Securemaps

Juan Miguel Segura

Ivan Miralles Gómez

ÍNDICE

DATOS DEL PROYECTO	
Título	Secure Maps
Autor / Director	Juan Miguel Segura y Ivan Miralles
Breve descripción	Secure Maps es una aplicación para dispositivos Android, que trata sobre reportar en un mapa los incidentes que el usuario considere oportunos, para que otros usuarios se puedan beneficiar de esto.

Este proyecto, trata sobre realizar una aplicación para dispositivos Android, para reportar los lugares donde han ocurrido algún tipo de incidente, como bien puede ser un atraco, robo etc. Desarrollando esta aplicación queremos aumentar la seguridad de las personas, sobre todo en lugares que el usuario desconozca.

Introducción

Secure Maps es una aplicación para dispositivos Android, la cual pretende concienciar a la sociedad de los riesgos de escoger una ruta a la hora de querer llegar a un destino.

Esta aplicación, permitirá que los usuarios puedan reportar incidentes relacionados con la seguridad de las personas, como bien pueden ser peleas, robos, atracos, etc. La finalidad del proyecto es que la gente conozca, de forma exacta, los lugares donde han sucedido estos problemas. De esta manera, se intenta llegar al objetivo de aumentar la seguridad de los usuarios, sobre todo en lugares en los que se desconoce el nivel de peligrosidad.

Esta sería la funcionalidad principal de la aplicación, pero además de ello, también contiene otras funcionalidades como marcar lugares favoritos, etiquetar lugares, etc.

Como idea principal del proyecto, y a modo de resumen técnico, la aplicación está pensada para que extraiga la información de una base de datos alojada en un servidor propio. Para ello, va a ser necesario crear una API propia, de esta manera estamos ampliando conocimientos que van algo más allá de lo aprendido en el curso.

PARTE I. Gestión del proyecto

Contexto

El desarrollo de la aplicación surge debido a la alta tasa de incidentes y peligrosidad que nos encontramos en algunos lugares de ciudades y pueblos, sobretodo lugares que conocemos.

Como posible solución a este grave problema, se ha pensado en diseñar la aplicación, conectada entre todos los usuarios que la utilicen, y que todos puedan ver los incidentes que han sido reportados.

Desarrollando esta idea, posiblemente se evitan muchos incidentes fortuitos que sufren algunas personas, ya que la idea es que al ver algún lugar en el que haya ocurrido algún incidente, o alguna zona con una alta tasa de reportes, el usuario busque una ruta alternativa para no exponerse al peligro, y de esta manera, aumentar su seguridad, independientemente de otros factores externos que le puedan afectar.

Justificación

La justificación por la cual nos hemos decantado a realizar este proyecto es por la necesidad de encontrar los lugares más inseguros, con el fin de evitarlos y poder así aumentar la seguridad e integridad de las personas.

En muchos medios de información, documentales, noticias, podemos encontrar que Barcelona es una de las zonas que tienen un alto nivel de criminalidad.

Basándonos en [blog.prosegur.es](), encontramos el siguiente ranking de "*ciudades más peligrosas de españa*":

1. Madrid
2. Barcelona
3. Sevilla
4. Valencia
5. Málaga
6. Alicante
7. Bilbao
8. Palma de Mallorca
9. Murcia
10. Las Palmas de Gran Canaria

Nosotros somos residentes en Barcelona, concretamente en Santa Coloma de Gramanet, una ciudad cercana a Badalona. Aquí nos encontramos con que, desgraciadamente, también es una de las zonas con mayor criminalidad, lo podemos comprobar leyendo [https://www.diaridebadalona.com]().

Por todos estos motivos, y por la escasez de aplicaciones con esta finalidad en el mercado, nos hemos decantado en crear Secure Maps.

Objetivos

A la hora de escoger este proyecto, nos hemos decantado por desarrollar esta aplicación Android para poder experimentar y aprender en el desarrollo de aplicaciones de la forma más profesional posible.

Como objetivos principales, predomina aprender a dominar el lenguaje Java para Android, y poder desarrollar la aplicación, tanto visualmente como internamente de la mejor manera posible.

Para ello, vamos a aplicar todos los conocimientos aprendidos en clase, y muchos de los cuales debemos aprender durante el desarrollo obteniendo información, mayoritariamente de internet.

Sin embargo, nuestros objetivos no solo son estos, sino que también ascienden a poder diseñar la aplicación, junto a una base de datos instalada en un servidor propio. De esta manera, también vamos a aprender los conocimientos de implementar una base de datos en un servidor y utilizarlo mediante una API programada por nosotros, para que la aplicación Android obtenga los datos en tiempo real.

Por otro lado, uno de los objetivos más importantes respecto a la aplicación es conseguir un alto volumen de usuarios que la utilicen, ya que esto va encadenado con el objetivo principal. Conforme más usuarios hagan uso de la aplicación, más reportes de incidentes se

realizarán y a su vez, más información de los lugares almacenaremos para determinar las zonas más seguras e inseguras y mostrárselo así al usuario.

Es por esto último, que a la hora de realizar el desarrollo de la aplicación, debemos de realizarlo de una manera atractiva para la mayor parte del público. Con esto ampliaremos nuestra experiencia de desarrollar aplicaciones de cara a los usuarios finales.

Alcance

En el momento de inicializar nuestro proyecto, empezamos desde cero con una plantilla en blanco.

Desde el principio hemos tenido claro la función principal de la aplicación, por la cual ha sido desarrollada, reportar incidentes. Es por ello que nos hemos basado en perfeccionar esta función y no entretenernos demasiado en otras funcionalidades menos destacadas.

Una de las necesidades que podría cubrir nuestra app, es la de buscar una ruta alternativa cuando hay muchos incidentes reportados en un lugar. Pero es algo en lo cual hemos decidido no centrarnos.

En una futura amplificación de Secure Maps, es probable que añadamos más funciones, pero las funcionalidades principales pensadas cuando desarrollamos el proyecto ya han sido aplicadas para este.

Planificación

Previsión de tareas de investigación

Para llevar a cabo nuestro proyecto, prevemos que vamos a tener que realizar diferentes tareas de investigación, sobre todo en el entorno de la programación de la aplicación.

A parte de la programación, también tenemos que preparar el entorno del servidor para instalar la base de datos y la api a la que se conectará la aplicación. Para ello, vamos a necesitar instalar un certificado SSL con Let’s Encrypt. Esta última, también es una tarea en la que deberemos investigar, ya que durante el curso no hemos realizado este tipo de trabajo.

Para realizar la API utilizaremos javascript, una tecnología muy útil orientada al desarrollo web. Este lenguaje, también deberemos investigar y estudiar, ya que durante el curso no hemos trabajado con él, debido a que nuestro curso está más orientado a aplicaciones de escritorio y no web.

Por último, y también como tarea de desarrollo, debemos de investigar cómo unir la API de JavaScript con Java en android. Sobre esto, hemos hecho algo parecido durante el curso con Spring Boot, diseñado para API's en Java, así que tenemos una base de conocimientos, pero deberemos investigar más a fondo para acabar de pulirlo.

Como podemos ver, hemos de investigar en diferentes tareas para llevar a cabo el proyecto tal como tenemos previsto. Esto no debe ser un inconveniente, aunque sí nos limita un poco el tiempo dedicado a desarrollarlo, ya que debemos investigar, adquirir nuevos conocimientos y después aplicarlos en la aplicación.

Definición de tareas

A la hora de planificar las tareas, hemos decidido usar como herramienta una hoja de cálculo. Se puede consultar Planificación Secure Maps.

Aun así, podemos ver un resumen en la siguiente imagen.

	Hores Planificades	Hores Reals
Projecte Integrat	198	
Blocs de Fases prèvies al desenvolupament	19	
Introduccion	8	
Contexto	1	
Justificacion	1	
Objetivos	1	
Alcance	1	
Planificación	2	
Presupuesto	2	
Descripcion del proyecto	11	
Análisis de requisitos	1	
Previsión de tareas de investigación	2	
Tecnologías a usar	2	
Definición de tareas	3	
Definición de funcionalidades	3	
Diseño	60	
Aplicación	25	
API	20	
Base de datos	15	
Desarrollo	95	
Implementación de la Aplicación	50	
Implementación de la API	35	
Creación base de datos	10	
Pruebas	17	
Unitarias	5	
Funcionals	6	
Usabilidad	6	
Lanzamiento	2	
Play Store	2	
Conclusiones	1	
Biografía	2	
Anexos	2	

Como podemos observar, la mayor parte del tiempo está dedicado al diseño y desarrollo del proyecto. Esto es debido a que es la parte más complicada y entretenida, el hecho de diseñar y programar la aplicación, y todo el entorno que la envuelve.

Presupuesto

En este proyecto tenemos un presupuesto ajustado y asequible, pero a la vez, suficiente para el correcto desarrollo del mismo.

Casi todo el proyecto lo vamos a poder desarrollar con un mínimo coste de gasto en material, ya que la mayoría del trabajo será la programación y desarrollo de la aplicación. Pero a su vez, vamos a necesitar un servidor físico para poder almacenar la base de datos y la API.

Respecto al servidor, se ha escogido esta opción ya que es una buena forma de tener el control sobre los datos de la base de datos y no depender de un proveedor externo. Sin embargo, esto también tiene sus contras, ya que al almacenar nosotros mismos el servidor físicamente, pueden surgir problemas técnicos, como por ejemplo fallos en la red, cosa que en un datacenter profesional sería difícil que sucediera.

A continuación, se va a mostrar la lista con los materiales y presupuesto necesario para realizar el proyecto.

Material	Coste	Tipo de pago
Raspberry PI 4 (Servidor)	40.95€	Único
Analista - 30h*	440€	Único
Diseñador - 60h*	600€	Único
Programador - 112h*	1.050€	Único
2 Equipos para programar*	1.158,02€	Único
Licencias de Software*	Gratis (Open Source)	-
Mantenimiento del servidor (red + electricidad)	50€	Mensual
Licencia Google Play	20€	Único
Total*	**3.408,95**	

Sueldo de analista
El sueldo del analista lo hemos consultado en la web indeed.com. Puesto que el sueldo medio anual son 28.180€, quedarían en 2.348€ mensuales y 14€/h. Esto multiplicado por las 30h dedicadas al proyecto, se queda en un total de 440€.

Sueldo de diseñador
El sueldo de un diseñador, consultado de nuevo en la web indeed.com es de 19.399€ anuales. Divididos en doce meses a doce pagas por iguales, quedarían en 1.616€, se quedaría en 10€ la hora. Por lo tanto, por un total de 60h, cuesta 600€ el trabajo de un diseñador en el proyecto.

Sueldo del programador
Nos hemos basado en hackaboss.com, puesto que el sueldo medio es de 18.000€ netos anuales, dividido en 12 pagas por iguales, se quedarían en 1.500€ mensuales, un total de 9,3€ la hora. Multiplicando este precio por 112€, queda un total de 1.050€

Equipos para programar
Se ha escogido este ordenador para realizar el desarrollo del proyecto debido a que se ajusta muy bien al precio y a las características necesarias. Con un i5-10400, 16GB de RAM y 512GB SSD, es suficiente para ejecutar Android Studio, que será principalmente el IDE usado.

Licencias
Puesto que estamos a favor del Software Libre, hemos decidido desarrollar todo el proyecto con Software Libre, por lo tanto, no tendremos ningún coste de licencias, ni siquiera en el Sistema Operativo ya que usaremos Ubuntu.

Total
El presupuesto total está basado en los dos meses de duración del proyecto, por lo tanto, los pagos mensuales se deben de pagar dos veces.

Leyes y normativas

Investigando en Internet las posibles leyes y normativas que pueden afectar en nuestro proyecto, únicamente hemos encontrado la ley de protección de datos, ya que en nuestra base de datos vamos a guardar información relevante de cada uno de los usuarios registrados.

Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.

A continuación, mostraremos las principales características de esta ley que afecta en el proyecto.

Finalidad
La finalidad de la LOPDGDD es proteger la intimidad, privacidad e integridad del individuo, en cumplimiento con el artículo 18.4 de la Constitución Española. Del mismo modo, regula las obligaciones del individuo en todo proceso de transferencia de datos para garantizar la seguridad del intercambio.

Notificación de brechas de seguridad
Las brechas en la seguridad que puedan afectar a los datos personales deben ser notificadas en un plazo máximo de 72 horas a la Autoridad de Control correspondiente (Agencia Española de Protección de Datos).

Consentimiento
El consentimiento, con carácter general, debe ser libre, informado, específico e inequívoco.
Las empresas deben revisar la forma en la que obtienen y guardan el consentimiento.
Actualmente existen prácticas que se encuadran en el llamado consentimiento tácito y que son aceptadas con la actual normativa pero dejarán de serlo cuando el Reglamento sea de aplicación.

Responsabilidad proactiva
Para cumplir la ley se han de adoptar medidas que garanticen de manera suficiente que están en condiciones de cumplir con las reglas, derechos y garantías que establece la normativa europea. Ésta entiende que actuar únicamente cuando ya ha tenido lugar la infracción no es suficiente como estrategia, ya que esa infracción puede ocasionar daños a los interesados que puede ser muy complicado compensar o reparar.

Para ello, todas las organizaciones que tratan archivos deben efectuar un análisis de riesgos de sus tratamientos para poder establecer qué medidas han de aplicar y cómo hacerlo.

PARTE II. Ejecución del proyecto

Análisis

Especificación de requisitos

Como requisito fundamental e indispensable, la aplicación debe garantizar o aumentar, de alguna forma, la seguridad de las personas intentando evadir incidentes fortuitos.

Cumpliendose este requisito sería suficiente, ya que es la idea principal y fundamental por la cual se ha desarrollado la aplicación. Sin embargo, Secure Maps también quiere cumplir otros requisitos, no tan indispensables pero sí necesarios para que la aplicación tenga éxito en el mercado.

Requisitos funcionales

Estos requisitos son los que debemos de tener en cuenta a la hora de desarrollar la aplicación, ya que son los requisitos que podemos prever que se cumplan y dependen de nosotros. A continuación mostramos un listado de las principales funcionalidades.

Login
Esta funcionalidad es la principal para que cualquier usuario, previamente registrado, pueda acceder a la aplicación. Una vez haya hecho login, la aplicación guardará los datos de

acceso y automáticamente, cada vez que se inicie, se logrará hasta que el usuario indique “cerrar sesión”.

Registro
Esta funcionalidad se puede usar cuando un usuario no tenga cuenta asociada a la aplicación y desee registrarse como nuevo usuario. Después de realizar esta acción, el sistema guardará los datos de acceso, y al igual que con el login, iniciará automáticamente sesión. El usuario podrá facilitar el nombre de usuario, email y contraseña.

Login con Google
Con esta funcionalidad, el usuario podrá registrarse en la aplicación con Google, de esta manera, se ahorrará crear unos nuevos datos de acceso.

Recuperar contraseña
Cualquier usuario puede recuperar la contraseña de la cuenta usando este método. Se enviará un correo con un enlace con el cual podrá acceder a la aplicación.

Reporte de incidentes.
Esta sería la funcionalidad principal, en la cual el usuario marca el lugar exacto donde ha ocurrido un incidente. A la vez y para ampliar la información de este reporte, el usuario puede añadir comentarios a la hora de marcarlo.

Reportar como anónimo
Esta funcionalidad es muy importante a la hora de asegurar la seguridad de la persona cuando crea un reporte. A la hora de reportar un incidente, el usuario podrá escoger si desea crearlo como usuario anónimo, de esta manera nadie podrá saber quién ha reportado ese incidente y se asegurará la integridad de la persona.

Lugares favoritos
Esta funcionalidad ya está implementada en diferentes aplicaciones de mapas y esto es debido a su gran uso y utilidad. El usuario puede marcar lugares favoritos ilimitados para recordar, entre otros, los lugares que más le han gustado.

Lugares etiquetados.
Esta funcionalidad es muy útil para que el usuario marque la ubicación de su trabajo, casa, iglesia etc. Y lo tenga siempre presente a la hora de ver el mapa.

Configuración personalizada.
El usuario podrá personalizar la configuración de su perfil en todo momento. También podrá eliminar su perfil y toda la información relacionada con este será eliminada definitivamente, de esta manera, el usuario tiene el control total sobre sus datos.

Foro en reportes
Esta funcionalidad es muy útil para abrir un foro sobre un incidente reportado. El usuario tendrá la opción de poner un comentario a la hora de reportar el incidente. Después, todos los demás usuarios podrán comentar en el foro y poner observaciones sobre dicho incidente.

Notificación de incidentes cercanos
Esta funcionalidad se iniciará cuando se produzca el siguiente escenario. Cuando alguien reporte un incidente a 1KM o menos del radio del lugar de residencia habitual del usuario, se le informará con una notificación en la aplicación. Para esto, será necesario que el lugar etiquetado como "*casa*" esté definido previamente.

Ranking de lugares reportados
Usando esta funcionalidad se podrá visualizar el ranking de lugares reportados, clasificado por ciudades. De esta manera, de una forma muy rápida y sencilla el usuario podrá visualizar los lugares con más o menos incidentes, aplicando un filtro que él mismo puede configurar.

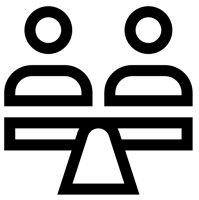
Requisitos no funcionales

Estos requisitos no funcionales, son muy importantes para que la aplicación cumpla las expectativas a nivel de descargas, ya que estos requisitos afectarán de forma directa al uso del usuario con la aplicación.

A continuación mostraremos una lista de estos:

Rendimiento
Este es un requisito indispensable para la experiencia más óptima posible del usuario con la aplicación. A su vez, también es indispensable para poder realizar todas las acciones que el usuario vea conveniente. En general, se ha de conseguir un buen rendimiento de la aplicación, ya que sin este no se llega a ningún lugar.

Estabilidad
La aplicación ha de ser estable, tanto en las conexiones con la base de datos como en su uso cotidiano. Si esta funcionalidad no se cumple, es muy probable que el usuario obtenga demasiados errores al interactuar con la aplicación y se acabe cansando de ella.



Accesibilidad
Este requisito también es muy importante, y quizás uno de los que menos importancia se les dá. La aplicación debe estar accesible para cualquier tipo de usuario ya sea refiriéndonos a la edad de uso como en el nivel del usuario. Es decir, que cualquier tipo de usuario, en cualquier rango de edad y con cualquier nivel de conocimientos de tecnología, pueda llegar a utilizar la aplicación sin ningún inconveniente.

Seguridad

Qué Secure Maps tenga un alto nivel de seguridad es indispensable. En nuestra aplicación, entre otros, guardamos datos personales de los usuarios, y esto es importante guardarlo con una alta seguridad en la base de datos. Para ello, usamos contraseñas encriptadas y seguras, de esta manera nos aseguramos que únicamente conexiones legítimas puedan acceder a los datos, además de una conexión cifrada con la api alojada en el servidor.

Eficiencia

La aplicación debe ser eficiente para cumplir todos los requisitos deseados. Por ejemplo, a la hora de crear un reporte, debe ser algo sencillo pero a la vez eficaz, algo que cualquier tipo de usuario pueda hacer y les sirva de ayuda a todos.
Aplicando este requisito es mucho más fácil avanzar y cumplir todas las demás funcionalidades deseadas.

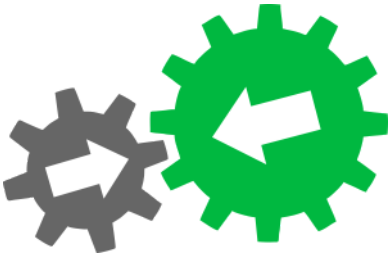
Privacidad

Al igual que con la seguridad, todos los datos del usuario deben ser privados y almacenarse de forma segura. En nuestro caso, Secure Maps permite crear reportes de forma anónima, guardando la privacidad del usuario que ha creado el reporte. De esta manera, el usuario tiene el control total sobre sus datos y sólo los compartirá cuando el desee. Por otro lado, el usuario tiene la opción de eliminar la cuenta con la que se ha registrado y quedarán eliminados, de forma permanente, todos los datos vinculados a esta.

Robustez

Este requisito también es muy importante y muy similar a la "*estabilidad*". La aplicación debe ser una app robusta, con el porcentaje de errores más bajo posible, al igual que el servidor, que en nuestro caso, debe de tener la suficiente capacidad como para atender todas las peticiones que reciba. Aplicando este requisito, conseguimos que el usuario quede satisfecho con el uso de nuestra aplicación.

Compatibilidad



Por último, que nuestra aplicación sea compatible con las diferentes versiones de teléfonos android es muy importante. Si bien no estará desarrollada para "*ios*", tenemos como objetivo que llegue al máximo de usuarios posibles, de esta manera conseguimos un mayor número de usuarios y descargas.

Diseño

Aplicación

Arquitectura

A continuación, vamos a definir la arquitectura de nuestro proyecto, concretamente de la aplicación Secure Maps, en diagramas UML creados con la aplicación Modelio, ya que es la única aplicación Open Source que ofrece todas las utilidades que necesitamos para crear correctamente los diagramas.

Hemos escogido los siguientes tres tipos de diagramas, ya que son los más adecuados en nuestro caso, para realizar el previo diseño de la aplicación.

1. **Diagrama de clases**. Muestra la estructura del sistema, subsistema o componente utilizando clases con sus características, restricciones y relaciones: asociaciones, generalizaciones, dependencias, etc.

2. **Diagramas de casos de uso**. Describe un conjunto de acciones (casos de uso) que algunos sistemas o sistemas (sujetos) deben o pueden realizar en colaboración con uno o más usuarios externos del sistema (actores) para proporcionar algunos resultados observables y valiosos a los actores u otros interesados del sistema(s).

Diagrama de clases

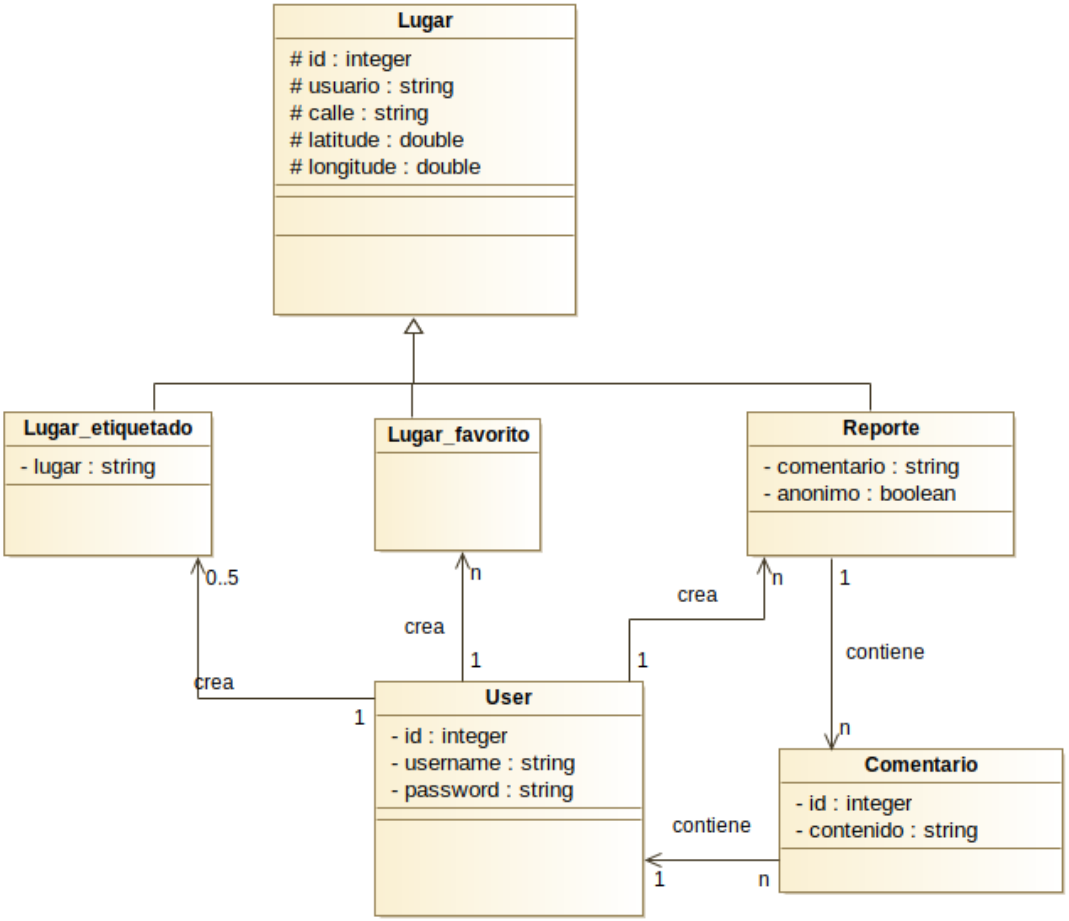
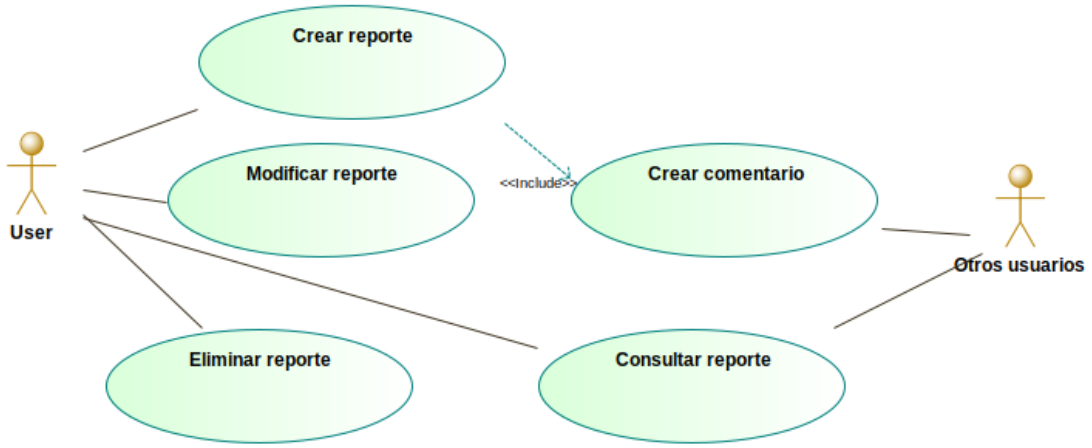


Diagrama de casos de uso

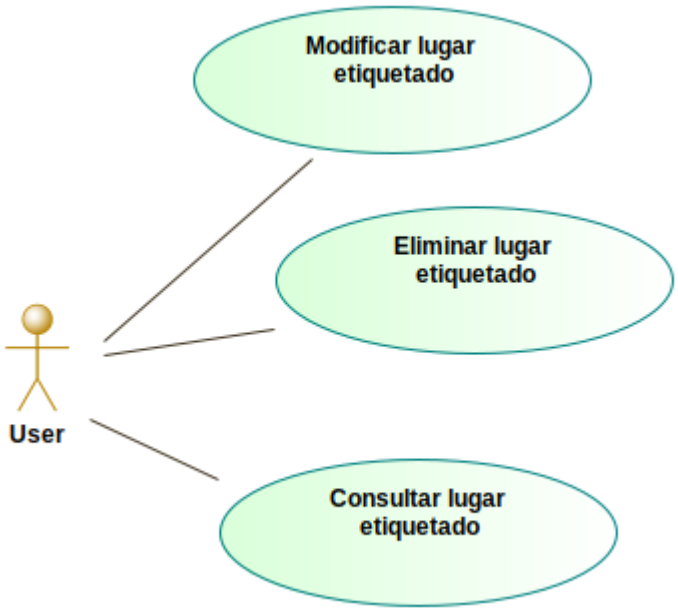
Reporte

En este caso, el usuario puede realizar todas las opciones disponibles referente a los reportes. Sin embargo, cualquier usuario registrado en la aplicación, puede consultar los reportes y añadir comentarios a él en el foro de este.

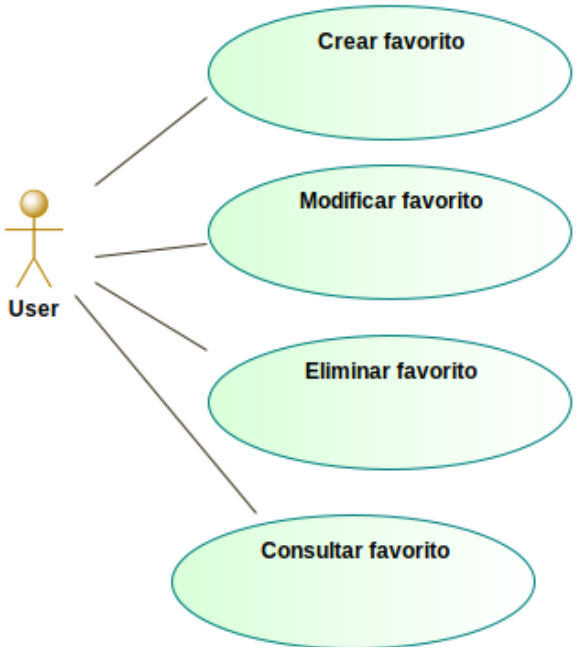


Lugar etiquetados

Únicamente el usuario registrado puede acceder a sus lugares etiquetados, y realizar sus respectivas acciones sobre estos.



Lugar favorito
Al igual que con los lugares etiquetados, únicamente el usuario puede acceder a sus lugares favoritos y sus respectivas funciones.



Seguridad

El apartado de la seguridad, es una de las cosas más importantes en prácticamente cualquier aplicación, por ello hay que tener mucho cuidado y aplicar el máximo de capas posibles para evitar posibles fallos de seguridad y hackeos.

Lado del servidor

En secure maps, implementaremos las siguientes medidas de seguridad, en el lado del servidor.

Servidor con conexión HTTPS
Esto es imprescindible para que los datos que se envían al realizar la conexión entre la aplicación y la api, vayan cifrados y nadie pueda esnifar el contenido de estos.

Puertos cerrados en el servidor
Imprescindible para evitar posibles puertas traseras y accesos al servidor no legítimos. Únicamente estará abierto el puerto de la conexión con la API, en nuestro caso, el puerto **5000**.

Contraseña segura para el acceso a la base de datos y servidor
Implantando una contraseña suficientemente segura, y validada por validadores oficiales (https://password.kaspersky.com/ - https://password.es/comprobador/), es prácticamente imposible que robots puedan acceder.

Firewall de seguridad
Añadiendo un firewall, podemos filtrar y evitar ataques masivos como DDOS o DDS. En este caso, al tener el servidor físicamente, el router ya incorpora un firewall suficiente para evitar esto.

Monitorización en el acceso a la base de datos y servidor
Añadiendo esta capa de seguridad extra, estaremos informados en todo momento de quién y cuándo ha accedido, tanto a la base de datos como al servidor. Esto lo podemos realizar creando un script en el archivo .bashrc, archivo que se inicia cada vez que se inicia sesión, que nos envíe un correo con el acceso y la hora de este.

Backups
Realizar backups del servidor es muy importante, por si en alguna parte del proyecto obtenemos un error que no conseguimos solucionar, o simplemente el servidor se infecta con algún virus. Con los backups podemos restaurar una imagen por completo de nuestro contenedor.

Lado de la aplicación

Además de todas las capas mencionadas anteriormente, aplicadas en el lado del servidor, implementaremos las siguientes en la aplicación.

Contraseñas encriptadas
Las contraseñas de los usuarios, en ningún momento se guardarán en texto plano, ni en local ni en el servidor, ya que en el caso de sufrir algún hackeo, puede peligrar la integridad de estas. Todas las contraseñas de los usuarios se reemplazan por un hash.

Acceso con Google
Esta es una medida de seguridad para los usuarios que no deseen escribir una contraseña en una aplicación que no conocen. De esta manera, únicamente obtenemos el usuario y el correo, pero nunca llegamos a tener la contraseña en nuestra base de datos.

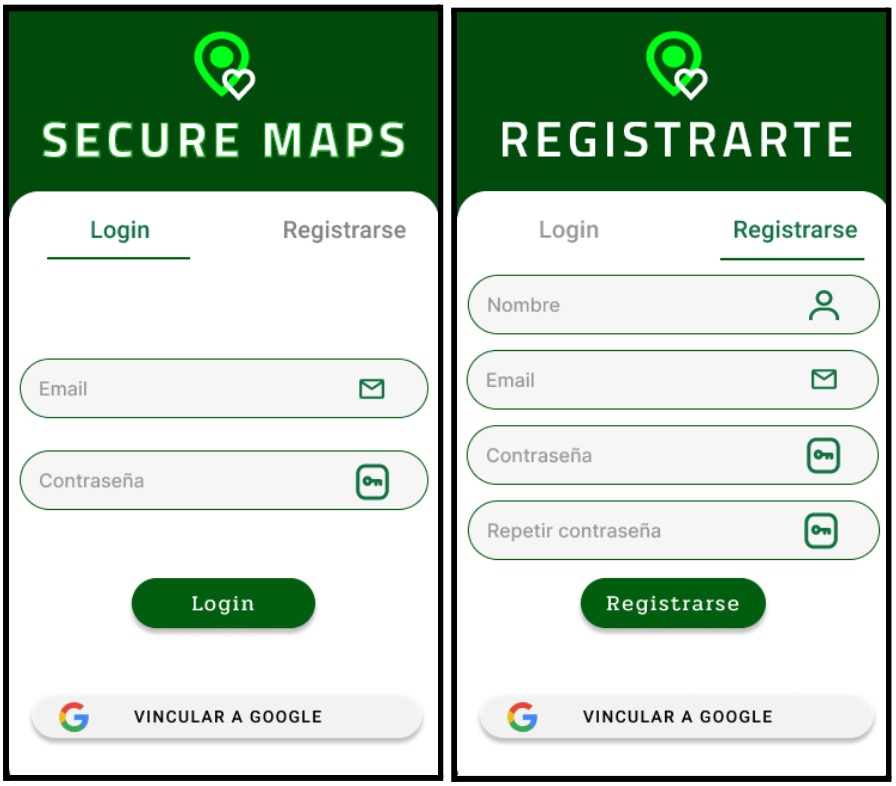
Reportar como anónimo
Esta funcionalidad podemos clasificarla como seguridad en la integridad de la persona, ya que reportando un incidente como anónimo, nadie sabrá quien ha creado ese reporte y no podrán aplicar represalias.

Interfície

Diseño en FIGMA

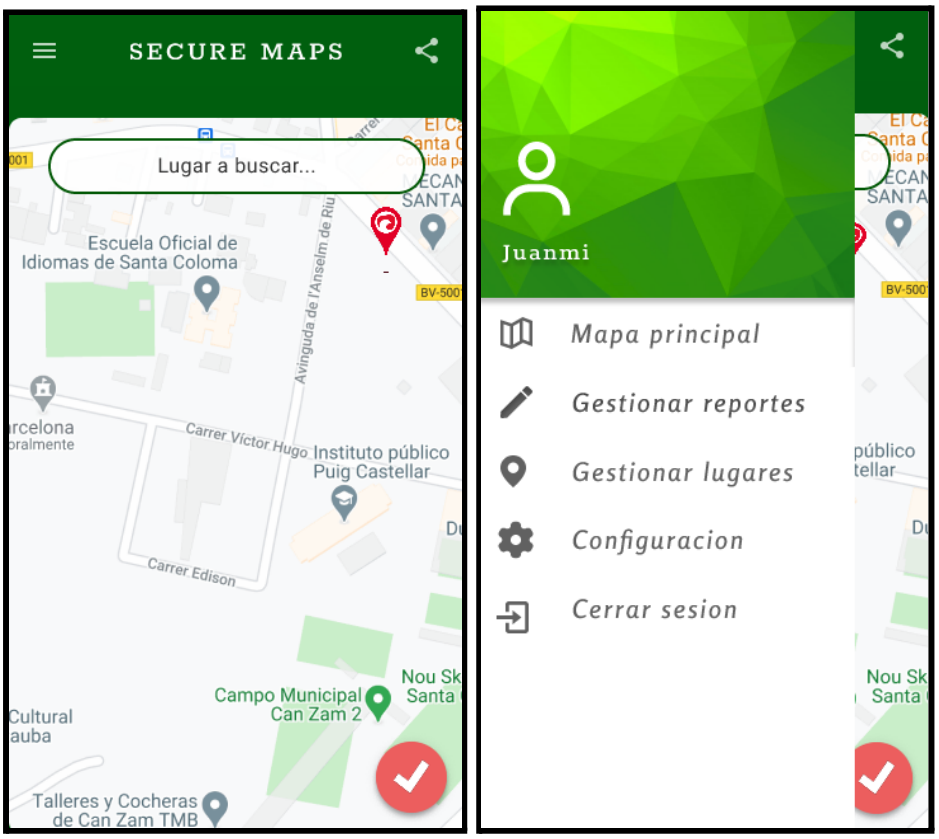
Login: En esta pantalla podremos iniciar sesión si tenemos ya una cuenta creada, en caso contrario, deberemos deslizar o pulsar a la sección de Registrarse.

Registrarse: Aquí podremos crear nuestra cuenta si no tenemos.

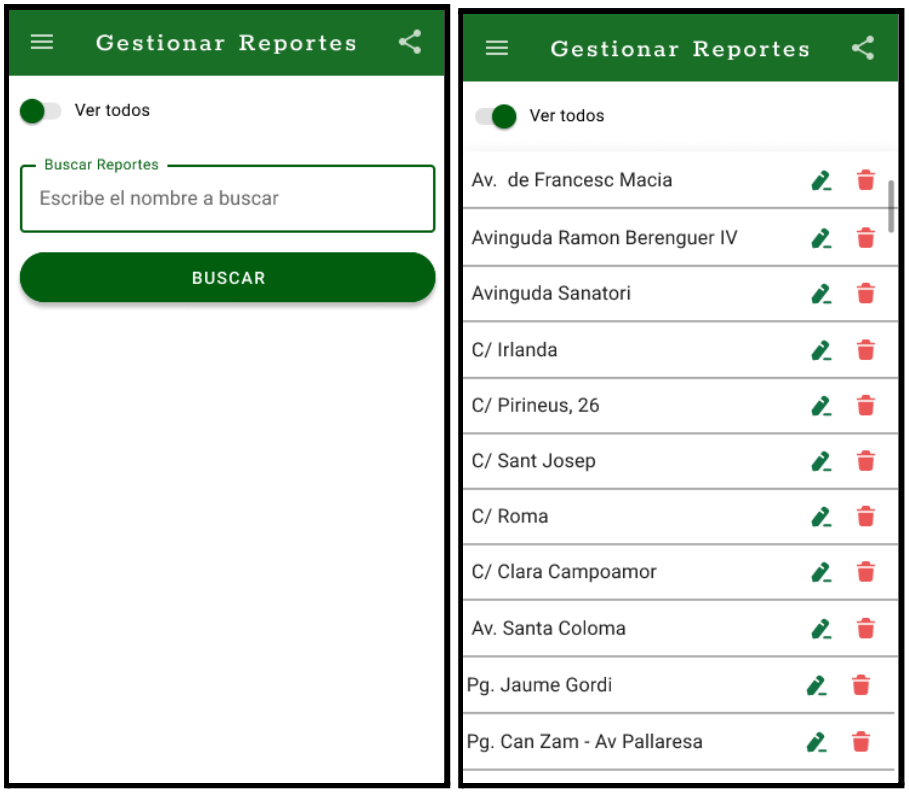


Mapa Principal: Aquí se encuentra el mapa el cual podremos interactuar con él, ya sea para crear un nuevo reporte, consultarlos de los otros usuarios, visualizar nuestros lugares etiquetados y favoritos.

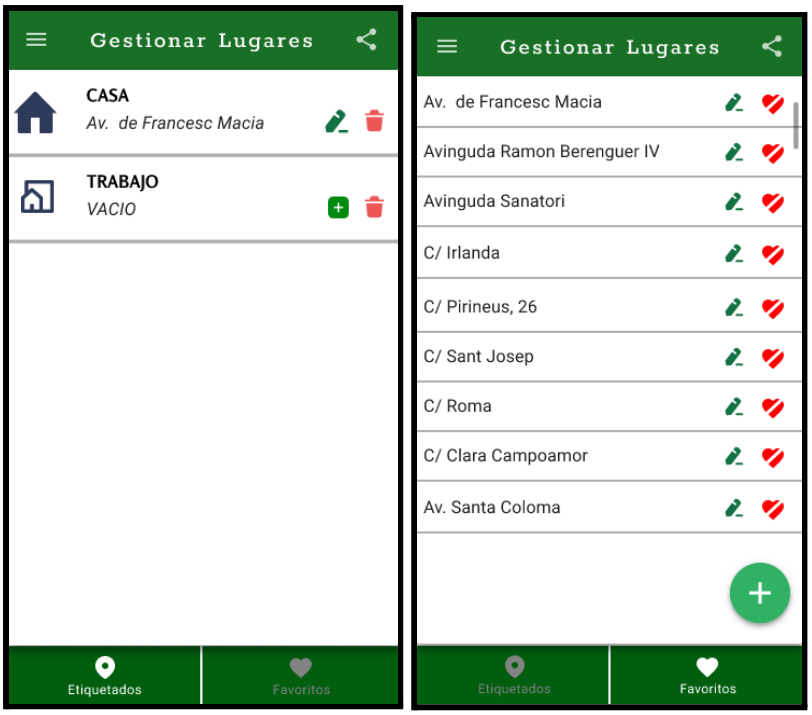
Menú lateral: En él podremos navegar a todas las secciones de nuestra aplicación y visualizar el nombre de usuario y la foto.



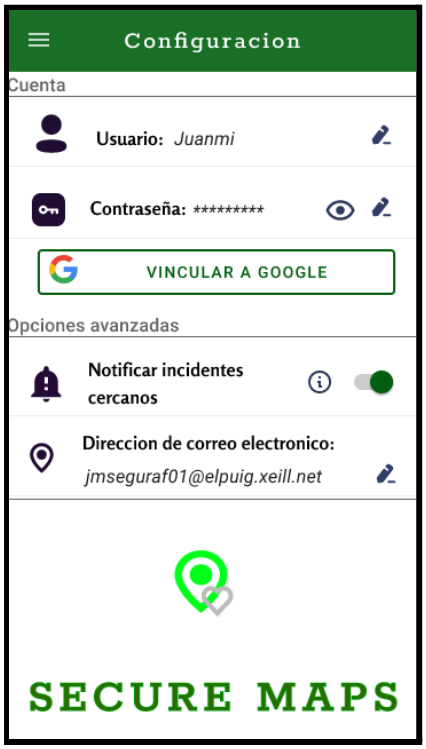
Gestionar Reportes: Encontraremos los reportes creados, los cuales podremos modificar, añadir o modificar el comentario y eliminarlo. También podremos buscarlos por el nombre de la calle.



Gestionar Lugares: Aquí encontraremos dos secciones, la primera es la de etiquetados, donde veremos las direcciones de nuestra casa, la del trabajo, gimnasio e iglesia. En la segunda se encuentra los lugares favoritos del usuario.



Configuración: En esta pantalla podremos modificar los datos del usuario, como puede ser, el correo electrónico y la contraseña. También podemos eliminar nuestra cuenta.



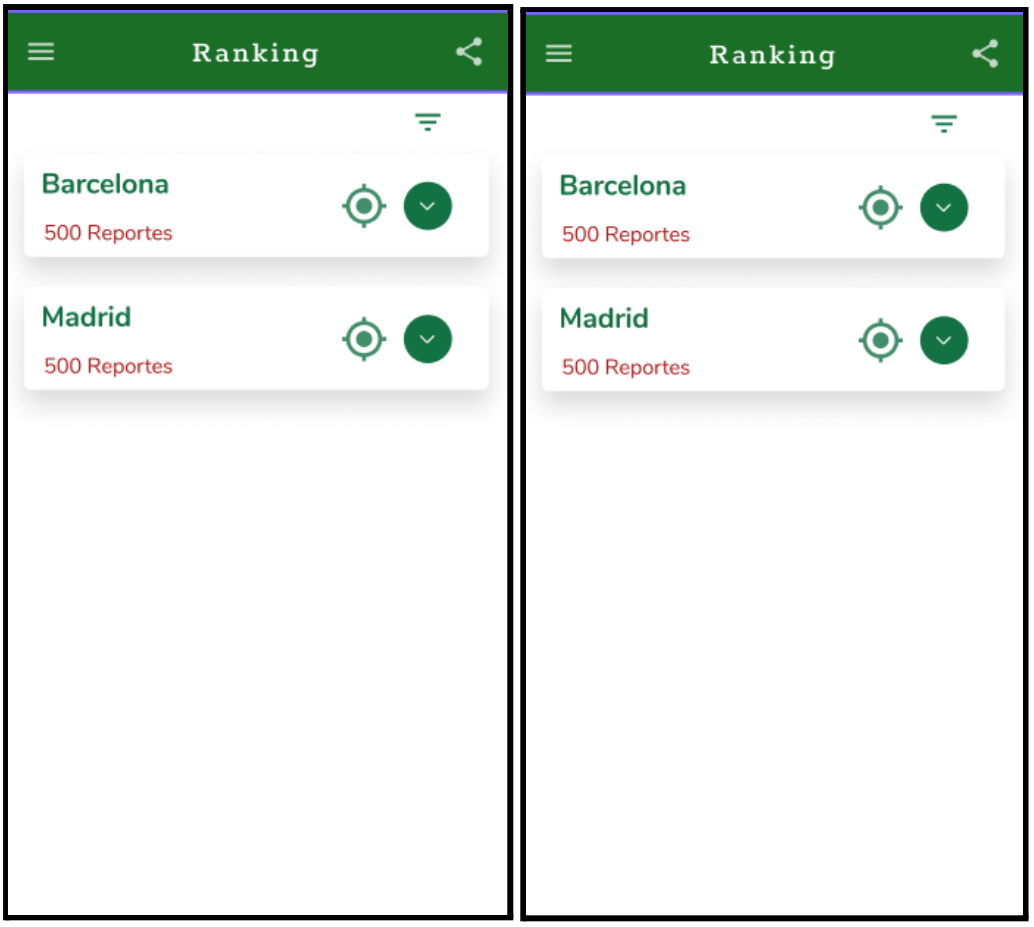
Diálogo de confirmación: Este diálogo aparecerá en diversas ocasiones en la aplicación, ya sea para eliminar la cuenta, cambiar la dirección de un lugar favorito o etiquetado.

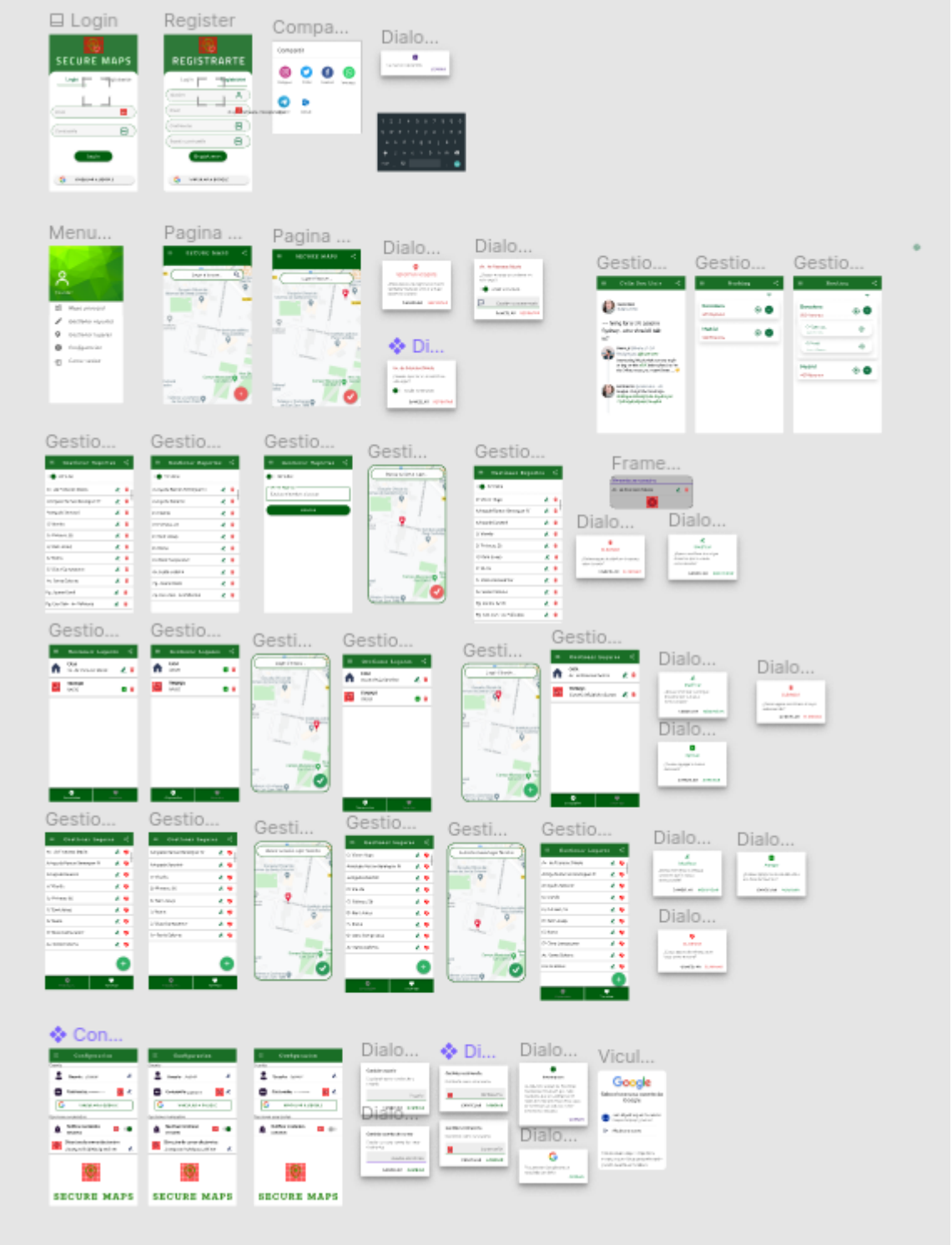
Diálogo de modificación: Nos aparecerá al editar el comentario de un reporte.

Diálogo de reporte: Se mostrará al crear un reporte.

Foro de reportes: Se nos mostrará al clicar en un reporte en el mapa principal. Veremos los comentarios que los usuarios han dejado en el reporte seleccionado.

Ranking: Se mostrará un ranking por ciudades, de las que tienen más reportes a la que menos. Si pulsamos una ciudad aparecerá un desplegable junto a las calles más reportadas en cuya ciudad.





[Enlace a figma.]

Criterios de usos de colores y diseño

El color principal escogido para nuestra aplicación es el ‘Verde’, ya que se identifica por el mapa que aparece nada más abrir la aplicación.

El mapa fue la nota detonante para decantarnos por este color, ya que es el primer color que se te viene a la cabeza cuando escuchas la palabra “Mapa”.

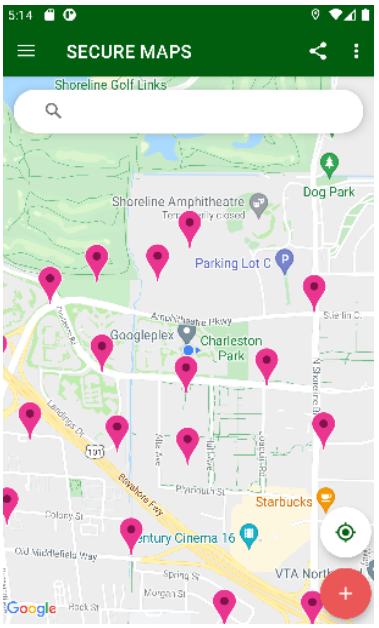
También creemos que el color verde da tranquilidad y seguridad, que es lo que queremos demostrar a nuestros usuarios.

Usabilidad

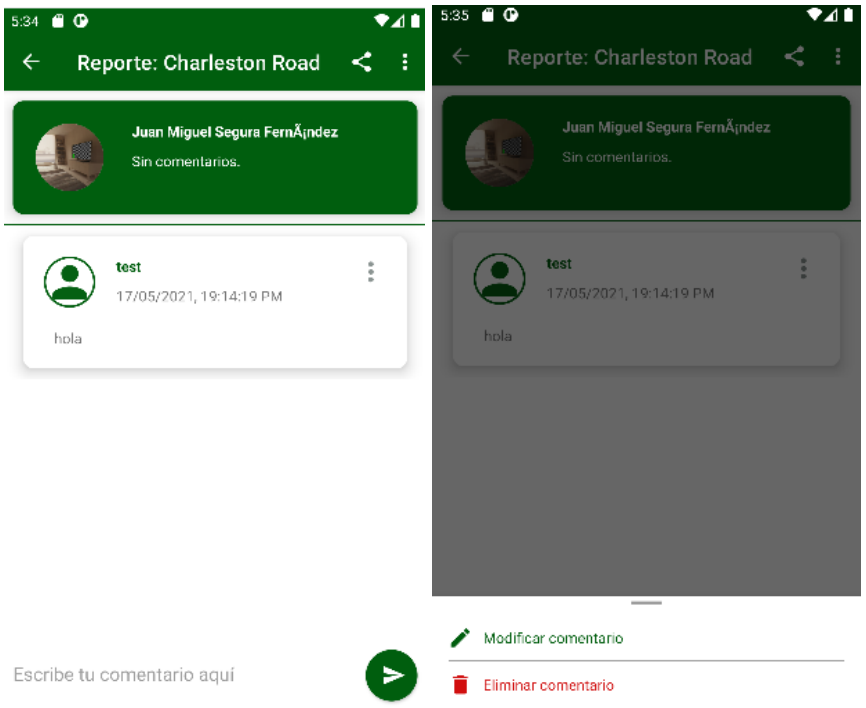
A continuación vamos a explicar la parte de usabilidad de la app pantalla por pantalla.

Login y Sign up: Podemos encontrar un “TabLayout” que nos permitirá navegar entre las dos pantallas. En la del ‘Login’ podemos apreciar dos campos de texto uno para el correo y otro para la contraseña, y por último, dos botones uno en verde que servirá para iniciar sesión si ya tienes una cuenta y el otro blanco para iniciar sesión con Google.
En el ‘Sign up’ encontramos 4 campos de texto, uno para el nombre del usuario, otro para el correo y los dos últimos para la contraseña. En este apartado tenemos los mismo botones que en el ‘Login’ pero en este caso el verde es para registrarse.

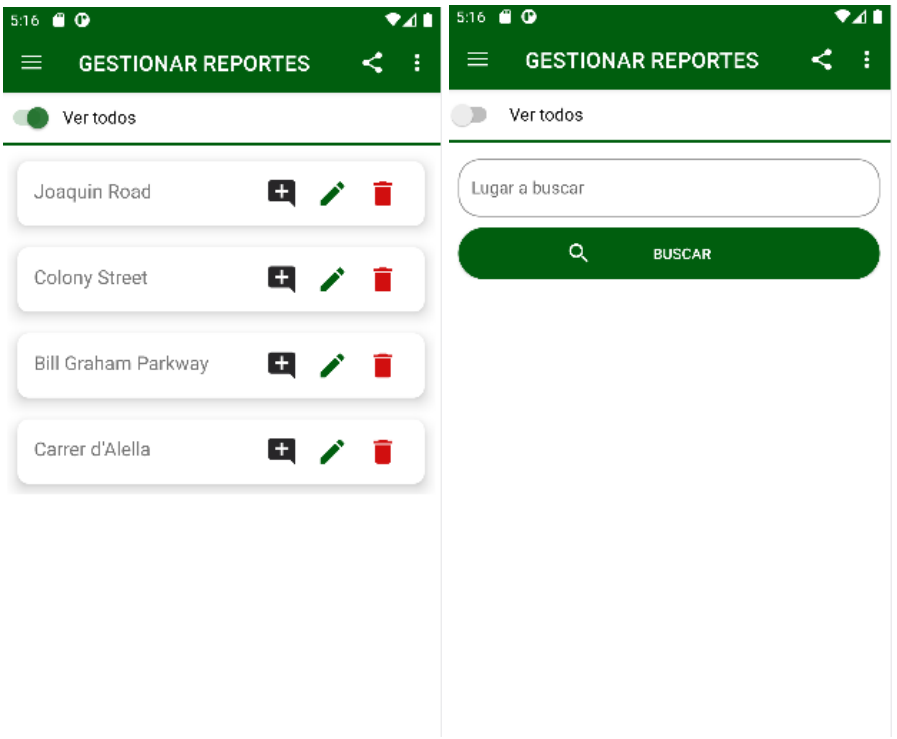
Mapa principal: Aquí lo primero que se encuentra es el mapa el cual podremos interactuar con él creando reportes. Para ello lo primero que debes de hacer es seleccionar un lugar en el mapa y darle al botón de + que se encuentra en la parte inferior derecha de la pantalla.
También podemos encontrar otro botón de localización, encima del botón de +, que sirve para recuperar la posición en el mapa donde nos encontramos.
En el mapa también podremos apreciar marcadores, que son los reportes creados por ti o por otros usuarios (los de color rojo son los propios y los rosas los de los demass usuarios), si los pulsamos nos lleva al foro del propio reporte.
Por último encontraremos un buscador en la parte superior de la pantalla para buscar cualquier lugar en el mapa.



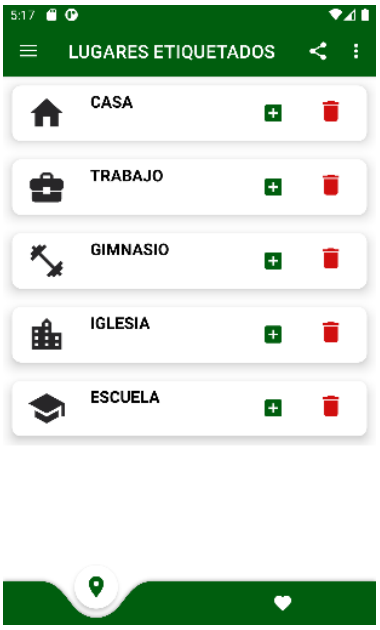
Foro de Reportes: Al abrir el foro de un reporte lo primero que veremos es el comentario principal del reporte en verde, con el usuario que lo ha creado (si no ha hecho el reporte como anónimo), la fecha de creación del comentario y el contenido de este. A continuación de la pantalla y separado por un divisor se encuentran los demás comentarios del foro en color blanco que contiene básicamente la misma información que el principal. También podemos apreciar que si es un comentario propio te aparece los tres puntitos verticales de opciones los cuales si los pulsas, te aparecera un dialog en la parte inferior de la pantalla con las opciones de editar el comentario o eliminarlo.



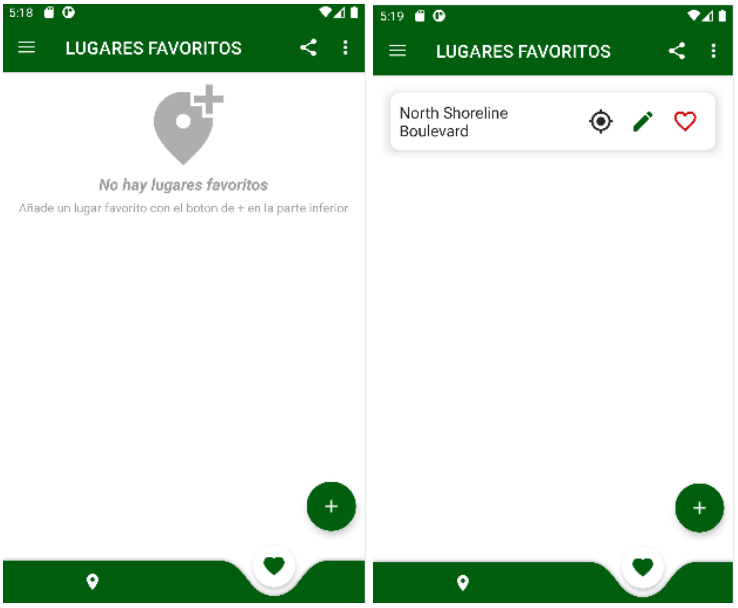
Gestionar Reportes: Lo primero que encontramos en esta pantalla es un 'Switch', el cual si está activado nos mostrará todos los reportes creados por uno mismo, y por otro lugar si está desactivado, nos mostrará un buscador para poder encontrar uno en concreto.
En cada reporte veremos 3 botones que uno sirve para editar el comentario del reporte, otro para editar la ubicación y por último para eliminarlo.



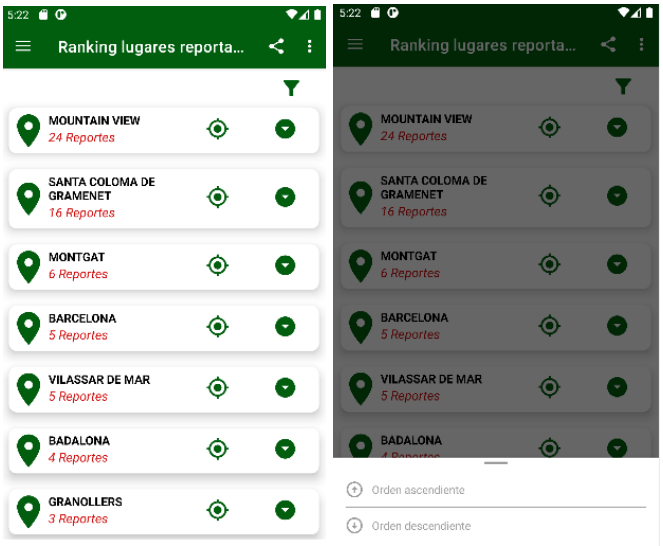
Lugares Etiquetados: En esta pantalla encontraremos 5 apartados que sirven para que quede registrado en la aplicación la ubicación de tu casa, trabajo, gimnasio, iglesia y escuela. En cada apartado podemos apreciar dos botones, en el caso que todavía no hayas registrado la ubicación, te aparecerá un icono de + para registrarla, en el caso de que ya se haya registrado nos aparecerá un icono de un lápiz por si queremos editar la ubicación. Y el segundo botón que nos aparecerá siempre es el de eliminar la ubicación.



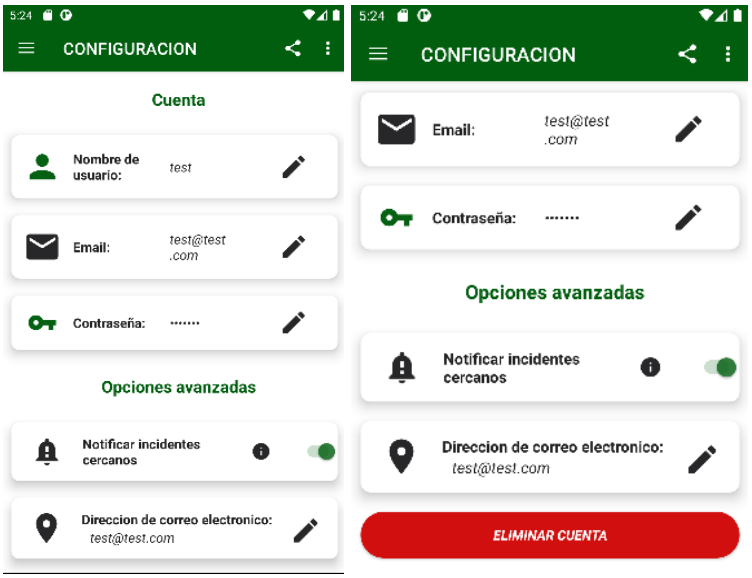
Lugares Favoritos: Al entrar por primera vez a esta pantalla solo veremos un icono con una breve explicación abajo que explica cómo añadir contenido en este apartado.
En la parte inferior derecha veremos el único botón de la pantalla con un símbolo de + para añadir un lugar favorito. Una vez añadido el lugar nos aparecerá en la pantalla acompañado de tres iconos, el primero empezando por la derecha sirve para localizar el lugar favorito en el mapa, el segundo para editar la ubicación de este y el último para eliminarlo de tus lugares favoritos.



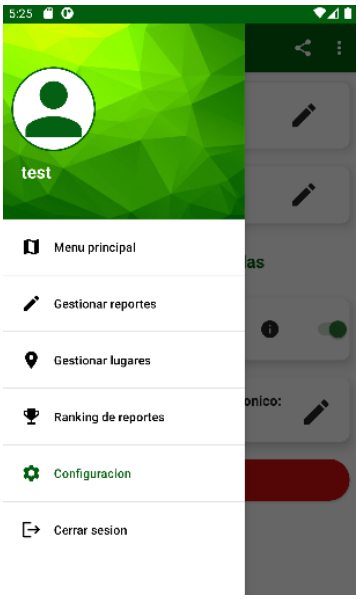
Ranking: En esta pantalla lo primero que nos llamara la atención es la lista de municipios que aparece. Cada municipio contiene dos botones, el primero empezando por la izquierda sirve para localizar el municipio en el mapa, el segundo es para desplegar la lista de calles reportadas en él. También podemos ver debajo de los nombres de los municipios los número de reportes que lleva cada uno. La lista de calles reportadas de cada municipio es simplemente informativa y no se puede interactuar con ella.
Por último podemos encontrar un icono de un embudo el cual si lo pulsamos nos aparecerá un 'fragment' desde la parte inferior de la pantalla, donde se encuentran dos opciones, una para ordenar la lista de municipios de los más reportados a menos y la segunda lo mismo pero al revés.



Configuración: En esta pantalla podremos apreciar dos secciones, la primera llamada 'Cuenta', donde podremos consultar nuestros datos de la cuenta, como: el nombre de usuario, el correo electrónico y la contraseña. Todos estos datos los podemos modificar pulsando el icono del lápiz que les acompaña en la parte derecha.
La segunda sección es la de 'Opciones avanzadas' donde podremos activar o desactivar si queremos recibir incidentes cercanos a nosotros y a que correo se enviarán estas notificaciones, por último tenemos un botón en rojo para eliminar la cuenta.



Menú lateral: Al ver el menú lo primero que vemos empezando por arriba, es la foto de perfil del usuario y su nombre, la foto se puede cambiar por la que tu quieras pulsando sobre ella. Más abajo encontramos todos los apartados donde podremos navegar por la aplicación, y por último tienes la opción de cerrar sesión.



Persistencia

API

Utilizar una API hoy en día, es imprescindible para realizar una conexión con la base de datos.

En nuestro proyecto hemos optado por programar nosotros mismos una API, para conectar la aplicación Secure Maps con nuestra base de datos que tenemos alojada en el servidor.

Gracias a esta API, podemos realizar todas las consultas, inserciones, modificaciones etc. que sean necesarias y poder trabajar de esta manera, de forma muy eficaz y estable.

Esta API, la tenemos alojada en el mismo servidor en el que se encuentra la base de datos. Teniendo la API y el servidor en “localhost”, conseguimos que la comunicación entre ambos sea mucho más rápida, esto se va a traducir en un mejor rendimiento en la aplicación.

Base de datos

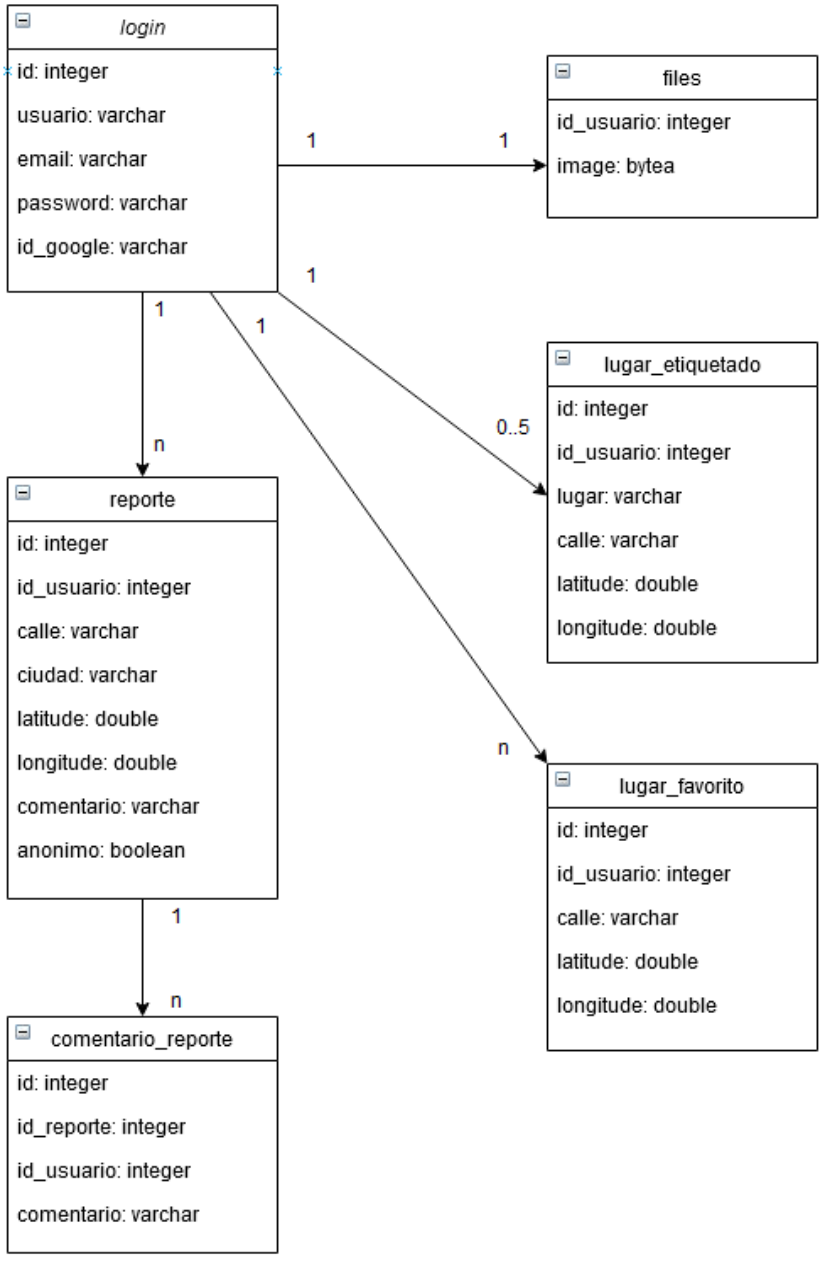
La base de datos del proyecto, utiliza el motor gestor de base de datos Postgres. Este es un gestor de base de datos muy potente y eficaz para llevar a cabo nuestro proyecto en él.

La base de datos está alojada en el servidor y únicamente podrá acceder a esta nuestra API. La autenticación se realiza a través de un usuario creado con los permisos necesarios para acceder a ella, y únicamente tendrá acceso el usuario indicado.

Esta base de datos es SQL, por lo tanto es relacional, algo muy importante en nuestro proyecto, ya que los datos se guardarán en distintas tablas a la vez, y el hecho de ser SQL permite la modificación de todos estos datos de una forma conjunta e instantánea.

Diseño de la base de datos

A continuación, mostraremos el diagrama del diseño de la base de datos de nuestra aplicación Secure Maps.



Como podemos observar, hay diferentes relaciones entre las tablas y a su vez estas tienen columnas con diferentes campos. La elección de las relaciones y los tipos de campos, lo hemos elegido adecuando, cada uno de ellos, a las necesidades de las funcionalidades de la aplicación.

Tecnologías a usar

En este apartado, vamos a definir los siguientes dos tipos de tecnologías: Tecnologías a usar y tecnologías a desarrollar.

Tecnología a usar

Estas tecnologías, son las que hemos utilizado durante la planificación, diseño y desarrollo de todo el proyecto.

Gracias a estas hemos podido realizar todo el proyecto tal y como teníamos pensado, pudiendo llegar a implementar todas las ideas iniciales que teníamos, y desarrollar la aplicación de forma eficaz con un buen rendimiento.

Todas las tecnologías utilizadas son Open-Source, de esta manera, al igual que el proyecto, fomentamos el uso y desarrollo de aplicación de código libre, algo necesario para la sociedad.

Android Studio

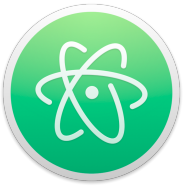
Android Studio es el IDE recomendado para desarrollar aplicaciones para Android. Con este IDE hemos podido programar toda la aplicación, tanto el Frontend como en Backend, incluso realizar pruebas en un emulador que viene incluido en este.

VIM

Vim es un editor de texto sin entorno grafico, orientado principalmente en el entorno de los servidores. En el proyecto, este editor de texto se ha usado para editar algunos ficheros de configuración en nuestro servidor. Es muy útil y eficaz, sobretodo a la hora de editar ficheros en un entorno de terminal.

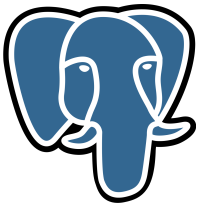
Atom

Atom es un editor de texto muy similar a Visual Studio, con un entorno gráfico, ágil y potente. Lo hemos utilizado, básicamente, para desarrollar la página web del proyecto, ya que es totalmente compatible con los lenguajes HTML y CSS, incluso tiene la opción de instalar módulos enfocados al desarrollo web.



Postgresql

Postgresql es el gestor de base de datos que hemos utilizado para diseñar nuestra base de datos. En el proyecto nos hemos decantado por utilizar postgresql, ya que es el gestor de base de datos que hemos utilizado durante todo el grado superior y cubre perfectamente nuestras necesidades para el proyecto.



Let's Encrypt

Let's Encrypt, es la única entidad certificadora que ofrece un certificado HTTPS de forma totalmente libre y gratuita. Hemos implementado Let's Encrypt para poner un certificado HTTPS en el servidor que implementa la página web del proyecto, y la API. De esta forma, sobre todo en la API, la comunicación con la aplicación es totalmente segura, ya que va encriptada. De esta manera, conseguimos evitar que se puedan esnifar datos.

Hoja de cálculo

Hemos utilizado la hoja de cálculo de Google para realizar la planificación de las tareas que debemos hacer durante el desarrollo del proyecto. Hemos optado por usar esta tecnología, ya que es muy común y eficaz para describir tareas de forma rápida.

Modelio

Modelio es un software Open Source para realizar diagramas de todo tipos. Este software lo hemos usado durante el curso, y cubre todas las necesidades que necesitábamos para realizar los diagramas sobre la arquitectura de la aplicación.

Apache

Apache es el servidor web que hemos utilizado para alojar nuestra página web. Hemos optado por utilizar este servidor, ya que es el más común en el sector de servidores web, al igual que Nginx.

Figma

Hemos utilizado Figma para desarrollar todas las interfaces de la aplicación previamente a desarrollarlas directamente en el código. Esta aplicación nos ha permitido hacernos una idea general sobre el diseño de la aplicación, para de esta manera poder modificar el diseño antes de implementarlo.

Linux

En todo el desarrollo del proyecto hemos usado como kernel de Sistema Operativo Linux, en concreto y generalmente, la distribución de Ubuntu 20.04. Este Sistema Operativo nos ha permitido realizar todo el desarrollo sin problemas, con una gran eficacia, incluso instalar todas las aplicaciones necesarias para este.

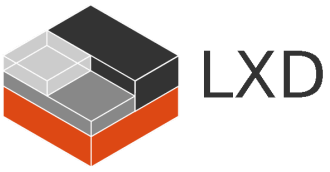
Git / Gitlab

Git es una tecnología que permite desarrollar código con varios usuarios a la vez, de forma que puedas organizar el código por tareas, y una vez desarrollarlas, juntarlo todo. En el proyecto hemos utilizado esta tecnología, ya que es lo ideal para un trabajo en grupo, y gracias a usarla hemos podido desarrollar partes independientes de la aplicación y finalmente juntar ambas.

Ubuntu Server 20.04

La distribución de Linux Ubuntu Server 20.04, es la elegida para montar nuestro servidor, el cual como hemos explicado anteriormente, mantiene la API y la base de datos. Hemos escogido esta distribución, ya que está destinada a los servidores, no incluye un entorno gráfico y por lo tanto, los recursos están más optimizados.

LXD Containers



Esta es una tecnología relativamente nueva la cual está desarrollada por Canonical, los creadores de Ubuntu. Esta tecnología permite virtualizar contenedores, con diferentes Sistemas Operativos utilizando el mismo kernel que la máquina física. En este caso, si la máquina física utiliza Linux, únicamente podremos virtualizar contenedores con distribuciones Linux. En el proyecto, hemos decidido virtualizar el servidor para poder hacer “snapshots” de seguridad, y en el caso de tener que recuperar una copia de seguridad, poderlo hacer de forma rápida y efectiva.

Tecnología a desarrollar

Ha sido posible crear nuestro proyecto, gracias a tecnologías que permiten el desarrollo del mismo. A continuación, mostraremos las tecnologías usadas para desarrollar tanto la aplicación, como la API, la base de datos y la página web.

Java
Java es un lenguaje multiplataforma, y en el proyecto el lenguaje principal en el que está desarrollada toda la aplicación Android. Este es un lenguaje muy usado a nivel global, y con el cual hemos estado trabajando durante todo el ciclo superior. Hemos optado por usar este lenguaje, ya que es con el lenguaje que hemos aprendido a programar para Android y se adapta perfectamente a las necesidades de nuestra aplicación.

Android
Android es un framework que trabaja sobre Java. Este está desarrollado exclusivamente para crear aplicaciones en dispositivos Android. Hasta hace poco, era único y el lenguaje principal para Android. Hoy en día, existen alternativas, pero nos hemos decantado por este ya que es con el que hemos trabajado, y al igual que java, se adapta perfectamente a las necesidades del proyecto.

XML
XML es un lenguaje de marcas muy usado en distintos campos. En este caso, ha sido usado para diseñar todo el frontend en Android, ya que al usar Java para el backend, es imprescindible realizar todo el diseño visual con XML. Al igual que los demás, es un lenguaje el cual hemos aprendido a usarlo durante todo el grado superior.

SQL
SQL es el lenguaje orientado a las bases de datos, en nuestro caso, tal como hemos comentado anteriormente, hemos usado postgresql. Hemos optado por usar SQL, para poder satisfacer todas las necesidades necesarias de consultas, inserciones, modificaciones etc. que son requeridas por la aplicación. Cabe destacar que la alternativa a SQL es NO-SQL, un lenguaje el cual no guarda claves foráneas sino copias de estas. En nuestra aplicación es más importante la integridad de los datos que la velocidad de consulta, es por esto que hemos optado por usar SQL.

JavaScript
JavaScript es un lenguaje muy flexible y ágil, el cual hemos utilizado para desarrollar toda nuestra API. En este caso, este lenguaje no lo hemos trabajado y para nosotros ha sido una tecnología totalmente nueva, pero hemos valorado hacerlo así porque consideramos que es uno de los lenguajes que más se adaptan a nuestro proyecto, sobretodo en el entorno del servidor.

NodeJs
NodeJs es un entorno de programación en el cual se ejecuta nuestro código de JavaScript, mencionado anteriormente. Hemos escogido utilizar esta tecnología, ya que es la más utilizada para JavaScript.

HTML / CSS

HTML y CSS son lenguajes para desarrollar páginas web, los cuales hemos aprendido a utilizar en grado medio y lo hemos aprovechado para desarrollar la página web del proyecto. Hemos optado por utilizar estos lenguajes ya que son relativamente sencillos, debido a que la página web no va a contener una gran cantidad de datos ya que simplemente es una web informativa, usar estos lenguajes es más que suficiente. Por otro lado, el desarrollo de la web debe de ser algo secundario, ya que la aplicación Android tiene muchas más importancia en nuestro proyecto.

Desarrollo

Implementación de la Aplicación

A continuación, definiremos las estrategias de desarrollo de nuestra aplicación, dividido en dos apartados.

Estrategia de desarrollo

A nivel de grupo, a la hora de desarrollar la aplicación nos hemos dividido la faena a partes iguales, y realizando un trabajo paralelo. Generalmente en cualquier trabajo en grupo, es muy importante no solaparse a la hora de diseñar y desarrollar un proyecto, evitando esto conseguiremos un óptimo rendimiento.

En nuestro caso, el desarrollo con Android principalmente depende de dos lenguajes de programación; XML y Java.

XML se encarga de toda la parte del diseño de la aplicación, o también llamado Frontend. El Frontend es la parte visual, lo que el usuario final va a visualizar y lo que más debe destacar para que sea una aplicación llamativa y elegante, a la vez que minimalista y profesional.

Por otro lado, tenemos Java, el lenguaje encargado de interpretar XML, todo el Backend de la aplicación, la lógica, conexión con la API, servicios externos, etc.

A continuación definimos lo que hace cada uno de estos lenguajes en nuestra aplicación.

XML
Como hemos comentado anteriormente, XML se utiliza para diseñar todo el Frontend de la aplicación. En nuestro proyecto, lo hemos utilizado exactamente para esto mismo. Con XML hemos creado todas las interfaces y todos los menús que comunican las pantallas entre sí.

Para hacer funcional todo esto, es necesario implementar código Java, pero con XML conseguimos un diseño elegante y profesional. Todos los elementos que se muestran en una interfaz, están previamente definidos y diseñados con este lenguaje.

También hemos utilizado algunas librerías que facilitan el diseño en algunos casos. Estas librerías, se definen en la interfaz con XML, y posteriormente se implementa el código Java necesario para hacerlo funcional.

En general, hemos obtenido una buena experiencia desarrollando con XML ya que nos ha permitido diseñar una aplicación con un aspecto muy profesional, aún así esta es una tecnología que cada día cae más en desuso, ya que hay alternativas como por ejemplo Kotlin o Flutter, que implementan el desarrollo de la interfaz en el propio código, y con un solo lenguaje te permite realizar todo lo necesario para diseñar tu aplicación completa.

JAVA

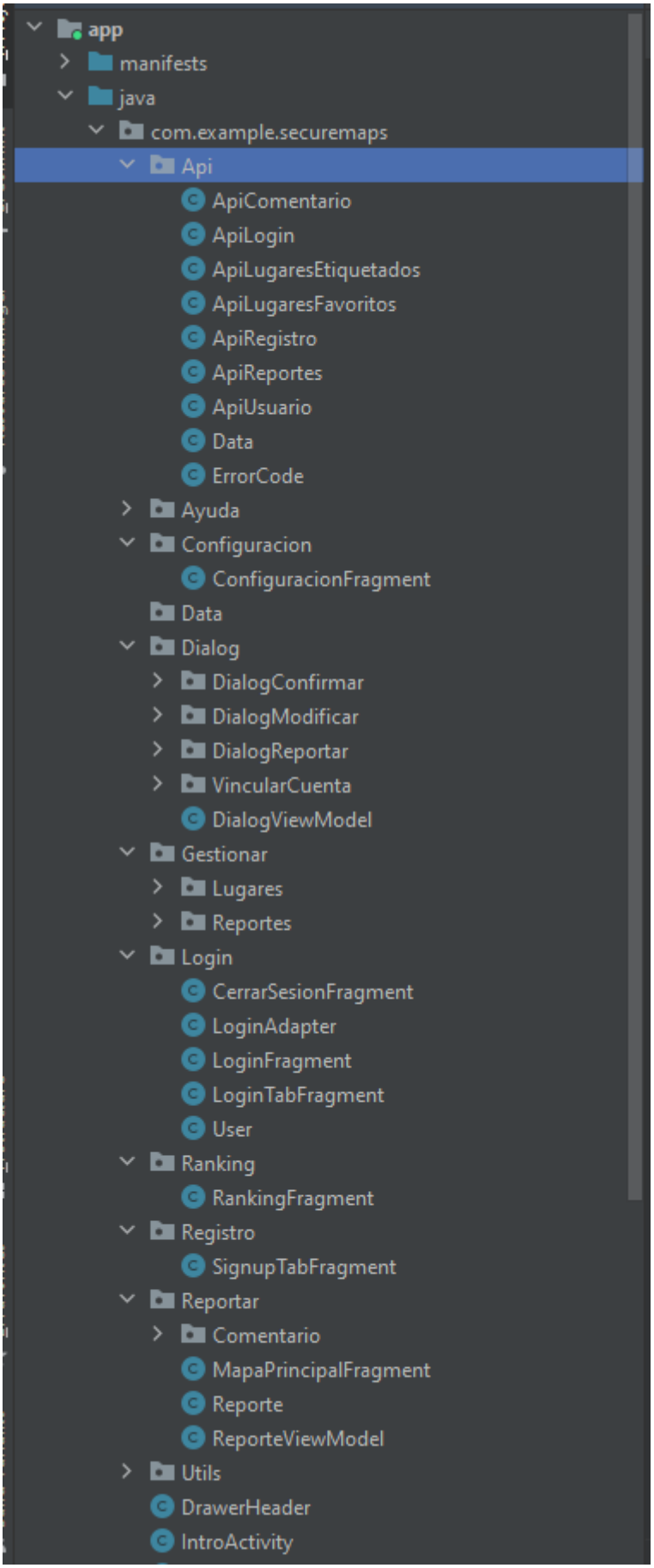
Java es el principal lenguaje en el diseño con Android. Hemos utilizado Java en todo momento, y con él hemos podido programar todas las funcionalidades necesarias para que nuestra aplicación interactúe tal como hemos definido al empezar el proyecto.

En el código Java del proyecto podemos encontrar multiples funcionalidades de la aplicación, como reportar incidentes, modificarlos, etc. Sin embargo, realmente todo este código lo que hace es llamar a la API para realizar las operaciones necesarias en la base de datos, y que todos estos cambios queden guardados de forma permanente.

La comunicación con la API es algo muy importante, ya que sin esta los cambios no se podrían guardar en un servidor externo, y esto es uno de los requisitos de la aplicación. Estas llamadas a la API, también están definidas y programadas en el código Java, sin embargo hemos utilizado una librería desarrollada exclusivamente para esto llamada “Retrofit”.

El uso de librerías en el código es algo muy común y eficaz, ya que hay librerías diseñadas exclusivamente para realizar una funcionalidad, y esta cumple perfectamente los requisitos necesarios. De echo, la mayoría de veces es mejor utilizar librerías, ya que están correctamente diseñadas para realizar una funcionalidad concreta de forma segura y eficaz.

Al desarrollar una aplicación compleja y con diferentes funcionalidades, es muy común que el código cada vez sea más largo y difícil de entender. Por ello, es importante tenerlo correctamente estructurado, de manera que, a la hora de modificar o ampliar funcionalidades en él, sea fácil de encontrar lo que andamos buscando.
En este proyecto, hemos estructurado el código por diferentes clases y a su vez, estas clases están dentro de diferentes paquetes. A continuación podemos ver un ejemplo sobre esto.


app
manifests
java
com.example.securemaps
Api
ApiComentario
ApiLogin
ApiLugaresEtiquetados
ApiLugaresFavoritos
ApiRegistro
ApiReportes
ApiUsuario
Data
ErrorCode
Ayuda
Configuracion
ConfiguracionFragment
Data
Dialog
DialogConfirmar
DialogModificar
DialogReportar
VincularCuenta
DialogViewModel
Gestionar
Lugares
Reportes
Login
CerrarSesionFragment
LoginAdapter
LoginFragment
LoginTabFragment
User
Ranking
RankingFragment
Registro
SignupTabFragment
Reportar
Comentario
MapaPrincipalFragment
Reporte
ReporteViewModel
Utils
DrawerHeader
IntroActivity

Tal y como está organizado, es mucho más fácil encontrar el código que debemos de modificar o ampliar, ya que dentro de cada paquete, únicamente se encuentran las clases que hacen referencia a este mismo. Por ejemplo, en el paquete “Login”, podemos encontrar todas las clases que contienen las funcionalidades del Login.

El proyecto está dividido en “Fragments”. Estos “Fragments” son diferentes pantallas dentro de un mismo “Activity”, la ventaja de esto consiste en poder acceder todas las variables desde los diferentes “Fragments”, ya que siempre trabajamos bajo un mismo Activity.

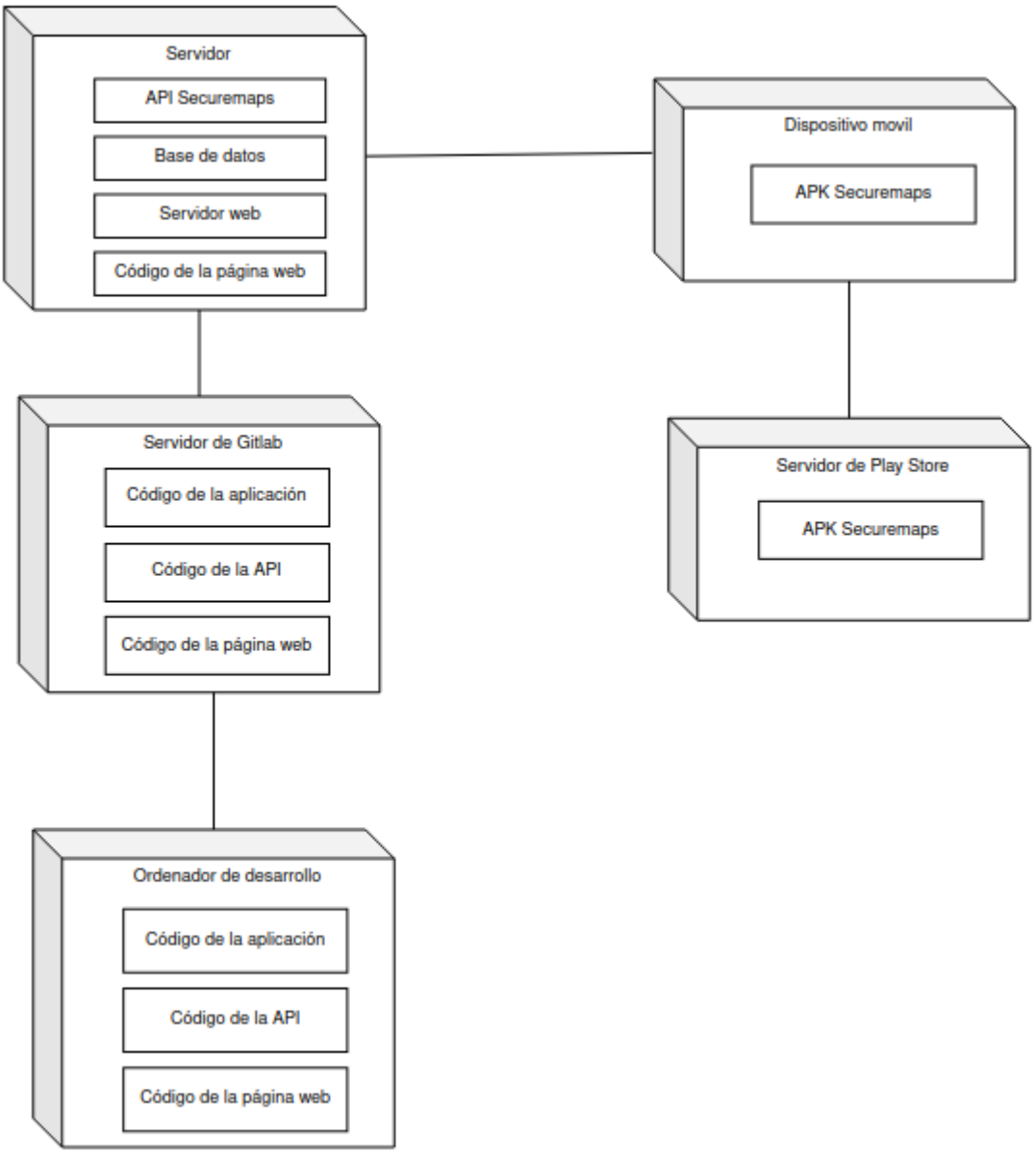
Por último, cabe destacar que para poder desarrollar el mismo proyecto y de forma simultánea, hemos utilizado Git. Esta es una tecnología que permite a varios desarrolladores trabajar en un mismo proyecto, realizando modificaciones y actualizaciones en este, incluso permite juntar los cambios en una misma rama del proyecto, siempre que estos no se solapen y están realizados en diferentes ficheros o diferentes partes del código fuente.

Durante este apartado hemos explicado de forma muy resumida cómo funciona el código de la aplicación Secure Maps, no obstante se puede consultar con más detalle todo el código en el repositorio de Gitlab.

Diagrama de despliegue

Los diagramas de despliegue se utilizan para visualizar los procesadores/ nodos/dispositivos de hardware de un sistema, los enlaces de comunicación entre ellos y la colocación de los archivos de software en ese hardware.

En este proyecto, tenemos diferentes dispositivos hardware, ya que interactuamos siempre desde un dispositivo móvil con la API y la base de datos ubicados en el servidor. A continuación mostraremos el diagrama de despliegue correspondiente.



Preparación del servidor

En este proyecto la implementación del servidor es fundamental, ya que en él alojaremos la página web, la API y la base de datos. Es necesario que el servidor tenga un buen rendimiento para que todos estos procesos se ejecuten tal y como esperamos.

Características

En este caso como servidor hemos usado una Raspberry PI 4, a continuación podemos ver sus características:

Sistema en un chip	Broadcom BCM2711
CPU	Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72
GPU	VideoCore VI
Memoria	2GB LPDDR4 RAM
Conectividad	802.11ac Wi-Fi / Bluetooth 5.0, Gigabit Ethernet
Vídeo y sonido	2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto
Puertos	2 x USB 3.0, 2 x USB 2.0
Alimentación	5V/3A vía USB-C, 5V vía cabezal GPIO
Expansión	Cabezal GPIO de 40 pines
Almacenamiento	1TB HDD

En la Raspberry, hemos instalado el Sistema Operativo Ubuntu Server 20.04 y le hemos asignado el nombre de host de “raspberry”.

```
ubuntu@raspberry:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.2 LTS
Release:        20.04
Codename:       focal
ubuntu@raspberry:~$
```

De forma estándar, el Sistema Operativo recomendado para las Raspberry es Raspberry Pi OS, sin embargo hemos optado por instalar un Ubuntu Server 20.04 ya que este no dispone de una interfaz gráfica, y de esta manera los recursos asignados a la interfaz gráfica (Memoria, CPU, GPU) podemos aprovecharla asignando a otros recursos que necesitaremos.

En este proyecto, la interfaz gráfica no es algo necesario ya que todas las acciones que ejecutaremos en el servidor podemos realizarlas mediante comandos.

Virtualización

Tal como hemos comentado anteriormente, una de las decisiones que hemos tomado ha sido la de virtualizar el servidor que utilizaremos para el proyecto.

Virtualizando el servidor, obtenemos las ventajas de poder realizar copias de seguridad en formato de “snapshots” para que si en algún momento el servidor deja de funcionar, podemos replicar su configuración y virtualizarlo de nuevo utilizando estas copias de seguridad. Además, utilizando la tecnología LXC Container, podemos virtualizar un contenedor permitiendo que este utilice todos los recursos de la máquina física, de esta manera podemos aprovechar todo el rendimiento de esta.

A continuación vamos a mostrar los comandos necesarios para crear un contenedor con LXC.

- Debemos instalar lxc con snap.
  - sudo snap install lxd
- Iniciar la configuración.
  - sudo lxd init
- Crear el contenedor, indicando el sistema operativo y el nombre del mismo.
  - lxc launch ubuntu:20.04 securemaps-db
- Iniciar el contenedor.
  - lxc start securemaps-db

Siguiendo los pasos indicados, habremos creado e iniciado el contenedor “securemaps-db” utilizando como Sistema Operativo ubuntu 20.04.

Podemos comprobar la creación del mismo usando el siguiente comando: lxc list securemaps-db

```
ubuntu@raspberry:~$ lxc list securemaps-db
+---------------+---------+----------------------+----------------------------------------------+-----------+-----------+
|     NAME      |  STATE  |         IPV4         |                     IPV6                     |   TYPE    | SNAPSHOTS |
+---------------+---------+----------------------+----------------------------------------------+-----------+-----------+
| securemaps-db | RUNNING | 10.148.102.14 (eth0) | fd42:acb7:a684:9a2:216:3eff:fed4:2edc (eth0) | CONTAINER | 2         |
+---------------+---------+----------------------+----------------------------------------------+-----------+-----------+
ubuntu@raspberry:~$
```

En la imagen anterior podemos ver un resumen del contenedor. Incluye el nombre, el estado, la IP que tiene y los snapshots del mismo.

Cabe destacar que la IP es privada y por lo tanto únicamente se podrá visualizar desde la Raspberry. Esto causa que a la hora de hacer alguna redirección de puertos, deba de pasar siempre por la raspberry y sea esta la que elija a qué IP enviar esta petición. Utilizando este método, como ventaja obtendremos una mayor seguridad ya que la máquina “securemaps-db” no es visible en la red y únicamente es visible para la Raspberry, así que usamos la Raspberry como firewall para procesar las peticiones.

Redirección de puertos

Para poder comunicarnos desde Internet con el servidor, es necesario abrir unos puertos en el Router de la red en la que se encuentra la Raspberry.

A continuación vamos a mostrar como configurar las redirecciones de puertos necesarias para que la aplicación se comunique con la API y tengamos acceso a la página web.

Para poder aplicar la configuración correcta, es importante saber que puertos utilizará cada servicio.

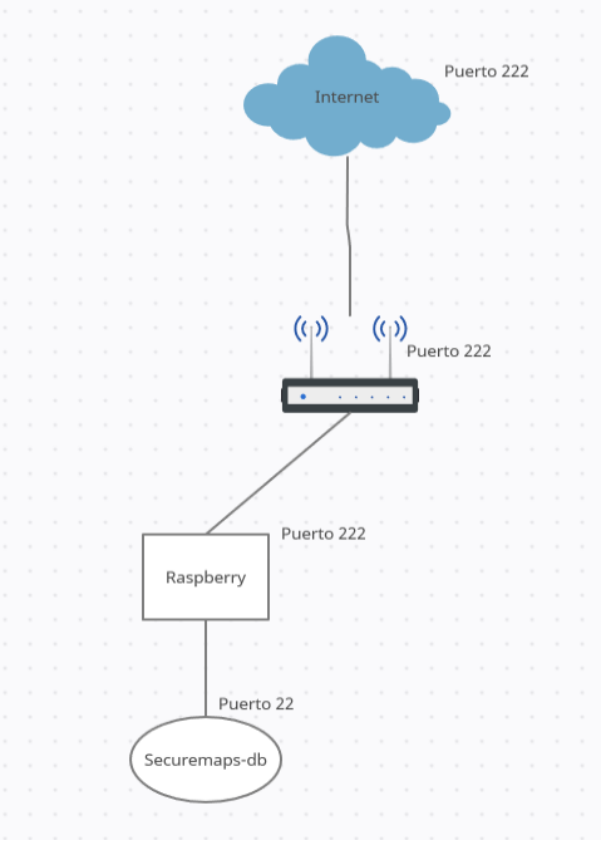
A continuación mostramos una tabla para simplificarlo.

Servicio	**Puerto del servidor**	**Puerto de la Raspberry**	**Puerto público del router**
Apache (HTTP)	80	8080	80
Apache (HTTPS)	443	4430	443
API (NodeJs)	8444	8444	8444
SSH	22	222	222

Como hemos comentado anteriormente, que el servidor tenga una IP de un rango privado que únicamente pueda comunicarse con él la raspberry, nos obliga a que todo el tráfico destinado al servidor pase por la raspberry. Por ello, el puerto público no siempre es el mismo que el puerto privado del servidor.

Por ejemplo, el servicio SSH es accesible desde Internet mediante el puerto 222. La petición pasará desde el Router a la Raspberry por el mismo puerto 222, sin embargo, la raspberry lo procesa y enviará la petición al puerto 22 del servidor “securemaps-db”.

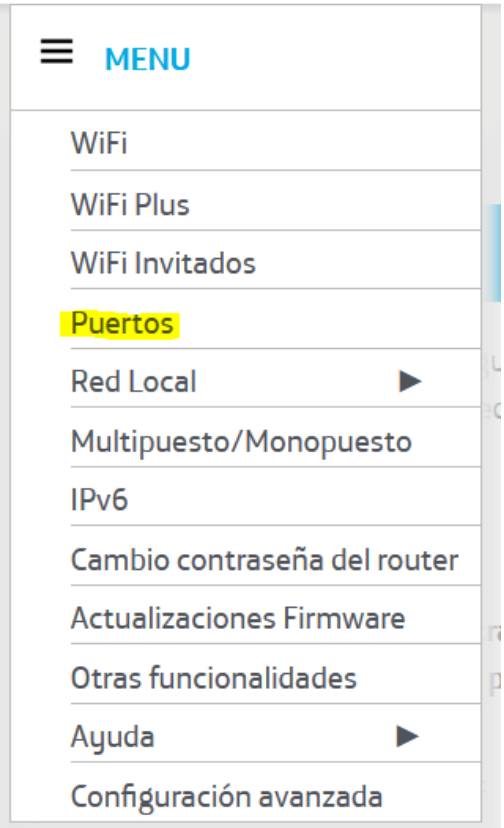
A continuación mostramos un diagrama de una petición SSH en la red.



Router

El siguiente paso será aplicar esta configuración en nuestro router. Para ello, accederemos a la interfaz web poniendo la IP de nuestro router en cualquier navegador. En nuestro caso la IP es 192.168.1.1, pero esto puede cambiar dependiendo del proveedor de internet.

Una vez hagamos login en el Router, debemos buscar alguna opción que muestre “Redirección de puertos” o similar.



Una vez entremos en la configuración de puertos, debemos aplicar la configuración que hemos mostrado anteriormente en la tabla.

En nuestro caso, la IP de la Raspberry es la “192.168.1.201”. Debe quedar algo similar a la siguiente imagen.

Una vez aplicada la configuración en el router, los puertos indicados quedarán abiertos en Internet, pero debemos de configurar las redirecciones en la Raspberry, ya que todos estos servicios estarán activos en el servidor “securemaps-db”.

Servidor

El siguiente paso será replicar esta configuración en el firewall de la Raspberry, para que todas las peticiones que vayan a los puertos mencionados anteriormente, automaticamente se reenvien a la IP del servidor.

Para ello, utilizaremos Iptables, el firewall de Ubuntu. Crearemos un script con el siguiente contenido, y a continuación lo ejecutaremos.

```
#!/bin/bash
# Para la api securemaps
sudo iptables -t nat -A PREROUTING -p tcp --dport 8444 -j DNAT --to-destination 10.148.102.14:8444
# Redirecciona el ssh al contenedor de securemaps
sudo iptables -t nat -A PREROUTING -p tcp --dport 222 -j DNAT --to-destination 10.148.102.14:22
# Redirecciona pagina web
sudo iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to-destination 10.148.102.14:80
sudo iptables -t nat -A PREROUTING -p tcp --dport 4430 -j DNAT --to-destination 10.148.102.14:443
~
```

```
ubuntu@raspberry:~$ vim iptables.sh
ubuntu@raspberry:~$ ./iptables.sh
```

Con el parámetro --dport indicamos el puerto por el cual llegará la petición a la raspberry, y posteriormente con el parámetro --to-destination indicamos la IP del servidor “securemaps-db” y el puerto de destino del mismo.

Una vez aplicada esta configuración, podemos realizar una prueba conectándonos por ssh al puerto 222 de nuestra IP pública para comprobar si la redirección se está aplicando correctamente.

Para averiguar cual es nuestra IP pública, podemos recurrir a la siguiente web /www.cual-es-mi-ip.net.

```
ubuntu@raspberry:~$ ssh -p 222 83.57.10.172
ubuntu@83.57.10.172's password:
```

Efectivamente, podemos comprobar que la redirección del puerto 222 al servicio SSH se está aplicando correctamente, ya que nos pide el usuario y contraseña del servidor.

Configuración interna

Una vez configuradas las redirecciones necesarias, vamos a ingresar al servidor y configurar los parámetros necesarios y los servicios.

Para empezar, entraremos en el contenedor usando el comando.

- lxc exec securemaps-db bash

Una vez nos encontramos dentro del servidor, crearemos el usuario “securemaps” con el comando.

- sudo adduser securemaps

Ahora, agregaremos este usuario al grupo con privilegios, para poder ejecutar acciones desde el sin requerir de “root”. Para ello, usamos los siguientes comandos.

- visudo
- Añadimos la siguiente línea: securemaps ALL=(ALL:ALL) ALL

Una vez el usuario “securemaps” tenga privilegios, vamos a proceder con la instalación de los servicios deseados.

Velocidad de conexión

Una vez configurado el servidor, vamos a realizar un test de velocidad para comprobar que la conexión con este, desde fuera de la red, será estable y rápida.

Al no tener entorno grafico, no podemos usar un navegador web para realizar la prueba, pero tenemos una aplicación llamada “speedtest” disponible en Linux, que nos permitirá realizar una prueba de velocidad desde la misma terminal. Vamos a instalar esta herramienta.

- Instalamos pip:
  - sudo apt install python3-pip

- Instalamos speedtest
  - sudo pip3 install speedtest-cli

- Iniciamos el test.
  - speedtest-cli

Una vez lo iniciamos, empezará a realizar el test y cuando acabe nos dará un resultado sobre la velocidad de descarga y subida de nuestro servidor.

Como podemos ver, obtenemos una muy buena velocidad para que las peticiones a este sean estables y rápidas.

Configuración del dominio

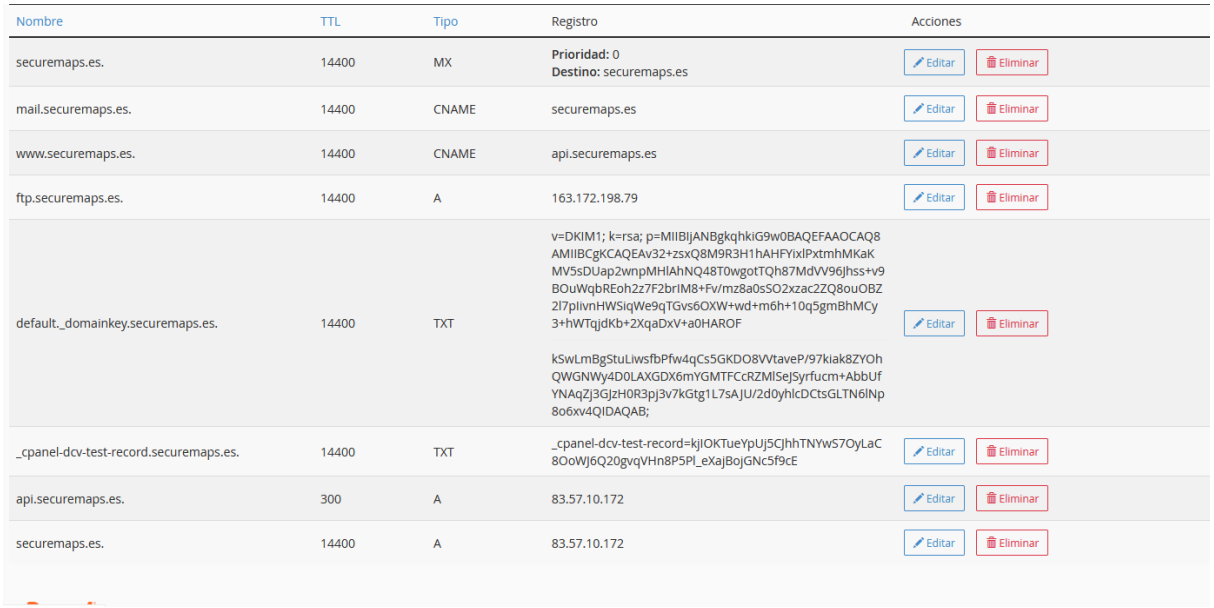
Para realizar el proyecto de una forma más profesional, hemos decidido adquirir un dominio llamado **securemaps.es**, usando esta opción podremos utilizar los servicios de la API y página web sin tener que utilizar la IP de nuestro router.

A la hora de comprar el dominio, hemos optado por adquirirlo en afxsolutions.com, ya que es una empresa local de Santa Coloma de Gramanet, en la que uno de los integrantes del grupo trabajamos.

Una vez comprado el dominio, únicamente hemos tenido que redirigir la IP del dominio, a la IP pública del router de la red en la que está alojada en servidor.

Además, hemos añadido un subdominio llamado **api.securemaps.es** para utilizarlo como acceso a la API que utilizará nuestra aplicación.

Para configurar esto, únicamente debemos de modificar las zonas DNS de nuestro dominio. Este proceso puede ser distinto dependiendo del proveedor del dominio. En nuestro caso con AFXSolutions, hemos realizado este proceso mediante cPanel. A continuación mostramos una imagen del resultado final.



Nombre	TTL	Tipo	Registro	Acciones
securemaps.es.	14400	MX	**Prioridad:** 0 **Destino:** securemaps.es	Editar Eliminar
mail.securemaps.es.	14400	CNAME	securemaps.es	Editar Eliminar
www.securemaps.es.	14400	CNAME	api.securemaps.es	Editar Eliminar
ftp.securemaps.es.	14400	A	163.172.198.79	Editar Eliminar
default._domainkey.securemaps.es.	14400	TXT	v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAv32+zsxQ8M9R3H1hAHFYixlPxtmhMKaKMV5sDUap2wnpMHlAhNQ48T0wgotTQh87MdVV96Jhss+v9BOuWqbREoh2z7F2brIM8+Fv/mz8a0sSO2xzac2ZQ8ouOBZ2l7pIivnHWSiqWe9qTGvs6OXW+wd+m6h+10q5gmBhMCy3+hWTqjdKb+2XqaDxV+a0HAROF kSwLmBgStuLiwsfbPfw4qCs5GKDO8VVtaveP/97kiak8ZYOhQWGNWy4D0LAXGDX6mYGMTFCcRZMlSeJSyrfucm+AbbUfYNAqZj3GJzH0R3pj3v7kGtg1L7sAJU/2d0yhlcDCtsGLTN6lNp8o6xv4QIDAQAB;	Editar Eliminar
_cpanel-dcv-test-record.securemaps.es.	14400	TXT	_cpanel-dcv-test-record=kJlOKTueYpUj5CJhhTNYwS7OyLaC8OoWJ6Q20gvqVHn8P5Pl_eXajBojGNc5f9cE	Editar Eliminar
api.securemaps.es.	300	A	83.57.10.172	Editar Eliminar
securemaps.es.	14400	A	83.57.10.172	Editar Eliminar

Una vez realizado esto, nos podremos dirigir a nuestro servidor mediante el dominio securemaps.es o api.securemaps.es. Podemos realizar una comprobación haciendo “ping” para comprobar que las peticiones se reciben correctamente.

```
juanmi@DESKTOP-GBUURD7:~$ ping securemaps.es
PING securemaps.es (83.57.10.172) 56(84) bytes of data.
64 bytes from 172.red-83-57-10.dynamicip.rima-tde.net (83.57.10.172): icmp_seq=1 ttl=64 time=0.671 ms
64 bytes from 172.red-83-57-10.dynamicip.rima-tde.net (83.57.10.172): icmp_seq=2 ttl=64 time=0.697 ms
^C
--- securemaps.es ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.671/0.684/0.697/0.013 ms
juanmi@DESKTOP-GBUURD7:~$ ping api.securemaps.es
PING api.securemaps.es (83.57.10.172) 56(84) bytes of data.
64 bytes from 172.red-83-57-10.dynamicip.rima-tde.net (83.57.10.172): icmp_seq=1 ttl=64 time=0.660 ms
64 bytes from 172.red-83-57-10.dynamicip.rima-tde.net (83.57.10.172): icmp_seq=2 ttl=64 time=0.723 ms
^C
--- api.securemaps.es ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.660/0.691/0.723/0.031 ms
juanmi@DESKTOP-GBUURD7:~$
```

Instalación de servicios

Base de datos

Como hemos comentado anteriormente, el gestor de base de datos que vamos a utilizar será postgres. En nuestro servidor, podemos instalarlo utilizando el siguiente comando.

```
securemaps@securemaps-db:~$ sudo apt install postgresql postgresql-contrib
Reading package lists... Done
Building dependency tree
Reading state information... Done
postgresql is already the newest version (12+214ubuntu0.1).
The following NEW packages will be installed:
  postgresql-contrib
0 upgraded, 1 newly installed, 0 to remove and 23 not upgraded.
Need to get 3932 B of archives.
After this operation, 67.6 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Una vez instalado, podemos comprobarlo usando el siguiente comando.

```
securemaps@securemaps-db:~$ sudo su postgres
postgres@securemaps-db:/home/securemaps$ psql
psql (12.6 (Ubuntu 12.6-0ubuntu0.20.04.1))
Type "help" for help.

postgres=#
```

Si se abre la consola de postgres, significa que se ha instalado correctamente. De todas formas, podemos comprobar que el servicio esté activo usando: **sudo service postgresql status**.

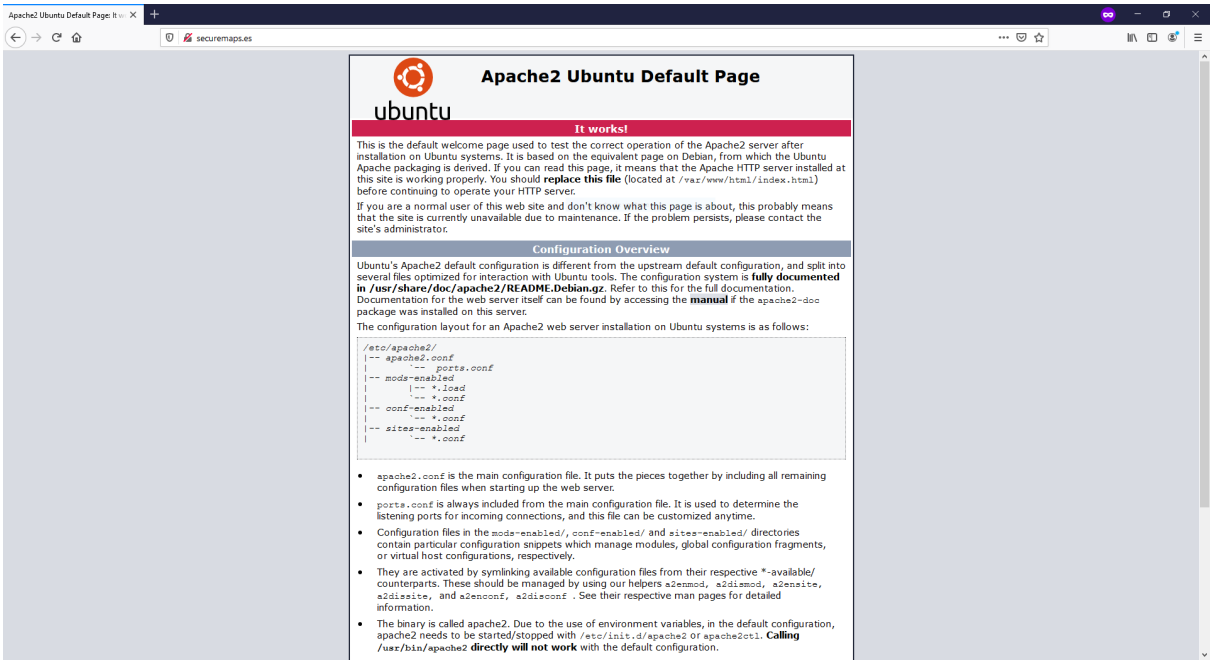
```
securemaps@securemaps-db:~$ sudo service postgresql status
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
     Active: active (exited) since Thu 2021-05-20 15:24:55 UTC; 3 days ago
   Main PID: 397 (code=exited, status=0/SUCCESS)
      Tasks: 0 (limit: 2100)
     CGroup: /system.slice/postgresql.service

May 20 15:24:55 securemaps-db systemd[1]: Starting PostgreSQL RDBMS...
May 20 15:24:55 securemaps-db systemd[1]: Finished PostgreSQL RDBMS.
securemaps@securemaps-db:~$
```

Servidor web

El siguiente paso será instalar Apache, el servidor web deseado para alojar nuestra página web del proyecto. Para ello, usamos el siguiente comando.

```
securemaps@securemaps-db:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.41-4ubuntu3.1).
0 upgraded, 0 newly installed, 0 to remove and 23 not upgraded.
securemaps@securemaps-db:~$
```

Ahora, podemos comprobar el correcto funcionamiento del servidor web ingresando con la url securemaps.es en cualquier navegador web, y se deberá de mostrar la página de inicio de Apache.



El siguiente paso será configurar HTTPS para que el acceso a la página web sea encriptado, de echo, también utilizaremos este certificado HTTPS para encriptar la conexión entre la API y la aplicación, de esta manera todos los datos que se envíen por el body o por la URL estarán encriptados y seguros.

Para crear este certificado HTTPS, utilizaremos la entidad certificadora Let's Encrypt, que nos ofrece de forma libre y gratuita un certificado HTTPS totalmente seguro.

Para instalar este certificado en nuestro servidor, utilizaremos Certbot, un software que nos automatizará la instalación y configuración del mismo. Vamos a realizar los siguientes comandos.

- Instalamos certbot
  - **sudo snap install --classic certbot**
- Creamos un enlace simbólico necesario para certbot
  - **sudo ln -s /snap/bin/certbot /usr/bin/certbot**
- Iniciamos certbot
  - **sudo certbot --apache**
- Ahora seguimos todos los pasos hasta finalizar la instalación.

Podemos comprobar como en la carpeta /etc/apache2/sites-available, se han creado diferentes archivos de configuración.

Copiamos el fichero 000-default.conf con el nombre de securemaps.es.conf

Ahora, volvemos a ejecutar el siguiente comando para crear un certificado para el dominio **securemaps.es**, ya que anteriormente hemos creado el certificado para el subdominio **api.securemaps.es**

- Creamos el certificado
  - sudo certbot -d securemaps.es

- Seguimos los pasos para que cree este nuevo certificado.

Ahora, nos quedarán los siguientes archivos en /etc/apache2/sites-available.

```
securemaps@securemaps-db:~$ ls /etc/apache2/sites-available/
000-default-le-ssl.conf  000-default.conf  default-ssl.conf  securemaps.es-le-ssl.conf  securemaps.es.conf
securemaps@securemaps-db:~$
```

Por último, vamos a editar los archivos **000-default.conf** y **securemaps.es.conf**, para que automáticamente cuando accedemos mediante el puerto 80, se nos redirija al puerto 443 (https) y tener una conexión segura y cifrada en cualquier caso.

Para ello, vamos a añadir las siguientes líneas en ambos ficheros.

000-default.conf

```
        #Include conf-available/serve-cgi-bin.conf
RewriteEngine on
RewriteCond %{SERVER_NAME} =api.securemaps.es
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
```

securemaps.es.conf

```
RewriteEngine on
RewriteCond %{SERVER_NAME} =securemaps.es
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
```

Por último, reiniciamos el servicio de apache con el comando: **sudo service apache2 restart**.

Siguiendo todos los pasos anteriores hemos creado dos virtualhosts, securemaps.es y api.securemaps.es. En ambos hemos creado certificados HTTPS con Let's Encrypt, y hemos redirigido las conexiones HTTP a HTTPS de forma automatica para que los datos siempre se envien seguros y encriptados.

Para comprobar que toda esta configuración se ha realizado, podemos ingresar en cualquier navegador y comprobar ambas url.

Api

El siguiente paso será instalar todos los servicios necesarios para poder ejecutar la API que, posteriormente crearemos.

Utilizaremos el lenguaje JavaScript, y su intérprete es NodeJs, por ello es necesario instalarlo en el servidor. A su vez, instalaremos NPM, para poder instalar los recursos necesarios para el correcto funcionamiento, ya que NPM lo hará todo mucho más fácil.

```
securemaps@securemaps-db:~$ sudo apt install npm
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp cpp-9 dpkg-dev fakeroot
  fontconfig-config fonts-dejavu-core g++ g++-9 gcc gcc-10-base gcc-9 gcc-9-base gyp javascript-common
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
  libauthen-sasl-perl libbinutils libc-ares2 libc-dev-bin libc6 libc6-dev libcc1-0 libcrypt-dev libctf-
  libctf0 libdata-dump-perl libdpkg-perl libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1
```

Lo siguiente será descargar la API del proyecto, que en el próximo apartado explicaremos el código. Al estar subida a gitlab, podemos usar el comando “git clone” para clonar el respositorio https://gitlab.com/jmseguraf01/securemaps_api.

```
securemaps@securemaps-db:~$ git clone https://gitlab.com/jmseguraf01/securemaps_api
Cloning into 'securemaps_api'...
warning: redirecting to https://gitlab.com/jmseguraf01/securemaps_api.git/
remote: Enumerating objects: 2513, done.
remote: Counting objects: 100% (2513/2513), done.
remote: Compressing objects: 100% (1935/1935), done.
remote: Total 2513 (delta 458), reused 2439 (delta 420), pack-reused 0
Receiving objects: 100% (2513/2513), 2.16 MiB | 3.30 MiB/s, done.
Resolving deltas: 100% (458/458), done.
securemaps@securemaps-db:~$ ls securemaps_api/
app.json  database  index.js  node_modules  package.json  package-lock.json  Procfile  README.md  sql  uploads
securemaps@securemaps-db:~$
```

Ahora, usamos el comando npm start para iniciar la API.

```
securemaps@securemaps-db:~/securemaps_api$ npm start

> node-js-getting-started@0.3.0 start /home/securemaps/securemaps_api
> node index.js

Listening on 8444
```

Si ponemos la URL api.securemaps.es:8444 en cualquier navegador nos debe de salir lo siguiente.

Y efectivamente podemos comprobar cómo está funcionando. Sin embargo, de esta manera no está utilizando el protocolo HTTPS y por lo tanto, todos los datos que envíe la aplicación a la API mediante el body o la URL, podrían ser esnifados y leídos por terceros.

Para solucionar esto y utilizar el certificado HTTPS de Let’s Encrypt, generado anteriormente, debemos agregar este trozo de código en nuestra API.

```
// Para crear la sesion con HTTPS
https.createServer({
  key: fs.readFileSync('../certs/privkey.pem'),
  cert: fs.readFileSync('../certs/fullchain.pem')
}, app).listen(PORT, function(){
  console.log("My HTTPS server listening on port " + PORT + "...");
});
```

Por último, para que funcione debemos de copiar los certificados en la carpeta **~/certs**, que por defecto están ubicados en **/etc/letsencrypt/live/api.securemaps.es**.

```
securemaps@securemaps-db:~$ cd certs/
securemaps@securemaps-db:~/certs$ sudo cp /etc/letsencrypt/live/api.securemaps.es/fullchain.pem .
securemaps@securemaps-db:~/certs$ sudo cp /etc/letsencrypt/live/api.securemaps.es/privkey.pem .
securemaps@securemaps-db:~/certs$ ls
fullchain.pem  privkey.pem
securemaps@securemaps-db:~/certs$
```

De nuevo, iniciamos la API y volvemos a acceder mediante un navegador web a la URL https://api.securemaps.es:8444. v

```
securemaps@securemaps-db:~/securemaps_api$ npm start

> node-js-getting-started@0.3.0 start /home/securemaps/securemaps_api
> node index.js

My HTTPS server listening on port 8444...
```

Implementación de la API

Para satisfacer todas las necesidades del proyecto, hemos decidido crear una API propia para así poder comunicar la aplicación con la base de datos mediante esta.

Como hemos comentado anteriormente, hemos decidido utilizar el lenguaje de programación JavaScript para desarrollar la API, utilizando como motor NodeJs, que ya hemos instalado en nuestro servidor.

Nuestra API contiene todas las funcionalidades de la aplicación, ya que es la que se comunica con la base de datos de forma directa. Esto nos obliga a desarrollar un código extenso y complejo, que a su vez debe estar bien estructurado para poder realizar modificaciones y ampliaciones de forma sencilla.

A la hora de realizar el desarrollo, hemos decidido utilizar únicamente los métodos *"GET"* y *"POST"*, aún sabiendo que hay más (*PUT, PATCH, DELETE*). Esta decisión la hemos

tomado debido a que, buscando información en Internet e incluso contactando con desarrolladores profesionales, muchas API's profesionales únicamente usan estos dos métodos, ya que satisfacen las necesidades de todos los demás y estos últimos han quedado en desuso.

POST

En nuestra API, únicamente utilizamos el método POST cuando enviamos datos desde la aplicación Android hacia la API. Una vez recibido estos datos, nuestra API ejecuta una operación en la base de datos y devuelve un código de ejecución a la aplicación. Este código puede ser 200 si la operación se ha ejecutado con éxito o 400 si ha habido algún error. Podemos ver un ejemplo a continuación.

```
// Actualiza los datos del usuario
app.post("/updateUser", restrict, async (req, res) => {
      result = await db.login.update(usuario_id, req.body.nuevo_usuario, req.body.nuevo_email);
      if (result > 0) {
        res.send(200);
      } else {
        res.send(400);
      }
});
```

En la imagen anterior podemos ver la operación "***/updateUser***" donde se actualizan los datos del usuario en la base de datos. Esta operación recibe diferentes parámetros por el body utilizando el método POST. Posteriormente estos parámetros se envían al método update, situado en el archivo login, y es este el que realiza la operación directamente con la base de datos. Podemos verlo a continuación.

```
// Actualiza usuario
update(id, usuario, email) {
  return this.db.result("UPDATE login set usuario = $2, email = $3 where id = $1", [id, usuario, email], result => result.rowCount);
}
```

Una vez realizada esta operación, devuelve la variable "result", que es un número entero que indica las filas que se han modificado en la base de datos.

Por último, en el código comprobamos que esta variable "*result*" sea mayor a uno, ya que esto significa que se ha realizado algún cambio, y dependiendo de esto enviamos a la aplicación un código de ejecución u otro.

```
      if (result > 0) {
        res.send(200);
      } else {
        res.send(400);
      }
```

De esta manera, utilizamos el método POST en nuestra API aplicándolo a diferentes funciones, ya que únicamente hemos explicado a modo de ejemplo, la función de cambiar los datos del usuario.

GET

El otro método utilizado para obtener datos es el método GET. Este lo usamos para realizar la operación contraria al método POST. Desde Android queremos obtener unos datos, y por lo tanto hacemos una petición a la API para que esta se comunique con la base de datos y nos envíe los datos solicitados. No obstante, en este caso también podemos enviar algún parámetro concreto para realizar la consulta en la BBDD mediante una condición. Por ejemplo, a continuación explicaremos el método GET utilizado para obtener todos los reportes de una ciudad.

```
// Obtiene los reportes por la ciudad
app.get('/getReportesWithCiudad', restrict, async (req, res) => {
  res.send({results: await db.reporte.getWithCiudad(req.query.ciudad)});
});
```

En este caso, realizamos una petición al fichero reporte y método getWithCiudad, que vamos a ver, enviándole como variable el parámetro “ciudad” que cogemos de la llamada a la API.

```
// Obtiene reportes consultando la ciudad
getWithCiudad(ciudad) {
  return this.db.any("SELECT * FROM reporte WHERE ciudad = $1", [ciudad]);
}
```

Podemos observar como la API realiza esta consulta a la base de datos, pidiendo todos los reportes que tengan como ciudad la variable “ciudad”.

En este caso, la variable “ciudad” sería la condición para realizar la consulta a la base de datos. Esta variable, se envía desde Android hacia la API y esta la recoge como “req.query.ciudad”. Utilizando el método GET, estos parámetros se deben de enviar por la URL y no por el body, por lo tanto la consulta vista desde la URL quedaría de la siguiente manera: https://api.securemaps.es:8444/getReportesWithCiudad?ciudad=Barcelona.

Una vez la consulta se realiza, se devuelven los datos extraídos y estos se envían a Android mediante un formato JSON, ya que este es el formato por defecto para coger los datos de una API. Esto lo podemos ver en el siguiente código.

```
  res.send({results: await db.reporte.getWithCiudad(req.query.ciudad)});
```

Ahora, la aplicación Android es la encargada de procesar estos datos. Como en este caso, la consulta a la base de datos devuelve “any”, significa que puede devolver más de un valor, por lo tanto debemos recoger la respuesta como un Array.

```
@GET(RUTA_GET_WITH_CIUDAD)
Call<Reportes> obtenerReportesPorCiudad(@Query(Data.CIUDAD) String ciudad);
```

```
public class Reportes {
    public List<Reporte> results;
}
```

Aún así, hay algunos casos en los que utilizamos GET, sin enviar datos de condición, pero GET siempre devuelve datos a Android, por eso es el método que se usa, mayoritariamente para realizar consultas con “select” en la base de datos.

Por otro lado, cabe destacar que en todos los métodos usados, ya sean GET o POST, se llama a la función “*restrict*”. Esta función se encarga de comprobar que el usuario y contraseña sean correctos. Para ello, desde la aplicación de Android se deben de enviar estos datos en cada petición.

Una llamada al método *restric*.

```
app.get('/getReportesWithCiudad', restrict, async (req, res) => {
```

A continuación podemos ver el código de la función *restrict*.

```
// Comprueba si el usuario esta logeado
async function restrict(req, res, next) {
    if(user = await db.login.auth(req.query.email, req.query.password, true)) {
      // Login correcto
      if (user.email != null && user.password != null) {
        usuario_id = user.id;
        next();
      }
      // Login incorrecto
      else {
        res.send(401);
      }
    } else {
      res.send(401);
    }
}
```

Esta función llama al fichero *login* método *auth*, que contiene el siguiente código.

```
// Autentica en el login
async auth(email, password, pwd_encriptado) {
  let res;
  try {
    if (pwd_encriptado) {
      // El password se envia encriptado
      res = await this.db.one("SELECT * FROM login WHERE email = $1 AND password = $2 LIMIT 1", [email, password]);
    }
    // El password se envia sin encriptar
    else if (!pwd_encriptado) {
      res = await this.db.one("SELECT * FROM login WHERE email = $1 AND password = crypt($2, password) LIMIT 1", [email, password]);
    }
  } catch(e){
    res = {};
  }
  return res;
}
```

Aquí comprobamos si es usuario y contraseña existen en la base de datos, y devolvemos el resultado a la función *restrict*. Esta función, comprueba si los datos devueltos no son nulos, y si son así, deja seguir con la siguiente acción. En caso de ser nulos, significa que el login es incorrecto, y por lo tanto, devuelve como código 401 (unauthorized) a la aplicación Android.

La API requiere una conexión a la base de datos y, en nuestro caso, necesita un usuario y contraseña de correo electrónico, que será desde el cual envía las notificaciones al usuario.

Todas estas credenciales deben de estar en un fichero a parte, ya que no se deben introducir en el código por seguridad, a su vez tampoco se deben de subir a Git. Este fichero, lo hemos declarado como *.env*, y en el código de la API cogemos su información de la siguiente manera.

```
auth: {
  user: process.env.EMAIL,
  pass: process.env.PASSWORD
}
```

En este caso, estamos cogiendo la variable email y password del fichero .env, para posteriormente usarlo al enviar los correos al usuario.

Durante todo este apartado hemos explicado de forma resumida cómo funciona la API. No obstante, el desarrollo de la misma tiene mucho trabajo y se puede consultar el código de forma más detallada desde el repositorio de gitlab.

Creación base de datos

Por último, pero no menos importante, vamos a explicar como esta estructurada y creada la base de datos. Como hemos comentado anteriormente, como gestor de base de datos usamos postgresql y realizamos todas las acciones sobre esta, en su propia terminal.

Para empezar, debemos de crear una base de datos, en nuestro caso la hemos llamado "**securemaps**" y posteriormente debemos de crear un usuario que tenga todos los permisos sobre esta base de datos.

Para ello lo primero que debemos hacer es entrar en la consola de la base de datos, con el usuario postgres, el usuario por defecto con permisos en la base de datos. Podemos usar el siguiente comando.

- **psql -h localhost -U postgresql**

Una vez dentro, creamos la base de datos.

- **create database securemaps;**

Y ahora creamos un rol de usuario

- **CREATE USER usuario WITH PASSWORD 'password';**

Posteriormente, debemos asignarle todos los permisos sobre la base de datos a este usuario.

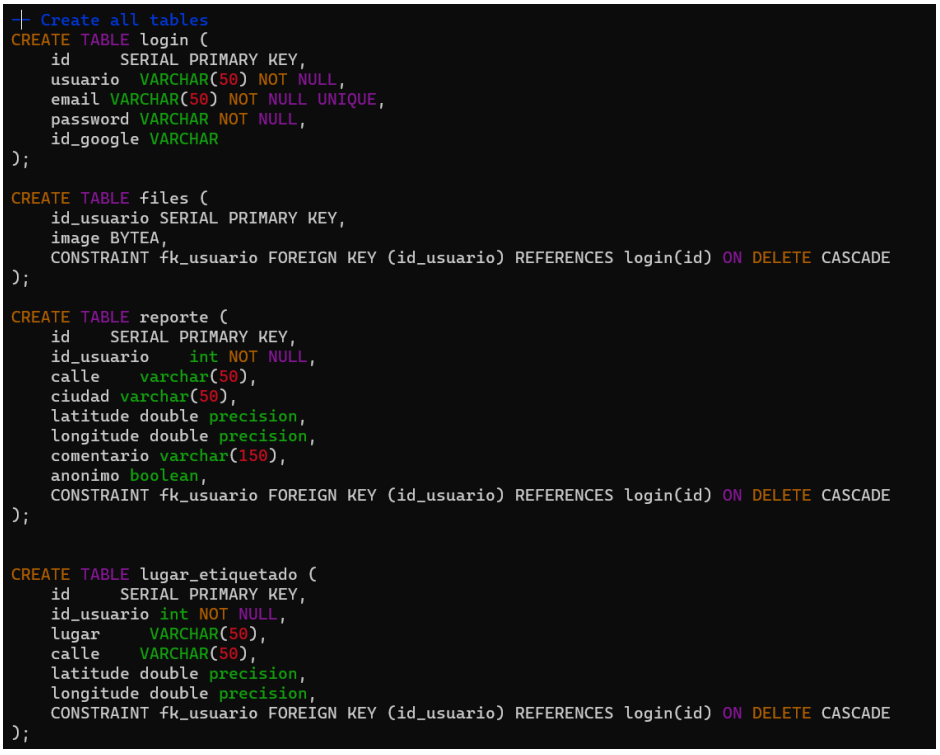
- **GRANT ALL PRIVILEGES ON DATABASE securemaps TO usuario;**

Una vez el usuario tiene todo los permisos sobre la base de datos, podemos conectarnos con ese usuario a esa base de datos y realizar todas las demás operaciones, como la creación de tablas. Para conectarnos usamos el siguiente comando.

- **psql -h loca.host -d securemaps -U usuario**

Para crear las tablas, hemos decidido crear un fichero .sql y posteriormente importarlo. De esta manera, todos los cambios que hagamos, referente a las tablas, los añadimos en ese fichero y de esta manera, en caso de tener que volver a crear toda la estructura de nuevo con importar el fichero es suficiente.

A continuación mostramos una captura del fichero, no obstante se puede encontrar en nuestro repositorio de la API.



```
-- Create all tables
CREATE TABLE login (
    id      SERIAL PRIMARY KEY,
    usuario  VARCHAR(50) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR NOT NULL,
    id_google VARCHAR
);

CREATE TABLE files (
    id_usuario SERIAL PRIMARY KEY,
    image BYTEA,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES login(id) ON DELETE CASCADE
);

CREATE TABLE reporte (
    id    SERIAL PRIMARY KEY,
    id_usuario    int NOT NULL,
    calle    varchar(50),
    ciudad varchar(50),
    latitude double precision,
    longitude double precision,
    comentario varchar(150),
    anonimo boolean,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES login(id) ON DELETE CASCADE
);


CREATE TABLE lugar_etiquetado (
    id      SERIAL PRIMARY KEY,
    id_usuario int NOT NULL,
    lugar     VARCHAR(50),
    calle    VARCHAR(50),
    latitude double precision,
    longitude double precision,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES login(id) ON DELETE CASCADE
);
```

```
CREATE TABLE lugar_favorito (
    id SERIAL PRIMARY KEY,
    id_usuario int NOT NULL,
    calle    VARCHAR(50),
    latitude double precision,
    longitude double precision,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES login(id) ON DELETE CASCADE
);

CREATE TABLE comentario_reporte (
    id SERIAL PRIMARY KEY,
    id_reporte int NOT NULL,
    id_usuario int NOT NULL,
    comentario VARCHAR(200) NOT NULL,
    CONSTRAINT fk_usuario FOREIGN KEY (id_usuario) REFERENCES login(id) ON DELETE CASCADE
);
```

Para importar este fichero, entramos en la base de datos y usamos el siguiente comando.

- **\i nombre_fichero.sql**

Podemos ver la estructura de la base de datos usando el comando **\d**.

```
pobresmap=> \d
                List of relations
 Schema |           Name            |   Type   | Owner
--------+---------------------------+----------+-------
 public | comentario_reporte        | table    | local
 public | comentario_reporte_id_seq | sequence | local
 public | login                     | table    | local
 public | login_id_seq              | sequence | local
 public | lugar_etiquetado          | table    | local
 public | lugar_etiquetado_id_seq   | sequence | local
 public | lugar_favorito            | table    | local
 public | lugar_favorito_id_seq     | sequence | local
 public | reporte                   | table    | local
 public | reporte_id_seq            | sequence | local
(10 rows)

pobresmap=>
```

En esta base de datos, hemos creado todas las tablas que necesitamos para realizar las funcionalidades descritas anteriormente. En cada una de las tablas, hemos creado un campo “id” secuencial, es decir, un contador que por cada registro suma +1. También es necesario en algunas tablas, coger claves foráneas de otras tablas, como por ejemplo en la tabla de “reporte”, cogemos la clave foránea del usuario en el campo “id_usuario”.

Por último, hemos añadido el método “*ON DELETE CASCADE*” en referencia al campo “*id_usuario*” por cada tabla. Al añadir esto, permite que al eliminarse un usuario se eliminen todos los registros de las demás tablas que hacen referencia a él.

Esto último es algo que no suelen hacer las bases de datos de aplicaciones famosas, ya que realmente no eliminan datos, sino que los ocultan para que no se puedan ver. Sin embargo, nosotros no hemos seguido esa pauta, ya que no lo consideramos ético y si el usuario desea eliminar sus datos de la aplicación, está en su derecho de eliminarlos de forma permanente de la base de datos.

Página web

Hemos desarrollado una página web para el proyecto para dar más visibilidad a este. Esta página web, únicamente contiene código en HTML y CSS muy sencillo, ya que el objetivo no es crear una web potente y compleja, sino dar visibilidad a la aplicación, que realmente es el trabajo complejo de este proyecto.

La página web, está alojada en el servidor web, y se puede acceder mediante HTTPS. Se puede acceder a ella con la URL securemaps.es.

Pruebas

Funcionals

Durante el desarrollo de la aplicación, hemos realizado diferentes pruebas sobre todas las funcionalidades de esta. Durante el desarrollo y de forma posterior a nivel de prueba, todas las funcionalidades han sido comprobadas con éxito, tal como hemos planeado al inicio del proyecto.

Aún así, al acabar toda la implementación y desarrollo de la aplicación, hemos probado a realizar un test realizando todas las acciones que envuelven a las funcionalidades de la misma, de nuevo con un resultado exitoso.

Usabilidad

Unas de nuestras pruebas sobre la aplicación, consiste en el test de Usabilidad. Tal como indica su nombre, este test se hace para comprobar que tal usabilidad tiene la aplicación enfocada a un usuario final.

Esta prueba nos va a permitir saber, de la forma más real posible, cual será la usabilidad de la aplicación a la hora de sacarla al mercado y que los usuario empiecen a descargarla. Cabe destacar que realizaremos el test con dos tipos de usuarios, ambos con diferente nivel del uso de las tecnologias, a continuación los definimos.

Usuario básico

Este usuario es un usuario con conocimientos de tecnología básicos, es decir, el usuario más común y posiblemente el tipo de usuario con más volumen de uso de la aplicación.

A continuación, vamos a mostrar una tabla sobre los resultados esperados a la hora de realizar este test a un usuario, los resultados finales y la conclusión.

Autor	Lloc	Data
Juan Miguel Segura Fernandez	Santa Coloma de Gramanet	27/05/2021

Objectiu Descripció de l'objectiu, raons i justificació de la prova

El objetivo general al que se quiere llegar con este test, es comprobar la eficiencia, usabilidad y autosuficiencia del usuario al ejecutar la aplicacion. Es necesario validar si realmente la aplicacion es apta para su uso en cualquier rango de edad, siempre contando desde la mayoria de edad (+18).

Tipus de prova Testejar funcionament, capacitat de resolució de l'usuari, etc.	Abast Es concreta la part a provar (tota l'app, una o varies funcionalitats)
Testear el funcionamiento de la aplicacion y la capacidad de autosuficiencia del usuario con esta.	Toda la aplicacion, incluyendo sus funcionalidades.

Participants Participants i tipus (expert/novell) que intervenen en la prova (stakeholders, desenvolupadors, extern, vinculat al projecte, etc.)

Alex Castro Fernandez con un nivel "basico" en el uso de tecnologias y aplicaciones.

Procediment Tasques a realitzar durant el test

Tasca	Temps minuts	Fet (%)	Descripció
1	1m	100%	Introducción al usuario de la app y explicación de la prueba
2	1m	100%	Registrarse
3	1.45m	100%	Crear, modificar y eliminar un reporte
4	1.15m	100%	Crear y eliminar un lugar etiquetado
5	1m	100%	Crear y modificar un lugar favorito
6	20s	100%	Visualizar lugar favorito
7	45s	100%	Mostrar ranking de reportes y ordenar por orden ascendente
8	1.15m	100%	Visualizar el foro de un lugar reportado
9	30s	100%	Modificar nombre de usuario
10	15s	100%	Eliminar cuenta

Treball de camp Atributs que es volen provar, mètriques i/o preguntes relacionades, valors desitjats, reals i anotacions

Atributs	Mètriques/Preguntes	Tipus (subjectiva/obj.)	Valor desitjat	Valor real	Anotacions
Efectivitat	Tasques resoltes en un temps limitat	Objetiva	Crear, eliminar y modificar reportes en el tiempo asignado anteriormente.	Cumplido	
	Percentatge de tasques completades amb èxit al primer intent.	Objetiva	Minimo un 50% de las tareas completadas en el primer intento.	Cumplido	
Eficiència	Temps transcorregut en cada pantalla	Objetiva	No mas de 20 segundos, a partir del tiempo maximo dedicado a cada tarea. Hay que tener en cuenta que cada pantalla esta asignada a una tarea.	Cumplido	
	Temps productiu	Objetiva	El tiempo total dedicado a cada funcion. Es decir, no debe demorarse mucho mas del tiempo total que suma todas las funciones que debe de realizar el usuario.	Cumplido	
Satisfacció	Nivell de dificultat	Subjetiva	Debe de ser un nivel no demasiado dificil, para que la aplicacion sea asequible a cualquier tipo de publico.	Cumplido	
	Agrada o no agrada	Subjetiva	Lo deseado es que sea una aplicacion agradable para el usuario. Aun así, cabe destacar que no se busca ser la aplicacion mas bonita del mercado, si no eficiente.	Cumplido	
Facilitat d'aprenentatge	Temps usat per acabar una tasca la primera vegada	Objetiva	El tiempo no debe ser superior a 15 segundos del tiempo asignado.	Cumplido	

Conclusions Conclusions dels resultats de la prova amb els errors detectats, les solucions plantejades i el treball futur

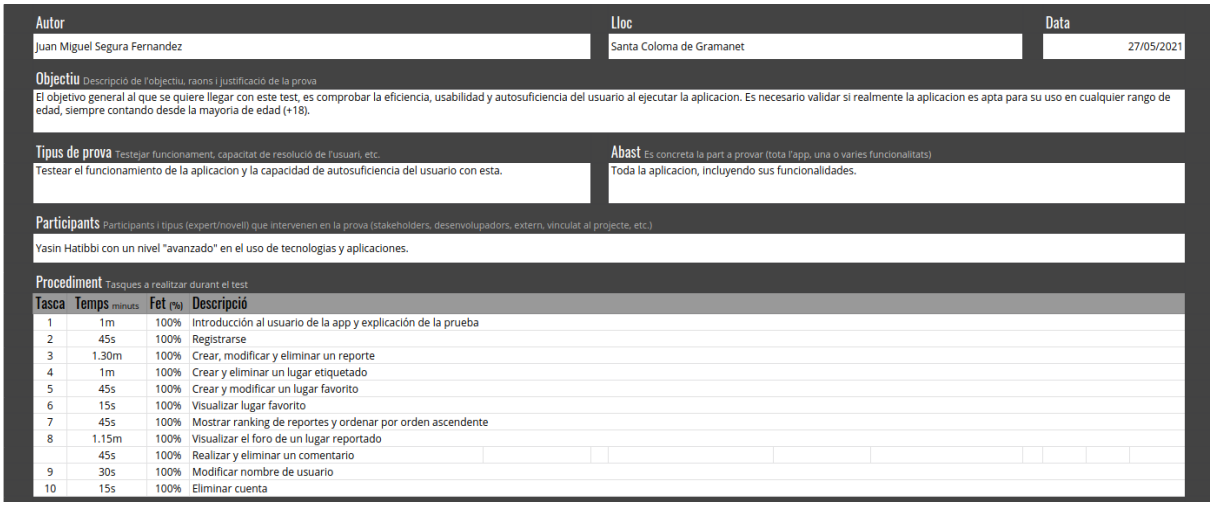
Como conclusión final despues de realizar este test de usabilidad, comprendemos que la funionalidad mas cosotas de realizar en la aplicación es marcar los lugares favoritos, ya que es la funcionalidad que se encuentra mas escondida en el menú.

Se puede ver el resultado del test desde esta hoja de cálculo.

Usuario avanzado

Este usuario es un usuario experto y avanzado con el uso de las tecnologías, perfectamente puede ser un compañero Informatico o alguien que está acostumbrado a tratar día a día con dispositivos tecnológicos.

A continuación se muestran unas capturas de pantalla del resultado del test, pero se puede consultar de forma detallada desde esta hoja de cálculo.



Autor	Lloc	Data
Juan Miguel Segura Fernandez	Santa Coloma de Gramanet	27/05/2021

Objectiu Descripció de l'objectiu, raons i justificació de la prova

El objetivo general al que se quiere llegar con este test, es comprobar la eficiencia, usabilidad y autosuficiencia del usuario al ejecutar la aplicacion. Es necesario validar si realmente la aplicacion es apta para su uso en cualquier rango de edad, siempre contando desde la mayoria de edad (+18).

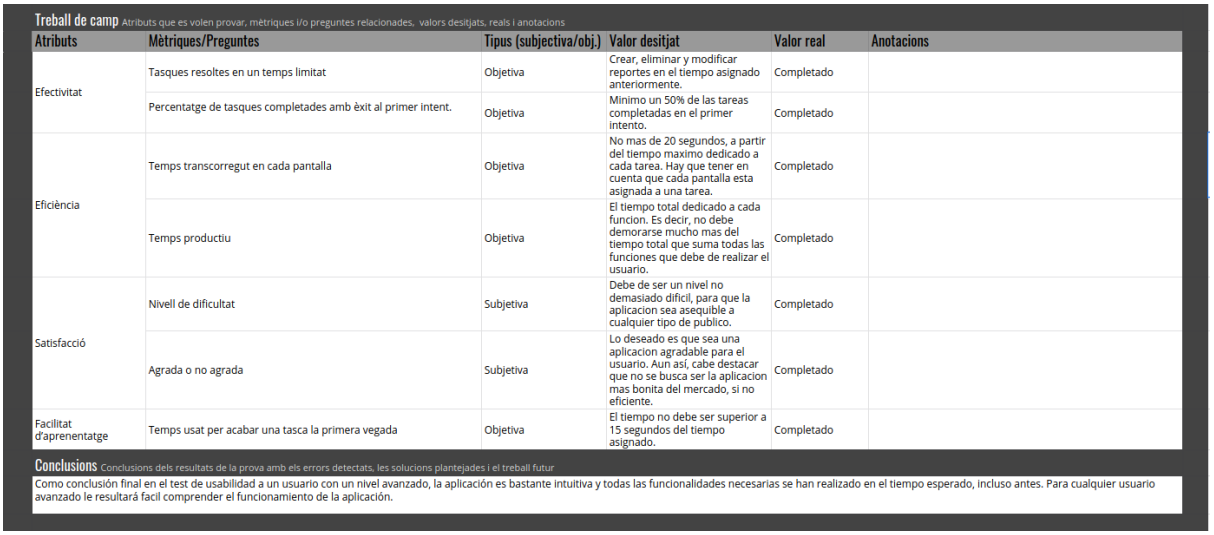
Tipus de prova Testejar funcionament, capacitat de resolució de l'usuari, etc.	Abast Es concreta la part a provar (tota l'app, una o varies funcionalitats)
Testear el funcionamiento de la aplicacion y la capacidad de autosuficiencia del usuario con esta.	Toda la aplicacion, incluyendo sus funcionalidades.

Participants Participants i tipus (expert/novell) que intervenen en la prova (stakeholders, desenvolupadors, extern, vinculat al projecte, etc.)

Yasin Hatibbi con un nivel "avanzado" en el uso de tecnologias y aplicaciones.

Procediment Tasques a realitzar durant el test

Tasca	Temps minuts	Fet (%)	Descripció
1	1m	100%	Introducción al usuario de la app y explicación de la prueba
2	45s	100%	Registrarse
3	1.30m	100%	Crear, modificar y eliminar un reporte
4	1m	100%	Crear y eliminar un lugar etiquetado
5	45s	100%	Crear y modificar un lugar favorito
6	15s	100%	Visualizar lugar favorito
7	45s	100%	Mostrar ranking de reportes y ordenar por orden ascendente
8	1.15m	100%	Visualizar el foro de un lugar reportado
	45s	100%	Realizar y eliminar un comentario
9	30s	100%	Modificar nombre de usuario
10	15s	100%	Eliminar cuenta



Treball de camp Atributs que es volen provar, mètriques i/o preguntes relacionades, valors desitjats, reals i anotacions

Atributs	Mètriques/Preguntes	Tipus (subjectiva/obj.)	Valor desitjat	Valor real	Anotacions
Efectivitat	Tasques resoltes en un temps limitat	Objetiva	Crear, eliminar y modificar reportes en el tiempo asignado anteriormente.	Completado	
	Percentatge de tasques completades amb èxit al primer intent.	Objetiva	Minimo un 50% de las tareas completadas en el primer intento.	Completado	
Eficiència	Temps transcorregut en cada pantalla	Objetiva	No mas de 20 segundos, a partir del tiempo maximo dedicado a cada tarea. Hay que tener en cuenta que cada pantalla esta asignada a una tarea.	Completado	
	Temps productiu	Objetiva	El tiempo total dedicado a cada funcion. Es decir, no debe demorarse mucho mas del tiempo total que suma todas las funciones que debe de realizar el usuario.	Completado	
Satisfacció	Nivell de dificultat	Subjetiva	Debe de ser un nivel no demasiado dificil, para que la aplicacion sea asequible a cualquier tipo de publico.	Completado	
	Agrada o no agrada	Subjetiva	Lo deseado es que sea una aplicacion agradable para el usuario. Aun así, cabe destacar que no se busca ser la aplicacion mas bonita del mercado, si no eficiente.	Completado	
Facilitat d'aprenentatge	Temps usat per acabar una tasca la primera vegada	Objetiva	El tiempo no debe ser superior a 15 segundos del tiempo asignado.	Completado	

Conclusions Conclusions dels resultats de la prova amb els errors detectats, les solucions plantejades i el treball futur

Como conclusión final en el test de usabilidad a un usuario con un nivel avanzado, la aplicación es bastante intuitiva y todas las funcionalidades necesarias se han realizado en el tiempo esperado, incluso antes. Para cualquier usuario avanzado le resultará facil comprender el funcionamiento de la aplicación.

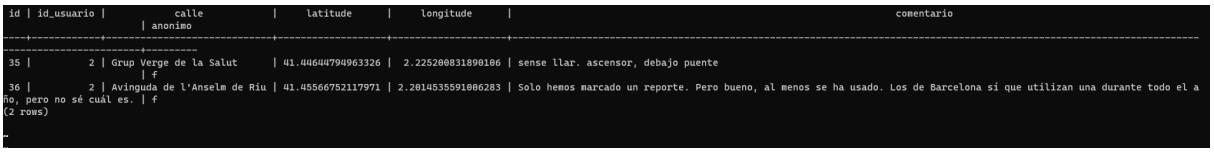
Lanzamiento

En referencia al lanzamiento de la aplicación, como explicaremos con más detalle posteriormente, vamos a subirla a la Play Store. No obstante, nuestra aplicación puede servir de base para muchas otras, ya que a modo resumido, se basa en marcar unos puntos en el mapa y realizar X acciones con estos puntos.

De hecho, tiene tanta adaptabilidad que en un proyecto del ayuntamiento Badalona ya se ha usado esta misma aplicación, tan solo unos días antes de realizar la entrega del proyecto.

El proyecto del ayuntamiento de Badalona consiste en buscar personas sin techo que viven en la calle, esto lo hacen voluntarios durante toda una noche, y deben anotar en un folio las calles y coordenadas donde se encuentran estas personas. Nuestro profesor, Daniel Martinez ha sido voluntario en este proyecto y se le ocurrió presentar nuestra aplicación al ayuntamiento que lo organiza. En el Ayuntamiento de Badalona, aceptaron usar nuestra aplicación para marcar los lugares donde se encontraban estas personas para posteriormente poder tener una lista fácil sobre ellos, así que creamos una base de datos para que no interfiriera con los demás datos del desarrollo y les facilitamos el “apk” de la aplicación.

Finalmente, la noche en la que se llevó a cabo la búsqueda, se usó la aplicación y podemos ver alguno de los comentarios que se hicieron en la tabla de “reportes”.



```
 id | id_usuario |           calle            |     latitude      |     longitude      |                                                                                                comentario                                                                                                | anonimo
----+------------+----------------------------+-------------------+--------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+---------
 35 |          2 | Grup Verge de la Salut     | 41.446447949633326 |  2.225200831890106 | sense llar. ascensor, debajo puente                                                                                                                                                      | f
 36 |          2 | Avinguda de l'Anselm de Riu | 41.455667521179771 | 2.2014535591086283 | Solo hemos marcado un reporte. Pero bueno, al menos se ha usado. Los de Barcelona sí que utilizan una durante todo el año, pero no sé cuál es. | f
(2 rows)
```

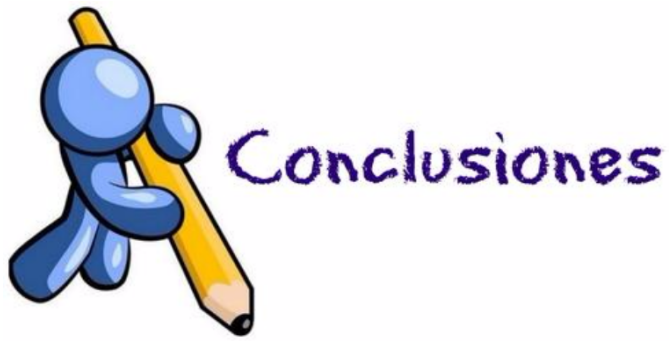
Play Store

La aplicación va a ser lanzada a la Play Store, ya que tenemos una licencia y la app puede ser bastante visible en este centro de descargas.

Debido a los plazos que ofrece Google para revisar la aplicación y finalmente publicarla en la Play Store, la aplicación todavía no estará disponible en esta plataforma, en la fecha de entrega de este documento, no obstante tenemos esperanzas de que para el día de la presentación si lo esté.

No obstante, una vez la aplicación sea aceptada por Google y esté subida completamente, añadiremos el enlace en la página web del proyecto: securemaps.es.

Conclusiones



Al empezar el proyecto, nos pusimos el reto de realizar una aplicación profesional, con un servidor administrado totalmente por nosotros, y a la vez crear una API, algo que nunca habíamos hecho, con un lenguaje totalmente nuevo para nosotros (JavaScript).

Conforme avanzaba el proyecto, hemos ido viendo los resultados del esfuerzo y la dedicación que le hemos dedicado, y finalmente como conclusión general, podemos afirmar que todas las horas de trabajo y horas de búsqueda de información han merecido la pena, ya que consideramos que nos ha quedado una aplicación lo más profesional posible.

Aún así, la aplicación se puede ampliar de forma infinita, añadiendo nuevas funcionalidades e incluso mejorando las actuales, no obstante aún con poco tiempo de proyecto (2 meses) consideramos que el resultado final ha sido muy bueno y hemos realizado un buen trabajo.

Por último, cabe destacar la buena gestión del trabajo que hemos realizado entre los dos integrantes del proyecto, trabajar de forma simultánea y continua nos ha permitido dividirnos las tareas y poder realizarlas de forma rápida y óptima.

Bibliografía

Consulta de información sobre zonas peligrosas.

- https://blog.prosegur.es/ciudades-mas-peligrosas-de-espana/
- https://www.diaridebadalona.com/noticia/badalona-es-la-sisena-ciutat-del-territori-espanyol-amb-mes-probabilitat-de-patir-un-robatori-a-la-llar/?fbclid=IwAR3xf6ZCtrrn-cy6PA_y8Qc74TK5y2lhcFCsVWjQO1rQYVEfiz48e-pe4p8

Consulta de leyes y normas.

- https://hackaboss.com/blog/salario-programador-espana-2018-2019/

Consulta sobre el material y precios.

- https://www.tiendatec.es/raspberry-pi/gama-raspberry-pi/1099-raspberry-pi-4-modelo-b-2gb-765756931175.html?src=raspberrypi
- https://www.pccomponentes.com/asus-s300ma-510400005d-intel-core-i5-10400-16gb-512gb-ssd

Consulta sobre los sueldos.

- https://es.indeed.com/career/analista-programador/salaries
- https://hackaboss.com/blog/salario-programador-espana-2018-2019/

Consulta sobre la administración del servidor

- https://elpuig.xeill.net/Members/vcarceler/articulos/contenedores-con-lxd-lxc/index_html
- https://ubuntu.com/tutorials/how-to-install-ubuntu-on-your-raspberry-pi#1-overview
- https://phoenixnap.com/kb/how-to-install-postgresql-on-ubuntu
- https://certbot.eff.org/

Consulta sobre código de programación

- https://stackoverflow.com/
- https://developer.android.com/
- https://github.com/