

DOCUMENTO DE GESTIÓN, ANÁLISIS, DISEÑO Y MEMORIA DEL PROYECTO

SISTEMAS Y TECNOLOGÍAS WEB

Hayk Kocharyan

757715@unizar.es

Luis García

739202@unizar.es

Germán Garcés

757024@unizar.es

Alberto Calvo

760739@unizar.es

22 de mayo de 2021

Resumen Ejecutivo

A lo largo de este documento se detallará diferentes aspectos sobre el sistema de cita previa Zitation desarrollado en este proyecto con el stack **MERN**. Se comentarán detalles sobre la organización del equipo, el diseño del sistema y detalles de implementación. Acerca de la implementación hablaremos de la arquitectura, los *framework* usados, **React** para *frontend* y **Express** para *backend*, junto a las diferentes librerías necesarias para la implementación de las funcionalidades claves del sistema, en el frontend los modals, formularios, calendario, panel de administrador, mapa *covid* y *captcha* de **Google**. En el caso de *backend* se corresponde al envío de emails mediante Gmail, la extracción de datos de **Transparencia Aragón**, la seguridad del sistema utilizando **Json Web Token** y **Geocoding**. Además se incluye el modelo de datos de **MongoDB**, la URL de nuestra **API REST**, una implementación de diferentes *widgets* usados en la Web, el despliegue del sistema en **Heroku**, y por último detalles sobre la validación y el *testing*. Para acabar el documento, contemplaremos los puntos débiles del sistema, posibles problemas a futuro y soluciones que se contemplan para estos y además se podrán leer conclusiones personales de los cuatro miembros del equipo.

Índice

Resumen Ejecutivo	2
1. Introducción	2
1.1. Resumen	2
1.2. Proyectos similares	2
2. Gestión y Organización	3
2.1. Diagrama de Gantt	3
2.2. Organización interna	3
2.3. Horas invertidas.	4
2.3.1. Hayk	5
2.3.2. German	6
2.3.3. Luis	7
2.3.4. Alberto	7
3. Diseño e Implementación del Sistema	8
3.1. <i>Frameworks</i> y Tecnologías	8
3.1.1. React	8
3.1.2. Express	8
3.1.3. MongoDB	8
3.2. Arquitectura alto nivel	8
3.2.1. Diagrama de despliegue	8
3.2.2. Diagrama de componentes	9
3.3. API Rest	12
3.4. Modelo de Datos	12
3.5. Detalles de Implementación Frontend	13
3.5.1. Formularios	13
3.5.2. Modals	13
3.5.3. Calendario	14
3.5.4. Panel de administrador	15
3.5.5. Captcha	17
3.5.6. Mapa	18
3.6. Detalles de Implementación Backend	19
3.6.1. Email	19
3.6.2. Extracción datos covid	20
3.6.3. Geocoding	21
3.6.4. Json Web Token	21
3.7. Detalles de Despliegue	21
3.7.1. Heroku	21
3.7.2. Despliegue Continuo	21
3.7.3. Poblar base de datos	22
3.8. Validación y Testing	22
3.9. Diseño de la interfaz	23
3.9.1. Modelo de Navegación	24
4. Conclusiones	44
4.1. Conclusiones del Proyecto	44
4.2. Problemas afrontados	44
4.3. Problemas a futuro	44
4.4. Conclusiones individuales	44
4.4.1. Hayk	44
4.4.2. Alberto	45
4.4.3. Luis	45
4.4.4. German	45

1. Introducción

1.1. Resumen

Zitation es una web que permite a empresarios de la ciudad de Zaragoza registrar sus negocios y ofrecer sus servicios a los ciudadanos zaragozanos. A través de **Zitation** pueden registrar sus servicios y ofrecer un servicio de cita previa para evitar colas y aglomeraciones en sus negocios. Además, como negocio pueden ver las diferentes reservas que tienen para cierta fecha y poder organizar su agenda. También se les ofrece poder gestionar las reservas, de esta forma pueden cancelar una reserva, borrar servicios o modificar sus horarios.

Las empresas, al registrarse, introducirán sus horarios de apertura y la duración media de sus servicios de esta manera se podrá mostrar a los clientes las franjas horarias en las que ofrecer citas.

Para completar el servicio de cita previa, existe otro tipo de usuarios que llamaremos clientes. Estos son potenciales clientes para las empresas, ciudadanos de Zaragoza. Los clientes pueden consultar las empresas registradas en nuestro sistema, consultar y ver cuando pueden realizar reservas y confirmar estas.

Los clientes recibirán un correo con información acerca de la reserva y además un **código QR** que al escanear les llevará a la web de **Zitation** para mostrarles los datos de la reserva y más información. Cuando cancelen su reserva también se les notificará por correo.

Los usuarios podrán dar un *feedback* a la empresa a través de opiniones, estas opiniones podrán ser votadas de forma positiva (*like*) aquellas opiniones con las que estén de acuerdo.

En **Zitation** también se ofrece un mapa con los últimos datos de la incidencia del **Covid-19** y los comercios registrados en Zaragoza. Estos datos se actualizan a diario y se obtienen de la base de datos del ayuntamiento de Zaragoza. Estos datos de incidencia se ofrecen por zonas de salud básica para ofrecer mas granularidad.

Por último, **Zitation** ofrece un panel de administrador para poder gestionar usuarios y empresas registradas, así como ver estadísticas acerca de las empresas, franjas horarias y categorías con más reservas.

1.2. Proyectos similares

Se pueden observar otros servicios de cita previa como **Booksby**, sistema de cita previa centrado en belleza y estética. Otro ejemplo que tenemos es **Bookitit** que ofrece un sistema de cita previa para que las empresas puedan incorporar en su web, página de **Facebook** o **Instagram**. Además ofrecen datos estadísticos, calendario, panel de control y notificaciones por correo.

2. Gestión y Organización

2.1. Diagrama de Gantt

En el siguiente diagrama de Gantt se muestra los pasos seguidos en el proyecto -de manera general- separados por equipos. Equipo de backend encargado de la API REST y MongoDB y equipo Frontend encargado de la aplicación web.

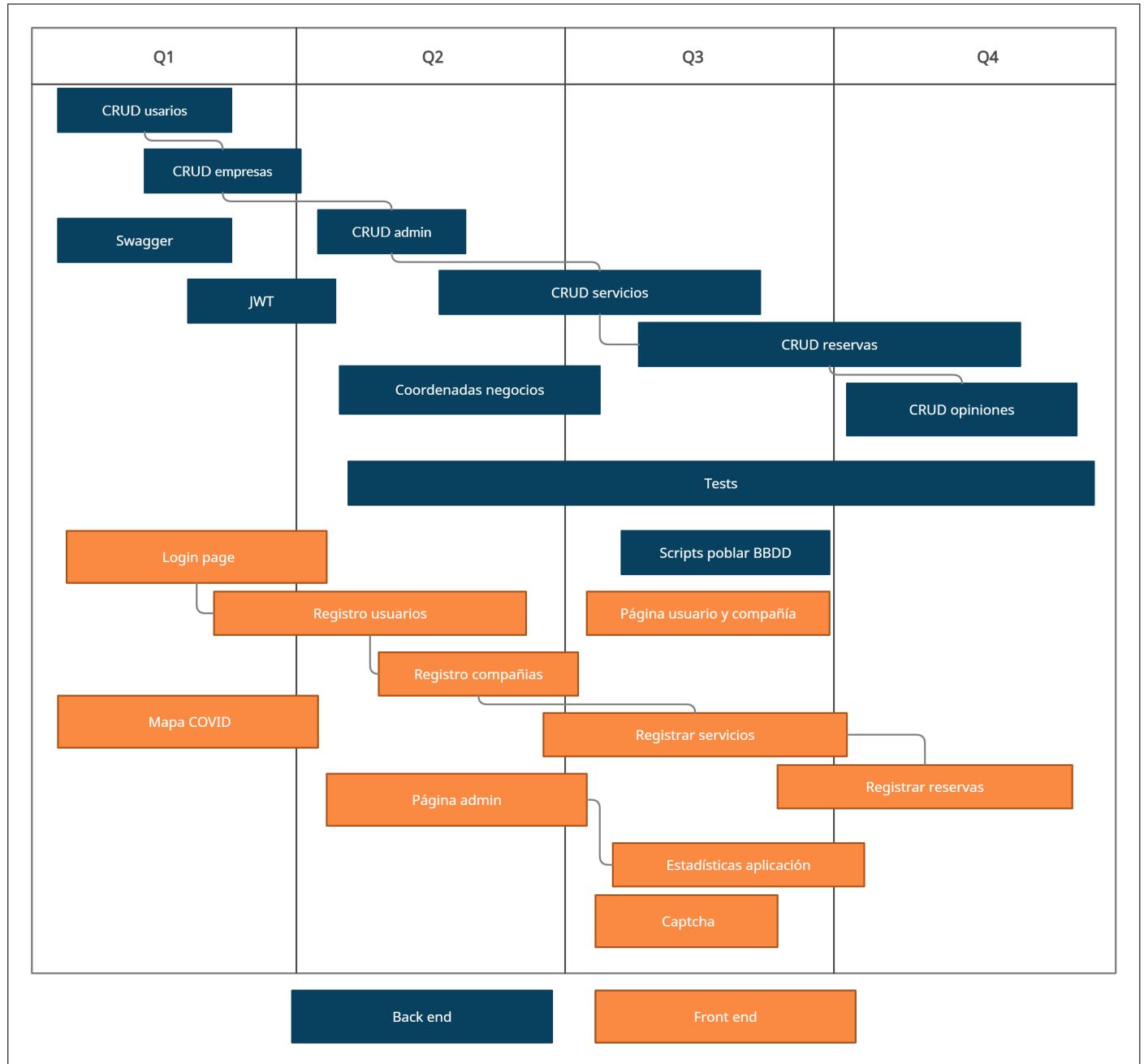


Figura 1: Diagrama de Gantt

2.2. Organización interna

A nivel interno el equipo se ha organizado a través de *sprints* semanales. Las primeras semanas se usaron para poner en común las diferentes ideas y elegir la que mas convenciese al equipo. Tras elegir la idea, se fijaron

las fuentes de datos, los requisitos del sistema y se realizó el reparto en dos equipos de dos miembros, un equipo para *frontend* y otro para *backend*.

Por otro lado se fijaron una franja de 2 horas dos días de la semana para poder realizar reuniones sin ninguna interferencia de otras asignaturas o compromisos.

Semanalmente el equipo se reunía para poner en común el trabajo avanzado durante el sprint, comentar los problemas generados y fijar las tareas para el siguiente sprint.

Una vez se tuvo claro las fuentes de datos, la idea, los requisitos y el reparto de equipos se empezó con los *sprints* de implementación. En cada sprint se fijaba unos requisitos a ser implementados tanto en *back* como en *front* y se trabajaba paralelamente.

2.3. Horas invertidas.

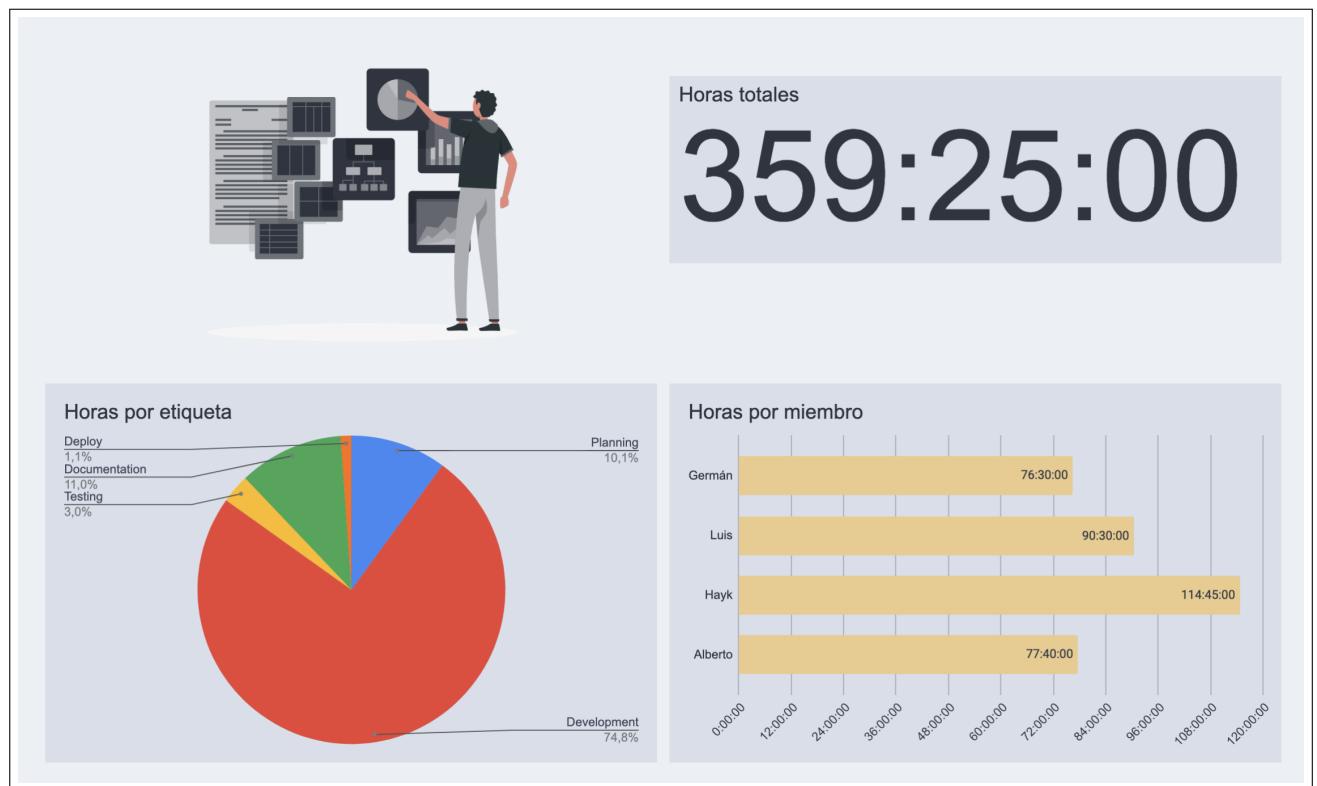


Figura 2: Vista general de los esfuerzos del equipo.

2.3.1. Hayk

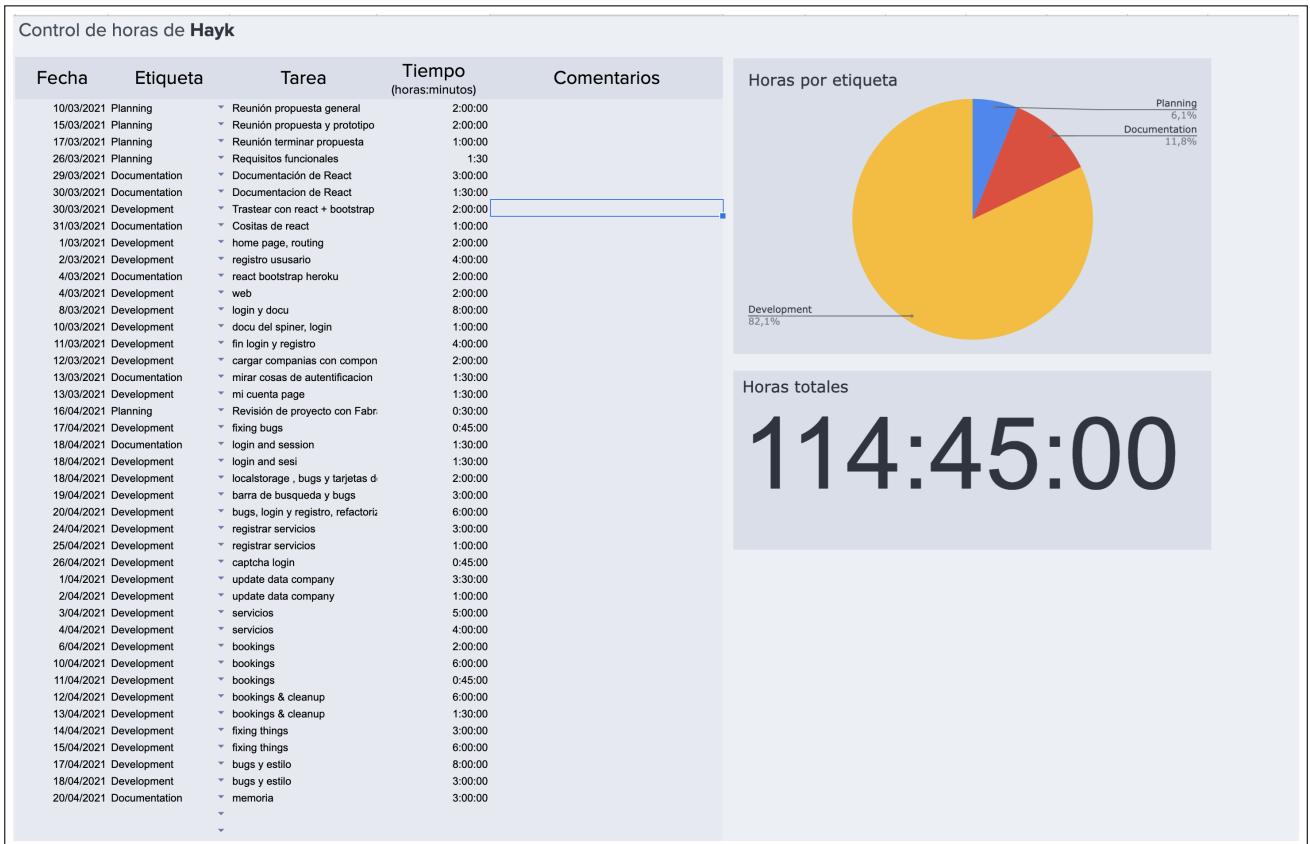


Figura 3: Vista de los esfuerzos de Hayk.

2.3.2. German

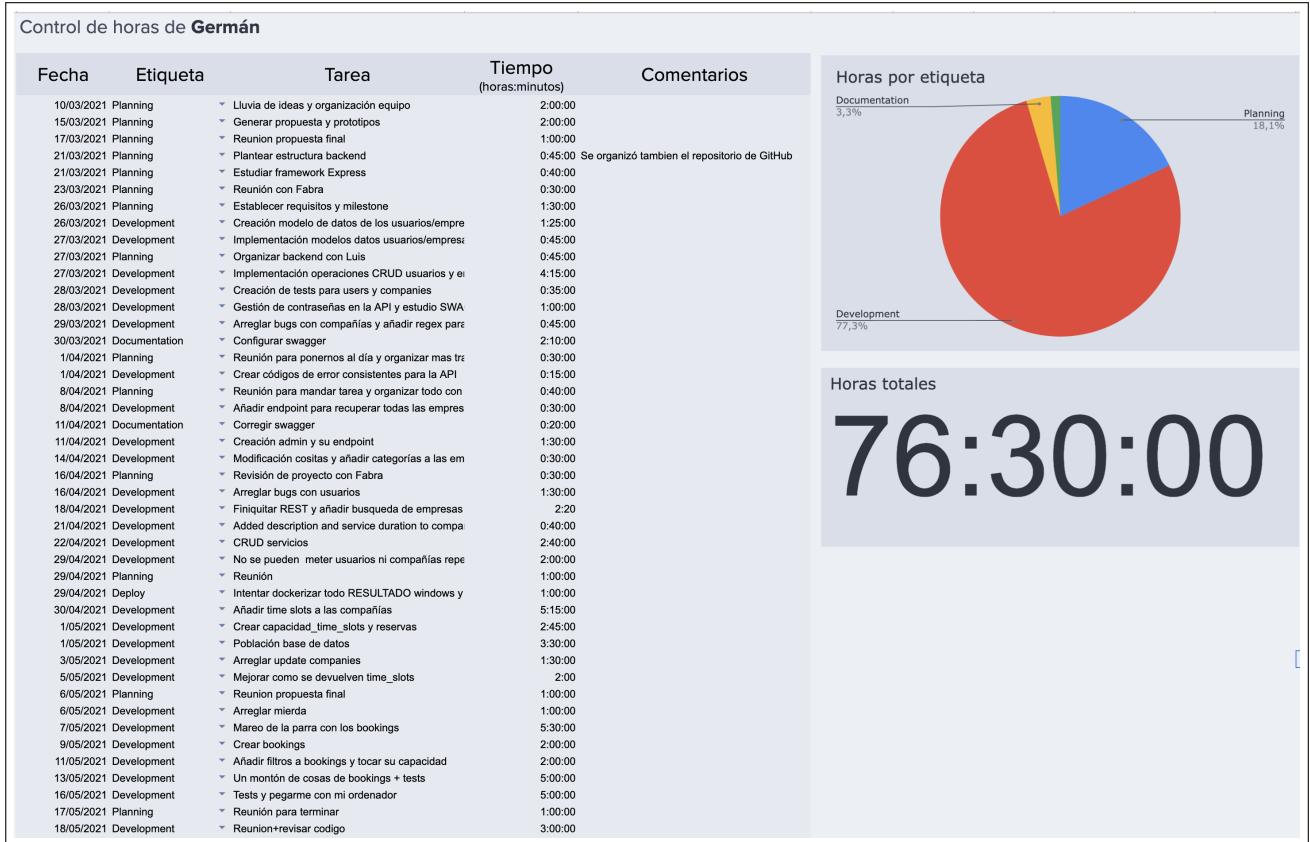


Figura 4: Vista de los esfuerzos de German.

2.3.3. Luis

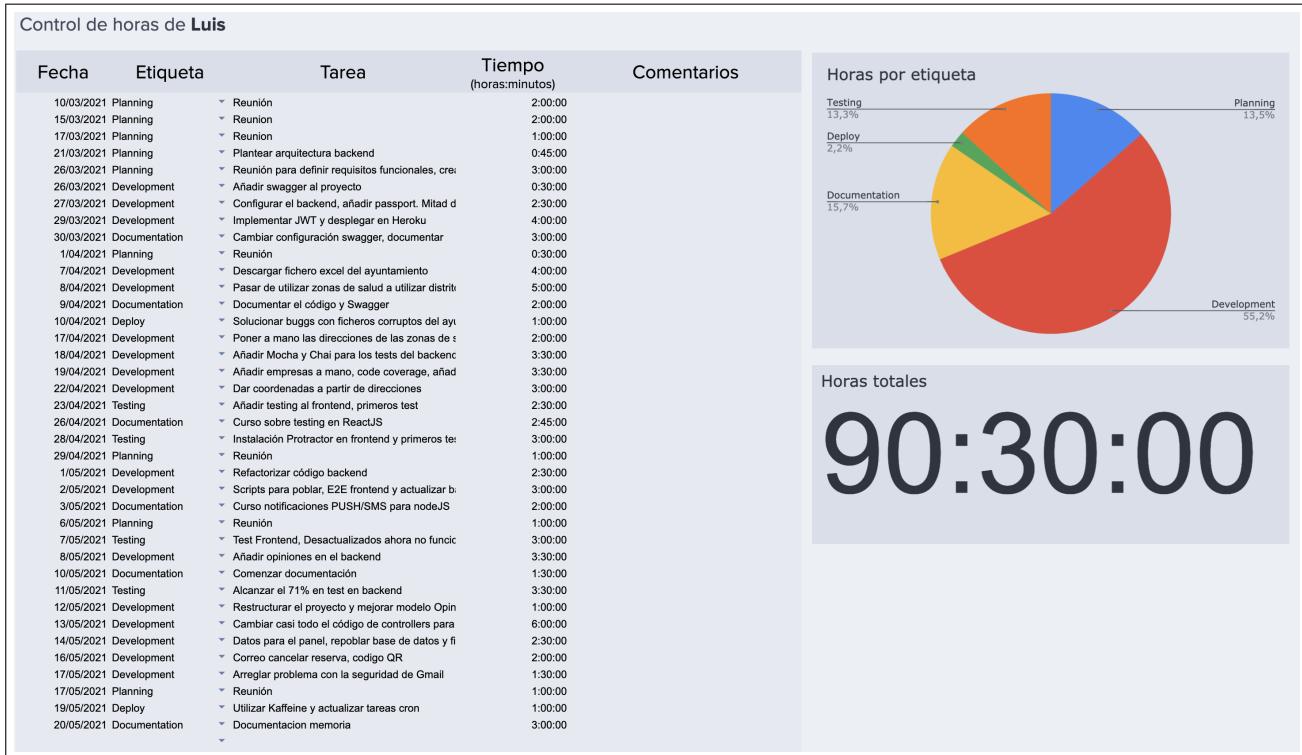


Figura 5: Vista de los esfuerzos de Luis.

2.3.4. Alberto

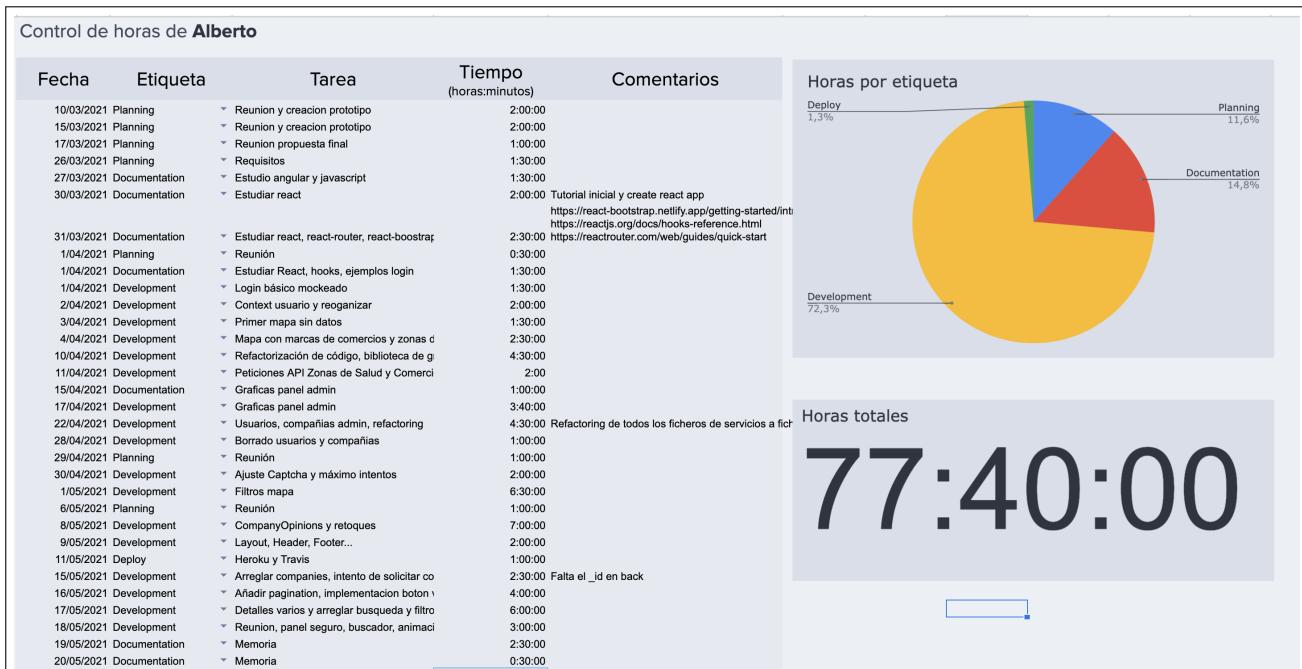


Figura 6: Vista de los esfuerzos de Alberto.

3. Diseño e Implementación del Sistema

3.1. Frameworks y Tecnologías

3.1.1. React

React es una biblioteca de JavaScript para construir interfaces de usuario desarrollada por Facebook. Al principio del proyecto se debatió si empezar con la herramienta que se proponía, Angular, o elegir otra distinta. Tras llevar a cabo algunos tutoriales de Angular, el equipo de frontend decidió que React podía ser una mejor opción debido a su sencillez y apoyo de una gran comunidad.

La curva de aprendizaje al principio fue un poco elevada pero menor que con Angular. En poco tiempo los miembros del equipo de frontend fueron capaces de entender la mecánica basada en componentes de este framework y conceptos avanzados como *hooks* y renderizado del DOM. Además, el contar con una gran comunidad detrás facilitó el desarrollo al poder reutilizar muchos componentes de terceros como *react-bootstrap*[9], *react-router*[13], *react-leaflet*[2] y otros.

3.1.2. Express

Express es un framework cuya finalidad es facilitar el desarrollo de aplicaciones web y API. La curva de aprendizaje fue relativamente rápida debida a la cantidad y la calidad de la documentación existente tanto en la página web oficial como en otros sitios como foros. El uso de este framework fue muy útil ya que nos ofrecía poder realizar enrutamiento de las peticiones de manera muy sencilla por lo que el desarrollo de la API no tuvo demasiada complejidad a la hora de recibir peticiones.

3.1.3. MongoDB

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. La curva de aprendizaje muy rápida debido a su facilidad de uso desde el framework *Mongoose*[5].

3.2. Arquitectura alto nivel

3.2.1. Diagrama de despliegue

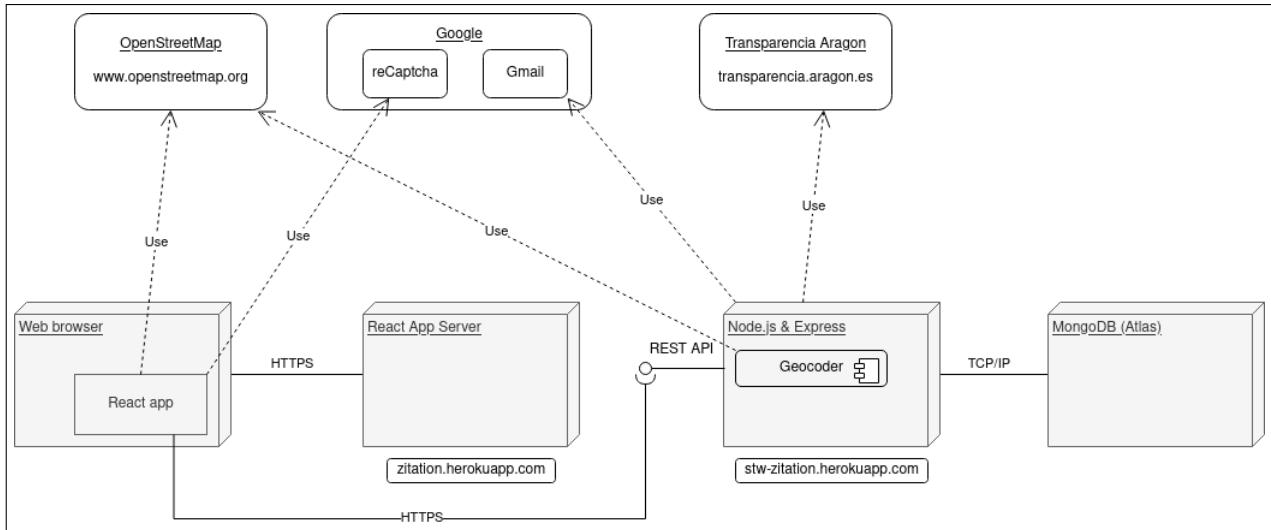


Figura 7: Diagrama de despliegue

3.2.2. Diagrama de componentes

Con la ayuda de una herramienta de generación automática de diagramas de componentes[1] podemos observar los paquetes y componentes de nuestro sistema tanto en frontend como backend.

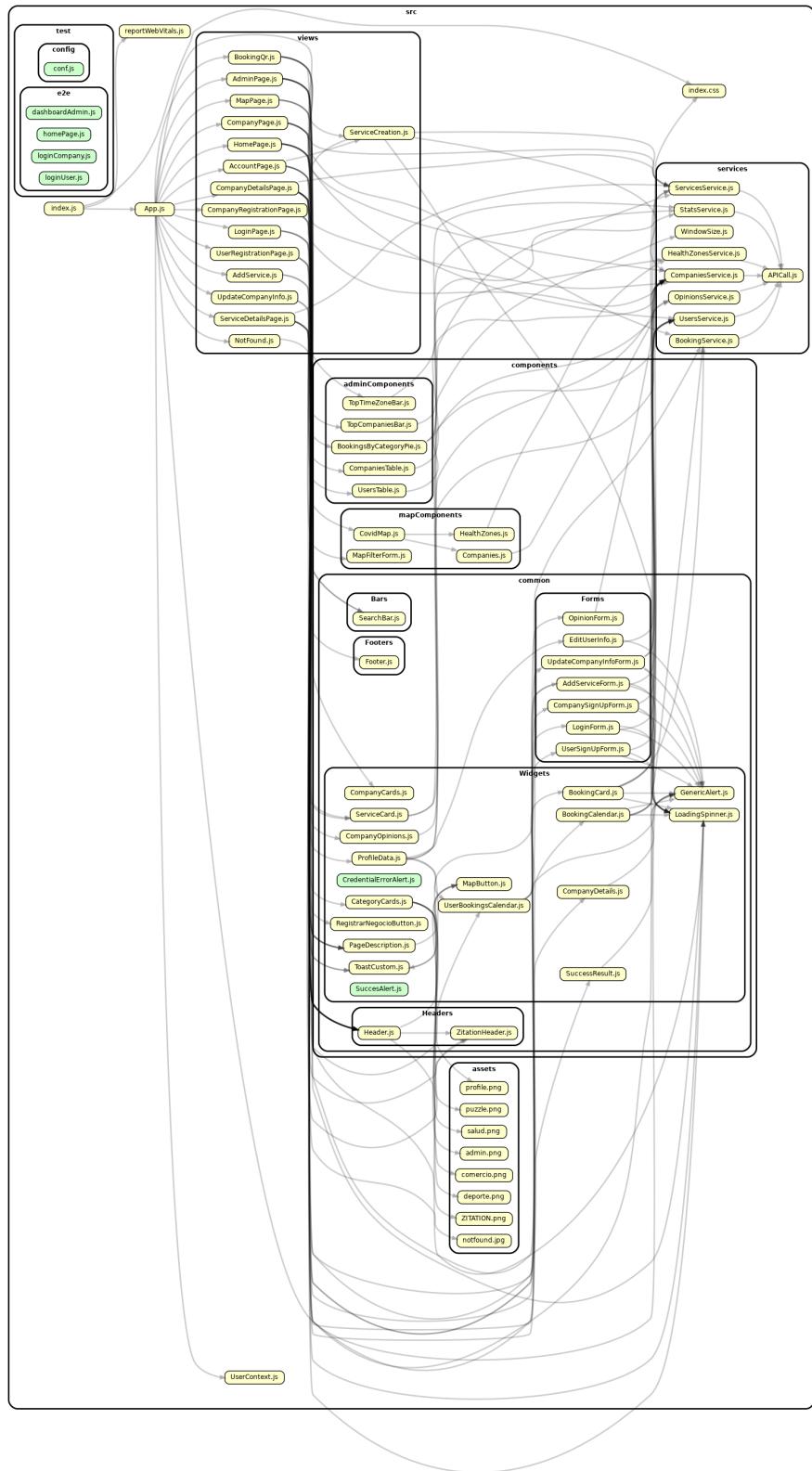


Figura 8: Diagrama de componentes frontend

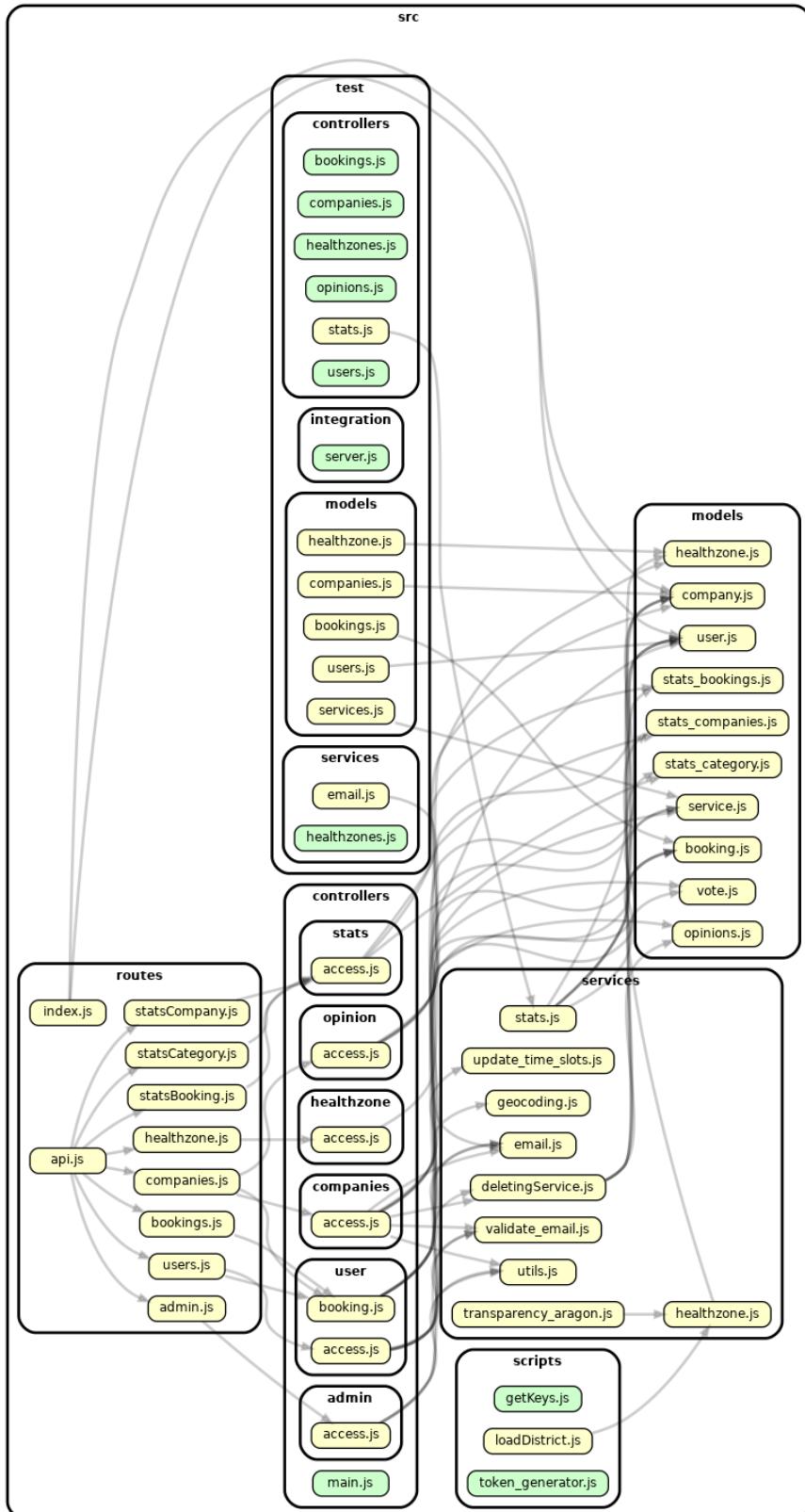


Figura 9: Diagrama de componentes backend

3.3. API Rest

El servidor *backend* de **Zitation** cuenta con una *API REST*, documentada con **Swagger** y accesible a través de la URL: <https://stw-zitation.herokuapp.com/api-docs/>

Esta API se ha dividido en varias partes dependiendo de si la operación es con usuarios, compañías, servicios o reservas. Para estos cuatro grupos se han creado los endpoints para las funciones CRUD (crear, leer, actualizar y borrar) y se han añadido algunos endpoints adicionales para poder sacar otros datos conjuntos como por ejemplo los datos necesarios para la creación de las estadísticas.

3.4. Modelo de Datos

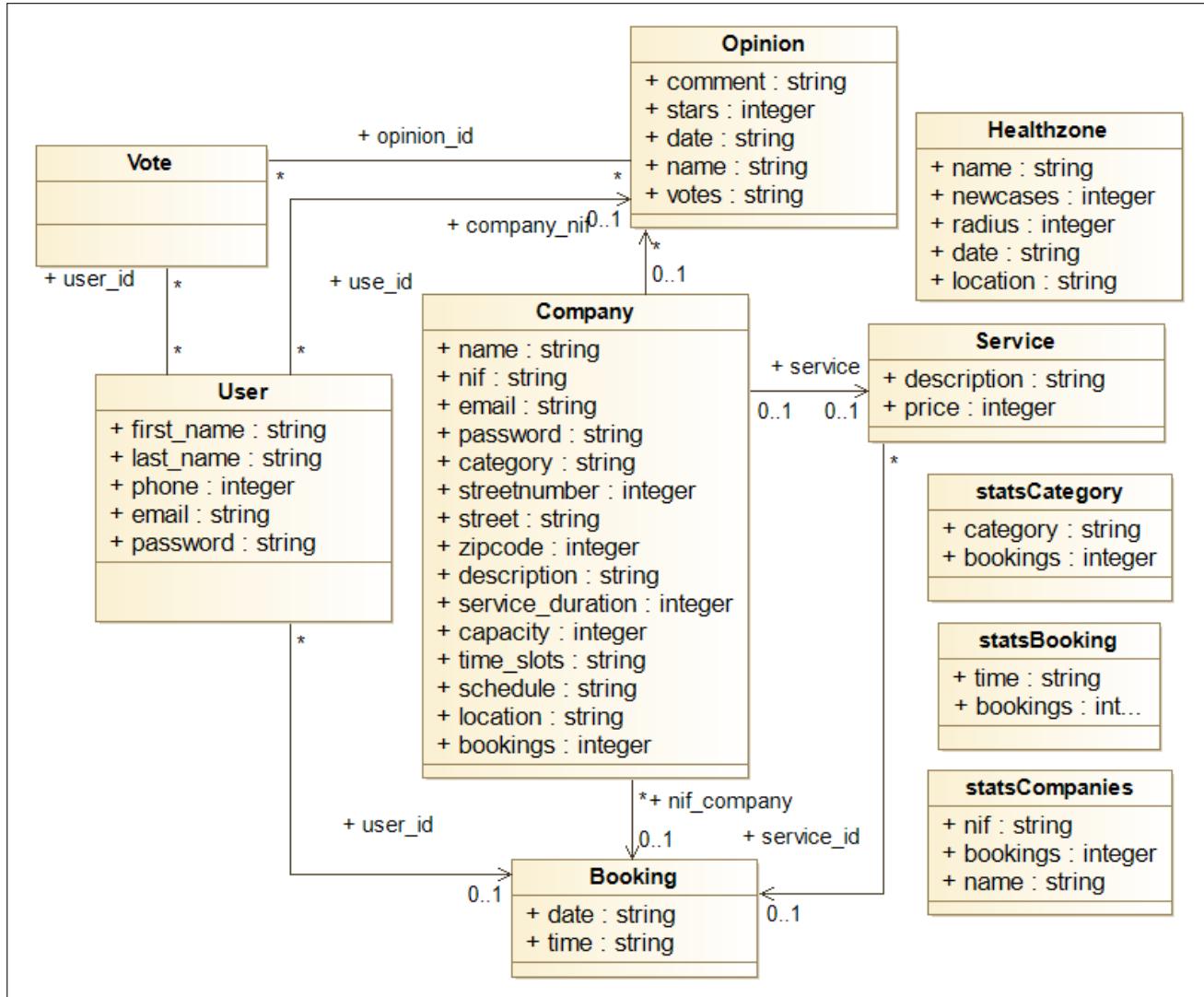


Figura 10: Diagrama de clases

Como base de datos se ha utilizado **MongoDB**. **Company** es la clase que representa a las empresas. **User** se corresponde al usuario que reserva horas en los servicios de las empresas. Cada empresa tiene asociados los servicios que ofrece que se encuentran representados en la clase **Service**. Los usuarios pueden escribir lo que opinan de las empresas, estas opiniones se representan en la clase **Opinion** que tiene asociado el usuario que la ha escrito y la empresa a la que está dedicada. Para evitar que los usuarios voten más de 1 sola vez un comentario se ha creado la clase **Vote** que registra quien a votado cada opinión para evitar que se repita.

La información relacionada con los casos de *coronavirus* en las zonas de salud de la ciudad de Zaragoza se guardan en la clase **Healthzone**. La reservas, que representan lo más importante de esta aplicación, se guardan en la clase **Booking** que registra en qué empresa se ha reservado, qué usuario lo ha hecho y en qué servicio. Finalmente para guardar las estadísticas se han utilizado las clases **statsCategory**, **statsBooking** y **statsCompanies**.

3.5. Detalles de Implementación Frontend

Para la implementación del frontend se ha hecho uso de un sustituto de Bootstrap, que es React Boostrap[9], un framework que reemplaza el clásico Bootstrap JavaScript. Cada componente de este framework está construido sobre React sin dependencias como JQuery. También se han usado *kits* de interfaz de usuario como es Antd[15] y Shards React.[11] para diseñar componentes más bonitos, usables y agradables a la vista sin perder mucho tiempo modificando las componentes con Boostrap y Html.

3.5.1. Formularios

Los formularios de la web se han implementado usando la librería de React Bootstrap[9] para el diseño de la interfaz y para la funcionalidad que hay por detrás se ha usado el **react-hook-form**[3] lo que permite realizar validaciones e implementar un formulario complejo y con toda funcionalidad sin complicarse, sin hacer mucho uso de HTML plano y sin crear miles de componentes.

Figura 11: Formulario de registro de usuario.

Figura 12: Formulario de registro de usuario con campos erróneos

3.5.2. Modals

Los *modals* (ver figura 13) son elementos que se usan en la web para dar retroalimentación a nuestro usuario en acciones como realizar o cancelar una reserva, errores desconocidos, cierres de sesión por token caducado al intentar realizar una operación que requiere de token entre mucho otros ejemplos. Para ello, se ha usado la componente Modal de la librería React Boostrap[9] o Shards [11].



Figura 13: Ejemplo de *Modals*.

3.5.3. Calendario

Uno de los elementos principales de la web es el calendario (ver figura 14). Esta componente se ha obtenido del Kit Antd[15], se ha personalizado para que quede acorde a la Web y se le ha añadido funcionalidad para poder gestionar las reservas de los diferentes usuarios.

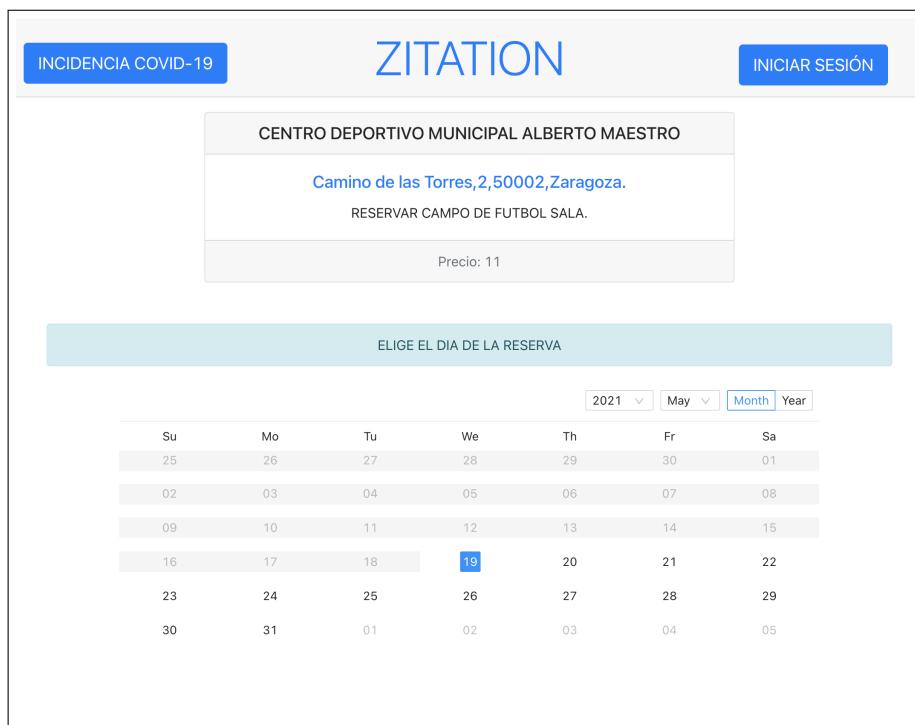


Figura 14: Ejemplo de Calendario.

3.5.4. Panel de administrador

Para poder administrar la web sin tener que adentrarse en el código o base de datos del sistema, se ha diseñado un panel de administración en el que se pueden realizar las siguientes tareas:

- Observar gráficas con estadísticas: categorías de comercios, franjas horarias y comercios con más reservas
- Listar cada uno de los usuarios con información y eliminarlo del sistema
- Listar cada uno de las compañías con información y eliminarla del sistema

The screenshot shows the 'Dashboard' section of the ZITATION administrator panel. At the top, there are three tabs: 'Estadísticas' (selected), 'Usuarios', and 'Compañías'. Below the tabs, it says 'Total: 13'. A table lists five users with columns for 'Nombre' (Name), 'Apellidos' (Last Name), and 'Opciones' (Options). Each row has a 'Borrar' (Delete) button in the 'Opciones' column. At the bottom left is a dropdown menu set to '5'. At the bottom right are page navigation buttons: '1' (selected), '2', '3', and '>'.

Nombre	Apellidos	Opciones
Admin	Mighty	<button>Borrar</button>
Borja	Pavon	<button>Borrar</button>
German	Garcés	<button>Borrar</button>
Saúl	Solanilla Hernández	<button>Borrar</button>
Eneko	Marcos Martínez	<button>Borrar</button>

Figura 15: Gestión de usuarios desde el panel de administrador

Las gráficas se han implementado con la biblioteca *chart.js* a través de un wrapper para React, *react-chartjs-2*. En esta ocasión se han utilizado gráficos de barras verticales y horizontales, y gráfica circular. Estas permiten obtener métricas de nuestro sistema y con vista a futuro se podrían proporcionar estadísticas a las empresas para comprobar como va su negocio.

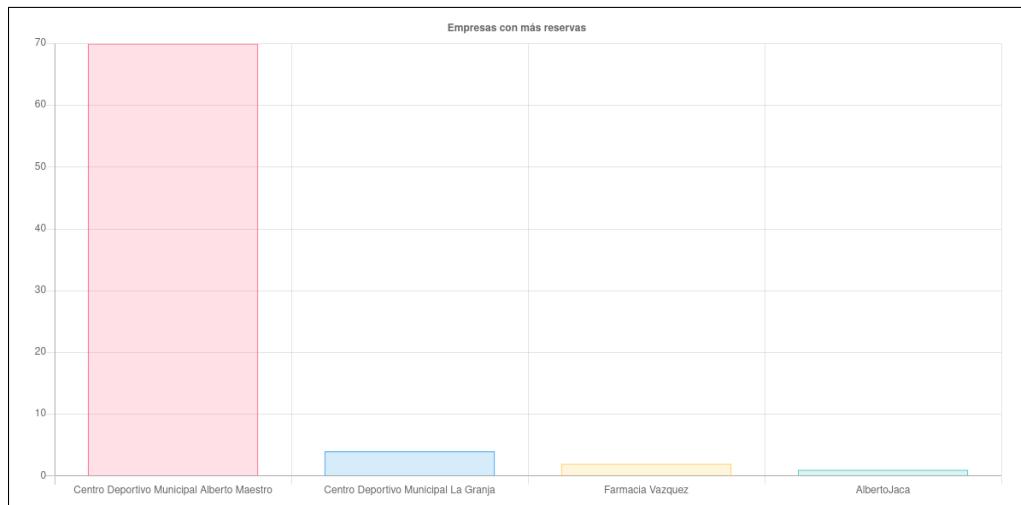


Figura 16: Gráfica de barras vertical de empresas con más reservas

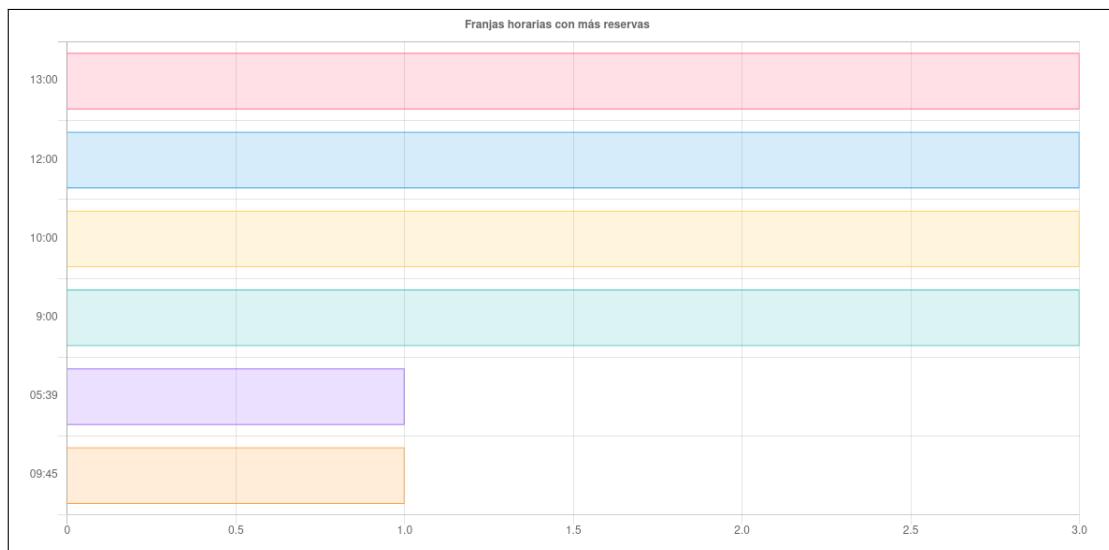


Figura 17: Gráfica de barras horizontal de franjas horarias con más reservas

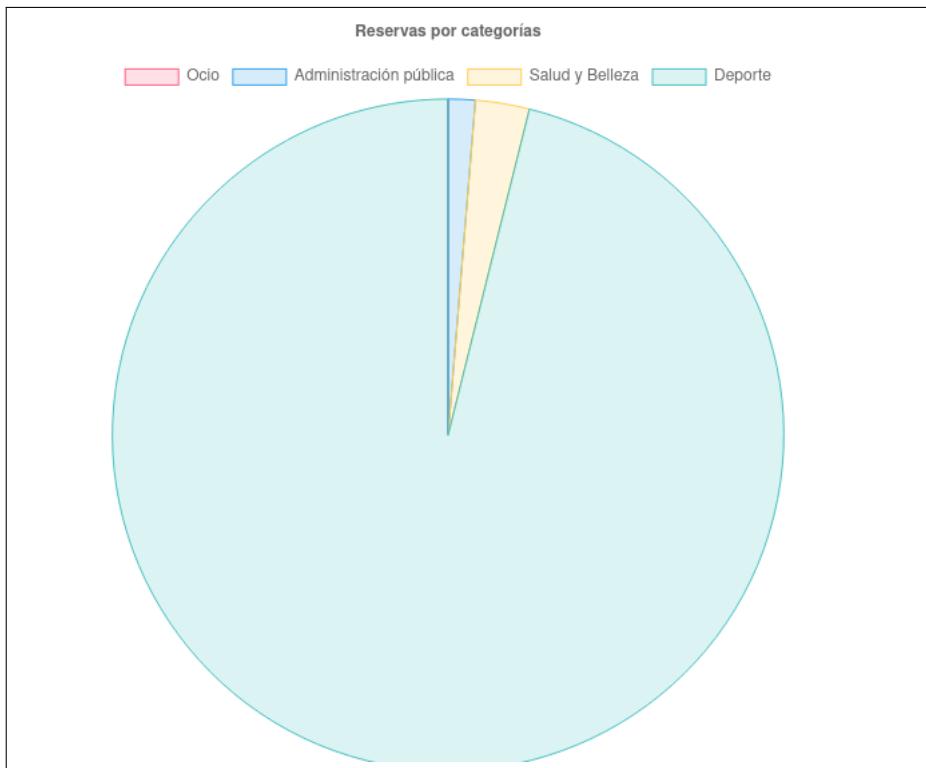


Figura 18: Gráfica circular de categorías con más reservas

3.5.5. Captcha

Para aumentar la seguridad en el registro de la web se ha añadido un captcha que bloquee el registro de bots. Para ello se ha utilizado uno de los más recomendados, reCaptchaV2. Este es desarrollado por Google y es uno de los más utilizados por su fiabilidad y facilidad de uso. Para poder utilizarlo es necesario registrarse con una cuenta de Google como desarrollador y conseguir una clave de la API. Esto permite obtener estadísticas de uso desde el panel de administrador en la consola de Google.

Número de móvil

Dirección de correo electrónico

Nunca compartiremos tus datos con terceros.

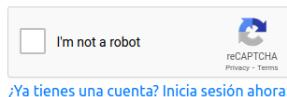
Nombre

Apellidos

Contraseña

 Mostrar

Confirmar Contraseña



[¿Ya tienes una cuenta? Inicia sesión ahora!](#)

[Sign Up](#)

Figura 19: Formulario de registro con reCaptchaV2

3.5.6. Mapa

El módulo del mapa se ha desarrollado con la ayuda de *react-leaflet* un wrapper de la biblioteca original de Javascript *leaflet*. Esta permite utilizar el mapa de OpenStreetMap y añadir componentes gráficos en él. En el caso de los comercios se han señalado mediante marcadores, mientras que las zonas de salud ha sido con zonas coloridas dependiendo del número de casos positivos de COVID-19.

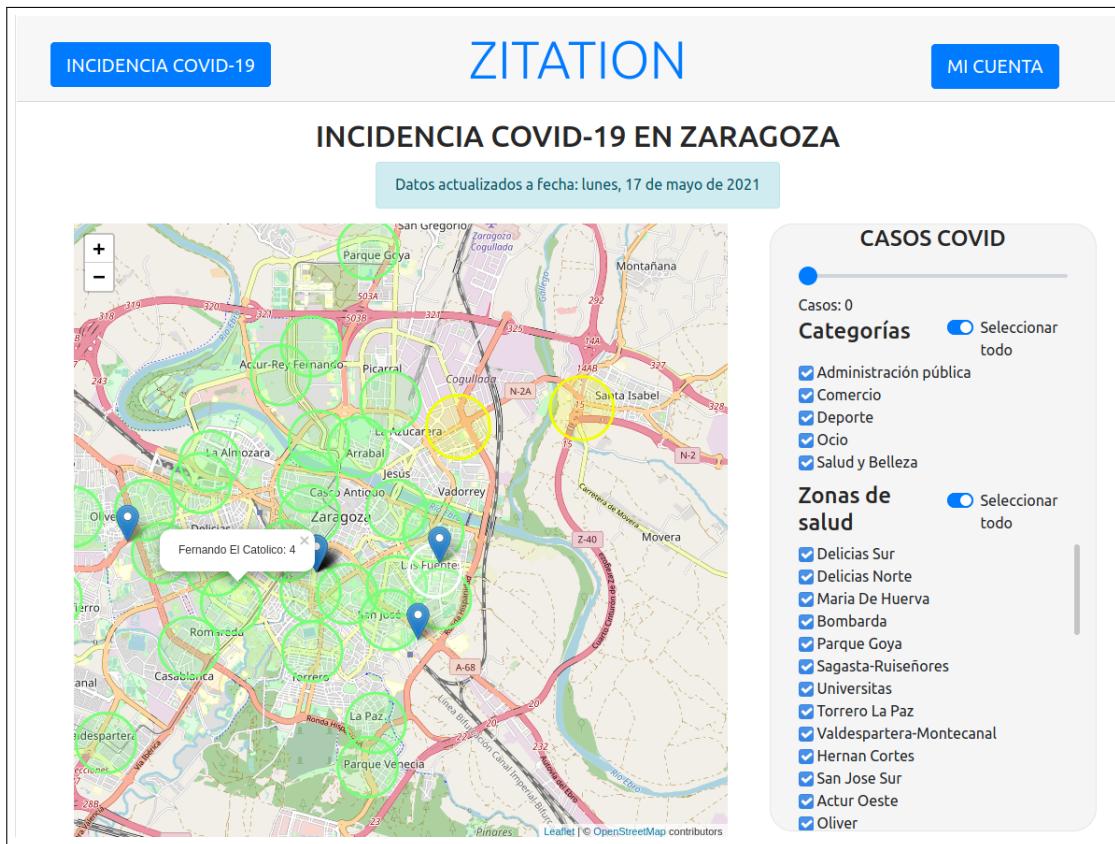


Figura 20: Mapa casos COVID-19 y comercios

3.6. Detalles de Implementación Backend

3.6.1. Email

Cuando un usuario o una empresa ingresan en la plataforma por primera vez reciben un correo, así mismo reciben un correo cuando los usuarios reservan hora en algún servicio. La implementación de esta funcionalidad se ha llevado a cabo gracias al uso de la biblioteca *node-mailer* [10]. Esta biblioteca permite enviar correos desde utilizando utilizando varios proveedores, en este caso **Gmail**. Para poder enviar correos sin riesgo y evitar que **Google** bloqueará la aplicación se decidió obtener credenciales e identificar a la aplicación de forma segura utilizando **Auth0**. Es necesario obtener para ello un *clientId* y *clientSecret* proporcionados por **Google** y dos tokens proporcionados por **Auth0** *refreshToken* y *accessToken*. Todo esto puede obtenerse siguiendo el siguiente tutorial: <https://developers.google.com/gmail/api/quickstart/nodejs>



Figura 21: Email de registro

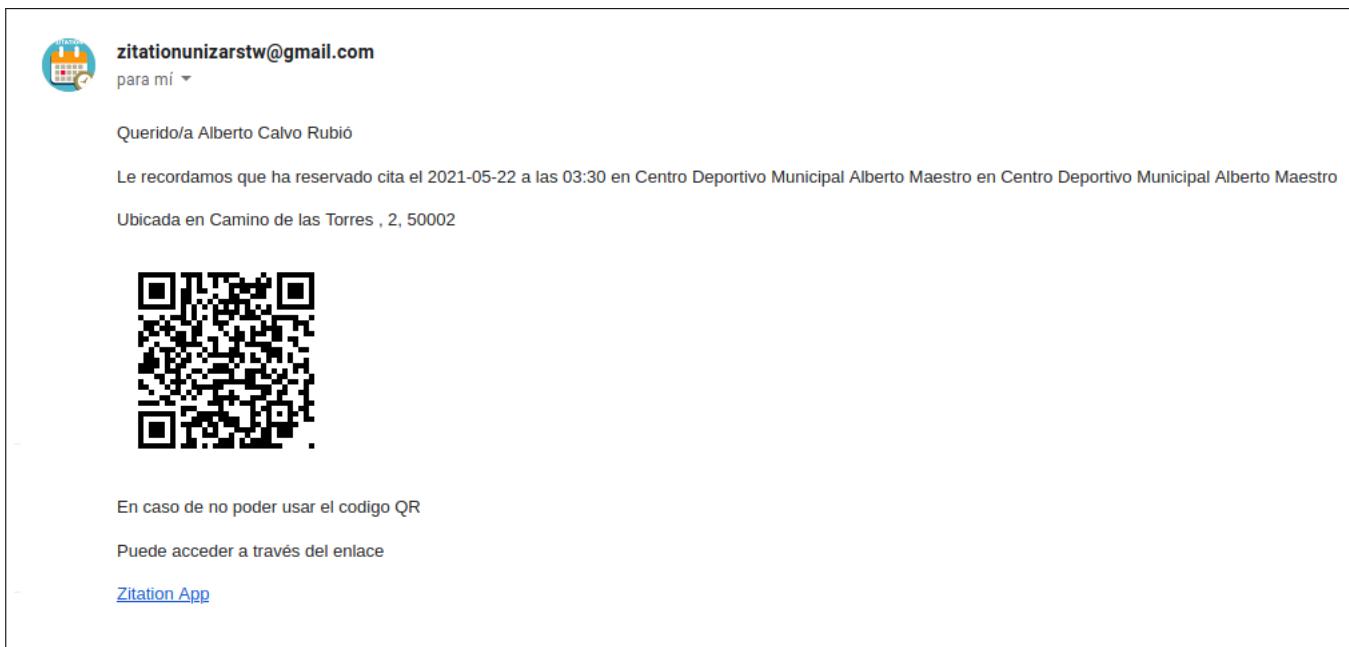


Figura 22: Email de reserva de cita

3.6.2. Extracción datos covid

Los datos de los casos de la ciudad de Zaragoza se obtienen a través de un enlace proporcionado por **Transparencia Aragón** [14]. Los datos se encuentran en un excel descargable a través del siguiente enlace: https://transparencia.aragon.es/sites/default/files/documents/20210517_casos_confirmados_zbs.xlsx. Mediante una petición GET se descarga el archivo y utilizando la biblioteca **exceljs** [12] se analiza el fichero **Excel**. Gracias a esta biblioteca se pueden tratar cada una de las hojas del excel como una matriz. Previamente se han guardado a mano las zonas de salud básica que comprenden exclusivamente a la ciudad de Zaragoza, se lee el fichero y se guardan los datos de aquellas zonas que se corresponden con las almacenadas en la base de datos. Este proceso se realiza una vez al día, ya que es la periodicidad con la que en teoría actualizan la fuente de datos, con una tarea programada. Esta tarea programa se hace utilizando la biblioteca **node-cron**[4], los datos se actualizan cada 5 horas.

3.6.3. Geocoding

Cuando una empresa se registre ha de proporcionar su dirección al completo, esto incluye su calle, su número y su código postal. Dadas estas variables utilizando la biblioteca **node-geocoder** [6] se puede obtener las coordenadas para posicionar los negocios en el mapa. **Node-geocoder** permite seleccionar proveedor, en este caso utiliza la API de **OpenStreetMaps** para dar las coordenadas.

3.6.4. Json Web Token

Con el objetivo de añadir seguridad a la aplicación se ha añadido el uso de **JWT**. Para validar las peticiones es necesario añadir un *middleware* donde se compruebe la veracidad del token JWT. Este *middleware* puede implementarse utilizando la biblioteca **passport** [8], **passport** nos permite definir estrategias de validación en el middleware, en este caso se aplica la estrategia de JWT sin sesión. Los tokens se generan utilizando la biblioteca **jsonwebtoken**. Estos tokens albergan el id que representa al usuario o a la empresa. Los tokens se cifran con una con la clave privada y se descifran el middleware con la clave pública. Las claves pública y privada se generan al arrancar la aplicación, no se guardan en el repositorio de Github, y cada que se reinicia la aplicación estas cambian. Los JWT tienen una duración de un día y cada vez que se inicia sesión cambian. Para generar las claves se ha utilizado la biblioteca **crypto** [7]

3.7. Detalles de Despliegue

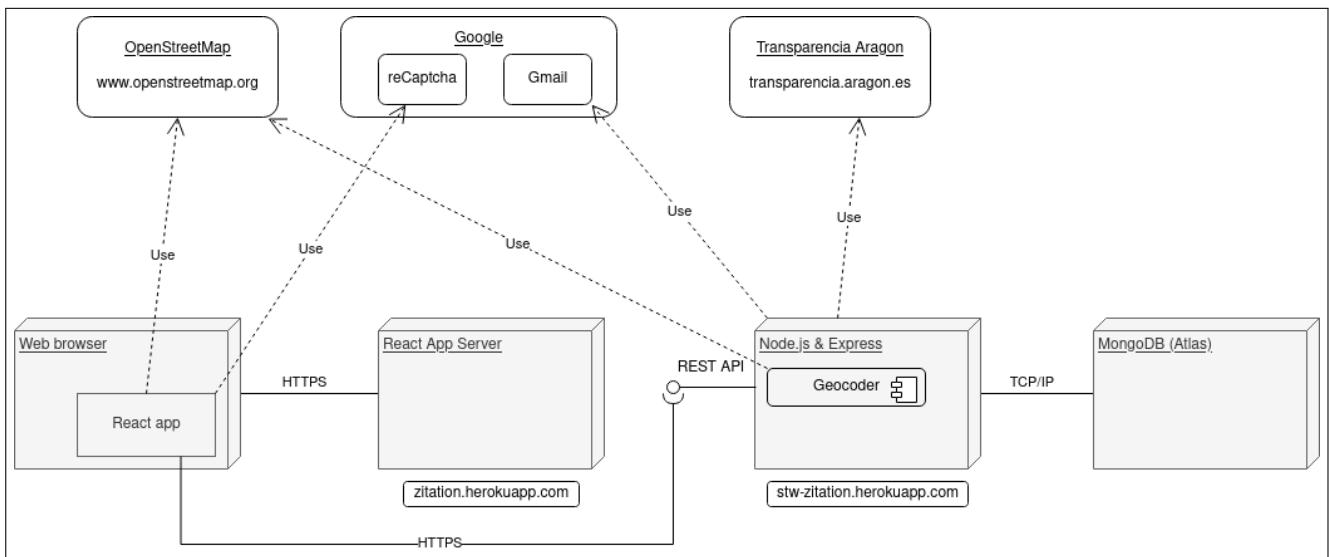


Figura 23: Diagrama de despliegue

3.7.1. Heroku

Tanto el servidor *backend* como el *frontend* web están desplegados en cloud público de **Heroku** con el plan gratuito. Sus direcciones son *backend*:`https://stw-zitation.herokuapp.com` y *frontend*:`https://zitation.herokuapp.com`. Debido al uso de Heroku de forma gratuita las aplicaciones se duermen a los 30 minutos de funcionamiento, por lo tanto para evitar esto se ha empleado **Heroku Kaffeine** en el *backend*. De esta manera el servidor se mantiene activo desde las 6:00 hasta las 0:00. Es necesario que el servidor *backend* se mantenga encendido debido a tareas programadas que ejecuta, como es la extracción de los casos de covid en la ciudad de Zaragoza.

3.7.2. Despliegue Continuo

Tanto el código de *backend* como el de *frontend* se encuentran alojados en repositorios de **GitHub**. **Heroku** tiene asociados estos repositorios de **GitHub** con el despliegue automático, de tal forma que cada vez que se

realice un commit la aplicación se volverá a construir y desplegar. Con el objetivo de evitar el despliegue de código se ha empleado integración continua. Se utiliza **Travis-CI** para la validación de cada *commit* que se realiza en el repositorio. De esta forma es posible ver si el código presenta algún error o si no ha pasado los tests en el caso de *backend*, de lo que se hablará a continuación, y evitar colapsar la aplicación.

Los enlaces a los repositorios de cada apartado son los siguientes: servidor backend https://github.com/luisgg98/STW-Sysadmins_backend y frontend web <https://github.com/AlbertoCalvoRubio/STW-frontend>

3.7.3. Poblar base de datos

En el repositorio de **GitHub** donde se encuentra el código hay un *script* escrito en **Python** para poblar la base de datos automáticamente. Este es un escrito de **Python 3**, hay que tener **Python 3.8** instalado para su correcto funcionamiento. De esta forma para poblar la base de datos hay que ejecutar el comando:

```
$ python3 population_script.py
```

3.8. Validación y Testing

Para la validación en el *backend* se ha utilizado el *framework* **Mocha** para realizar pruebas en **NodeJS** junto a la biblioteca **Chai**. El proyecto cuenta con una batería de pruebas que comprueban el funcionamiento de la API y la conexión con la base de datos. El repositorio de **GitHub** donde se encuentra el código del servidor se encuentra integrado con **Travis-CI** de esta forma cada vez que se hace push a este repositorio se ejecutan los tests. El servidor se encuentra desplegado en **Heroku** con despliegue automático cada vez que se actualiza el repositorio, sin embargo si los tests fallan no se producirá este despliegue evitando así desplegar código malfuncionando. Para obtener el code coverage del proyecto utilizamos el módulo **Istanbuljs/NYC**, una vez se han ejecutado los tests se muestra un informe de *code coverage* que podemos visualizar a través de **Travis**.

99 passing (5s)					
File	% Stmt	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	75.47	69.09	62.35	75.4	
...lers/companies	69.44	63.75	61.54	69.44	
access.js	69.44	63.75	61.54	69.44	...79,387,418,443
...ers/healthzone	66.67	50	66.67	66.67	
access.js	66.67	50	66.67	66.67	14-18
...ollers/opinion	72.45	67.74	62.07	72.16	
access.js	72.45	67.74	62.07	72.16	...74-182,235-239
controllers/stats	71.43	100	66.67	71.43	
access.js	71.43	100	66.67	71.43	14-15,29-30,44-45
controllers/user	71.51	67.97	59.34	71.22	
access.js	75.64	76.67	65.38	75.64	...43,159,168,174
booking.js	70.33	65.31	56.92	69.88	...31-538,556-557

Figura 24: Informe NYC tras completar los tests

Con el objetivo de evaluar la accesibilidad de la aplicación y se ha utilizado herramientas de evaluación automática. La primera en ser utilizada ha sido **GTMetrix**. **GTMetrix** es una herramienta que permite calcular el tiempo de carga de la página web, el tamaño total de la página y el número de peticiones que esta realiza.

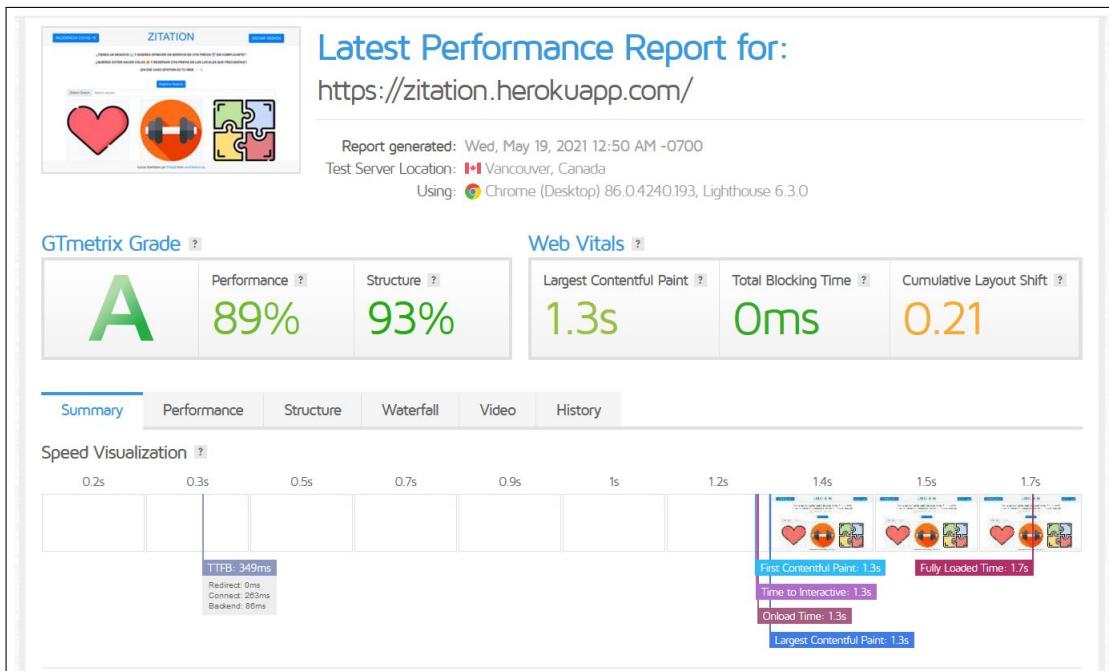


Figura 25: Resultado análisis GTMetrix

El resultado es una valoración positiva que indica que la aplicación es ligera y rápida de cargar. Puede accederse a esta evaluación a través del enlace : <https://gtmetrix.com/reports/zitation.herokuapp.com/5Aw1SgKe/> Otra herramienta empleada para validar la accesibilidad ha sido **TAW**, esta es una herramienta automática para analizar la accesibilidad de sitios web.

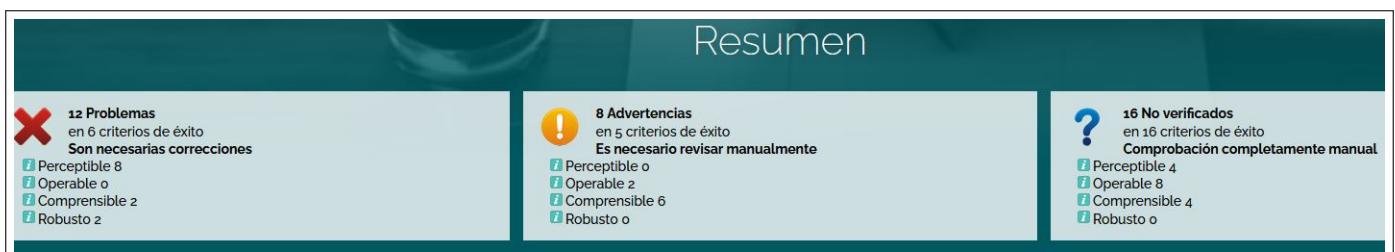


Figura 26: Resumen TAW análisis

La herramienta detecta 12 problemas de los cuales 8 de hechos son perceptibles y serían fáciles de solucionar. 8 advertencias y 16 cuestiones no verificadas. Al final de este documento en el anexo se encuentra el informe completo producido por **TAW**.

Para la validación del *frontend* se ha **Protactor** para hacer pruebas *E2E*. Las pruebas se han llevado a cabo con el driver de **Google Chrome** y se testean todos *endpoints* a los que puede acceder un usuario registrado o una empresa registrada. De los usuarios se prueban que sean capaces de iniciar sesión, editar su perfil, buscar el servicio de una empresa, valorar una empresa, votar una opinión y hacer una reserva de un servicio. De las empresas se prueban que sean capaces de iniciar sesión, editar sus datos y registrar un servicio. Sin embargo tras actualizar en la interfaz de la aplicación web estos tests han quedado desactualizados y por falta de tiempo no se han podido modificar para que se adapten a la versión final de la aplicación y funcionen correctamente.

3.9. Diseño de la interfaz

3.9.1. Modelo de Navegación

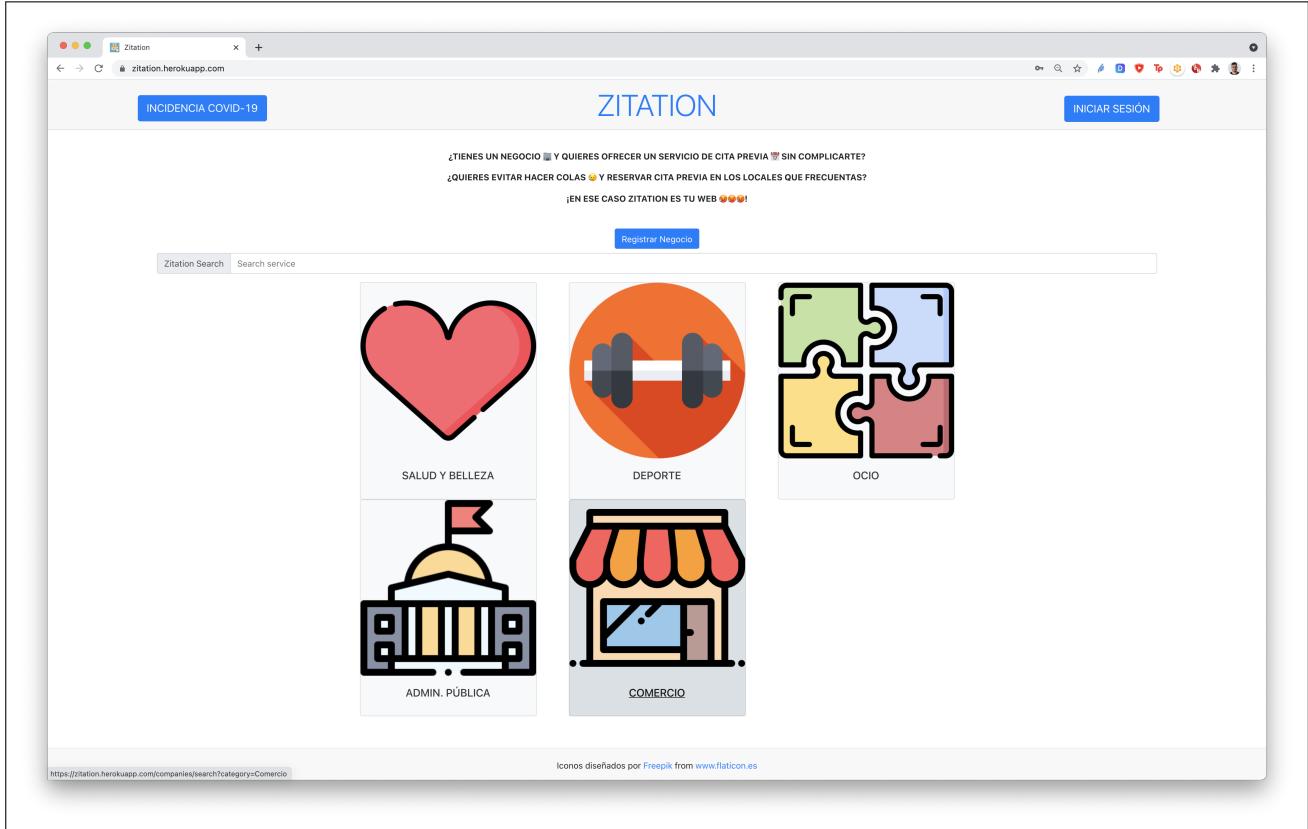


Figura 27: Pantalla de inicio

Vista de la pantalla de inicio. Al seleccionar alguno de los íconos se accede a un listado de empresas registradas bajo esa categoría. Se dispone de un botón de inicio de sesión en la esquina superior derecha, y en la izquierda el botón de acceso al mapa de incidencia del Covid-19. También se dispone de una descripción y un botón de registro para las empresas.

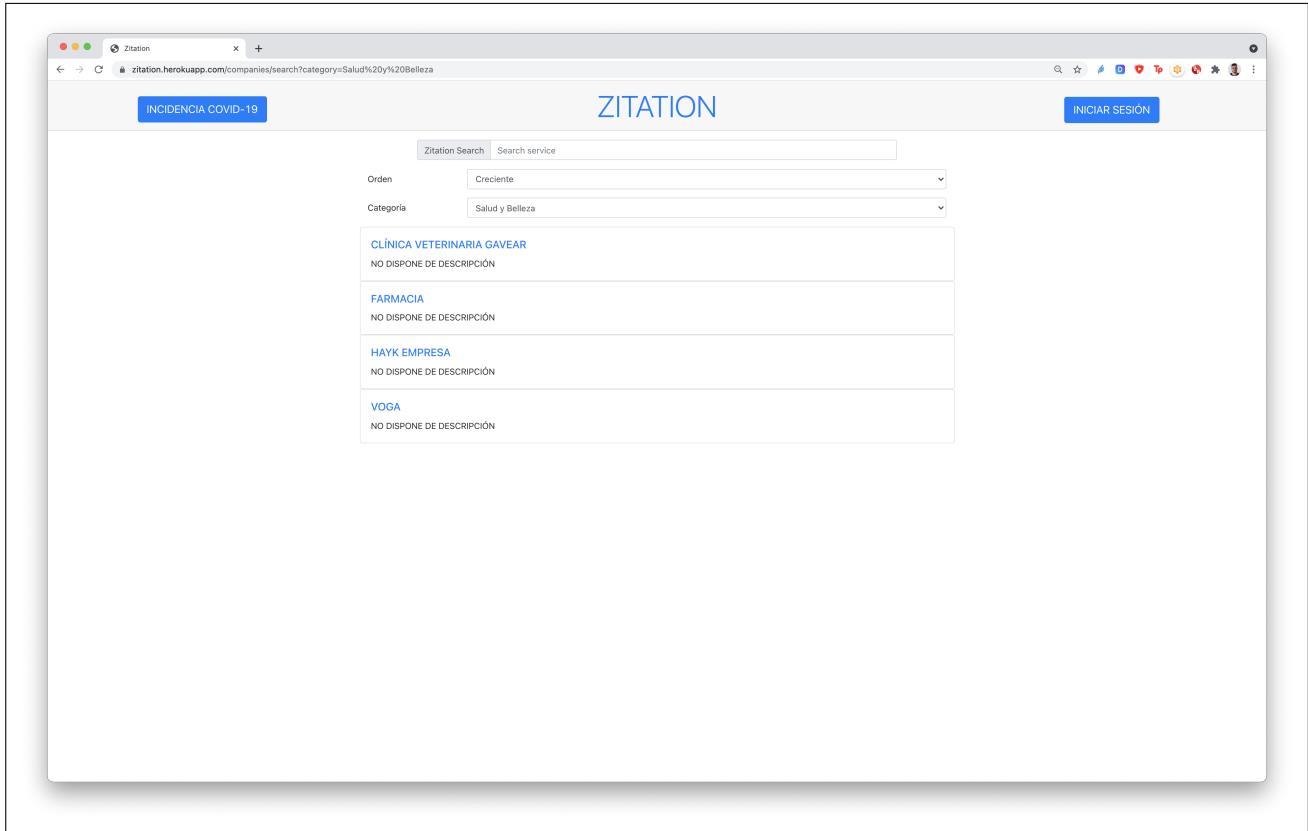


Figura 28: Pantalla de listado de empresas con filtros

Pantalla de listado de empresas al realizar una búsqueda o seleccionar una categoría.

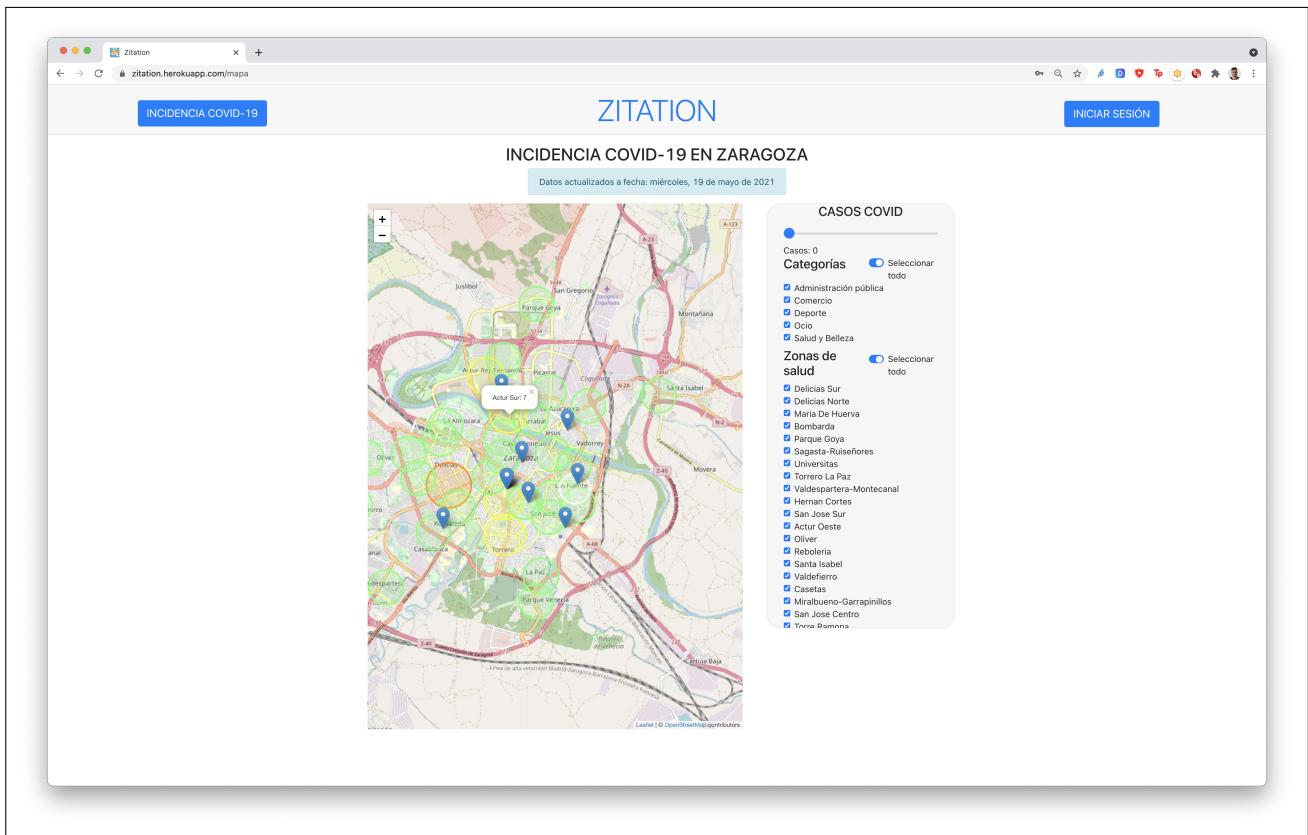


Figura 29: Pantalla de mapa de incidencia Covid-19

Vista del mapa de la incidencia del Covid-19 con diferentes filtros y una nota de la última fecha de actualización de datos.

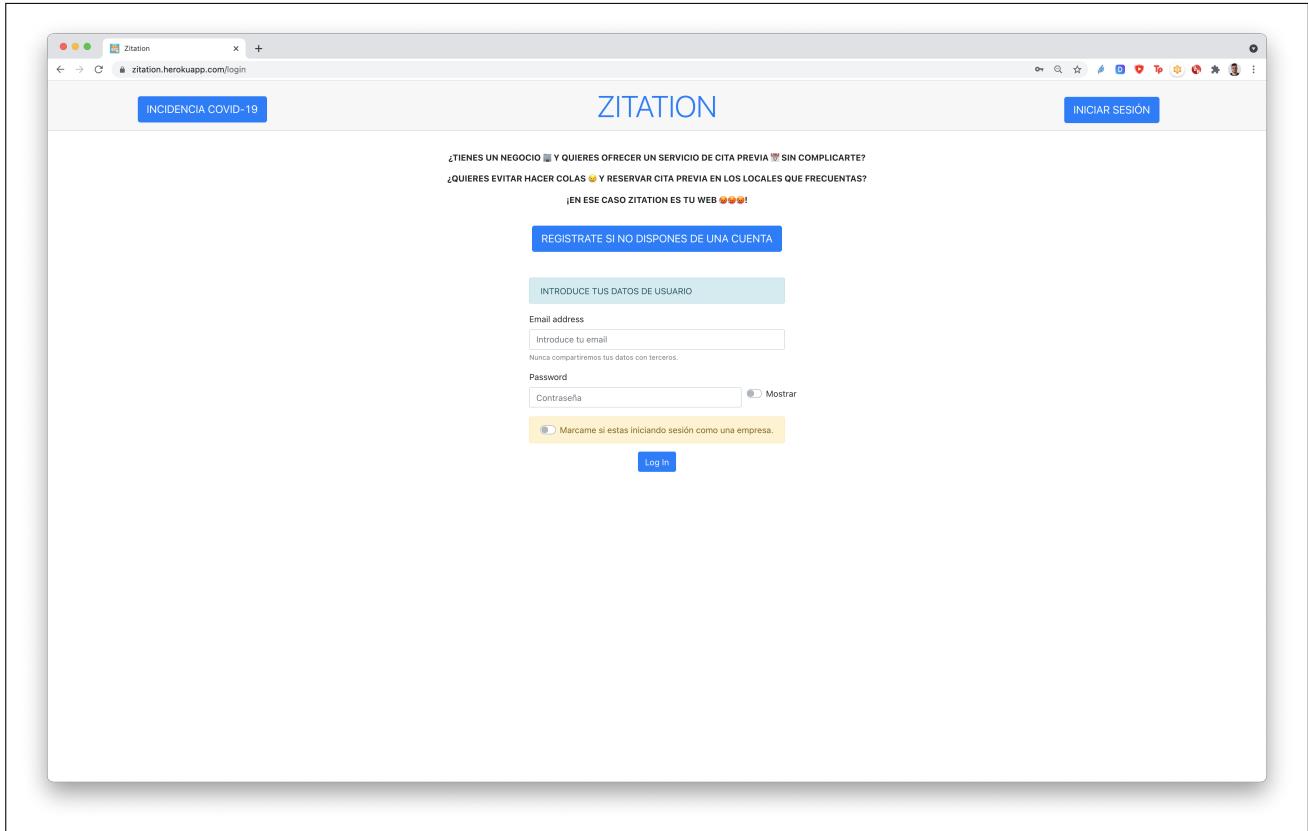


Figura 30: Pantalla de login

Vista del login. Se puede realizar el inicio de sesión como una empresa o como un usuario seleccionando un *switch*.

The screenshot shows a registration form for 'ZITION' on a web browser. The form includes fields for mobile number, email, name, surname, password, and password confirmation. It also features a reCAPTCHA verification step. A link for existing users to log in is provided at the bottom.

ZITION

¿TIENES UN NEGOCIO 🏪 Y QUIERES OFRECER UN SERVICIO DE CITA PREVIA 📅 SIN COMPLICARTE?
¿QUIERES EVITAR HACER COLAS 🎩 Y RESERVAR CITA PREVIA EN LOS LOCALES QUE FRECUENTAS?
¡EN ESE CASO ZITION ES TU WEB 🎉🎉!

Número de móvil
Número de móvil +34 6

Dirección de correo electrónico
Introduce el correo
Nunca compartiremos tus datos con terceros.

Nombre
Nombre

Apellidos
Apellidos

Contraseña
Contraseña Mostrar

Confirmar Contraseña
Repite contraseña

I'm not a robot reCAPTCHA Privacy - Terms

Ya tienes una cuenta? Inicia sesión ahora!

Sign Up

Figura 31: Pantalla de registro de un usuario normal

Vista del formulario de registro de un usuario con la recaptcha de Google para evitar bots.

The screenshot shows a web browser window for the Zitation platform. The URL is zitation.herokuapp.com/registroNegocio. The page has a light blue header with the Zitation logo and a navigation bar with links like 'INCIDENCIA COVID-19' and 'INICIAR SESIÓN'. The main content area is titled 'REGISTRO' and contains a form for registering a business. The fields include:

- NIF de la empresa: A12345678
- Email address: [redacted]
- Nombre: [redacted]
- Contraseña: [redacted] (with a 'Mostrar' link)
- Confirmar Contraseña: [redacted]
- Dirección del establecimiento:
 - Ci La calle: [redacted]
 - Número de la calle: 12
 - Código postal: 50011
- Duración media de los servicios: [redacted]
- Capacidad:
 - Capacidad: [redacted]
 - La capacity es el número de clientes que puede atender simultáneamente en su negocio.
- Categoría: [redacted]
- reCAPTCHA: A standard reCAPTCHA interface with a checkbox labeled 'I'm not a robot' and a CAPTCHA image.
- Sign Up: A blue button at the bottom right.

Figura 32: Pantalla de registro de una empresa

Vista del formulario de registro de una empresa con la recaptcha de Google para evitar bots. Se piden varios datos relaciones con ubicación y detalles de los servicios a ofrecer.

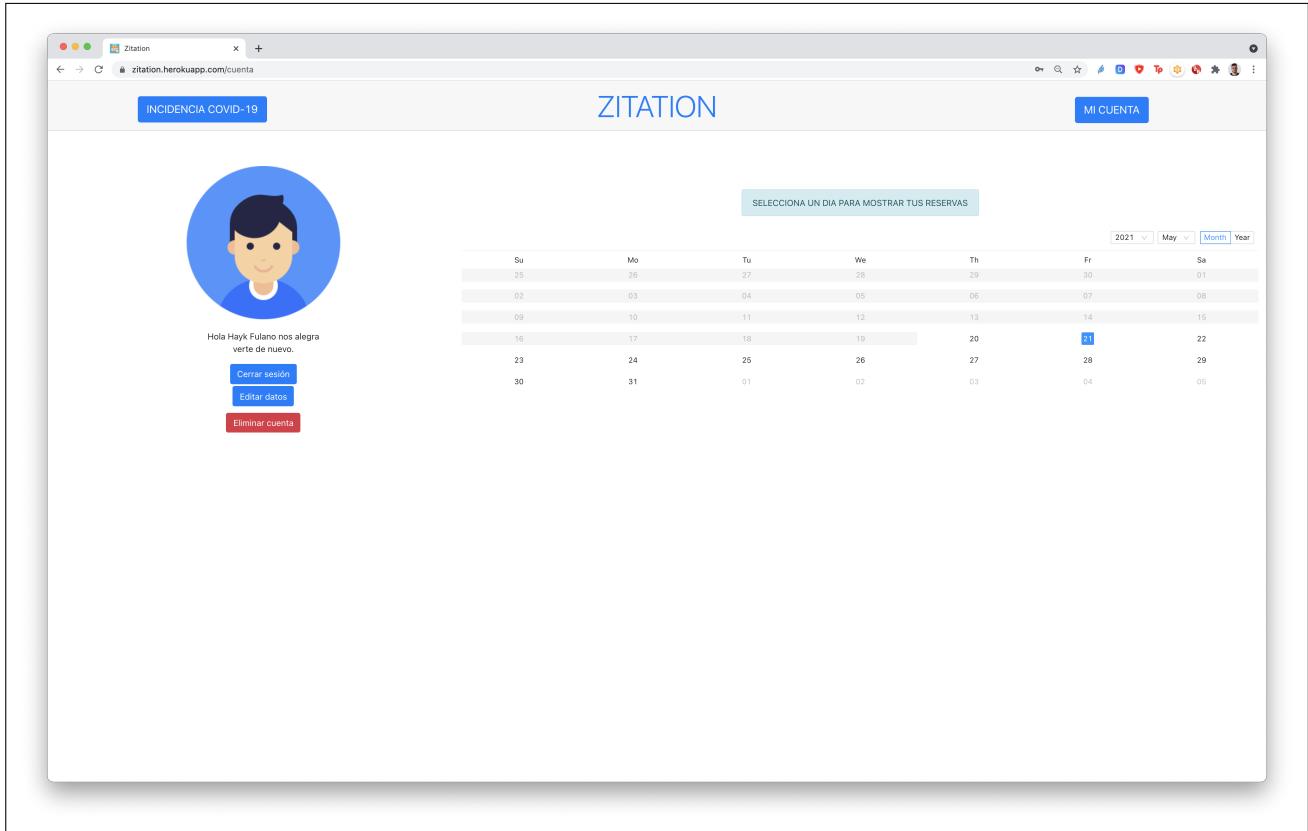


Figura 33: Pantalla de perfil del usuario

Vista de la pantalla del perfil de un usuario, donde se ven botones como cerrar sesión, eliminar cuenta y modificar los datos de un usuario. También se ve un calendario que al seleccionar abre las reservas para ese día de este usuario.

The screenshot shows a web application interface for editing company information. At the top, the title 'ZITION' is displayed. Below it, a message says 'Rellena aquellos campos que quieras modificar.' (Fill in those fields you want to modify.).

Descripción de su empresa:
Valor actual: null
Escribe una descripción de tu empresa

Cuentos más detalles mejor para tus clientes.

Duración media de los servicios que ofrece:
Valor actual: 30
0
Indique el valor en minutos.

Capacidad de los servicios:
Valor actual: 2
1
Esto indica cuantos clientes puede atender simultáneamente en su empresa.

Quiero actualizar mi horario
 Dispongo de horario partido

Horarios: Solo se actualizan los días que seleccionas una hora. Si no tocas uno de los selectores de hora NO se fijará la hora para ese día.

Turno 1	Lunes	Martes	Miercoles	Jueves	Viernes	Sábado	Domingo
Horario apertura turno 1	00:00	00:00	00:00	00:00	00:00	00:00	00:00
Horario cierre turno 1	00:00	00:00	00:00	00:00	00:00	00:00	00:00

Turno 2

Turno 2	Lunes	Martes	Miercoles	Jueves	Viernes	Sábado	Domingo
Horario apertura turno 2	00:00	00:00	00:00	00:00	00:00	00:00	00:00
Horario cierre turno 2	00:00	00:00	00:00	00:00	00:00	00:00	00:00

Contraseña:
Introduce la contraseña para validar tus datos.
Introduce la contraseña para poder validar tus datos.

Figura 34: Pantalla de perfil de para editar los datos de un usuario

Pantalla de modificación de datos de una empresa.

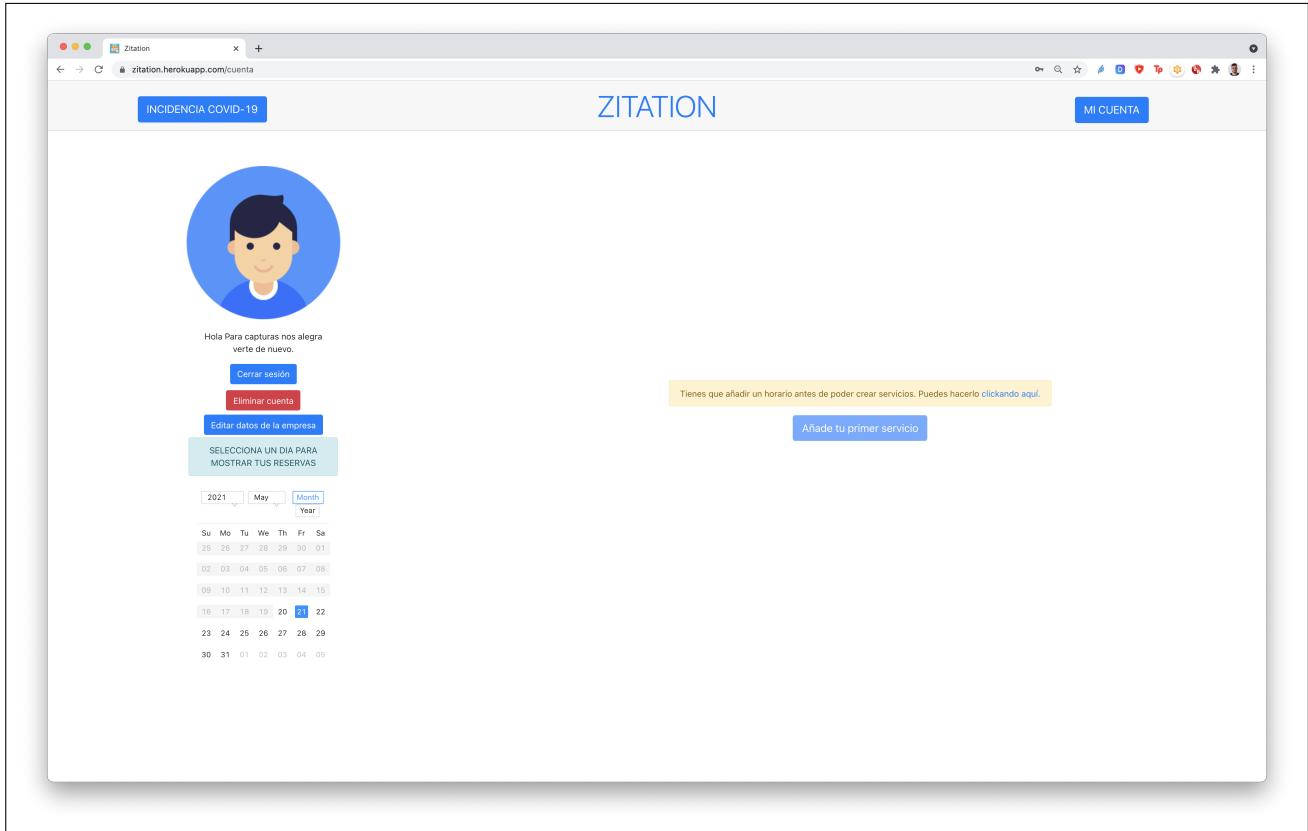


Figura 35: Pantalla de perfil de una empresa sin horario registrado

Pantalla de perfil de una empresa con botones de cerrar sesión, eliminar cuenta, modificar datos de la empresa y un calendario que al seleccionar el día muestra las reservas para ese día. A la derecha se listarán los servicios pero la empresa no ha registrado todavía su horario por lo que no puede añadir un servicio.

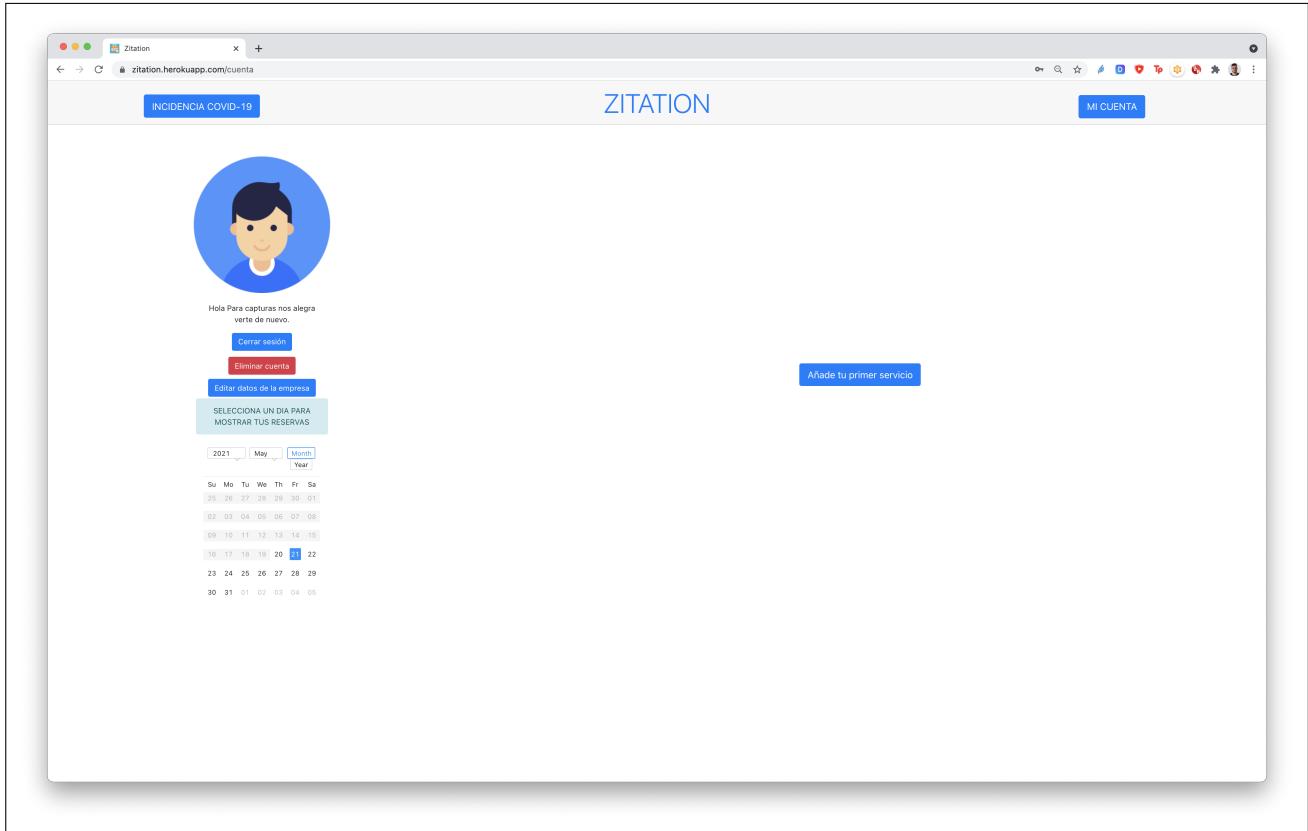


Figura 36: Pantalla de perfil de una empresa sin servicios

Perfil de una empresa que ya ha modificado su horario de apertura, se le permite añadir servicios.

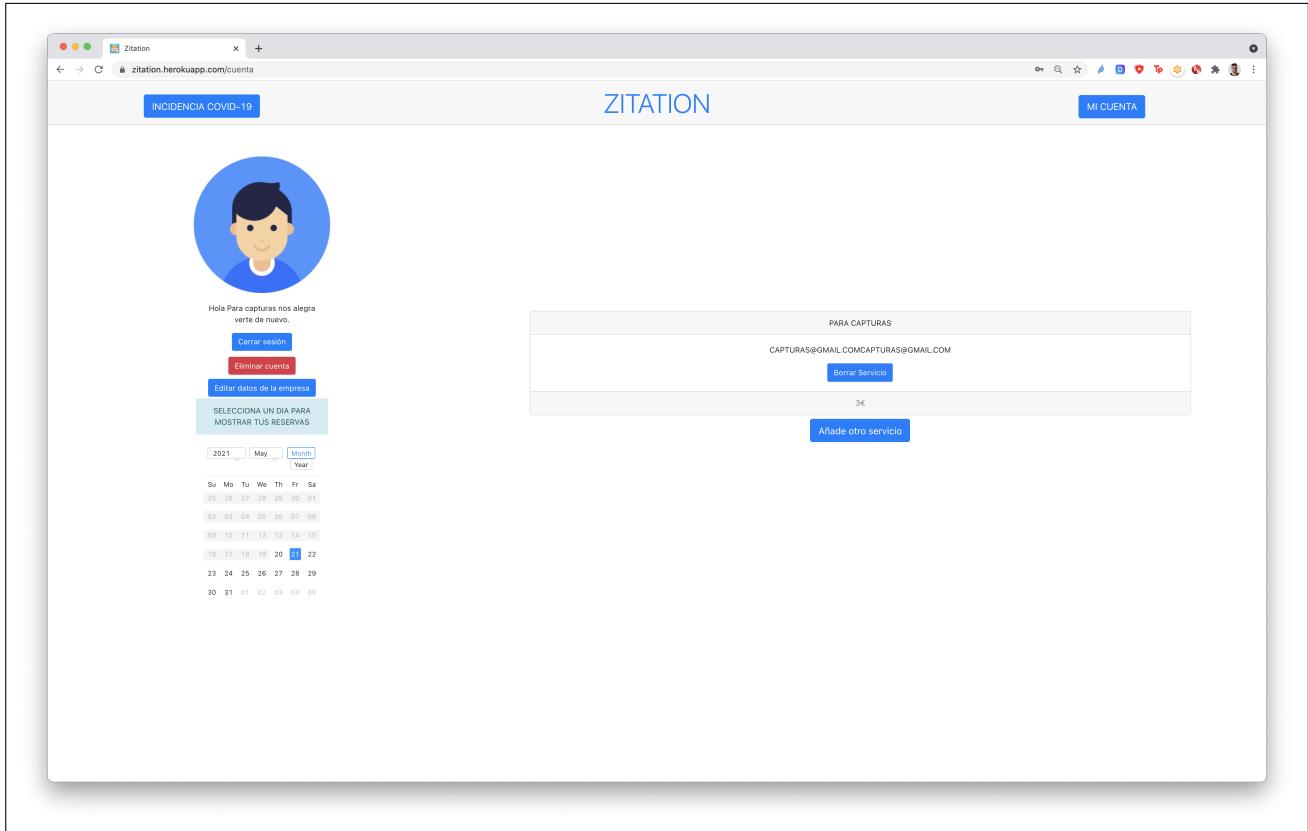


Figura 37: Pantalla de perfil de una empresa con servicio registrado

Perfil de una empresa con su primer servicio, se le permite borrar este.

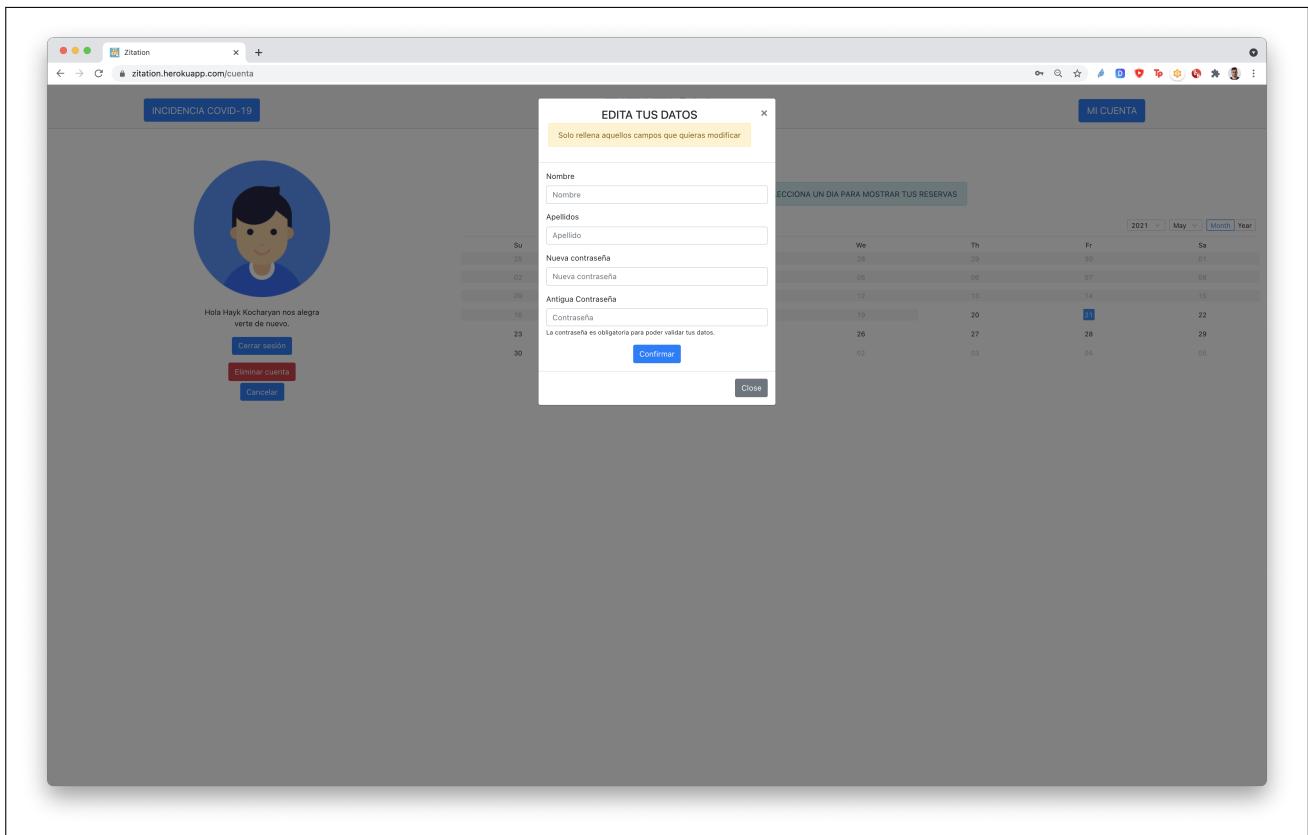


Figura 38: Pantalla de perfil de para editar los datos de un usuario

Pantalla del formulario para actualizar los datos de un usuario.

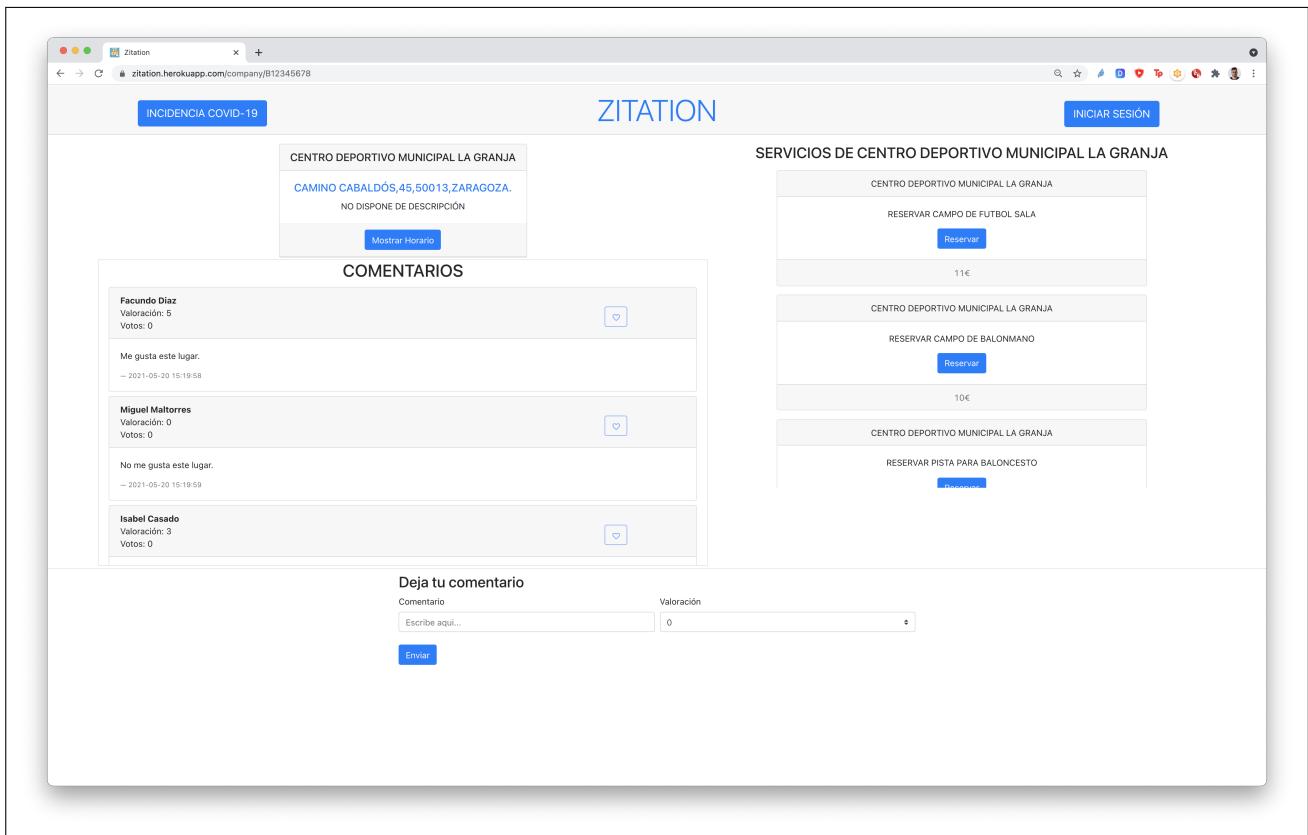


Figura 39: Pantalla de detalles de una empresa

Vista de los detalles de una empresa. Se permite añadir comentarios, ver el horario y ver los diferentes servicios de esta empresa.

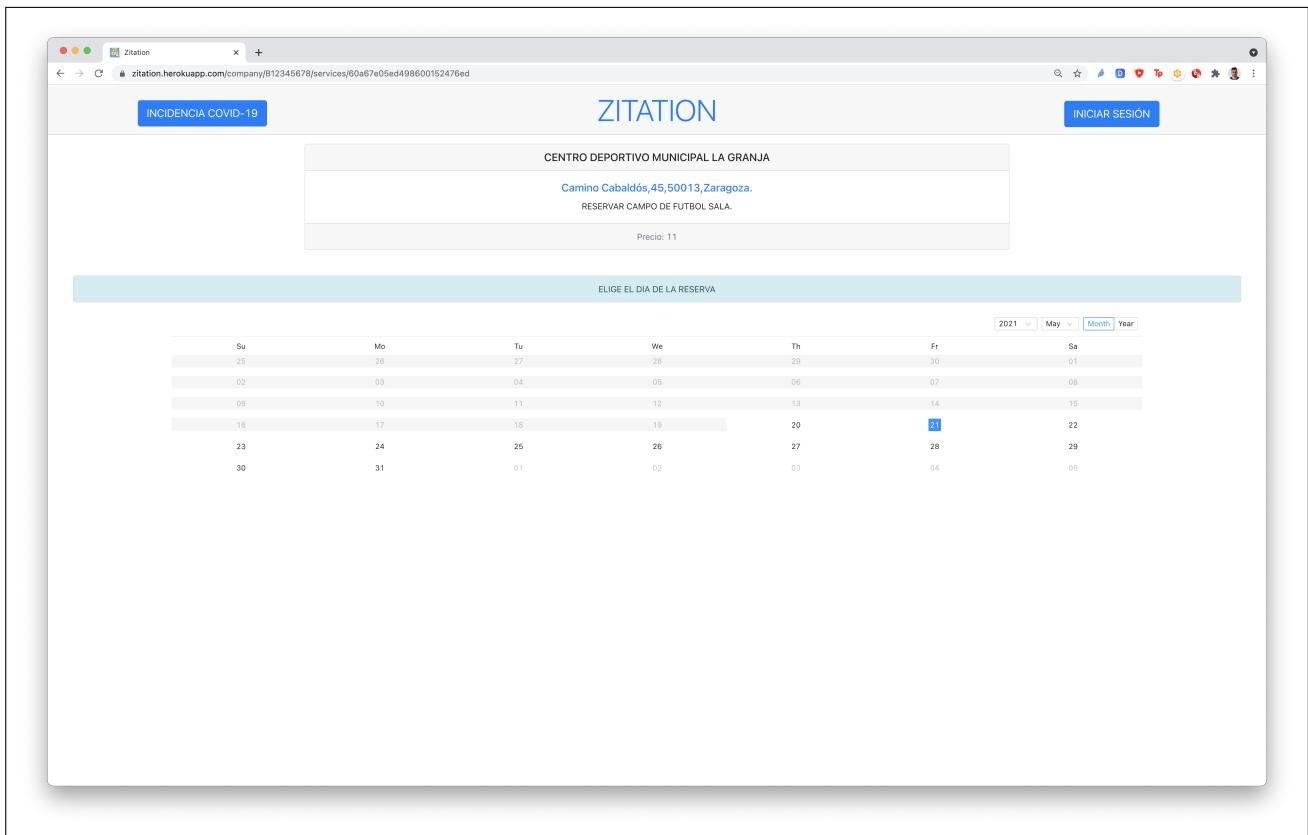


Figura 40: Pantalla de información de un servicio de una empresa

Vista de los detalles de un servicio, acceso al calendario para poder elegir el dia y reservar.

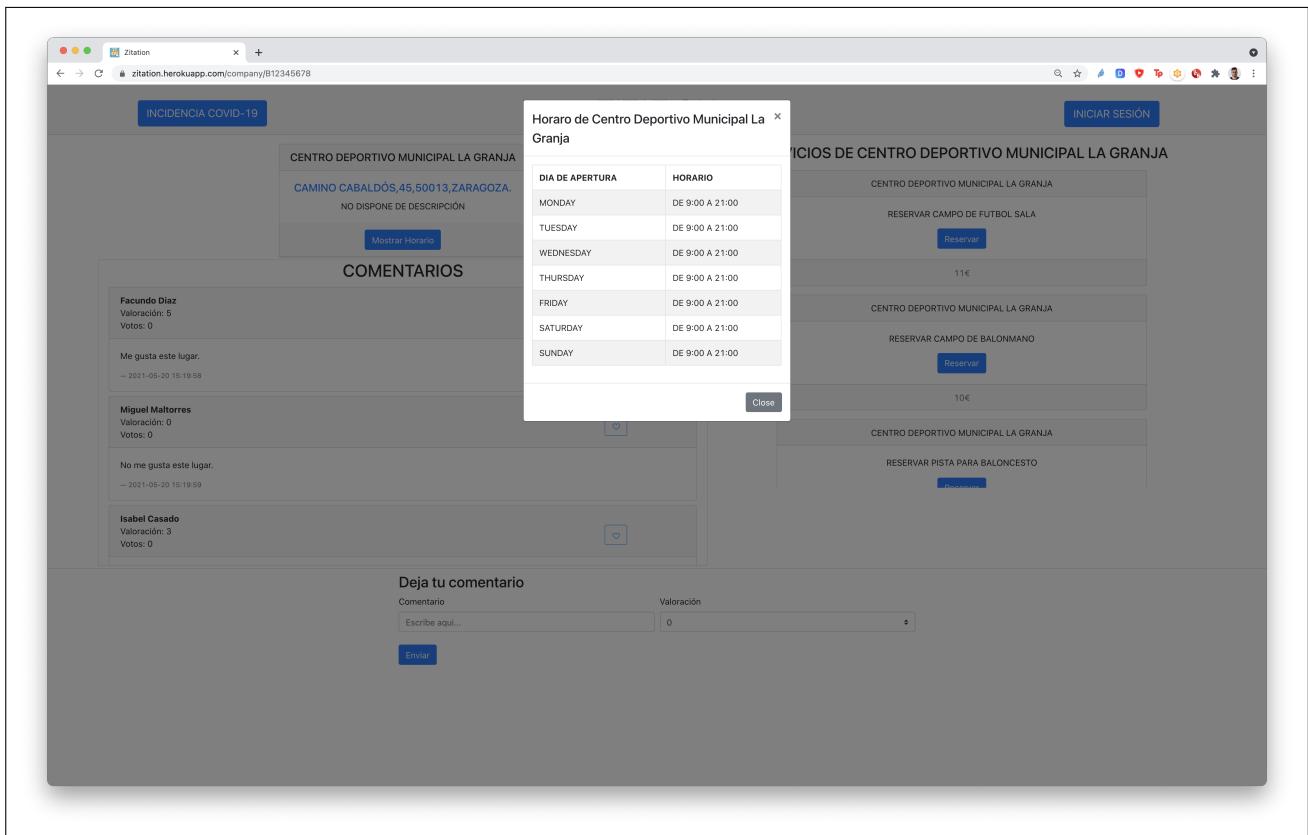


Figura 41: Pantalla de horario de una empresa

Pantalla al mostrar el horario de una empresa

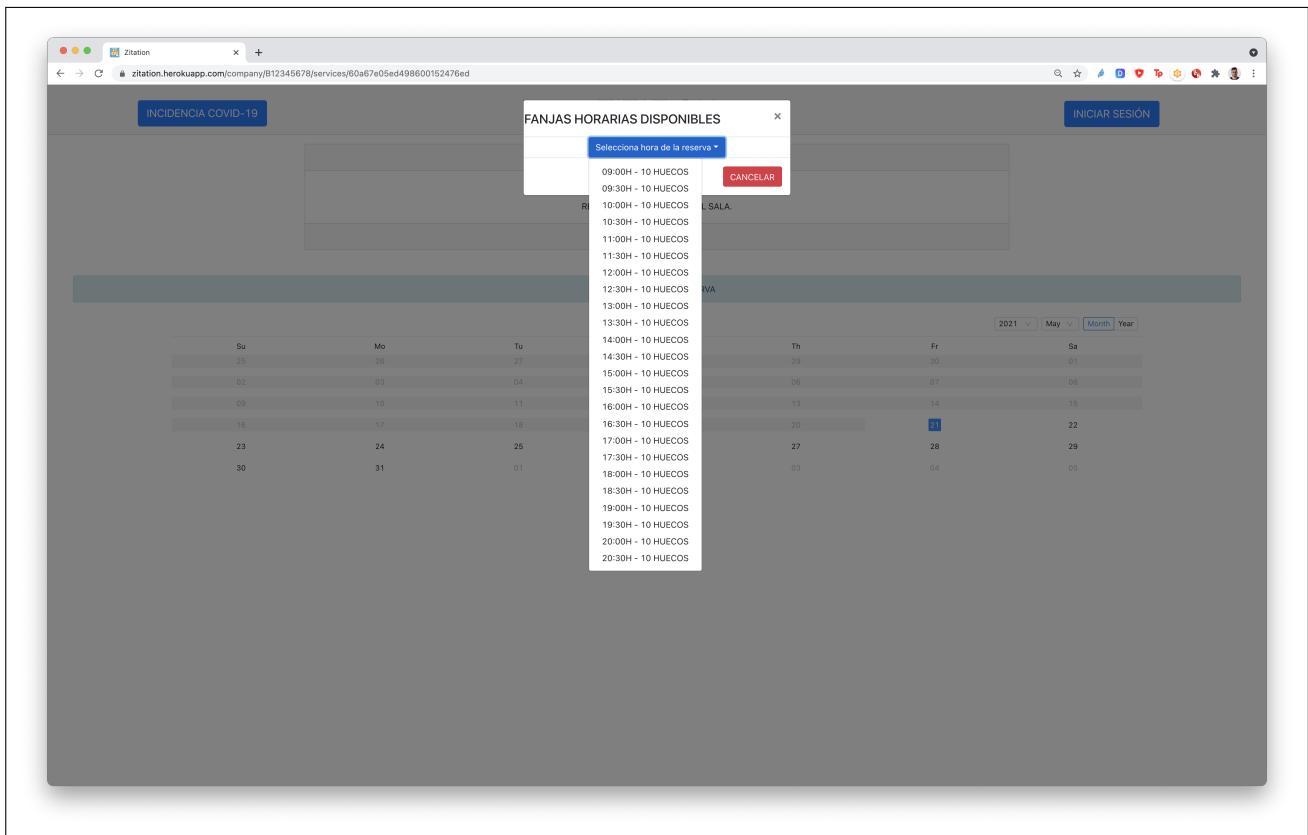


Figura 42: Pantalla para ver las horas y huecos disponibles de un servicio

Pantalla para ver las franjas horarias y huecos disponibles de un dia a reservar.

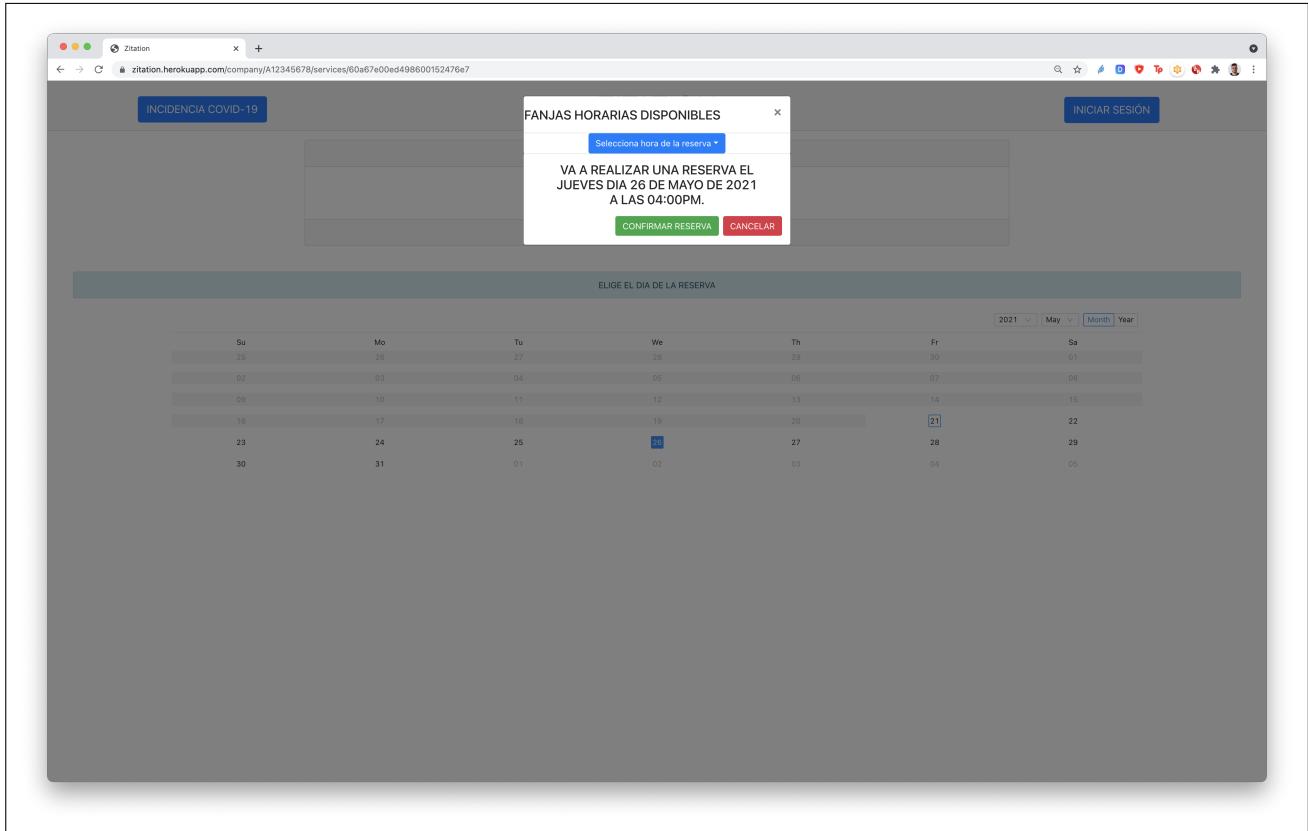


Figura 43: Pantalla de vista previa de la reserva que se va a realizar

Pantalla de vista previa de la reserva que se va a realizar.

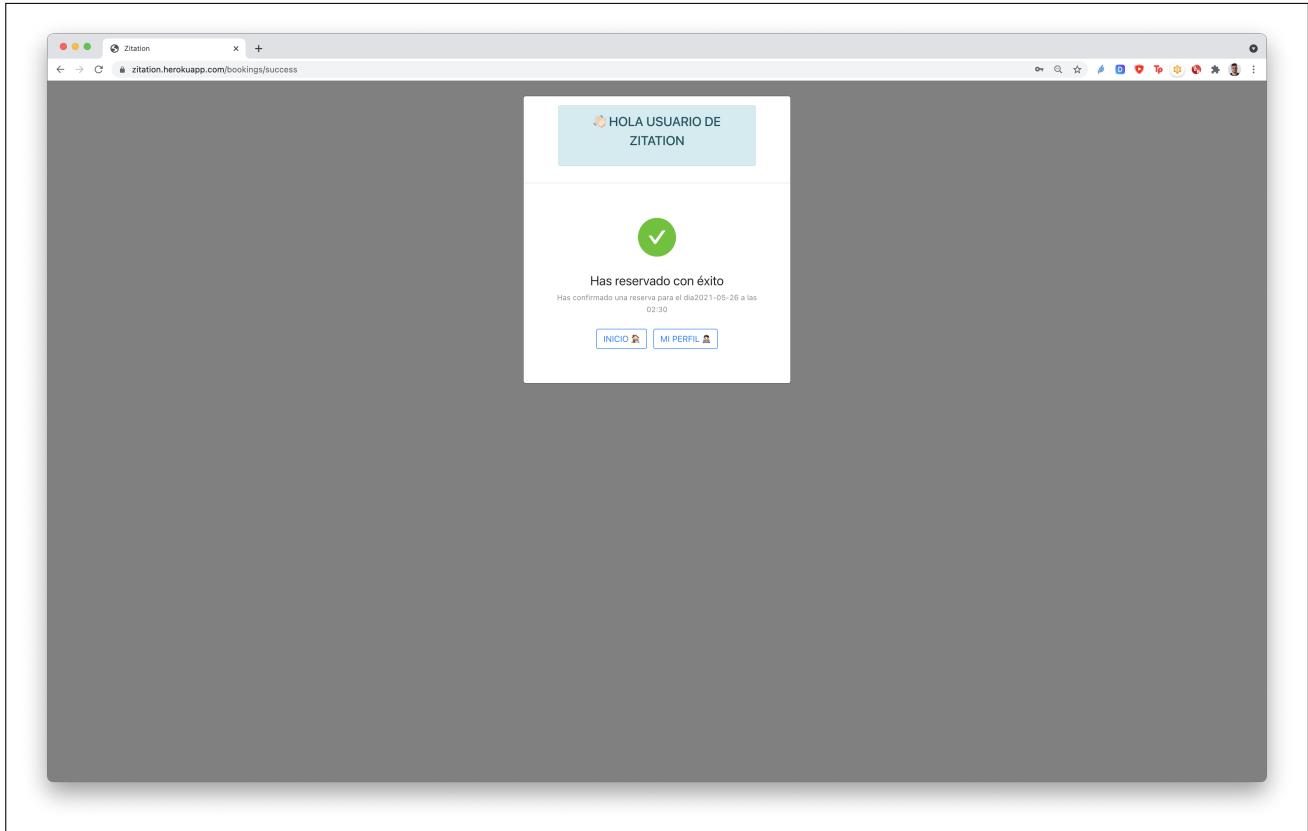


Figura 44: Pantalla de confirmación de reserva

Pantalla de confirmación de una reserva.

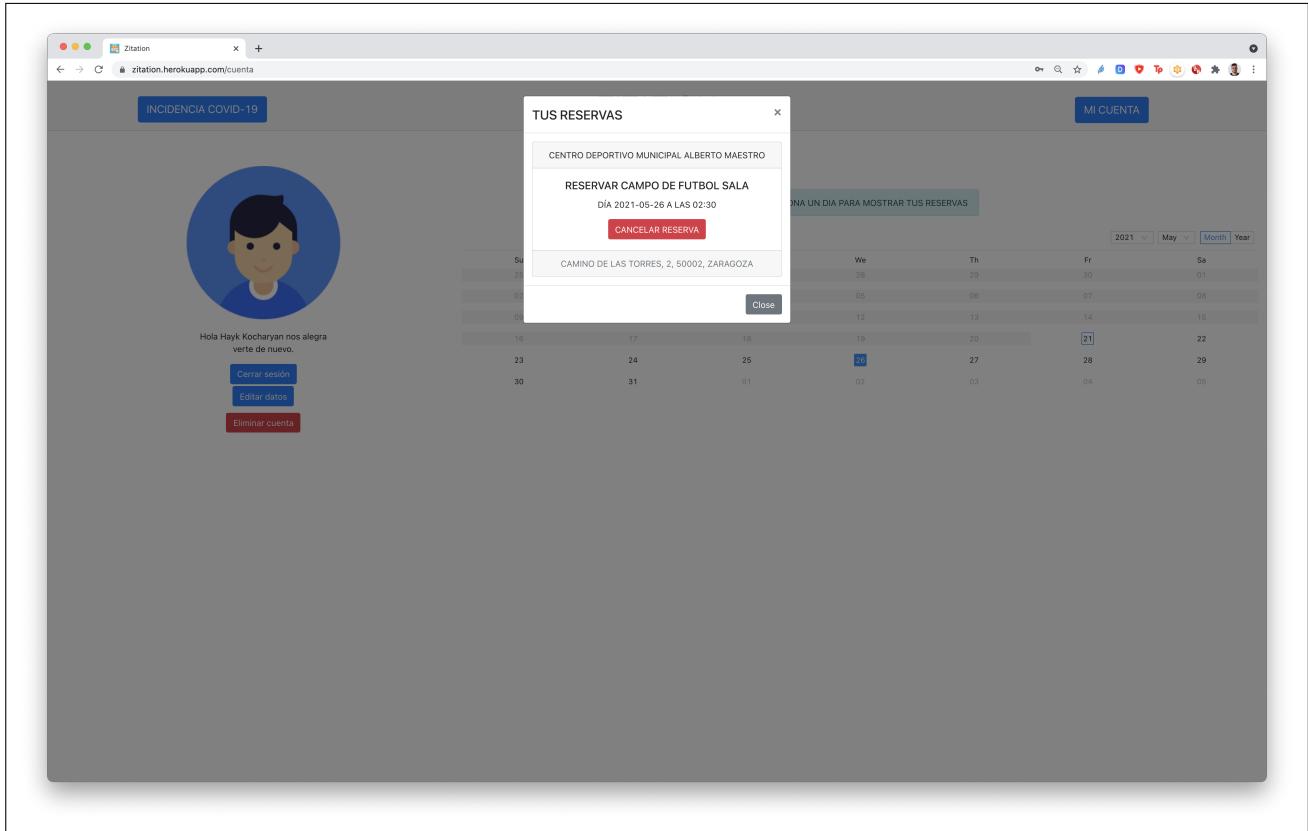


Figura 45: Pantalla de vista previa de las reservas que tiene una empresa

Pantalla de listado de reservas de un usuario para un día que a seleccionado.

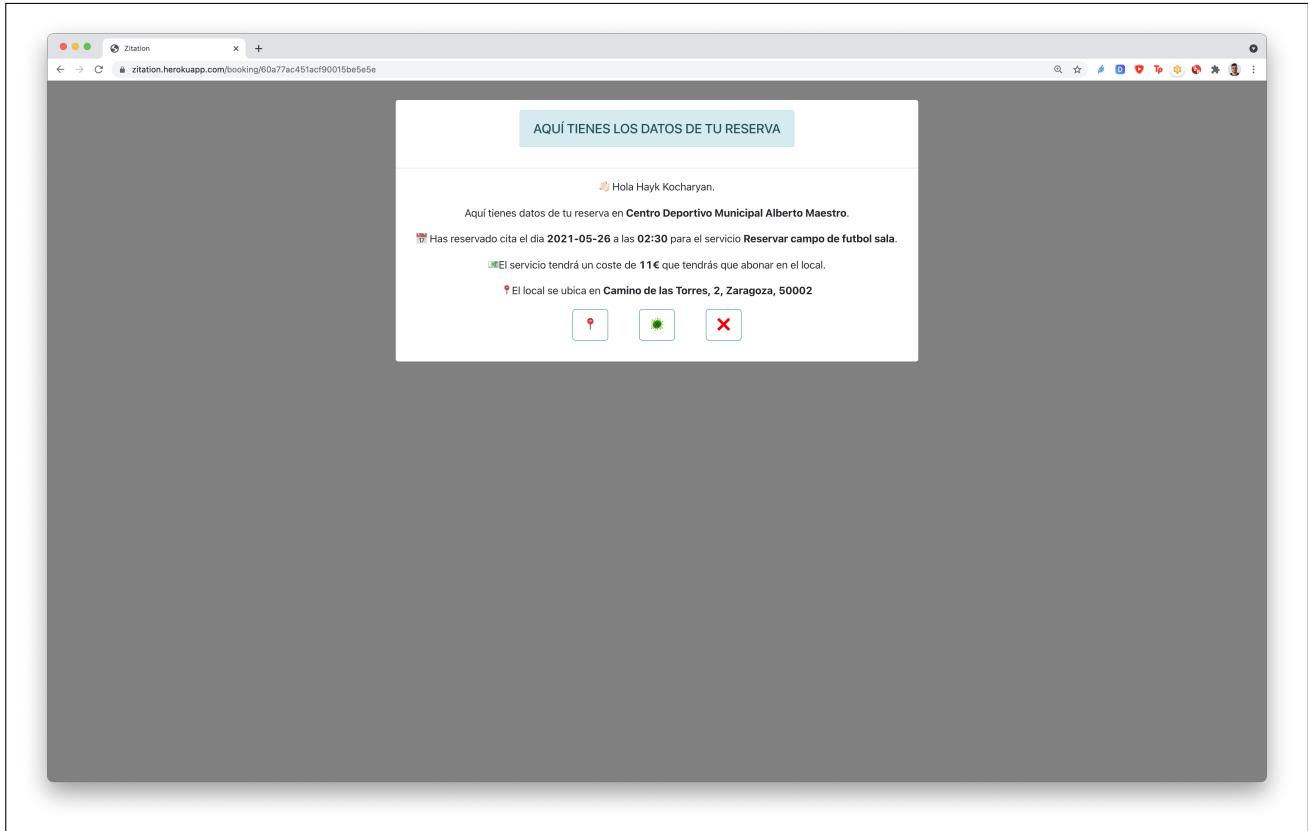


Figura 46: Pantalla de resumen de reserva al escanear el código QR

Pantalla de detalles de una reserva al escanear el código QR mandado por correo.

4. Conclusiones

4.1. Conclusiones del Proyecto

4.2. Problemas afrontados

Al comienzo del proyecto surgió el problema de decidir que framework utilizar en frontend. Esto supuso tener que valorar ambos y aprender un poco de cada uno, restando tiempo al resto del proyecto. Aunque esto es una situación que puede ocurrir en muchos proyectos reales. Uno de los principales problemas encontrados a sido el uso de **JavaScript**, este lenguaje a priori parece ser sencillo y fácil de controlar, pero a lo largo que se va avanzando hemos visto que hay muchas cosas que pensamos que tenemos bajo control pero al ver los errores vemos que el código no ejecuta lo que nosotros creemos que debería.

Un problema fue la falta de concordancia entre la hora de los servidores de **Heroku** y nuestra hora local. Esto último derivó en confusiones, ya que por un momento pensábamos que las tareas programadas estaban funcionando mal, y no era así.

4.3. Problemas a futuro

Dada la inexperiencia en las nuevas tecnologías que se han utilizado, se comenzó diseñando algunos elementos no muy bien estructurados. En caso de continuar con el proyecto a nivel empresarial, se debería refactorizar algunas partes de código para aumentar la organización y reusabilidad.

En el servidor no se ha aprovechado todo el potencial de **Express**, ya que **Express** permite definir *middlewares* para cada recurso de la API y tan solo lo hemos empleado una vez. posible que la solución aplicada en el servidor *backend* no escale correctamente, actualmente para las pruebas que hemos realizado y para pocos usuarios es una solución sólida, pero muy posiblemente lleve a condiciones de carrera y fallos si es utilizada por muchos usuarios. Otro posible error a futuro serían las tareas programadas, actualmente hay 2 tareas programadas que se ejecutan cada 4 y 2 horas, estas tareas en el futuro debería de hacerse en otra máquina. Es decir para evitar una sobrecarga habría que crear un servicio en una o varias máquinas encargadas exclusivamente a actualizar los datos de casos covid o actualizar estadísticas. Un claro problema a futuro es que dependemos de **Transparencia Aragón**, cuando esta plataforma pública dejé de proporcionarnos un Excel con los datos de casos de coronavirus en Aragón una de las funcionalidades clave de nuestro proyecto dejará de funcionar.

El diseño web se debería mejorar debido a que se han realizado con conocimientos básicos de los frameworks y eso lleva a no se haya exprimido del todo la potencia de estos. Por ejemplo se podría mejorar el calendario para dar mas funcionalidades como mostrar los días que no se puedan hacer reservas sin tener que seleccionar el día para ello. Otro ejemplo es la forma en la que una empresa modifica su horario, es una funcionalidad que se debería mejorar ya que está bastante pobre y no es lo más cómodo. Por último, mas consistencia en la web a la hora de tratar la retroalimentación en acciones que llevan a dar un feedback positivo o negativo. Esto ocurre porque fue una funcionalidad que se introdujo a última hora por lo que no todas las acciones de la web tienen este tipo de retroalimentación.

4.4. Conclusiones individuales

4.4.1. Hayk

He trabajado muy a gusto con mis compañeros, era la primera vez que realizaba un trabajo en grupo con ellos y sinceramente he estado muy cómodo y volvería a repetir. Como mejora para la siguiente ocasión quizás seria la organización, un poco de mejora en este aspecto ayudaría a llevar mucho mejor el proyecto.

En cuanto a las tecnologías usadas, si hubiésemos tenido más tiempo para aprender React, el producto final habría sido muchísimo mejor. Me he dado cuenta que al final es cuando más controlaba del framework y he tenido que retocar muchas cosas que había hecho al principio. Muchos de estos retoques han sido parches mal hechos para mantener la funcionalidad, ya que si tuvieses que rehacer de nuevo la componente, llevaría muchas horas por lo que no llegaríamos a la entrega.

He echado en falta que el *backend* estuviese un paso por delante del front. En nuestro caso hemos ido casi sincronizados, o un día por delante el *backend* respecto al *frontend*. También es verdad que en alguna ocasión puntual el *frontend* no ha podido avanzar debido a no disponer de recursos del *backend* pero se ha resuelto con éxito estas situaciones.

4.4.2. Alberto

En esta asignatura he seguido profundizando en las tecnologías web. Me ha ayudado bastante haber cursado previamente Ingeniería Web. Por ello, en esta ocasión me quería centrar más en el desarrollo del frontend puro y aprender un framework dedicado a ello, en este caso React. También me ha tocado aprender Javascript, pero al final del proyecto puedo decir que estoy muy contento con nuestro trabajo y con las cosas que he aprendido a lo largo de él. Con más tiempo nos gustaría seguir añadiendo cosas y detalles. También añadir que valoro bastante haber aprendido el stack **MERN**.

4.4.3. Luis

Estoy satisfecho con el trabajo realizado. He aprendido **JavaScript** lo cual me interesaba bastante aprender antes de terminar la carrera y conocer un *stack* de desarrollo tan popular como **MERN** utilizado en la industria me parece positivo. A pesar de la distancia y no habernos podido reunir físicamente en la universidad como grupo nos hemos organizado bastante bien y hemos hecho un buen trabajo. En este proyecto he podido aplicar las metodologías de trabajo vista en asignaturas previas y desarrollar para *backend*, ya que en asignaturas anteriores había hecho *frontend*.

4.4.4. German

He realizado un proyecto sobre unas tecnologías que no había usado con anterioridad y no solo he aprendido muchísimo por el camino si no que también he terminado realizando un trabajo bien hecho y robusto por lo que estoy muy satisfecho con el resultado final del trabajo. No solo he aprendido a crear una API REST en **NodeJS** y **Express** si no que también he estudiado buenas prácticas para su creación que son independientes de la plataforma de desarrollo. Al igual que otros compañeros, conforme se fueron aprendiendo las tecnologías se fueron arreglando funcionalidades de la aplicación mas antiguas que se habían realizado sin esos conocimientos necesarios.

También he aprendido a usar con soltura **MongoDB** y **Mongoose** para realizar todo tipo de consultas. En último lugar, he trabajado muy cómodo con el equipo ya que siempre estaban dispuestos a escuchar y a echar una mano por lo que el ambiente de trabajo ha sido inmejorable.

References

- [1] *dependency-cruiser*. URL: <https://github.com/sverweij/dependency-cruiser>.
- [2] React Leaflet. *React Leaflet*. URL: <https://react-leaflet.js.org/>.
- [3] Bill Luo. *React Hook Form*. URL: <https://react-hook-form.com>.
- [4] Lucas Merencia. *Node cron*. URL: <https://www.npmjs.com/package/node-cron>.
- [5] Mongoose. *Mongoose*. URL: <https://mongoosejs.com>.
- [6] Nchaulet. *Node Geocoder*. URL: <https://www.npmjs.com/package/node-geocoder>.
- [7] NodeJS. *Crypto*. URL: <https://nodejs.org/api/crypto.html>.
- [8] *Passport JWT*. URL: <http://www.passportjs.org/packages/passport-jwt/>.
- [9] React. *React Bootstrap*. URL: <https://react-bootstrap.github.io>.
- [10] Andris Reinman. *Nodemailer*. URL: <https://nodemailer.com/about/>.
- [11] Design Revision. *Shards React*. URL: <https://designrevision.com/downloads/shards-react/>.
- [12] Guy On Roche. *ExcelJS*. URL: <https://www.npmjs.com/package/exceljs>.
- [13] React Router. *React Router*. URL: [https://reactrouter.com/](https://reactrouter.com).
- [14] *Transparencia Aragón*. URL: <https://transparencia.aragon.es/COVID19>.
- [15] XTech. *Antd UI-Kit*. URL: <https://ant.design>.

Anexo

Hayk Kocharyan

757715@unizar.es

Luis García

739202@unizar.es

Germán Garcés

757024@unizar.es

Alberto Calvo

760739@unizar.es

22 de mayo de 2021

Perceivable						
Information and user interface components must be presentable to users in ways they can perceive.						
Tipology	Checking	Techniques	Result	Problems	Line numbers	
1.1.1-Non-text Content						
Forms	Form controls without label i	H44 H65	X	1	784	
Images	Images without "alt" attribute i	H97	X	5	784 784 784 784 784	
1.3.1-Info and Relationships						
Forms	Form controls without associated label i	H44 H65	X	1	784	
Structure and semantics	None h1 element in the document i	H42	X	1		
1.3.3-Sensory Characteristics						
Presentation	Sensory Characteristics i	G96	?	1		
1.4.1-Use of Color						
Presentation	Information using color i	G14 G122 G182 G183	?	1		
1.4.3-Contrast (Minimum)						
Presentation	Contrast i	G18 G148 G174	?	1		
	Contrast for large fonts i	G145 G148 G174	?	1		
1.4.5-Images of Text						
Images	Images that can be replaced by markup language i	C22 C30 G140	?	1		

Figura 1: Resumen TAW análisis

Operable						
User interface components and navigation must be operable.						
Tipology	Checking	Techniques	Result	Problems		
2.1.1-Keyboard						
Scripts	Automatic movement of the focus i	G90	?	1		
2.1.2-No Keyboard Trap						
Web page	Changing focus with the keyboard i	G21	?	1		
2.2.1-Timing Adjustable						
Web page	Session time limit i	G133 G198	?	1		
	Limit of time controlled by a script i	G198 G180 SCR16	?	1		
	Reading texts on the move i	G4 G198 SCR33 SCR36	?	1		
2.2.2-Pause, Stop, Hide						
Web page	Moving or blinking content i	G4 SCR33 G187 G152 SCR22 G186 G191	?	1		
2.3.1-Three Flashes or Below Threshold						
Presentation	Flashes below threshold i	G19 G176 G15	?	1		

Figura 2: Errores operables parte 1

2.3.1-Three Flashes or Below Threshold						
Presentation	Flashes below threshold		G19 G176 G15		1	
2.4.1-Bypass Blocks						
Navigation	Skip repeated blocks of content		G1 G123 G124		1	
	Blocks of content		H50 H70 SCR28		1	
Structure and semantics	None h1 element in the document		H62		1	
2.4.2-Page Titled						
Web page	Descriptive title		G88		1	1
2.4.3-Focus Order						
Navigation	Logical sequence of navigation		G52 H4 SCR26 SCR37 SCR27		1	
2.4.5-Multiple Ways						
Web site	Multiple ways of localization		G123 G64 G63 G161 G126 G185		1	
2.4.7-Focus Visible						
Scripts	Changing focus with the event 'onFocus'		F55		1	

Figura 3: Errores operables parte 2

Understandable						
Information and the operation of user interface must be understandable.						
Tipology	Checking		Techniques	Result	Problems	
3.1.1-Language of Page						
Web page	Language stated in the document and actual language		H57		1	
3.1.2-Language of Parts						
Web page	Changes in language		H58		1	
3.2.1-On Focus						
Scripts	Changing content with the event "onfocus"		G107		1	
	Changing focus with the event "onfocus"		F55		1	
	Opening a new window with the event "onfocus"		G107		1	
	Opening a new window with the event "onload"		F52		1	
Web page	Changing content		G107		1	
3.2.2-On Input						
Forms	Form with no standard submission method		H32		1	
	Changes caused by the event "onchange" in a selector		H84		1	
3.2.3-Consistent Navigation						
Web site	Consistent navigation		G61		1	

Figura 4: Errores entendibles parte 1

3.2.4-Consistent Identification						
Web site	Consistent denomination		G197		1	
3.3.1-Error Identification						
Forms	Identification of errors in form fields		G83 SCR18		1	784
	Identifique los valores que deben indicarse con formatos especiales		G84 G85 SCR18 SCR32		1	784
3.3.2-Labels or Instructions						
Forms	Labeling of form controls		H44 H65		1	784
3.3.3-Error Suggestion						
Forms	Provide suggestions for incorrect values in forms		G83 G84 G85 G177 SCR18 SCR32		1	784
3.3.4-Error Prevention (Legal, Financial, Data)						
Forms	Preventing errors in legal, financial or data forms		G164 G98 G155		1	784
	Prevention of errors with delete actions in legal, financial or data forms		G99 G168 G155		1	784
	Prevention of errors in tests		G98 G168		1	784

Figura 5: Errores entendibles parte 2

Robust						
Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.						
Tipology	Checking		Techniques	Result	Problems	Line numbers
4.1.1-Parsing						
Web page	Web page well-formedness		G134		1	784
4.1.2-Name, Role, Value						
Forms	Form controls without label		H44 H65		1	

Figura 6: Errores robustos