Visualización de analíticas de uso de rutas georeferenciadas para visitas arqueológicas

Pedro Angel García Morales

Resumen– Los administradores de rutas arqueológicas tienen problemas para controlar los usos de estas a tiempo real, teniendo que recurrir a cálculos manuales y controles de aforo estándar. Tampoco pueden saber que rutas son más utilizadas, dónde pasan más tiempo los usuarios y como varían las estadísticas en el tiempo. Este trabajo se centrará en el desarrollo de una página Front-end para el control de datos analíticos provenientes de rutas arqueológicas. Se usa el entorno de ejecución de Node.js (JavaScript) con el framework React. Mediante ese framework se crean los diferentes componentes que muestran los datos proporcionados a través una API.

Palabras clave– Front-end, Prototipo, React, Node.js, Javascript, API, Analíticas, OpenStreetMap, Json-server, Heatmap, Leaflet

Abstract– Administrators of archaeological routes have problems to control occupancy in real time, having to do manual statistics and gauging controls. They also do not know which routes are most used, where users spend more time and how statistics change over time. This work will focus on the development of a Front-end page for the control of analytical data coming from archaeological routes. It uses the Node.js (JavaScript) runtime environment with the React framework. This framework creates the different components that display the data provided through an API.

Keywords– Front-end, Prototype, React, Node.js, Javascript, API, Analytics, OpenStreetMap, Json-server, Heatmap, Leaflet

1 Introducción

La arqueología es una ciencia que se dedica al estudio de la sociedad humana mediante el análisis de la documentación y restos del pasado hasta hoy. Una ruta arqueológica comprende los restos que investigan los arqueólogos, exponiéndose de manera visual para el espectador. Esto permite aprender sobre costumbres y culturas de otras épocas sin tener un gran conocimiento previo. Uno de los principales problemas que tienen los administradores de estas rutas, es el control de las rutas. No obtienen la suficiente información de ellas para saber que está pasando en cada una de ellas. Recurren a métodos de control de aforo numérico, y una vigilancia activa mediante cámaras de video. Se encuentran con problemas de que la gente pasa más tiempo en unas zonas que otras, y que hay rutas que son más populares. Para mejorar su control sobre ellas deciden instalar un sistema de control GPS (Sistema de Posicionamiento Global), vincularlo con unos sensores que les permiten saber cuántas personas hay en las rutas, y qué puntos son los más calientes. A su vez les permite calculas otras métricas, cómo popularidad y porcentajes de uso. Una vez instalado, se encuentran con el problema de qué son incapaces de tratar con toda esa información. Tienen la necesidad de comunicar los sensores con algún lugar que sea capaz de sintetizar toda esa información, mostrarla de forma sencilla y visual. Este TFG (Trabajo de Fin de Grado) pretende abordar el problema tratando la situación desde la comunicación y tratamiento de los datos. Consiste en la obtención de los datos por parte de los sensores mediante una API (Interfaz de Programación de Aplicaciones). API [1] es una interfaz que mediante unos protocoles y definiciones nos permite intercambiar información entre dos sistemas diferentes. Posteriormente, se proyecta la información en una página web utilizando los componentes del framework React 18. React[2] una biblioteca de JavaScript que nos proporciona

• E-mail de contacto: 1531410@uab.cat
• Mención realizada: Tecnologias de la Información
• Trabajo tutorizado por: Aitor Alsina Rodríguez (DEIC)
• Curso 2021/2022

un marco de trabajo con conceptos y prácticas comunes a la hora de enfrentar un problema, orientada a aplicaciones una sola página. Empleará elementos visuales de manera sencilla para mostrar la información relevante en diferentes formatos: texto, tablas, gráficos y mapa de calor. Los componentes se montarán sobre una plantilla basada en Boostrap [3], una biblioteca centrada en el diseño de páginas web y aplicaciones.

Se presentarán los objetivos que tiene este trabajo, la metodología de trabajo y la planificación que se ha seguido, el estado del arte dónde se analizarán las decisiones tomadas y las alternativas, la arquitectura por la que se compone el proyecto, los pasos seguidos en el desarrollo de los prototipos, se mostrarán los resultados del desarrollo, y finalmente se resumirá el contenido en unas conclusiones incluyendo una línea de trabajo futuro.

2 Objetivos

El objetivo principal de este trabajo es dar respuesta a la necesidad de obtención, tratamiento y exposición de los datos. Para conseguirlo, a continuación se definen estos conceptos clave.

En la obtención de datos tenemos la problemática de que no existe una manera de que ambos sistemas se entiendan, por lo cual existe la necesidad de crear una API para permitir un flujo de información del sensor al servidor. Para solucionarlo se utilizará un mock, que nos permite simular las respuestas de los sensores, para que una vez acabo el proyecto, se puede integrar con cambios mínimos.

En el tratamiento de los datos tenemos que convertir los datos a información accesible, extraer los datos que nos interesan, darle el formato de salida adecuado.

Finalmente, en la proyección de los datos hemos de integrar los datos de salida en los componentes React adecuados: texto, tabla, gráfico y mapa de calor. Integrando los componentes una vez realizados en la plantilla Boostrap de administrador.

3 Metodología

La metodología que se seguirá es de tipo prototipo incremental, con la cual se crea un primer prototipo, realizando iteraciones incrementales para mejorar el prototipo. De esta manera se producen nuevas versiones hasta tener uno que cumpla los objetivos propuestos. En este trabajo se harán un total de tres prototipos mediante esta metodología que se detallarán en el siguiente apartado.

Se apoyará mediante en la metodología de Kanban [4]. Esta metodología consiste en gestionar las tareas de un proyecto de manera más visual, utilizando unas tarjetas, las cuales están agrupadas en diferentes columnas. Esas columnas dependerán del tipo del proyecto, y para este proyecto se utilizarán las siguientes: Por Hacer, Haciendo, Testeando, Hecho. La selección de las columnas se basa en los principios base de la metodología dónde están las tres principales (Por Hacer, Haciendo, Hecho), que permiten simplificar el estado de una tarea. También se añade la columna de testeo, una columna adicional que se usan en algunos entornos de programación. Su elección es realmente útil en los entornos web con los que se trabajan para conseguir los objetivos mencionados. Ya que cada componente puede dividirse en varias tareas diferentes, de manera modular, unificándose en la fase final del proyecto. En Kanban se lleva un control en cuanto a tareas dónde cada una tenga un coste igual o inferior a dos días de trabajo (10 horas). Se detallarán más sobre las tareas a continuación.

3.1. Planficación

La planificación del proyecto se ha centrado en el desarrollo de tres fases, una para cada prototipo, para permitir un correcto desarrollo de prototipado incremental. En una primera fase se van a crear los componentes React (apdo. 5.1): estadísticas porcentuales, tabla mensual de ocupación y popularidad, gráficos de ruta. Estos no dispondrán de conexión al Back-End (fig. 2), solamente mostrarán unos datos estáticos asignados con anterioridad. En la segunda fase se pretende dotar la conexión del Front-End del servidor del Back-End, con lo cual sea capaz de mostrar los datos que va recibiendo cada cierto tiempo, de manera que cualquier cambio se vea reflejado. En la tercera fase se realizará el prototipo final con el componente mapa de calor, y los cambios menores para ajustarse a la guía proporcionado por el tutor.

Entregas	Objetivos	Fecha
Informe Inicial	Definir objetivos, alcance, metodología y búsqueda de métodos para lograrlo	06/03
Informe de Progreso 1	Revisar objetivos y metodología, desarrollo del primer prototipo.	10/04
Informe de Progreso 2	Últimos cambios realizados, mostrar los resultados y explicación de las conclusiones. Desarrollo del segundo prototipo.	22/05
Informe final	Finalización del informe y acabar el desarrollo.	12/06
Presentación	Realización de la presentación resumiendo los aspectos más importantes.	26/06
Póster	Creación de un poster con los apartados más importantes del trabajo resumidos.	03/07

4 Estado del arte

En este apartado se explica las diferentes tecnologías actuales que permiten realizar un Front-end y los criterios que se han utilizado para la toma de decisiones. En la primera parte tenemos diferentes framework con los que podemos trabajar para la programación del Front-end, Angular, Vue o React. Vamos a hacer un análisis de los tres, teniendo en cuenta criterios como la dificultad de aprendizaje, tiempo de ejecución, requisitos, renderizado, modelo, uso, documentación, comunidad, y compañías que lo emplean.

4.1. Framework

Se realiza a continuación un análisis de los tres framework más utilizados para elegir uno con base en las características del trabajo.

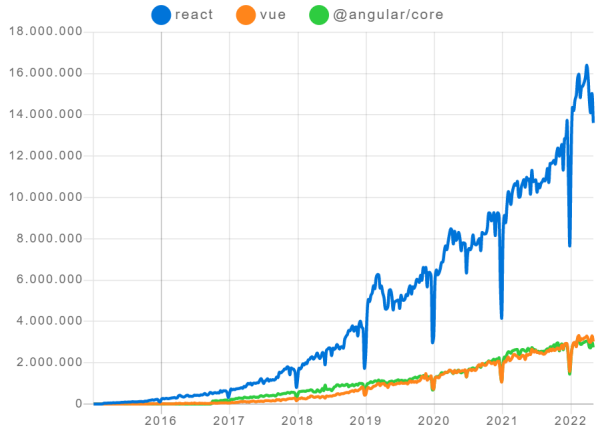


Fig. 1: Descargas de los diferentes frameworks durante los últimos años. Npmtrends [5].

4.1.1. Angular

Es un framework JavaScript que utiliza TypeScript[6], desarrollado por Google en el año 2010. TypeScript es lenguaje de programación que parte de la base de JavaScript añadiendo nuevos métodos, clases y tipos. El aprendizaje de este framework radica en el requisito obligatorio de aprender también a usar TypeScript, que introduce tipos estáticos a JavaScript. Su tiempo ejecución inicial es largo debido a su extenso código fuente, que puede alcanzar los dos o tres segundos. Para la creación de una aplicación podemos emplearlo sin requerir de otros componentes o librerías adicionales. Se renderiza el código en el cliente, lo que permite la carga de la página sin tener que recargarla cada vez que se realiza un cambio. Está basado en MVC (Modelo Vista Controlador), con lo cual el modelo se encarga de gestionar la información de la aplicación, vista muestra toda o parcialmente esa información, y el controlador gestiona el control entre la relación modelo-vista, a través de la interacción con el usuario, siempre validando sus acciones de este último. Su uso recae en entornos de producción, y aplicaciones con MaterialUI[7], una biblioteca que permite crear interfaces de usuario y sus componentes. Su documentación es extensa y variada, mantenida por Google, con una amplia comunidad.

4.1.2. Vue.js

Vue.js es un framework JavaScript de código libre desarrollado el año 2014 por Evan You y mantenido por él mismo conjuntamente por sus colaboradores. A diferencia de Angular, dispone de un tiempo ejecución rápido. Es un framework muy ligero que es fácil de aprender y utilizarlo; con una gran utilización en las startups y entornos de producción. En su contra, tenemos una dependencia de otras aplicaciones de terceros para su funcionamiento en un entorno web moderno. Tenemos una vinculación de datos bidireccional, con lo cual el usuario puede interactuar con los datos del servidor; con un renderizado en la parte del servidor. Se apoya en un modelo de DOM Virtual, permite controlar los cambios en el DOM real y actualizar solo los componentes que han sufrido una modificación. Su documentación está mantenida por la comunidad, y a pesar de ser el último de los tres en crearse, mantienen un gran auge en sus descargas en los últimos años. Es empleado por compañías como Facebook, Adobe, Alibaba, etcétera.

4.1.3. React

React es un framework Javascript que fue lanzado por Facebook en el año 2013, utilizado inicialmente para mejorar la mantenibilidad de los anuncios de la compañía. Es un framework orientado a interfaces de usuario, con una dificultad media de aprendizaje y rápida ejecución. Para crear una web funcional de la actualidad, requiere la instalación de otras librerías para explotar todo su potencial. Consta de un sistema de vinculación de los datos unidireccional, del servidor al cliente. Usa el mismo modelo DOM Virtual que se ha explicado anteriormente. Se emplea normalmente para entornos de producción e interfaces personalizadas de usuario. Cuenta con una gran documentación, y una gran cantidad de descargas (Fig. 1), que avalan su enorme comunidad.

Se ha elegido entre los tres framework el uso de React por qué está orientado a las interfaces de usuario, tiene un gran comunidad y de documentación, además que es suficiente tener una vinculación de datos unidireccional.

4.2. Plantilla Boostrap

El proyecto utiliza una plantilla basada en Boostrap para mostrar los componentes React. Con el objetivo de seleccionar una de ellas de una gran lista [8], se explican varias plantillas que encajan con una interfaz moderna, sencilla y visualmente agradable.

4.2.1. AdminKit

La plantilla Adminkit [9] está bajo licencia de MIT[10], que permite gratuitamente la copia, comercialización, modificación, incluyendo la misma licencia. Tiene una versión gratuita y otra de pago. La gratuita contiene menos páginas y complementos visuales. Visualmente contiene un apartado de estadísticas de forma cuadrada, un apartado de calendario, un gráfico de sector de uso, un gráfico de movimientos, y un mapa de uso real.

4.2.2. Material Dashboard 2

La plantilla Material Dashboard 2 [11] está bajo licencia de MIT, con dos versiones diferentes: gratuita y premium. La segunda proporciona bloques de código prediseñado para facilitar la integración y más de tres cientos de diseños de elementos visuales. Respecto al diseño, también contiene un apartado cuadrado de estadísticas, un apartado central de tres gráficos pequeños, una tabla de proyectos, y un historial gráfico de compras.

4.2.3. Star Admin 2 Free

La plantilla Star Admin 2 Free [12] está bajo licencia de MIT, solo existe una versión gratuita, visualmente muy similar al anterior, pero tiene un diseño más minimalista, con un apartado de ilustraciones, además de todo lo mencionado en el Material Dashboard 2.

Se selecciona finalmente el Adminkit por qué contiene un bloque destinado únicamente a mapas, que permitirá una integración más intuitiva y con menores cambios en la plantilla original.

4.3. API

El desarrollo necesita una fake API para poder obtener la información que se muestra en el Front-end. Se analizan varias librerías del repositorio NPM [13] que cumplen esa función.

4.3.1. json-server

El paquete json-server, bajo licencia MIT, permite con la creación de un único fichero de tipo json devolver su contenido total, o parcialmente aprovechando la estructura jerárquica que dispone este formato. De manera que una vez accedido a la ruta, devuelve en contenido existente en el fichero dentro de la estructura.

4.3.2. json-mock-api

Este paquete json-mock-api, bajo licencia MIT, permite crear una estructura de ficheros, donde cada carpeta con fichero de tipo json se puede acceder mediante una ruta. La ruta no es más que la dirección IP del servidor local, más el directorio que se encuentra el fichero a visualizar. Se ha de tener en cuenta el directorio de referencia que utilizamos. Además, también permite personalizar las respuestas obtenidas según el tipo de la llamada: GET, POST, PUT, DELETE.

4.3.3. mocks-server

Esta librería mock-server, bajo licencia MIT, se ha generado para entornos para simular condiciones de producción, que permite usar protocolos de seguridad. Además de tener como característica base su trazabilidad, escalabilidad, soporte a CORS (Intercambio de recursos de origen cruzado), y muchas funciones más. Su configuración es tan extensa como complicada, lo que permite a su vez una gran personalización en las respuestas, y tratamiento de la información.

Se elige por su facilidad, eficacia, y poca configuración la librería json-server, ya que permite realizar todo lo que se propone en el trabajo sin requerir opciones extra que aportan el resto de ellas. El uso de paquetes de difícil configuración no aportaría más valor para un sistema prototipado, puesto que dificultaría su posterior versión final, y dejando de lado en lo que se centraría este trabajo.

4.4. Mapa de calor

Para el mapa de calor existen diferentes formas de realizarlo. Para realizarlo debemos diferenciar dos capas: Map Base, Map Layer. El primero sirve como estructura base del mapa de calor, proporcionando el lugar de la ruta sin aplicar la capa posterior de calor. El último proporciona los puntos a insertar en la primera capa, y la intensidad con la cual se van a mostrar (calor).

Para la capa inicial tenemos la posibilidad de usar Google Maps, OpenStreetMap, NPAC. NPAC tiene una limitación, solo muestra algunas regiones de Estados Unidos, y como nuestra intención es mostrar una zona cercana Barcelona, queda descartado. Para utilizar Google Maps se requiere una cuenta de pago activa, y muestra un mensaje de que se limita su uso únicamente en desarrollo en el modo de prueba. Por lo tanto, se decanta por la opción de OpenStreetMap. Para cargar un mapa se necesita una librería que cargue el mapa en cuestión, se usará Leaflet, por ser la más completa y mayor documentada.

Para la capa superior, dónde están los puntos con su respectiva intensidad, se emplea heatmap por ser la más utilizada, y con información hasta la fecha. No se encuentran otras que fueran relevantes. Permite añadir una capa encima del mapa OpenStreetMap cargado por Leaflet, con una gama de colores fríos y cálidos según la intensidad.

4.5. Gráficos

Los gráficos forman parte de los componentes visuales que se agregaran mediante React, para su creación se necesita depender de librerías que permitan su visualización mediante los datos de la API. Se comparan tres librerias de una gran lista disponible [14].

4.5.1. chart.js

Es una librería de uso libre bajo licencia MIT, que tiene nueve tipos de gráficos: área, barras, burbujas, sectores, línea, híbridos, polares, radar y de dispersión. Su configuración tiene bastantes opciones a configurar antes de poder visualizar un gráfico.

4.5.2. recharts

Es una librería de utilización libre bajo licencia MIT, que permite disponer de gráficos similares a la anterior, con la ventaja de tener una configuración más sencilla, y con una gran cantidad de ejemplos disponibles. Tiene la capacidad de ser *responsive*, lo cual permite adaptarse al tamaño del dispositivo, para mejorar el renderizado.

4.5.3. anychart

Es una librería de uso comercial que dispone de una versión de prueba, proporciona todo tipo de gráficos, más de setenta de ellos. Contiene una documentación extensa, con ejemplos para realizar cada tipo de ellos. La creación del gráfico es sencilla.

Debida su capacidad, *responsive* su mayor facilidad de configuración, y su coste nulo, se ha elegido utilizar recharts, que dispone del gráfico de líneas capaz de adaptarse a las diferentes situaciones.

5 Arquitectura

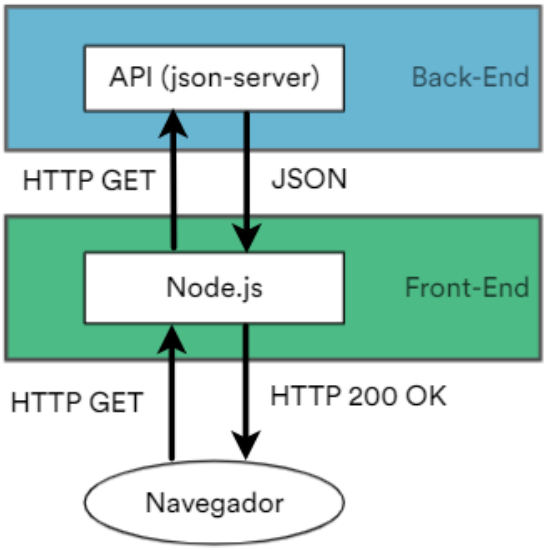


Fig. 2: Diagrama de arquitectura.

Dentro de la arquitectura (fig.2), tenemos dos partes bien diferenciadas: Front-end, Back-end. En nuestro proyecto nos servimos de un node.js que nos permite mostrar la página web, que realiza llamadas GET[15], una técnica de envío de información mediante el protocolo HTTP[16]. Estas llamadas nos permiten enviar una solicitud de información a nuestro servidor Back-end (json-server) a una ruta para que nos devuelva la información en formato json. Una vez que nuestra Front-end la recibe la trata para poder ser mostrada en los componentes React, que hablamos en el siguiente subapartado.

5.1. Front-end: Componentes

En este apartado se describen los componentes que se muestran en el Front-end, que muestran la información que obtienen desde una API, actualizando cada intervalo de tiempo de 5 segundos. Se basan en mostrar la información sobre dos rutas, enfocándose en las características de popularidad y ocupación. Se dividen en tres bloques: estadísticas porcentuales, tabla mensual, y gráficos de ruta; formados por un total de ocho componentes.

- Estadísticas porcentuales: formadas por cuatro componentes diferentes que muestran de manera porcentual de la ocupación y popularidad de ambas rutas.
- Tabla mensual de ocupación y popularidad: formado por un único componente que muestra en una table de cinco columnas con el mes, la ocupación de la ruta 1 y de la ruta 2, la popularidad de la ruta 1 y de la ruta 2.
- Gráficos de ruta: formados por dos componentes que muestran la relación entre popularidad y ocupación de cada ruta.
- Mapa de calor: formado por un solo componente que muestra las zonas de más actividad dentro de una ruta mediante un gradiente de color que varía en función del tiempo que pasan los visitantes.

5.2. Back-end: API

Para solucionar la problematica de no tener una API para obtener los datos, se utiliza la libreria *json-server* que permite ejecutar un servidor local, y mediante unas rutas obtener los datos que utilizamos en la arquitectura Front-end (apdo. 5.1).

- `GET /api/v1/routes`

  Descripción: Devuelve información sobre todas las rutas. En los datos que proporciona tenemos un identificador de la ruta, el nombre, el porcentaje, ocupación y la popularidad.

  Parámetros: No requiere de ninguno.

- `GET /api/v1/activity`

  Descripción: Devuelve el registro histórico de todas las rutas. Concretamente una lista con un registro por cada mes y ruta, con la ocupación y popularidad.

  Parámetros: No requiere ninguno.

- `GET /api/v1/chart/{id}`

  Descripción: Devuelve la información necesaria para la creación del gráfico. Se trata de una lista con las estadísticas porcentuales de la ruta cuyo identificador se ha pasado por parámetro.

  Parámetros: `id` (valor numérico, obligatorio), sirve como identificador de la ruta.

6 Desarrollo

El desarrollo se ha dividido en tres prototipos diferentes: prototipo inicial, prototipo funcional, prototipo final. El primer prototipo abarca la creación de los componentes React (apdo. 5.1): estadísticas básicas, tabla mensual de ocupación y popularidad, y los gráficos de la ruta. Todo ello sin utilizar la API, solo con datos estáticos. El segundo genera la fake API, y conecta los componentes del anterior prototipo con ella. El último añade el componente mapa de calor, que visualizará los puntos calientes de una ruta.

En esta sección entraremos más en detalle con los pasos seguidos para el desarrollo de cada uno de los tres prototipos.

6.1. Prototipo inicial

En los primeros pasos que realizamos, es la instalación del entorno de React. Para ello necesitamos tener instalado npm, el gestor de paquetes que nos permite descargar todas las dependencias que utilizamos. Con él se puede realizar la descarga de *react*, *react-dom* [17]; una librería que nos permite renderizado de componentes en el DOM (Modelo de Objetos del Documento). Usamos también *web-pack* [18] que permite comprimir las dependencias de la web en un solo fichero, que mejorará nuestra página para obtener el Single-Page Aplicación (Aplicación de una Sola Página). También se instala *babel-core* [19] que permite la traducción de código JavaScript de última generación a una compatible con cualquier navegador. Una vez instalado, creamos nuestros componentes en *App.js*. Para que sean capaces de verse en el fichero *index.html*, el archivo que mues-

tra la página de inicio, se añaden unos delimitares de código HTML[20], div, con unos identificadores personalizados para indicar dónde React integrará nuestro componente.

Para el primer prototipo, se trata de simplificar el código de los componentes, para evitar los menores cambios en el codigo HTML, se decide que las estadísticas porcentuales se muestren en cuatro componentes diferentes: OcupR1, OcupR2, PopR1, PopR2. De esa manera se puede insertar los componentes en diferentes lugares, respectando el código fuente de la plantilla Bootstrap. También se crea un componente padre que permitirá la visualización de todos los componentes del proyecto, siguiendo la estructura de componentes de React. Al insertar los div con los identificadores de cada componente en diferentes regiones del código fuente, surge un problema. Debido a la estructura de componentes del framework, en que un nodo raíz (padre) contiene el resto de los componentes. Cómo los componentes hijos están en diferentes zonas de código, fuera de la jerarquía del padre, se utilizan Portales [21], una forma de renderizar los componentes fuera de jerarquía del DOM del padre (fig. 3).

```
function PopR2({PopR2Text}) {
  return ReactDOM.createPortal(
     <h1 className="mt-1 mb-3">{PopR2Text}</h1>,
     container4
   );
}
const container4 = document.getElementById('app4');
```

Fig. 3: Utilización de Portales React en el componente PopR2.

La tabla mensual es un componente que devuelve una tabla HTML (Lenguaje de Marcas de Hipertexto), en ella tenemos las siguientes columnas: Mes, Ocupación Ruta 1, Ocupación Ruta 2, Popularidad Ruta 1, Popularidad Ruta 2.

Se puede ver el resultado del prototipo en la figura 12 del Apéndice A.1.

6.2. Prototipo funcional

En la fase del prototipo funcional tenemos la conexión del Front-end al Back-end (fig. 2). Para lograr esa conexión surge la necesidad de desarrollar una mock API (apdo. 5.2), que nos permite devolver resultados a llamadas con datos que no son verdaderos, pero sirven para comprobar el funcionamiento de los sistemas. Mediante la libreria json-server utilizaremos la información estructurada en un fichero de tipo json, el cual se nombrará como db.json, para ejecutar nuestra API. Se ejecutará dicha libreria a partir del siguiente comando: `json-server --watch db.json --routes routes.json`

Una vez ejecuta la instrucción se abrirá un servidor local en el puerto 3000, con lo cual se cargará de forma jerárquica las rutas en el archivo.

El contenido del fichero db.json mantiene las rutas que se han explicado en apdo. 5.2. En esta parte profundizamos en como se obtienen y tratan los datos. Para conseguir mostrar los datos de estadísticas porcentuales que se muestran en la fig. 4, primero se ha de realizar una llamada a la ruta /api/v1/activity. Con una única llamada se pueden obtener los datos de los cuatro componentes del bloque mencionado.

```
{
  "routes": [
    {"id":1, "name": "route1", "occupation": "50%", "popularity": "68%",
       "var1": "-3.22%", "var2": "-2.12%" },
    {"id":2, "name": "route2", "occupation": "30%", "popularity": "32%",
       "var1": "-4.41%", "var2": "-1.21%" }
  ],
  "activity": [
    {"route": "1", "month": "enero 2020", "occupation": "51%", "popularity": "65%"},
  {"route": "1", "month": "febrero 2020", "occupation": "55%", "popularity": "58%"},
  {"_comments": "More data"},
  {"route": "2", "month": "enero 2020", "occupation": "33%", "popularity": "35%"},
  {"route": "2", "month": "febrero 2020", "occupation": "21%", "popularity": "42%"},
  ],
  "chart": [
    {"id": 1, "content": [
      {"month": "enero 2020", "occupation": "51", "popularity": "65"},
    {"month": "febrero 2020", "occupation": "55", "popularity": "58"},
    {"_comments": "More data"},
  ]},
  {"id": 2, "content": [
    {"month": "enero 2020", "occupation": "33", "popularity": "35"},
    {"month": "febrero 2020", "occupation": "21", "popularity": "42"},
    {"_comments": "More data"},
  ]}
  ]
}
```

Fig. 4: Fichero db.json.

Para recoger esa información, el componente padre efectúa la llamada. La respuesta que obtenemos se divide en variables diferentes mediante la iteración de los datos, separándolos mediante la variable ruta. Se utiliza una funcionalidad de React llamada Hooks.

Los Hooks fueron creados para reducir el uso de clases para controlar estados. Ahorrando la necesidad de contruir una sistema de jerarquico para que estos inyectaran información a a otros componentes para controlar los estados.

```
let [text, setText] = useState('')
```

Con esta instrucción podemos utilizar posteriormente la función *setText* pasarle por parámetro un *string*, y este se asignará posteriormente a la variable *text*. Siguiendo este patrón se asigna los datos recibidos de iterar la respuesta de la llamada y son almacenados mediante las funciones de los *Hooks*[22].

Los mismos pasos se realizan para las rutas de /api/v1/chart/{id}, /api/v1/routes, pero en estos casos son llamadas que se realizan en paralelo. Se crea un intervalo, que utiliza las funciones que gestionan las llamadas, cada un periodo de tiempo de 5 segundos (5000ms). Las respuestas más comunes por parte del servidor json-server son:

- HTTP 200 OK: Cuando nos devuelve la información correctamente.
- HTTP 304 NOT MODIFIED: Cuando la información no ha cambiado desde la última respuesta.
- HTTP 404 NOT EXISTS: Cuando la llamada se ha ejecutado a una ruta inexistente.

En el primer caso, la información será mostrada por los componentes. En el siguiente caso la información de los componentes no variará, así que no habrá cambios. El último se debe a problema de configuración, ya que puede haber un error en la escritura de la ruta, o la ejecución de un servidor con un archivo db.json erróneo.

La información que se intercambia entre lel componente padre y sus hijos, se realiza mediante *Props*[23]. Permite que al visualizar el componente hijo desde el padre, puede pasarle información mediante una Prop.

```
<OcupR1 OcupR1Text=OcupR1Text</OcupR1>
```

De esta manera, el hijo podrá usar la información obteniéndola mediante una variable por parámetro. Cuando la información está en los hijos se utiliza una función para los cuatro componentes de estadísticas básicas (fig. 7) para que pasando la información devuelve el código HTML con ella incluida. En el caso de la tabla (fig. 9) se usa una función que itera la array con la información y devuelve por cada conjunto un código HTML utilizando *tr* para la fila, y td para las columnas. En el gráfico (fig. 8) se pasa por parámetro una variable *data* con toda la información de ambos ejes: mes y porcentaje. En el cual se muestran tanto el porcentaje de ocupación como el de popularidad.

El gráfico se ha realizado mediante el uso de la libreria recharts (apdo. 4.5.2). Con él se ha creado una LineChart, gráfico de líneas, el cual consta de dos ejes X, Y. Eje X tiene el nombre de *mes*, y el Y no tiene nombre asignado. Está formado por dos líneas *monotone*, de un solo color, de colores rojo (popularidad) y azul (ocupación).

Se puede ver el resultado del prototipo en la figura 13 del Apéndice A.2.

6.3. Prototipo final

En el prototipo final se realizan los mapas de calor de las rutas 1 y 2. Para hacerlas, primero necesitamos obtener puntos de los mapas. Se proponen dos rutas pertenecientes a la provincia de Barcelona:

- Ruta prehistorica, La Roca del Vallès
- Yacimiento ibérico de la Fuente de Caña, Avinyó Nou

Se delimitan las dos rutas mediante la unión de puntos en un mapa con el uso de la aplicación de GPSVisualizer [24], una plataforma online que permite en su modo *draw* visualizar mapas, y crear una ruta mediante puntos con el cursor (fig. 5).

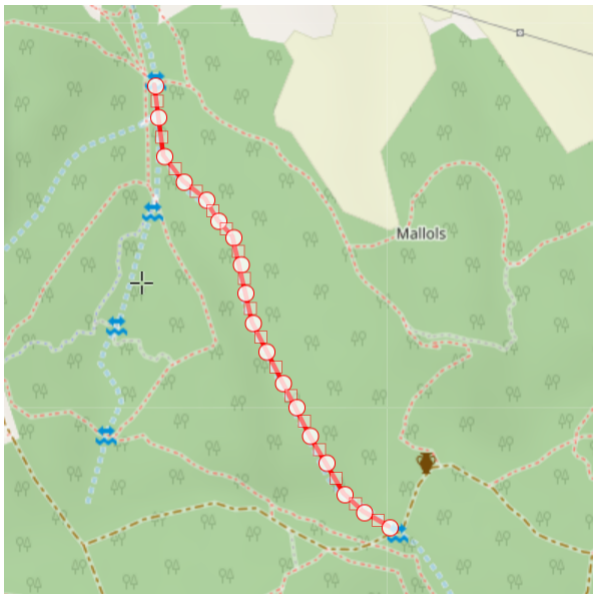


Fig. 5: Marcado de puntos para la creación de una ruta.

Después se exporta la ruta a formato GSPX. Un formato que nos permite guardar las coordenadas de todos puntos. Para poder trabajar con él, necesitamos convertirlo en un formato que pueda leerlo la librería de Leaflet llamado geojson (fig. 6). Para realizar el cambio se utiliza la aplicación de MyGeodata Converter [25], una página que permite intercambiar entre diferentes tipos de formato, y se usa el cambio de formato GSPX a formato geojson subiendo el fichero exportado anteriormente. .

```
const geojson = {
  type: "FeatureCollection",
  crs: { type: "name", properties: { name: "urn:ogc:def:crs:OGC:1.3:CRS84" } },
  features: [
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3367620, 41.5901400, 500] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3368049, 41.5898032, 500] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3368907, 41.5893862, 400] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3371696, 41.5891135, 900] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3374915, 41.588921, 1080] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3376632, 41.5886965, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3378778, 41.58852, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.337985, 41.5882313, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3380494, 41.5879266, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3381567, 41.5876058, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3383498, 41.587301, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3385859, 41.5869642, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.338779, 41.5867075, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3389721, 41.5864028, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3392081, 41.586114, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3394656, 41.5857772, 50] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3397446, 41.5855847, 5000] } },
 { type: "Feature", geometry: { type: "Point", "coordinates": [ 2.3401093, 41.5854243, 3000] } }
  ]
};
```

Fig. 6: Datos en formato geojson.

Con Leaflet (apdo. 4.4) podemos utilizar el mapa que actuará como base, OpenStreetMap, lo fijaremos en un punto de la ruta con una ampliación suficiente como visualizarla en su totalidad. Para cargar el mapa se ejecutan llamadas a OpenStreeMap con las coordenadas, para que uniéndose las imágenes que se proporcionan se pueda ver un mapa. La librería heatmap (apdo. 4.4) se encarga de crear una capa por encima de mostrar los puntos contenidos en geojson, también se pasa por parámetro la intensidad con lo que se muestra los puntos en el mapa. Según el nivel de intensidad se mostrará un color más frío (azul) o más cálido (rojo). Se puede ver el resultado del prototipo en la figura 14 del Apéndice A.3.

7 Resultados

En resultados se mostrarán los aspectos que se valoraran para ver el grado de cumplimiento los objetivos propuestos inicialmente en el proyecto. Para ello se explicará el aspecto, apoyándose de imágenes para permitir una verificación más sencilla y visualmente atractiva.

Un primer objetivo que se valora es haber podido realizar todos los componentes mencionados en los Objetivos y explicados en la fase de Arquitectura. Otro aspecto es comprobar si se ha asegurado la comunicación entre Front-End y Back-End, siendo este el objetivo principal del segundo prototipo. Para ello vamos a ver si es capaz de mostrar los datos proporcionados por la API (fig. 4). Finalmente, se revisa si con el cumplimiento de los dos anteriores se consigue solucionar el problema explicado en la introducción del trabajo.

Esta figura 7 muestra la visualización del componente de estadísticas básicas, que muestra información de manera sencilla y visualmente atractiva. Lo que permite que el gestor de rutas pueda ver el estado actual con una sola mirada.

Los gráficos (fig. 8) son elementos visuales que permiten visualizar la información de las rutas para poder detectar cualquier problema con ella, y tener una gestión eficaz. Por

Fig. 7: Componentes del bloque de estadísticas básicas.

cada ruta habrá que realizar una llamada diferente a la API (apdo. 5.2) especificando el identificador de la ruta.

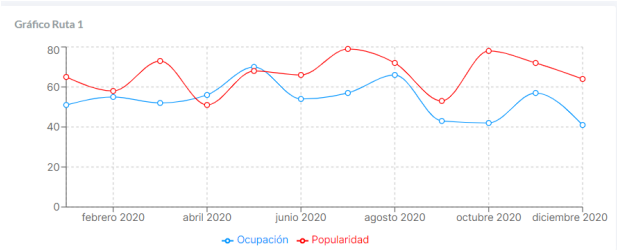


Fig. 8: Componente del gráfico de ruta.

La tabla proporciona la información para una consulta ágil y rápida, para comprobar métricas de otras fechas o múltiples variables en un mismo tiempo.

Estadísticas en tabla

Mes	Ocupación Ruta 1	Ocupación Ruta 2	Popularidad Ruta 1	Popularidad Ruta 2
enero 2020	51%	65%	33%	35%
febrero 2020	55%	58%	21%	42%
marzo 2020	52%	73%	21%	27%
abril 2020	56%	51%	35%	49%
mayo 2020	70%	68%	25%	32%
junio 2020	54%	66%	24%	34%
julio 2020	57%	79%	27%	21%
agosto 2020	66%	72%	31%	28%
septiembre 2020	43%	53%	31%	47%
octubre 2020	42%	78%	21%	22%
noviembre 2020	57%	72%	35%	28%
diciembre 2020	41%	64%	31%	36%

Fig. 9: Componente de la tabla mensual de ocupación y popularidad.

El mapa de calor (fig. 10 y 11) proporciona los puntos calientes de una ruta para controlar de manera eficaz las zonas dónde las personas pasan más tiempo, y saber los gustos de los visitantes, con el objetivo mejorar la experiencia de los usuarios.

Como se ha visto en las figuras anteriores se han podido realizar todos los componentes: estadísticas básicas, tabla de métricas, gráficos de ruta, y los mapas de calor. Por lo tanto, se cumple uno de los objetivos principales.

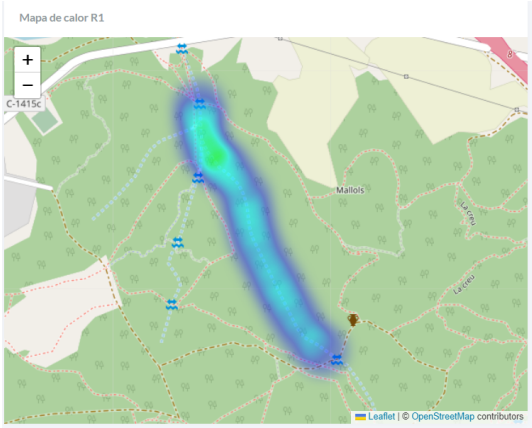


Fig. 10: Componente de mapa de calor de la ruta 1.

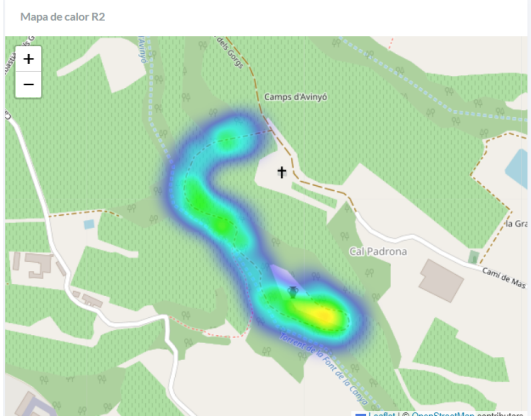


Fig. 11: Componente de mapa de calor de la ruta 2.

En las figuras 7, 8, 9, podemos ver la misma información contenida en el fichero db.json (fig. 4), la información proporcionada por la API. Si sufre una modificación este archivo, se ve reflejado el cambio en el componente afectado en pocos segundos. Por lo tanto, podemos asegurar que existe una comunicación eficaz entre Front-end (Apéndice A.3) y Back-End (fig. 4). Cumpliendo nuestro segundo objetivo funcional.

Mediante el cumplimiento de esos dos objetivos principales se proporciona una visualización de datos geo referenciados de las rutas arqueológicas, para que un gestor pueda utilizarlos para un control eficaz de las mismas.

8 Conclusiones

En este trabajo se ha desarrollado un prototipo completamente funcional de Front-End (Apéndice A.3) mediante un sistema de prototipado incremental. La creación del Front-End, con componentes React, permite ahora a los gestores de ruta conocer el uso real por parte de los usuarios en las rutas a través de la visualización de información mediante estadísticas, gráficos y mapas de calor. Durante la creación de la solución se han presentado varios problemas, la necesidad de una fuente de información como una API que se ha resuelto mediante una librería (apdo. 5.2). La necesidad de generar información geográfica de las rutas (apdo. 6.3), solucionado mediante la unión de puntos en un mapa con la plataforma de GpsVisualizer, y los cambios de formato GSPX a geojson mediante la página de MyGeodata

Converter. Cada prototipo ha ido mejorando el anterior. El primero solo mostraba información estática sin necesidad de API en los componentes de textos porcentuales, tablas de datos y gráficos. El segundo conectaba la información a la API propuesta (apdo. 5.2). El último agregaba el componente mapa de calor 4.4, acercándonos finalmente a los objetivos propuestos. En el proyecto, para realizar todos los pasos anteriores, se ha hecho un estado del arte para elegir las opciones más adecuadas y óptimas para el proyecto de desarrollo, definiendo los diferentes framework, visualizadores de mapas, y librerías.

Una línea de continuación del trabajo sería la creación de un sistema de control de acceso a los datos, para que únicamente los usuarios indicados puedan acceder al sistema de visualización de los datos. Se podría incluir más componentes para mostrar estadísticas diferentes que sean adecuadas para el gestor. Así como desplegagle para filtrar de manera visual los datos.

Su realización ha permitido adquirir conocimientos transversales en el ámbito de la comunicación de datos entre dos sistemas diferentes (Front-End, Back-End), así como la administración de componentes dinámicos mediante un framework, como es React. Ha permitido ver todos los pasos de un entorno real: la creación de los datos, las peticiones de información, el formateo de los datos, y finalmente la visualización; el aspecto en que se ha centrado el trabajo.

Agradecimientos

Me gustaría agradecer a mi tutor Aitor Alsina Rodríguez por su guía a la hora de realizar este trabajo, aportando su tiempo para procurar que siguiera las líneas establecidas y no me perdiera en el camino.

Agradecer también a mi familia por apoyarme para llegar a redactar este trabajo con la mejor de mis intenciones.

Referencias

[1] ¿Qué es una API?. (2017). Consultado en febrero del 2022. [En línea]. Disponible en https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

[2] React – Una biblioteca de JavaScript para construir interfaces de usuario. (2022). Consultado en febrero del 2022. [En línea]. Disponible en https://es.reactjs.org/

[3] Boostrap Team. (2022). Introduction. Consultado en febrero del 2022. [En línea]. Disponible en https://getbootstrap.com/docs/5.1/getting-started/introduction/

[4] Qué es Kanban: Definición, Características y Ventajas. (2022). Consultado en febrero del 2022. [En línea]. Disponible en https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

[5] @angular/core vs react vs vue — npm trends. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://www.npmtrends.com/react-vs-vue-vs-@angular/core

[6] TypeScript is JavaScript with syntax for types. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://www.typescriptlang.org/

[7] MUI: The React component library you always wanted. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://mui.com/

[8] 15 Best Free Bootstrap Admin Templates 2022 - aThemes Consultado en marzo del 2022. [En línea]. Disponible en https://athemes.com/collections/free-bootstrap-admin-template/

[9] GitHub - adminkit/adminkit: AdminKit is a free, open-source HTML dashboard, admin template based on Bootstrap 5. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://github.com/adminkit/adminkit

[10] MIT License. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://choosealicense.com/licenses/mit/

[11] Material Dashboard 2 by Creative Tim. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://www.creative-tim.com/product/material-dashboard/?partner=49926

[12] Star Admin 2 Free by Creative Tim. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://www.bootstrapdash.com/product/star-admin-free/?ref=8

[13] npm. (2022). Consultado en febrero del 2022. [En línea]. Disponible en https://www.npmjs.com/

[14] Gerchev, I., Rahman, S. (2022). 15 JavaScript Libraries for Creating Beautiful Charts - Sitepoint. Consultado en marzo del 2022. [En línea]. Disponible en https://www.sitepoint.com/best-javascript-charting-libraries/

[15] GET - HTTP — MDN. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://developer.mozilla.org/es/docs/Web/HTTP/Methods/GET

[16] HTTP — MDN. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://developer.mozilla.org/es/docs/Web/HTTP

[17] ReactDOM – React. (2022). Consultado en febrero del 2022. [En línea]. Disponible en https://es.reactjs.org/docs/react-dom.html

[18] Concepts — webpack. (2022). Consultado en febrero del 2022. [En línea]. Disponible en https://webpack.js.org/concepts/

[19] What is Babel? · Babel. (2022). Consultado en febrero del 2022. [En línea]. Disponible enhttps://babel.dev/docs/en/

[20] HTML: Lenguaje de etiquetas de hipertexto — MDN. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://developer.mozilla.org/es/docs/Web/HTML

[21] Portales – React. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://es.reactjs.org/docs/portals.html

[22] Presentando Hooks – React. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://es.reactjs.org/docs/hooks-intro.html

[23] Componentes y propiedades – React. (2022). Consultado en marzo del 2022. [En línea]. Disponible en https://es.reactjs.org/docs/components-and-props.html

[24] Gpsvisualizer.com. 2022. GPS Visualizer: Freehand Drawing Utility: Draw on a map and save GPX data. Consultado en marzo del 2022. [En línea]. Disponible en https://www.gpsvisualizer.com/draw/

[25] Mygeodata.cloud. 2022. GPX to GeoJSON Converter Online - MyGeodata Cloud. Consultado en marzo del 2022. [En línea]. Disponible en https://mygeodata.cloud/converter/gpx-to-geojson

APÉNDICE

A.1. Vista del prototipo inicial

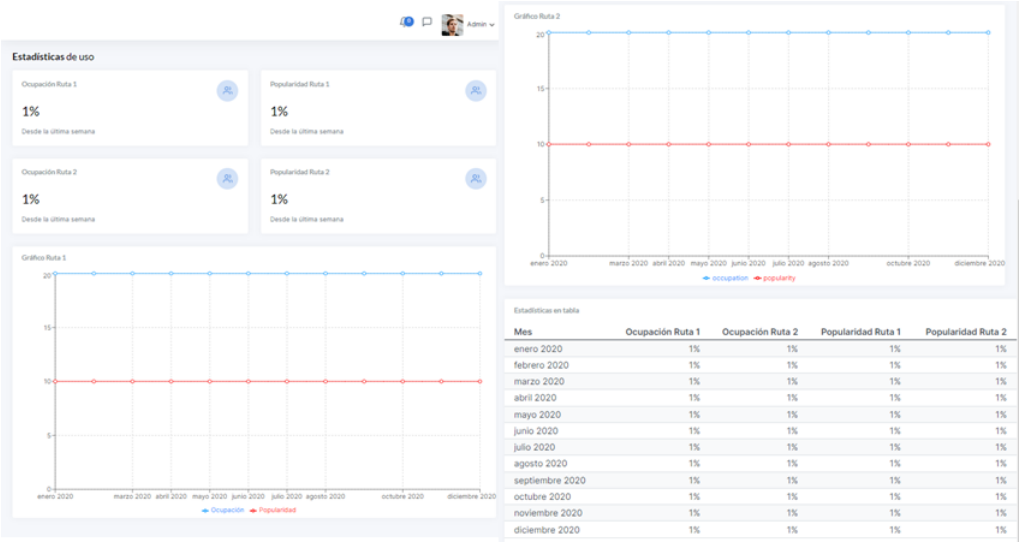


Fig. 12: Prototipo incial.

A.2. Vista del prototipo funcional

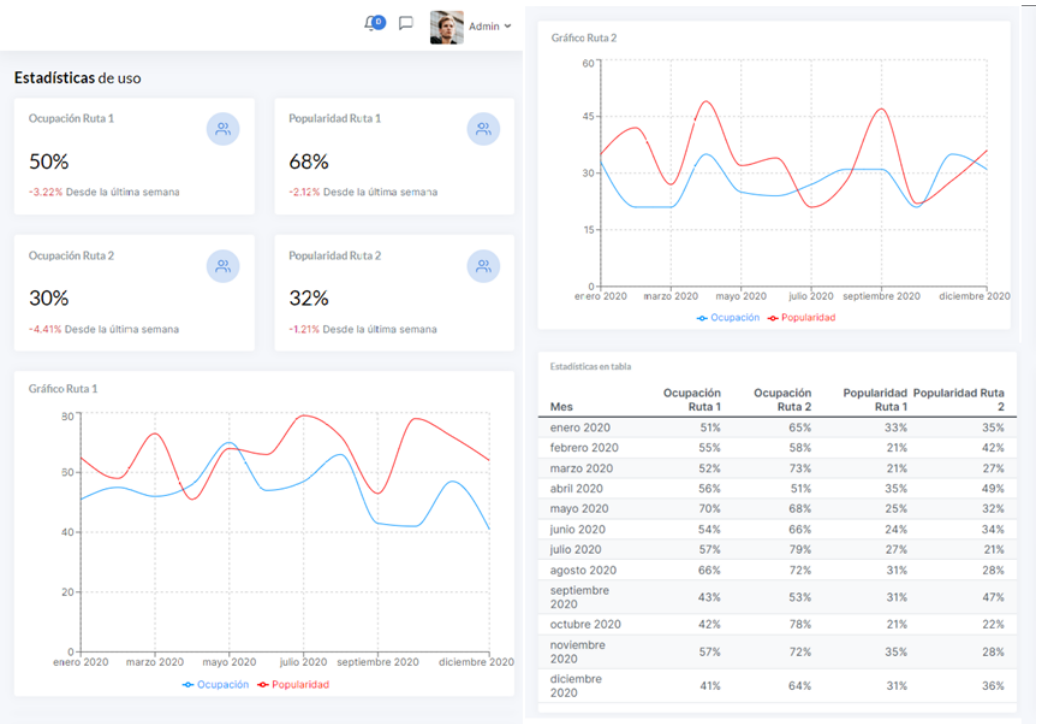
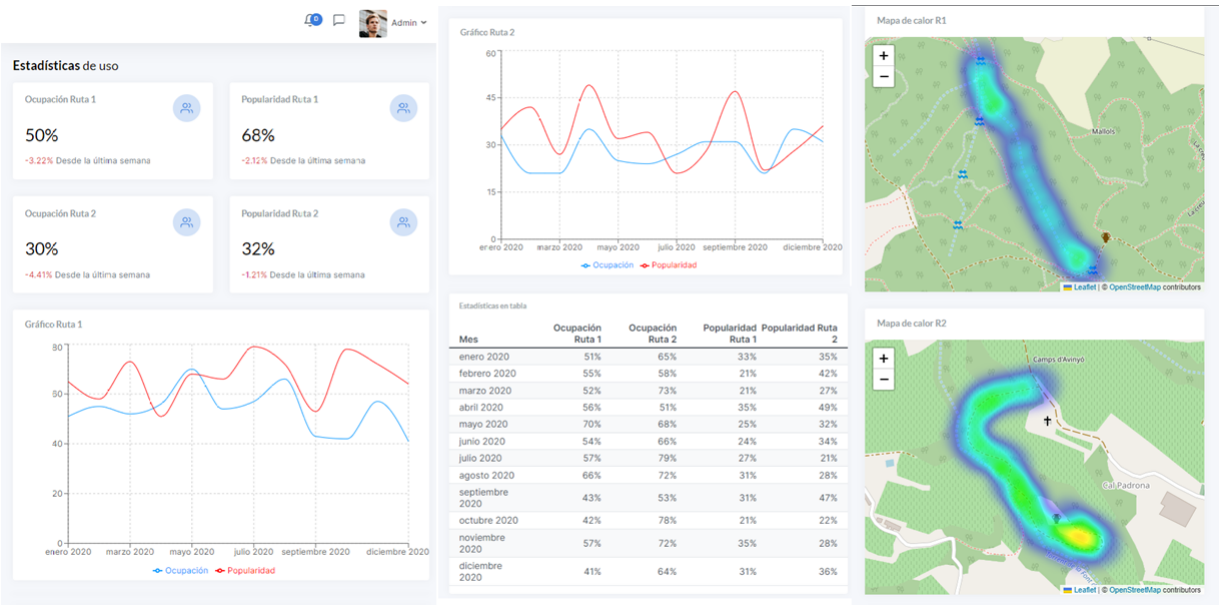


Fig. 13: Prototipo funcional.

A.3. Vista del prototipo final



Mes	Ocupación Ruta 1	Ocupación Ruta 2	Popularidad Ruta 1	Popularidad Ruta 2
enero 2020	51%	65%	33%	35%
febrero 2020	55%	58%	21%	42%
marzo 2020	52%	73%	21%	27%
abril 2020	56%	51%	35%	49%
mayo 2020	70%	68%	25%	32%
junio 2020	54%	66%	24%	34%
julio 2020	57%	79%	27%	21%
agosto 2020	66%	72%	31%	28%
septiembre 2020	43%	53%	31%	47%
octubre 2020	42%	78%	21%	22%
noviembre 2020	57%	72%	35%	28%
diciembre 2020	41%	64%	31%	36%

Fig. 14: Prototipo final.