

*Equipo Ryan Dahl*

# Sistemas y Tecnologías Web

11 de Septiembre del 2020

## Descubre Aragón



Eduardo Gimeno - 721615

Gonzalo Berné - 715891

Jorge Fernández - 721529

Joaquín Puyuelo - 604736

---

# Índice

<b>Descubre Aragón</b>	<b>1</b>
RESUMEN DEL PROYECTO	3
DATOS DEL EQUIPO:	3
LISTADO DE REQUISITOS	4
PROPUESTAS SIMILARES	6
ARQUITECTURA	7
Vista de módulos	8
Vista de componente y conector	9
Vista de despliegue	10
MODELO DE DATOS	11
API REST	19
IMPLEMENTACIÓN	20
MODELO DE NAVEGACIÓN	25
Pantalla Login	25
Pantalla Registry	26
Pantalla Index-User	27
Pantalla Profile	28
Pantalla Change-Password	29
Pantalla Chat-User	29
Pantalla Index-Admin	30
ANALÍTICAS	31
DESPLIEGUE DEL SISTEMA	36
VALIDACIÓN	38
PROBLEMAS ENCONTRADOS DURANTE EL DESARROLLO	40
Problemas con el registro y autenticación de los usuarios	40
Problemas a la hora de importar datos	40
ANÁLISIS DE PROBLEMAS POTENCIALES	43
DISTRIBUCIÓN DE TIEMPO	44
CONCLUSIONES	45
VALORACIÓN PERSONAL DE CADA MIEMBRO	46
ANEXO	47

---

## 1. RESUMEN DEL PROYECTO

### DATOS DEL EQUIPO:

Nombre: Equipo Ryan Dahl

Integrantes:

Nombre	NIP	Función
Eduardo Gimeno	721615	Back-End
Joaquín Puyuelo	604736	Front-End
Gonzalo Berné	715891	Front-End
Jorge Fernández	721529	Back-End

Repositorio del proyecto: <https://github.com/EduardoGimeno/Sistemas-y-Tecnologias-Web>

Aplicación desplegada: <https://turismoaragon.herokuapp.com/>

Back-End desplegado: <http://back-turismoaragon.herokuapp.com/>

Se ha desarrollado una aplicación orientada al turismo en Aragón, que permita consultar y descargar información sobre agencias de viaje, albergues, refugios, alojamientos de turismo rural, alojamientos hoteleros, apartamentos turísticos, cafeterías y restaurantes, campings turísticos, guías turísticos, oficinas de turismo y puntos de información turística (todos estos datos se denominan entradas). Además, permitirá a los usuarios contactar por e-mail con los dueños de los alojamientos para iniciar un proceso de reserva. Habilitando una zona de chat específica para cada conversación de los usuarios con los dueños.

La información que se va a utilizar en la base de datos de la aplicación pertenece a una página web del gobierno de Aragón denominada Aragón Open Data, la cual es de acceso libre. El cual contiene un banco de datos con una gran cantidad de información sobre temas muy diversos como turismo, deporte, economía, medio ambiente, medio rural, salud, etc. Dicha base de datos es una iniciativa que persigue que los datos y la información, especialmente la que poseen las administraciones públicas, estén disponibles para el conjunto de los ciudadanos. La publicación de sus datos se hace de forma abierta y reutilizable por lo que pueden ser utilizados de forma libre en nuestro proyecto.

El usuario final para el que se planteó la aplicación es cualquier tipo de persona que desee realizar turismo en la comunidad autónoma de Aragón. El funcionamiento básico del sistema será el siguiente: Los usuarios deberán registrarse primero en la aplicación introduciendo algunos datos sobre ellos, que serán su nombre, apellidos, fecha de nacimiento, teléfono de contacto, país de origen, provincia, correo electrónico y una contraseña. Una vez tengan una cuenta propia en el sistema, los usuarios podrán consultar distintos datos de guías, restaurantes, hoteles, etc, que se sitúan en Aragón. Además de poder ponerse en contacto con alojamientos ubicados en las cercanías de los lugares que decidan visitar.

### LISTADO DE REQUISITOS

A continuación se puede observar el listado de requisitos iniciales que se planteó previamente al desarrollo del proyecto:

Requisitos funcionales	
Usuario	
ID	Descripción
RFU-1	El sistema permitirá al usuario registrarse
RFU-2	El sistema requerirá al usuario nombre, apellidos, fecha de nacimiento, e-mail, contraseña, teléfono, país y provincia para poder registrarse.
RFU-3	El sistema permitirá al usuario acceder autenticándose mediante e-mail y contraseña
RFU-4	El sistema permitirá al usuario acceder autenticándose utilizando la cuenta de Gmail (autenticación contra Google).
RFU-5	El sistema permitirá al usuario consultar información sobre las entradas.
RFU-6	El sistema permitirá al usuario filtrar las entradas por varios parámetros.
RFU-7	El sistema permitirá al usuario contactar con una entrada de tipo alojamiento mediante un chat.
RFU-8	El sistema permitirá al usuario consultar la ubicación de una entrada en un mapa.
RFU-9	El sistema permitirá al usuario descargar datos de búsquedas en formato CSV y PDF
RFU-10	El sistema permitirá al usuario descargar datos de entradas en formato CSV y PDF
RFU-11	El sistema permitirá al usuario modificar sus datos personales

RFU-12	El sistema permitirá al usuario darse de baja
RFU-13	El sistema permitirá al usuario consultar estadísticas sobre municipios y entradas de tipo alojamiento
<b>Administrador</b>	
RFA-1	El sistema permitirá al administrador acceder autenticándose mediante nombre de usuario y contraseña
RFA-2	El sistema permitirá al administrador gestionar a los usuarios (eliminar, banear, enviar e-mail)
RFA-3	El sistema permitirá al administrador consultar estadísticas sobre los usuarios y sobre los datos importados
<b>Sistema</b>	
RFS-1	El sistema permitirá a la gerencia de las entradas de tipo alojamiento responder a los mensajes de los usuarios en un chat
<b>Requisitos no funcionales</b>	
RNF-1	La contraseña tanto del usuario como del administrador deberá tener como mínimo 8 caracteres alfanuméricos
RNF-2	El sistema guardará las contraseñas encriptadas utilizando MD-5
RNF-3	El sistema calculará diariamente las estadísticas mediante procesos en segundo plano
RNF-4	El sistema generará un enlace que enviará por e-mail a la gerencia de una entrada de tipo alojamiento para permitir responder a un cliente
RNF-5	El enlace que permite a la gerencia de una entrada de tipo alojamiento acceder al chat con un cliente tendrá una validez de 72 horas prorrogables
RNF-6	El sistema mostrará las estadísticas al administrador mediante gráficas
RNF-7	El sistema actualizará las entradas mediante procesos en segundo plano

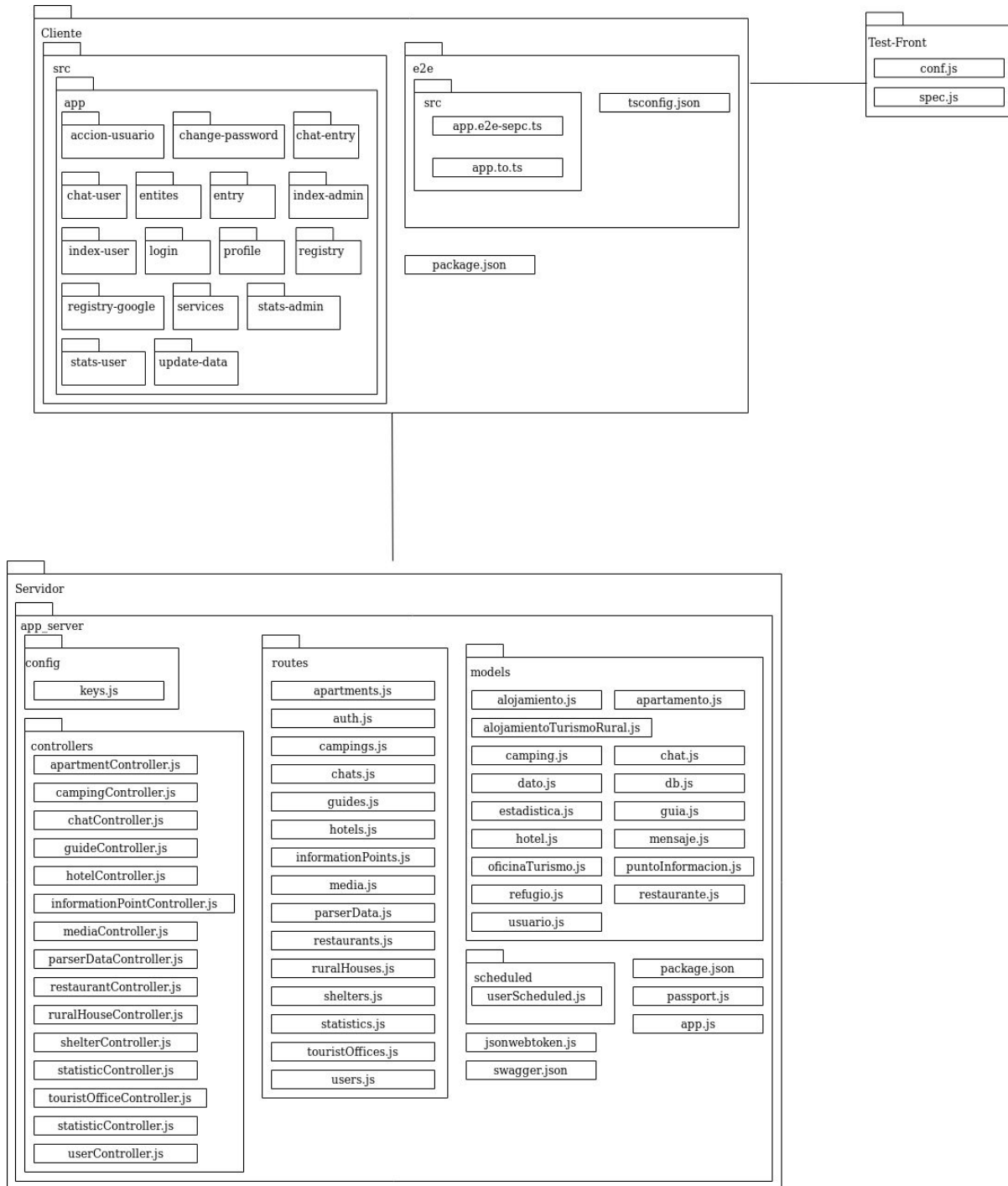
---

## 2. PROPUESTAS SIMILARES

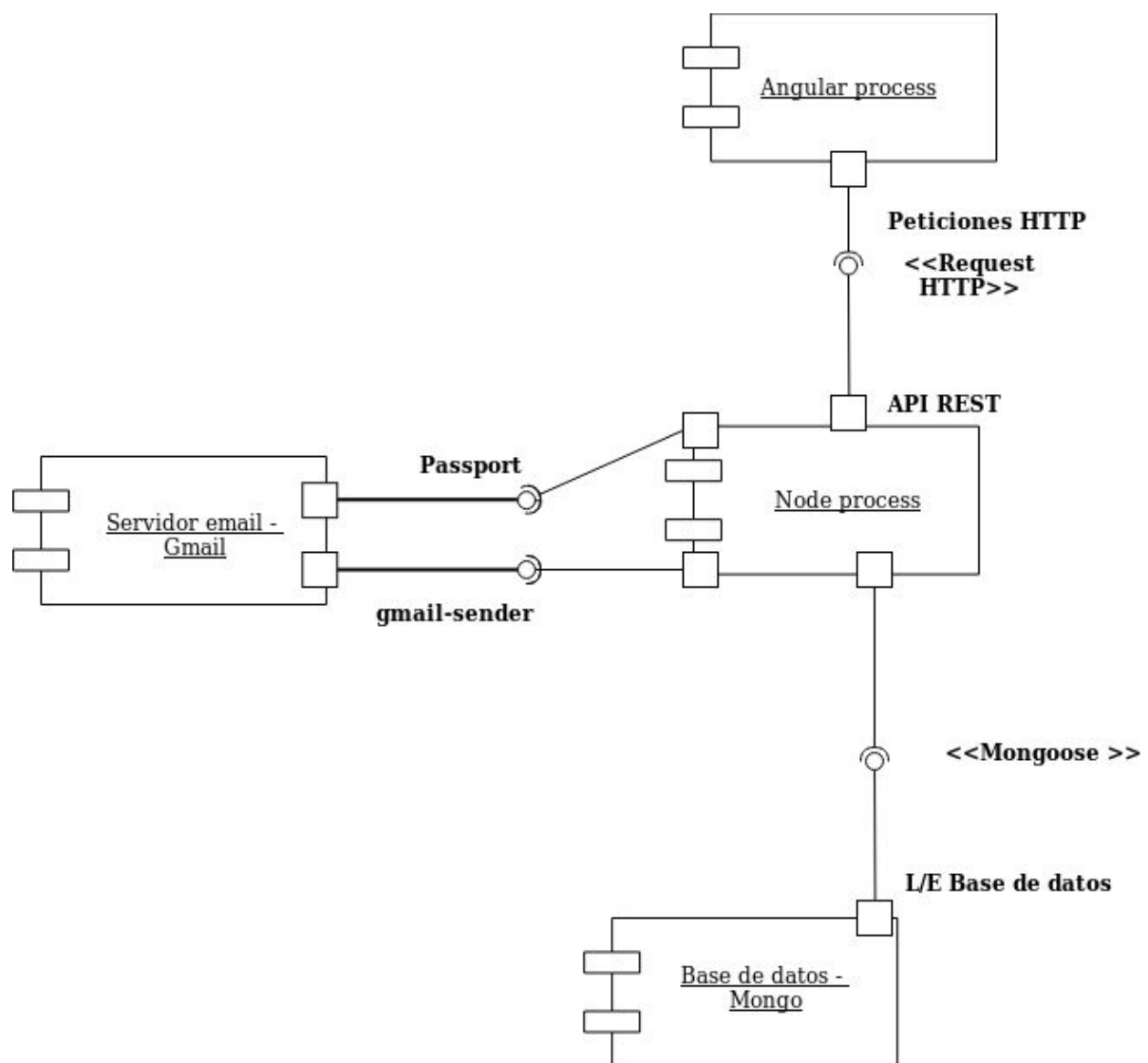
Se buscó información sobre cualquier tipo de aplicación similar para poder realizar una comparativa sobre posibles implementaciones que mejorarían dicha versión, pero no hemos encontrado una aplicación que permita la planificación completa de una visita o unas vacaciones en la comunidad de Aragón, desde el alojamiento hasta las diferentes actividades y sitios de interés de la zona. De esta forma, se considera que la aplicación puede tener éxito ya que rellenará un hueco que actualmente se encuentra libre en la temática del turismo en nuestra comunidad autónoma. Una página web que presentaba una temática parecida que se encontró fue [www.turismodearagon.com](http://www.turismodearagon.com), aunque su temática es más bien informativa, y presenta una amenaza competitiva para nuestra aplicación. Se conocen también algunas aplicaciones orientadas al turismo de Aragón, como por ejemplo una desarrollada por la empresa Quelinka (<https://www.quelinka.com/app-turismo-de-aragon/>) pero dado que están orientadas de forma total a la oferta de esquí tampoco se consideran posibles alternativas viables al proyecto que se ha desarrollado.

### 3. ARQUITECTURA

#### Vista de módulos



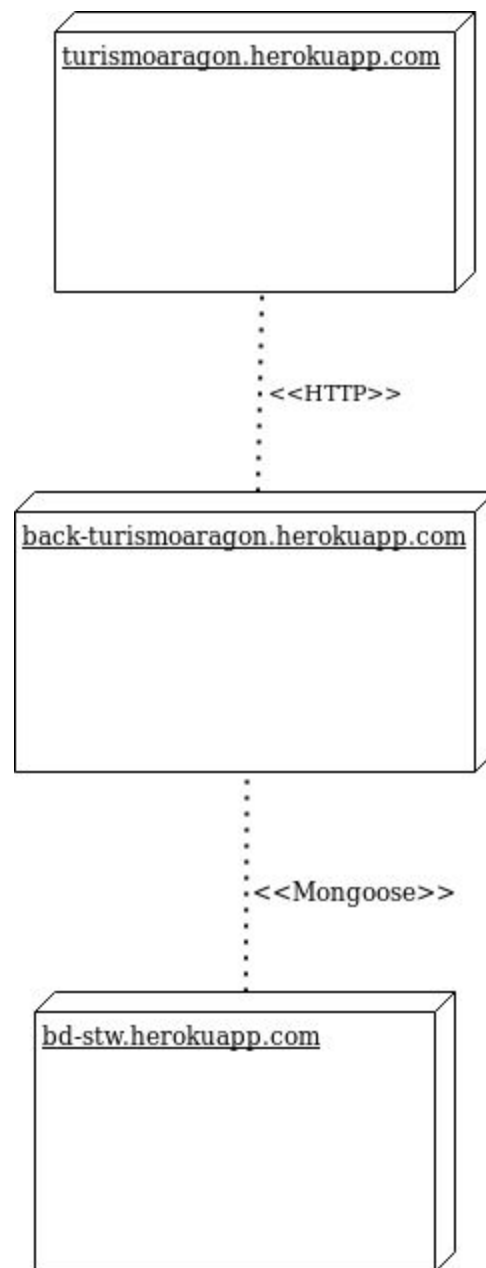
## Vista de componente y conector





---

## Vista de despliegue



---

## 4. MODELO DE DATOS

Se han creado un total de quince esquemas para poder modelar los datos necesarios para la aplicación. Los ficheros creados para los modelos residen en la carpeta models.

El esquema creado para la información de los usuarios, contenido en el fichero usuario.js, contiene trece campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
nombre	String	si	Nombre del usuario
apellidos	String	si	Apellidos del usuario
email	String	si	Email con el que el usuario se ha registrado y accede a la aplicación
fechaNacimiento	Date	no	Fecha de nacimiento del usuario
contrasena	String	no	Contraseña con la que el usuario accede a la aplicación. en caso de que se haya registrado a través de la propia aplicación
telefono	String	no	Teléfono del usuario
país	String	no	País del usuario
provincia	String	no	Provincia del usuario
activo	Boolean	no	Indica si la cuenta está habilitada o no
finBan	Date	no	Fecha a partir de la cual el usuario podrá acceder de nuevo a su cuenta en caso de ser baneado temporalmente
admin	Boolean	no	Indica si el usuario es un usuario normal o administrador

---

Como se puede observar, hay algunos campos importantes respecto a la información del usuario que no están marcados como requeridos. En el apartado Problemas encontrados durante el desarrollo se explica a que es debido (ver [Problemas con el registro y autenticación de los usuarios](#)).

El esquema creado para la información de los mensajes de los chat, contenido en el fichero mensaje.js, contiene tres campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
emisor	String	no	Emisor del mensaje
texto	String	no	Contenido del mensaje
hora	Date	no	Hora en la que el mensaje ha sido enviado

El esquema creado para la información de los chat, contenido en el fichero chat.js, contiene cinco campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
nomEntrada	String	no	Nombre del alojamiento
nomUsuario	String	no	Nombre del usuario
emailEntrada	String	no	Email del alojamiento
emailUsuario	String	no	Email del usuario
mensajes	Array de esquema de mensajes	no	Mensajes del chat

Para las entradas que son alojamientos, es decir, refugios, alojamientos de turismo rural, hoteles, campings y apartamentos, se ha creado un esquema con los datos que comparten, el cual luego es utilizado por cada uno de ellos en su esquema específico.

Este esquema contiene la información común a todos los alojamientos, contenido en alojamiento.js, contiene diez campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
signatura	String	no	Identificador utilizado por el Gobierno de Aragón
nombre	String	no	Nombre del alojamiento
dirección	String	no	Dirección del alojamiento
codigoPostal	Number	no	Código postal del alojamiento
provincia	String	no	Provincia donde se encuentra el alojamiento
comarca	String	no	Comarca donde se encuentra el alojamiento
municipio	String	no	Municipio donde se encuentra el alojamiento
capacidad	Number	no	Capacidad del alojamiento
email	String	no	Email de contacto del alojamiento
teléfono	String	no	Teléfono del alojamiento

Al igual que ocurre en el esquema de usuario, hay campos que no son requeridos, pese a que son importantes, también ocurre en los esquemas del resto de entradas. En el apartado Problemas encontrados durante el desarrollo se explica a que es debido (ver [Problemas a la hora de importar datos](#)).

El esquema creado para la información de los alojamientos de turismo rural, contenido en el fichero alojamientoTurismoRural.js, contiene tres campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
común	Esquema de alojamiento	no	Datos comunes de todos los alojamientos
espigas	Number	no	Categoría del alojamiento de turismo rural

tipo	String	no	Tipo de alojamiento de turismo rural (casa, apartamento, ...)
------	--------	----	---

El esquema creado para la información de los apartamentos, contenido en el fichero apartamento.js, contiene un campo, el cual se describen a continuación:

Nombre	Tipo	Requerido	Descripción
común	Esquema de alojamiento	no	Datos comunes de todos los alojamientos

El esquema creado para la información de los campings, contenido en el fichero apartamento.js, contiene un campo, el cual se describen a continuación:

Nombre	Tipo	Requerido	Descripción
común	Esquema de alojamiento	no	Datos comunes de todos los alojamientos

El esquema creado para la información de los hoteles, contenido en el fichero hotel.js, contiene tres campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
común	Esquema de alojamiento	no	Datos comunes de todos los alojamientos
grupo	String	no	Empresa a la que pertenece el hotel
estrellas	Number	no	Categoría del hotel

El esquema creado para la información de los refugios, contenido en el fichero apartamento.js, contiene un campo, el cual se describen a continuación:

---

Nombre	Tipo	Requerido	Descripción
común	Esquema de alojamiento	no	Datos comunes de todos los alojamientos

El esquema creado para la información de las oficinas de turismo, contenido en el fichero oficinaTurismo.js, contiene siete campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
signatura	String	no	Identificador utilizado por el Gobierno de Aragón
nombre	String	no	Nombre de la oficina de turismo
dirección	String	no	Dirección de la oficina de turismo
provincia	String	no	Provincia donde se encuentra la oficina de turismo
municipio	String	no	Municipio donde se encuentra la oficina de turismo
teléfono	String	no	Teléfono de la oficina de turismo
horario	String	no	Horario de la oficina de turismo

El esquema creado para la información de los puntos de información, contenido en el fichero puntoInformacion.js, contiene cinco campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
signatura	String	no	Identificador utilizado por el Gobierno de Aragón
nombre	String	no	Nombre del punto de información

dirección	String	no	Dirección donde se encuentra el punto de información
provincia	String	no	Provincia donde se encuentra el punto de información
municipio	String	no	Municipio donde se encuentra el punto de información

El esquema creado para la información de los restaurantes, contenido en el fichero restaurante.js, contiene diez campos, los cuales se describen a continuación:

Nombre	Tipo	Requerido	Descripción
signatura	String	no	Identificador utilizado por el Gobierno de Aragón
nombre	String	no	Nombre del restaurante
dirección	String	no	Dirección donde se encuentra el restaurante
codigoPostal	Number	no	Código postal del restaurante
provincia	String	no	Provincia donde se encuentra el restaurante
comarca	String	no	Comarca donde se encuentra el restaurante
municipio	String	no	Municipio donde se encuentra el restaurante
capacidad	Number	no	Capacidad del restaurante
teléfono	String	no	Teléfono del restaurante
categoría	Number	no	Categoría del restaurante

El esquema creado para la información de los guías, contenido en el fichero guia.js, contiene diez campos, los cuales se describen a continuación:

---

Nombre	Tipo	Requerido	Descripción
signatura	String	no	Identificador utilizado por el Gobierno de Aragón
nombre	String	no	Nombre del guía
apellidos	String	no	Apellidos del guía
teléfono	String	no	Teléfono del guía
español	String	no	El guía puede hablar español
inglés	String	no	El guía puede hablar inglés
francés	String	no	El guía puede hablar francés
alemán	String	no	El guía puede hablar alemán
italiano	String	no	El guía puede hablar italiano
Otros	String	no	El guía puede hablar otros idiomas diferentes a los anteriores

El esquema creado para los datos que conforman las estadísticas, contenido en el fichero dato.js, contiene los siguientes campos, sigue un modelo clave-valor:

Nombre	Tipo	Requerido	Descripción
nombre	String	no	Nombre para identificar al dato.
valor	Number	no	Valor numérico del dato



---

El esquema creado para las estadísticas, contenido en el fichero estadistica.js, contiene los siguientes campos:

Nombre	Tipo	Requerido	Descripción
nombre	String	no	Nombre para identificar a la estadística.
valor	Array de esquema de datos	no	Array con los datos que conforman la estadística.

---

## 5. API REST

Una vez instalado swagger mediante `npm install`, en el fichero `app.js` se han llevado a cabo los imports necesarios, tal y como se indica en la página de npm. Se asigna a una variable la ruta del documento donde se encuentra la documentación de la API y con `app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument))`; se indica la ruta donde estará accesible la documentación desde el navegador.

Se ha utilizado un único fichero json para documentar la API, en primer lugar se han establecido algunos parámetros de información, como la versión o el título. Después se ha establecido el host, así como la base path y un tag para cada controlador para poder agrupar las operaciones disponibles por este criterio. Se ha indicado que el esquema utilizado es http y que consume y produce ficheros json. Una vez realizado esto, se documenta cada operación, indicando el tipo de operación que es, asignándole un tag para indicar a que controlador pertenece, en caso de necesitar parámetros, se indica el nombre, si se ubica en el path, la descripción y el esquema que usan. Por último se indica las respuestas que proporciona la operación, con el código, la descripción y el esquema. En la parte final del fichero se documentan los esquemas utilizados, con sus propiedades.

La documentación de la API se encuentra accesible a través de la siguiente URL, <http://back-turismoaragon.herokuapp.com/api-docs/>.

---

## 6. IMPLEMENTACIÓN

A continuación, se van a resumir algunos aspectos que se consideran de importancia en la implementación tanto del Front-End como del Back-End.

### Front-End

Para el desarrollo del front-end se ha utilizado el framework Angular como hay que hacer a la hora de aplicar el stack MEAN en un proyecto.

Se pueden dividir los componentes que se han creado para esta aplicación en 3 tipos:

- **Modelos:** El modelo de datos se utiliza para representar las entidades necesarias en la aplicación. Estas entidades son enviadas desde el back-end respondiendo a una petición realizada por un usuario a través del front-end. El modelo de datos utilizado en el front-end es igual al utilizado en el back-end ya que se ha considerado que todos los datos almacenados sobre las entidades son necesarios de cierta manera (ya sea para mostrarlos u operar con ellos) en el front-end.
- **Páginas:** Cada página que se muestra en la aplicación es un componente de Angular. Es importante destacar que se ha utilizado Bootstrap para la implementación de los componentes en las diversas páginas así como para organizarlas. La navegación entre páginas se realiza mediante el uso del módulo Route disponible en el propio Angular. Se han definido algunos campos en hojas de estilo (css) generales para que, por ejemplo, el estilo del fondo de las páginas o de los botones sea igual en todas las páginas sin tener que repetir código. También se ha utilizado la biblioteca jQuery para la interacción con algunos de los componentes integrados en los HTML de las páginas.

Se utilizaron las librerías Geocoding API y Maps JavaScript API en el componente entry. El cual está destinado a mostrar toda la información de una de las entradas que aparecen como resultado de una búsqueda de un usuario. Para poder hacer uso de ellas fue necesario realizar un largo proceso en el cual primero se creó una cuenta en Google Cloud Platform. En la creación de dicha cuenta, fue necesario aportar la tarjeta de crédito de uno de los componentes del equipo para así recibir acceso a una prueba gratuita con una duración de 1 año y 300\$ de créditos promocionales. Una vez creada dicha cuenta, se pudieron activar dichos servicios y haciendo uso de una API-Key suministrada se pudo hacer uso de estas librerías. La Maps API se utilizó para poder mostrar mapas en las entradas. A esta se le unió la Geocoding API la cual permite introducir la dirección de una calle en concreto y a partir de esta sacar las coordenadas de latitud y longitud de la misma. Todo ello junto, hizo posible que al introducir la dirección de uno de los lugares almacenados en nuestra base de datos, se muestre su ubicación exacta en un mapa de

---

Google Maps. En el componente profile y en las pantallas de registro se puede elegir entre distintos países de la unión europea y sus respectivas provincias. Esta funcionalidad se implementa utilizando la librería npm: country-state-city. La cual realiza una búsqueda en una base de datos con todas las ciudades, provincias y países.

- **Servicios:** Los servicios son componentes dedicados a la interacción del front-end con la API Rest implementada por el back-end. Estos servicios son:
  - **User Service:** Este componente contiene todas las funciones encargadas de realizar las llamadas a la API referidas a los usuarios. Tanto usuarios normales como el administrador utilizan este servicio. Las funciones de los usuarios son “Log in”, “Log in con google”, “Registrar” y “Actualizar el usuario”; El administrador utiliza este servicio a través de funciones como “Buscar usuarios” o “Actualizar usuarios”. Este servicio también se encarga de las llamadas para enviar correos electrónicos. Además, para la autorización de los usuarios en el back-end se utiliza el módulo JSON Web Token por lo que este servicio también consta con una llamada al back-end llamada “recover” donde, pasando como parámetro el token recibido durante la autenticación, se recupera la entidad del usuario que tiene la sesión iniciada en el dispositivo.
  - **Current User Service:** Este componente está compuesto de los servicios que hacen de pasarela entre algunas llamadas de “User Service” para separar cierta lógica de la implementación y para que algunas acciones sean más accesibles.

Sin embargo, la función más importante que realiza este componente es mantener activa la sesión que un usuario ha abierto. Para ello a través de la función “checkLog” utiliza el servicio de las cookies de los navegadores guardando una variable booleana que indica si hay una sesión abierta o no y almacenando el token recibido del back-end. De esta manera, aunque se pierda la entidad usuario al recargarse una página, este componente utiliza la llamada a la API “recover” pasando como parámetro el token almacenado en la cookie para verificar y recibir la identidad del usuario. Una vez se ha recibido, la página que ha sido recargada y ha solicitado conocer el usuario lo recibe utilizando el patrón observador a través de una variable “observer”. Esta implementación también deniega la entrada de los usuarios a cualquier página que requiera haber iniciado sesión y todavía no se haya autenticado (redirige a la pantalla de “login”).

- 
- Entry Service: En este componente se encuentran todos los servicios relativos a las entradas. Los usuarios los utilizan cuando realizan una búsqueda de cualquier tipo de entrada (tanto alojamientos como guías, puntos de información turística...). En este fichero también se encuentran las llamadas al endpoint “statistics” de la API para mostrar las estadísticas relativas de las entradas tanto a los usuarios como a los administradores. También incluye las llamadas a la API para descargar los datos de una búsqueda en CSV o PDF y por último cubre la interacción con el endpoint “parser” que posibilita al administrador actualizar las entradas de la base de datos cuando el quiera con los últimos cambios añadidos directamente en la fuente de datos utilizada (Open Data Aragón).
  - Chat Service: En este componente se incluye toda la interacción entre el front-end y la API Rest relativa a los módulos de los chats, tanto para los usuarios comunes como para los gerentes de las entradas.

### **Back-End :**

En primer lugar se han implementado los distintos modelos necesarios para gestionar los datos necesarios para el funcionamiento de la aplicación. El aspecto a destacar más relevante de la implementación de esta parte sería la “jerarquía” creada para las entradas de tipo alojamiento, ya que tanto los hoteles, refugios, alojamientos de turismo rural, apartamentos y campings tienen una serie de datos en común. Estos serían, signatura, nombre, dirección, código postal, provincia, comarca, municipio, capacidad, email y teléfono. Por ello, se ha creado un modelo que todos utilizan, llamado alojamiento, el cual es el parámetro común que todos tienen. La idea detrás de esto es evitar repetir los mismos campos en todos estos esquemas.

Pasando a los controladores de las entradas, implementan el mismo tipo de operaciones, obtener un listado de 20, indicando que 20 del listado total como parámetro. Se realiza de esta forma para facilitar mostrarlos en la web, ya que al usuario se le muestra un listado de 20 por página. Para conseguir esto, se utilizan las operaciones skip y limit junto con find de mongoose. Otra operación que se implementa es contar el número total, utilizando la operación count de mongoose, obtener una entrada en concreto mediante su identificador, este se obtiene como parámetro en el path y se utiliza en la operación findById. Por último, se implementa una operación de búsqueda, en la cual el resultado se página de la misma forma que en la primera operación descrita, en este caso, la operación find de mongoose se utiliza con expresiones regulares, ya que se puede dar el caso de que alguno de los parámetros de búsqueda estuviese vacío, y en ese caso cualquier valor para dicho parámetro es válido.

Respecto al controlador de las operaciones del usuario, cuenta con las mismas que las descritas anteriormente, además de obtener los datos del usuario a partir del token de sesión, para obtener el

---

email del usuario, necesario para la operación find, se utiliza la operación decodetoken de jwtinterface. Otra operación nueva para este controlador es contar el número de usuarios por email, se utiliza la operación count aplicando un filtro, en este caso el email del usuario. La operación utilizada para guardar un nuevo usuario recibe en el cuerpo de la petición los datos de este y utilizando la operación save de mongoose junto con el esquema se guarda en la base de datos. Por último, la operación utilizada para actualizar los datos del usuario recibe los nuevos datos, al igual que la operación anterior, en el cuerpo de la petición y el método seguido es similar salvo que en esta ocasión se utiliza la operación findOneAndUpdate.

El controlador de las operaciones del chat, consta de operaciones como crear un nuevo chat, recibiendo los datos en el cuerpo de la petición y utiliza la operación save y el esquema creado para guardarlos en la base de datos. Actualizar un chat con un mensaje nuevo, por parte del usuario o por parte del hotel. Se recibe los nuevos datos del chat en el cuerpo de la petición y se actualiza utilizando la operación findOneAndUpdate, junto con el id del chat. Estas operaciones descritas anteriormente hacen uso del módulo npm gmail-send para notificar por email que hay nuevos mensajes en un chat. En primer lugar se define la cuenta desde la que se va a enviar el correo, seguido del destinatario, el asunto y el cuerpo, acto seguido se utiliza la operación send que proporciona el módulo, junto con los datos mencionados, para enviar el email. Por último, se cuenta con una operación para obtener todos los chat de un usuario, recibiendo como parámetro el email del usuario, el cual se usa como filtro para la operación find, y otra operación para obtener un chat dado su identificador, este se recibe como parámetro y se utiliza en la operación findById.

Pasando al controlador para generación de archivos, cuenta con dos operaciones, una para generar un archivo csv a partir de un json y otra para generar un archivo pdf también a partir de un json. La primera de ellas, hace uso del módulo npm jsonexport para generar el csv y recibe el json como parámetro. La segunda hace uso del módulo pdf-creator-node, es necesario crear una plantilla html a partir de la cual se construirá el pdf junto con el json recibido como parámetro en la operación create proporcionada por el módulo.

Según se estableció en los requisitos, los datos externos deberían importarse mediante procesos en segundo plano, pero debido a [problemas encontrados](#), se ha optado por crear un controlador y que cuando se solicite desde la web se importen nuevos datos. Cada tipo de entrada tiene su propia operación y en todas se sigue el mismo proceso. En primer lugar se borran los datos guardados en la base de datos para evitar conflictos con los nuevos, una vez hecho esto, se realiza una petición a la URL donde se puede obtener el json, aunque no es un json al uso, con los nuevos datos. Por último, es necesario tratar estos datos, ya que no se encuentran como se utilizan en el sistema y finalmente, una vez tratados, se guardan.

Al igual que en la situación anterior, para calcular las estadísticas también se iba a realizar mediante procesos en segundo plano, pero finalmente se ha tomado una nueva decisión de diseño y se ha creado un controlador con una operación para calcular cada estadística, alegando que así tanto los usuarios como el administrador podrán verlas actualizadas en todo momento y no solo una vez al día.

---

Para comprobar que esta solución era viable, se ha examinado el tiempo de respuesta para cada operación y se ha considerado que era aceptable, dado que en un principio se pensaba que debido a los cálculos podría ser un proceso un poco lento.

Respecto a los usuarios baneados, se ha utilizado el módulo node-schedule, también se ha intentado utilizar para la importación de datos, para crear un proceso en segundo plano que compruebe, una vez al día, si la fecha actual concuerda con la fecha en la que el usuario puede recuperar su cuenta, en caso afirmativo actualiza los datos del usuario. Estas operaciones se encuentran dentro de un `scheduleJob` proporcionado por el módulo, además ha sido necesario indicar en el fichero de configuración que se ejecute la función que contiene el `scheduleJob` mediante `app.listen`.

Para el registro mediante la cuenta de Google, se ha usado el módulo passport, indicando en el router a qué página se debe redirigir al usuario en caso de que ya esté registrado o no. En caso de que no lo esté se le redirige a una página para que complete los campos que no se puede obtener a través de la cuenta de Google, como el país, fecha de nacimiento, etc.

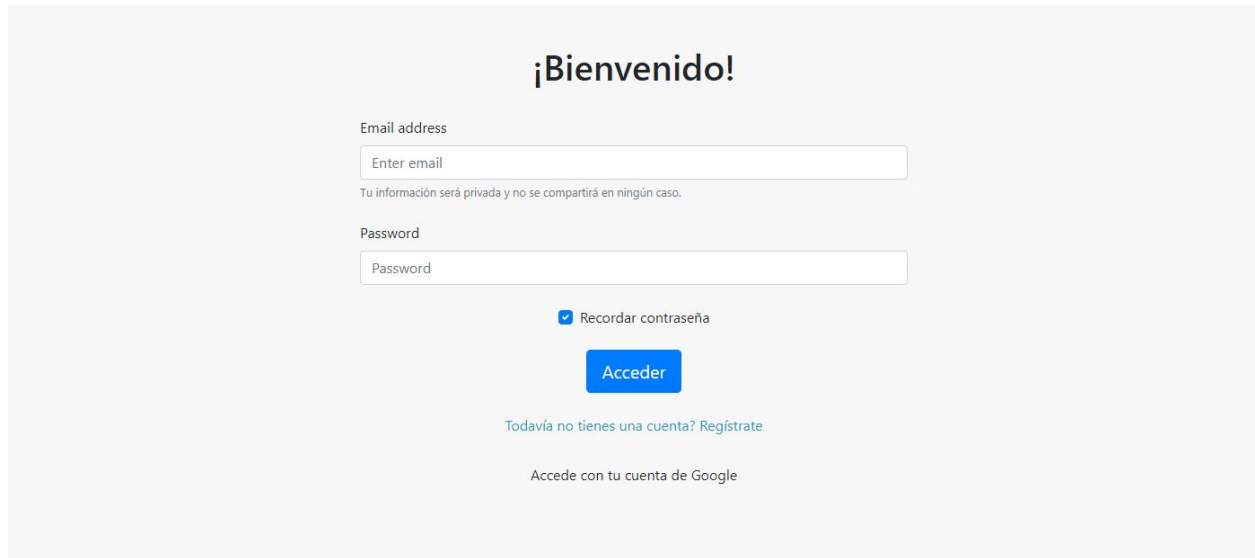
En cada operación de todos los controladores, a excepción de la operación utilizada para actualizar un chat con un mensaje nuevo por parte de un hotel, se comprueba el token de sesión mediante la función `checkToken` definida previamente. Esta función recibe como parámetro el token y mediante la operación `verifyToken` de `jsonwebtoken` comprueba su validez. En el fichero de configuración se han añadido las sentencias necesarias para configurar un registro interno con Morgan, se ha utilizado la opción `dev` para obtener una salida concisa, coloreada por estado de respuesta.

---

## 7. MODELO DE NAVEGACIÓN

En esta sección se mostrarán las pantallas del cliente de la aplicación, explicando el funcionamiento de cada una de ellas, su utilidad y la forma de navegar entre ellas. La primera de ellas será el Login.

### Pantalla Login



The image shows a login screen with a light gray background. At the top center, the text '¡Bienvenido!' is displayed in a bold, black, sans-serif font. Below this, there are two input fields. The first is labeled 'Email address' and contains the placeholder text 'Enter email'. Below the email field, a small line of text reads 'Tu información será privada y no se compartirá en ningún caso.' The second input field is labeled 'Password' and contains the placeholder text 'Password'. Below the password field, there is a checkbox with a blue checkmark and the text 'Recordar contraseña'. Centered below the checkbox is a blue button with the white text 'Acceder'. Below the button, there is a link that reads 'Todavía no tienes una cuenta? Regístrate' in a teal color. At the bottom, there is a text link that reads 'Accede con tu cuenta de Google'.

Esta es la primera pantalla que aparece al acceder a la aplicación. Desde ella los usuarios accederán a su sección personal de la aplicación, y en caso de que no tengan una cuenta podrán registrarse haciendo click en el enlace que pregunta si aún no posees una cuenta. Además se ha implementado la posibilidad de que los usuarios accedan con su cuenta de Google. Esta vista se comunica con `users.js`, el cual contiene las distintas rutas de las operaciones de los usuarios, y mediante el cual se accede a `UserController.js`. Por ejemplo, a la hora de iniciar sesión se emplea `/login`.



## Pantalla Registry

### Regístrate en Descubre Aragón

Nombre	Apellidos
<input type="text"/>	<input type="text"/>
Fecha de nacimiento	Teléfono
<input type="text"/>	<input type="text"/>
País	Provincia
<input type="text"/>	<input type="text"/>
Correo Electrónico	Tu información será privada y no se compartirá en ningún caso.
<input type="text" value="Introduzca su e-mail"/>	
Contraseña	Confirmar contraseña
<input type="password"/>	<input type="password"/>

Confirmar Registro

En esta sección los usuarios podrán registrarse en la aplicación introduciendo los datos solicitados. Y una vez se complete el registro (en caso de que todos los datos se introduzcan correctamente, se redirigirá al nuevo usuario a su sección personal de la aplicación, al igual que ocurre cuando este accede a la aplicación desde el login del sistema. Para realizar el registro la vista se comunica con el back-end y emplea `UserController.addUser`.

## Completa tu registro en Descubre Aragón

Faltan los siguientes campos de añadir para completar su registro:

Fecha de nacimiento

Teléfono

País

Provincia

Confirmar Registro

En el caso de que el acceso a la aplicación se realice mediante un login con una cuenta de gmail del usuario, se pedirán algunos datos extra que faltan, para así completar con éxito el registro del usuario. Para ello se ha implementado una pantalla alternativa de registro, en la cual se dispone tan sólo de 4 campos que rellenar.

### Pantalla Index-User

The screenshot shows the 'Index-User' screen of the application. At the top, there is a header bar with a user profile icon and name 'Nombre Apellidos España, Zaragoza' on the left, a home icon and 'Inicio' in the center, and a search icon and 'Buscar' on the right. Below the header, the screen is divided into three main sections. On the left is a vertical sidebar with five buttons: 'Inicio' (highlighted in blue), 'Perfil', 'Conversaciones', 'Estadísticas', and 'Salir' (highlighted in red). The central section contains a search bar with 'Hoteles' entered, a dropdown menu showing '[1]', and a list of results. On the right is a filter panel with several dropdown menus: 'Provincia:' (set to 'Todas'), 'Comarca:' (set to 'Todas'), 'Municipio:' (set to 'Todas'), 'Estrellas' (with 'Mínimo' set to '1 (Una)' and 'Máximo' set to '5 (Cinco)'), and 'Idioma (Guías turísticas):' (set to 'Todos'). At the bottom of the filter panel is a blue 'Buscar' button.

Esta es la pantalla a la que se accede al hacer tanto login en la aplicación como registrarse en ella. Como se puede apreciar está compuesta por tres secciones principales. A la izquierda se dispone de una serie de botones con los que poder navegar entre las distintas secciones de la aplicación. El botón que se encuentra resaltado en azul indica en todo momento la sección en la que uno está situado. Los usuarios disponen de una pantalla principal, un perfil en el que pueden realizar cambios, una sección de conversaciones para hablar con los propietarios de los alojamientos que visiten y una sección de estadísticas. En la columna de enmedio se mostrarán los resultados de las búsquedas que se realicen.

Las búsquedas se realizan en la sección de más a la derecha de la pantalla, donde se puede observar los distintos filtros disponibles (provincia, comarca, municipio, estrellas del lugar e idioma de los guías turísticos disponibles). Cabe destacar que en todas las pantalla se dispone del menú de la parte superior, que indica la ventana en la que uno se encuentra actualmente, además de una breve descripción con algunos datos del usuario en la esquina superior izquierda. Por último en cuanto a las llamadas que se realizan al back-end, desde esta pantalla se utilizan todos los controladores de cada uno de los tipos de lugar. Estas son: /getAll, /count, /get y /search. Además también se utilizan las operaciones de /media para descargar los datos en formato csv y pdf.

### Pantalla Profile

Esta es la sección del perfil de cada usuario, donde pueden realizar algunos cambios a su información personal mediante una llamada a la operación `UserController.updateUser` del Back-End. Estos cambios serían el país al que pertenece, su provincia correspondiente y su correo electrónico. La búsqueda de los países y sus provincias se ha implementado mediante el uso de una librería externa que se encontró en npm: `country-state-city`

Mediante el uso de dicha librería se pudo implementar la posibilidad de elección de los países de la unión europea junto con sus provincias correspondientes. Además, se cuenta con la posibilidad de cambiar la contraseña pulsando en el botón “Cambiar contraseña” mediante el cual se accede a una pantalla diseñada a tal efecto. Se muestra a continuación una captura de esta última (su estructura es similar a la de “profile” pero cambiando los contenidos de la columna central).

## Pantalla Change-Password

Nombre Apellidos  
España, Zaragoza

### Cambiar Contraseña

Bienvenido, Nombre Apellidos

Introduzca su contraseña actual

Contraseña actual.

Introduzca su nueva contraseña

Nueva contraseña.

Confirme su nueva contraseña

Repetir la contraseña nueva.

Guardar cambios

Del mismo modo que en la anterior, en esta pantalla se emplea la operación del controlador `UserController.updateUser` para actualizar la contraseña del usuario. Para ello el usuario debe introducir la contraseña actual, la nueva contraseña y de nuevo la nueva para confirmar con éxito.

## Pantalla Chat-User

Nombre Apellidos  
España, Zaragoza

### Conversaciones

Entrada 14:28

Tu: "Último mensaje enviado"

ZARAGOZA CENTRO 14:28

Tu: "Último mensaje enviado"

Enviar

Esta es la ventana que aparece cuando los usuarios pulsan el botón de conversaciones. Desplegándose en la columna central las distintas conversaciones activas que tienen. Además cuando se selecciona una de ellas se abre un chat en la tercera columna con la conversación seleccionada. Desde esta pantalla se utiliza el controlador `chatController.js` al cual se realizan llamadas para hacer

uso de las operaciones `/updateChatEntry` y `/getChat`. Además dentro de una conversación seleccionada se utilizan el resto de las operaciones de dicho controlador.

### Pantalla Index-Admin

The screenshot displays the 'Index-Admin' interface. At the top, there is a header bar with a user profile icon labeled 'Administrador', a home icon labeled 'Inicio', and a search icon labeled 'Buscar'. Below the header, the main content area is divided into three sections. On the left is a sidebar with four buttons: 'Inicio' (blue), 'Estadísticas' (white with a bar chart icon), 'Datos' (white with a document icon), and 'Salir' (red). The central section contains a list of three user entries. Each entry has a header with the user's name, followed by a row containing their email address, location, and country, and a blue arrow button. The users listed are Gonzalo Berné (email: gera1397@gmail.com, location: Western, country: Malta), Eduardo Gimeno (email: eduardo.gimeno12@gmail.com, location: Zaragoza, country: España), and Saúl Alarcón (email: karmy.sac@gmail.com, location: Zaragoza, country: España). On the right side of the main content area, there is a search form with three input fields labeled 'Nombre', 'Apellidos', and 'E-mail'. Below these fields is a blue 'Buscar' button. Underneath the search button, there is a text label 'Descargar datos de la búsqueda:' followed by two blue buttons labeled 'CSV' and 'PDF'.

Esta pantalla posee de toda la funcionalidad de la que disponen los administradores de la aplicación. A la cual, los administradores acceden a través de la misma ventana de login que los usuarios. En esta ventana se realizan también llamadas al controlador `UserController.js` para recibir la información de los usuarios de la aplicación además de ser posible a su vez descargar la lista con los usuarios en formato csv o pdf.

---

## 8. ANALÍTICAS

Se han implementado tanto para los usuarios como para los administradores una serie de estadísticas que informan de forma clara de datos específicos de la página web. Estas estadísticas se diseñaron utilizando la librería `@rinminase/ng-charts`. Los usuarios disponen de las siguientes estadísticas:

**Cantidad de puntos de interés y establecimientos:** Esta estadística muestra la cantidad de cada uno de los distintos tipos de lugares de los que dispone información la web que hay respecto del total. Este diagrama deja entrever la realidad de Aragón, y es que la mayor parte de su turismo es rural. Cómo se puede apreciar existen más establecimientos de apartamentos y casas de turismo rural que hoteles.

**Cantidad de hoteles por municipio:** Muestra el número total de hoteles que hay en cada uno de los municipios de la comunidad autónoma de Aragón. Esta estadística permite conocer de forma exhaustiva los municipios donde se concentran mayores cantidades de hoteles, siendo el principal foco Zaragoza con una diferencia abismal seguido de Huesca. Por otro lado, el resto de municipios no superan la veintena de hoteles.

**Distribución de los idiomas hablados por cada guía:** Muestra el porcentaje de guías que son capaces de hablar cada uno de los idiomas representados en el diagrama. Mediante esta estadística se puede saber que por ejemplo casi el 100% de los guías son bilingües ya que hablan el inglés además del español. Y también que los turistas franceses son muy frecuentes ya que una basta cantidad de ellos ha decidido aprender francés como tercera lengua.

**Cantidad de restaurantes de 3 tenedores/tazas por comarca:** Muestra el número de restaurantes con la calidad descrita que existen en cada una de las comarcas. Mediante estadísticas de este tipo se podría estudiar donde se concentra la mayor cantidad de lugares con mejor calidad gastronómica y/o servicio.

Los administradores disponen por otro lado de estas:

**Provincias de procedencia de los usuarios:** Número de usuarios que pertenecen a cada provincia. (No necesariamente ubicada en Aragón). Se puede evaluar el éxito de la aplicación en cada lugar mediante la contabilización del número de usuarios que se registran en cada lugar.

**Distribución de las entradas por porcentajes:** Similar a la estadística de la que disponen los usuarios, en este diagrama se muestra el porcentaje del total de entradas que representa cada lugar.

**Cantidad de chats por hotel:** Número de conversaciones que se han iniciado con cada uno de los hoteles. Mínimo ha de tener una para aparecer representado en el diagrama. Con esta gráfica se podrá

---

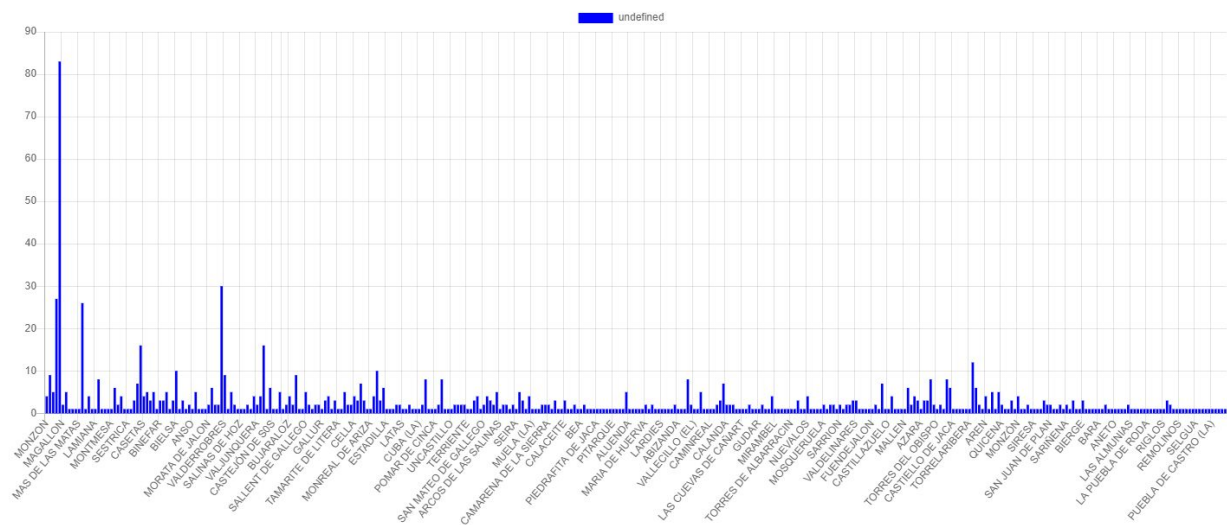
conocer el éxito y/o interés que suscitan los distintos lugares en los usuarios. Permitiendo así realizar recomendaciones personalizadas a los usuarios.

**Fechas de recuperación de las cuentas:** Representa la cantidad de usuarios que recuperan su cuenta en cada fecha, en caso de haber sido bloqueados temporalmente en la aplicación.

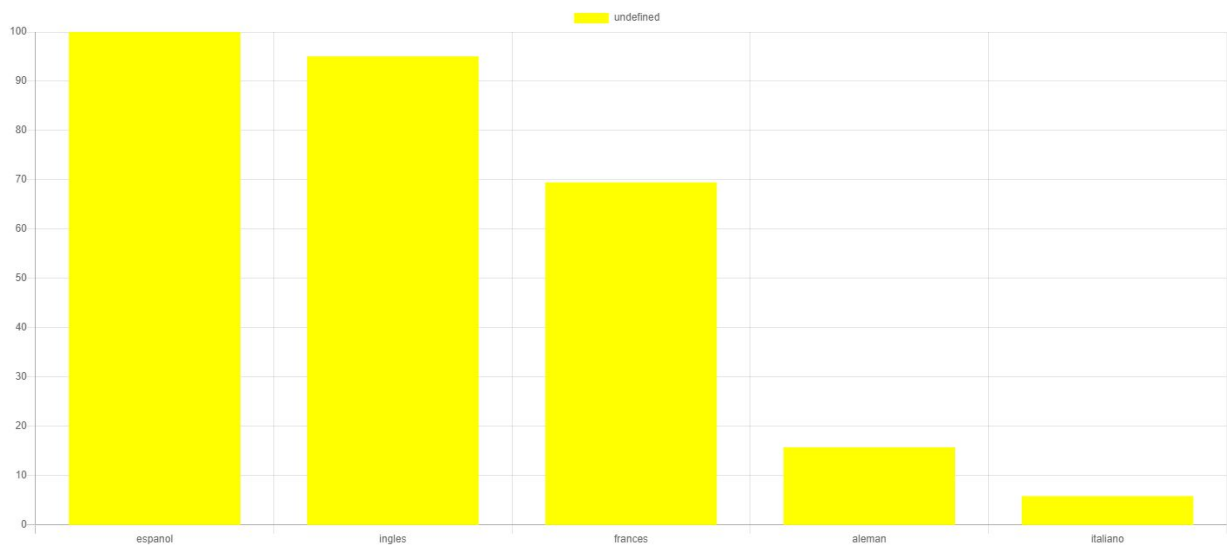
A continuación aparecen algunas imágenes de las estadísticas funcionando en la web.



## Cantidad de hoteles por municipio

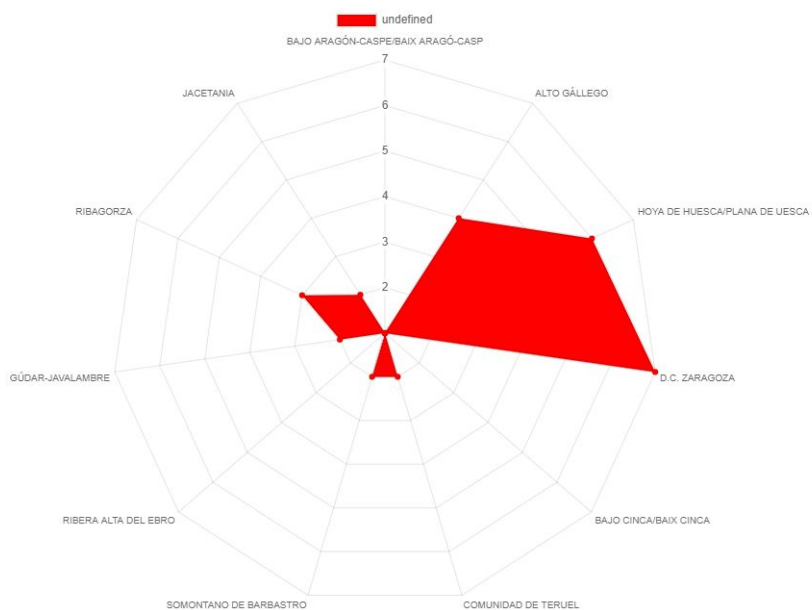


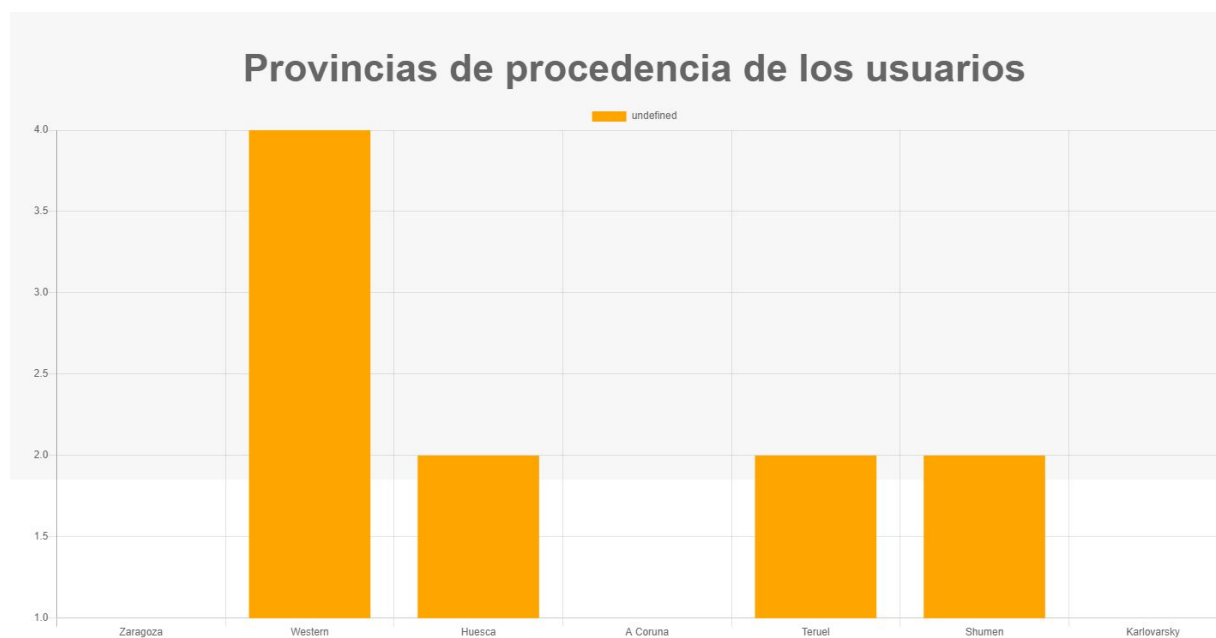
## Distribución de los idiomas hablados por cada guía



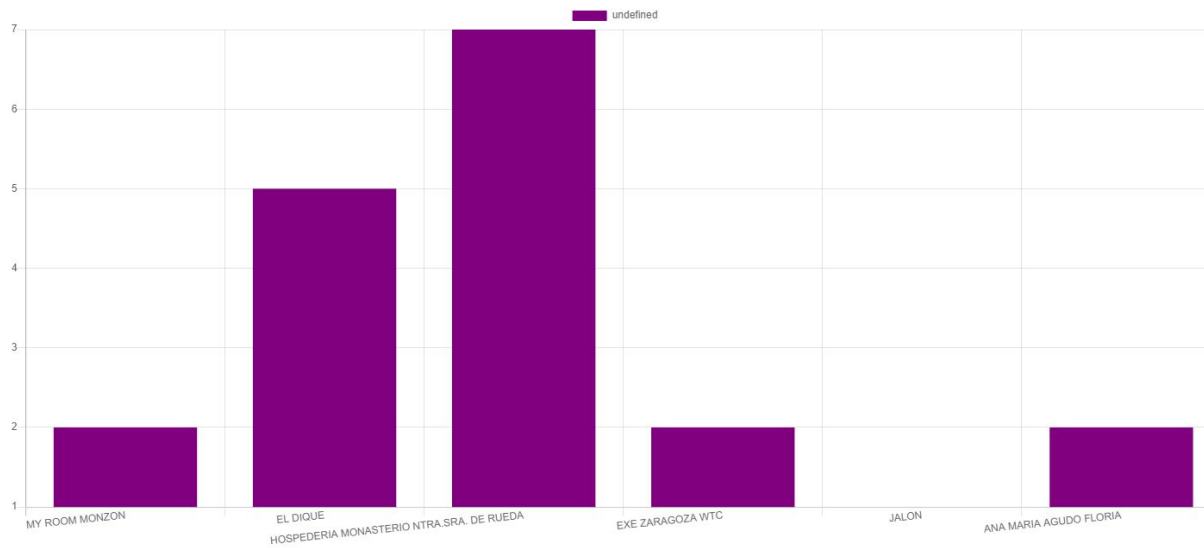


## Cantidad de restaurantes de 3 tenedores/tazas por comarca

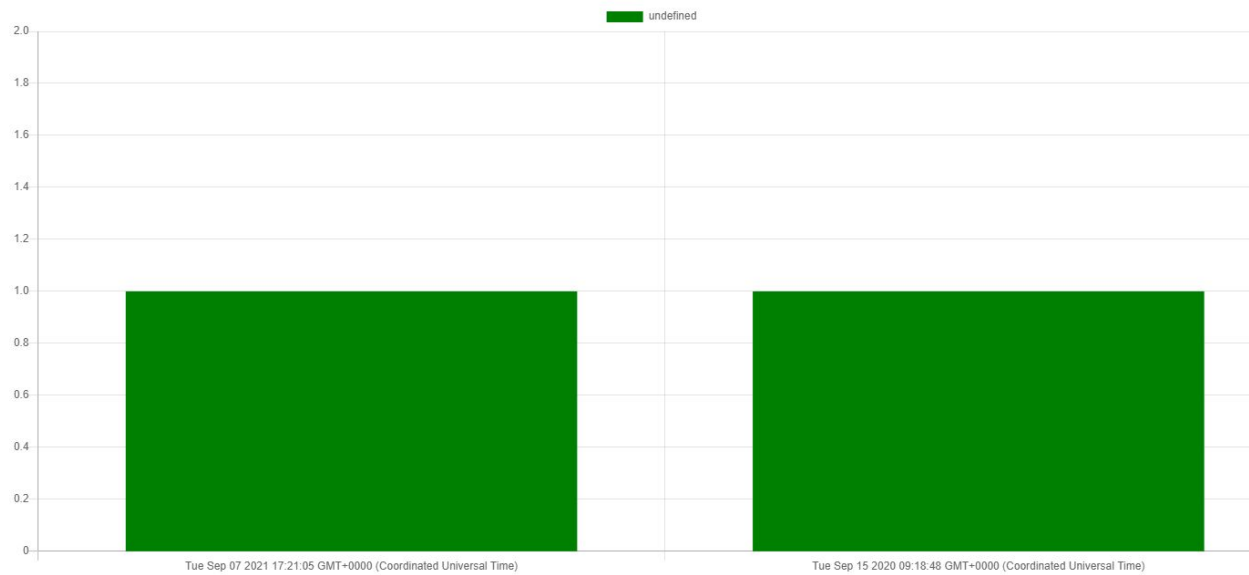




## Cantidad de chats por hotel



## Fechas de recuperación de las cuentas



## 9. DESPLIEGUE DEL SISTEMA

Para realizar el despliegue, se ha decidido realizar el despliegue en 3 máquinas diferentes, una para cada nodo necesario para nuestro sistema (aplicación, servidor web y base de datos).

En el nodo que almacena la base de datos de nuestro sistema, para poder realizar el despliegue de una base de datos MongoDB, se ha necesitado incluir en la máquina el add-on mLab-MongoDB, una plataforma ofrecida como Database-as-a-Service que permite desplegar esta base de datos y utilizarla como servicio para nuestra aplicación. Gracias a la facilidad que proporciona mongoose para realizar la conexión con la base de datos, únicamente nos es necesario conocer la URI de la misma para poder utilizarla.



The screenshot displays the 'Application Logs' interface for a deployment. The logs show a sequence of events for a release created by user 721615@unizar.es. The steps include: initial release, enabling Logplex, attaching the mLab-MongoDB add-on (ref: mongolab-parallel-99697), running release v3 commands, and finally completing provisioning and setting the MONGODB\_URI. The interface includes an 'Autoscroll with output' checkbox and a 'Save' button.

```
2020-06-10T08:38:09.536149+00:00 app[api]: Release v1 created by user 721615@unizar.es
2020-06-10T08:38:09.536149+00:00 app[api]: Initial release by user 721615@unizar.es
2020-06-10T08:38:09.666453+00:00 app[api]: Release v2 created by user 721615@unizar.es
2020-06-10T08:38:09.666453+00:00 app[api]: Enable Logplex by user 721615@unizar.es
2020-06-10T08:40:54.551416+00:00 app[api]: Attach MONGODB (@ref:mongolab-parallel-99697) by user 721615@unizar.es
2020-06-10T08:40:54.551416+00:00 app[api]: Running release v3 commands by user 721615@unizar.es
2020-06-10T08:40:54.562022+00:00 app[api]: Release v4 created by user 721615@unizar.es
2020-06-10T08:40:54.562022+00:00 app[api]: @ref:mongolab-parallel-99697 completed provisioning, setting MONGODB_URI. by user 721615@unizar.es
```

Como se puede observar, los pasos seguidos para el despliegue de este nodo del sistema fueron muy simples: primero se realizó la creación de la máquina para después incluir en la misma el add-on comentado anteriormente que proveía a la misma de una base de datos MongoDB.

Para los dos nodos restantes de nuestro sistema, se ha utilizado la misma estrategia en ambos, la cual se basa en que primero Heroku trata de encontrar un fichero Procfile y, en caso de no encontrarlo, como en este caso en el cual ese manejo se realiza mediante el fichero package.json, trata de generar un script de nombre web en función del script de start definido en el package.json. Es importante también la presencia en el package.json del dato engines, en el cual se va a especificar la versión de node que va a ser utilizada para el despliegue del repositorio en la máquina.



The screenshot shows the configuration for the 'web' process in Heroku. The command 'npm start' is entered in the command field. A toggle switch is turned on, and the cost is listed as '\$0.00'.

```
web  npm start
```

En la imagen podemos ver la existencia del proceso web, en base a lo definido en el package.json.

---

Otra función ofrecida por Heroku, es la de definición de variables de entorno, `config_var` para Heroku, que en nuestro caso se han utilizado para definir las variables relacionadas con las claves necesarias tanto para la autenticación con Passport y la integración de Google con el mismo, como para la clave secreta utilizada para la firma del JWT. De esta manera, podemos referenciar en el código fuente el uso de estas variables de la siguiente manera:

```
var clienteID = process.env.CLIENTE_ID  
  
var secretID = process.env.SECRET_ID  
  
var jwtClave = process.env.JWT_CLAVE
```

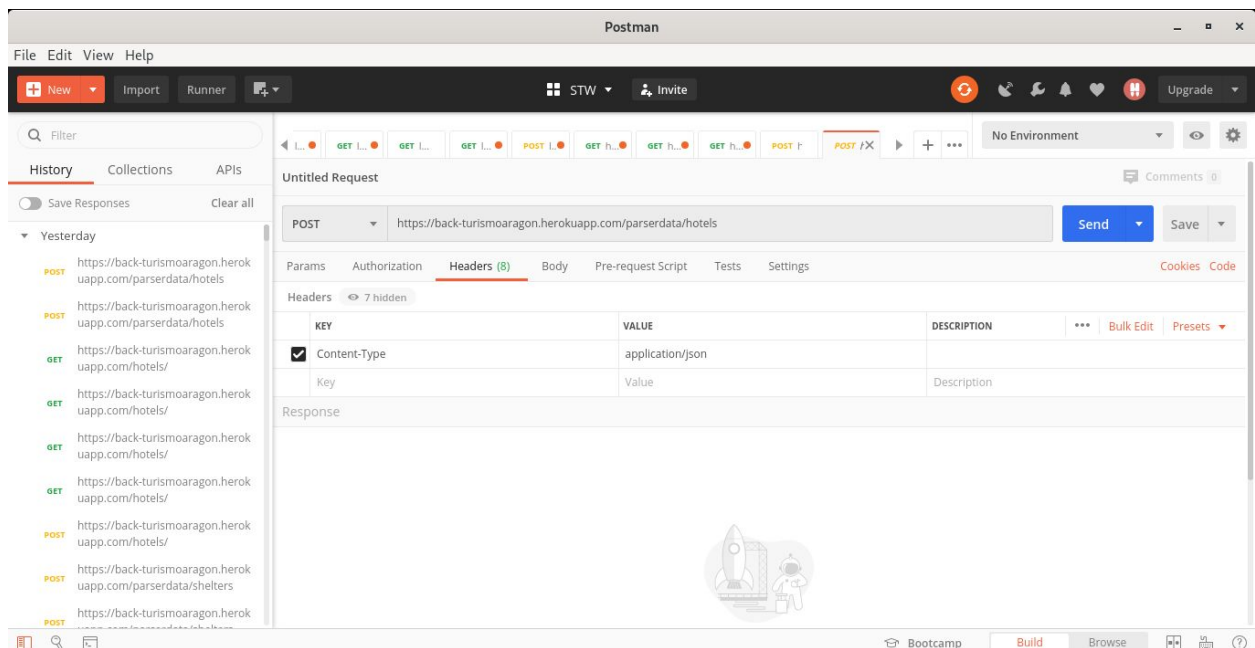
En cuanto al método utilizado para el despliegue en Heroku, se ha optado por el uso de Heroku CLI, sin embargo debido a que ya existía previamente un repositorio en el cual se almacenaban los códigos fuente, en vez de realizar el clonado del repositorio de git local generado por Heroku, se ha añadido el mismo como repositorio remoto del ya existente, de manera que al ejecutar `git push heroku master`, se realice el push al repositorio de Heroku y se realice un nuevo deployment de la aplicación.

## 10. VALIDACIÓN

A la hora de realizar la validación del front-end se ha utilizado la herramienta protractor. Esta herramienta permite automatizar la sucesión de eventos en las páginas web simulando la interacción de un usuario real con la aplicación. Protractor necesita de dos ficheros para trabajar “spec.js” y “conf.js”. En “conf.js” se encuentra la configuración donde se elige, entre otras cosas, el puerto donde se va a lanzar el simulador, los navegadores a testear y los ficheros donde se encuentran los tests (en este caso solo uno, “spec.js”).

En el fichero “spec.js” se han definido los tests a pasar por la herramienta para la validación de la aplicación. Estos tests consisten en la navegación entre las diferentes páginas de la aplicación empezando por acceder a la pantalla de login, ingresar un email y una contraseña de un usuario previamente creado y validando el acceso. Para que la herramienta cambié de pantalla correctamente, los diferentes botones con los que cuenta la aplicación están identificados con un “id” y de la misma manera cada pantalla de la aplicación muestra el título de esa página en concreto a través de la misma variable en el HTML por lo que protractor valida rápida y correctamente la pantalla en la que se encuentra. Con esta herramienta se ha validado correctamente la navegación por las diferentes pantallas de la aplicación tanto de los usuarios comunes como del administrador.

Por parte del back-end, para probar las distintas operaciones de los controladores, se ha utilizado Postman, tanto cuando se estaba desplegando en local el servidor y la base de datos, como una vez desplegados en Heroku.



---

A la hora de realizar integración entre el cliente y el servidor se ha utilizado principalmente la consola de desarrollador del navegador para ver el estado de cada petición desde la vista del cliente.

---

## 11. PROBLEMAS ENCONTRADOS DURANTE EL DESARROLLO

### Problemas con el registro y autenticación de los usuarios

En el desarrollo del registro y autenticación de usuarios mediante Passport y en concreto la posibilidad de realizar este procedimiento utilizando una cuenta de Google ha generado diversos problemas.

El primero de ellos consistió en que dado que para poder realizar esta autenticación lo primero que se debe realizar es generar una serie de credenciales para la aplicación en Google, en la creación de estas llega una parte en la cual se pregunta por cuáles van a ser las direcciones que van a poder realizar este tipo de peticiones con esas credenciales, dando a un fallo en caso de que la petición recibida no estuviera incluida dentro del rango de direcciones. Situación similar ocurría con el callback de la misma.

Por otro lado, una vez solucionados esos problemas básicos y consiguiendo la funcionalidad completa del módulo Passport, nos dimos cuenta que la información a la cual se permite acceder utilizando este método en comparación con los modelos definidos previamente en nuestra aplicación que contenían una serie de parámetros obligatorios, era inferior, por lo cual se tuvieron que modificar los modelos de modo que los únicos campos obligatorios fueran aquellos comunes a los dos métodos posibles de autenticación, con Google y mediante Formulario de registro. Esto resultó en la necesidad desde la aplicación web, de realizar un formulario complementario en el caso del registro mediante Google que permitiera recolectar la información necesaria para complementar la información básica.

### Problemas a la hora de importar datos

A la hora de realizar la importación de los datos disponibles en Aragón Open Data, se tenía como idea inicial realizar inicialmente una actualización de los datos que ya existieran en nuestro sistema, y la introducción de los nuevos datos. Sin embargo, debido a que surgieron problemas con la búsqueda de los datos actuales con el fin de actualizarlos posteriormente, se tomó la decisión de realizar un borrado completo de los documentos y un nuevo poblado del modelo. Además, apareció un problema extra, dado que un pequeño número de datos necesarios en nuestra aplicación como datos de tipo entero, aparecían en la fuente de datos como dato null, por lo tanto se debió realizar una modificación en los modelos, restringiendo el número de datos requeridos de manera obligatoria. Además de este pequeño problema, la fuente de datos también tenía añadido otro problema, y es que el formato del fichero JSON que proporcionaba no coincide con el formato JSON que se llevaba trabajando en la aplicación durante todo el desarrollo del proyecto, y hacía mucho más complicado el acceso a los datos, debido a esto se tuvo que crear una función auxiliar que permitiera parsear los datos fuente a el mismo formato utilizado en nuestro servidor web:



---

```
/*
 * Función utilizada para obtener un JSON bien estructurado
 * con el que poder extraer los datos más fácilmente
 */
test = function(datos) {

    let ejemplo = datos;

    let claves = ejemplo[0];

    let retVal = [];

    ejemplo.shift();

    ejemplo.forEach(function(item, index) {

        let aux = {};

        claves.forEach(function(jsonitem, jsonindex) {

            if (item[jsonindex] == null) {

                aux[jsonitem] = "";

            } else {

                aux[jsonitem] = item[jsonindex];

            }

        });

        retVal.push(aux);

    });

};
```

---

```
    return retVal;  
}
```

Respecto a la creación de procesos en segundo plano para importar nuevos datos, la idea inicial era crear un proceso para cada tipo de entrada, utilizando el módulo node-schedule, pero durante las pruebas realizadas se observó que en ocasiones el json no se recibía correctamente. Al no encontrar un motivo claro, se optó por probar con un controlador y que los datos se actualizaran cuando llegase una petición por parte del administrador. Al ver que esta segunda solución funcionaba correctamente, al margen de los problemas descritos en el párrafo anterior, se prefirió optar por ella, aunque se desviase del requisito inicial.

---

## 12. ANÁLISIS DE PROBLEMAS POTENCIALES

En relación al servidor web de nuestra aplicación, el primero problema potencial que se nos ocurre es, debido al uso de tokens, que ocurre si estos tokens acaban siendo interceptados, dado que existen un pequeño número de peticiones en las cuales el token se utiliza como parámetro para poder obtener determinada información. Ante este hecho, se han definido todos aquellos puntos de entrada que no son los estrictamente necesarios para el acceso al sistema, para que antes de realizar ninguna tarea comprueben que se ha enviado el token identificativo en la petición y que este sea válido, una vez realizada esta comprobación se procederá a ejecutar la operación, en caso contrario se informará de que el token enviado no es correcto. De esta manera se cubre uno de los principales problemas potenciales.

Un problema potencial derivado del anterior, es referente a los tiempos de expiración que se incluyen en los tokens

Otro problema en torno al servidor web , son las NoSQL Injections que puede sufrir nuestra base de datos, este tipo de problemas si existe actualmente en nuestra implementación, dado que por ejemplo procedemos a realizar consultas en la base de datos con los mismos datos que recibe la petición, sin realizar ninguna comprobación de los mismos.

---

## 13. DISTRIBUCIÓN DE TIEMPO

Se ha llevado a cabo un control de esfuerzos, junto con las tareas dedicadas en una hoja excel, debido a que no cabe bien en el documento, se encuentra disponible en <https://github.com/EduardoGimeno/Sistemas-y-Tecnologias-Web/blob/master/Documentacion/Control/Esfuerzos.xlsx>.

Se ha utilizado la página <https://monday.com/lang/es/> para crear un tablero al que agregar las tareas de cada parte del sistema, asignándole un miembro del equipo encargado de llevarla a cabo, así como una fecha límite y una prioridad. Se ha podido exportar el tablero a una hoja excel, la cual se encuentra disponible en <https://github.com/EduardoGimeno/Sistemas-y-Tecnologias-Web/blob/master/Documentacion/Control/Tablero-Junio.xlsx>. Este tablero es el utilizado hasta junio, el utilizado hasta septiembre se encuentra disponible en <https://github.com/EduardoGimeno/Sistemas-y-Tecnologias-Web/blob/master/Documentacion/Control/Tablero-Septiembre.xlsx>.

También se han utilizado las issues de GitHub para definir las tareas de cada parte del sistema, creando dos tableros, uno para el front-end y otro para el back-end. Se ha utilizado como una vista global del sistema.

---

## 14. CONCLUSIONES

El desarrollo de este proyecto ha sido interesante a la hora de conocer una serie de nuevas tecnologías que se utilizan mucho actualmente. Node y Express permiten crear un servidor que soporte peticiones REST en muy poco tiempo y de forma muy rápida, es uno de los aspectos más positivos que tienen, la rapidez y sencillez que presentan a la hora del desarrollo. Integrar una base de datos como Mongo con Node y Express resulta sencillo con Mongoose, aunque Mongo presenta algunos inconvenientes como actualizar los documentos, resulta más sencillo borrarlo y volver a guardarlo que aplicar una actualización en sí sobre el mismo. El uso de angular en el cliente de la aplicación ha sido sin duda una experiencia agri dulce, debido a que como se ha podido apreciar es un framework para aplicaciones web con una gran cantidad de posibilidades, potente y que permite realizar trabajos muy interesantes. Pero por otro lado, como se estudió en la asignatura, tiene una curva de dificultad muy pronunciada, además de que puede resultar algo complejo. De los dos miembros del equipo que desarrollaron la aplicación, uno había trabajado con angular antes, lo que facilitó la toma de contacto, mientras que el otro miembro tuvo que estar estudiando el funcionamiento del entorno durante un tiempo al comienzo del desarrollo. El despliegue a través de Heroku no ha requerido demasiado tiempo, la configuración de las máquinas resulta sencilla y la integración con GitHub lo facilita mucho.

Por otra parte, respecto a la gestión del proyecto debido a lo acontecido los meses de marzo y abril, nos resultó complicado y en ese último tramo final del curso hubo mucho trabajo acumulado de todas las asignaturas, lo cual nos complicó el desarrollo del trabajo, siendo necesario seguir durante el verano.

---

## 15. VALORACIÓN PERSONAL DE CADA MIEMBRO

### **Gonzalo Berné:**

A pesar de que no pudimos finalizar el proyecto correctamente para la primera convocatoria, considero que hemos realizado un buen trabajo para esta segunda donde hemos implementado todos los puntos que quedaban pendientes y hemos pulido el resto de cosas ya incluidas. He aprendido muchas cosas nuevas mientras desarrollaba la aplicación, he trabajado casi por completo en el front-end y a pesar de que Angular era una tecnología con la que ya había trabajado me han surgido retos los cuales considero que he resuelto correctamente. Las tecnologías utilizadas en la parte del back-end sí que eran nuevas para mí por lo que me ha resultado muy satisfactorio aprender a utilizarlas. En definitiva, construir una aplicación web con el stack mean me parece una opción muy correcta sobretodo ahora que he participado en un proyecto con estas tecnologías.

### **Eduardo Gimeno:**

Este ha sido el primer proyecto en el que he usado estas tecnologías. En proyectos anteriores había usado Spring en el servidor, junto con bases de datos SQL, como PostgreSQL o MySQL. Programar el back-end con Node y Express ha resultado más rápido que con Spring en distintos aspectos, el uso de Mongoose junto con Mongo también agiliza mucho el desarrollo, por ejemplo en que no es necesario crear repositorios con operaciones para guardar o recuperar información de la base de datos, pero tiene, también, algunos inconvenientes, como no poder realizar join y operaciones de este estilo. Esto último se ha notado a la hora de calcular las estadísticas. Respecto al despliegue, Heroku ya lo había utilizado anteriormente, y en mi opinión, resulta algo más sencilla de utilizar que AWS o Google Cloud, aunque es una lástima que ya no vayan a proporcionar soporte para Mongo a partir de noviembre.

### **Jorge Fernández:**

Si bien el proyecto no ha concluido como planteamos, la valoración personal sobre el proyecto y en concreto sobre la parte de back-end que me toca es muy positiva, dado que me ha permitido interactuar con distintos módulos de muy variadas características, lo cual se ha trasladado en una importante búsqueda de información y solución de problemas, tanto con Nodejs como con MongoDB y todo ello me ha permitido desarrollar unos conocimientos que resultan muy útiles para la vida profesional.

### **Joaquín Puyuelo:**

Considero que el trabajo desarrollado en este proyecto me ha ayudado a aprender en gran medida los conceptos del desarrollo en Front-End. A pesar de que al comienzo del trabajo se hizo algo duro el aprender a utilizar Angular, valoro muy positivamente los conocimientos adquiridos, ya que estoy seguro de que me serán de gran utilidad en el desarrollo de aplicaciones en un entorno de la vida real.

---

## 16. ANEXO

Email creado para utilizar en la cuenta de administrador:

- descubrearagonstw@gmail.com
- STW-1920

Email utilizado para las entradas, para comprobar el funcionamiento del chat:

- entradaexample@gmail.com
- STW-STACK-MEAN

Cuenta de administrador:

- descubrearagonstw@gmail.com
- STW-1920