs y modo de viaje de una ruta

Lista todos los POIs que conforman una ruta, así como el modo de viaje de la misma.

Try it out

Parameters

Name	Description
Authorization * <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
id * <small>required</small> string <small>(path)</small>	Id de la ruta.

Responses

Response content type: application/json

Code	Description
200	<div>Datos de la ruta solicitada.</div> <div>Example ValueModel</div> <div><pre>{ "travelMode": "string", "routePOIs": [{ "_id": "string", "name": "string", "description": "string", "tags": "string", "lat": 0, "lng": 0, "owner": "string", "image": "string", "url": "string" }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

POST

/url/ Acortar una URL

Crea una URL acortada a partir de la URL que se proporciona y la devuelve.

Parameters

Try it out

Name	Description
Authorization <small>* required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
url <small>* required</small> string <i>(body)</i>	URL que se quiere acortar.

Responses

Response content type application/json

Code	Description
200	<div>URL acortada.</div> <div>Example Value Model</div> <div><pre>{ "urlshort": "string"}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

POST

/url/{id} Redirige a la URL correspondiente a la URL acortada

Mapea la URL acortada (el id en base 58) en busca de su URL real y redirige a esa dirección.

Parameters

Try it out

Name	Description
id <small>* required</small> string <i>(path)</i>	ID codificado en base 58 de la URL acortada

Responses

Response content type application/json

Code	Description
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/stats/{email}/mostRated

POIs mejor valorados

Obtiene una lista de los 5 POIs mejor valorados del usuario.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div><div>Lista de los POIs más valorados</div><div>Example ValueModel</div><div><pre>{ "pois": [{ "name": "string", "rating": 0 }] }</pre></div></div>
404	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel</div><div><pre>{ "success": true, "message": "string" }</pre></div></div>
500	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel</div><div><pre>{ "success": true, "message": "string" }</pre></div></div>

GET

/stats/{email}/mostFavorite

POIs con más favoritos.

Obtiene una lista de los 5 POIs que más favoritos han obtenido.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <small>(path)</small>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div><div>Lista de los POIs con más favoritos.</div><div>Example ValueModel</div><div><pre>{ "pois": [{ "name": "string", "favNumber": 0 }]}</pre></div></div>
404	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel</div><div><pre>{ "success": true, "message": "string"}</pre></div></div>
500	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel</div><div><pre>{ "success": true, "message": "string"}</pre></div></div>

GET

/stats/{email}/poiByDate

POIs creados en el último año.

Obtiene una lista de todos los POIs creados en el último año.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista de los POIs creados en el último año.</div> <div>Example ValueModel</div> <div><pre>{ "pois": [{ "name": "string", "creationDate": 0 }] }</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>

GET

/stats/{email}/poiByLocation

POIs creados según continente.

Obtiene una lista de todos los POIs creados en cada continente.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <small>(path)</small>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista de los POIs creados en cada continente.</div> <div>Example ValueModel</div> <div><pre>{ "pois": [{ "continent": "string", "poiNumber": 0 }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/stats/{email}/longestRoutes

Lista de las rutas más duraderas.

Obtiene una lista de las 5 rutas más duraderas creadas por el usuario.

Parameters

Try it out

Name	Description
Authorization <small>* required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>* required</small> string <small>(path)</small>	Email del usuario propietario de los POIs.

Responses

Response content typeapplication/json

Code	Description
200	<div>Lista de las rutas más duraderas.</div> <div>Example ValueModel</div> <pre>{ "routes": [{ "routeId": "string", "duration": 0 }] }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/stats/{email}/followers

Número de seguidores

Número de seguidores del usuario.

Parameters

Try it out

Name	Description
Authorization <small>* required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>* required</small> string <small>(path)</small>	Email del usuario propietario de los POIs.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número de seguidores del usuario.</div> <div>Example ValueModel</div> <pre>{ "followers": 0 }</pre>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/stats/{email}/duplicatedPois

Lista de los POIs más duplicados.

Obtiene una lista de los 5 POIs del usuario más duplicados.

Parameters

Try it out

Name	Description
Authorization required string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail required string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista de los POIs más duplicados.</div> <div>Example ValueModel</div> <div><pre>{ "pois": [{ "name": "string", "duplicated": 0 }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/stats/{email}/poisInRoutes

Lista del número de rutas por el número de POIs.

Obtiene una lista del número de rutas que tienen un determinado número de POIs en rangos de 5. El último rango es 31+.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista del número de rutas por número de POIs.</div> <div>Example ValueModel</div> <div><pre>{ "routes": [{ "routeNumber": 0, "rank": "string" }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/stats/{email}/transportsUsage

Lista del número de rutas que usan cada transporte.

Obtiene una lista del número de rutas que emplean cada tipo de transporte.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <small>(path)</small>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista del número de rutas por transporte.</div> <div>Example ValueModel</div> <div><pre>{ "routes": [{ "routesNumber": 0, "transport": "string" }] }</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>

GET

/stats/{email}/mostRequestedRoutesById

Lista de las rutas más recreadas por ID.

Obtiene una lista de las 5 rutas más recreadas a partir de su ID.

Try it out

Parameters

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista de las rutas más recreadas por ID.</div> <div>Example ValueModel</div> <div><pre>{ "routes": [{ "routeId": "string", "requestedNumber": 0 }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

GET

/admin/users/

Listar todos los usuarios del sistema

Lista todos los usuarios del sistema con información a la que sólo el admin puede acceder, a excepción de los usuarios administradores, que no los devuelve.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content type: application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "users": [{ "email": "string", "name": "string", "lastname": "string", "admin": true, "firstLogin": true, "favs": ["string"], "follows": ["string"], "isActive": true, "ban": 0 }]}</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div><pre>{ "success": true, "message": "string"}</pre></div>

PUT

/admin/users/{email}

Modifica la información de un usuario

Permite a un administrador modificar una información determinada del usuario en concreto.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <small>(path)</small>	Email del usuario que sirve como identificador.
name <small>required</small> string <small>(body)</small>	Nombre del usuario.
lastname <small>required</small> string <small>(body)</small>	Apellido del usuario.
newEmail <small>required</small> string <small>(body)</small>	Nuevo email del usuario.

Responses

Response content type: application/json

Code	Description
200	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel<div><pre>{ "success": true, "message": "string"}</pre></div></div></div>
404	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel<div><pre>{ "success": true, "message": "string"}</pre></div></div></div>
500	<div><div>Mensaje de feedback para el usuario.</div><div>Example ValueModel<div><pre>{ "success": true, "message": "string"}</pre></div></div></div>

PUT

/admin/users/{email}/ban

Banea al usuario

Permite a un administrador banear a un usuario de forma permanente o temporal.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <i>(path)</i>	Email del usuario que sirve como identificador.
time <small>required</small> number <i>(body)</i>	Número de días que estará baneado el usuario. 0 implica baneo permanente.

Responses

Response content type

application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>

GET

/stats/{email}/longestRoutesByDistance

Lista de las rutas más largas.

Obtiene una lista de las 5 rutas más largas creadas por el usuario.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.
userEmail <small>required</small> string <i>(path)</i>	Email del usuario propietario de los POIs.

Responses

Response content type: application/json

Code	Description
200	<div>Lista de las rutas más largas.</div> <div>Example ValueModel</div> <div><pre>{ "routes": [{ "routeId": "string", "length": 0 }] }</pre></div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div><pre>{ "success": true, "message": "string" }</pre></div>

PUT

/admin/users/{email}/unban

Desbana al usuario

Permite a un administrador desbanear a un usuario.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <small>(path)</small>	Email del usuario que sirve como identificador.

Responses

Response content type

application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>

PUT

/admin/users/{email}/useDragonBalls

Reactiva una cuenta de usuario.

Permite a un administrador emplear sus todopoderosas bolas de dragón para reactivar una cuenta de usuario que éste había borrado.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.
email <small>required</small> string <small>(path)</small>	Email del usuario que sirve como identificador.

Responses

Response content type

application/json

Code	Description
200	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>
404	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <div> <pre>{ "success": true, "message": "string" }</pre> </div>

GET

/adminStats/totalUsers

Número de usuarios totales del sistema

Devuelve el número de usuarios totales registrados en el sistema, incluidos usuarios baneados y usuarios con cuentas desactivadas.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número de usuarios totales del sistema.</div> <div>Example ValueModel</div> <div>{ "totalUsers": 0 }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>

GET

/adminStats/totalPois

Número de POIs totales del sistema

Devuelve el número de POIs totales creados en el sistema.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <small>(header)</small>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número de POIs totales del sistema.</div> <div>Example ValueModel</div> <div>{ "totalPois": 0 }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>

GET

/adminStats/totalRoutes

Número de rutas totales del sistema

Devuelve el número de rutas totales creadas en el sistema.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content type: application/json

Code	Description
200	<div>Número de rutas totales del sistema.</div> <div>Example ValueModel</div> <div>{ "totalRoutes": 0 }</div>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <div>{ "success": true, "message": "string" }</div>

GET

/adminStats/usersStatus

Número de usuarios activos, inactivos, baneados temporalmente y permanentes

Devuelve el número de usuarios cuyas cuentas se cuentan actualmente activas, inactivas, con baneos permanentes y con baneos temporales.

Parameters

Try it out

Name	Description
Authorization required string (header)	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content type application/json

Code	Description
200	<div>Número de usuarios en cada estado distinto.</div> <div>Example Value Model</div> <pre>{ "usersStatus": [{ "status": "string", "usersNumber": 0 }] }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/adminStats/poisPerUser

Número medio de pois por usuario

Devuelve el número medio de pois totales creados en el sistema en función del número de usuarios totales registrados en el sistema.

Parameters

Try it out

Name	Description
Authorization required string (header)	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content type application/json

Code	Description
200	<div>Número medio de pois por usuario</div> <div>Example Value Model</div> <pre>{ "poisPerUser": 0 }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example Value Model</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/adminStats/lastLogins

Número de logins de usuarios por mes durante el último año

Devuelve el número de logins de usuarios registrados en el sistema durante el último año, agrupados por meses.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número de usuarios en cada estado distinto.</div> <div>Example ValueModel</div> <pre>{ "lastLogins": [0] }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/adminStats/signUpAndRemove

Número de usuarios nuevos registrados y de cuentas que se han dado de baja

Devuelve dos listas que reflejan los nuevos registros y las cuentas que se han dado de baja en el último año agrupados por meses, respectivamente.

Parameters

Try it out

Name	Description
Authorization <small>required</small> string <i>(header)</i>	JWT estándar: Authorization: Bearer + JWT.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número de registros y de bajas en el último año</div> <div>Example ValueModel</div> <pre>{ "signUps": [0], "removes": [0] }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>

GET

/adminStats/routesPerUser

Número medio de rutas por usuario

Devuelve el número medio de rutas totales creados en el sistema en función del número de usuarios totales registrados en el sistema.

Parameters

Try it out

Name	Description
Authorization required string (header)	[JWT estándar: Authorization: Bearer + JWT.

Responses

Response content typeapplication/json

Code	Description
200	<div>Número medio de rutas por usuario</div> <div>Example ValueModel</div> <pre>{ "routesPerUser": 0 }</pre>
500	<div>Mensaje de feedback para el usuario.</div> <div>Example ValueModel</div> <pre>{ "success": true, "message": "string" }</pre>

Models

FeedbackMessage

description:

Mensaje de feedback que se devuelve al usuario en caso de error o acierto en una determinada operación.

success:

boolean

required: true

True si la operación ha ido con éxito. False si ha habido algún error.

message:

string

required: true

Mensaje que describe el resultado de una operación.

}

User

description:

Schema del modelo de User que representa un usuario público del sistema.

email:

string

uniqueItems: true

required: true

Email del usuario que sirve como identificador.

name:

string

required: true

Nombre del usuario.

lastname:

string

required: true

Apellido del usuario.

admin:

boolean

required: true

True si el usuario es un administrador.

firstLogin:

boolean

required: true

True si el usuario es la primera vez que inicia sesión tras haberse creado la cuenta o cambiado la contraseña.

fav:

[string]

required: true

description:Lista con los ID de los POIs favoritos del usuario.

follows:

[string]

required: true

description:Lista con los ID de los usuarios a los que sigue.

}

UserForAdmin

description:

Schema del modelo de User que representa la información de un usuario de la que dispone un administrador.

email:

string

uniqueItems: true

required: true

Email del usuario que sirve como identificador.

name:

string

required: true

Nombre del usuario.

lastname:

string

required: true

Apellido del usuario.

admin:

boolean

required: true

True si el usuario es un administrador.

firstLogin:

boolean

required: true

True si el usuario es la primera vez que inicia sesión tras haberse creado la cuenta o cambiado la contraseña.

fav:

[string]

required: true

description:Lista con los ID de los POIs favoritos del usuario.

follows:

[string]

required: true

description:Lista con los ID de los usuarios a los que sigue.

isActive:

boolean

required: true

Indica si la cuenta está o no activa.

ban:

integer

required: true

Indica si el usuario está baneado. >0 días restantes del ban, 0 ban permanente, -1 no baneado

}

```

POI ▾ {
  description: Schema del modelo de POI que representa un POI público del sistema.
  _id: string
    unique: true
    required: true
    ID del POI en el sistema.
  name: string
    required: true
    Nombre del POI.
  description: string
    required: true
    Descripción del POI.
  tags: string
    required: true
    Conjunto de tags del POI separados por un 's'.
  lat: number ($double)
    required: true
    Latitud del POI.
  lng: number ($double)
    required: true
    Longitud del POI.
  owner: string
    required: true
    Email del usuario que ha creado el POI que sirve como identificador.
  image: string
    required: true
    String en base64 que representa la imagen adjunta al POI, si la hay. Si no la hay, será un carácter vacío.
  url: string
    required: true
    URL adjunto al POI. Si no hay URL adjunto, será un carácter vacío.
}

Route ▾ {
  description: Schema del modelo de Route que representa un una ruta pública del sistema.
  routePOIs: > [...]
  travelMode: string
    required: true
    Modo de transporte de la ruta. Valor discreto entre: 'DRIVING' OR 'WALKING' OR 'BICYCLING' OR 'TRANSIT'
}

```