Universidad Autónoma de Madrid
Escuela Politécnica Superior

Proyecto fin de máster

PLATAFORMA DE RECOMENDACIÓN DE ESPACIOS PÚBLICOS Y PRIVADOS BASADA EN RATINGS

Máster en Ingeniería Informática

Noemi Escudero del Olmo
Febrero 2018

PLATAFORMA DE RECOMENDACIÓN DE ESPACIOS PÚBLICOS Y PRIVADOS BASADA EN RATINGS

AUTORA: Noemi Escudero del Olmo
TUTORA: Rosa Carro Salas

Dpto. de Ingeniera Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Febrero 2018

Resumen

Resumen

Hoy en día lo normal ya no es contratar un hotel sin antes mirar sus valoraciones, o probar un restaurante sin haber recibido una recomendación o haber leido previamente opiniones acerca del servicio. Cada vez se tienen más en cuenta los comentarios y las valoraciones que se leen a través de Internet en las redes sociales como Twitter o webs más especializadas como Booking, y no necesariamente dichas valoraciones tienen que haber sido escritas por expertos culinarios o críticos importantes, sino por gente normal que va a un lugar, experimenta, y luego recomienda o no su propia experiencia. Cada día estos comentarios en la red hacen más mella que la propia publicidad que pueda contratar la empresa, convirtiéndose así en un boca a boca peligroso o muy favorecedor que no tiene fronteras.

Además, en los últimos años han surgido multitud de empresas que han sabido aprovechar los datos generados por los usuarios para conseguir beneficio obteniendo estadísticas o vendiéndolas a terceros. Los usuarios no solo generan datos de utilidad cada vez que comentan en una web acerca de un restaurante, sino que también cada vez que se conectan, desde qué lugar se conectan, su rango de edad y su género también son factores a tener en cuenta.

Con estos dos temas como inspiración, se ha ideado, diseñado y construido una plataforma dividida en un framework basado en un servicio REST, una aplicación Android y un pequeño portal web.

Por un lado, el framework pretende proporcionar todas las herramientas necesarias para desplegar de forma rápida cualquier aplicación (web, smartphone, escritorio) que necesite un backend genérico de cara a ofrecer un servicio de red social de valoración de cualquier concepto, incluyendo la recogida de datos de los usuarios.

Por otro lado, la aplicación Android, que utiliza dicho framework, se centra en tomar la ubicación del usuario y mostrar en un mapa los lugares de interés que hay alrededor suyo. El usuario puede visitar el perfil de cada lugar para ver las valoraciones de sus características personalizadas y los comentarios dejados por otros usuarios de la aplicación.

Y por último, la web, muestra alguna de las estadísticas que se podrían recoger y visualizar en gráficas acerca de los usuarios, sus votos y sus comentarios.

Palabras clave

Valoraciones, opiniones, comentarios, buscador, ubicación, cercanía, Smartphone, Android, Python, backend, estadísticas, Django.

Abstract

Abstract

Nowadays, it is no longer normal to hire a hotel without first looking at its ratings, or to try a restaurant without having received a recommendation or having previously read opinions about the service. The comments and ratings that are read through the Internet on social networks such as Twitter or more specialized websites such as Booking are increasingly taken into account, and these assessments do not necessarily have to be written by culinary experts or important critics. They are indeed written by people who go to a place, experience it, and then decide to share their own experience.

These comments are usually affecting sales more than advertising, becoming a very important issue to take into account by companies all around the world.

In recent years, many companies have been able to take advantage of the data generated by users to get benefit by obtaining statistics and selling them to third party companies. Users not only generate useful data every time they comment on a website, but also every time they connect, offering data as the place they are connecting from, their age or gender which are really useful bits of information to classify customers.

With these two ideas as inspiration, a platform has been designed and built. It is divided into a framework based on a REST service, and in an Android application in addition to a small web portal.

On the one hand, the framework aims to provide all the necessary tools to quickly deploy any application (web, smartphone, desktop) that needs a generic backend in order to offer a social network service for the valuation of any concept, including the collection of user data.

On the other hand, the Android application, which uses this framework, focuses on taking the user's location and displaying the places of interest around them on a map. The user can visit the profile of each place to see the ratings of their features and the comments left by other users of the application.

And finally, the web portal, shows some of the statistics that could be collected and display them formatted in graphs containing data about users, their votes and their comments.

Keywords

Ratings, reviews, comments, search, location, proximity, Smartphone, Android, Python, backend, statistics, Django.

Agradecimientos

Agradezco todo el apoyo proporcionado por Guille durante estos meses de intenso trabajo, sobretodo en esos días de agobio, stress y desesperación.

Índice general

Índice de figuras

1 Introducción

Actualmente vivimos en un mundo donde las valoraciones de la comunidad sobre algo, son más importantes incluso que aquello que se valora. La publicidad es muy importante, y puede hacer que algo insignificante sea conocido y aclamado, y sin embargo, algo potencialmente revolucionario pero sin la publicidad necesaria, puede nunca llegar a ser conocido.

Los avances en la tecnología han provocado un salto en la rapidez con la que se mueve la información. La publicidad, sumada a la crítica social, hace de Internet la vía por la cual la gente se expresa mejor y más libremente.

Además, el hecho de que Internet aparentemente respalde la intimidad del autor detrás de la pantalla, hace que la gente se atreva a expresar su opinión con más voracidad y sin pensar su alcance.

Esto podría dañar seriamente la reputación tanto de personas como de instituciones o empresas. Un buen ejemplo de ello, son las empresas de restauración o los hoteles. No hay más que ver redes sociales como Twitter, o webs de búsqueda de restaurantes, donde los usuarios se vuelven auténticos críticos de su experiencia, y no temen a ninguna reprensión por decir lo que piensan, sobre todo cuando se critica algún aspecto negativo.

También cabe destacar que las empresas responsables de las plataformas sociales dan las máximas facilidades para obtener y estudiar los datos acerca de los comentarios de los usuarios, lo que transforma a estas plataformas y a Internet en general en una enorme plataforma de Marketing.

Partiendo de esta idea, se propone realizar un proyecto que utilice la opinión de la comunidad para ofrecer un servicio gratuitamente, a la vez que se obtienen datos que puedan ser utilizados para campañas de marketing y Business Inteligence. GoTo es una plataforma de recomendación de lugares de interés basada en la opinión de los usuarios y contenida en una aplicación móvil. A través de dicha aplicación, los usuarios podrán dar de alta lugares, así como puntuarlos conforme a una serie de criterios, ofreciendo a otros usuarios una forma fácil de localizar lugares que les puedan ser de interés a partir de las opiniones de la comunidad.

En el proceso, se guardarán las actividades de los usuarios con el objetivo de ofrecer estadísticas a los administradores de los lugares dados de alta y a empresas de terceros para servir de apoyo a estudios estadísticos orientados al Business Intelligence.

Más allá del caso práctico, la parte técnica del proyecto se presenta como un framework en forma de servicio, llamado Starppy, que almacena información y características sobre diversas entidades genéricas. Es capaz de almacenar las opiniones de la comunidad sobre las entidades y ordenarlas en forma de estadísticas utilizables en campañas de Business Inteligence más allá de la naturaleza de las entidades y que podría utilizarse para la programación de otras aplicaciones o portales web que necesitasen de esta funcionalidad.

1.1. Motivación

Como se ha comentado en el apartado anterior, hoy en día no se hace nada antes sin tomar una referencia u opinión. Incluso la opinión de alguien que se desconoce, es más importante que la publicidad misma de un sitio. Por ello, aportar a la comunidad de programadores una herramienta de gestión de bases de datos para que pueda realizar una aplicación de Smartphone acerca de ratings y valoraciones, es algo que actualmente está ya arraigado y que no se prevé que quede obsoleta a corto plazo.

Por eso, la motivación principal para realizar este trabajo es crear una aplicación que pudiera llegar a ser de utilidad real para los usuarios por un lado y de interés comercial a empresas por otro. Los usuarios obtendrían una forma sencilla de encontrar lugares de interés amparados por la comunidad sin ningún coste y los administradores de los lugares obtendrían una forma de publicitarse y de recibir feedback de la comunidad y estadísticas útiles. También empresas o instituciones públicas podrían utilizar los datos obtenidos para estudios estadísticos.

Además de una motivación personal para aprovechar el proyecto para aprender tecnologías nuevas como las que se van a presentar a continuación: Django REST Framework, Android Studio, etc.

1.2. Objetivos

Para la realización de la plataforma, se propone una arquitectura basada en servicios, donde uno o varios servidores ofrecen un servicio REST al que se conectan los terminales móviles a través de una app nativa de Android. Para ello se presentan una serie de objetivos:

- **Creación de un paradigma de rating genérico basado en características**, ya que actualmente se encuentran en el mercado webs bastante concretas (como Booking para hostelería) que no abarcan más allá de las fronteras a las que están dedicadas.

  Con este proyecto se pretende definir una serie de características base genéricas y cuantitativas que puedan personalizarse y aplicarse a todo tipo de lugares, públicos o privados y que sirvan a los usuarios de la aplicación como medio para obtener opiniones y para transmitir las suyas propias a la comunidad.

  - **Diseño e implementación de un servicio web basado en REST** que de persistencia a la plataforma. Este servicio estará también unido a una base de datos que mantendrá la información del sistema. Para esta parte, se propone utilizar el framework Django de Python acompañado de Django REST Framework como tecnologías principales para proveer de este servicio. En este servicio, también se deben tener en cuenta las herramientas necesarias para desplegar y mantener el servicio en un servidor Linux. Este servicio recibirá el nombre de **Starppy**.

  - **Diseño e implementación de una aplicación nativa de Android** que consuma el servicio REST antes mencionado y provea a los clientes de la funcionalidad de la plataforma. Para esta parte se ha escogido utilizar Android Studio y el lenguaje JAVA. Dicha aplicación recibirá el nombre de **GoTo**.

Como objetivos personales, se busca la investigación y mejora de las capacidades tanto en el diseño de la arquitectura de este tipo de aplicaciones como en la implementación utilizando las herramientas antes mencionadas:

- **Aprender Django Rest Framework y las tecnologías asociadas:** Aunque se tienen previos conocimientos sobre Python, tanto la librería Django como Django REST Framework son algo distintas de utilizar y se requiere un aprendizaje. El objetivo es entender el flujo de datos dentro de una aplicación Django así como todo lo necesario para su mantenimiento y desplegado en un servidor Linux.
- **Aprender Android Studio:** Debido a que durante el Grado y el Máster no se ha cursado ninguna asignatura donde se hayan estudiado estas tecnologías, éste es un buen momento para afianzar nuevos conocimientos relevantes muy demandados por las empresas hoy en día.
- **Aprender LaTeX:** De la misma manera, durante el grado y el máster, no se ha tenido oportunidad de tener contacto con LaTeX. Con el aprendizaje del lenguaje de este compositor de textos, no solo se se busca una mejor calidad tipográfica para este informe, sino también obtener un nuevo conocimiento que puede ser de utilidad en la escritura de otros informes en el futuro, ya que LaTeX es muy utilizado, sobre todo en el ámbito universitario y de investigación.

1.3. Público

Por lo que ya se puede deducir según lo escrito anteriormente, el público al que está dirigido este proyecto se divide en principalmente en tres grupos:

- **Usuarios comunes:** El proyecto está dirigido a gente que en su tiempo libre le gusta visitar lugares, y no sabe a dónde ir. Aquí cabe todo el mundo que se descargue la aplicación y la use. No hay restricción de edad, sexo o condición. Todo el mundo que tenga un smartphone Android puede participar y aportar.
- **Empresas TI:** También está dirigido a todas las empresas que se dedican a hacer estudios estadísticos sobre el impacto social de las opiniones de la gente, o a empresas que ofrezcan estudios estadísticos a los dueños de locales, ya que podrían utilizar GoTo como medio para dicho fin.
- **Otros programadores y empresas:** El hecho de que el diseño del servidor REST se haga en forma de framework genérico, permitirá que otros programadores y empresas que quieran desarrollar otras aplicaciones similares, puedan utilizar Starppy como base para sus aplicaciones o sitios web.

2

Estado del arte

A continuación se expone el estado del arte en lo relacionado al proyecto que se va a desarrollar. Se van a tratar dos temas principales, por un lado, se va a comentar la importancia de los smartphones y de las aplicaciones móviles en la sociedad de hoy en día, cómo se ha llegado a esta situación y cómo se ha creado un tejido empresarial alrededor de estas. Por otro lado se va a tratar la evolución de los servicios en red que hacen posible que este tipo de aplicaciones funcionen, así como las tendencias en materia de tecnología a la hora de desarrollar estos servicios.

2.1. Smartphones y aplicaciones

2.1.1. Revolución de los Smartphones

Actualmente los smartphones son parte integral de la sociedad, existen aplicaciones para prácticamente cualquier cosa y las personas las utilizamos continuamente, desde para saber cuándo va a venir el autobús, hasta para sacar dinero o pagar en un establecimiento.

Todo este panorama digital se venía gestando desde hace dos décadas y ha evolucionado hasta el día de hoy. Al inicio de la década de los 90, hubo una revolución con la llegada de la web, se abrió un nuevo abanico de posibilidades de comunicación que desembocó en una verdadera revolución digital.

De la misma manera que eso supuso un antes y un después para la sociedad, el 29 de junio de 2007 sucedió de nuevo, Apple combinaba un reproductor de música, un asistente personal y un teléfono en un único dispositivo con pantalla táctil y con pequeñas aplicaciones, el iPhone, sembrando así el concepto de lo que hoy conocemos como Smartphone [33]. Apple había triunfado en su afán de transformar el teléfono móvil en una herramienta de comunicación más completa, o lo que hoy en día es: una herramienta de trabajo, ocio y comunicación a la vez.

Paralelamente, otras empresas estaban trabajando también en sistemas software para dispositivos móviles, por lo que fue cuestión de tiempo la aparición de nuevas soluciones. El producto estrella de estas investigaciones es la aparición de Android el 5 de noviembre de 2007 [4] por parte de Android Inc. Aunque después sería desarrollada por el propio Google [6].

2.1.2. Las aplicaciones móviles

Las aplicaciones móviles conllevan numerosas ventajas frente a otras tecnologías, tener un dispositivo que se puede llevar en el bolsillo, sin depender de la ubicación donde te encuentres da bastante comodidad al consumidor y permite a los desarrolladores implementar, por ejemplo, sistemas de notificaciones mucho más eficientes. Un gran ejemplo son las aplicaciones de mensajería instantánea (como whatsapp, telegram o line), que ya se han vuelto más comunes que las propias llamadas de teléfono [26]. También el hecho de que son dispositivos compuestos, da la posibilidad a los desarrolladores de utilizar la cámara, el micrófono, los acelerómetros, la brújula o el GPS para extender las capacidades de sus programas, e implementar navegadores, lectores de códigos de barras o incluso juegos de realidad aumentada como PokemonGo o Ingress [12].

Los smartphones y en concreto las aplicaciones están tan integrados en la sociedad que incluso están afectando de forma severa en ella. Un caso de esto es, por ejemplo, que se están dando casos de adicciones severas al Smartphone, sobre todo entre estudiantes [29] , o casos como el de los servicios de Airbnb, Uber o Cabify que han tenido problemas legales [13]. En algunos países como España, estos últimos dedicados al alquiler de vehículos han incluso producido huelgas entre el sector del taxi [20].

Por supuesto no todo es malo, también hay aplicaciones que afectan a la sociedad para bien, como Lazzus [19], un asistente de movilidad para personas ciegas, o Babbel [32], una aplicación orientada al aprendizaje de idiomas. Y como éstas, infinidad de aplicaciones que hacen la vida de las personas más sencilla.

2.1.3. Metadatos y Business Intelligence

Estas aplicaciones y servicios no solo ofrecen un beneficio al consumidor, sino también a las empresas que las controlan, puesto que su uso genera metadatos. En el caso de las aplicaciones móviles, la generación de metadatos es mayor que en otro tipo de soportes, puesto que también pueden aportar datos sobre la posición geográfica del usuario, las horas de uso, las conexiones inalámbricas a las que el usuario se conecta u otros tipos de datos que por ejemplo una aplicación web no puede aportar de manera tan invisible para el usuario.

A partir de esto, ha surgido un tejido empresarial que su intención es lucrarse de los datos que los usuarios dejan por internet y especialmente a través de sus móviles. Estas son las consultoras, empresas que intentan ayudar a otras a entender a su cartera de clientes y mejorar su estrategia de marketing y que realizan un análisis de las bases de datos.

También hay que tener en cuenta que la recolección de los datos de usuarios se lleva utilizando desde el principio de las aplicaciones móviles y que no siempre es para fines lucrativos, sino que puede reportar al usuario datos útiles, como hace Google Maps al recoger las ubicaciones de todos los dispositivos que tienen instalado dicha aplicación y de esa manera saber dónde se ha formado un atasco.

El usuario debe tener presente que el hecho de que no pague por una aplicación no significa que sea gratis [28]. Muchas aplicaciones han adoptado esta forma de financiación, que implica ofrecer un servicio aparentemente gratis al usuario, que le anima a descargarse la aplicación, mientras se recogen datos acerca de él y de esta forma la aplicación se mantiene rentable. Otro ejemplo distinto, serían los juegos freemium, que se basan en poder ser jugados de manera gratuita (free to play) pero incluyen micropagos que hacen más sencillo el juego a los jugadores que decidan pagar y así sostener económicamente el juego [40].

A parte de todas estas empresas que utilizan el Business Inteligence para crecer y saber más sobre el tipo de gustos tienen sus clientes para poder ofrecerles productos más personalizados, existen otro tipo de empresas que se dedican a recoger estadísticas, por ejemplo, a través de encuestas, las cuales luego venden a terceros, y ofrecen a los que responden sus encuestas acumular puntos y canjearlos por premios [34]. Todo esto plantea un problema ético, ya que hay empresas que literalmente trafican con los datos sin que el usuario se dé cuenta realmente de lo que están haciendo con ellos, ya que muchas veces se acepta los términos y condiciones de uso sin leerlas detenidamente.

2.2. Background tecnológico

Para que la situación tecnológica y social actual en relación a los smartphones y las aplicaciones pueda haberse dado, ha sido necesario que exista un contexto no solo social sino también tecnológico.

2.2.1. Sistemas operativos para dispositivos móviles

A finales de los años 80 y principios de los 90, junto con la aparición de los primeros teléfonos móviles comerciales, apareció la necesidad de desarrollar software para ellos. En esta época, los teléfonos móviles eran dispositivos complejos y con hardware diseñado a medida. En estos sistemas embebidos no era posible el desarrollo de aplicaciones y de nueva funcionalidad, puesto que el software se limitaba a controlar el funcionamiento del dispositivo.

Conforme el hardware de estos dispositivos se iba haciendo más potente, nuevas necesidades aparecieron, los usuarios cada vez demandaban que sus dispositivos móviles cubrieran más funciones.

En el año 1996, hicieron su aparición los dispositivos Palm que funcionaban a través del primer SO móvil moderno, el PalmOS [41]. Este sistema estaba escrito en C++ y permitía la programación de aplicaciones por parte de terceros a parte de las propias aplicaciones con las que contaba el dispositivo, esto permitió la integración del dispositivo con sistemas industriales o de gestión, algo de gran importancia en sectores como la paquetería.

De la misma manera, en 1998 Symbian Ltd. desarrolló un sistema similar, el Symbian OS que fue utilizado en multitud de dispositivos como Nokia, Samsung, Motorola o Sony Ericcson.

Symbian [42] supuso un gran avance, puesto que como la gran mayoría de los fabricantes importantes de dispositivos móviles lo usaban, el desarrollo de aplicaciones se generalizó al poderse usar una aplicación en varios dispositivos.

Sin embargo, no fue hasta la llegada de los 3 grandes SO móviles, a saber, Android, iOS y Windows Phone (ya descontinuado) [1] que el desarrollo de aplicaciones se ha generalizado. Los fabricantes proveen a los programadores de kits de desarrollo muy completos y documentados que hacen más fácil el desarrollo así como de medios de distribución como Apple Store (iOS) o Play Store (Android) que hacen que cualquiera pueda publicar una aplicación para que cualquiera la utilice. Este avance técnico ha propiciado en gran medida el estado social que se comentaba en el apartado anterior.

Pero el desarrollo de estos sistemas no se detiene aquí, no se limitan únicamente a dispositivos como teléfonos o tablets, si no que recientemente se ha extendido a relojes inteligentes, wearables y otros dispositivos portátiles.

2.2.2. Redes móviles

De la misma manera que los SOs para dispositivos móviles han sido muy importantes para formar el panorama social que hoy tenemos en torno al Smartphone y a las aplicaciones, las redes móviles también han sido de vital importancia.

El grueso de las aplicaciones instaladas en los smartphones que se usan a diario utilizan Internet de una manera u otra, por tanto la aparición de la posibilidad de utilizar la red de telefonía no solo para hacer llamadas o enviar SMSs ha sido vital para el desarrollo del Smartphone.

Al inicio, la manera de enviar y recibir datos a través de las redes de telefonía inalámbricas era parecida al funcionamiento de los módems convencionales, se hacía una llamada al proveedor del servicio y se transmitía la información a través de la línea. Más tarde, con la aparición del GPRS [2] (comúnmente conocido como 2G) permitió la incorporación por primera vez del protocolo TCP/IP y hasta 54 mbits/s de ancho de banda. Estas condiciones propiciaron que empezaran a aparecer servicios web destinados a móviles y que las empresas telefónicas comenzaran a ofrecer tarifas que permitieran este tipo de conexiones.

Recientemente con la aparición de tecnologías de banda ancha como HSPA, 3G y 4G [22] se ha mejorado notablemente la velocidad y la calidad de las transmisiones, permitiendo incluso servicios de streaming de vídeo y música en alta definición a través de este tipo de redes móviles, como Netflix, Amazon video o Spotify.

2.2.3. Servicios y nube

Otro factor importante en el desarrollo de las aplicaciones móviles tal y como las conocemos hoy es la proliferación de los servicios web, puesto que una parte integral de cada una de las aplicaciones móviles que se desarrollan hoy en día es el componente remoto que la dota de persistencia y funcionalidad.

En un principio, los servicios que se proveían a partir de la red eran únicos, un cliente y un servidor se comunicaban para ofrecer el servicio. Sin embargo, con la proliferación de la web, la aparición de diferentes dispositivos que permiten navegar por internet hizo que los servicios empezaran a estructurarse en frontend y backend.

Uno o varios servidores de backend proveen de persistencia y funcionalidad a diferentes clientes (frontend) que se limitan a mostrar la información al usuario. El backend es independiente de la plataforma, y a través de un protocolo concreto, se comunica con el frontend, que sí es específico de la plataforma. Esta arquitectura permite por ejemplo, reutilizar la lógica del servicio agrupándola en un backend común y reduciendo la carga de procesamiento de los clientes, de esta forma, estos clientes pueden funcionar en dispositivos poco potentes.

Este tipo de arquitectura se ha visto impulsada por tecnologías como la arquitectura REST [24], que es un estilo de diseño de aplicaciones distribuidas en la web que se sirve de HTTP para establecer una comunicación sin estados entre un cliente y un servidor. Este tipo de protocolos son mucho más sencillos de implementar que los antiguos protocolos RPC como CORBA [45] o los basados en XML como SOAP [46].

Sin embargo, en la actualidad, las arquitecturas de servicios en la red están empezando a evolucionar hacia los microservicios [3]. Este paradigma consiste en construir una aplicación como un conjunto de pequeños servicios, que son independientes y están permanentemente comunicados a través de protocolos ligeros. Cada servicio se encarga de ofrecer una funcionalidad completa y todos juntos forman una misma aplicación.

Este tipo de arquitectura ofrece muchas ventajas, sobre todo en cuanto a reutilización de componentes, desarrollo en paralelo de cada servicio, la tolerancia a fallos que le da el hecho de que cada parte de la aplicación esté distribuida, autorización y autenticación a través de servicios de terceros o la posibilidad de subcontratar servicios para ahorrar costes.

Alrededor de estas arquitecturas de microservicios están apareciendo distintas tecnologías que hacen su desarrollo más sencillo, como Docker y Vagrant [7] que a través de virtualización hacen el desplegado de estos microservicios una tarea más sencilla, Puppet y Chef [44] que automatizan las tareas de desplegado o ElasticSearch y Kibana [37] que automatizan y centralizan el mantenimiento.

3
Análisis

En este apartado se va a exponer el proceso de análisis del sistema que se quiere construir, partiendo de los objetivos a cubrir, detallando las diferentes decisiones de diseño y qué circunstancias han sido analizadas para llegar a dichas conclusiones. Este apartado está dividido en dos partes diferenciadas:

- **Análisis de funcionalidad:** En este apartado se detallan las circunstancias que han llevado a las diferentes decisiones de diseño que se han tomado en cuanto a la funcionalidad que se quiere ofrecer con este servicio a partir de un estudio de mercado de las diferentes aplicaciones, empresas y servicios parecidos al sistema que se quiere construir.
- **Análisis de las diferentes tecnologías** disponibles para el desarrollo del sistema, tanto para la aplicación móvil como para el framework que se desarrollará para sustentarla.

3.1. Análisis de funcionalidad

Para poder llegar a la solución propuesta en los objetivos del proyecto, se ha partido de un análisis de las empresas que actualmente se dedican a ofrecer servicios parecidos.

Estas empresas ofrecen servicios de clasificación y búsqueda similares al que se pretende implementar, pero referidos a temas más específicos. En concreto: buscadores de restauración, hostelería, ocio, actividades y ofertas.

Este tipo de servicios pueden aportar mucha información al proyecto, porque, entre otras cosas, en ellos se da mucha importancia a la localización, ya que los usuarios suelen buscar establecimientos de restauración y hostelería cercanos a donde se dirigen o a donde se encuentran. También son un buen ejemplo de cómo presentar las opiniones de la comunidad a los usuarios que buscan un establecimiento.

Con este análisis se pretende analizar las ventajas e inconvenientes de cada uno de estos servicios con el fin de ofrecer una buena solución a los objetivos propuestos.

3.1.1. Buscadores de restauración y hostelería

TripAdvisor

Tripadvisor ofrece un buscador de hoteles, alquiler vacacional, restaurantes, además de ofrecer rutas, atracciones, actividades y visitas guiadas a través de su web y su aplicación móvil. Su interfaz destaca por las opiniones de los usuarios y el número de *estrellas* que tiene cada lugar. Además, en cada publicación de hotel o restaurante también añade un apartado de características y un mapa. Lo primero que hacer en TripAdvisor es marcar la ubicación para que optimice la búsqueda de hoteles, restaurantes o atracciones. Para los hoteles, posee un filtro para seleccionar el precio, los servicios, la distancia, el tipo de alojamiento, la puntuación mínima, la categoría y el estilo del hotel. Cuando se selecciona un hotel se puede ver en un resumen las puntuaciones (entre 1 y 5) y el mapa, las fotos y los restaurantes y atracciones que tiene cerca. TripAdvisor no permite pagar por la estancia de un hotel, sino que es socio de Booking.com, Expedia y Hoteles.com y son estos sitios quienes gestionan las reservas.

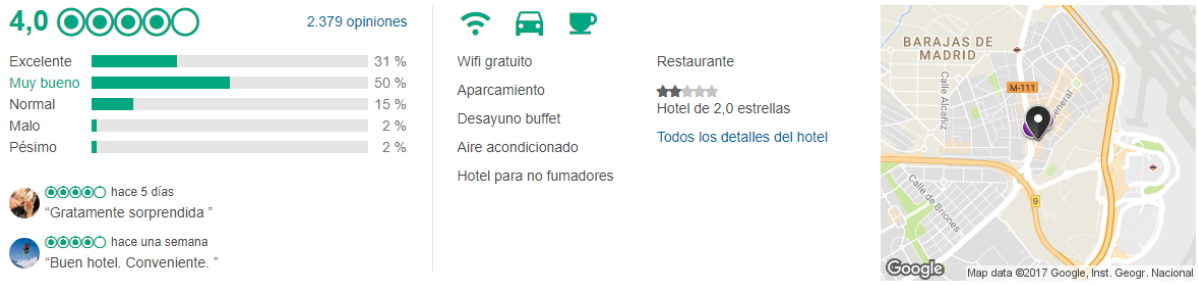


Figura 3.1: Valoraciones, características y mapa en TripAdvisor

En el apartado de restaurantes, hay un filtro para buscar por el tipo de establecimiento, y cuando se elige uno, a parte de las fotos y las opiniones, muestra unas puntuaciones que resaltan la comida, el servicio, la calidad/precio y la atmósfera. En el apartado de Qué Hacer se puede ver un filtro de atracciones que abarca: monumentos y puntos de interés, museos, compras, naturaleza y parques, spas y bienestar, actividades al aire libre, eventos, diversión y entretenimiento, visitas guiadas, transporte, comida y bebida, clases y talleres, y vida nocturna entre otros. También incluye estadísticas como el tiempo medio de la zona por meses del año.

Booking

Booking ofrece un servicio de búsqueda de alojamientos a través de su web y aplicación móvil. Se caracteriza por contener un buscador con un filtro compuesto de bastantes apartados. También muestra una puntuación del hotel y su ubicación. También detalla gran cantidad de características del alojamiento, un mapa, y guarda comentarios de los usuarios.

Las características del alojamiento que se pueden votar a parte de la puntuación global, son: limpieza, confort, instalaciones y servicios, personal, relación calidad-precio, wifi gratis y ubicación.

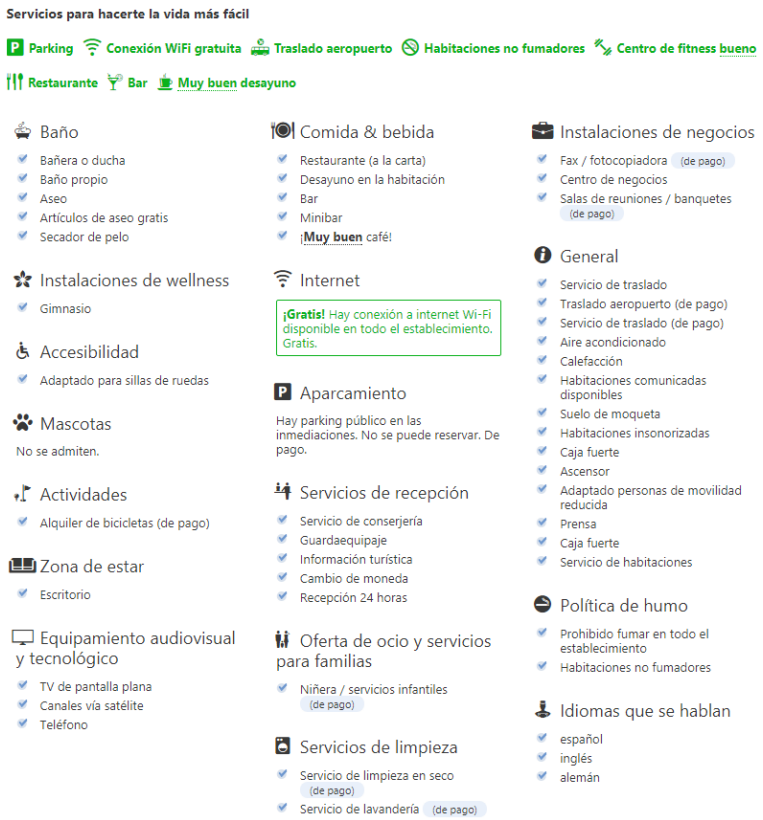


Figura 3.2: Características en Booking

En el apartado de comentarios, se puede filtrar los idiomas que se quieren leer. Se destacan los comentarios de los usuarios que son Gurús de la ciudad (han viajado 3 veces a esa ciudad y tienen comentarios útiles sobre los alojamientos). El sistema obtiene estadísticas automáticas de los hoteles para marcarlos como Favoritos para viajes en pareja o ideal para estancia de una noche. Dichas estadísticas las obtiene guardando las estancias de los usuarios y abstrayendo el perfil de la estancia. Para los propietarios del alojamiento permite incluir fotos y una descripción extensa del sitio.

También tiene un apartado de servicios más populares en los alrededores del alojamiento ordenados por distancia, divididos entre lugares de interés y aeropuertos.

Expedia

Es una empresa dedicada a encontrar viajes. Se puede acceder desde su web o su aplicación móvil a sus múltiples buscadores: de hoteles, de vuelos, alquiler de coches, ofertas, última hora y actividades.

En el apartado de hoteles, se puede incluir también el precio del vuelo y del alquiler de coche en las búsquedas. El filtro de hoteles se basa en puntuación del establecimiento, la zona, los servicios, el tipo de establecimiento y la cercanía a zonas populares El apartado de actividades se estructura en torno a una ubicación. Dentro de la actividad se puede ver la fecha, el precio, el mapa, la duración y aspectos destacados, pero no se pueden leer valoraciones de otros usuarios. El filtro de actividades se divide en recomendaciones, tours y actividades.

Figura 3.3: Características y comentarios en Expedia

Los hoteles muestran fotos, mapas, lugares de interés cercanos y dos tipos de valoraciones de usuarios: una media de la puntuación sobre 5, y un porcentaje del número de usuarios que lo recomendaría. Hay un apartado de opiniones, algo escondido, donde se puede ver las puntuaciones que han dejado los usuarios en las categorías de: limpieza de la habitación, servicio y personal, comodidad de la habitación y estado general del hotel. En los comentarios de los usuarios, otros usuarios pueden votar si les ha resultado útil o no.

Expedia también anima a que los establecimientos se publiquen en su web.

Kayak

Es un servicio de búsqueda de ofertas de hoteles, vuelos y coches de alquiler. La aplicación de móvil tiene una característica extra frente a la web, permite poner alertas cuando el precio de algún elemento haya bajado de un umbral.

El buscador de hoteles tiene la opción de añadir filtros tales como la estrellas, las valoraciones, el rango de precio, las características gratuitas, la ubicación, los servicios, el ambiente, el tipo de alojamiento y los proveedores (ya que Kayak es en realidad un metabuscador que obtiene datos de otras webs).

Cuando se selecciona un hotel aparece un desplegable donde se pueden ver los detalles, el mapa, una comparación de precios de distintas webs y unas puntuaciones separadas en categorías: ambiente, bar y bebidas, instalaciones, servicio, desayuno y ubicación. También tiene una nota media dependiendo del rasgo de la persona que lo ha valorado: una pareja, una familia, alguien que ha ido por negocios o un viajero solitario.

Cabe destacar un apartado de exploración, donde no hay que marcar un destino o una ubicación, simplemente te recomienda sitios a los que ir, según categorías.

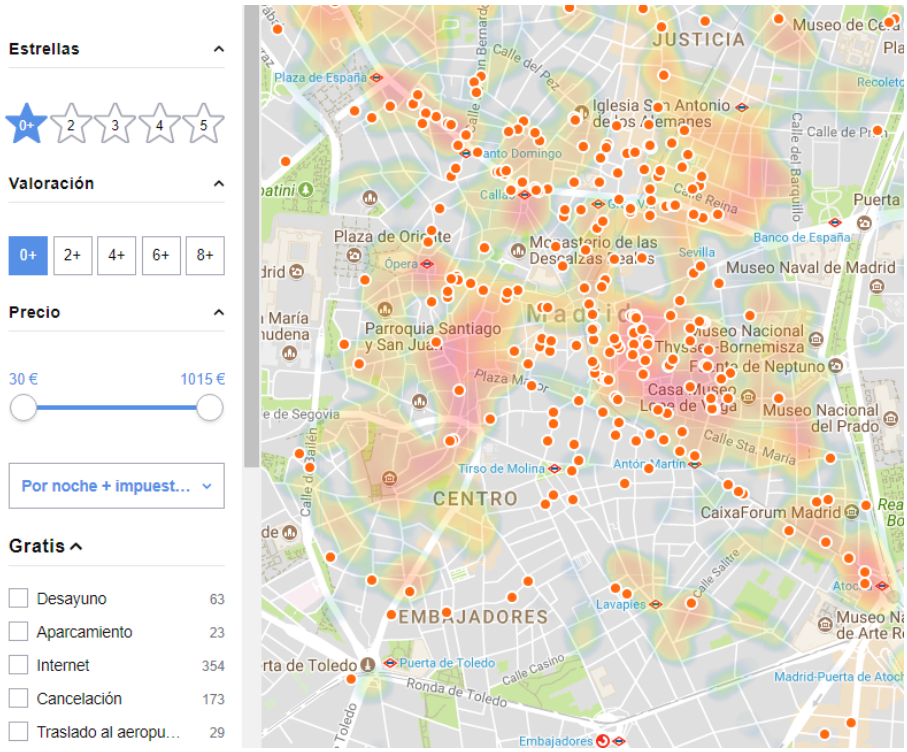


Figura 3.4: Mapa y filtros en Kayak

Trivago

Es una empresa dedicada únicamente a buscar hoteles a través de su web o aplicación móvil. El primer filtro para búsquedas es el de la ubicación, y luego presenta otros opcionales como la puntuación del hotel, las valoraciones de los huéspedes entre 0 y 10, la distancia respecto a lugares de interés o una dirección concreta. También presenta una gran variedad de filtros adicionales para destacar detalles del hotel.

Cuando se selecciona un hotel se pueden ver fotos, la información junto al mapa y un apartado de opiniones donde se pueden ver las puntuaciones dedicadas a varias de las características, tales como la ubicación, las habitaciones, el servicio, la limpieza, la calidad-precio, el confort, las instalaciones, el edificio, los desayunos y las comidas.

En los comentarios se da mucha importancia a la puntuación y a la fecha en la que se publicó el comentario. El filtro se divide en opiniones positivas, negativas y neutras.

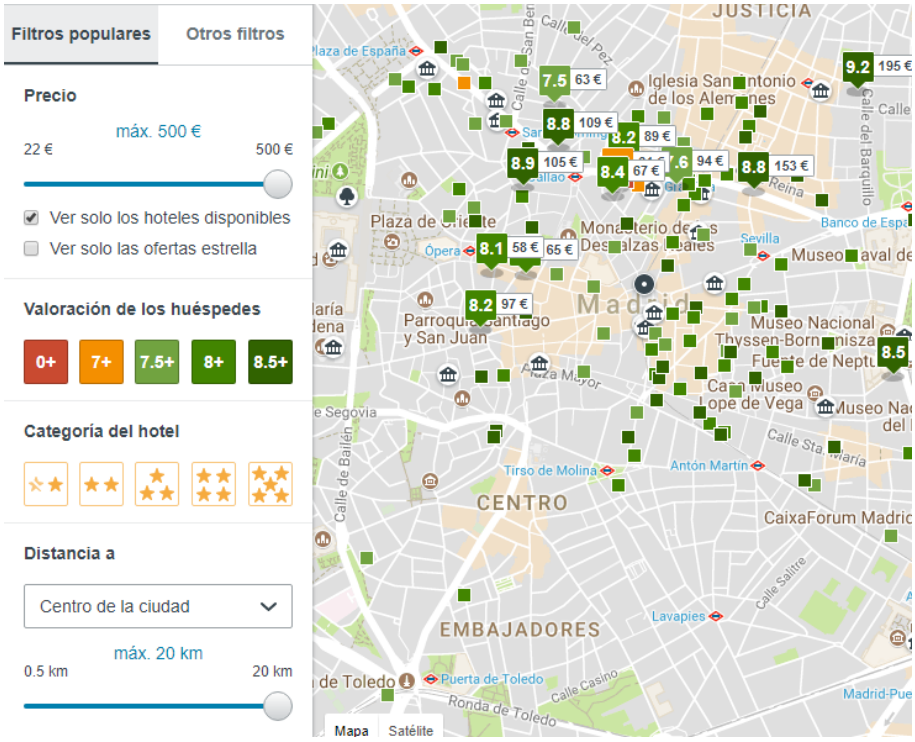


Figura 3.5: Mapa y filtros en Trivago

3.1.2. Buscadores de ocio

Fiestify

Es una aplicación móvil que recoge la oferta de ocio de la Comunidad de Madrid. En varios aspectos se parece al proyecto que se está desarrollando.

Por un lado, los usuarios comunes disponen de un buscador gratuito de ocio que además, tiene soporte para notificaciones para avisar de los eventos según el gusto específico de cada usuario. Por otro lado, está también dirigido a empresas para que se anuncien en la propia aplicación. Fiestify ofrece una serie de ventajas gratuitas a empresas a cambio de quedarse con una pequeña cuota por afiliación Premium y un porcentaje de las comisiones de cada venta que se haga a través de la aplicación. Pero anunciarse dentro de la aplicación es completamente gratis.

Los usuarios que no tengan claro qué hacer con su tiempo de ocio pueden consultar recomendaciones de planes en la aplicación, además de crear una lista con los planes que más le interesen para estar al tanto de cualquier novedad. Los filtros de los planes de ocio se basan en el tipo de plan, la fecha, la localización, si el evento es apto para niños y se puede ordenar por distintas categorías como la cercanía.

En la parte dedicada a las empresas, cabe destacar que ofrecen tanto estadísticas sobre la visualización que ha tenido un anuncio publicado por la propia empresa como estadísticas sobre la venta de entradas que ha habido a través de Fiestify.

La mayor ventaja que proponen para que las empresas usen su aplicación es que el usuario solo tiene que introducir su cuenta bancaria en un sitio (Fiestify) y ellos ya se encargan de realizar todas las compras de entradas a los eventos, que de otra forma, el usuario tendría que introducir sus datos de facturación en cada evento al que quisiera asistir.

Figura 3.6: Oferta a empresas de Fiestify

3.1.3. Buscadores de actividades

Planap

Es una web dedicada a la búsqueda de planes de fin de semana de multiaventura, spa, rafting, senderismo, paintball, barranquismo, etc. Los planes suelen contar con algún tipo de descuento económico, ya que ningún plan es gratuito. A parte de buscar planes, también se puede publicar un plan. Esto está dirigido sobre todo a empresas. Ellos se cuidan de eliminar los planes creados por empresas que hayan recibido malos comentarios o hayan tenido malas experiencias.

Figura 3.7: Ofertas de Planap

El filtro para los planes no es demasiado extenso. El propio usuario tiene que insertar palabras claves para buscar y añadir la provincia. El único filtro que proponen es según el público específico: dirigido a todos, a despedidas de soltero, a familias, a parejas, a empresas o a escolares.

Una vez seleccionado un plan, se puede ver con más detalle un mapa con su ubicación y una breve descripción. Sin embargo en ninguna parte se da importancia a las valoraciones de excursionistas pasados ni se muestra ninguna opinión sobre el plan.

3.1.4. Buscadores de ofertas

Atrápalo

Es una web dedicada a conseguir ofertas y descuentos de todo tipo. Los usuarios pueden buscar algo concreto o buscar ofertas en el apartado de Hallazgos.

En el buscador de actividades, hay un filtro previo, a parte de la ubicación y la fecha, donde se puede señalar el tipo de actividad que se busca: cultura, gastronomía, cursos, deportes y aventuras, bienestar, belleza o actividades infantiles. Cada uno de estos tipos, tiene a su vez subcategorías.

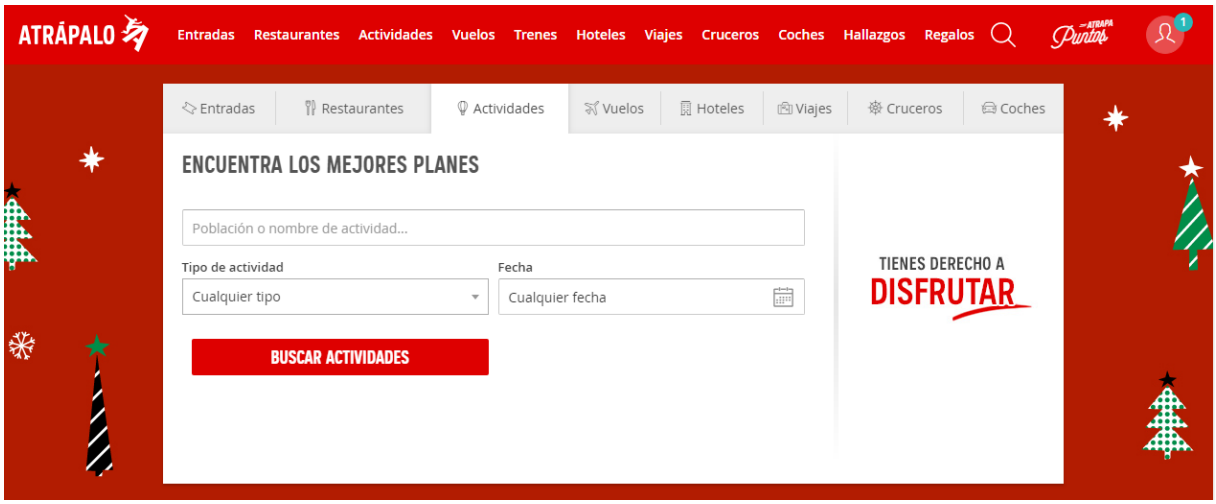


Figura 3.8: Buscador de Atrapalo

Dentro de una actividad lo primero que se observa son las fotos y la nota media, además de las fechas, la duración, el horario y la edad mínima. Después aparece una descripción y el mapa. Y debajo la compra de entradas del evento o actividad. Más abajo están las opiniones de los usuarios junto a su votación personal sobre distintos aspectos. En Atrápalo, las características a votar varían dependiendo sobre si es, por ejemplo, una guía a un museo, o un espectáculo. En éste último, los criterios a medir son la calidad del espectáculo, la calidad/precio, la puesta en escena/producción del evento, y la interpretación artística. Sin embargo, en una guía de museo, los criterios a puntuar son la calidad/precio, la calidad de la actividad y la atención del personal/guía. También hay una opción para ordenar los comentarios con lo que se consigue ver rápidamente el comentario con peor valoración, la mejor valoración, el más reciente y el de mayor relevancia.

Atrápalo también recoge las ofertas de restaurantes, hoteles, vuelos, cruceros y escapadas. Todas estos tipos de ofertas contienen su propio buscador personalizado y separado, con filtros de distintos tipos y categorías y demás rasgos. Los comentarios de los usuarios también tienen notas personalizadas dependiendo de si se trata un restaurante o un hotel.

3.2. Conclusiones del análisis de funcionalidad

Después de haber realizado el análisis de mercado, a continuación se detallan una serie de características principales que son compartidas por la mayoría de estas aplicaciones, y que se han tenido en cuenta a la hora del diseño de GoTo.

- **Localización:** Los buscadores están en su gran mayoría centrados en la localización. Los usuarios estructuran sus búsquedas independientemente de lo que busquen alrededor de su posición o del lugar donde desean ir, por tanto las pantallas principales de las aplicaciones y las búsquedas suelen sustentarse en un mapa así como la visualización y la interactividad con los resultados de las búsquedas.
- **Filtros de búsqueda:** Otro elemento principal en las aplicaciones estudiadas son los filtros. Entre todos ellos, el más importante es el filtro de distancia, aunque también hay filtros por categorías, puntuaciones u opiniones. Es importante que estos filtros estén también representados con elementos que sean intuitivos y faciliten al usuario su cambio de valor. Un ejemplo recurrente de esto último es la representación de puntuaciones mediante estrellas.
- **Opiniones, comentarios y contenido:** Uno de los pilares de estas aplicaciones son las opiniones y los comentarios de la comunidad, ya que son realmente lo que permite a los usuarios obtener la información que buscan.

  Es importante que las opiniones estén expresadas con elementos como notas, estrellas o likes, ya que esto ayuda al usuario a obtener la información que busca de un vistazo.

  De la misma manera, la forma de mostrar los comentarios debe ser clara, y deben estar ordenados de acuerdo a la importancia del comentario o de la persona que comenta.
- **Perfiles:** En muchas de estas aplicaciones, los perfiles de hoteles o restaurantes están mantenidos por las empresas o particulares responsables de ellos. Desde estos perfiles, los responsables pueden editarlos y visualizar estadísticas que tengan que ver con sus negocios.

  También es necesario para todas estas aplicaciones que haya contenido extra que ofrezca más información sobre los elementos que busca el usuario, como descripciones o datos de contacto. Entre este contenido destacan las fotografías, que suelen ser parte central de los perfiles buscados y un elemento muy importante para los usuarios.

3.3. Valor diferencial

A partir del estudio de mercado también se desprenden una serie de ideas que en general, no están cubiertas por aplicaciones que estén actualmente en funcionamiento, y que aportan un valor añadido al servicio que se está desarrollando en comparación con los servicios actuales:

- **Genericidad:** Como se ha visto en el apartado anterior, todas las empresas analizadas se dedican a ofrecer búsquedas sobre diferentes elementos, pero cada una de ellas se especializa en un campo concreto y son poco flexibles. GoTo pretende mejorar este aspecto:
  - **Genericidad de contenido:** En los servicios estudiados, el contenido que se muestra al usuario cuando busca elementos es siempre igual para cualquiera de ellos. Por ejemplo, en TripAdvisor, si se busca un restaurante, en todos ellos se encontrarán los mismos campos: fotos, descripciones, comentarios...

Con GoTo se busca que el administrador de un elemento tenga más flexibilidad, pudiendo añadir un número arbitrario de características que describan el elemento, dentro de unos tipos de características definidas.

  - **Framework:** En este trabajo no solo se pretende desarrollar una aplicación, sino también un framework que permita a otros programadores desarrollar aplicaciones similares. Esto hace que el diseño tenga que ser genérico e independiente de los elementos de búsqueda.

- **Flexibilidad de alta:** En las aplicaciones analizadas, los elementos de búsqueda son administrados por usuarios que han tenido que darse de alta y certificar que son, por ejemplo, los dueños de un restaurante u hotel. Con GoTo se pretende que cualquier usuario pueda dar de alta un lugar. De esta manera se facilita disponer de más información de la comunidad. Para evitar duplicidades y permitir que los administradores legales de los sitios no pierdan su derecho sobre los perfiles, se propone un sistema de cuentas verificadas.

3.4. Solución propuesta

A continuación se detalla qué características va a tener el proyecto para dar una solución a lo propuesto y aportar el valor añadido antes comentado. Por una parte, la aplicación contendrá:

- **Entidades:** Son aquellos elementos que se buscan en la aplicación. En el framework son planteadas de forma genérica, ya que debe ser reutilizable para cualquier otro tipo de aplicación. Se han diseñado con una serie de características a valorar y son creadas y gestionadas por usuarios. En concreto, en la aplicación desarrollada en este trabajo, pueden representar cualquier lugar o espacio, como una playa o un restaurante, un hotel, etc.

  Sin embargo, esto plantea el problema de que, a priori, cualquier persona puede dar de alta cualquier lugar, produciendo duplicidades o que los lugares privados dados de alta sean controlados por usuarios que no tienen ninguna vinculación con las propiedades a las que se hacen referencia.

  Para evitar este problema se propone un sistema de verificaciones de cuentas de usuario tal y como se mencionará más adelante.

  Dichas entidades se compondrán de:

  - **Características:** las entidades pueden tener características que son incluidas por el creador del espacio en la aplicación y serán visibles por los usuarios. Estas características tendrán diferentes tipos predefinidos de acuerdo a la información que contengan, ya sea una descripción, una puntuación, etc.
    Las características más allá del tipo se dividen a su vez en dos grupos diferenciados:
    * **Características fijas:** Son características que definen información inmutable acerca del lugar y por tanto son definidas por el usuario que dio de alta el lugar y no están sujetas a opinión de los usuarios.
    * **Características mutables:** Son características también definidas por el usuario que dio de alta el lugar, pero su valor depende de las opiniones de los usuarios con respecto a estas.

  Además de las características, los usuarios de la aplicación podrán dejar comentarios acerca de los lugares. Dichos comentarios tendrán una puntuación basada en likes y dislikes que servirá para dar mayor credibilidad al comentario y para animar a la comunidad a expresar su aprobación sin necesidad de escribir otro comentario.

- **Usuarios:** hay dos tipos de usuarios, los verificados y los no verificados. Cada vez que se conecte el usuario a la aplicación, se guarda su ubicación, para usarla en el centrado del mapa a la hora de realizar búsquedas.

  El sistema de verificación está inspirado en otros sistemas similares como los de Twitter o Facebook, en los que las cuentas que lo soliciten podrán acreditar documentos que demuestren su vinculación con la propiedad que han dado de alta, y una vez comprobados manualmente, un icono aparecerá en la cuenta de usuario y del lugar indicando que son cuentas oficiales.

- **Buscador de lugares:** El usuario puede buscar en un radio de cercanía, sumándole etiquetas para estrechar más las búsquedas y que sólo aparezcan los lugares de interés.

- **Panel de estadísticas:** En este panel, los usuarios autorizados tendrán acceso a visualizar diferentes métricas asociadas a las estadísticas obtenidas a partir de los lugares dados de alta.

3.5. Herramientas Software

A continuación se van a exponer las tecnologías seleccionadas para construir el servicio y la aplicación que se van a desarrollar y las razones de por qué se han escogido.

En primer lugar se van a exponer las tecnologías escogidas para la programación del servicio que sustenta a la aplicación Android (Starppy), y posteriormente las tecnologías implicadas en el desarrollo de esta última (GoTo).

3.5.1. Servicio y framework

En el caso de Starppy, se buscaban una serie de características principales:

- Una característica muy importante para el servicio debía ser la **compatibilidad**, ya que si se quería estructurar el servicio como un framework reutilizable, debía ser compatible con multitud de lenguajes de programación de manera sencilla, así como con multitud de gestores de bases de datos, servidores web y sistemas operativos.

- Se buscaba también un conjunto de características que permitieran un **desarrollo ágil, mantenible y modularizable**, dado que se quería dar la mayor facilidad de integración a otros programadores. También se valoraría una buena legibilidad de código, así como facilidades para la documentación y publicación.

- Se buscaba también utilizar únicamente **software libre**.

Tipo de servicio

Finalmente se ha escogido utilizar un tipo de servicio REST JSON sobre HTTP frente a otros tipos de servicio, como SOAP u otros protocolos basados en XML, ya que ofrece algunas ventajas importantes:

- Los servicios REST no tienen estado, por lo que son muy fáciles de implementar, sobre todo en entornos ligeros como webs o aplicaciones móviles. Esto hace que también existan multitud de librerías en muchos lenguajes de programación que facilitan el desarrollo.

- El hecho de que las comunicaciones sean a través de HTTP hace que la gestión de las comunicaciones, la autenticación y autorización sean también sencillas, a través de una interfaz uniforme (POST, GET, DELETE). Además el JSON es un lenguaje de datos fácilmente interpretable y manejable por las personas.
- La arquitectura de este tipo de sistemas permite estructurar los elementos accesibles a través de una URL, lo que provee de una estructura ordenada y jerárquica, lo que hace más sencilla la integración por parte de otros programadores.
- La especificación OpenAPI (antes Swagger Specification [38]) permite definir este tipo de servicios de tal forma que la documentación pueda ser generada automáticamente.

Implementación del servicio

Para cumplir con estas premisas, se ha seleccionado Django y su plugin Django Rest Framework con el fin de ofrecer un servicio REST sobre HTTP desarrollado en Python. A continuación se van a comentar las razones de esta elección:

Python

Python [15] se caracteriza principalmente por su sintaxis tan limpia y por promover el buen uso de las normas a la hora de escribir código, empezando por forzar una indentación que facilite ver la estructura y el funcionamiento del código de un primer vistazo.

Éste lenguaje incorpora módulos, excepciones, escritura dinámica, tipos de datos dinámicos y clases. Permite escribir parte del código en C o en C++, y funciona en varias versiones del Sistema Operativo Unix (Ubuntu, Debian, etc), además de en Mac y Windows. Esto aporta la posibilidad de desplegar fácilmente Starppy sea cual sea el S.O. en el que se esté trabajando.

Python es muy útil y se utiliza mucho en el ámbito profesional ya que con él se pueden resolver muchos y muy distintos problemas. Cuando se instala Python, ya trae consigo una gran librería con multitud de librerías y funciones predefinidas útiles en varios campos como el procesamiento de strings, la utilización de los protocolos de Internet, la ingeniería del software y la posibilidad de interactuar con las interfaces de los sistemas operativos.

Python es también apoyado por una gran comunidad de programadores que destinan sus esfuerzos a realizar más librerías y paquetes con utilidad. Además, estos trabajos suelen ser publicados para que el resto de la comunidad de programadores puedan usarlos en sus proyectos personales, y favorecer a la comunidad.

Además, es sencillo encontrar guías y tutoriales, con ejemplos de uso y comentarios, sin contar con la multitud de libros disponibles y la amplia oferta de IDEs que son compatibles con la sintaxis de python.

Django

Django [14] es el Framework de Python para desarrollo web. Fue desarrollado siguiendo el paradigma de Modelo-Vista-Controlador. La motivación central de Django es la agilidad en el desarrollo, trabajo y el ahorro de recursos a la hora de crear sitios web complejos, siguiendo el principio Don't Repeat Yourself. De esta forma se anima al programador a re-utilizar su código, además de facilitar una gran conectividad y claridad entre los componentes que componen el sistema.

Django está enteramente escrito en Python. Para más facilidades, Django incorpora una versión liviana de servidor web, para realizar pruebas y trabajos de depuración de forma más rápida y directa. Posee compatibilidad con los motores de base de datos más utilizados incluyendo SQLite para hacer pequeñas pruebas de forma más sencilla. Una de sus mayores ventajas es su API de abstracción de base de datos a través de modelos. A través de diferentes clases y métodos, se representan en forma de objetos enteramente descritos en python todas las entidades de la base de datos, de forma que su manejo es mucho más sencillo además de ser independiente del motor de base de datos subyacente.

Otro de los aspectos positivos de utilizar Django en este proyecto, es que incorpora un panel de administración web de la base de datos. Con él se puede modificar de forma sencilla los permisos de los usuarios y añadir, borrar o modificar elementos contenidos en la base de datos.

Como pasa con Python, Django también da soporte a una documentación extensa y con ejemplos en multitud de idiomas.

Django REST framework

Django REST framework [39] es un paquete de herramientas que se ensamblan junto a Django para crear un potente framework de trabajo donde se facilita la creación de APIs para servicios web.

Es conveniente utilizar esta herramienta ya que gestiona de una manera simple al exposición de los modelos de Django a través de un servicio REST, además de gestionar otros elementos como la autentificación o la serialización de los elementos.

También cuenta con una extensa documentación, ejemplos, guías y tutoriales para comenzar a usarlo rápidamente.

En este proyecto será ventajoso utilizar esta tecnología ya que facilitará y agilizará el desarrollo del API web que comunique la base de datos con la aplicación de móvil.

3.5.2. Aplicación Android

Android Studio

Android Studio [11] es el IDE, oficialmente soportado por Google para programar una aplicación para Android.

En Android Studio se utiliza el lenguaje Java (su sintaxis) junto con el Android Software Development Kit (Android SDK) para realizar aplicaciones. Dichas aplicaciones funcionan gracias a una DVM, Dalvik Virtual Machine, que es una implementación similar a una Java Virtual Machine, preparada para compilar en tiempo de ejecución. A partir de la versión 4.4 de Android se introdujo ART, Android Runtime [31], que es un nuevo entorno de ejecución que transforma las aplicaciones a instrucciones máquina tras su instalación. Esto evita que se haga cada vez que se abre una aplicación, ahorrando bastantes recursos.

Las ventajas más importantes que presenta son:

- El Android Studio puede funcionar tanto en Windows, en Mac o en Linux, lo que lo hace muy útil ya que facilita al programador trabajar en cualquier entorno. También incluye otras facilidades como control de versiones o descarga automática de librerías externas y SDKs desde los repositorios de Google.

- Facilita mucho el desarrollo su modo de diseño WYSISYG para la interfaz gráfica de la aplicación, que entre otras cosas contiene colecciones de elementos gráficos predefinidos, lo que hace muy sencillo construir una aplicación según la guía de estilo del Material Design de Google [18].
- Contiene todas las herramientas de compilación y empaquetado necesarias, así como los drivers adecuados para depurar la aplicación directamente sobre un dispositivo móvil, lo que hace muy ágil el desarrollo, ya que pueden probarse los cambios rápidamente durante el desarrollo.

En este trabajo se va a utilizar este IDE debido a la cantidad de ventajas que presenta a la hora de programar una aplicación de móvil, hacer pruebas, diseñarla y resolver errores. También es cierto que si se quiere desarrollar para las últimas versiones de Android, no existen muchas más alternativas ya que este IDE es el soportado por Google para las versiones actuales de Android.

Java

En el entorno proporcionado por Google para programar para Android, se puede programar en C++ (usando el NDK) y en Java (usando el SDK). Se ha escogido Java [36] como lenguaje para desarrollar la aplicación por la mayor agilidad en el desarrollo frente a C++.

Programar en C++ provee al programador mucha más libertad para manejar los recursos hardware del dispositivo móvil por lo que las aplicaciones son mucho más eficientes, pero a costa de una interfaz de mucho más bajo nivel. Por ejemplo, con C++ no se podría hacer uso de los diseñadores WYSIWYG para la interfaz, o del recolector de basura de Java. Además el hecho de tener que compilar las aplicaciones hasta código máquina y no un código interpretable por la máquina Dalvik de Android, produce incompatibilidades y caídas de rendimiento dada la amplitud de posibilidades de hardware que soporta Android.

Retrofit

Como se ha comentado antes, la aplicación Android deberá comunicarse con un servidor REST a través de HTTP, para ello se ha seleccionado la librería RetroFit [21].

Esta librería, aparte de ser software libre, provee a cualquier aplicación Android de un cliente HTTP muy ligero diseñado para realizar peticiones a un servicio REST.

La sintaxis, a base de decoradores para definir los endpoints del servicio, junto con su facilidad de instalación y a la inclusión de gestores de autorización y autentificación hacen a RetroFit una buena elección para este proyecto.

4

Requisitos

4.1. Estructura de diseño: Módulos y subsistemas

Con el fin de mejorar la comprensión y facilitar el análisis y el diseño del proyecto, este se ha dividido en dos grandes módulos y estos a su vez en subsistemas independientes:

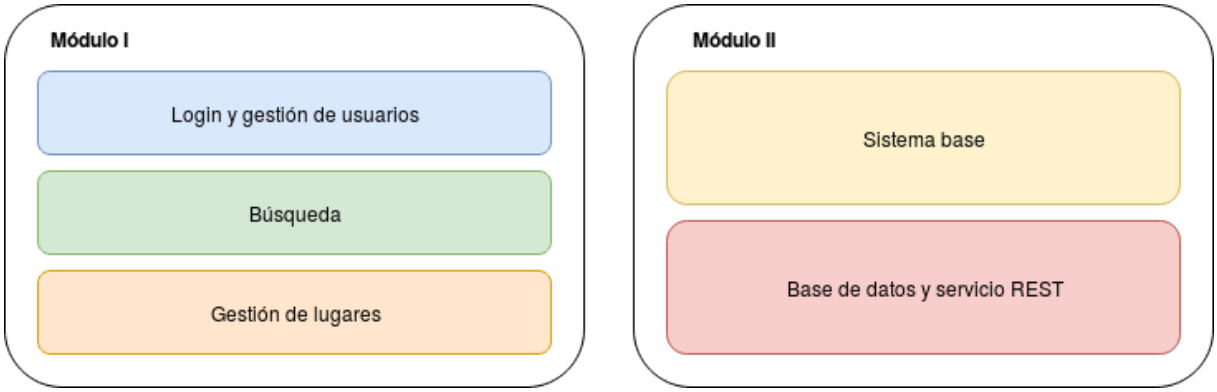


Figura 4.1: Esquema de los diferentes subsistemas del proyecto

4.1.1. Módulo I: Aplicación Android

Este módulo consta de la aplicación Android a través de la cual los usuarios de la aplicación tendrán acceso al sistema, así como todo lo necesario para empaquetarla e instalarla en los dispositivos móviles. Este módulo consta de los siguientes subsistemas:

- **Subsistema I: Login y gestión de usuarios.** En este subsistema se engloban todas las pantallas y funcionalidad que permitirán a los usuarios de la aplicación autentificarse, crear nuevas cuentas y gestionar su perfil.

- **Subsistema II: Búsqueda.** En este subsistema, se incluyen las pantallas a través de las cuales los usuarios de la aplicación podrán realizar búsquedas y cambiar los filtros de acuerdo a sus preferencias, así como visualizar e interactuar con los resultados de dichas consultas.
- **Subsistema III: Gestión de lugares.** A través de las pantallas de este subsistema, los usuarios podrán dar de alta lugares en el sistema y definir sus características, así como visualizar otros lugares compartidos con la comunidad y aportar sus opiniones acerca de estos.

4.1.2. Módulo II: Backend y servicio REST

En este módulo se engloban todas aquellas estructuras de backend que permiten funcionar al sistema. Estos servicios proveerán a los clientes en los dispositivos móviles de la persistencia y la funcionalidad que necesitan. De la misma manera, este módulo también se ha diseñado dividido en subsistemas:

- **Subsistema IV: Sistema base.** En este subsistema se engloban todas las actividades y configuraciones encaminadas a proveer un entorno Linux para el servicio REST del sistema. Esto incluye la instalación de un gestor de base de datos para mantener una base de datos relacional, la instalación y configuración de un servidor de aplicaciones para albergar el servicio REST y las configuraciones y herramientas necesarias para la monitorización y el mantenimiento de estas estructuras.
- **Subsistema V: Base de datos y servicio REST.** En este subsistema se engloba la creación y configuración de una base de datos relacional que dote de persistencia a los clientes móviles así como un servicio REST que les permita, a través de peticiones HTTP con formato JSON, añadir, modificar o eliminar información de la base de datos con su consiguiente autorización y autentificación. De esta forma, otros programadores de la comunidad podrán utilizar este servicio para el desarrollo de sus aplicaciones.

4.2. Análisis de requisitos

4.2.1. Subsistema I: Login y gestión de usuarios

Requisitos funcionales

- La aplicación constará de una pantalla de login donde los usuarios introducirán sus credenciales (usuario y contraseña). Esta pantalla deberá también informar si hay algún problema con las credenciales y redirigir al usuario si procede a la pantalla adecuada tras pulsar el botón de login.
- En la pantalla principal deberá haber también un botón de crear cuenta que utilizará los mismos campos de escritura que el login, y que permitirá al usuario crear una nueva cuenta. También se le informará de si el usuario que ha escogido ya no está disponible.
- El usuario tendrá acceso permanente a una pantalla donde podrá modificar los datos de su perfil (información básica sobre su identidad, su avatar, etc.) cada vez que lo desee. También contará con una lista de los lugares dados de alta por el usuario y le permitirá añadir nuevos lugares o eliminar alguno ya existente.

Requisitos no funcionales

- El mecanismo de login y autenticación se hará a través de autentificación básica HTTP. En producción se necesitará desplegar con su correspondiente servidor HTTPS.
- La contraseña deberá tener mínimo 8 caracteres. De la misma forma el usuario deberá tener como mínimo 4.

4.2.2. Subsistema II: Búsqueda

Requisitos funcionales

- Esta pantalla debe contar de un mapa centrado en la posición GPS del usuario, un desplegable para filtrar por distancia los objetivos y una caja de texto para introducir términos de búsqueda.
- Cada vez que se edite uno de los filtros disponibles en la pantalla, la aplicación debe realizar la búsqueda de lugares dentro del filtro de distancia y que tengan coincidencias con las palabras del texto de búsqueda. La aplicación mostrará entonces dichos resultados en tarjetas en una lista con scroll. También situará un marcador en el mapa por cada uno de los resultados obtenidos.
- Al entrar en esta pantalla, se realizará una primera búsqueda automática con unos filtros predefinidos: distancia basada en la ubicación GPS del usuario y sin texto de búsqueda.
- Al hacer clic sobre uno de los resultados, el mapa se centrará sobre el marcador en el mapa de dicho resultado. También aparecerán iconos sobre el mapa para iniciar una navegación usando el teléfono o para ver la localización con la aplicación Maps.
- Si se hace clic sobre cualquier marcador del mapa, la aplicación redireccionará al usuario a la página de visualización de dicho lugar.
- Si no hay resultados para una búsqueda, debe informarse al usuario de ello.

Requisitos no funcionales

- La ubicación para realizar la búsqueda se obtendrá de la red móvil o wifi y de las coordenadas GPS obtenidas del teléfono. Para ello se pedirá los permisos necesarios al usuario.
- Para la realización del mapa de la pantalla de búsqueda se utilizará los servicios de Google Maps con un API KEY.

4.2.3. Subsistema III: Gestión de lugares

Requisitos funcionales

- La gestión de los perfiles de lugar correrá a cargo del creador del mismo. Esto implica poder modificarlos o borrarlos.
- Los lugares constaran de una información mínima obligatoria como el nombre, el avatar, una descripción y una ubicación del lugar.

- Los lugares podrán contener comentarios hechos por los usuarios. Estos comentarios podrán ser votados por el resto de usuarios haciendo que se quede reflejado el acuerdo o desacuerdo general.
- Los lugares tendrán características que podrán ser creadas por el administrador, y votadas por la comunidad de usuarios. Dichas características podrán representarse en forma de estrellas, de nota de 0 a 10 o de me gusta/no me gusta.
- Las características de un lugar podrán ser borradas por el creador del perfil del lugar.
- Cuando un usuario crea el perfil de un lugar, la ubicación por defecto será la del usuario en ese momento.
- Si un perfil de lugar no tiene características y/o comentarios se informará de ello.

Requisitos no funcionales

- Internamente el sistema debe de negar el acceso de los usuarios a la edición de los datos de perfiles de lugar que no hayan sido creados por ellos mismos.

4.2.4. Otros requisitos generales de la interfaz:

Requisitos funcionales

- La interfaz seguirá los cánones dispuestos por el Material Design propuesto por Google [18], tanto en la colocación y en el aspecto de los controles, como en colores y tipografías.
- En las pantallas que lo requieran, deberá incluirse en la barra superior de la aplicación una flecha para volver a la pantalla anterior.
- En las pantallas que lo requieran, se añadirá un menú arrastrable (drawer) a través del cual se podrá navegar por las diferentes pantallas de la aplicación.

Requisitos no funcionales

- La aplicación móvil debe funcionar en smartphones con Sistema Operativo Android y con una versión 5.0 o mayor.
- La navegación en la aplicación debe ser fluida y deben incorporarse barras de progreso o elementos en movimientos en las zonas de pantalla donde los datos puedan tardar tiempo en aparecer, para informar al usuario de que la aplicación no está bloqueada.
- Todas las cadenas de la aplicación estarán en español, pero deberán hacer uso del sistema de cadenas de Android, de manera que la aplicación esté preparada para ser traducida a cualquier idioma de manera sencilla.
- En todo momento la aplicación debe informar de los errores de conexión o cualquier otro error acontecido en la aplicación, sin dar información técnica innecesaria.

4.2.5. Subsistema IV: Sistema base

Requisitos funcionales

- El sistema base contará con un gestor de base de datos relacional que contenga la base de datos del servicio.
- El sistema debe contar con un servidor de aplicaciones que permita ejecutar el servicio y exponerlo al exterior.
- El sistema dispondrá también de una herramienta de monitorización que lleve un control sobre los recursos y el rendimiento del equipo (procesador, memoria y red) y permita en caso de fallo consultar el estado del servidor.
- El sistema contará con un sistema de control de procesos que se encargue de mantener todos los programas antes mencionados en ejecución, de detectar si se bloquean y reiniciarlos si es necesario y de llevar un control de los mensajes de estado que estos programas emitan.
- El servidor será desplegado de manera remota, por lo que debe tener un servicio SSH activado para tener comunicación con él.

Requisitos no funcionales

- El gestor de base de datos y el servidor de aplicaciones deberán ser capaces de atender las peticiones necesarias para el funcionamiento fluido de la interfaz.
- El sistema base debe tener unas garantías de estabilidad y seguridad, por eso deberá ser implementado sobre Linux con las últimas actualizaciones de seguridad instaladas.

4.2.6. Subsistema V: Base de datos y servicio REST

Requisitos funcionales

- La estructura de la base de datos y del servicio REST deben ser genéricas, permitiendo que puedan utilizarse como servicio base para otras aplicaciones de búsqueda, visualización, opinión y adquisición de estadísticas más allá de la aplicación que se está desarrollando.
- La base de datos del sistema debe incluir información acerca de los lugares contenidos en el sistema y sus características, los perfiles de los usuarios y datos asociados así como las opiniones y comentarios de la comunidad.
- El servicio REST contará con acceso a través de un navegador web. De esta forma se presentará el resultado de consultas al servicio de manera más agradable para los desarrolladores con el fin de hacer más fácil su integración en aplicaciones diferentes a esta que se desarrolla.
- El servicio REST debe incluir también una especificación de la misma en el estándar OpenAPI, que permita generar la documentación necesaria para que otros desarrolladores integren el servicio en sus aplicaciones.
- El servicio contará con un portal web de administración, donde los administradores del sistema podrán visualizar, modificar, añadir y borrar los datos almacenados con el fin de depurar posibles errores y agilizar las pruebas del servicio, haciendo más fácil así su integración con más aplicaciones.

- El servicio REST permitirá la lectura, modificación y borrado de la información de la base de datos a través de peticiones HTTP estándar.
- El servicio REST contará con un servicio de autentificación que evite el acceso a la información a clientes no autentificados. Esta será a través de autenticación HTTP.

Requisitos no funcionales

- Se deberá crear un dominio DNS para poder permitir a los clientes móviles acceder al servicio REST aunque la IP del host de este cambie. Para ofrecer conectividad HTTPS hará falta obtener un certificado SSL para este dominio.

5

Diseño e implementación

En este apartado se va a detallar la arquitectura diseñada, tanto para para la aplicación Android como para el backend que lo sustenta así como para el pequeño portal web para consultar estadísticas.

El diseño de GoTo comenzó con el diseño de la aplicación de móvil de modo que cumpliera con todas las expectativas del análisis del problema que se había realizado. Paralelamente, se empezó también a diseñar un Framework genérico que sirviera de persistencia y centralización de datos para esta aplicación, sin perder de vista que sirviese a otros programadores que quisieran realizar una aplicación de móvil del mismo estilo o con un objetivo similar, para poder, en última instancia ser liberado como proyecto open source.

Por último, se diseñó una aplicación web que sirve a los usuarios para visualizar las estadísticas de los lugares que han dado de alta a través de la aplicación.

A continuación se detalla en primer lugar el diseño, la arquitectura del backend y el portal web de estadísticas y posteriormente el diseño de pantallas y navegación de la aplicación android.

5.1. Diseño, implementación y desplegado del Backend

El backend de la aplicación es una infraestructura remota que provee a los clientes móviles de los datos que necesitan para poder hacer posible que GoTo funcione. Este backend se compone de varias partes diferenciadas: un servicio REST interactivo a través del navegador, una base de datos, una aplicación web de administración, y una aplicación web de estadísticas.

5.1.1. Base de datos

Dado que la información es el centro del proyecto, la base de datos es el corazón del backend de la aplicación, puesto que es la encargada de mantener la persistencia que permite mostrar los datos a los usuarios o calcular las estadísticas.

Es importante tener en cuenta, que con este backend se quiere proveer de un servicio genérico, que sirva no solo para esta aplicación, sino para otras que tengan una estructura similar. De forma que los datos se modelizan en torno a dos conceptos principales, las entidades (entities) y las caracteristicas (features) asociadas a ellas, que constituyen también las dos tablas principales de la base de datos:

- **Entity (entidad):** Una entidad es aquel elemento con el que se quiera relacionar a los usuarios por medio de búsquedas y opiniones. En el caso de GoTo, la tabla Entity va a contener lo que luego en la aplicación se denominará como *Lugar*. De la misma manera, una Entidad podría modelizarse en otra aplicación como *Vino* y entonces se convertiría en una aplicación para comentar y votar acerca de distintos tipos de vinos. Como se puede apreciar, esto ofrece una extensa variedad de posibilidades a la hora de realizar una aplicación basándose en el framework modelizado en este backend. Los campos mínimos para identificar a una Entidad son:
  - **Id de cuenta:** Un id que relaciona la entidad con la cuenta del usuario que la creó.
  - **Nombre:** Un nombre descriptivo para ser mostrado a los usuarios.
  - **Descripción:** Un texto que acompaña al nombre pensado para ser también mostrado al usuario y aportar información algo más detallada acerca de la entidad.
  - **Foto:** Un fichero de imagen listo para ser incorporado en el cliente y ofrecerlo al usuario para identificar mejor la entidad.
- **Feature (característica):** Todas las entidades están modelizadas a partir de características, que son valores nominados cuyo valor cambia de acuerdo a las valoraciones de usuarios de la aplicación. Estas características tienen también un tipo asociado, que permite a las aplicaciones cliente saber cómo deben presentar estos valores a los usuarios.

  En el caso de GoTo, un lugar podría por ejemplo, tener una característica llamada *Trato del personal* que podría ser presentada como una nota del 1 al 10 o como una graduación de 0 a 5 estrellas en la aplicación cliente. Su valor cambia de acuerdo a las valoraciones de los usuarios.

  Un lugar puede tener varias características, pero una característica es única de un solo lugar.

  Se modelizan en torno a los siguientes campos:
  - **Id de entidad:** Es un identificador que relaciona la característica con la entidad a la que pertenece.
  - **Nombre:** El título de la característica.
  - **Tipo:** Un valor numérico que indica al cliente cómo tratar el valor de la característica de cara al usuario.

Las entidades se modelizan principalmente a través de las características, y los usuarios pueden, a través del cambio de sus valores, aportar información sobre la entidad. Sin embargo, esta no es la única forma que tienen los usuarios para participar, también pueden escribir comentarios refiriéndose a una entidad y así aportar más información sobre ella. De esta manera se define también una nueva tabla:

- **Comentarios:** A través de este elemento, se almacenan los comentarios que los usuarios hacen sobre las entidades. Estos comentarios constan de un texto y pueden a su vez ser puntuados por otros usuarios como positivos o negativos. Los comentarios se modelizan de la siguiente manera:

  - **Id de cuenta:** Identificador para relacionar el comentario con la persona que lo ha escrito.
  - **Id de entidad:** Identificador que relaciona el comentario y la entidad sobre la que se ha comentado.
  - **Timestamp:** Fecha y hora a la que se hizo el comentario para recoger datos para las estadísticas.
  - **Título:** Título del comentario, pensado para ser mostrado como un resumen corto del comentario en la aplicación cliente.
  - **Description:** Texto que forma el grueso del comentario.

Para guardar cada una de las interacciones de los usuarios con las características y con los comentarios, se han diseñado dos tablas más:

- **Rate:** Es una tabla que sirve para relacionar los votos que ha dado cada usuario a cada característica de cada entidad. De esta forma, a la hora de crear unas estadísticas útiles, se puede relacionar a los usuarios con las entidades a través de sus opiniones y a través del tiempo.

  Para llevar a cabo esta tarea se necesitan los siguientes campos:

  - **Id de característica:** el identificador que relaciona la votación con la característica en concreto (que a su vez pertenece a una entidad) que se ha votado.
  - **Id de cuenta:** Es el identificador de usuario que ha votado, de esta forma se puede acceder fácilmente a sus datos de edad, género, etc.
  - **Value:** Este campo recoge el valor que el usuario ha asignado a la característica.
  - **Timestamp:** En este campo se guarda la fecha y la hora a la que se hizo la votación.

- **RateCommentary:** se trata de una tabla con el mismo objetivo que la tabla Rate, pero en vez de dirigida a recoger los votos de una característica, recoge los votos que se han hecho a un comentario, y de esta forma se puede ver si la gente está de acuerdo o no con lo que comentan el resto de usuarios.

  Para recoger estas estadísticas, se disponen de los siguientes campos:

  - **Id de comentario:** Identificador que relaciona el voto con el comentario al cual se refiere.
  - **Id de cuenta:** Identificador del usuario que vota.
  - **Valor:** Este campo recoge si la votación ha sido a favor o en contra.
  - **Timestamp:** Campo para guardar la fecha y la hora a la que se votó.

También se han creado otras tablas que son necesarias para el servicio:

- **User:** Esta tabla es añadida por defecto por Django. Django ofrece un sistema de autenticación genérico y se ha aprovechado en el proyecto. Por tanto esta tabla guarda los nombres de usuario con los que se registra la gente en GoTo y sus respectivas contraseñas en forma de hash.

- **Account:** Esta tabla contiene información personal sobre cada usuario. Con el fin de ofrecer a través del servicio información sobre los usuarios que la componen así como para tener información como el sexo y la edad de los participantes y poder así calcular estadísticas:

- **Nombre:** Un nombre de usuario con el cual él pueda identificarse con su perfil.
- **Id de usuario:** Identificador que enlaza con la tabla User.
- **Edad:** Campo que contiene la edad del usuario.
- **Género:** Campo que contiene el género del usuario.
- **Avatar:** Un fichero de imagen listo para ser incorporado en el cliente y ofrecerlo a través de la interfaz para identificar mejor a los usuarios.

A continuación se muestra un diagrama de Entidad-Relación que representa lo que es la estructura de la base de datos:

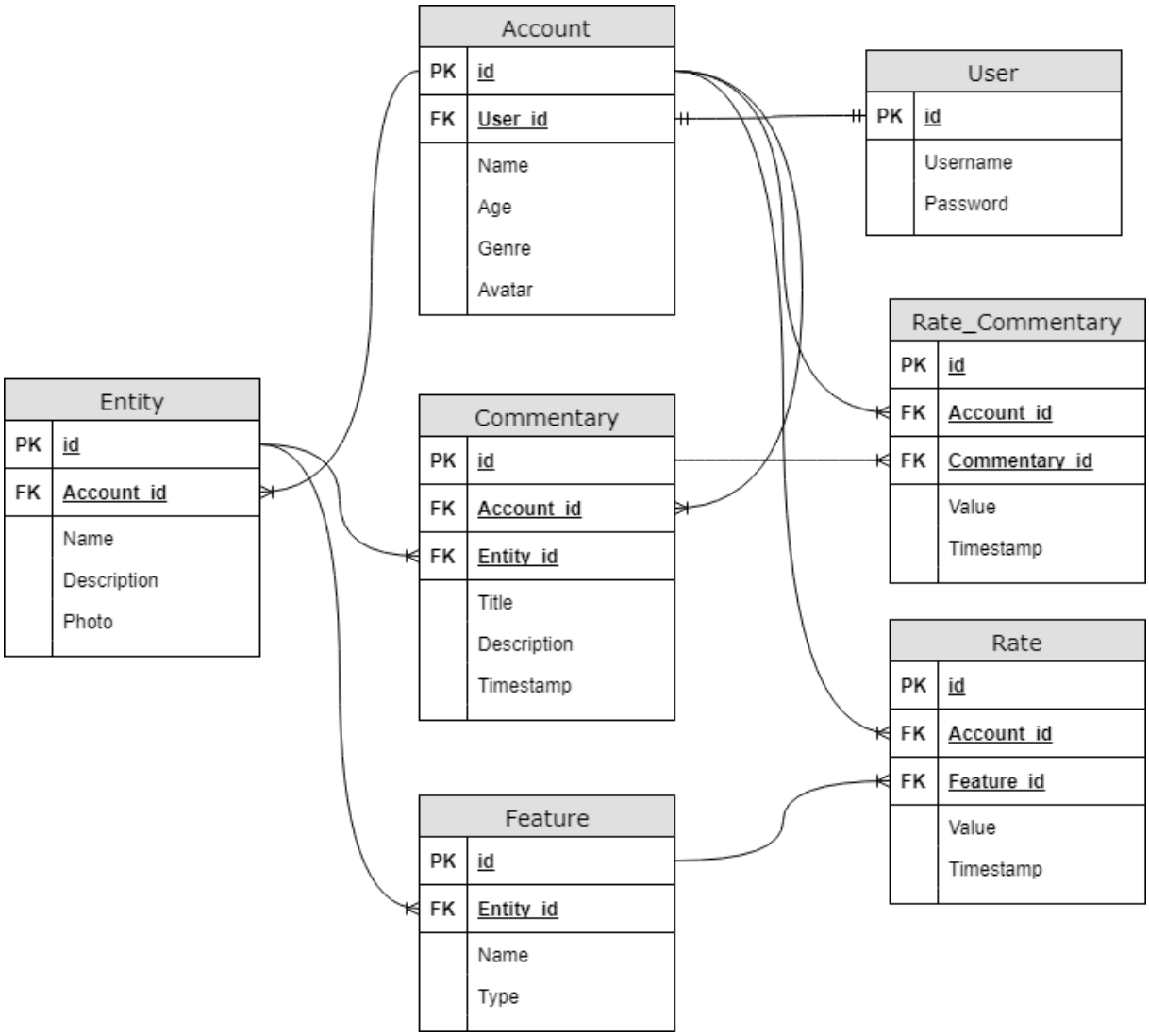


Figura 5.1: Diagrama Entidad-Relación de la base de datos

5.1.2. Servicio REST

El servicio REST de GoTo está construido sobre Django y su librería REST Framework. Este servicio hace público un API a través del cual los clientes utilizando peticiones HTTP estándar pueden añadir, borrar o modificar entidades, características, comentarios, etc.

Estructura del servicio

El servicio se estructura en endpoints, a través de los cuales se puede interactuar con los diferentes elementos de la base de datos expuestos en el apartado anterior. Todos los endpoints se estructuran sobre la URL base del servicio:

`http://<ip_servicio>:<puerto_servicio>/<endpoint>`

El servicio REST es accesible a través del navegador, y si se accede a través de este método, se muestra una interfaz web que hace más fácil al programador interactuar con el servicio. Por ejemplo, pueden verse todos los endpoints disponibles en el servicio entrando en la URL base:

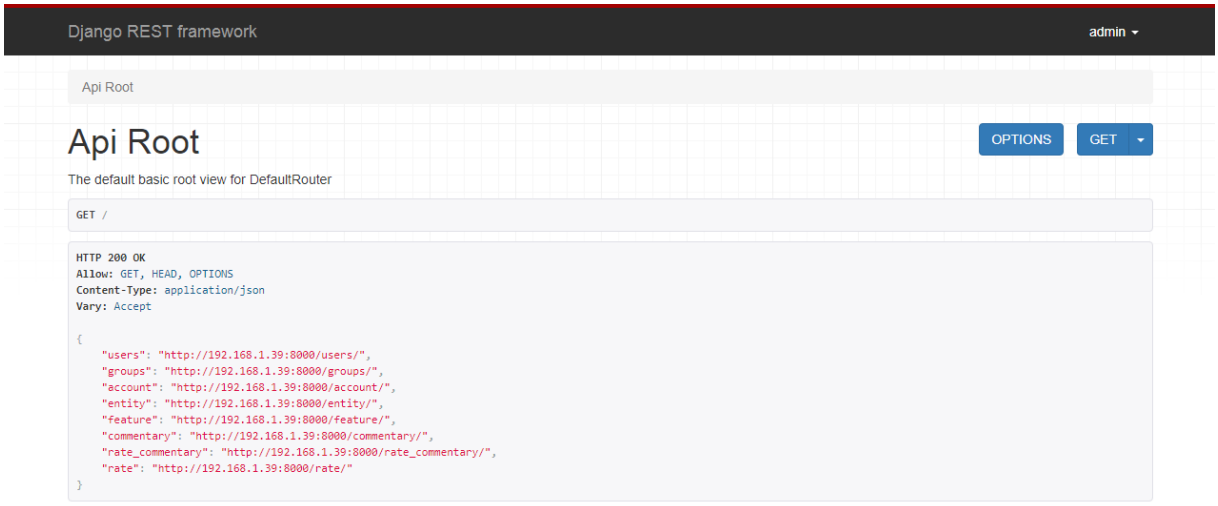


Figura 5.2: Raíz del API con el listado de los endpoints disponibles

Para obtener una salida limpia en formato JSON sin HTML para ser leído por un cliente, basta con añadir *?format=json* a la URL:

{"users":"http://192.168.1.39:8000/users/?format=json","groups":"http://192.168.1.39:8000/groups/?format=json","account":"http://192.168.1.39:8000/account/?format=json","entity":"http://192.168.1.39:8000/entity/?format=json","feature":"http://192.168.1.39:8000/feature/?format=json","commentary":"http://192.168.1.39:8000/commentary/?format=json","rate_commentary":"http://192.168.1.39:8000/rate_commentary/?format=json","rate":"http://192.168.1.39:8000/rate/?format=json"}

Figura 5.3: Devolución del API en formato JSON

Desde cualquier endpoint del sistema, pueden listarse, editarse y borrarse los elementos a los que el endpoint se refiera, haciendo uso de peticiones HTTP de tipo GET, POST/PUT, PATCH y DELETE respectivamente.

La autorización de todo el API se realiza a través de autentificación estándar HTTP, por lo que en producción, el API debe estar securizado por HTTPS.

Para referirse a la lista completa de elementos para listarlos o añadir nuevos, la URL a utilizar es la URL base del endpoint, por ejemplo:

`http://<ip_servicio>:<puerto_servicio>/rate`

El servicio permite también referirse a un elemento en concreto, por ejemplo para listarlo, editarlo o borrarlo. Para ello basta con añadir el ID del elemento sobre la URL base del endpoint:

`http://<ip_servicio>:<puerto_servicio>/rate/1`

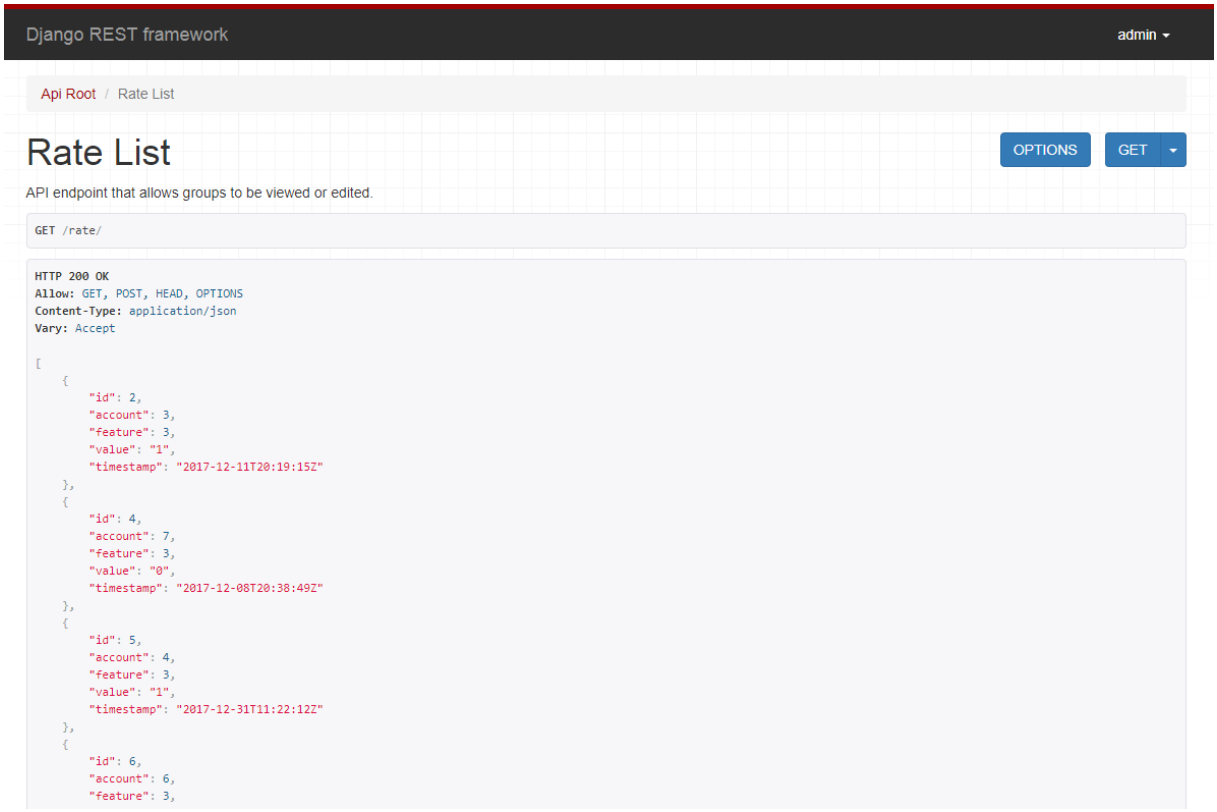


Figura 5.4: Endpoint de tipo lista para Rate

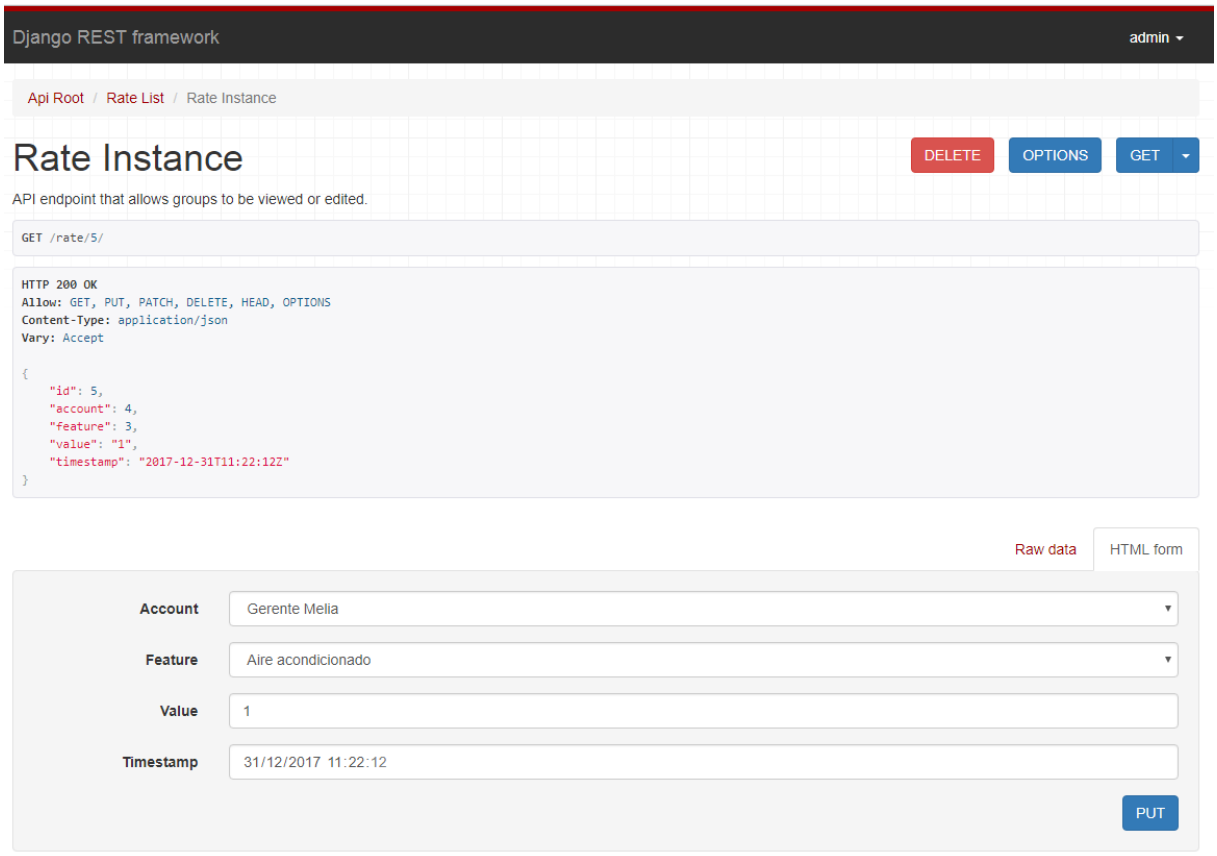


Figura 5.5: Endpoint de tipo detalle para Rate

Muchos elementos del API también soportan ciertas funciones, como por ejemplo en el caso de una cuenta de usuario, puede editarse su avatar:

```
http://<ip_servicio>:<puerto_servicio>/account/1/update_avatar
```

Estas funciones son accesibles a partir de la URL de un elemento en concreto redireccionado por su ID y son siempre solicitadas a través de una petición GET o POST. La respuesta de estas peticiones es siempre el elemento sobre el que se ha invocado la funcionalidad.

De la misma manera, existen funciones que no se aplican a un elemento en concreto y se ejecutan sobre la URL base del endpoint, como es el caso de la obtención de una cuenta de usuario de acuerdo al id de usuario:

```
http://<ip_servicio>:<puerto_servicio>/account/get_account_by_user_id/
```

La respuesta a estas llamadas es siempre una lista de elementos, de acuerdo al resultado de la función solicitada.

Documentación del servicio

La información sobre las diferentes funciones y la sintaxis de los elementos, puede ser obtenida desde la documentación del API, que está presente en el propio servidor en el endpoint */docs*. Esta documentación está estructurada según el estándar OpenAPI (ahora llamado swagger) [38] e incluye toda la información necesaria para que un programador incluya el servicio en una aplicación propia:

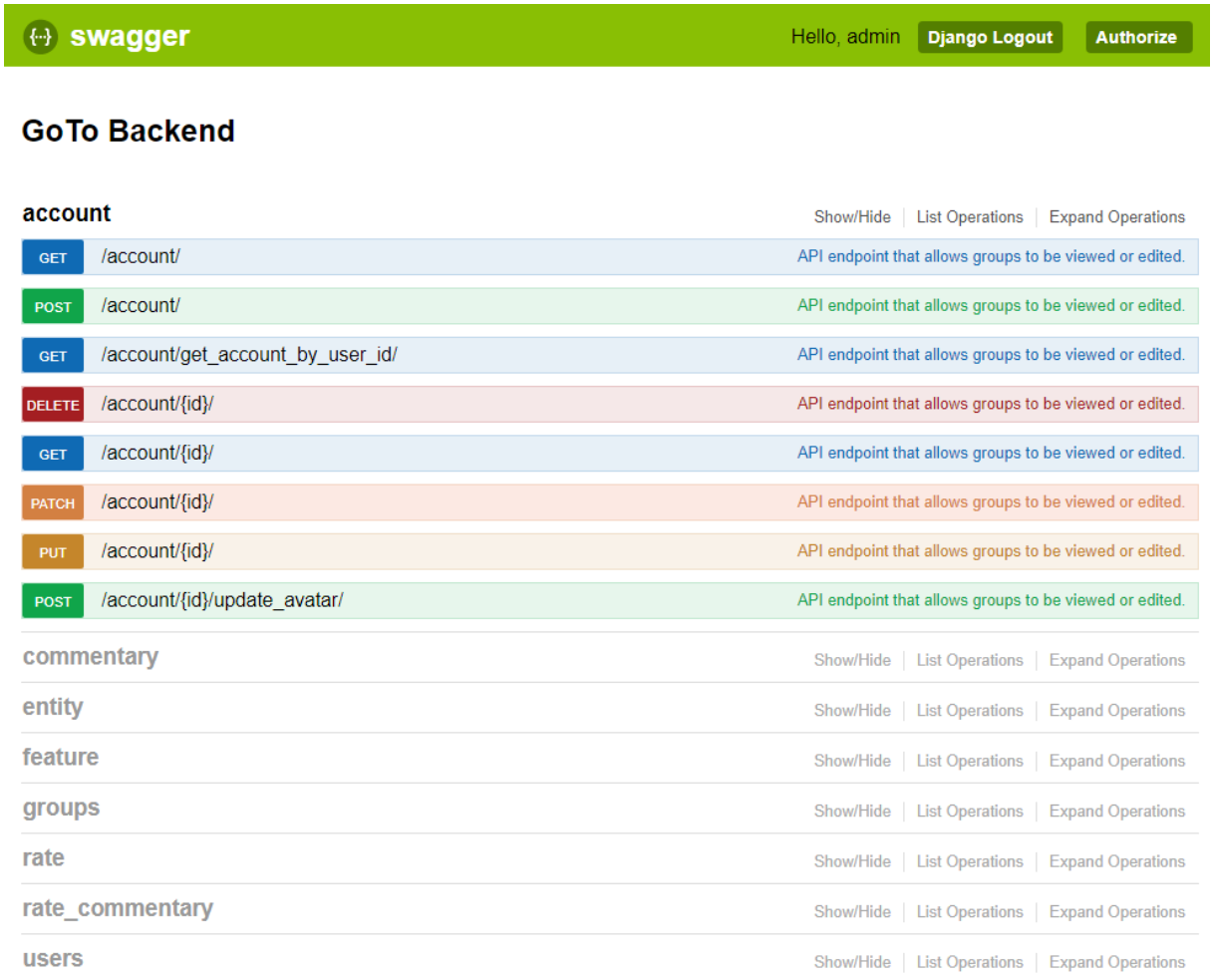


Figura 5.6: Documentación generada en formato OpenAPI

Esta interfaz ha sido generada utilizando el paquete django-rest-swagger [27] de Django y REST framework.

5.1.3. Aplicación web de administración

Con el objetivo de que los administradores del sistema tengan un mejor control sobre la base de datos y hacer más sencillo el proceso de depurado en la integración del servicio por parte de los programadores que quieran utilizarlo, se ha implementado una interfaz web de administración que permite consultar, modificar y añadir registros desde el propio navegador. Está disponible a través del endpoint */admin*:

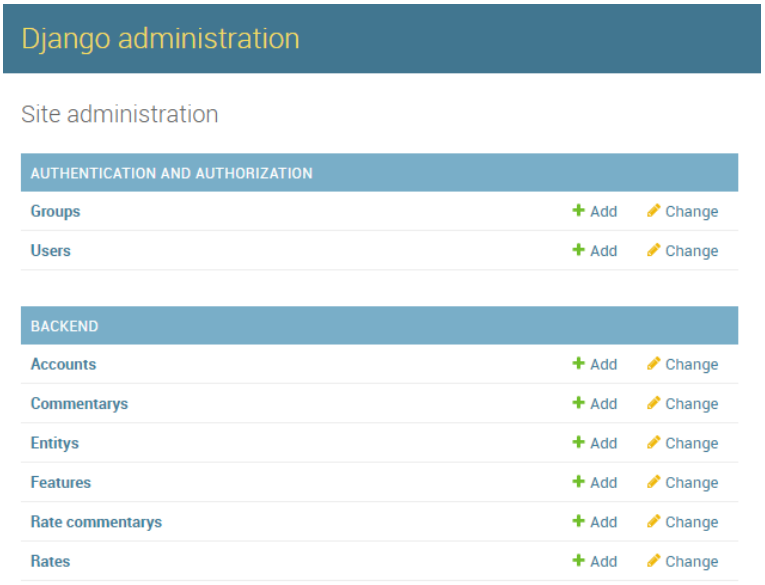


Figura 5.7: Administración de las tablas de la base de datos

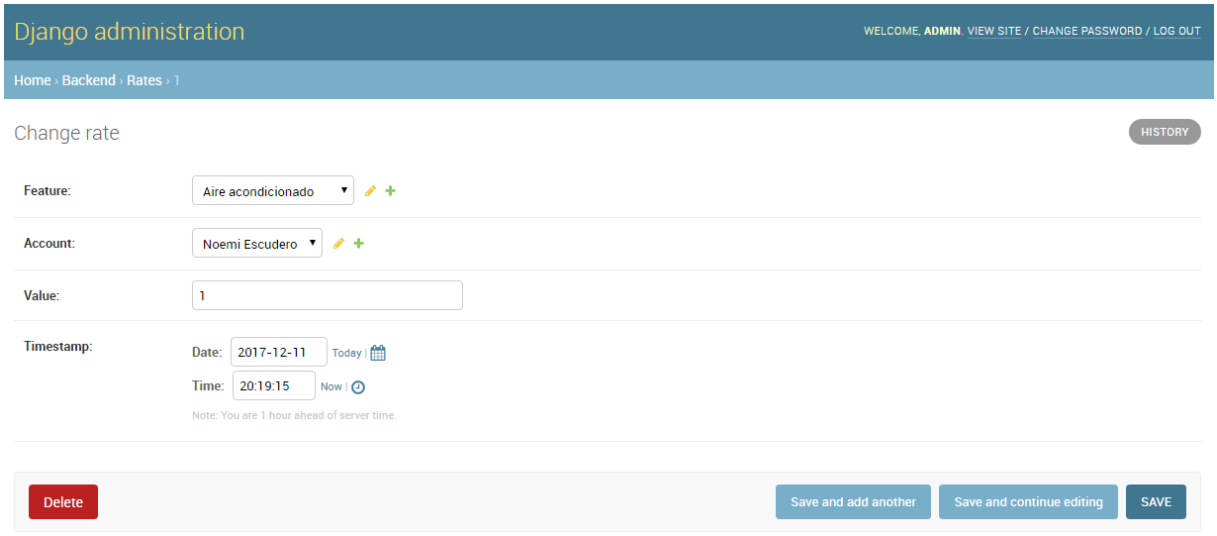


Figura 5.8: Administración de un elemento de la base de datos

Esta interfaz ha sido generada utilizando el paquete admin de Django.

5.1.4. Aplicación web de estadísticas

Finalmente, cuando se terminó la aplicación de Smartphone de GoTo que era la idea original del proyecto, se vio que las estadísticas sobre los datos se presentarían mucho mejor si el usuario accedía a una página web desde un navegador en un PC, ya que si se introducía directamente en la aplicación de móvil, en una pantalla pequeña sería más difícil visualizar las estadísticas.

Esta parte es la pantalla de visualización de estadísticas de las entidades creadas por el usuario. En el caso de GoTo, son lugares. En dicha pantalla el usuario tendría acceso a distintas estadísticas sobre la gente que visita, comenta y vota el perfil del lugar creado por él, y de esta forma saber si todo va bien o en qué mejorar. El diseño de la web que se ha creado posteriormente para mejorar la pantalla de estadísticas de GoTo ha sido el siguiente:

Pantalla de login

Figura 5.9: Wireframe de la pantalla de login de la web de estadísticas

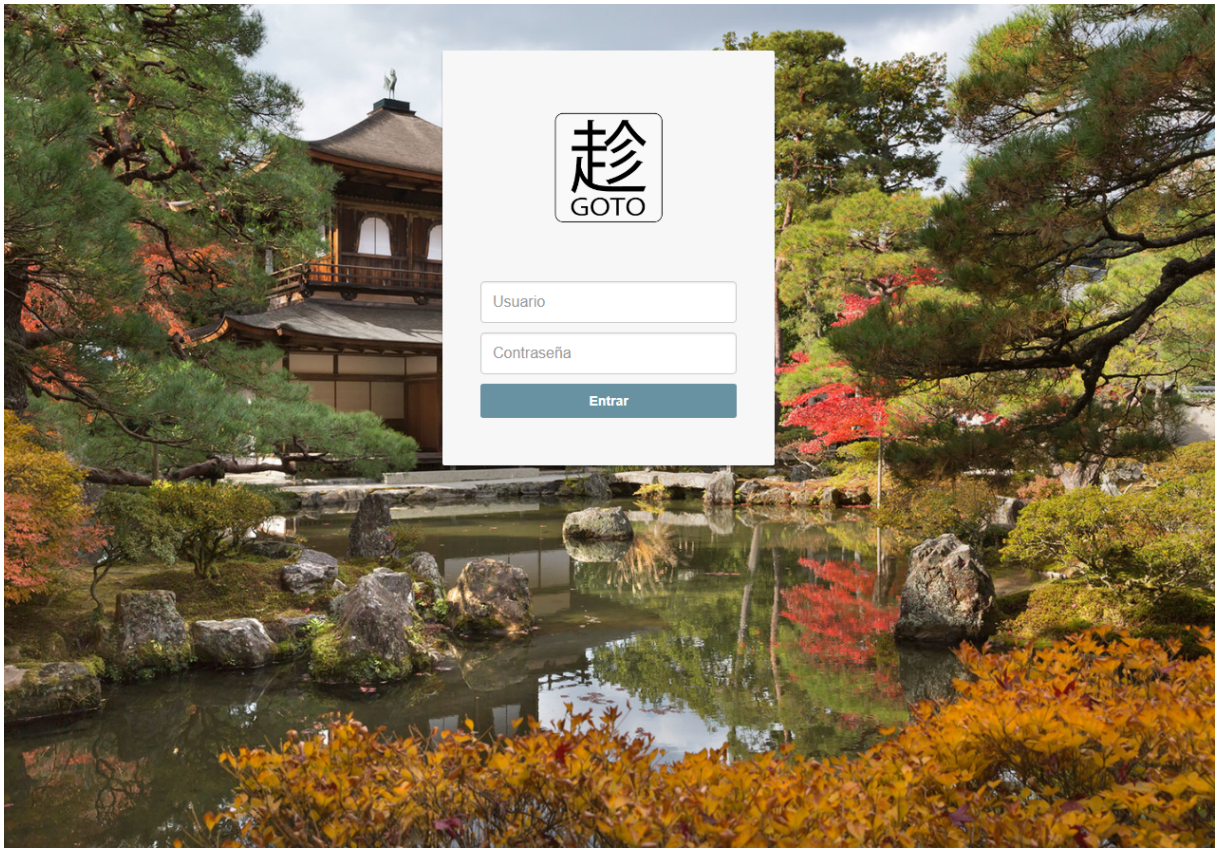


Figura 5.10: Pantalla login de la web de estadísticas

Esta pantalla se compone de un diseño sencillo, para que el usuario ponga las mismas credenciales que pone en la aplicación GoTo.

Pantalla de estadísticas

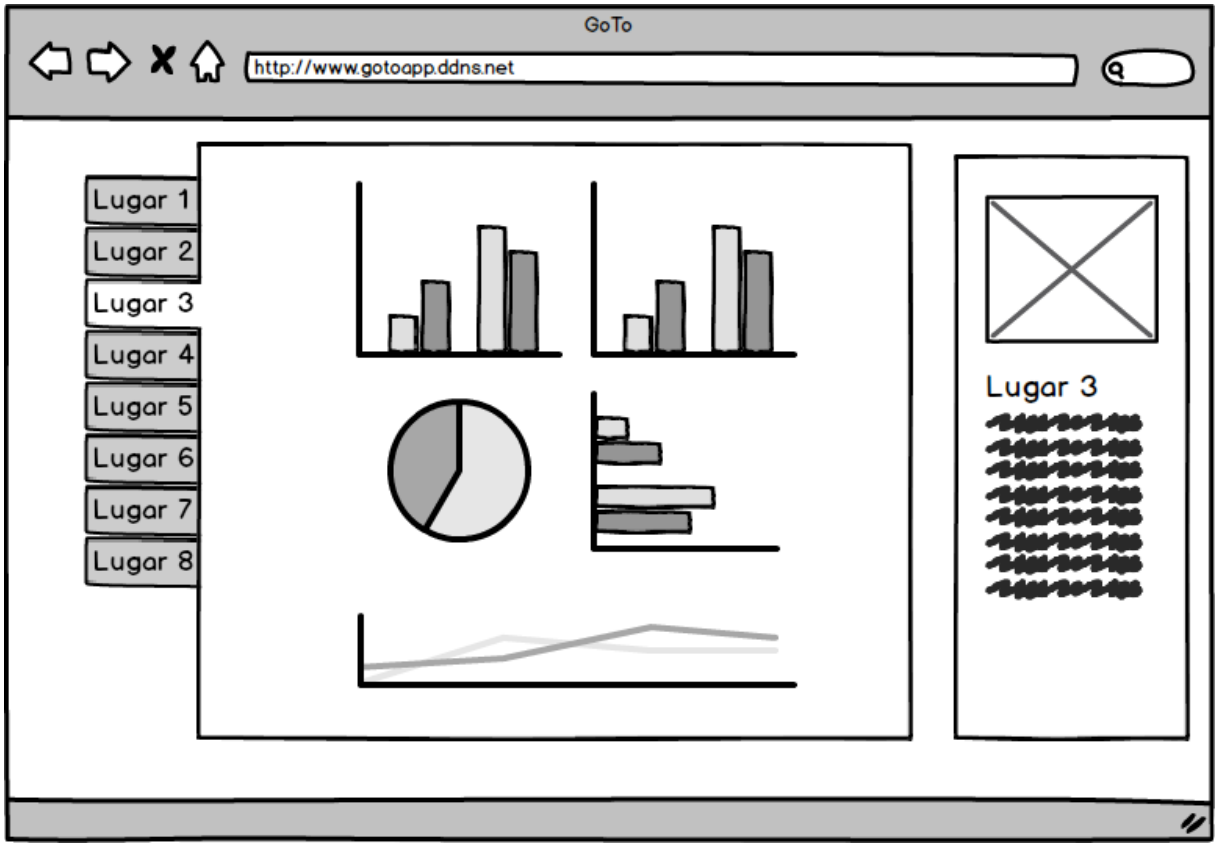


Figura 5.11: Wireframe de la pantalla de estadísticas en la web

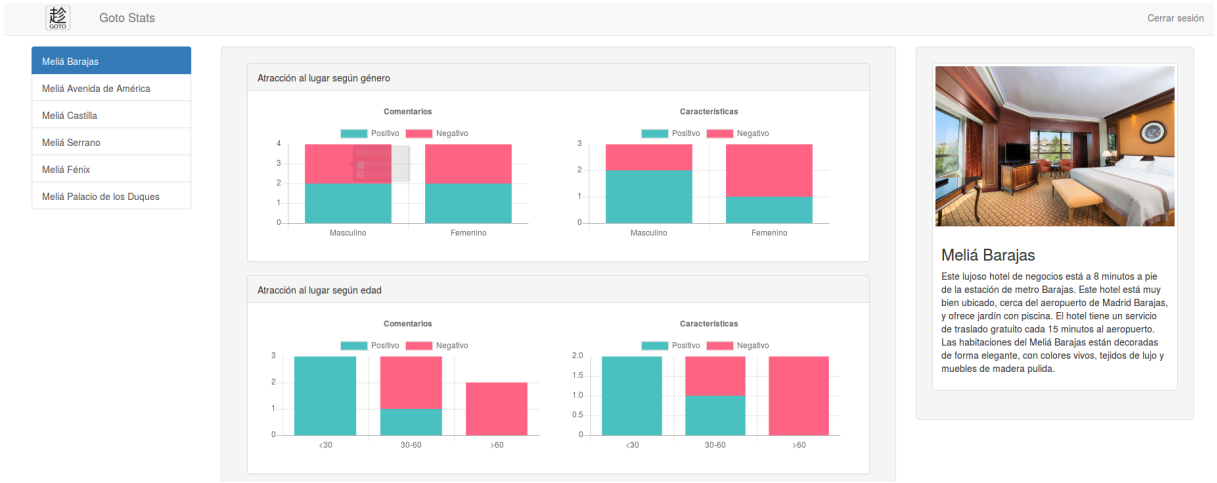


Figura 5.12: Pantalla de estadísticas en la web (Parte superior)

Figura 5.13: Pantalla de estadísticas en la web (Parte inferior)

En la pantalla de estadísticas, en el lateral izquierdo, el usuario podrá seleccionar la entidad de la que quiera consultar las estadísticas, y aparecerá solamente las que él ha creado.

En el centro se situarán varias gráficas interactivas de forma que con un simple vistazo el usuario pueda saber el rendimiento del perfil del lugar.

En el lateral derecho aparecerán datos informativos sobre el perfil de lugar, tales como la foto o la descripción.

En cuanto a la implementación, se ha codificado una aplicación web de django servida también por el mismo servidor responsable de la publicación del servicio REST. Está disponible a través del endpoint */stats*

Esta web hace uso de Twitter Bootstrap [30] para tener un aspecto moderno y depurado y de Chart.js [8] para la implementación de las gráficas interactivas. También se apoya en jQuery [17] para que estas librerías externas puedan funcionar con normalidad.

5.1.5. Desplegado del sistema

El sistema se ha desplegado sobre un sistema Linux basado en Debian. Las diferentes partes del backend se han desplegado de acuerdo al siguiente esquema:

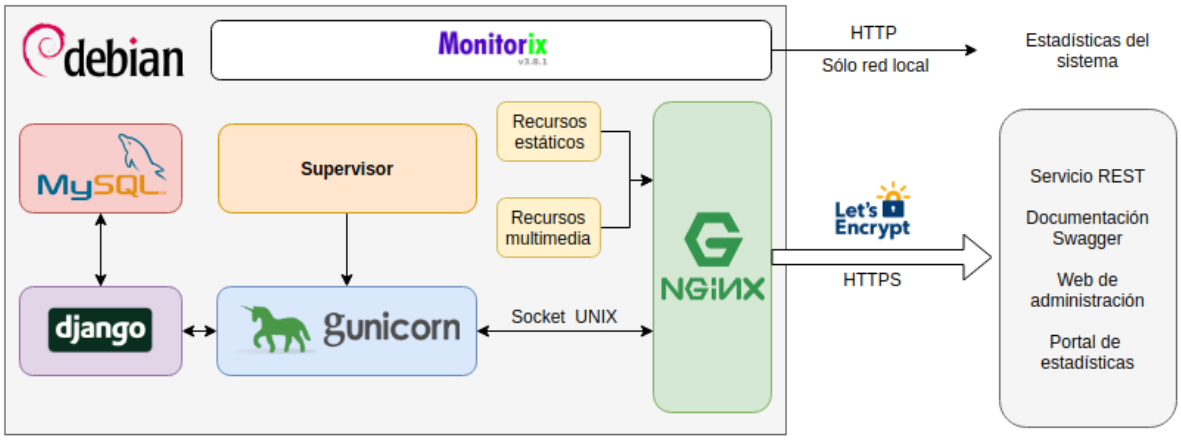


Figura 5.14: Esquema de los diferentes componentes del backend desplegado

- **Servidor gunicorn:** El servidor REST y la aplicación de visualización de estadísticas son aplicaciones Django servidas a través de un servidor Gunicorn [5] en modo solo local. Este servidor está mantenido por supervisor [10] , un programa escrito en python que actúa de watchdog para garantizar que el servicio está siempre disponible.

- **Servidor Nginx:** Se encarga de servir los ficheros estáticos, incluyendo los ficheros subidos por los usuarios. Este servidor también sirve de proxy para el acceso a Gunicorn. Nginx recibe las peticiones HTTPS de Internet y se las transmite a Gunicorn a través de un socket UNIX. Para poder ofrecer HTTPS a través de Nginx, se ha utilizado la herramienta acme-nginx [35] que configura automáticamente Nginx para obtener y utilizar un certificado de la iniciativa Let's Encrypt de la Linux Software Foundation [16].

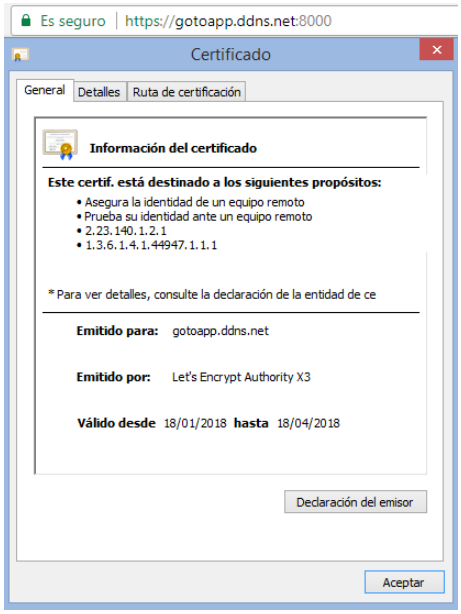


Figura 5.15: Certificado SSL

- **Monitorización:** Para la monitorización de todo el sistema, se ha instalado la herramienta Monitorix [23] que permite obtener estadísticas del rendimiento del equipo con el fin de solucionar y prevenir posibles problemas.

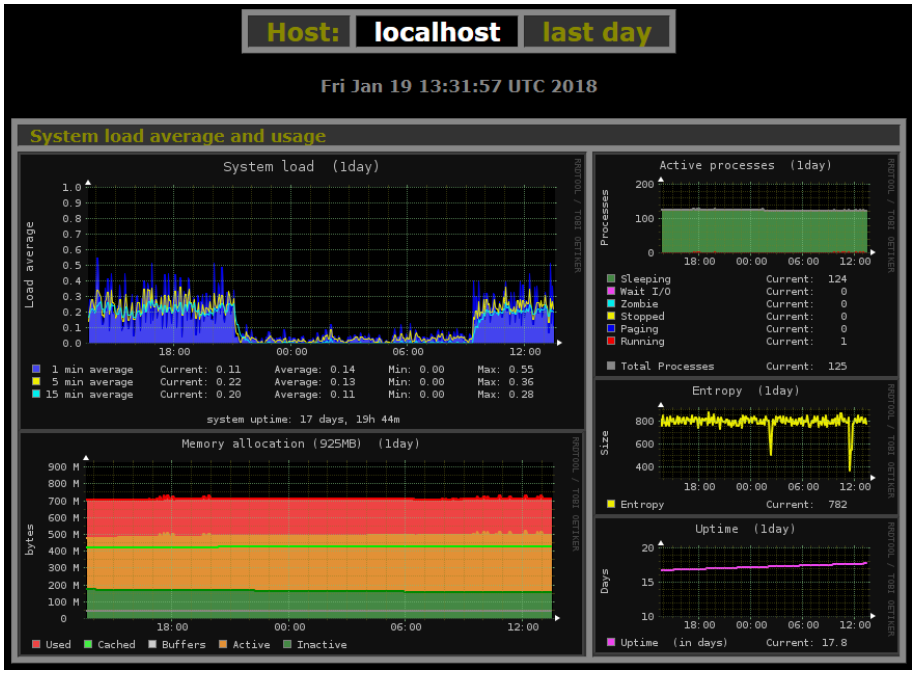


Figura 5.16: Visualización de la monitorización

- **Base de datos:** La base de datos durante el desarrollo de la aplicación ha sido SQLite [9], ya que es la base de datos por defecto que integra Django cuando se instala, es fácil de gestionar y permite portar la base de datos a otros equipos sin necesidad de instalar un gestor de base de datos. En producción se ha decidido utilizar la base de datos MySQL. El layout de base de datos es algo que no se ha diseñado explícitamente, ya que Django se encarga de crear las tablas y todas las estructuras que necesita para su funcionamiento a partir de los modelos definidos, por lo que podría integrarse cualquier tipo de base de datos compatible con Django.

5.2. Diseño de la aplicación de Android

En este apartado se va a tratar cada una de las fases en el diseño y la implementación de la aplicación Android del proyecto. En primer lugar se va a tratar el aspecto gráfico de la aplicación, haciendo hincapié en el desarrollo de la interfaz a través de herramientas de prototipado. Posteriormente se darán detalles sobre la implementación de esta interfaz en Android Studio, la arquitectura técnica de la aplicación y las librerías utilizadas en el desarrollo.

5.2.1. Diseño de la aplicación Android

El proceso de diseño de la interfaz que la aplicación tendría finalmente se ha dividido en tres fases diferenciadas:

- El diseño de la aplicación de móvil de GoTo en un primer momento se diseñó en papel. Esta primera fase, sirvió como herramienta para enumerar cada uno de los elementos que finalmente comprenderán la aplicación a partir de las necesidades que se querían cubrir desprendidas del análisis de mercado y los objetivos del proyecto. Esta fase también ayudó a definir la estructura de la base de datos y del backend.

- Después, con ayuda de la herramienta Software Balsamiq, se realizó una maqueta de mayor fidelidad, que sirvió entre otras cosas para mejorar la colocación de los elementos en la pantalla y refinar el estilo de la aplicación.

  Las diferentes maquetas realizadas de las pantallas de la aplicación, se utilizarán para comentar el diseño pormenorizado de cada pantalla en los siguientes apartados.

- Tras ello, se pasó a realizar una maqueta con ayuda del Android Studio, ya que el propio IDE trae consigo herramientas para realizar blueprints y configurar la interfaz y sus componentes. De esta forma se obtuvo una maqueta de la aplicación que funcionaba sobre un smartphone. Aunque lo datos que mostraba eran fijos, fue de gran utilidad para experimentar y mejorar la navegación entre las diferentes pantallas de la aplicación.

  De esta forma, se ahorraba tiempo ya que esta última maqueta es funcional, y únicamente quedaría darle la lógica a cada uno de los componentes. Esta parte se explicará más adelante en el apartado de implementación.

Una de las decisiones tomadas a raíz de la realización de estas maquetas, ha sido limitar la aplicación al formato vertical, ya que se llegó a la conclusión de que los elementos quedarían mejor colocados y podría presentarse mejor la información en forma de listas:

- Si el usuario hace uso del móvil en la calle y con una sola mano, el móvil es mucho más fácil de agarrar y consultar en forma vertical.

- Como GoTo es una aplicación de búsquedas, presentar los datos en forma de lista vertical ordenada es mucho más efectivo cuando el layout de la aplicación es también vertical, sobre todo a la hora de hacer scroll sobre las listas.

También se ha decidido seguir la guía de diseño de Material Design de Google, tanto en colores como en el aspecto de los controles. De la misma manera, se utilizan iconos estándar (como la flecha para volver atrás o las tres líneas horizontales para indicar menú) como metáforas visuales a lo largo de todo el diseño de la aplicación.

A continuación se presentan los distintos bocetos que se hicieron para seguir el diseño:

Pantalla de login

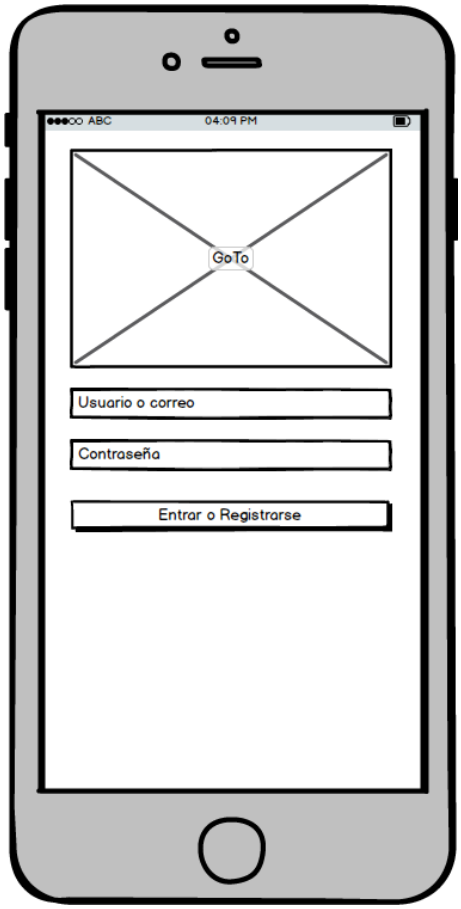


Figura 5.17: Wireframe de la pantalla de login

En la pantalla de login aparecen un botón para poder hacer login o registrar una nueva cuenta. Además de otros dos campos donde escribir el nombre de usuario y en otro la contraseña.

Pantalla de Búsqueda

Figura 5.18: Wireframe de la pantalla de búsqueda

La pieza central de esta pantalla es un mapa que ocupa la mitad de la misma. De esta manera se dota de importancia suficiente a la localización, que es el tema central de la aplicación. Debajo del mapa se pueden cambiar las opciones del buscador (por kilómetros y/o palabras clave), y debajo aparecen las fichas de los lugares relacionados con la búsqueda realizada.

Con los elementos de esta pantalla se busca una mejor interacción del usuario con la aplicación:

- Con el filtro por kilómetros se busca que el usuario pueda centrar la búsqueda en torno a él de una forma rápida y apoyada en el mapa.
- Con la búsqueda de palabras clave se quiere conseguir un funcionamiento más sencillo que a través de listas de categorías. Las palabras clave que el usuario coloque serán buscadas en el título y en la descripción de los lugares, de forma que el usuario no tiene que revisar largas listas de categorías. Lo negativo de esté método es que se pierde precisión en los resultados.
- Presentar los resultados de las búsquedas en un lista vertical hace muy natural el movimiento de scroll, tal y como se comentaba en apartados anteriores.

- Con la ficha de cada uno de los lugares se busca presentar al usuario la mayor cantidad de información en el menor espacio posible. Añadir una pequeña foto, el título del lugar y un pequeño fragmento de la descripción hacen que el usuario pueda identificar fácilmente lugares previamente conocidos así como hacerse una idea de en qué consisten los lugares desconocidos y de si pueden o no satisfacer sus necesidades.

Pantalla Crear/Editar un Usuario

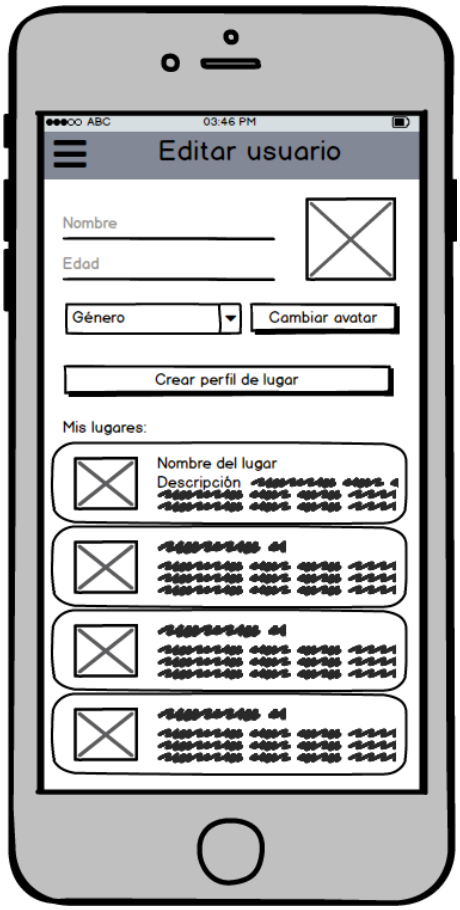


Figura 5.19: Wireframe de la pantalla de editar un usuario

En la pantalla de editar un usuario se puede editar el nombre, la edad, el género y el avatar. Además se provee de un acceso rápido a crear perfiles de lugar y debajo aparecen las fichas de lugares ya creados por el usuario con la misma estructura que en la pantalla anterior.

Cada vez que uno de estos campos sea modificado, dicho cambio será almacenado, de forma que no hace falta presionar en ningún botón de guardado y se evita perder cambios al presionar el botón de atrás del teléfono o al cambiar de pantalla por error. Un mensaje flotante aparecerá en la pantalla avisando al usuario de que sus cambios han sido almacenados.

Pantalla Crear/Editar un Lugar

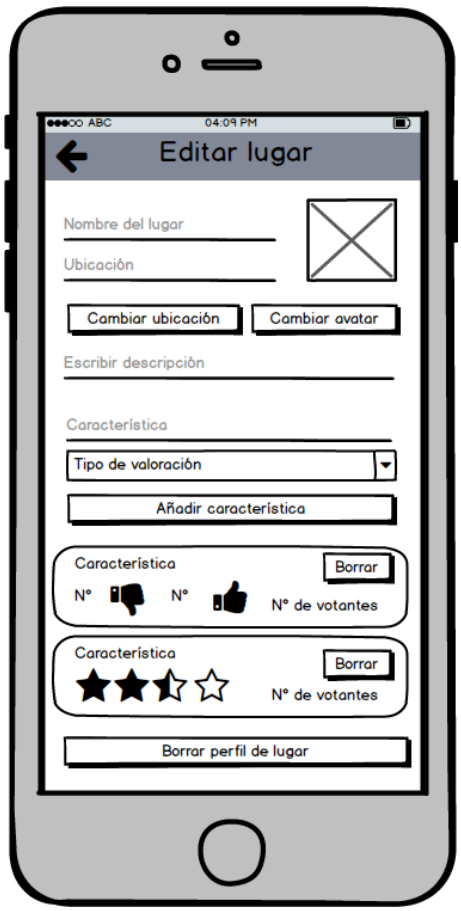


Figura 5.20: Wireframe de la pantalla de editar un lugar

Esta pantalla únicamente es accesible para el usuario creador del perfil del lugar y en ella se puede editar los distintos datos informativos acerca del lugar: su nombre, la ubicación, la foto, y la descripción.

El creador del perfil también puede añadir características que luego serán las que aparezcan en el perfil público del lugar y que el resto de la comunidad de GoTo puede valorar. A parte de poder añadir características, también podría borrarlas, sin embargo, no tiene acceso a borrar los comentarios del resto de usuarios para impedir que borre críticas negativas para que el resto de usuarios no las lean.

Como en las pantallas anteriores, si uno de los campos es modificado, los cambios son guardados automáticamente.

Pantalla de Perfil público de un Lugar

Figura 5.21: Wireframe de la pantalla de un perfil de lugar

La pantalla del perfil de lugar es similar a la editar, pero sin poder editar los campos. En ella se puede leer el nombre, la ubicación, la descripción y ver la foto del perfil. También se pueden leer las características del sitio y votarlas.

Por último el usuario podría participar añadiendo un comentario si alguna vez ha visitado el lugar y quiere compartir su opinión con la comunidad. Esto es positivo para animar a los usuarios a la participación dentro de los perfiles de los lugares de interés, ya que los votos son anónimos.

Los datos que se recogen de las votaciones de las características y los comentarios, serán los que aparezcan en forma de gráficos en la página web de estadísticas, donde se habrán anonimizado, y simplemente se utilizarán para hacer cálculos estadísticos.

5.2.2. Navegación por la Aplicación Android

Una vez presentadas las pantallas (actividades) que conforman GoTo, se expone a través de un esquema cómo es la navegación entre pantallas:

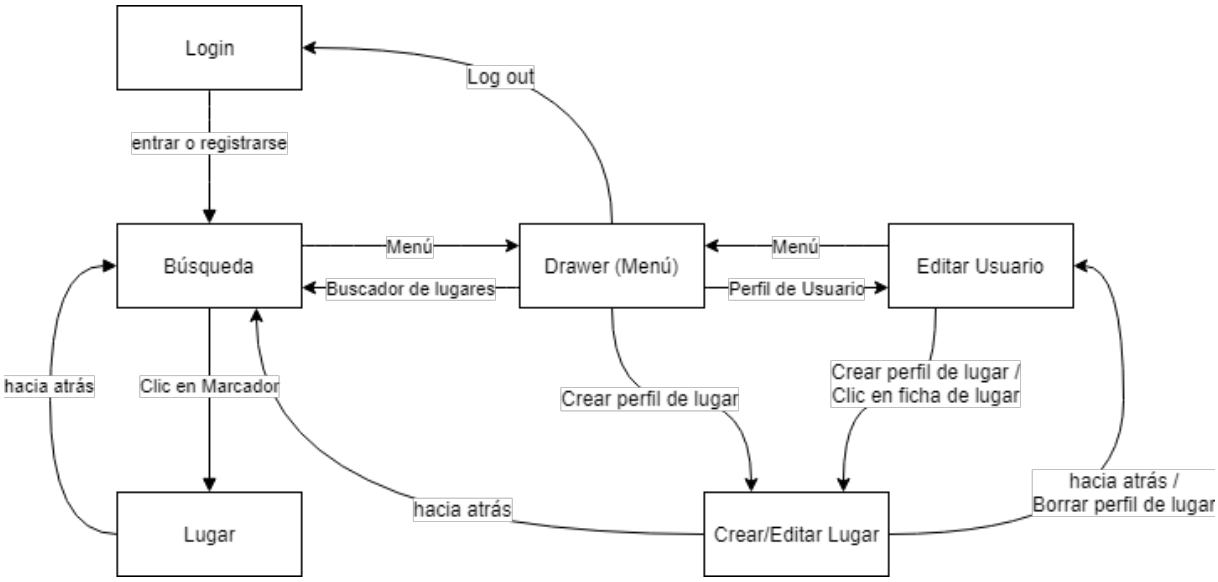


Figura 5.22: Esquema de la navegación por la aplicación

5.3. Implementación de la aplicación Android

A continuación se presentan los detalles sobre la implementación del diseño expuesto anteriormente. Para la programación de la interfaz, se ha seguido un método de desarrollo incremental.

- Se dividieron las tareas de programación en unidades funcionales que pudieran probarse por separado, de acuerdo a la división por subsistemas presentada en apartados anteriores.
- De esta forma, la programación de la aplicación avanzó progresivamente según se codificaban y se probaban cada una de las partes hasta completar todos los trabajos.
- Por último, se comprobó la cohesión entre las partes y que la navegación entre las pantallas fuera la correcta.

5.3.1. Entorno de desarrollo

La aplicación de Smartphone está escrita en Java, utilizando el IDE Android Studio, en un entorno Ubuntu Linux 16.04 LTS. Este entorno ha permitido utilizar dispositivos físicos y los emuladores que suministra Google para ejecutar la aplicación y comprobar su funcionamiento. Los dispositivos utilizados han sido:

- Google Nexus 5 Simulado (API 26)
- BQ Aquaris E5 HD (Android 5.0)
- Lenovo Tab3-710F (Android 5.0.1)
- Xiaomi MI 5 (Android 7.0)
- Xiaomi MI Max 2 (Android 7.1.1)

Una decisión importante previa al desarrollo de la aplicación Android ha sido elegir qué versión del API de Android utilizar durante el desarrollo. El propio Android Studio ofrece información sobre las diferentes funcionalidades añadidas en cada versión y el porcentaje de terminales compatibles:

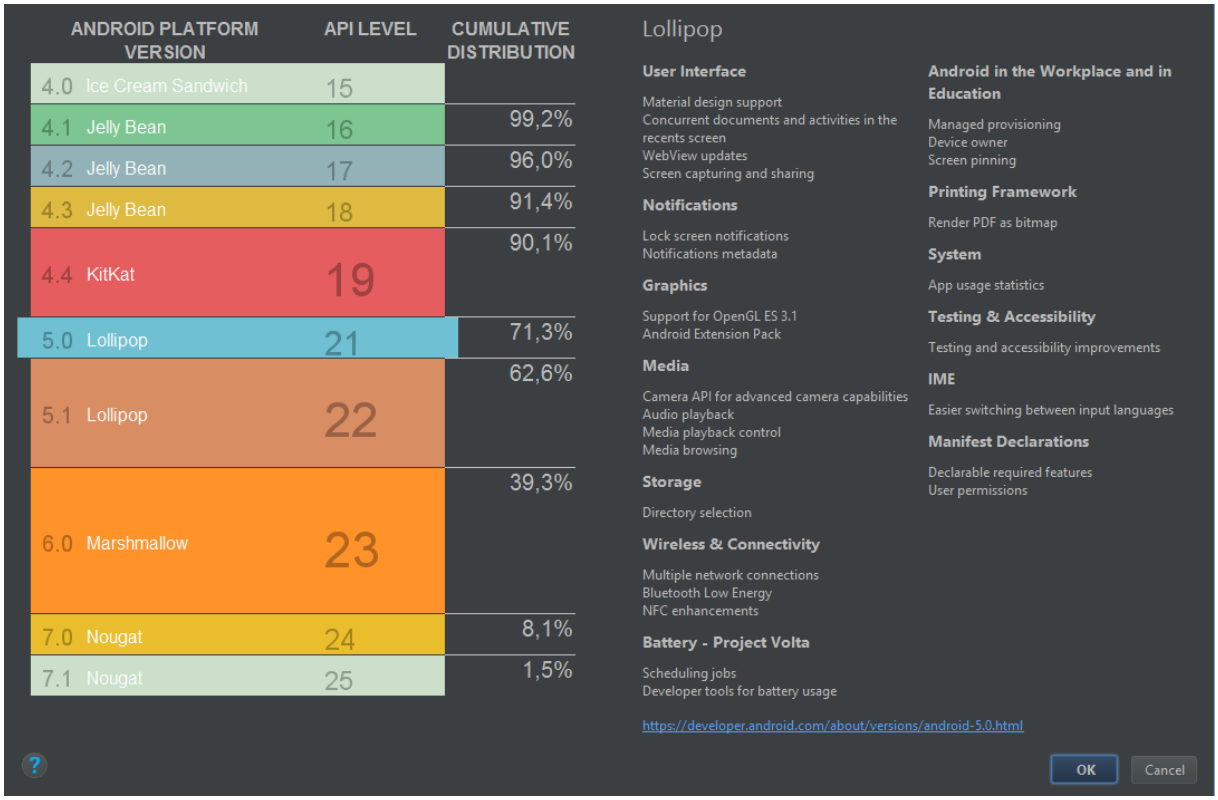


Figura 5.23: Información de Android Studio acerca del nivel de API y sus versiones

Finalmente se eligió utilizar la version 21 del API que da soporte a Smartphones con una versión de Android superior a 5.0, ya que ofrecía una alto porcentaje de compatibilidad y soporte para características como los mapas interactivos o la solicitud de permisos de aplicación individuales.

5.3.2. Arquitectura lógica de la aplicación

Desde el punto de vista de la programación, la aplicación está dividida en cuatro partes diferenciadas: modelo, vista, controlador y ficheros de configuración.

Modelo

Son clases escritas en Java que consumen el servicio REST expuesto por el backend. Estas clases están basadas en la librería externa Retrofit, que hace más sencilla la interacción con el REST. Estas clases están a su vez diferenciadas en dos grupos:

- **Objetos de datos:** Son estructuras construidas a semejanza de cada uno de los elementos de la base de datos que se exponen a través del servicio REST. Son utilizados como interfaz para manipular los datos o llamar a funciones remotas en el interior de la aplicación más fácilmente.

- **Controladores:** Son clases que se encargan de conectar con el servidor y generar todas las estructuras necesarias para consumir el servicio. Estas estructuras gestionan la comunicación para así mantener sincronizada la información entre los objetos de datos locales y el servidor.

```
52
53      @GET("/users/login/?format=json")
54      Call<User> login();
55
56      @FormUrlEncoded
57      @PATCH("account/{account_id}/?format=json")
58      Call<Account> updateAccountName(
59              @Path("account_id") int accountId, @Field("name") String name);
60
61      @FormUrlEncoded
62      @PATCH("account/{account_id}/?format=json")
63      Call<Account> updateAccountGenre(
64              @Path("account_id") int accountId, @Field("genre") String genre);
65
66      @FormUrlEncoded
67      @PATCH("account/{account_id}/?format=json")
68      Call<Account> updateAccountAge(
69              @Path("account_id") int accountId, @Field("age") Integer age);
70
71      @FormUrlEncoded
72      @POST("account/{account_id}/update_avatar/?format=json")
73      Call<Account> updateAvatar(
74              @Path("account_id") int accountId, @Field("image_data") String imageData);
```

Figura 5.24: Ejemplo de código donde se declara la comunicación con los endpoints del servidor

Vista

La vista de la aplicación está compuesta de diferentes ficheros XML que contienen la apariencia visual de la aplicación. Cada uno de los XML pertenece a una actividad, a un fragmento, o a elementos repetibles como diálogos.

La vista de la aplicación, desde un punto de vista técnico, se ha diseñado de acuerdo a dos principios básicos:

- Estos ficheros han sido diseñados para ser reutilizables e incluibles dinámicamente unos dentro de otros.
- La interfaz se basa en layouts predefinidos, cuyas dimensiones son calculadas en tiempo de ejecución para mantener su forma independientemente del tamaño de la pantalla.

Figura 5.25: Ejemplo del resultado de un XML en forma de blueprint y su diseño

Controlador

Se conforma de las actividades y fragmentos. Las actividades forman cada una de las pantallas de la aplicación y los fragmentos codifican componentes reutilizables de la aplicación que pueden ser incluidos dinámicamente en tiempo de ejecución dentro de actividades u otros fragmentos, como por ejemplo cada una de las fichas de resultados de una búsqueda.

```
303     public void loadData(){
304         //Cambiar el nombre del lugar
305         EditText nombre = (EditText) findViewById(R.id.nombreLugar);
306         nombre.setHint(myEntity.name);
307         //Cambiar la descripción
308         EditText descripcion = (EditText) findViewById(R.id.descripcion);
309         descripcion.setHint(myEntity.description);
310         descripcion.setTextAlignment(View.TEXT_ALIGNMENT_TEXT_START);
311         //Cambiar photo
312         ImageView photo = (ImageView) findViewById(R.id.photo);
313         new DownloadImageTask(photo).execute(myEntity.photo);
314         //Cambiar la ubicación
315         EditText ubicacion = (EditText) findViewById(R.id.ubicacion);
316         Geocoder geocoder = new Geocoder( context: this, Locale.getDefault());
317         String result = null;
318         //Poner nombre de calle y localidad obteniendolo desde las coordenadas
319         try {
320             List<Address> addresses = geocoder.getFromLocation(myEntity.latitude, myEntity.longitude,  maxResults: 1);
321             if (addresses != null && addresses.size() > 0) {
322                 Address address = addresses.get(0);
323                 result = address.getAddressLine( index: 0) + ", " + address.getLocality();
324                 ubicacion.setHint(result);
325             }
326         }catch (IOException exception){
327             Log.d( tag: "ErrorGeocoder", exception.getMessage());
328         }
329     }
```

Figura 5.26: Ejemplo de código donde se cargan los elementos necesarios para mostrar el perfil de un lugar

Ficheros de configuración

Ha sido necesario modificar ciertos ficheros de configuración del sistema de generación de Android (Gradle) para, por ejemplo, colocar los permisos que la aplicación debe tener para poder funcionar, API Keys para el servicio de Google Maps o la inclusión de librerías externas.

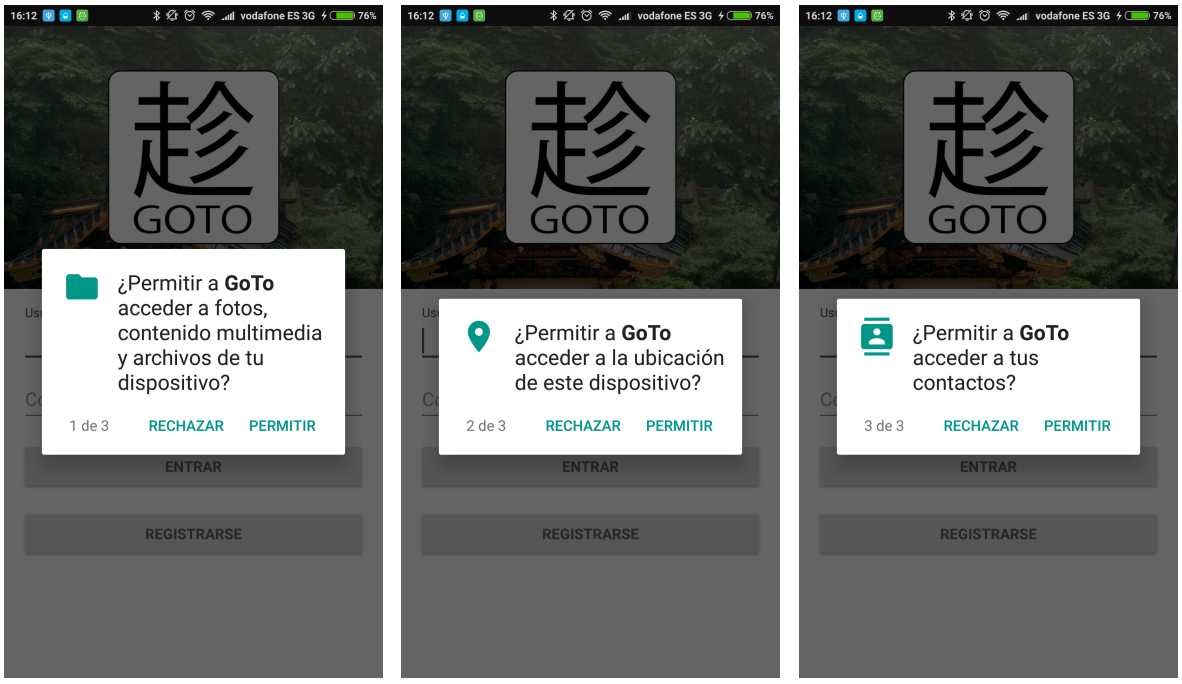
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.escudero.noemi.agoto">

    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

Figura 5.27: Ejemplo de código donde se declaran los permisos que requiere la aplicación

5.3.3. Funcionamiento

A continuación se va a detallar el funcionamiento interno de cada una de las diferentes pantallas de las que dispone la aplicación:

Pantalla de permisos



(a) Permiso de acceso a multimedia (b) Permiso de acceso a GPS (c) Permiso de acceso a contactos

Figura 5.28: Distintos permisos que pide GoTo

La aplicación GoTo necesita poseer ciertos permisos para poder funcionar correctamente:

- Acceso a la ubicación GPS debido al uso continuo que hace del API de Google Maps.
- Acceso al almacenamiento interno para acceder a la galería y subir fotos como avatares de lugares y usuarios.
- Acceso a los contactos para el autocompletado del login, si el teléfono es compatible.

Para ello, nada más instalar el APK de GoTo, y si el dispositivo tiene una versión de Android superior a 6.0, lo primero que se puede observar es un pop-up solicitando los permisos necesarios.

Si la versión de android es inferior a 6.0, los permisos se solicitarán durante la instalación de la aplicación.

Pantalla de login

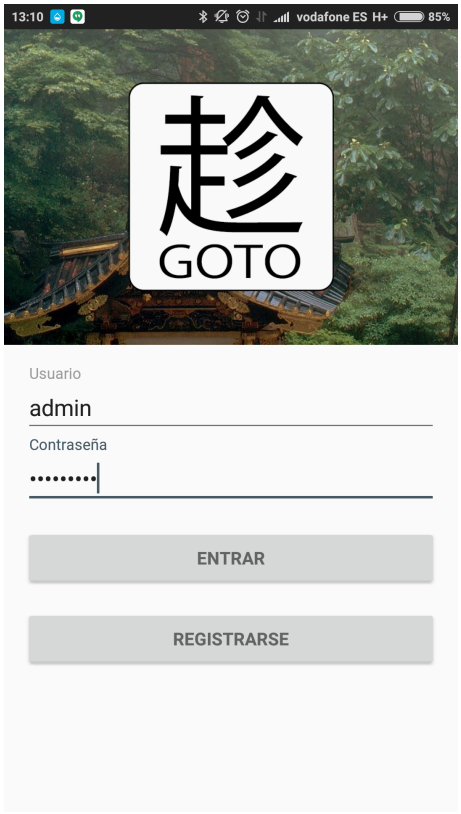


Figura 5.29: Pantalla de Login

Ésta pantalla es la primera que se ve siempre que se inicia la aplicación. El usuario puede optar por dos acciones diferenciadas: *entrar* y *registrarse*.

Si pulsa en *registrarse*, el sistema utiliza una petición al servicio para crear una nueva cuenta. Se mostrará un error si el nombre de usuario ya existe previamente en la base de datos. Tras ello, se sigue el mismo procedimiento de login que se explica a continuación.

Si pulsa en *entrar* el sistema realizará el login en el servicio y dará paso al usuario a la pantalla de búsqueda.

Se muestran errores en caso de usuario o contraseña incorrecta o de error en la conexión del dispositivo con Internet o con el servidor.

Pantalla de Menú

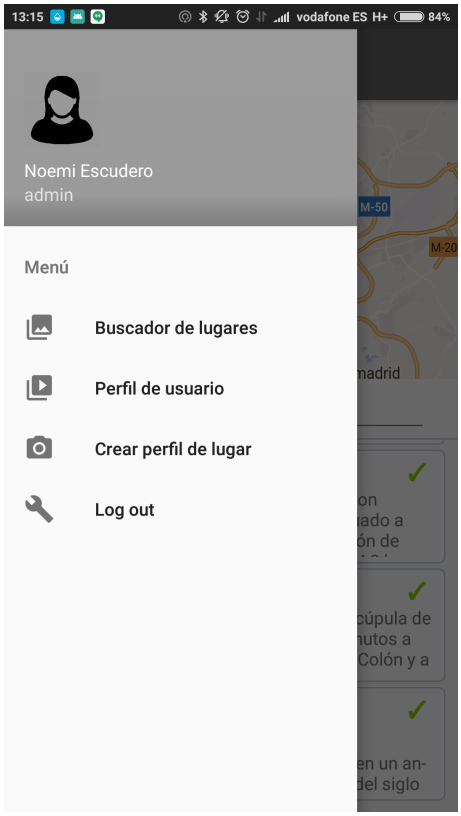


Figura 5.30: Drawer de Menu

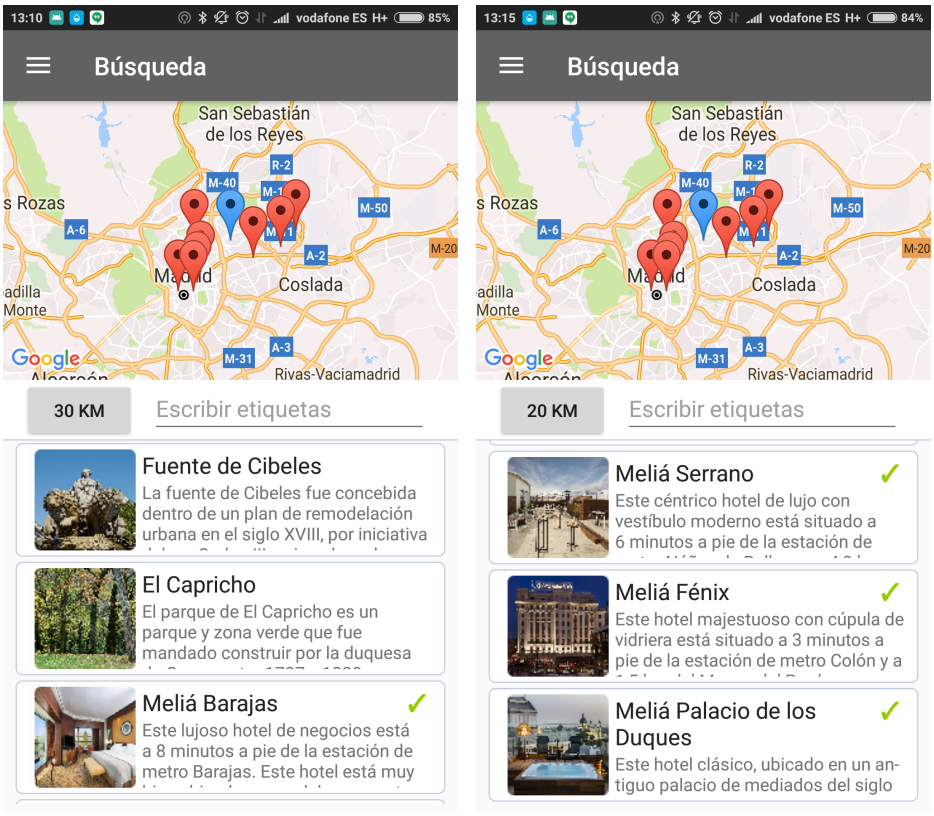
El menú consta de un deslizable (drawer) que aparece al arrastrar desde el lateral izquierdo o pulsando el botón de menú. Ocupa un porcentaje de la pantalla, sombreando en el fondo la actividad desde la que se abrió y es común a la mayoría de las actividades.

En la parte de arriba se podrá echar un rápido vistazo al avatar que actualmente tiene el dueño de la cuenta, y debajo podrá ver el nombre que tiene puesto en su perfil, y el nombre de usuario con el que ha accedido a la cuenta.

Por lo tanto, es una forma sencilla, de que si una persona tiene varias cuentas, pueda ver con cuál ha accedido en cada momento.

De la misma manera hay también accesos directos a: el buscador de lugares, su perfil personal, una actividad donde crear un perfil de lugar nuevo, y a la página de log in, en el caso de que haga clic en *logout*.

Pantalla de Búsqueda de lugares



(a) Ejemplo de búsqueda 1 (b) Ejemplo de búsqueda 2

Figura 5.31: Pantalla de Búsqueda

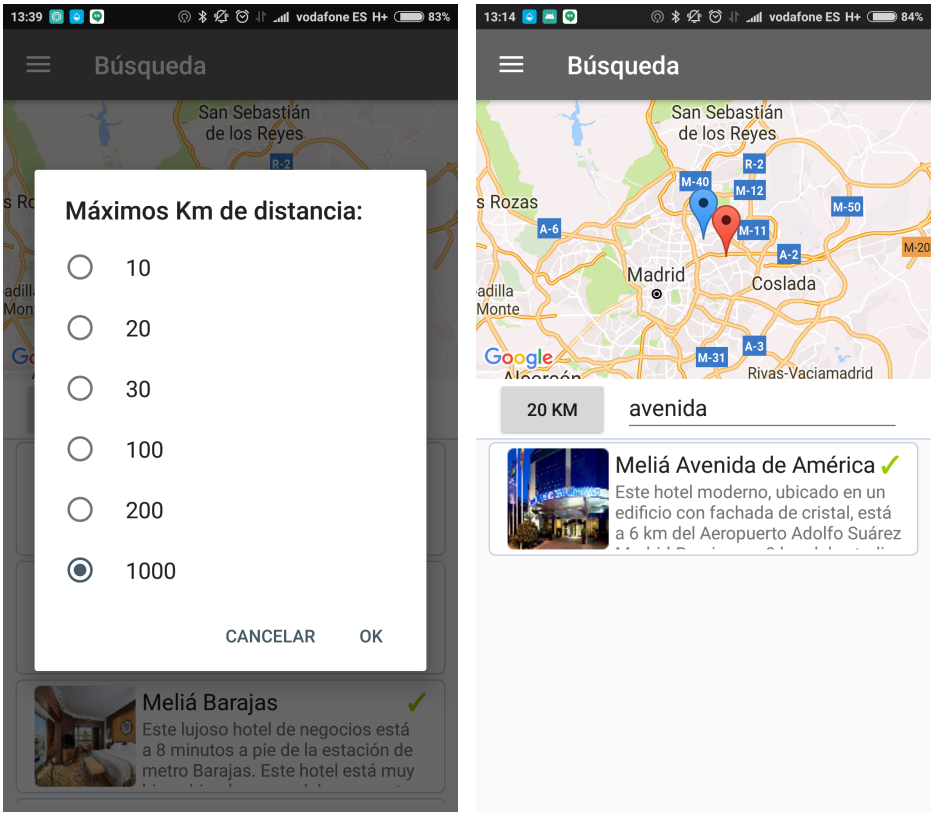
Esta pantalla es la primera que ve el usuario, cuando ha metido correctamente sus credenciales.

La pieza central de la pantalla es un control de Google Maps. En el caso de disponer de una ubicación válida, el mapa se centrará automáticamente. Esta ubicación es obtenida a través del GPS y/o de la red del dispositivo.

Para el selector de distancia en kilómetros, se ha utilizado el algoritmo de distancia de Vincenty [43] para determinar un área circular alrededor de las coordenadas proporcionadas por el GPS del móvil, y de esta forma buscar en la base de datos los lugares que se encuentren dentro de los límites establecidos.

También, si lo prefiere el usuario, puede escribir etiquetas dentro del recuadro de escritura, por ejemplo *bar, Madrid* y éstas etiquetas filtraran a su vez la búsqueda realizada por la distancia. Estas palabras serán incluidas en una búsqueda dentro de los nombres de los perfiles de lugar y en sus descripciones.

Ambas búsquedas son lanzadas al detectar algún cambio en el valor de los filtros, y los cálculos son realizados en el servidor.



(a) Filtro de distancia (b) Búsqueda con filtro de palabras clave

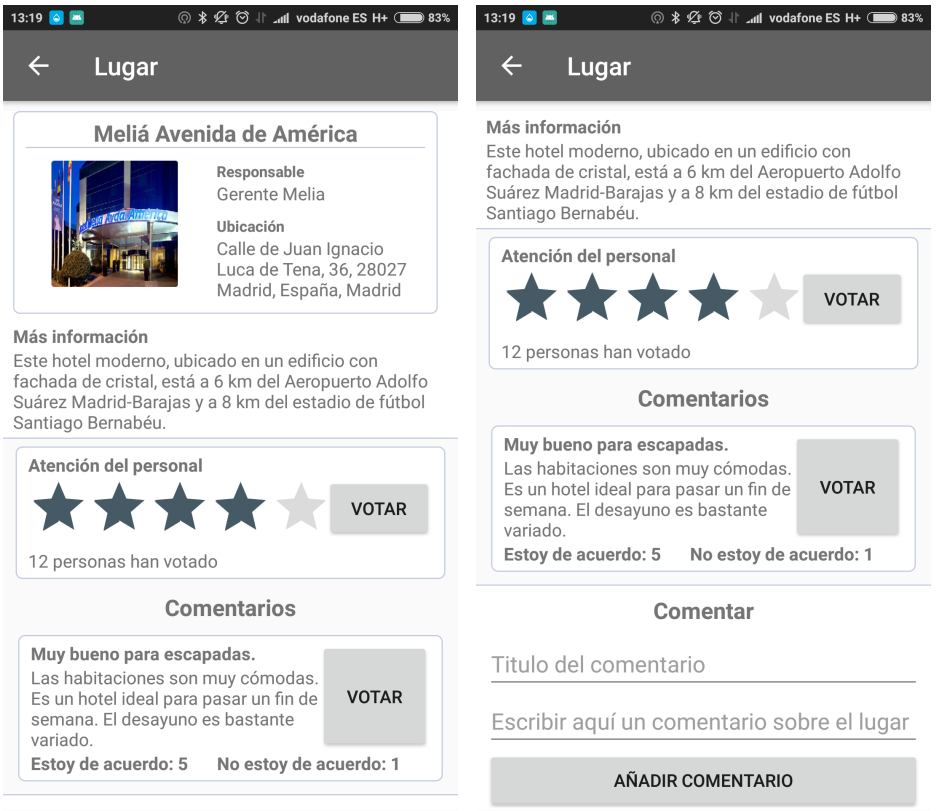
Figura 5.32: Pantalla de Búsqueda usando sus filtros

Debajo del apartado de filtros, aparecerán el resultado de la búsqueda. En caso de que no haya coincidencias, se avisará de ello. De la misma manera, un icono rotativo aparecerá en esta zona mientras se esperan los resultados del servidor.

En caso de haber resultados, aparecerán las fichas de los resultados de la búsqueda. Si el usuario hace clic en una de esta fichas, el mapa automáticamente se centrará en él para poder ver qué más lugares de interés hay alrededor suyo.

El usuario puede hacer clic encima del marcador del mapa que desee, y esta acción le llevará a visitar temporalmente la página de perfil de dicho lugar, ya que si hace clic en el icono de la flecha, volverá de nuevo a la página de búsqueda.

Pantalla de perfil de lugar



(a) Pantalla de perfil de lugar (Parte superior)

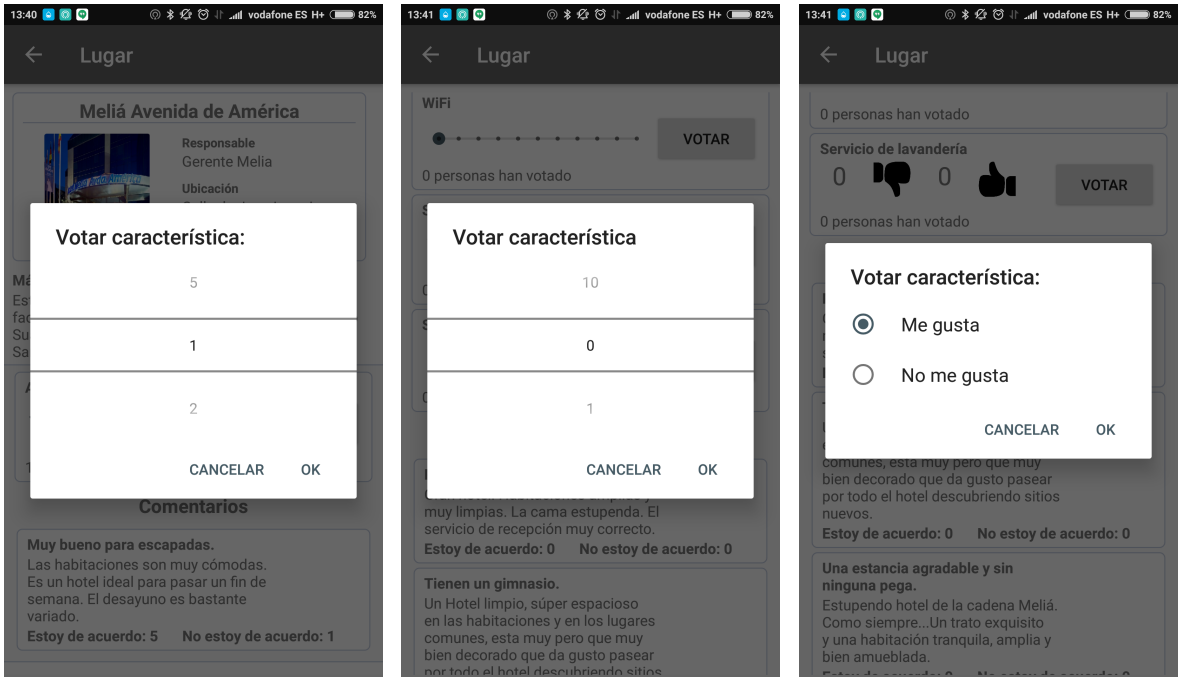
(b) Pantalla de perfil de lugar (Parte inferior)

Figura 5.33: Pantalla de Perfil de Lugar

Esta pantalla es la que se muestra cuando el usuario hace clic en un marcador del mapa.

Se presenta una pequeña ficha de información acerca del lugar. La ubicación es transformada a lenguaje natural utilizando los servicios de localización de Google Maps.

Debajo de la ficha de información aparece un apartado que contiene todas las características que el dueño del perfil creó para que el resto de usuarios pudiera votar. Cada una de estas características está codificada como un fragmento que se muestra de forma diferente de acuerdo al tipo de la característica. Estos fragmentos también gestionan las votaciones de los usuarios a través de pop-ups que permiten publicar puntuaciones que serán enviadas al servidor y aplicadas al valor general de la característica:



(a) Votar característica de tipo estrellas (b) Votar característica de tipo nota (c) Votar característica de tipo me gusta

Figura 5.34: Pantalla para votar característica

Debajo de las características, aparecerán los comentarios del resto de usuarios. El comentario también puede votarse positiva o negativamente. De igual forma que con las características, si el usuario hace clic en el botón de *votar*, se le abrirá un selector de opción y tras enviar su voto verá actualizado el recuento de votos en dicho comentario.

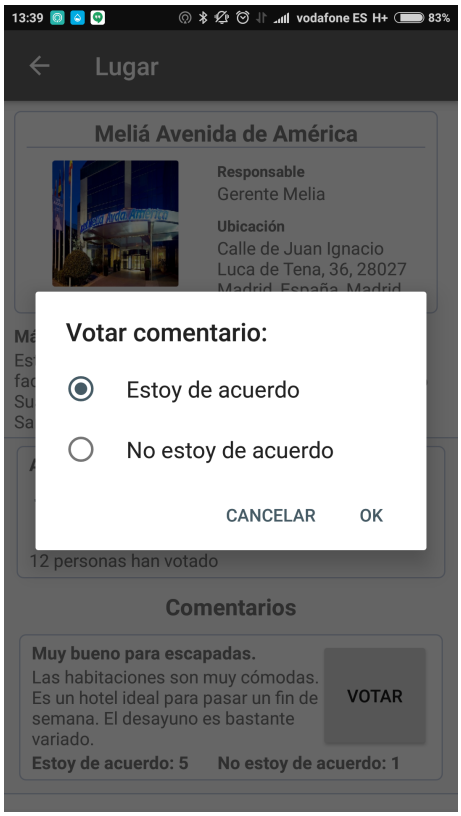
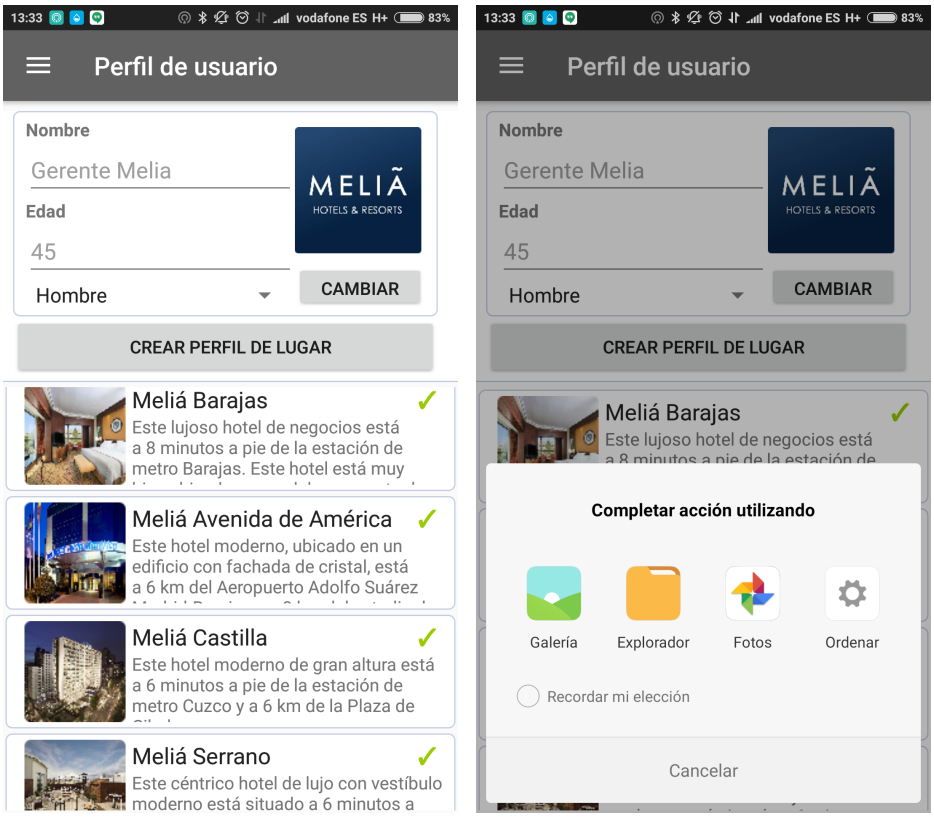


Figura 5.35: Pantalla para votar un comentario

Por último se da la posibilidad al usuario de publicar un nuevo comentario.

Pantalla de perfil de usuario



(a) Perfil de Usuario (b) Acceso a multimedia del teléfono

Figura 5.36: Pantalla de Perfil de Usuario y de Cambiar Avatar

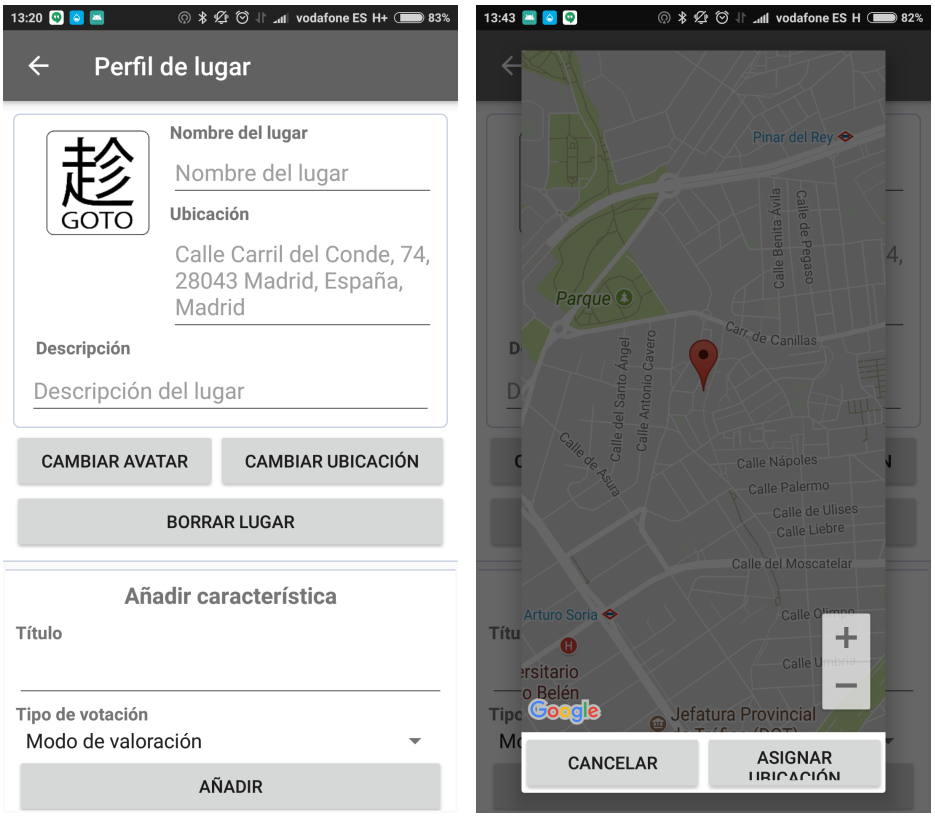
Esta pantalla presenta al usuario una pequeña ficha de información sobre él mismo. En el caso de que el perfil sea nuevo y sea la primera vez que acceda a ésta página, verá información por defecto, que deberá rellenar con los datos que desee. Todos los campos son actualizados en el servidor al ser cambiados en los controles.

También podrá cambiar la foto de su perfil, y haciendo clic en *cambiar avatar* le aparecerá la opción de subir una foto desde la galería de su telefóno.

El botón *crear perfil de lugar* le llevará a la pantalla de crear perfil, igual que accediendo a través del menú.

Por último, se presenta un listado de todos los perfiles de lugar que ha creado previamente. Este listado es muy parecido al listado de resultados que aparece tras la realización de una búsqueda en la actividad de búsquedas ya que el fragmento es en este caso reutilizado. Tras hacer clic en un resultado, se envía al usuario a la página de edición de lugar.

Pantalla editar/crear perfil de lugar



(a) Perfil de Lugar nuevo (b) Cambiar ubicación del Lugar

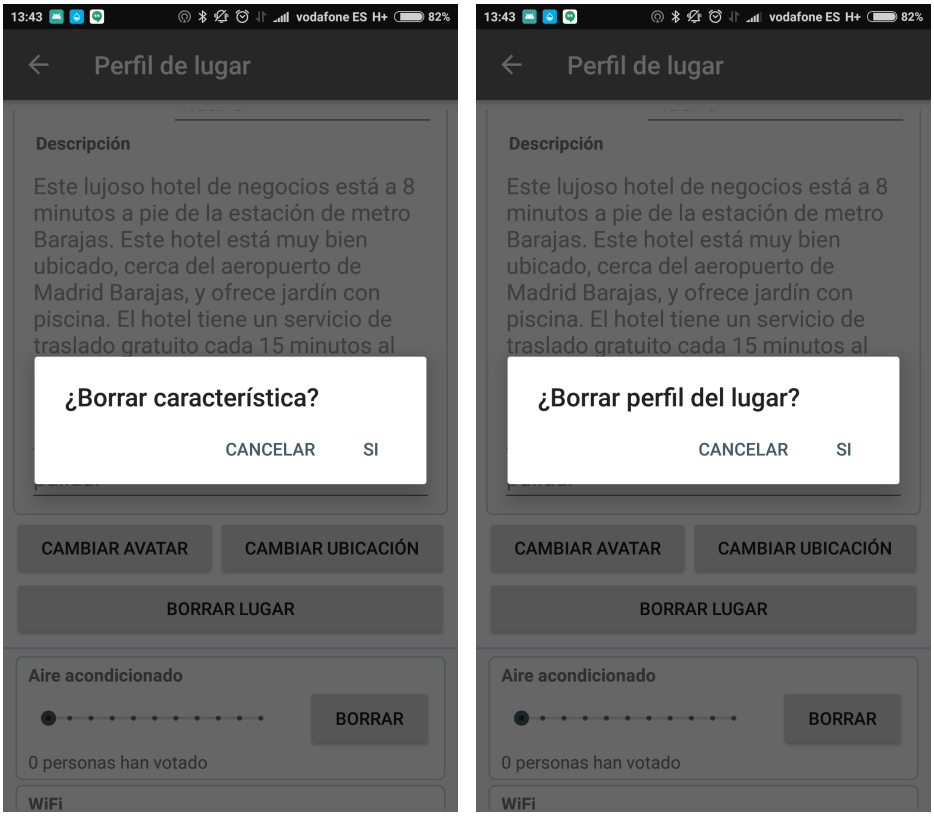
Figura 5.37: Pantalla de Crear Lugar y de Cambiar Ubicación

Esta pantalla es la que aparece cuando se quiere crear un perfil de lugar o cuando se quiere editar. En el caso de la creación, aparece con valores por defecto. En ambas situaciones, se está utilizando la misma actividad.

También se muestra ver una pequeña ficha de información del lugar, como su foto, su nombre y su ubicación. Y todo ello puede ser modificado en cualquier momento.

Para cambiar la foto, hay que hacer clic en el botón de *cambiar avatar*, que directamente abrirá la galería del móvil para subir una nueva foto. Si el usuario hace clic en *cambiar ubicación* le aparecerá un pop-up que contendrá un mapa de Google Maps, donde podrá cambiar de posición el marcador (por defecto aparecerá en la ubicación que tiene en ese momento el usuario si está disponible).

Más abajo se presenta un apartado donde el administrador del lugar puede crear características de diferente tipo. Deberá rellenar el nombre de la característica, el tipo y hacer clic en *añadir*. También tendrá acceso a las características que añadió previamente y tendrá la opción de borrarlas. Sin embargo, deberá tener en cuenta que borrando la característica se perderán todas las votaciones hechas sobre ella por el resto de usuarios.



(a) Acción de borrar característica (b) Acción de borrar perfil de lugar

Figura 5.38: Pantallas de Borrar Característica y Perfil de Lugar

Y por último tendrá la opción de borrar el perfil completo del lugar, haciendo clic en el botón *borrar lugar*. Tras hacer click, un pop-up de confirmación aparecerá y si procede, será redireccionado a su página de perfil personal habiendo borrado el perfil del lugar.

5.4. Herramientas adicionales

En el desarrollo del proyecto, ha sido necesario el uso de algunas herramientas adicionales:

- El desarrollo del proyecto se ha emplazado en un sistema de repositorios git llamado GitLab. Gitlab es muy útil en casos donde se quiere un servidor doméstico y personal, ya que ofrece la posibilidad de instalar el servidor completo en un ordenador personal y utilizarlo. De esta manera se evitaría subir el código a servidores desconocidos en la nube.
- A la hora de desarrollar también se ha tenido que solicitar una API Key para poder usar el servicio de localización de Google Maps. A parte de ser utilizado dicho servicio para mostrar mapas, también se ha usado para traducir las coordenadas de latitud y longitud a lenguaje natural. De esta forma, se hace más sencillo mostrar al usuario una búsqueda.
- Como el sistema está desplegado en un servidor que se encuentra tras una IP dinámica, ha sido necesario configurar un sistema de DNS que permita incluir un nombre de dominio fijo en la aplicación. Para ello se utiliza el servicio de no-ip, que permite actualizar la IP a la que se refiere el nombre de dominio elegido automáticamente.
- Para la creación de esta documentación, se ha empleado LaTeX [25].

6

Conclusiones y trabajo futuro

6.1. Conclusiones

Durante la realización del proyecto, se han conseguido en gran medida los distintos objetivos que se propusieron al inicio. Se ha desarrollado un sistema completo, que cumple con los requisitos y la funcionalidad esperados, destacando de entre ellos la generalidad del sistema:

- Se ha presentado un Framework, construido con la intención de que sirva a la comunidad de desarrolladores de código libre para crear sus propias aplicaciones. De esta manera, otros programadores tendrán una base genérica sobre la que partir para generar y desplegar rápidamente distintos entornos de recomendación sobre diferentes conceptos independientemente de la naturaleza de estos.
- Se ha realizado una aplicación para Android que representa un caso de uso del Framework, y que además es una idea de cómo se podría explotar la recolección de datos de los usuarios para generar estadísticas, además de dar un servicio a los usuarios de recomendación de lugares cercanos e interesantes, si tiene en cuenta las opiniones y valoraciones del resto de usuarios.
- Finalmente se añadió una web donde los usuarios pueden consultar diferentes estadísticas.

En este trabajo se ha abordado un problema/necesidad real y se ha ideado y desarrollado una solución desde principio a fin, pasando por todas las fases del desarrollo de Software, y poniendo en práctica el conocimiento adquirido en las distintas asignaturas del Máster en Ingeniería Informática, especialmente Diseño de Sistemas Interactivos gracias a la cual se practicó la realización de maquetas y prototipos centrados en el usuario y Gestión y Dirección de Proyectos Tecnológicos, la cual ha servido para conocer técnicas de organización que se han puesto en práctica en el tiempo de realización de este proyecto.

Durante el desarrollo de las diferentes partes del proyecto, se han tenido que ir incorporando cambios y mejoras al diseño por lo que también se ha practicado el hecho de volver a iterar el ciclo de diseño y desarrollo de forma ágil, para incorporar algo no planificado al inicio del proyecto, pero que al final se vió necesario incluir para dar un mayor valor.

Se ha avanzado también mucho en el aprendizaje de las tecnologías propuestas para el desarrollo de este proyecto. Tanto con Django y REST Framework como con Android Studio y su SDK asociado, no se tenía ninguna experiencia previa, y tras el desarrollo del backend y de la aplicación para Android, se ha acabado con un buen dominio de estas tecnologías y una sólida base para seguir aprendiendo sobre estos temas en el futuro, ya que dichas tecnologías son muy buscadas en las empresas y proporcionan una gran ventaja en el mercado laboral.

Por lo tanto, no solo se han aprendido nuevas tecnologías, sino a trabajar pensando en programar un código limpio y bien estructurado de cara a que lo usen otros desarrolladores y reanudar el diseño de una nueva parte tras ya estar en la fase final del proyecto.

6.2. Trabajo futuro

De cara al futuro, se presentan una serie de actividades que complementarán el trabajo realizado y presentado en este informe. A continuación se exponen unos pasos que se deberían tomar de cara a un uso público y una utilización libre:

- **Autenticación y permisos:** Para poder completar la seguridad del registro de usuarios en la aplicación Android y en la web, se debería utilizar una herramienta más completa que la securización estándar de Django: utilizar el sistema OAuth, que no solo aporta mayor seguridad, si no que también es un estándar libre utilizado por grandes compañías como Google, Twitter, Github, etc. que permitiría una mejor integración del servicio. También se podría mejorar el sistema de permisos actual del backend y mejorarlo para hacer uso del sistema de grupos y permisos que contiene Django, con el fin de hacerlo más flexible para los desarrolladores que quieran usarlo.

- **Mejoras de desplegado:** De cara a manejar un gran número de peticiones, sería muy util encapsular el sistema en *boxes* de vagrant o en contenedores de docker para facilitar y automatizar su desplegado no solo en servidores comunes, sino también en las diferentes plataformas de computación en la nube.

- **Más estadísticas:** Las estadísticas mostradas en la web son solamente una parte de todo lo que se podría llegar a mostrar de la recaudación de datos de los usuarios, a parte de los distintos usos que se le podría dar. Otra posible mejora es incorporar lo servicios de Google Analytics para complementar los datos ya recogidos por la aplicación.

- **Publicar el proyecto en la Play Store:** Como punto final quedaría publicar la aplicación de Android en la Play Store de Google para que todo el mundo la pudiera utilizar y la comunidad se llenase de gente participativa que recomendase y crease lugares a los que ir.

- **Publicar el Framework como Open Source:** Para finalizar, también se tendría que dejar público el framework para que la comunidad de desarrolladores lo utilizase para sus propios proyectos, ya que se ha realizado con esa intención. Una plataforma que se podría utilizar sería GitHub ya que cuenta con una gran comunidad de programadores.

Bibliografía

[1] Brown A. *Microsoft admits Windows Phone is dead, leaves Surface Phone launch uncertain.* URL: https://www.express.co.uk/life-style/science-technology/864074/MIcrosoft-Windows-Phone-Dead-Surface-Phone-Release-Date (visitado 01-2018).

[2] Davis A. *IRM GPRS y 3G Security Overview.* URL: https://web.archive.org/web/20080209213430/http://www.gprssecurity.com/ (visitado 12-2017).

[3] Esaú A. *Qué son Microservicios y ejemplos reales de uso.* URL: https://openwebinars.net/blog/microservicios-que-son/ (visitado 01-2018).

[4] Open Handset Allianze. *Industry Leaders Announce Open Platform for Mobile Devices.* URL: http://www.openhandsetalliance.com/press_110507.html (visitado 11-2017).

[5] Chesneau B. *Gunicorn WSGI server.* URL: http://docs.gunicorn.org/en/latest/index.html (visitado 01-2018).

[6] Elgin B. *Google Buys Android for Its Mobile Arsenal.* URL: https://www.webcitation.org/5wk7sIvVb?url=http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm (visitado 11-2017).

[7] campusMVP. *Docker vs Vagrant: diferencias y similitudes y cuándo usar cada uno.* URL: https://www.campusmvp.es/recursos/post/Docker-vs-Vagrant-diferencias-y-similitudes-y-cuando-usar-cada-uno.aspx (visitado 01-2018).

[8] Contributors of Chart.js. *Chart.js.* URL: http://www.chartjs.org/docs/latest/ (visitado 01-2018).

[9] SQLite Consortium. *SQLite.* URL: https://sqlite.org/index.html (visitado 01-2018).

[10] Agendaless Consulting. *Supervisor: A Process Control System.* URL: http://supervisord.org/ (visitado 01-2018).

[11] Android Developers. *Meet Android Studio.* URL: https://developer.android.com/studio/intro/index.html (visitado 12-2017).

[12] Ravenscraft E. *Augmented Reality Showdown: Pokémon Go vs. Ingress.* URL: https://lifehacker.com/augmented-reality-showdown-pokemon-go-vs-ingress-1783801702 (visitado 01-2018).

[13] elEconomista.es. *Barcelona formaliza la multa a Airbnb y le pondrá otra de 600.000 euros si sigue ofertando pisos ilegales.* URL: http://www.eleconomista.es/empresas-finanzas/noticias/8456522/06/17/Barcelona-formaliza-la-multa-a-Airbnb-y-le-pondra-otra-de-600000-euros-si-sigue-ofertando-pisos-ilegales.html (visitado 11-2017).

[14] Django Software Foundation. *Django documentation.* URL: https://docs.djangoproject.com/en/2.0/ (visitado 12-2017).

[15] Python Software Foundation. *Tutorial de Python.* URL: http://docs.python.org.ar/tutorial/3/real-index.html (visitado 12-2017).

[16] The Linux Foundation. *Lets Encrypt.* URL: https://letsencrypt.org/ (visitado 01-2018).

[17] The jQuery Foundation. *jQuery API.* URL: http://api.jquery.com/ (visitado 01-2018).

[18] Google. *Material Design, Dialogs.* URL: https://material.io/guidelines/components/dialogs.html (visitado 12-2017).

[19] Equipo de Google España. *Enhorabuena a los ganadores del Concurso Start Ups Innovación Móvil 2015.* URL: https://espana.googleblog.com/2015/11/enhorabuena-los-ganadores-del-concurso.html (visitado 12-2017).

[20] Gutiérrez H. *Los taxistas van hoy a la huelga en toda España contra Uber y Cabify.* URL: https://elpais.com/economia/2017/11/28/actualidad/1511891704_468822.html (visitado 11-2017).

[21] Square Inc. *Retrofit, A type-safe HTTP client for Android and Java.* URL: http://square.github.io/retrofit/ (visitado 12-2017).

[22] Hernández J. *Qué significa G, E, 3G, H/3G+, H+, 4G.* URL: https://www.emezeta.com/articulos/tecnologias-moviles-g-e-3g-h-4g (visitado 12-2017).

[23] Sanfeliu J. *What is Monitorix?* URL: http://www.monitorix.org/ (visitado 01-2018).

[24] Juanda. *Arquitectura de una API REST.* URL: https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html (visitado 01-2018).

[25] Lamport L. *LaTeX. A document preparation system.* URL: https://www.latex-project.org/ (visitado 01-2018).

[26] Cid M. *Confirmado: preferimos comunicarnos mediante chat, por encima de las llamadas.* URL: https://www.xatakamovil.com/movil-y-sociedad/confirmado-preferimos-comunicarnos-mediante-chat-por-encima-de-las-llamadas (visitado 01-2018).

[27] Gibbons M. *Django REST Swagger.* URL: https://django-rest-swagger.readthedocs.io/en/latest/ (visitado 01-2018).

[28] Hachman M. *The price of free: how Apple, Facebook, Microsoft and Google sell you to advertisers.* URL: https://www.pcworld.com/article/2986988/privacy/the-price-of-free-how-apple-facebook-microsoft-and-google-sell-you-to-advertisers.html (visitado 01-2018).

[29] Kwon M. y col. *Development and Validation of a Smartphone Addiction Scale (SAS).* URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0056936 (visitado 11-2017).

[30] Otto M. y Thornton J. *Bootstrap.* URL: http://getbootstrap.com/ (visitado 01-2018).

[31] Vitas M. *ART vs Dalvik - introducing the new Android runtime in KitKat.* URL: https://infinum.co/the-capsized-eight/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat (visitado 12-2017).

[32] Zatti M. y Garrido R. *Aprender idiomas con una app es posible: un estudio de la Universidad de la Ciudad de Nueva York confirma la eficacia de Babbel.* URL: https://press.babbel.com/es/releases/2016-09-29-Spanisch-Study_SPA.html (visitado 12-2017).

[33] Kerris N. y Dowling S. *Apple reinvents the phone with iPhone.* URL: https://www.apple.com/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/ (visitado 11-2017).

[34] NetQuest. *NetQuest.* URL: https://www.netquest.com/es/home/encuestas-online-investigacion (visitado 01-2018).

[35] Contributors of acme nginx. *Python acme client for Nginx.* URL: https://github.com/kshcherban/acme-nginx (visitado 01-2018).

[36] Oracle. *The Java Tutorials.* URL: https://docs.oracle.com/javase/tutorial/ (visitado 12-2017).

[37] Losada S. *QUÉ ES ELK: ElasticSearch, Logstash y Kibana*. URL: https://openwebinars.net/blog/que-es-elk-elasticsearch-logstash-y-kibana/ (visitado 01-2018).

[38] SmartBear Software. *What Is OpenAPI?* URL: https://swagger.io/docs/specification/about/ (visitado 01-2018).

[39] Christie T. *Django REST framework*. URL: http://www.django-rest-framework.org/ (visitado 12-2017).

[40] Franceschin T. *PUBLICIDAD vs FREEMIUM vs SUSCRIPCIÓN*. URL: http://www.vrainz.com/publicidad-vs-freemium-vs-suscripcion/ (visitado 01-2018).

[41] Holwerda T. *The Palm operating system*. URL: http://mobile.osnews.com/story.php/26838/Palm-Im-ready-to-wallow-now/page4 (visitado 12-2017).

[42] Ocock T. *Symbian OS one of the most successful failures in tech history*. URL: https://techcrunch.com/2010/11/08/guest-post-symbian-os-one-of-the-most-successful-failures-in-tech-history-2/ (visitado 12-2017).

[43] Vincenty T. *Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations*. URL: https://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf (visitado 01-2018).

[44] UpGuard. *Puppet vs Chef Revisited*. URL: https://www.upguard.com/articles/puppet-vs.-chef-revisited (visitado 01-2018).

[45] Cunningham W. *Xml Rpc Vs Corba*. URL: http://wiki.c2.com/?XmlRpcVsCorba (visitado 01-2018).

[46] w3schools. *XML Soap*. URL: https://www.w3schools.com/xml/xml_soap.asp (visitado 01-2018).