Plataforma para el fomento turístico basado en recompensas mediante tecnologías Blockchain

Jaime Tamames Hergueta

Sergio Pino Holgado

Trabajo de fin de grado del Grado en Ingeniería de Software

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Junio 2019

Director: Juan Antonio Recio García

Colaborador: Felipe Santi (Sismotur)

Agradecimientos

Queremos aprovechar estas primeras líneas para agradecer a todas las personas que nos ayudado a llegar hasta aquí.

En primer lugar, queremos darle las gracias a la persona que nos guiado en este trabajo de fin de grado, Juan Antonio Recio García. Por darnos la oportunidad de realizar este proyecto tan interesante en colaboración con la empresa Sismotur, por tu tiempo y por la atención recibida. Como no, también queremos darle las gracias a Sismotur, en especial a Felipe Santi por esta oportunidad y por ayudarnos cada vez que lo hemos necesitado.

También nos gustaría agradecer a todos los profesores de la Facultad de Informática de la UCM y el personal que trabaja en ella haciendo posible que cada año se gradué muchísima gente como nosotros.

Por ultimo, queremos mostrar nuestro agradecimiento a todos los amigos y compañeros con los que hemos disfrutado una preciosa etapa de nuestras vidas.

Índice general

Glosario

API Application Programming Interface. Permite la comunicación entre distintos componentes software. 5, 7, 18, 33, 44, 45, 47, 48, 62, 63, 64, 65, 66, 67, 69, 70, 89, 90, 107, 108, 109, 110, 111, 112

Árboles cifrados de Merkle Un árbol hash de Merkle o árbol hash es una estructura de datos en árbol, binario o no, en el que cada nodo que no es una hoja está etiquetado con el hash de la concatenación de las etiquetas o valores (para nodos hoja) de sus nodos hijo. Son una generalización de las listas hash y las cadenas hash. Proporciona un método de verificación segura y eficiente de los contenidos de grandes estructuras de datos. 30

Bluetooth LE Bluetooth Low Energy. Es una tecnología de red de área personal. 22, 27

Cadena de bloques Una cadena de bloques, conocida en inglés como blockchain, es una estructura de datos en la que la información contenida se agrupa en conjuntos (bloques) a los que se les añade metainformaciones relativas a otro bloque de la cadena anterior en una línea temporal, de manera que gracias a técnicas criptográficas, la información contenida en un bloque solo puede ser repudiada o editada modificando todos los bloques posteriores. 28, 29, 30, 34

CSS Cascading Style Sheets. Es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de los documentos web y las interfaces de usuario. 5, 42, 64

Endpoint URL's que reciben o retornan información de una API. 7, 18, 45, 48, 67, 107, 108, 109, 110, 111, 112

HTML HyperText Markup Language. Hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web. 5, 42, 64

HTTP Hypertext Transfer Protocol. Es el protocolo de comunicación que permite las transferencias de información en la World Wide Web (Internet). 45

JavaScript Es un lenguaje de programación interpretado orientado a objetos. Se utiliza del lado del cliente, implementado como parte de un navegador web. 5, 42, 62, 64, 66, 69

MongoDB Es un sistema de base de datos NoSQL orientado a documentos de código abierto. 5, 42, 47, 66

NodeJS Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor. 5, 38, 42, 66

Ripple Considerada la criptomoneda de los bancos, Ripple es un sistema virtual de pagos en tiempo real basado en la tecnología de blockchain que es utilizada por instituciones financieras como una forma más rápida y económica que sus sistemas de respaldo tradicionales, ya que les permite enviar y recibir dinero y liquidar transacciones a una velocidad de entre 5 y 10 segundos (mucho más rápida que Bitcoin).. 33

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Resumen

España tiene una fuerte cultura y gran diversidad de tradiciones, no solo a nivel nacional si no autonómico. Su posición geográfica permite disfrutar de un clima bastante regular, sobre todo en el sur. La gastronomía también tiene un fuerte marcado regional, ofreciendo un abanico de posibilidades muy extenso y de calidad. Es por eso por lo que España recibe un total aproximado de 82 millones de turistas extranjeros al año, recibiendo mas turistas que Estados Unidos y solo por debajo de Francia.

Con esta afluencia de gente se ha tenido que invertir mucho para satisfacer las necesidades del turista, por eso España tiene una de las mejores infraestructuras de servicios turísticos a nivel mundial. Con las nuevas tecnologías que se han ido desarrollando durante las ultimas décadas el turista goza de servicios que hace 20 años no se podían ni imaginar. Hoy en día la gran mayoría somos conocedores de aplicaciones o paginas web donde encontrar hoteles, vuelos, trenes, restaurantes y actividades que hacer en cualquier ciudad del mundo. Pero se esta perdiendo el turismo cultural o histórico, sobre todo en la gente joven. Es muy común que veas en redes sociales a gente posando frente a la Puerta de Alcalá, la Torre Eiffel o la Fontana di Trevi, se han ido de viaje a cualquiera de estas ciudades, las visitan, las disfrutan, pero realmente no conocen muy bien su historia.

Sismotur es una empresa turística, líder en la elaboración de planes de señalización y en servicios de información turística inteligente. Inventrip, una de sus plataformas, pone a tu disposición la oferta turística de un lugar. Mediante el uso de beacons, una tecnología basada en Bluetooth, te permite recibir directamente en tu dispositivo móvil la información relevante de ese lugar. En este Trabajo de Fin de Grado se ha colaborado con Sismotur para crear una aplicación que junto a su red de señalización beacon, los dispositivos móviles de los usuarios y la plataforma de criptomoneda Stellar, permita incentivar a los turistas a informarse mas y mejor sobre los lugares que visitan. Ofreciendo recompensas en forma de criptomoneda a los turistas que realicen las rutas propuestas.

1.2. Abstract

Spain has a strong culture and a great diversity of traditions, not only nationally but also autonomously. Its geographical position allows to enjoy a fairly regular climate, especially in the south. Gastronomy also has a strong regional mark, offering a wide range of possibilities and quality. That is why Spain receives a total of approximately 82 million foreign tourists a year, receiving more tourists than the United States and only below France.

With this influx of people has had to invest a lot to meet the needs of tourists, so Spain has one of the best infrastructures of tourism services worldwide. With the new technologies that have been developed during the last decades the tourist enjoys services that 20 years ago could not even imagine. Today many of us know applications or web pages where to find hotels, flights, trains, restaurants and activities to do any city in the world. But cultural or historical tourism is being lost, especially in young people. It is very common to see people posing in front of the Puerta de Alcalá, the Eiffel Tower or the Fontana di Trevi in social networks, they have gone to any of these cities, visit them, enjoy them, but they do not really know very well his story.

Sismotur is a tourism company, leader in the development of signal plans and intelligent tourist information services. Inventrip, one of its platforms, puts at your disposal the tourist offer of a place. Through the use of beacons, a technology based on Bluetooth, allows you to receive directly on your mobile device the relevant information of that place. In this End of Degree Project, we have collaborated with Sismotur to create an application that, together with its beacon signaling network, the users' mobile devices and the Stellar cryptocurrency platform, allows tourists to learn more and better about the places they visit. Offering rewards in the form of a cryptocurrency to tourists who make the proposed routes.

1.3. Estructura

Organizaremos el contenido de esta memoria de la siguiente forma:

- En este capítulo podrá encontrar un breve resumen de este Trabajo de Fin de Grado junto a la estructura del mismo.
- En el segundo capítulo describiremos la motivación para desarrollar esta aplicación y los objetivos que queremos lograr.
- En el tercer capítulo explicaremos que son los sistemas turísticos, que son los beacons y veremos los diferentes tipos de criptomonedas que hemos barajado para el desarrollo de la aplicación.
- En el cuarto capitulo detallamos como nos hemos organizado a nivel interno, las herramientas que hemos utilizado para ello y las que hemos utilizado para desarrollar el software. También explicamos que tipo de metodología hemos seguido y por qué.
- El quinto capítulo detalla los tipos de usuarios que tiene la aplicación, los requisitos funcionales de cada tipo de usuario y la funcionalidad completa.
- En el sexto capítulo se detalla tanto la arquitectura utilizada como las tecnologías empleadas para el desarrollo de la aplicación.
- En el séptimo capitulo detallamos los casos de uso de todas las acciones que pueden llevar a cabo los usuarios, las precondiciones, postcondiciones y los errores que podrían darse.
- El octavo capítulo por decirlo de alguna forma es el mas personal de todos, en el damos un pequeño feedback de lo que nos ha parecido el desarrollo de este proyecto y las mejoras a futuro que se podrían hacer.
- En el ultimo capítulo describimos como desplegar y preparar el entorno para su ejecución. También detallamos los endpoints de la API.

Capítulo 2

Motivación y objetivos

2.1. Motivación

Ponte en la situación de que te quieres ir de viaje con tu pareja. No tienes muy claro el destino por lo que decides investigar un poco. Hace poco te han hablado de Inventrip una aplicación que te ayuda a planificar tu viaje, te ofrece rutas turísticas con información sobre los lugares que vas a visitar y que además si completas estas rutas te dan una recompensa en forma de criptomoneda que mas tarde podrás canjear en restaurantes de la zona.

Gracias a la información que te brinda Inventrip eliges el destino que mas te ha gustado. Al llegar allí, después de instalarte en tu alojamiento, decides salir con tu pareja a conocer el lugar. Si es la primera vez que lo visitas puede que solo sepas cosas de oídas o recomendaciones de amigos, gracias a Inventrip puedes completar esa información de forma sencilla. Al abrir la aplicación ves multitud de puntos de interés turísticos, restaurantes y sitios de ocio, pero no solo eso, si no que te recomienda una ruta turística que al completarla te dará una recompensa, una cantidad de dinero en forma de criptomoneda que podrás canjear por una botella de vino gratis en varios restaurantes de la zona a elegir.

Con todo organizado empezáis la ruta caminando hacia el punto mas cercano, al llegar te acercas a la señal inteligente, tu dispositivo móvil se comunica con el beacon por Bluetooth.

Te muestra información sobre el lugar en tu idioma y te comunica que la prueba de paso del primer punto de la ruta se ha realizado correctamente. Continuas la ruta dirigiéndote a los demás puntos, en los que como en el primero aprendes historia y características del lugar, a su vez vas validando las pruebas de paso de esos puntos.

Al llegar al ultimo punto la aplicación te notifica que has completado todas las pruebas de paso necesarias para obtener tu recompensa, dentro de la aplicación te diriges a tu cartera y efectivamente ves que se te ha transferido una cantidad de Lumens, la moneda de la red de criptomoneda Stellar, la cual mas tarde podrás canjear en los restaurantes que te señala.

Pasado un rato decidís que es buena hora para cenar y elegís un restaurante de la lista. Al llegar, después de acomodaros, habláis con el camarero acerca de la recompensa. De forma sencilla el camarero verifica vuestra recompensa y acto seguido os trae la botella de vino.

2.2. Dominio

2.2.1. Sismotur

Sismotur es una empresa de turismo creada en el año 2000 con el objetivo de ofrecer consultoría enfocada a la implementación de sistemas de señalizaciones inteligentes.

Para ello se han servido de las ultimas tecnologías del mercado ofreciendo dos herramientas muy potentes, así es como ellos mismo las definen:

> **"Inventrip"** es un servicio de gestión e información de señalización inteligente que permite a sus viajeros consultar su oferta turística, crear y reservar viajes personalizados y compartirlos en redes sociales o utilizando las últimas tecnologías: Beacons, NFC, códigos QR.

> Sismotur es líder en la elaboración de planes de señalización y para ello implanta **"Signing"**, un servicio web especializado para la planificación y gestión de la señalización territorial y urbana utilizado por más de 30 administraciones públicas.

Hoy en día el perfil del turista ha cambiado mucho, gracias a las nuevas tecnologías que tenemos al alcance de la mano podemos conocer cualquier cosa en cualquier momento y lugar. Sismotur aprovecha estas tecnologías para junto a administraciones publicas ofrecer un servicio turístico de calidad, veraz y unificado en un solo sistema. Ya que las búsquedas en google o motores de búsqueda de turismo ofrecen información fragmentada y que podría no estar verificada por alguien con el conocimiento adecuado sobre el lugar en cuestión.

Este sistema acompaña al turista desde que empieza a planificar el viaje en su casa hasta que una vez finalizado ofrece su valoración. Ofreciendo en todo momento un servicio exclusivamente pensado en la satisfacción del turista.

2.2.2. Inventrip

El objetivo final de este Trabajo de Fin de Grado es que fuera implementado en la aplicación Inventrip, por eso creemos que es importante ahondar un poco mas en su funcionamiento y las posibilidades que ofrece.

Inventrip pone a tu disposición la oferta turística de un lugar, con ella puedes planificar viajes a medida y disponer de tu planificación en cualquier dispositivo.

En tres sencillos pasos te permite crear una agenda de viaje, seleccionando por días lo que se quiere visitar, donde comer y donde dormir. Una vez finalizada la planificación puedes compartirla con tus compañeros de viaje.

Figura 2.1: Planificación de viaje con Inventrip

Para ello Sismotur cuenta con dos canales de información, el primero es la señalización turística que implantan en los destinos. Las administraciones publicas sacan a concurso la administración e implantación de las señales físicas para informar y guiar al turista en un destino. Además de eso Sismotur integra en sus señales unos dispositivos llamados Beacons, que utilizan una señal Bluetooth LE que ofrece información directa sobre el lugar que se esta visitando.

Estos dispositivos se integran en la propia señal de información como se puede apreciar en la siguiente ilustración.

El segundo canal de información seria la propia aplicación de Inventrip, que dispondría de

Figura 2.2: Integración de un Beacon en señal turistica

toda la información previamente explicada de forma online.

De esta forma cuando el turista esta usando la App de Inventrip en su dispositivo móvil cerca de una señal que integra un Beacon estos dos dispositivos se comunican juntando toda la información en un mismo lugar.

Esto le permite al turista obtener respuesta a preguntas comunes como ¿Dónde estamos?, ¿Qué tenemos alrededor?, ¿Qué estamos visitando? Como si te estuviera atendiendo un agente de la oficina de turismo «in situ».

Este tipo de señalización ofrece ventajas notables, por ejemplo, puedes actualizar la información que ofrece un Beacon de forma muy simple y añadir todos los idiomas que quieras. Algo que en una señal física convencional no puedes hacer, ya que tienes un espacio limitado. También puedes acceder a toda la información que brindan los Beacon sin tener conexión a internet en tu dispositivo móvil, es común que cuando viajas a menudo tengas que desactivar la conexión a internet para evitar elevados costes de roaming.

2.3. Objetivos

Han sido varios los objetivos que nos han mantenido motivados durante todos estos meses pero los mas importantes son los siguientes:

- Aplicar los conocimientos y tecnologías adquirido/as durante el grado. Lo cual nos ha permitido ampliar contenidos para muchas asignaturas que se nos habian hecho cortas o que nos hubiera gustado profundizar más.
- Aprender como funcionan las nuevas tecnologías basadas en Blockchain, algo realmente novedoso y que ha llegado para cambiar el mundo tecnologíco tal y como lo conocemos hoy; nos hemos sentido parte de este nuevo cambio, pudiendo tocar algunas tecnologías que en el grado no tuvimos ocasión de ver.
- Definir e implementar una solución a un problema real y ser capaces de generar un prototipo funcional que haga uso de las nuevas tecnologías. Aunque durante el grado hemos tenido muchas sesiones prácticas para aplicar lo aprendido durante las clases teóricas, no es posible profundizar mucho en cada materia; por eso el TFG es el momento ideal para aprender más sobre algunos campos poco estudiados durante los cursos.
- Tener la posibilidad de trabajar en conjunto con una empresa y conocer el proceso de desarrollo de software en un entorno de trabajo real.

Capítulo 3

Estado del arte

3.1. Sistemas turísticos

Podemos definir un sistema como el conjunto de recursos que interactúan entre si para poder llegar a un objetivo común. Un sistema turístico esta conformado por varios elementos. El turista por un lado, el lugar de salida, la zona por donde el turista se va a desplazar y el lugar de destino. Estos elementos interactúan entre si y son dependientes unos de otros para su correcto funcionamiento.

También hay que definir que es la demanda y la oferta turística para poder comprender el sistema en su totalidad.

- **Demanda turística**: Esta conformada por el conjunto de servicios y productos que el mercado requiere para satisfacer sus necesidades.
- **Oferta turística**: Esta conformada, dentro de un espacio geográfico, como los servicios y productos ofrecidos al publico para crear un mercado competitivo que atraiga a turistas.

La relación entre la oferta y la demanda es obvia. La demanda se ve afectada internamente sobre todo por la motivación que el turista tenga sobre ese lugar en cuestión, externamente intervienen mas factores como la política, la situación demográfica y social, las prestaciones

tecnológicas, la seguridad... Por otro lado la oferta se ve afectada por los recursos turísticos, las empresas que se relacionan con el sector turístico, la infraestructura y las instituciones gubernamentales sobre la que se desarrolla la actividad...

Sismotur es una empresa de servicios turísticos que oferta por un lado mediante su plataforma «Signing» la planificación, elaboración y gestión de planes de señalización territorial y urbana; por otro lado mediante su plataforma «Inventrip» ofrece un servicio de información y gestión de la señalización turística inteligente, que permite consultar la oferta turística, reservar y construir viajes a medida.

Con el desarrollo de este Trabajo de Fin de Grado Sismotur quiere añadir a sus servicios la posibilidad de de hacer una oferta turística mas atractiva para los demandantes por medio del sistema ya explicado, remunerar a aquellas personas que completen rutas turísticas mediante el pago a través de una criptomoneda.

3.2. Beacons

Un beacon es un dispositivo hardware de bajo consumo, el cual emite una señal broadcast. Utilizando una conexión Bluetooth LE para enviar mensajes o avisos a dispositivos cercanos sin la necesidad de sincronizarlos previamente.

Son muy reducidos en tamaño, lo cual permite ponerlos o integrarlos en casi cualquier lugar o estructura. Es por lo que se usan frecuentemente para la localización precisa dentro de entornos cerrados o espacios abiertos sin tener que depender de la señal GPS, que es fácilmente manipulable.

Sismotur usa estos dispositivos en sus señalizaciones para proporcionar los datos requeridos en cada situación. Gracias a que están integrados dentro de la propia señal, no son manipulables por personas ajenas. Esto es importante puesto que la aplicación que queremos desarrollar utiliza estos beacons como prueba de paso, certificando que el usuario ha estado en ese punto.

3.3. Tecnología Blockchain

3.3.1. ¿Qué es Blockchain?

Una Cadena de bloques o blockchain es, en el más simple de los términos, una serie de registros de datos inmutables con marca de tiempo que es administrada por un grupo de ordenadores que no son propiedad de una sola entidad. Cada uno de estos bloques de datos se asegura y se vincula entre sí utilizando principios criptográficos.

La red blockchain no tiene una autoridad central, es la definición misma de un sistema democratizado. Dado que es un libro contable compartido e inmutable, la información que contiene está abierta para que todos y cada uno la puedan ver. Por lo tanto, todo lo que se construye en el blockchain es transparente por su propia naturaleza y todos los involucrados son responsables de sus acciones.

3.3.2. ¿Cómo funciona Blockchain?

Imagine una hoja de cálculo que se propaga miles de veces en una red de ordenadores. Luego imagina que esta red está diseñada para actualizar regularmente esta hoja de cálculo y ya tienes un conocimiento básico de como funciona la Cadena de bloques.

La información contenida en una Cadena de bloques existe como una base de datos compartida y continuamente conciliada. Esta es una forma de usar la red que tiene beneficios obvios. La base de datos de blockchain no se almacena en una sola ubicación, lo que significa que los registros que mantiene son verdaderamente públicos y fácilmente verificables. No existe una versión centralizada de esta información para que un pirata informático la corrompa. Alojado por millones de ordenadores a la vez, sus datos son accesibles para cualquier persona en Internet.

3.3.3. ¿Quién mantiene las redes Blockchain?

La red blockchain esta mantenida por una red peer-to-peer. La red es una colección de nodos que están interconectados entre sí. Los nodos son ordenadores individuales que toman una entrada, realizan una función en ella y dan una salida. La Cadena de bloques utiliza un tipo especial de red llamada red de igual a igual", que divide su carga de trabajo entre los participantes, que tienen el mismo privilegio, denominados ïguales". Ya no hay un servidor central, ahora hay muchos servidores distribuidos y descentralizados.

3.3.4. ¿Quién usará Blockchain?

Como infraestructura web, no es necesario conocer la red de blockchain para que sea útil en su vida.

Actualmente, las finanzas ofrecen los casos de uso más fuertes para la tecnología. Por ejemplo El Banco Mundial estima que se enviaron más de 430 mil millones de dólares estadounidenses en transferencias de dinero en 2015. Y en este momento hay una gran demanda de desarrolladores de blockchain.

El blockchain potencialmente elimina al intermediario para este tipo de transacciones. La computación personal se volvió accesible para el público en general con la invención de la Interfaz Gráfica de Usuario (GUI), que tomó la forma de un .escritorio". De manera similar, la GUI más común diseñada para la Cadena de bloques son las llamadas aplicaciones de «cartera», que la gente usa para comprar cosas con Bitcoin y almacenarlas junto con otras criptomonedas.

3.4. Bitcoin

3.4.1. ¿Qué es Bitcoin?

Bitcoin fue lanzado en 2009 como un software de código abierto, se suele decir a menudo que fue la primera criptomoneda del mundo y se define como una moneda digital que solo existe electronicamente. Por estos motivos fue nuestro primer contacto con las criptomonedas, ya que todas o casi todas las demás criptomonedas que han surgido posteriormente se basan en su tecnología.

Bitcoin no tiene una autoridad emisora central o una institución política que controle la cantidad de Bitcoin en circulación, es por eso que se dice que es una moneda descentralizada.

El proceso es bastante simple: los titulares de Bitcoin pueden hacer transferencias de bitcoins a traves de una red de igual a igual. Todas las trasferencias quedan registradas en la Cadena de bloques, mas conocida como el libro mayor. Cada «bloque» de la Cadena de bloques esta formado por una estructura de datos basada en Árboles cifrados de Merkle. Muy útiles para detectar fraudes o archivos dañados. Si un solo archivo en una cadena es fraudulento, la propia Cadena de bloques mediante comprobaciones evita que dañe el resto del libro mayor.

La propia Cadena de bloques controla cuando se producen bitcoins y cuantos se producen. También realiza un seguimiento de dónde están los bitcoins y garantiza que las transacciones sean precisas. Por lo que significa que el suministro esta controlado por diseño.

3.4.2. ¿Cómo funciona Bitcoin?

Una de las características más atractivas de Bitcoin es su implacable proceso de verificación, que minimiza en gran medida el riesgo de fraude. Dado que Bitcoin está descentralizado, los voluntarios, conocidos como «mineros», verifican y actualizan constantemente la Cadena de bloques. Una vez que se verifica una cantidad específica de transacciones, se agrega otro bloque a la Cadena de bloques y el proceso continúa de forma habitual.

3.4.3. ¿Qué es la “minería”?

En lugar de que un solo servidor central verifique cada transacción, básicamente cada persona en la red verifica cada transacción.

Simplifiquemos el proceso para poder entenderlo mejor: a los mineros se les presenta un problema matemático complicado y el primero en resolver el problema matemático agrega el bloque verificado de transacciones al libro mayor. Los cálculos se basan en una Prueba de trabajo (POW), o la prueba de que se gastó una cantidad mínima de energía para obtener una respuesta correcta.

No hay seres personas reales sentadas frente al ordenador con trozos de papel de cuaderno y calculadoras. Se utiliza hardware para realizar la minería de Bitcoin.

El sistema de recompensas incorporado de Bitcoin compensa a los mineros exitosos con una porción de bitcoins. La recompensa cambia con el tiempo según la programación de Bitcoin, y la recompensa de bloque se reduce a la mitad cada cuatro años.

A grandes rasgos y de forma reducida este seria el funcionamiento de Bitcoin. El libro «Mastering Bitcoin» de Andreas M.Antonopoulos fue nuestro libro de referencia en cuanto a esta materia. Gracias a el pudimos entender en profundidad el funcionamiento de esta tecnología, incluso a bajo nivel.

3.5. Stellar

3.5.1. ¿Qué es Stellar?

Stellar se anuncia a sí misma como una infraestructura de pagos distribuida de fuente abierta, basada en la premisa de se necesita «una red financiera mundial abierta a todos». El proyecto está cubriendo esta necesidad, conectando a individuos, instituciones y sistemas de pago a través de su plataforma.

El equipo de Stellar quiere que las transacciones monetarias sean más baratas, más rápidas y más confiables de lo que son bajo los sistemas actuales.

A diferencia de Bitcoin, Stellar no tiene el mimso nombre para su infraestructura que para su moneda. Llamando a esta Lumens.

3.5.2. ¿Cómo funciona Stellar?

Al igual que en Bitcoin, Stellar es una red descentralizada. En lugar de recopilar información en una fuente centralizada como un banco, se distribuye entre nodos interconectados en la red Stellar.

Cualquier persona puede configurar un nodo de verificación con Stellar Core, la columna vertebral de la red Stellar Network que realiza la verificación real utilizando el Stellar Consensus Protocol (SCP). Piense en SCP como el algoritmo de Stellar para verificar las transacciones. Es muy diferente de Bitcoin, que utiliza la prueba de trabajo para verificar las transacciones.

Por lo tanto, Stellar Network es solo una serie de Stellar Cores, que trabajan en conjunto para verificar las transacciones y asegurarse de que todo esté actualizado. Una vez que se verifique una transacción, se agregará al libro mayor público.

3.5.3. Características

A continuación detallaremos las características por las que elegimos Stellar en lugar de otra criptomoneda.

- Al igual que Ripple, Stellar puede manejar intercambios entre monedas basadas en fiat y entre criptomonedas.
- Los Lumens tienen tarifas bajas (cada transacción tiene una tarifa menor, 0,00001 Lumenes, asociada a ella). Esta característica fue determinante a la hora de elegir una criptomoneda, debido a que la actividad que se quiere llevar a cabo estaria financiada por entidades publicas, minimizando el coste en cada transacción.
- El tiempo de confirmación es de 3 a 5 segundos.
- Puede realizar hasta mil transacciones por minuto.
- Su API es simple y limpia, ademas de disponerla en distintos lenguajes de programación.
- Stellar.org, la organización que apoya a Stellar, está centralizada como Ripple y está destinada a manejar transacciones multiplataforma y micro transacciones como Ripple. Sin embargo, a diferencia de Ripple, Stellar.org no tiene fines de lucro y su propia plataforma es de código abierto y descentralizada. Por lo tanto, tienen la ventaja de sentirse un poco más como una compañía tradicional que puede conectarse con otras compañías por un lado, pero tienen la transparencia del código abierto, de estar distribuida y de propiedad comunitaria que Ethereum y Bitcoin tienen por el otro. Algunos podrían ver que esto tiene lo mejor de ambos mundos. Hasta ahora, las principales compañías generalmente han abrazado a Ripple y Stellar.

3.6. Ethereum

3.6.1. ¿Qué es Ethereum?

Al igual que Bitcoin , Ethereum es una red de Cadena de bloques pública distribuida. Aunque hay algunas diferencias técnicas significativas entre los dos, la distinción más importante a tener en cuenta es que Bitcoin y Ethereum difieren sustancialmente en el propósito y la capacidad. Bitcoin ofrece una aplicación particular de la tecnología blockchain, un sistema de efectivo electrónico de igual a igual que permite los pagos en línea de Bitcoin. Mientras que la Cadena de bloques de Bitcoin se usa para rastrear la propiedad de la moneda digital (bitcoins), la Cadena de bloques Ethereum se enfoca en ejecutar el código de programación de cualquier aplicación descentralizada.

En la Cadena de bloques Ethereum, en lugar de minar para Bitcoin, los mineros trabajan para ganar Ether, un tipo de token de criptografía que alimenta la red. Más allá de una criptomoneda negociable, Ether también es utilizado por los desarrolladores de aplicaciones para pagar tarifas de transacción y servicios en la red Ethereum.

3.6.2. ¿Cómo funciona Ethereum?

La estructura de la Cadena de bloques de Ethereum es muy similar a la de Bitcoin, ya que es un registro compartido de todo el historial de transacciones. Cada nodo de la red almacena una copia de esta historia.

La gran diferencia con Ethereum es que sus nodos almacenan el estado más reciente de cada contrato inteligente, además de todas las transacciones de Ethernet.

Para cada aplicación Ethereum, la red debe realizar un seguimiento del .estado.º la información actual de todas estas aplicaciones, incluido el saldo de cada usuario y dónde está almacenado.

Bitcoin usa resultados de transacciones no gastados para rastrear quién tiene la cantidad de Bitcoin.

Aunque suena más complejo, la idea es bastante simple. Cada vez que se realiza una transacción de Bitcoins, la red rompe"la cantidad total como si fuera papel moneda, emitiendo Bitcoins de manera que los datos se comporten de manera similar a las monedas físicas o al cambio.

Para realizar transacciones futuras, la red de Bitcoin debe sumar todos sus cambios, que se clasifican como 'gastados' o 'no gastados'.

Etereum, por el contrario, utiliza las cuentas.

Al igual que los fondos de cuentas bancarias, las fichas de Ether aparecen en una billetera y se pueden transferir a otra cuenta. Los fondos siempre están en alguna parte, pero no tienen lo que podríamos llamar una relación continua.

Capítulo 4

Proceso de desarrollo

4.1. Herramientas

En esta sección detallamos cuales han sido las herramientas que hemos utilizado a lo largo de todo el proceso.

4.1.1. Productividad y comunicación

Debido a las diferencias en nuestros horarios de trabajo y la dificultad para adaptarnos optamos por mantener una comunicación casi exclusivamente online y para ello nos ayudamos de algunas de las herramientas disponibles:

- Google Hangouts: con la que realizábamos las videollamadas para concretar aspectos que nos era más complicado hablar por escrito.
- Telegram: la gran mayoria de las comunicaciones eran mediante esta vía, que nos ha facilitado enormemente las tareas pues en cualquier momento podíamos solicitar ayuda sin tener en cuenta la hora y/o horarios de la otra persona y ser contestados a la máxima brevedad.
- Google Drive: donde hemos ido compartiendo diversa información que por temas de or-

ganización era más cómodo que enviarla por Telegram. Al estar disponible toda esta información en único punto es más sencillo consultar cualquier tema.

4.1.2. Código

Toda la gestión tanto de código como de documentación (o la propia memoria) ha sido realizada integramente usando GitHub, concretamente sus repositorios privados gratuitos. Toda esta gestión se hace mediante el uso de la herramienta Git, bien desde la terminal de comandos o desde el cliente de escritorio que provee GitHub o el de Atlassian que se llama SourceTree. Esta forma de organizar y gestionar el código nos ha permitido mantenernos sincronizados en todo momento de forma eficiente y segura, pues funciona a la vez como una copia de seguridad, al no tener el código centralizado en un único punto.

Aunque cada uno trabajábamos en horarios muy distintos, siempre teníamos disponible el trabajo que había realizado la persona anterior, que junto con los mensajes que quedan registrados en cada commit, nos permitía tener una foto del estado actual del proyecto y de los cambios introducidos por la otra persona, todo esto sin necesidad de mantener una comunicación intermedia entre nosotros.

Otra de las bondades de Git es que permite la creación de «ramas» o «branches»; estas ramas permiten bifurcar el código generando una versión paralela del mismo y cuyos cambios no afectan a la rama principal o «master». Esto nos ha facilitado el flujo de trabajo cuando ambos trabajábamos de forma simultánea en alguna de las piezas.

4.1.3. Desarrollo

Para el desarrollo hemos empleado diversas herramientas. Cada uno tenía sus propias preferencias y no ha sido un problema gracias al uso de Git como gestor de código.

- WebStorm: IDE empleado en la elaboración de las dos aplicaciones escritas en NodeJS; te permite depurar el código de forma avanzada para poder corregir errores o mejorar el rendimiento y la estabilidad.

- Visual Studio Code: es un editor de texto «hipervitamido» que tiene la posibilidad de agregar plugins de terceros ampliando su abanico de posibilidades de forma prácticamente ilimitada.
- TexMaker: entorno para la elaboración de documentos usando LaTeX.

4.2. Metodología

Lo primero que hicimos antes de comenzar a trabajar en cualquiera de las partes del proceso fue definir la metodología que íbamos a emplear, siendo Agile la que más se ajustaba a nuestras necesidades, pues uno de los problemas que teníamos que resolver era la distancia y el horario tan diferentes de cada uno. De esta forma dicidimos planificar reuniones semanales y conversaciones diarias en las que resolver los problemas que fueran surgiendo.

4.2.1. Agile

Hemos seguido una aproximación Agile, en la medida que nos ha sido posible debido a nuestros horarios. De esta forma hemos dividido el trabajo en periodos semanales (iteraciones) en los que marcábamos unos objetivos a cumplir. Cada domingo de la semana realizábamos una revisión del estado del proyecto para determinar en qué punto estábamos y donde queríamos llegar en la próxima iteración.

Cada semana asignábamos las tareas que debían ser terminadas en el período establecido. No hemos tenido ningún inconveniente con esta forma de trabajo pues cada uno era responsable de su parte y nos comprometiamos a tenerla lista para no provocar una paralización en el resto de tareas. Para garantizar que todas fueran terminadas en el tiempo establecido las tareas eran partes pequeñas y sencillas de ejecutar. Cuando alguno teníamos un bloqueo avisábamos al compañero para que continuara el con esa parte.

En forma de «dailies», cada día manteníamos una conversación para comprobar el trabajo realizado el día anterior y comprobar posibles bloqueos o dudas. Por las dificultades evidentes que nos suponía la distancia, estas reuniones las realizábamos usando Telegram y/o Google Hangouts.

Capítulo 5

Arquitectura

En esta sección se describe como está formado el sistema y porqué. Se detallan las tecnologías que han sido empleadas, los patrones de diseño y la estructura de cada una de las piezas. Durante todo el proceso se han tenido en cuenta los estándares de desarrollo, así como las convenciones de cada tecnología, aunque en algunos casos haya sido necesario idear una forma alternativa bien por ausencia de estándares o bien por necesidad de adaptar algo muy concreto al proyecto.

5.1. Tecnologías

A continuación se listan algunas de las tecnologías más importantes que se han empleado para la construcción del sistema así como una breve explicación (si la hubiera) de porqué se han elegido sobre otras disponibles.

NodeJS

Puesto que uno de los requisitos era el emplear JavaScript como lenguaje principal del proyecto, NodeJS era el claro ganador frente a sus posibles rivales puesto que no existía la necesidad de formarnos pues ya disponiamos de los conocimientos necesarios.

JavaScript

Es el lenguaje empleado en NodeJS. Viene impuesto como requisito para la elaboración del sistema. Se propuso el usar Goland, un leguaje de Google, pero por temas de mantenibilidad y conocimientos del equipo se decidió que era mejor JavaScript. Uno de los puntos determinantes a la hora de elegir un lenguaje era la disponibilidad de las librerias oficiales de Stellar. Tanto Golang como JavaScript disponen de estas librerias mantenidas de forma oficial.

HTML5+CSS3

Todas las vistas de la WebApp se han elaborado mediante estos dos lenguajes, para la maquetación del sitio y para el aspecto visual del mismo. El motor de plantillas que se ha empleado con NodeJS es «ejs», que permite de una forma rápida y sencilla mostrar todo tipo de contenido sin perder la estructura HTML, ya que otros motores, al tener su propio lenguaje de marcado añaden un grado más de dificultad.

MongoDB

La necesidad de mantener los datos persistidos mediante un sistema rápido, fiable, seguro y fácil de usar nos llevó a elegir este sistema para persistir la información. MongoDB dispone de robustas librerias disponibles para NodeJS y trabajar con ellas hace que el tratamiento de datos pase a un segundo plano por la tremenda facilidad de uso. Para este proyecto se ha elegido la librería «Mongoose», que dispone de mucha documentación en internet, lo que facilita aún más el desarrollo.

5.2. Arquitectura del sistema

Dada la complejidad del sistema y tras un período de diseño y modelado llagamos a la conclusión que lo mejor era optar por una arquitectura distribuida en varias piezas de software y no limitarnos a un sistema monolítico que convertiría el proyecto en algo tedioso de mantener y modificar. Esta división se hizo teniendo en cuenta el uso del sistema y se dividió en 3 piezas fundamentales:

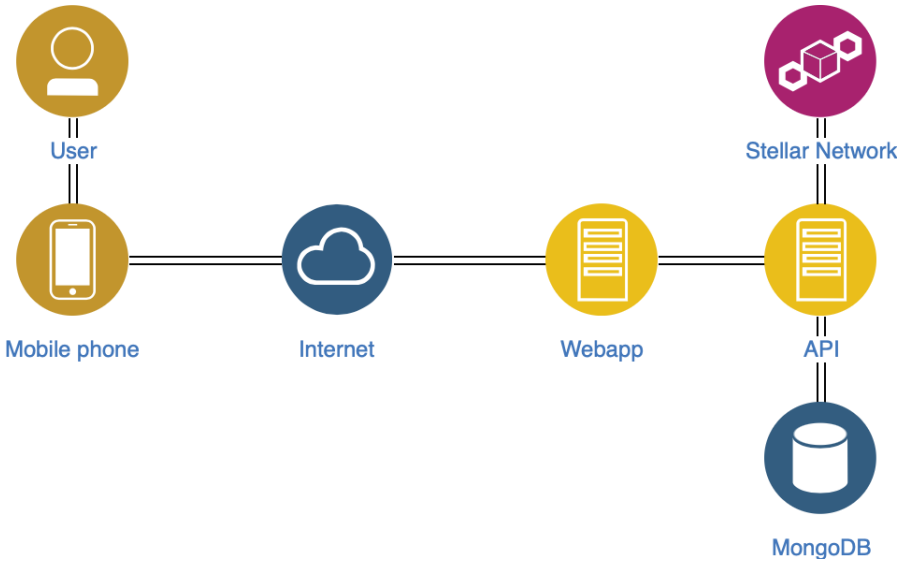

Figura 5.1: Arquitectura del sistema

- Webapp: desde la que los usuarios podrán interactuar con el sistema de forma segura y rápida.

- API: donde se desarrolla toda la lógica y se ejecutan todas las operaciones importantes. Se encarga de comunicar el resto de piezas del ecosistema y abstraer al usuario de operaciones complejas.

- Repositorio: aquí es donde se almacena toda la información tanto del propio sistema como de los usuarios.

5.2.1. Webapp

Es la pieza visible para el usuario, donde se encuentran los paneles de control desde los que realizar todas las operaciones disponibles.

En un inicio se pensó implementar esta parte como una aplicación móvil, pero por la complejidad de tener que construir una para iOS y otra para Android se optó por la forma que a nuestro parecer es la más flexible y adaptable: construir una aplicación web. La problemática actual de crear un producto para el mercado móvil es la necesidad de contar con mínimo dos aplicaciones (iOS y Android) para llegar al mayor número de usuarios posible. Haciendo una sola Webapp nos garantizamos alcanzar todos los mercados pues actualmente la práctica totalidad de dispositivos móviles independientemente de la version o del sistema operativo que utilicen tienen un navegador instalado desde el que se puede acceder a nuestra aplicación.

La estructura interna elegida está distribuida por capas. Las capas son las siguientes:

- Router: todas las peticiones HTTP llegan aquí, se procesan y se construye una respuesta que suele ser la vista solicitada por el usuario. Por la simplicidad de la lógica de esta pieza se decidió no aumentar la complejidad estructural añadiendo una capa más de lógica para gestionar las peticiones pues la mayoría son peticiones muy simples al tener toda la lógica delegada en la API.

- Vistas: es lo que el usuario ve. Se utiliza para organizar y presentar la información generada por el router de forma clara y vistosa para el usuario. Para facilitar la creación de las vistas se ha elegido un motor generador de vistas llamado EJS que permite describir las plantillas para formatear la información.

En los tiempos que corren hay que tener en cuenta la seguridad como un aspecto importante al desarrollar cualquier sistema informático, y más aún cuando este será desplegado en internet. Por ello la Webapp no permite realizar ningúna acción importante sin estar logeado y/o validado en el sistema; de esta forma, para poder interactuar con la API es necesario loguearse en el sistema y a partir de entonces todas las peticiones serán realizadas de forma validada.

5.2.2. API

El núcleo del sistema, donde reside la lógica que realiza todas las funciones para intercomunicar todas las piezas. Recibe las peticiones mediante HTTP, las procesa y devuelve un resultado. La API pone a disposicion del usuario una serie de rutas o endpoints (definidos en el apéndice) donde poder realizar las peticiones y como hacerlas. Estas peticiones pueden requerir ser validadas o no.

El código tiene una estructura por capas:

- Router: donde son recibidas todas las peticiones y desviadas al controlador adecuado, de forma transparente para el usuario. Se encarga de realizar las llamadas necesarias para chequear que las peticiones que lo requieren estén validadas correctamente, denegando el acceso en caso necesario.
- Controladores: la lógica que resuelve las peticiones recibidas desde el router. Contacta con el almacen en caso necesario para obtener o guardar información.
- Modelos: aquí se definen todas las entidades y tambien se encargan de realizar validaciones a la hora de interactuar con el almacén de datos. Cuando el controlador necesita guardar una entidad primero es el modelo el encargado de verificar que la información que se quiere almacenar esté correctamente formada.

Aunque estas capas a su vez pueden hacer uso de otras capas intermedias llamadas «middlewares» o «helpers». Esta forma de separar el código lo hace mucho mas mantenible y escalable para implementaciones futuras.

Los middlewares son los pasos intermedios por los que pasan algunas peticiones antes de llegar a la lógica final que formará la respuesta que espera el usuario. Un ejemplo muy claro para ayudar a entender como funciona un middleware: supongamos que se recibe una petición para eliminar un recurso del almacén, esta petición ha de ir validada y para comprobarlo hacemos uso de un middleware preparado para tal efecto, de forma que en un paso previo a la eliminación del recurso se comprobará que efectivamente quien ha realizado la petición dispone de los permisos

necesarios para hacerlo. Una característica interesante de los middlewares es que se pueden utilizar tantos como sean necesarios, de esta forma podemos hacer varias validaciones antes de llegar al propio controlador, por ejemplo: validar si el usuario esta registrado en el sistema y si dispone de los permisos apropiados para realizar un tipo de acción.

Los helpers son pequeñas porciones de código reunidas y organizadas en ficheros que permiten reutilizar cierta lógica en varios puntos del proyecto si se desea y a su vez mantener el código en general limpio y ordenado. Un ejemplo sería: durante el registro de un usuario es necesario crear un hash con su contraseña para ser almacenado en lugar del texto plano que envía el usuario; ese paso para crear el hash se externaliza en una función dentro de un fichero que trataremos como un helper.

5.2.3. Almacén de datos

Para agilizar el proceso optamos por uno de los nuevos modos de gestión de información alternativos a las bases de datos relacionales clásicas. MongoDB nos ha permitido acelerar el desarrollo de nuestras aplicaciones y centrarnos en las partes realmente importantes abstrayéndonos casi por completo de la tarea de trabajar con la interacción de un almacén de datos. Para hacer esta interacción solo era necesario crear un modelo de datos para identificar cada entidad de nuestro sistema y mediante el uso de una librería llamada Mongoose, realizar todo tipo de consultas a MongoDB.

5.2.4. Stellar Network

Se denomina Stellar Network al conjunto de funcionalidades que hacen posible el funcionamiento de la criptomononeda Lumens desarrollada por la organización Stellar. Esta tecnología es de código abierto, distribuida y de propiedad de la comunidad.

La estructura es simple, se divide en dos piezas. Por un lado cuentan con una API llamada Horizon que permite enviar transacciones a la red y verificar el estado de las cuentas de forma sencilla. Por otro lado tienen el llamado Stellar Core, un software ejecutado por diversos

servidores. Estos servidores son mantenidos por diferentes individuos y entidades. Su función principal es mantener una copia local del libro mayor, comunicarse y estar sincronizado con otras instancias de Stellar Core en la red.

Stellar cuenta con dos redes, una publica donde se hacen transferencias reales entre individuos y otra red denominada «horizon-testnet» que como su nombre indica es una red para hacer tests y que los Lumens que circulan en ella no tienen valor alguno. Esta red nos fue muy útil antes de empezar a hacer llamadas reales desde nuestra aplicación. Podíamos hacer pruebas de cualquier tipo sin ningún riesgo gracias a Stellar Laboratory, una herramienta web desarrollada por Stellar. En ella pudimos explorar los endpoints de la API, hacer transacciones, firmarlas y ver que se hacían efectivas en la o las respectivas cuentas.

Todo el desarrollo para este Trabajo de Fin de Grado se ha hecho contra la «horizon-testnet» debido a que nos permite conseguir un funcionamiento idéntico al que conseguiríamos en la red publica de Stellar pero sin costes ni riesgos. Si en algún momento se quisiera cambiar de la red de pruebas a la red publica tan solo habría que modificar una sola variable en toda la aplicación.

Nuestra implementación

Una de las partes más importantes del presente trabajo ha sido la de implementar un sistema de pago de recompensas haciendo uso de la red de Stellar. Gracias a Stellar hemos podido implantar un sistema de pagos seguros, rápidos y potentes.

Cada vez que un usuario se registra en la plataforma se crea una cuenta para el, que consiste en un par de claves; una secreta que servirá para firmar transacciones y otra pública que será la empleada para recibir pagos. Este sistema permite realizar un pago con solo conocer la clave pública de la cuenta a la que queremos enviar el pago.

Cuando un turista consigue validar todos los puntos que conforman una de las rutas propuestas, tiene la posibilidad de validar su recompensa: de esta forma el sistema verifica que efectivamente el turista ha pasado por todos los puntos y realiza el pago desde la cuenta de la institución que ha creado la recompensa.

De esta forma hemos conseguido un sistema autónomo donde varios usuarios pueden interactuar con la seguridad de que los pagos se realizan de forma correcta y eficáz.

Capítulo 6

Casos de uso

En esta sección se detallan las acciones que permite el sistema según el actor. Algunas de estas acciones son generales por lo que se han incluido todas en una sección a fin de no repetir para cada actor.

6.1. Generales

6.1.1. Crear cuenta

Todo usuario que quiera interactuar con el sistema necesita una cuenta de usuario. En este proceso se introducen los datos requerido y se selecciona el tipo de usuario deseado. Ver tabla 8.1

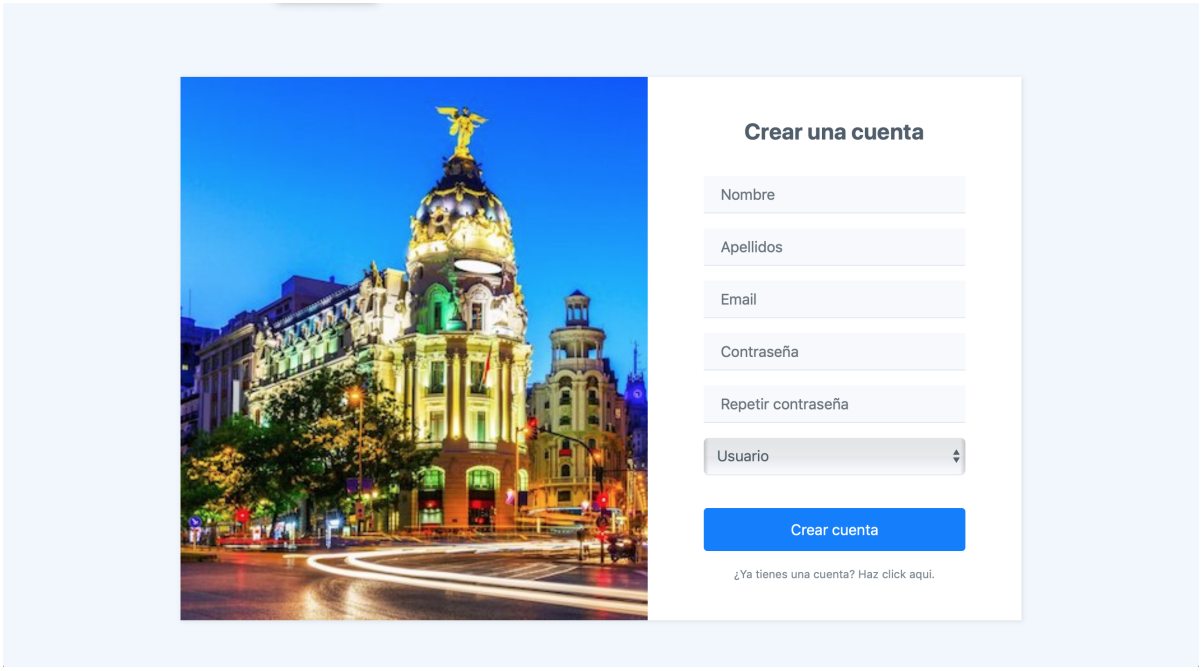

Figura 6.1: Creación de una cuenta

6.1.2. Acceso al sistema

Todo usuario que quiera interactuar con el sistema tiene que haber introducido sus credenciales antes. Mediante el proceso de login, el usuario introduce su nombre y contraseña y es validad por el sistema. Ver tabla 8.2

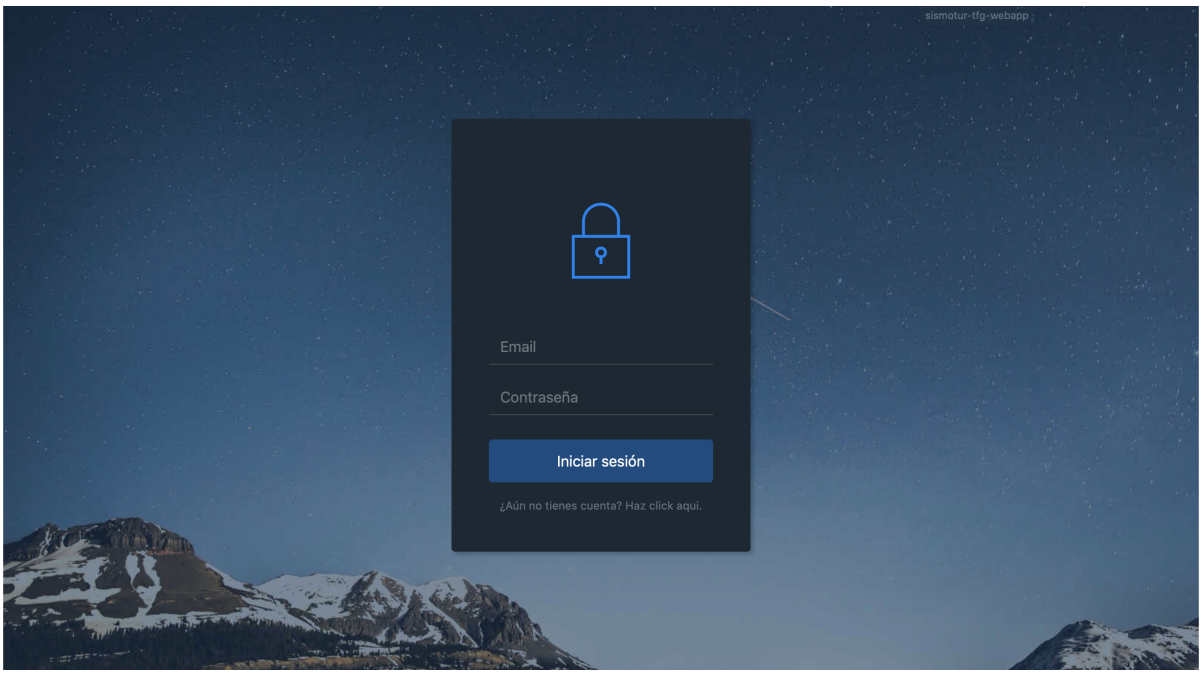

Figura 6.2: Acceso al sistema

6.1.3. Cerrar sesión

Cuando un usuario quiere dejar de usar la aplicación, puede cerrar la sesión actualmente abierta. Ver tabla 8.3

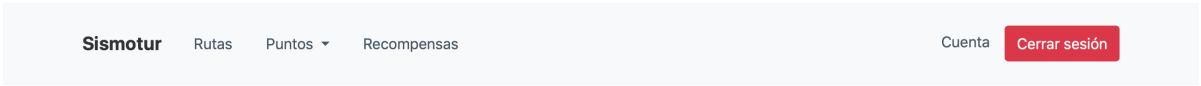

Figura 6.3: Cerrar la sesión

6.2. Turista

6.2.1. Suscribirse a una ruta

Cuando un turista llega a su destino turístico y se plantea realizar una de las rutas incluidas en el sistema lo primero que tiene que hacer es suscribirse a una de estas rutas, puede hacerlo desde el listado de rutas. Ver tabla 8.5

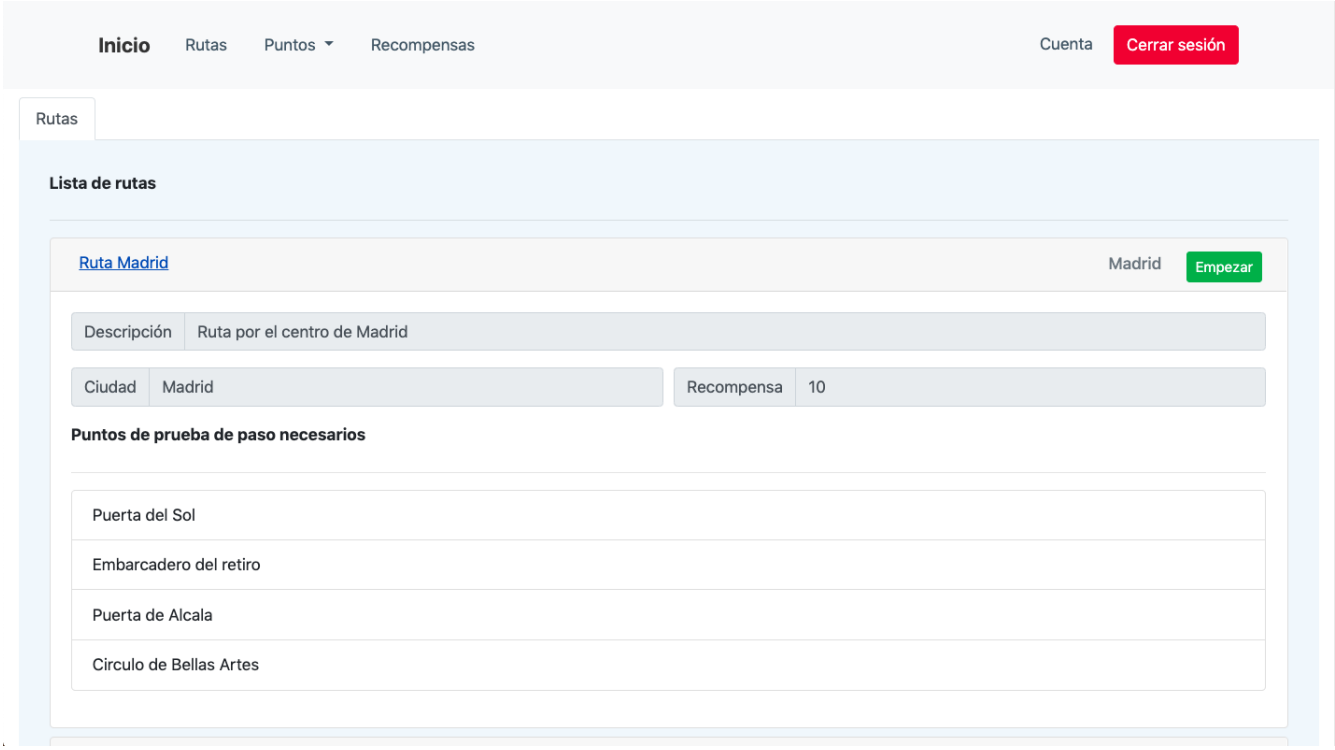

Figura 6.4: Suscribirse a una ruta

6.2.2. Validar punto de prueba de paso

A medida que va pasando por cada uno de los puntos que conforman la ruta, el turista puede ir validando mediante la aplicación su progreso. Puede seleccionar cada uno de estos puntos desde el listado de puntos de prueba de paso. Ver tabla 8.6

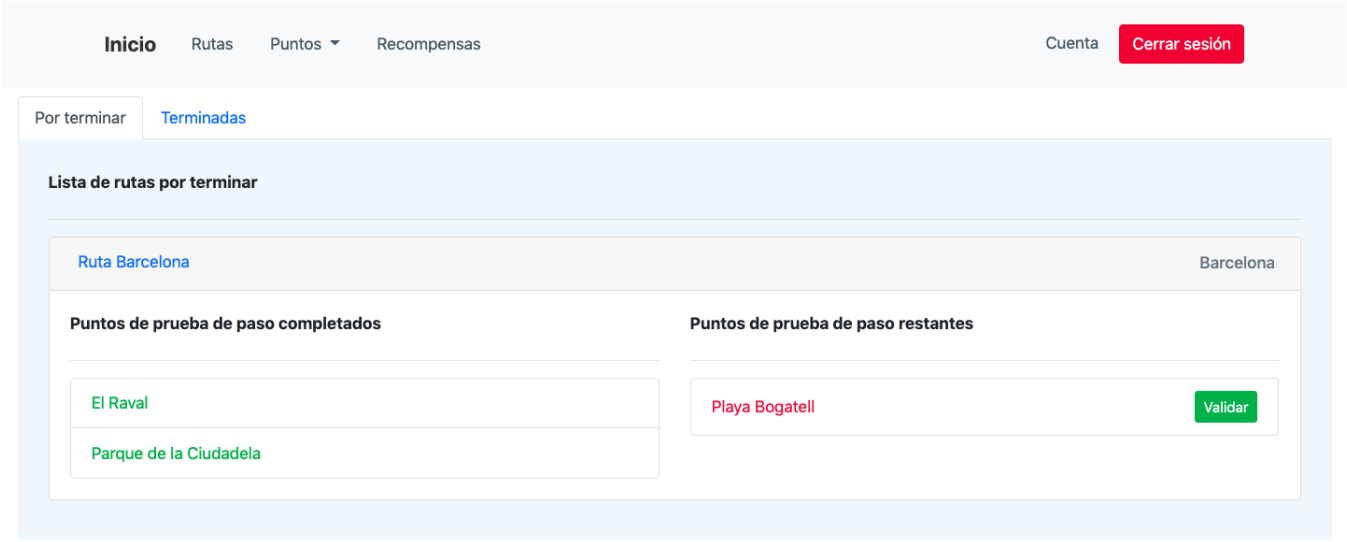

Figura 6.5: Validar punto de prueba de paso

6.2.3. Validar recompensa

Una vez que ha pasado por todos los puntos de prueba de paso y estos han sido correctamente validados, el turista querrá obtener su recomensa, para lo cual debe acudir a uno de los puntos autorizados disponibles y validad su recompensa con el hostelero, quien hará las comprobaciones oportunas. Ver tabla 8.10

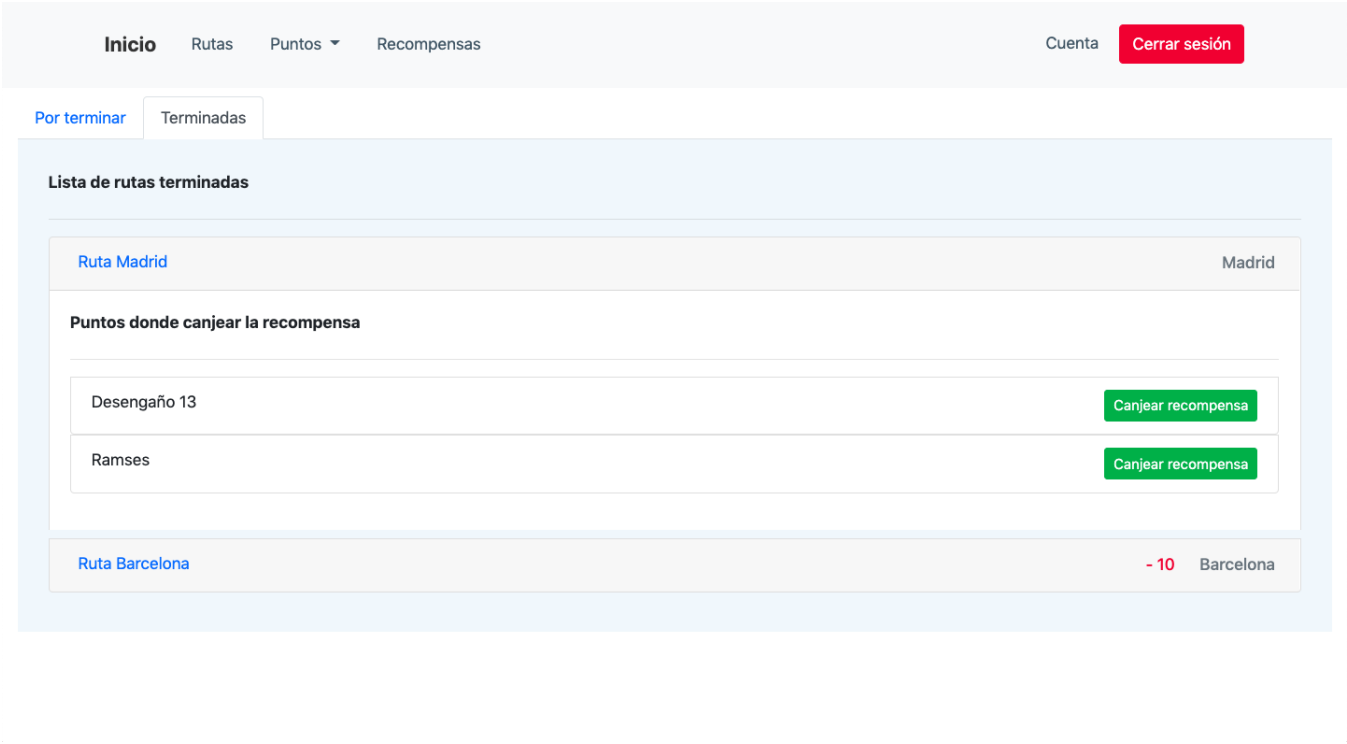

Figura 6.6: Validar recompensa

6.3. Hostelero

6.3.1. Crear nuevo punto de recompensa

El hostelero es el encargado de dar de alta su negocio como punto de recompensa, que es donde el usuario acudirá una vez ha terminado de recorrer una ruta. Ver tabla 8.13

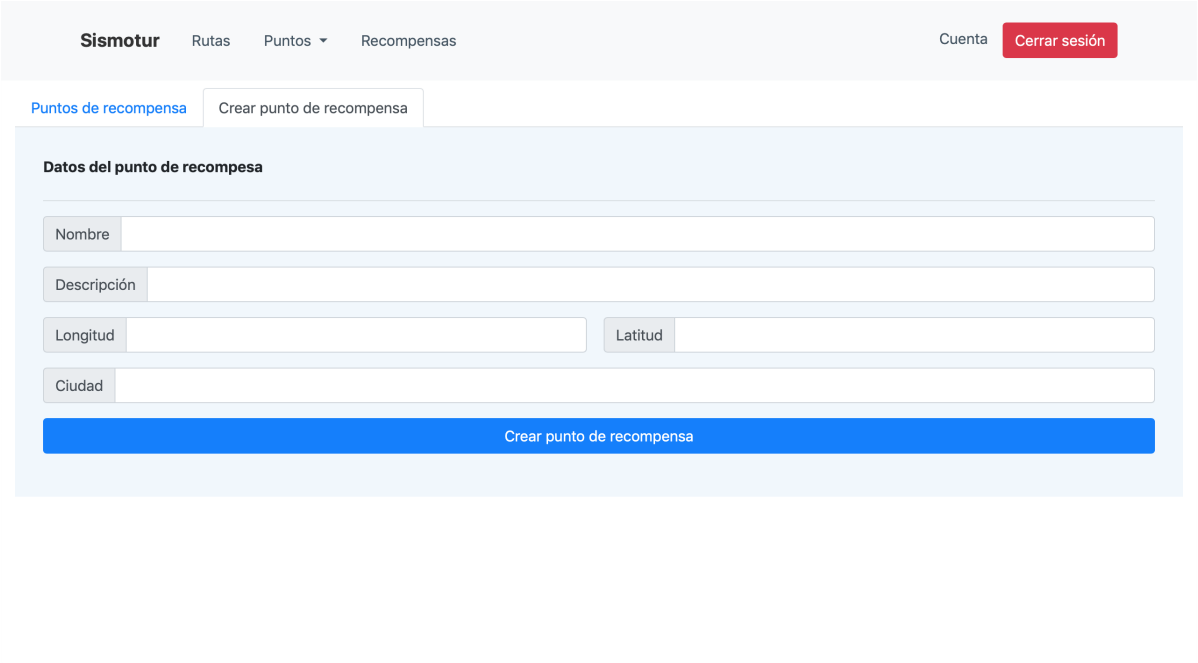

Figura 6.7: Crear punto de recompensa

6.3.2. Obtener listado de puntos de recompensa

El hostelero puede obtener un listado de los puntos de recompensa creados por el. Ver tabla 8.14

6.3.3. Borrar punto de recompensa

Cuando se deja de usar un punto de recompensa, puede ser eliminado del sistema. Ver tabla 8.16

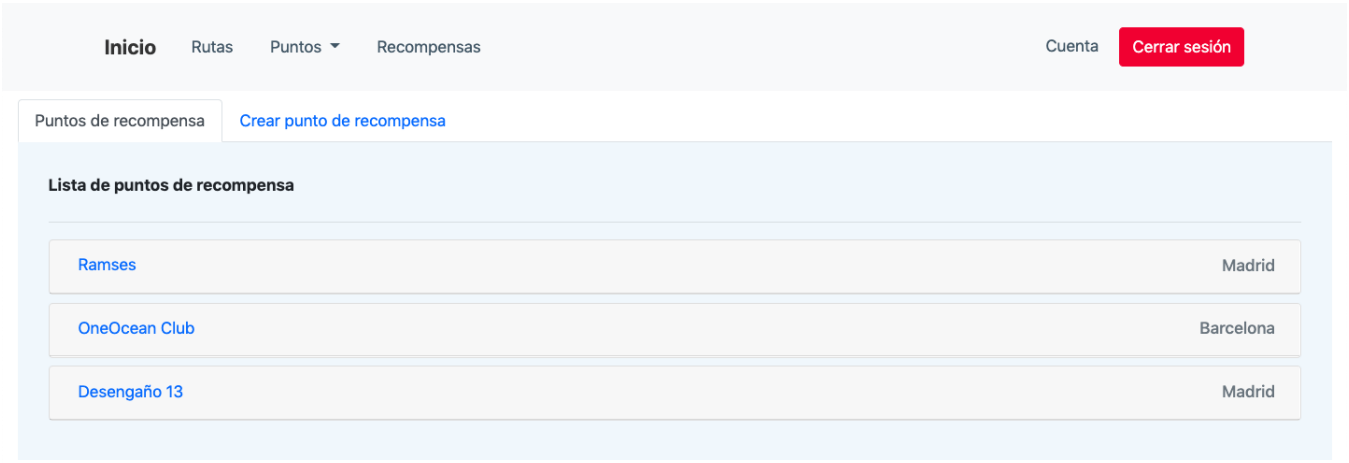

Figura 6.8: Listado puntos de recompensa

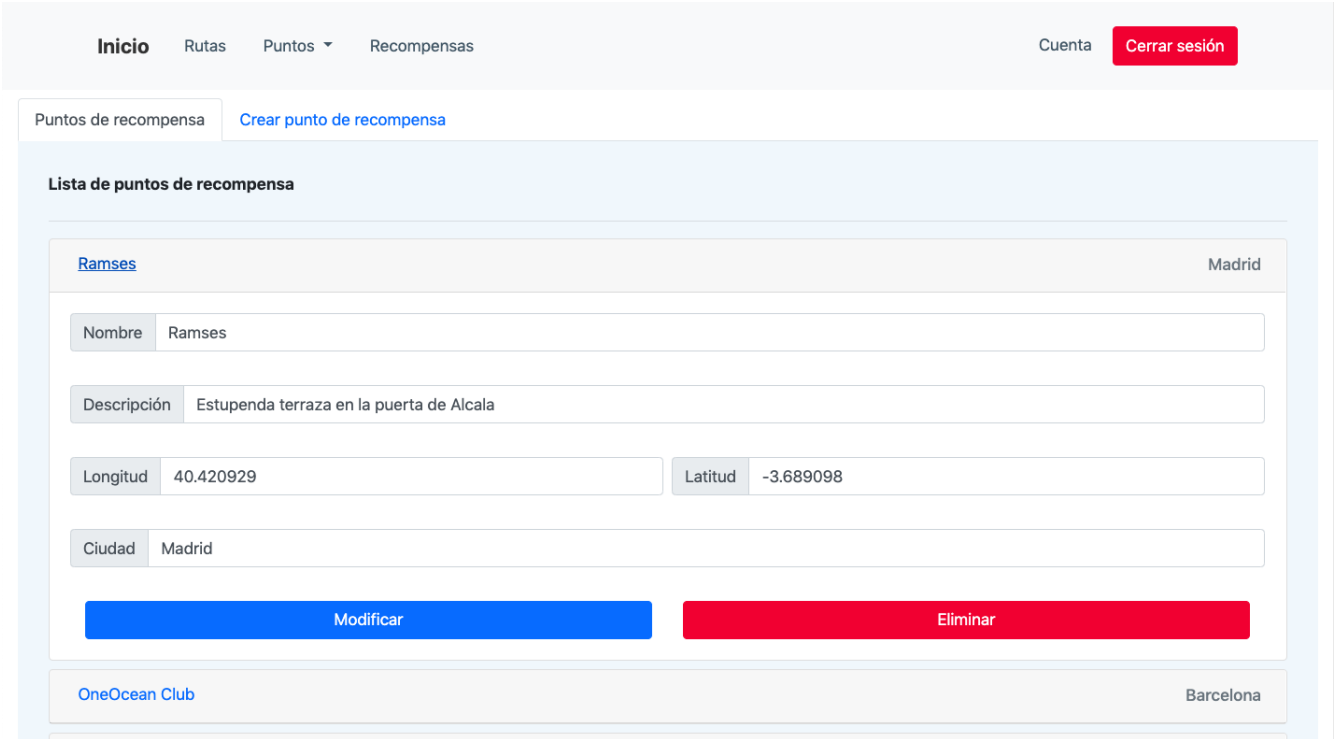

Figura 6.9: Borrar punto de recompensa

6.4. Institución

6.4.1. Crear un nuevo punto de prueba de paso

Para crear los puntos de prueba de paso que forman una ruta. Ver tabla 8.22

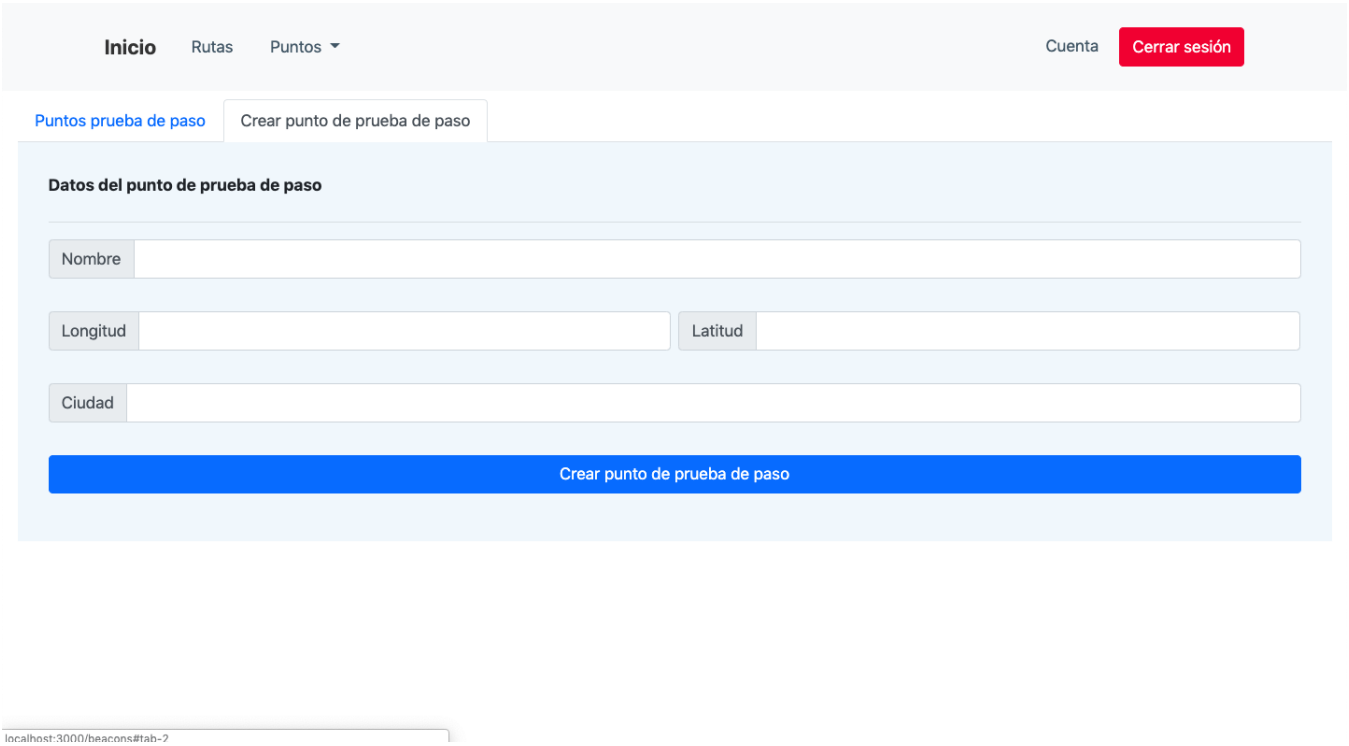

Figura 6.10: Crear un punto de prueba de paso

6.4.2. Obtener listado de puntos de prueba de paso

Se puede obtener un listado con todos los puntos de prueba de paso que forman una ruta. Ver tabla 8.23

6.4.3. Borrar punto de prueba de paso

Si algún punto deja de ser necesario por cualquier motivo, este puede ser eliminado del sistema. Ver tabla 8.25

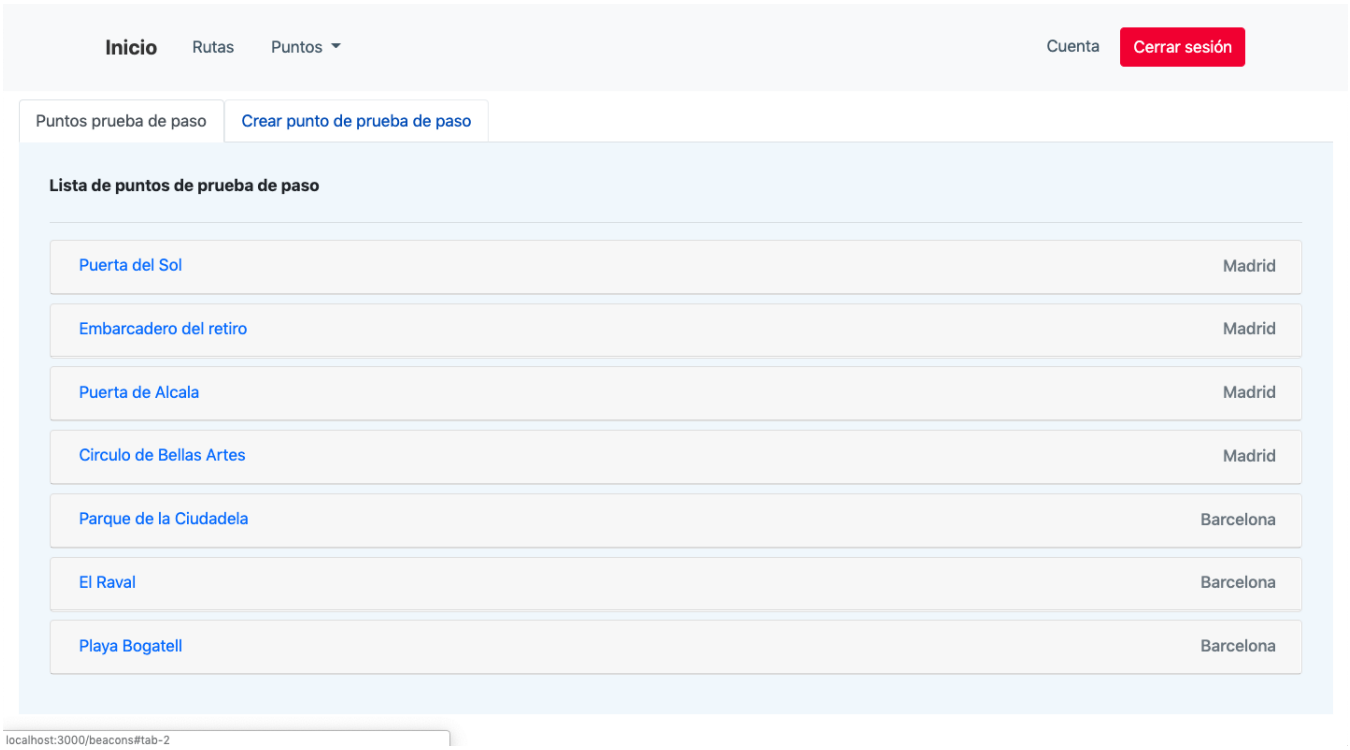

Figura 6.11: Obtener listado de puntos de prueba de paso

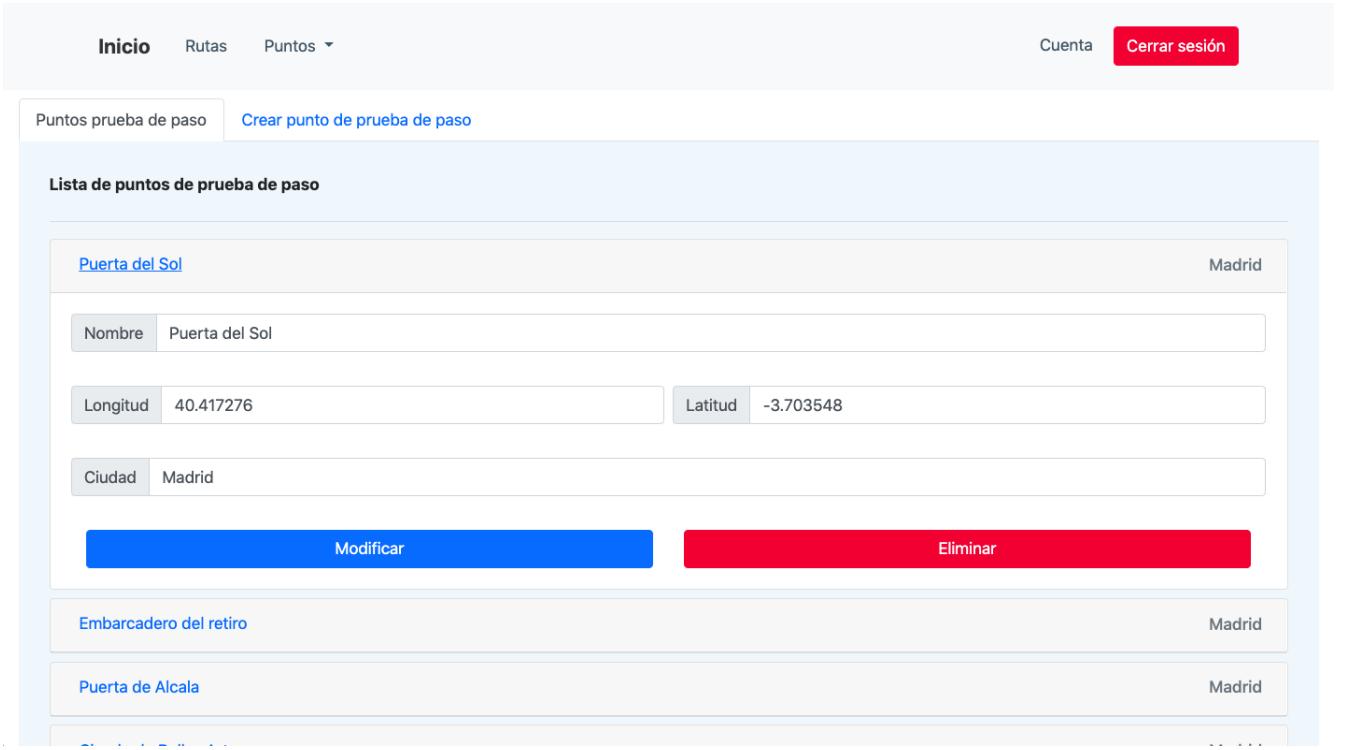

Figura 6.12: Borrar punto de prueba de paso

Capítulo 7

Conclusiones

7.1. Resumen

Ha sido muy interesante para nosotros plantear una solución a un problema real, ir un poco mas allá de lo que estamos acostumbrados en las asignaturas de la carrera.

En cuanto a las tecnologías utilizadas para el desarrollo de la aplicación no hemos tenido grandes problemas, o bien ya habíamos trabajado con ellas durante la carrera o uno de los dos sabia un poco mas y nos enseñábamos mutuamente. El desafío mas grande diríamos que ha sido en cuanto a las criptomonedas. Aunque ya estábamos un poco informados antes de empezar el proyecto respecto a este tema, el conocimiento que teníamos no era suficiente para poder llevar a cabo el proyecto. Estudiamos y comparamos tanto en características como a bajo nivel varios tipos de criptomonedas hasta llegar a Stellar. Es sin duda la que mejor se adaptaba al proyecto.

7.2. Competencias adquiridas

El desarrollo de este Trabajo de Fin de Grado nos ha permitido afianzar y adquirir una serie de nuevos conocimientos, no solo tecnológicos si no en relación con el mundo laboral.

- JavaScript fue uno de los requisitos para implementar el sistema, para nosotros no fue problema ya que habíamos trabajado antes con este lenguaje. Con esta base clara tuvimos que elegir una arquitectura que fuera modular y escalable, por eso decidimos separar la lógica en una API. Dado que era la primera vez que desarrollábamos este tipo de arquitectura podemos decir que nos ha permitido adquirir nuevas capacidades.
- La comunicación con Stellar Network se ha hecho a través de su API Horizon. Nunca habíamos trabajado con APIs de terceros, pero la documentación para desarrolladores que ofrece Stellar nos ha facilitado mucho esta tarea. Con esto también nos hemos dado cuenta de la importancia que tiene el desarrollo de una buena documentación cuando desarrollas una API que quieres que sea usada por otras personas. Aunque finalmente no usamos Ethereum como criptomoneda para el proyecto si que investigamos sobre la forma de implementarlo y hemos de decir que la documentación sobre su API no era tan extensa y detallada como la de Stellar.
- En cuanto a la parte laboral, podemos decir que nos ha sentado un poco los pies en la tierra. Durante la carrera hemos desarrollado proyectos con una base solida en cuanto a requisitos y funcionalidades. En cambio al enfrentarnos a un proyecto real y que en un principio no estaba totalmente definido nos hemos dado cuenta de la importancia que tiene hacer una buena ingeniería previa al desarrollo del producto, definir los requisitos y funcionalidades para evitar que en una etapa mas avanzada del proyecto surjan cambios que retrasen el desarrollo.

7.3. Trabajo futuro

Es este apartado describiremos una serie de ideas posibles para mejorar la aplicación.

En primer lugar, esta prueba de concepto se debería integrar en la aplicación Inventrip de Sismotur, tanto en su versión web como en iOS y en Android. Puesto que es para lo que esta pensada. Una vez integrada las posibilidades crecen de manera notable.

- Implementar un sistema de mapas y GPS. Google Maps Platform ofrece las APIs y los SDKs necesarios para poder llevar a cabo las siguientes funciones:
  - Ver las rutas sobre el mapa y que la aplicación nos pueda guiar desde donde estemos hasta el inicio de la ruta y de ahí hasta el final.
  - Ver todos los puntos de prueba de paso sobre el mapa, por si queremos explorar por nuestra cuenta y visitar alguno de ellos. La aplicación también nos guiaría hasta el punto elegido.
  - Ver todos los puntos de recompensa sobre el mapa y que la aplicación nos pueda guiar desde donde estemos hasta el bar, restaurante, mercado...
- Implementar un sistema de valoración para que los usuarios puedan dar su opinión acerca de las rutas realizadas para incentivar el uso de la aplicación.
- Dar visibilidad a las subvenciones que financian las recompensas de las rutas.
- Implementar el sistema para que el administrador pueda dar de alta y controlar quien tiene perfil de hostelero y quien no, dado que ahora no se tiene control sobre el como se registran los usuarios, ya sean administrador, hostelero o usuario.

7.4. Contribuciones

7.4.1. Jaime Tamames Hergueta

Lo primero que hicimos tanto Sergio como yo por recomendación de Felipe Santi, colaborador de Sismotur, fue leernos el libro «Mastering Bitcoin: Programming the Open Blockchain» de Andreas M. Antonopoulos. Esta lectura nos dio todo el conocimiento previo de como funciona Bitcoin y la tecnología blockchain.

Mas tarde tras varias reuniones con Felipe y después de descartar criptomonedas llegamos a Stellar, una criptomoneda que cumplía con los requisitos de Sismotur y que se podía implementar al proyecto. Tras varias pruebas en la herramienta Stellar Laboratory, su red de pruebas, confirmamos su uso en el proyecto.

Horizon, la API de Stellar, ofrece su kit de desarrollo software (SDK) en varios lenguajes. Junto a Felipe decidimos que lo mejor por mantenibilidad era desarrollar todo el proyecto usando JavaScript, lenguaje soportado por Horizon.

Una vez tuvimos clara la arquitectura que íbamos a seguir para desarrollar el proyecto y nos dividimos las tareas acorde a lo que a cada uno le gustaba mas, nos pusimos en marcha con el desarrollo.

Yo me encargue de la webapp, es decir el frontend. Al principio, aunque teníamos clara la arquitectura y las tecnologías que íbamos a usar para desarrollar el proyecto, no estaban muy definidas las funcionalidades finales de la aplicación. Cosa de la que me encargue junto a Felipe.

El frontend esta desarrollado, como es de esperar, en HTML en su versión 5 junto a la librería CSS de Bootstrap. Aunque tampoco he trabajado mucho con esta librería me resulta muy fácil e intuitiva para desarrollar interfaces web. Según íbamos avanzando, Sergio con el backend y yo con el frontend, a menudo surgían inconvenientes o pequeños cambios de requisitos que hemos podido solucionar sin demorar mucho el proyecto.

En cuanto a la memoria decidimos hacerla con LaTeX, sin el uso de ninguna plantilla. Por lo

que hemos tenido que aprender a como generar el documento completo con todas sus partes y la estructura que requiere.

A diferencia de la parte de desarrollo, para escribir esta memoria, no hemos hecho un reparto previo. Ambos éramos conocedores del total del proyecto, por lo que hemos ido aportando con las partes que mas cómodos nos encontrábamos. A continuación, detallo las secciones en las que mas he trabajado:

- Agradecimientos
- Glosario
- Resumen y estructura
- Dominio
- Motivación y objetivos
- Estado del arte
- Descripción funcional
- Arquitectura
- Casos de uso
- Conclusiones

Quiero aprovechar este apartado que es solo «mío» para agradecer a mi compañero Sergio la oportunidad de desarrollar este Trabajo de Fin de Grado juntos. Nos conocemos casi desde el principio de la carrera y siempre ha sido un placer trabajar contigo. Espero encontrar en el futuro compañeros en el ámbito laboral con los que me entienda tan bien como contigo. Lo dicho, un placer.

7.4.2. Sergio Pino Holgado

Desde el inicio del proyecto y dadas nuestras situaciones personales decidimos seguir un enfoque de trabajo basado en «microtareas» y de este modo poder hacer un reparto equitativo del trabajo total. Es por ello que ambos hemos tenido la oportunidad de aportar de igual forma al trabajo. Como si de una empresa dividida en departamentos se tratase, hicimos una división lógica para el desarrollo del prototipo en dos piezas fundamentales: la webapp y la API.

Por supuesto, antes de llegar a la idea de construir una API tuvimos que pasar por un proceso de diseño en el que determinar que componentes necesitabamos para dar forma a nuestra idea.

Una vez teníamos claro el diseño de nuestra arquitectura y decidimos quién sería responsable de cada parte, me puse manos a la obra para ver que tecnologías eran las más aptas. Primeramente se debatió la posibilidad de desarrollar usando el lenguaje de programación de Google: Golang (Go) por disponer de librerías oficiales para Stellar, aunque al final optamos por usar JavaScript.

Por la parte del almacenamiento de datos en un principio estudié la posibilidad de hacerlo con MySQL, que es la plataforma con la que más experiencia tenía pero me decanté por MongoDB por su facilidad de implementación con NodeJS, la agilidad a la hora de trabajar y que se trataba de una tecnología relativamente nueva que me iba a aportar mucho más conocimiento y valor que MySQL.

Gran parte del tiempo lo dediqué a investigar como funcionaba Stellar, como realizar las transacciones básicas y como implementarlo en nuestro sistema, haciendo uso de la red de pruebas que tienen a disposición de los desarrolladores.

Una parte muy importante del trabajo ha sido elaborar una memoria tratando de reunir en ella todos los puntos importantes por los que hemos ido pasando a lo largo de estos meses. Muchas cosas se nos han quedado en el tintero, siendo el principal motivo la cantidad de tiempo disponible para dar forma a nuestra idea. Al igual que el resto del trabajo tratamos de repartir la memoria en partes iguales.

Es importante resaltar que una parte del desarrollo de la memoria lo destinamos al aprendizaje de LaTeX ya que todo el documento lo hemos realizado nosotros sin el uso de una plantilla.

A continuación detallo un listado de los apartados a los que principalmente he aportado, aunque siempre el contenido de uno ha sido revisado y corregido y/o ampliado por el otro. Puesto que ambos hemos aportado por igual y hemos trabajado conjuntamente, tenemos el contexto necesario para realizar cualquier parte de la memoria; no seguimos ningún criterio para hacer esta separación de tareas, simplemente ibamos añadiendo contenido según nuestra disponibilidad.

- Proceso de desarrollo
- Requisitos del sistema
- Arquitectura
- Casos de uso
- Conclusiones
- Apéndice sobre como desplegar el sistema
- Apéndice con los endpoints de la API

Quiero remarcar que todo el trabajo realizado, aun habiendo cierta especialización en algunas partes, como la API o la webapp, ambos hemos estado trabajando por igual y de forma autónoma, es decir, no hemos tenido que «perseguirnos» para finalizar nuestra parte.

7.5. Summary

It has been very interesting for us to propose a solution to a real problem, go a little beyond what we are used to in the subjects of the career.

Regarding the technologies used for the development of the application we have not had any major problems, either we had already worked with them during the career or one of the two knew a little more and we taught each other. The biggest challenge we would say has been in terms of cryptocurrencies. Although we were already a little informed before starting the project on this subject, the knowledge we had was not enough to carry out the project. We study and compare both in characteristics and at low level various types of cryptocurrencies until we reach Stellar. It is undoubtedly the one that best suited the project.

7.6. Knowledge acquired

The development of this Final Degree Project has allowed us to consolidate and acquire a series of new knowledge, not only technological but in relation to the world of work.

- JavaScript was one of the requirements to implement the system, for us it was not a problem since we had worked with this language before. With this base we had to choose an architecture that was modular and scalable, that's why we decided to separate the logic in an API. Since it was the first time we developed this type of architecture we can say that it has allowed us to acquire new capabilities.
- Communication with Stellar Network has been done through its Horizon API. We had never worked with third-party APIs, but the developer documentation offered by Stellar has made this task much easier. With this we have also realized the importance of developing good documentation when you develop an API that you want to be used by other people. Although finally we don't use Ethereum as cryptocurrency for the project if we investigate on how to implement it and we have to say that the documentation on its API was not as extensive and detailed as Stellar's.
- Regarding the work part, we can say that our feet have been sitting on the ground a bit. During the career we have developed projects with a solid base in terms of requirements and functionalities. On the other hand, when facing a real project and that at first was not totally defined, we have realized the importance of doing a good engineering prior to the development of the product, defining the requirements and functionalities to avoid changes in a more advanced stage of the project that delay development.

7.7. Future work

In this section we will describe a series of possible ideas to improve the application.

First, this proof of concept should be integrated into the Inventrip application of Sismotur, in its web version, in iOS and Android. It was thought for that. Once integrated, the possibilities grow significantly.

- Implement a map and GPS system. Google Maps Platform offers the necessary APIs and SDKs to carry out the following functions:
  - See the routes on the map and the application can guide us from where we are until the beginning of the route and from there to the end.
  - See all the proof of pass points on the map, in case we want to explore on our own and visit any of them. The application would also guide us to the chosen point.
  - See all the reward points on the map and the application can guide us from where we are to the bar, restaurant, market ...
- Implement a valuation system so that users can comment on the routes carried out to encourage the use of the application.
- Give visibility to the grants that finance the rewards of the routes.
- Implement the system so that the administrator can register and control who has a hotelier profile and who does not, given that now there is no control over how users register, whether they are an administrator, a hotelier or a user.

Capítulo 8

Apéndices

8.1. Requisitos funcionales

A continuación detallamos los requisitos funcionales de cada actor para tener claro que puede hacer cada uno dentro de la aplicación. Teniendo claro esto sera mas fácil entender la funcionalidad de la aplicación completa.

8.1.1. Generales

A continuación se detallan los requisitos generales, que aplican de igual forma para todos los usuarios.

Cuadro 8.1: Crear cuenta

Crear cuenta

Descripción	El usuario quiere crear una cuenta para poder interactuar con el sistema.
Precondición	–
Secuencia	1. El usuario accede a la pagina de creación de cuenta. 2. El usuario introduce todos los datos del formulario. 3. El sistema verifica que todos los datos son válidos. 4. El sistema crea la nueva cuenta y la almacena.
Errores	1. Si los datos introducidos por el usuario son incorrectos, la cuenta no se crea y se informa al usuario.
Postcondición	La cuenta es creada y almacenada.
Notas	Existen tres tipos de cuenta de usuario: ▪ Turista ▪ Hostelero ▪ Institución

Cuadro 8.2: Acceder al sistema

Acceder al sistema

Descripción	El usuario quiere acceder al sistema con sus credenciales.
Precondición	El usuario está registrado en el sistema.
Secuencia	1. El usuario accede a la página de login. 2. El usuario introduce sus datos de acceso. 3. El sistema verifica que el usuario existe y los datos son correctos y realiza el proceso de login.
Errores	1. Si los datos introducidos por el usuario no son correctos el sistema devuelve a la pagina de login.
Postcondición	Se genera un nuevo token que es enviado al usuario para que pueda hacer peticiones de forma validada.
Notas	

Cuadro 8.3: Cerrar sesión

Cerrar sesión

Descripción	El usuario quiere terminar la sesión.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario pulsa el botón para terminar la sesión. 2. El sistema elimina la sesión del usuario.
Errores	–
Postcondición	La sesión es eliminada del sistema.
Notas	

8.1.2. Turista

Cuadro 8.4: Consultar rutas disponibles

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Cuadro 8.5: Suscribirse a una ruta

Suscribirse a una ruta

Descripción	El usuario quiere suscribirse a una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El usuario selecciona una ruta disponible para suscribirse. 3. El sistema registra la ruta que ha elegido el usuario.
Errores	–
Postcondición	La ruta elegida por el usuario queda relacionada con este.
Notas	

Cuadro 8.6: Validar punto de prueba de paso

Validar punto de prueba de paso

Descripción	El usuario quiere validar un punto de prueba de paso al que ha llegado.
Precondición	El usuario está logeado en el sistema y tiene el código de validación del punto de prueba de paso.
Secuencia	1. El usuario accede a la vista de las rutas a las que está suscrito. 2. El usuario selecciona el punto de prueba de paso y pulsa el botón para validar. 3. El sistema verifica que los datos sean correctos y registra la validación del punto de prueba de paso.
Errores	1. Si el usuario no se encuentra próximo al punto de prueba de paso se devuelve un error. 2. Si el usuario ha enviado erroneamente los datos se devuelve un error. 3. Si el usuario ya ha intentado validar este punto anteriormente se devuelve un error.
Postcondición	–
Notas	

Cuadro 8.7: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.8: Consultar puntos de recompensa

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Cuadro 8.9: Consultar cartera

Consultar cartera

Descripción	El usuario quiere conocer el estado de su cartera.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de cartera. 2. El sistema busca los datos y los muestra al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.10: Validar recompensa

Validar recompensa

Descripción	El usuario quiere validar una recompensa que ya ha completado.
Precondición	El usuario está logeado en el sistema y ha completado la ruta.
Secuencia	1. El usuario accede a la vista de rutas y muestra al hostelero el código. 2. El hostelero valida con el sistema el código proporcionado por el usuario. 3. El hostelero entrega la recompensa al usuario.
Errores	1. Si el usuario intenta validar una recompensa incompleta o invalida, el hostelero no le entregará el obsequio.
Postcondición	El usuario recibe su recompensa.
Notas	

8.1.3. Hostelero

Cuadro 8.11: Consultar rutas disponibles

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Cuadro 8.12: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.13: Crear punto de recompensa

Crear punto de recompensa

Descripción	El usuario quiere crear un nuevo punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo hostelero.
Secuencia	1. El usuario accede a la vista para crear un nuevo punto de recompensa. 2. El usuario introduce los datos y envía el formulario. 3. El sistema valida que los datos introducidos sean correctos y crea el nuevo punto de recompensa.
Errores	1. Si alguno de los datos es incorrecto el sistema devuelve un error.
Postcondición	El punto de recompensa está creado en el sistema.
Notas	

Cuadro 8.14: Consultar puntos de recompensa

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Cuadro 8.15: Actualizar punto de recompensa

Actualizar punto de recompensa

Descripción	El usuario quiere modificar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere modificar e introduce los datos a modificar. 3. El sistema valida que los nuevos datos sean correctos y hace las modificaciones.
Errores	1. Si alguno de los datos es incorrecto o no se puede crear por algún motivo se devuelve un error.
Postcondición	El punto de recompensa queda actualizado con los nuevos datos.
Notas	

Cuadro 8.16: Borrar punto de recompensa

Borrar punto de recompensa

Descripción	El usuario quiere eliminar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere eliminar. 3. El sistema elimina el punto de recompensa del sistema.
Errores	1. Si el punto de recompensa tiene alguna relación activa en el sistema se devuelve un error.
Postcondición	El punto de recompensa queda eliminado del sistema.
Notas	

Cuadro 8.17: Validar recompensa de un turista

Validar recompensa de un turista

Descripción	El usuario quiere validar la recompensa de un turista.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para validar el punto de recompensa. 2. El usuario escanea el código del turista y envía la petición al servidor. 3. El sistema valida el código.
Errores	1. Si el código no es correcto se devuelve un error. 2. Si el código ya ha sido validado anteriormente se devuelve un error.
Postcondición	El código del usuario queda validado.
Notas	

8.1.4. Institución

Cuadro 8.18: Crear ruta

Crear ruta

Descripción	El usuario quiere crear una nueva ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear una ruta. 2. El usuario introduce los datos del formulario y envía la petición al servidor. 3. El sistema valida y crea la ruta.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda creada en el sistema.
Notas	

Cuadro 8.19: Consultar rutas

Consultar rutas

Descripción	El usuario quiere consultar las rutas creadas.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de las rutas.
Errores	–
Postcondición	–
Notas	

Cuadro 8.20: Actualizar ruta

Actualizar ruta

Descripción	El usuario quiere actualizar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda actualizada en el sistema.
Notas	

Cuadro 8.21: Borrar ruta

Borrar ruta

Descripción	El usuario quiere borrar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere borrar. 3. El sistema elimina la ruta del sistema.
Errores	1. Si la ruta no se puede eliminar se devuelve un error.
Postcondición	La ruta queda eliminada del sistema.
Notas	

Cuadro 8.22: Crear punto de prueba de paso

Crear punto de prueba de paso

Descripción	El usuario quiere crear un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear un punto de prueba de paso. 2. El usuario introduce los datos del nuevo punto y envia la petición al servidor. 3. El servidor valida los datos y almacena el nuevo punto.
Errores	1. Si el punto no puede ser creado se devuelve un error.
Postcondición	El punto queda creado en el sistema.
Notas	

Cuadro 8.23: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere consultar los puntos de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista con el listado de puntos de prueba de paso.
Errores	–
Postcondición	–
Notas	

Cuadro 8.24: Actualizar punto de prueba de paso

Actualizar punto de prueba de paso

Descripción	El usuario quiere actualizar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si el punto de prueba de paso no es validado devuelve un error.
Postcondición	El punto de prueba de paso queda actualizado en el sistema.
Notas	

Cuadro 8.25: Borrar punto de prueba de paso

Borrar punto de prueba de paso

Descripción	El usuario quiere borrar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere borrar. 3. El sistema elimina el punto de prueba de paso del sistema.
Errores	1. Si el punto de prueba de paso no se puede eliminar se devuelve un error.
Postcondición	El punto de prueba de paso queda eliminado del sistema.
Notas	

8.2. Repositorios

Todo el código desarrollado con el prototipo del proyecto se encuentra de forma pública en dos repositorios de GitHub.

- **API** https://github.com/tfg-ucm/api
- **Webapp** https://github.com/tfg-ucm/webapp

8.3. Como desplegar

8.3.1. Preparación del entorno

Para poder desplegar el sistema completo es necesario hacerlo por separado para cada una de las piezas que lo forman. La webapp y la API+almacén son «independientes» entre sí para desplegar ya que es posible que este funcionando la webapp pero la api no esté disponible; no es así en el caso de la API y el almacén de datos, que sí es necesario que se ejecute antes el almacén que la API, pues en su inicio tratará de conectarse para poder realizar el resto de funciones.

El orden propuesto para desplegar es el siguiente:

1. Almacén de datos
2. API
3. WebApp

Almacén de datos

1. Ejecución de mongod con un fichero en concreto `mongod --dbpath ~/mongodb/data/db`

API

1. Instalación de dependencias: `npm install`
2. Ejecución de la API: `npm start`

WebApp

1. Instalación de dependencias: `npm install`
2. Ejecución de la WebApp: `npm start`

8.4. Descripción funcional

8.4.1. Actores del sistema

En la aplicación existen tres tipos de usuarios muy diferenciados.

El turista, que es el usuario final de la aplicación, es decir quien disfruta de todas sus posibilidades. Buscar rutas, subscribirse a ellas para mas tarde poder validar los puntos de prueba de paso, buscar puntos donde validar su recompensa y por ultimo validarla.

El hostelero, es un usuario intermedio, esto quiere decir que se beneficia de la aplicación pero a su vez hace posible que el usuario final pueda disfrutarla. Este actor podrá crear puntos donde validar recompensas y validar las recompensas de los turistas que lo soliciten.

La institución, es un usuario de control y administración de la aplicación, hace posible y da el servicio al hostelero y el turista. Tendrá total control en cuanto a crear, modificar y eliminar rutas, puntos de prueba de paso y puntos de recompensa.

8.4.2. Requisitos del sistema

A continuación detallamos los requisitos funcionales de cada actor para tener claro que puede hacer cada uno dentro de la aplicación. Teniendo claro esto sera mas fácil entender la funcionalidad de la aplicación completa.

Generales

A continuación se detallan los requisitos generales, que aplican de igual forma para todos los usuarios.

Crear cuenta

Descripción	El usuario quiere crear una cuenta para poder interactuar con el sistema.
Precondición	–
Secuencia	1. El usuario accede a la pagina de creación de cuenta. 2. El usuario introduce todos los datos del formulario. 3. El sistema verifica que todos los datos son válidos. 4. El sistema crea la nueva cuenta y la almacena.
Errores	1. Si los datos introducidos por el usuario son incorrectos, la cuenta no se crea y se informa al usuario.
Postcondición	La cuenta es creada y almacenada.
Notas	Existen tres tipos de cuenta de usuario: ▪ Turista ▪ Hostelero ▪ Institución

Acceder al sistema

Descripción	El usuario quiere acceder al sistema con sus credenciales.
Precondición	El usuario está registrado en el sistema.
Secuencia	1. El usuario accede a la página de login. 2. El usuario introduce sus datos de acceso. 3. El sistema verifica que el usuario existe y los datos son correctos y realiza el proceso de login.
Errores	1. Si los datos introducidos por el usuario no son correctos el sistema devuelve a la pagina de login.
Postcondición	Se genera un nuevo token que es enviado al usuario para que pueda hacer peticiones de forma validada.
Notas	

Cerrar sesión

Descripción	El usuario quiere terminar la sesión.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario pulsa el botón para terminar la sesión. 2. El sistema elimina la sesión del usuario.
Errores	–
Postcondición	La sesión es eliminada del sistema.
Notas	

Turista

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Suscribirse a una ruta

Descripción	El usuario quiere suscribirse a una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El usuario selecciona una ruta disponible para suscribirse. 3. El sistema registra la ruta que ha elegido el usuario.
Errores	–
Postcondición	La ruta elegida por el usuario queda relacionada con este.
Notas	

Validar punto de prueba de paso

Descripción	El usuario quiere validar un punto de prueba de paso al que ha llegado.
Precondición	El usuario está logeado en el sistema y tiene el código de validación del punto de prueba de paso.
Secuencia	1. El usuario accede a la vista de las rutas a las que está suscrito. 2. El usuario selecciona el punto de prueba de paso y pulsa el botón para validar. 3. El sistema verifica que los datos sean correctos y registra la validación del punto de prueba de paso.
Errores	1. Si el usuario no se encuentra próximo al punto de prueba de paso se devuelve un error. 2. Si el usuario ha enviado erroneamente los datos se devuelve un error. 3. Si el usuario ya ha intentado validar este punto anteriormente se devuelve un error.
Postcondición	–
Notas	

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Consultar cartera

Descripción	El usuario quiere conocer el estado de su cartera.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de cartera. 2. El sistema busca los datos y los muestra al usuario.
Errores	–
Postcondición	–
Notas	

Validar recompensa

Descripción	El usuario quiere validar una recompensa que ya ha completado.
Precondición	El usuario está logeado en el sistema y ha completado la ruta.
Secuencia	1. El usuario accede a la vista de rutas y muestra al hostelero el código. 2. El hostelero valida con el sistema el código proporcionado por el usuario. 3. El hostelero entrega la recompensa al usuario.
Errores	1. Si el usuario intenta validar una recompensa incompleta o invalida, el hostelero no le entregará el obsequio.
Postcondición	El usuario recibe su recompensa.
Notas	

Hostelero

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Crear punto de recompensa

Descripción	El usuario quiere crear un nuevo punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo hostelero.
Secuencia	1. El usuario accede a la vista para crear un nuevo punto de recompensa. 2. El usuario introduce los datos y envía el formulario. 3. El sistema valida que los datos introducidos sean correctos y crea el nuevo punto de recompensa.
Errores	1. Si alguno de los datos es incorrecto el sistema devuelve un error.
Postcondición	El punto de recompensa está creado en el sistema.
Notas	

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Actualizar punto de recompensa

Descripción	El usuario quiere modificar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere modificar e introduce los datos a modificar. 3. El sistema valida que los nuevos datos sean correctos y hace las modificaciones.
Errores	1. Si alguno de los datos es incorrecto o no se puede crear por algún motivo se devuelve un error.
Postcondición	El punto de recompensa queda actualizado con los nuevos datos.
Notas	

Borrar punto de recompensa

Descripción	El usuario quiere eliminar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere eliminar. 3. El sistema elimina el punto de recompensa del sistema.
Errores	1. Si el punto de recompensa tiene alguna relación activa en el sistema se devuelve un error.
Postcondición	El punto de recompensa queda eliminado del sistema.
Notas	

Validar recompensa de un turista

Descripción	El usuario quiere validar la recompensa de un turista.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para validar el punto de recompensa. 2. El usuario escanea el código del turista y envía la petición al servidor. 3. El sistema valida el código.
Errores	1. Si el código no es correcto se devuelve un error. 2. Si el código ya ha sido validado anteriormente se devuelve un error.
Postcondición	El código del usuario queda validado.
Notas	

Institución

Crear ruta

Descripción	El usuario quiere crear una nueva ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear una ruta. 2. El usuario introduce los datos del formulario y envía la petición al servidor. 3. El sistema valida y crea la ruta.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda creada en el sistema.
Notas	

Consultar rutas

Descripción	El usuario quiere consultar las rutas creadas.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de las rutas.
Errores	–
Postcondición	–
Notas	

Actualizar ruta

Descripción	El usuario quiere actualizar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda actualizada en el sistema.
Notas	

Borrar ruta

Descripción	El usuario quiere borrar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere borrar. 3. El sistema elimina la ruta del sistema.
Errores	1. Si la ruta no se puede eliminar se devuelve un error.
Postcondición	La ruta queda eliminada del sistema.
Notas	

Crear punto de prueba de paso

Descripción	El usuario quiere crear un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear un punto de prueba de paso. 2. El usuario introduce los datos del nuevo punto y envia la petición al servidor. 3. El servidor valida los datos y almacena el nuevo punto.
Errores	1. Si el punto no puede ser creado se devuelve un error.
Postcondición	El punto queda creado en el sistema.
Notas	

Consultar puntos de prueba de paso

Descripción	El usuario quiere consultar los puntos de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista con el listado de puntos de prueba de paso.
Errores	–
Postcondición	–
Notas	

Actualizar punto de prueba de paso

Descripción	El usuario quiere actualizar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si el punto de prueba de paso no es validado devuelve un error.
Postcondición	El punto de prueba de paso queda actualizado en el sistema.
Notas	

Borrar punto de prueba de paso

Descripción	El usuario quiere borrar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere borrar. 3. El sistema elimina el punto de prueba de paso del sistema.
Errores	1. Si el punto de prueba de paso no se puede eliminar se devuelve un error.
Postcondición	El punto de prueba de paso queda eliminado del sistema.
Notas	

8.5. API endpoints

8.5.1. Cuentas

Crear cuenta

Verbo	`POST`
Endpoint	`/api/v1/users`
Descripción	Se crea una nueva cuenta de usuario con los datos recibidos siempre que estos sean correctos.

Consultar cuenta

Verbo	`GET`
Endpoint	`/api/v1/users/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve la cuenta.

Modificar cuenta

Verbo	`PUT`
Endpoint	`/api/v1/users/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar cuenta

Verbo	`DELETE`
Endpoint	`/api/v1/users/:id`
Descripción	Si la peticion tiene los permisos necesarios, bien porque un admin quiera eliminar o un usuario quiera eliminar su propia cuenta.

8.5.2. Autenticación

Login

Verbo	`POST`
Endpoint	`/api/v1/users/login`
Descripción	Recibe los datos introducidos por el usuario y los valida con los existentes en la DB. Devuelve el objeto con el token que sera almacenado en el navegador del usuario y con la que ”firmara”las peticiones que haga en el futuro a la API.

Logout

Verbo	`POST`
Endpoint	`/api/v1/users/logout`
Descripción	Elimina la sesion del almacen de datos.

8.5.3. Puntos de prueba de paso

Crear punto

Verbo	`POST`
Endpoint	`/api/v1/beacons`
Descripción	Se crea un nuevo punto de prueba de paso con los datos necesarios del beacon y su localizacion siempre que estos sean correctos.

Consultar punto

Verbo	`GET`
Endpoint	`/api/v1/beacons/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de puntos

Verbo	`GET`
Endpoint	`/api/v1/beacons`
Descripción	Obtiene un listado con los puntos registrados por una cuenta.

Modificar punto

Verbo	`PUT`
Endpoint	`/api/v1/beacons/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar punto

Verbo	`DELETE`
Endpoint	`/api/v1/beacons/:id`
Descripción	Si la peticion tiene los permisos necesarios el punto de prueba de paso se elimina.

Validar punto

Verbo	`POST`
Endpoint	`/api/v1/beacons/validate`
Descripción	El sistema valida un punto de prueba de paso para un usuario.

8.5.4. Puntos de recompensa

Crear punto de recompensa

Verbo	`POST`
Endpoint	`/api/v1/rewards/point`
Descripción	Se crea un nuevo punto de recompensa con los datos necesarios y su localizacion siempre que estos sean correctos.

Consultar punto de recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de puntos de recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/point`
Descripción	Se obtiene un listado con los puntos de recompensa creados por una cuenta.

Modificar punto de recompensa

Verbo	`PUT`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar punto de recompensa

Verbo	`DELETE`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Si la peticion tiene los permisos necesarios el punto de recompensa se elimina.

8.5.5. Recompensas

Crear recompensa

Verbo	`POST`
Endpoint	`/api/v1/rewards`
Descripción	Al crear una recompensa se especifican las pruebas de paso necesarias y los puntos de recompensa autorizados a otorgarlas.

Consultar recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de recompensas

Verbo	`GET`
Endpoint	`/api/v1/rewards`
Descripción	Obtiene un listado con las recompensas que ha recibido un usuario.

Modificar recompensa

Verbo	`PUT`
Endpoint	`/api/v1/rewards/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar recompensa

Verbo	`DELETE`
Endpoint	`/v1/rewards/:id`
Descripción	Si la peticion tiene los permisos necesarios la recompensa se marca como ”no entregable”.

Validar recompensa

Verbo	`POST`
Endpoint	`/v1/rewards/validate`
Descripción	Valida que todos los puntos esten validados y emite un pago al turista.

Reclamar recompensa

Verbo	`POST`
Endpoint	`/v1/rewards/claim/:name`
Descripción	Realiza el pago al establecimiento para recibir el obsequio final.