



UNIVERSIDAD DE MÁLAGA



## Graduado en Ingeniería del Software

UrbanNav Madrid: Una herramienta visual  
sobre la movilidad en Madrid

UrbanNav Madrid: A visual tool on mobility in  
Madrid

Realizado por  
Álvaro Yuste Moreno

Tutorizado por  
Eduardo Guzmán de los Riscos

Departamento  
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, (junio de 2024)



## ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

### GRADUADO EN INGENIERÍA DEL SOFTWARE

#### **UrbanNav Madrid: Una herramienta visual sobre la movilidad en Madrid**

#### **UrbanNav Madrid: A visual tool on mobility in Madrid**

Realizado por  
**Álvaro Yuste Moreno**

Tutorizado por  
**Eduardo Guzmán de los Riscos**

Departamento  
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA  
MÁLAGA, JUNIO DE 2024

Fecha defensa:  
septiembre de 2024

# Resumen

El objetivo principal de este trabajo de fin de grado es desarrollar una aplicación que proporcione información visual relativa a la movilidad en Madrid. Este enfoque puede ayudar a los ciudadanos a comprender las dinámicas de movimiento y a identificar las zonas más peligrosas, entre otros aspectos. Una ciudad como Madrid es particularmente adecuada para este tipo de análisis debido a su alta densidad de población. Todo esto se hará mostrando gráficas, tablas y mapas con los datos correspondientes.

Los datos empleados en la aplicación se han obtenido a partir del repositorio de Open Data que proporciona el Ayuntamiento de Madrid. Se trata de un conjunto de datos abiertos bastante extenso, del que se han utilizado solo aquellos relacionados con los accidentes, estacionamientos, tráfico, radares y multas.

Este trabajo abarca todo el proceso de creación de una aplicación web full-stack, desde el análisis de los datos y el desarrollo de los requisitos hasta las pruebas del sistema, incluyendo el propio desarrollo de la aplicación.

**Palabras clave:** Aplicación de movilidad de Madrid, React, Express, OpenStreetMap.

# Abstract

The main objective of this final degree project is to develop an application that provides visual information related to mobility in Madrid. This approach can help citizens understand movement dynamics and identify the most dangerous areas, among other aspects. A city like Madrid is particularly suitable for this type of analysis due to its high population density. All this will be done by showing graphs, tables and maps with the corresponding data.

The data used in the application has been obtained from the repository of Open Data provided by the Madrid City Council. It is a quite extensive open data collection, of which only those have been used related to accidents, parking, traffic, speed cameras and fines.

This work covers the entire process of creating a full-stack web application, from data analysis and requirements development to system testing, including the development of the application itself.

**Keywords:** Madrid mobility application, React, Express, OpenStreet-Map.



# Índice

<b>1. Introducción</b>	<b>9</b>
1.1. Motivación . . . . .	9
1.2. Objetivos . . . . .	9
1.2.1. Accidentes . . . . .	10
1.2.2. Estacionamientos . . . . .	10
1.2.3. Radares . . . . .	10
1.2.4. Tráfico . . . . .	10
1.3. Metodología . . . . .	11
1.4. Aplicaciones similares . . . . .	11
1.4.1. Visualiza Madrid con datos abiertos . . . . .	11
1.5. Estructura del documento . . . . .	14
<b>2. Tecnologías y Herramientas</b>	<b>15</b>
2.1. Back-end . . . . .	15
2.2. Front-end . . . . .	16
2.3. Planificación . . . . .	19
2.4. Modelos y bocetos . . . . .	19
2.5. Control de versiones . . . . .	19
<b>3. Especificación y Análisis</b>	<b>21</b>
3.1. Análisis de datos . . . . .	21
3.1.1. Accidentes . . . . .	22
3.1.2. Estacionamientos . . . . .	25
3.1.3. Tráfico . . . . .	26
3.1.4. Radares . . . . .	28
3.1.5. Multas . . . . .	28
3.1.6. Distritos y barrios . . . . .	30
3.2. Requisitos funcionales . . . . .	30

3.3.	Casos de uso . . . . .	33
<b>4.</b>	<b>Diseño del Sistema</b>	<b>47</b>
4.1.	Modelo de datos . . . . .	47
4.2.	Diseño de la interfaz de usuario . . . . .	48
4.3.	Arquitectura de la aplicación . . . . .	53
<b>5.</b>	<b>Implementación</b>	<b>55</b>
5.1.	Implementación . . . . .	55
5.1.1.	Iteración 0: 1 - 22 de febrero de 2024 . . . . .	55
5.1.2.	Iteración 1: 23 de febrero - 1 de abril de 2024 . . . . .	59
5.1.3.	Iteración 2: 2 - 25 de abril de 2024 . . . . .	66
5.1.4.	Iteración 3: 26 de abril - 26 de mayo de 2024 . . . . .	70
5.1.5.	Iteración 4: 27 de mayo - 17 de junio de 2024 . . . . .	77
<b>6.</b>	<b>Conclusiones y Líneas Futuras</b>	<b>81</b>
6.1.	Conclusiones . . . . .	81
6.2.	Líneas Futuras . . . . .	82
<b>Apéndice A. Manual de Usuario</b>		<b>87</b>
A.1.	Introducción . . . . .	87
A.2.	Navegación . . . . .	87
A.3.	Inicio . . . . .	87
A.4.	Accidentes . . . . .	87
A.4.1.	Mapa . . . . .	88
A.4.2.	Flujo de accidentes . . . . .	89
A.4.3.	Leyenda . . . . .	90
A.4.4.	Gráfica . . . . .	91
A.4.5.	Filtro . . . . .	91
A.5.	Estacionamientos . . . . .	93
A.5.1.	Mapa . . . . .	93
A.5.2.	Leyenda . . . . .	93
A.5.3.	Filtro . . . . .	94

A.6.	Tráfico	94
A.6.1.	Mapa	95
A.6.2.	Leyenda	95
A.6.3.	Flujo de tráfico	95
A.6.4.	Gráfica	96
A.6.5.	Filtro	96
A.7.	Infracciones	97
A.7.1.	Radares	97
A.7.2.	Multas	98

## Apéndice B. Manual de Instalación

101



# 1

# Introducción

## 1.1. Motivación

Desde los inicios de este proyecto, se concibió con la idea de que abarcara íntegramente el ciclo de vida del desarrollo de una aplicación web: desde el análisis de datos y la captura de requisitos, hasta la realización de pruebas exhaustivas del sistema, pasando por el desarrollo integral de los servicios y la interfaz de usuario. De esta manera se buscaba poner en práctica todo lo aprendido en durante los estudios de grado.

La idea de estudiar el conjunto de datos sobre la movilidad de Madrid y hacer una herramienta que los analizase y mostrara la información mediante mapas fue del tutor de TFG. La premisa fundamental de la herramienta es su simplicidad de uso: intentando buscar el objetivo de que cualquier individuo, independientemente de su experiencia técnica, pueda beneficiarse de ella. Ya sea para obtener información sobre el tráfico en áreas específicas durante intervalos de tiempo determinados, realizar conclusiones sobre los riesgos asociados a la conducción en una metrópoli densamente poblada como Madrid, o localizar radares de tráfico, la herramienta está diseñada con el fin de ser un recurso valioso y potente.

Es importante destacar que, dada la naturaleza evolutiva del desarrollo de software, el proyecto se concibe como una base sólida, centrada principalmente en la gestión de los datos, sobre la cual se pueden realizar mejoras continuas y añadir nuevas funcionalidades. Así, el resultado final, aunque consideramos que es robusto, está abierto a futuras mejoras.

## 1.2. Objetivos

El propósito principal de este proyecto es desarrollar una aplicación web que funcione como herramienta analítica para visualizar diversos conjuntos de datos sobre la movilidad en Madrid. Dada la extensión de los datos disponibles, se ha llevado a cabo un estudio exhaustivo

para seleccionar aquellos que ofrecen un equilibrio óptimo entre relevancia, utilidad y viabilidad de implementación dentro del tiempo asignado. Los conjuntos de datos seleccionados incluyen 4 áreas: accidentes, tráfico, estacionamientos y radares/multas, a partir de las cuales se establecen diferentes objetivos.

#### **1.2.1. Accidentes**

- Visualizar en un mapa las zonas con incidencia de accidentes, marcadas según la cantidad y gravedad de los mismos.
- Presentar estadísticas de accidentes en forma de gráficos, clasificados por edad de los involucrados, tipo de vehículo, clima, gravedad de la lesión, tipo de accidente, y tests de drogas y alcohol.
- Facilitar la búsqueda de accidentes específicos mediante filtros como fecha, hora, rango de edad, sexo, distrito, condiciones meteorológicas, tipo de vehículo y tipo de accidente.
- Permitir la selección de un periodo de tiempo específico para observar la frecuencia de accidentes por hora o día en un distrito o barrio determinado.

#### **1.2.2. Estacionamientos**

- Mostrar en el mapa todas las zonas de estacionamiento, diferenciadas por colores según el uso que se le deba dar al estacionamiento.
- Proveer de una función de búsqueda de zonas de estacionamiento por distrito, barrio, tipo de aparcamiento y color.

#### **1.2.3. Radares**

- Indicar la ubicación de todos los radares fijos en el mapa.

#### **1.2.4. Tráfico**

- Representar en el mapa las zonas según la densidad media de tráfico.
- Mostrar gráficos que ordenen los distritos o barrios por volumen de tráfico, de mayor a menor.

- Ofrecer filtros para visualizar la densidad de tráfico en el mapa y en los gráficos, basados en un mes específico, un día concreto o un rango de tiempo.
- Posibilitar la selección de un periodo de tiempo específico para analizar el flujo de tráfico en un distrito o estación de tráfico particular.

### **1.3. Metodología**

Para el desarrollo del proyecto se utilizará una metodología incremental iterativa. Este enfoque se basa en dividir el proceso de desarrollo en fases cíclicas llamadas iteraciones. Cada iteración representa un ciclo de trabajo en el que se implementan funcionalidades incrementales del producto. Esto permite una gran flexibilidad a la hora de hacer cambios en los requisitos. En principio se implementarán los requisitos básicos que doten a la web de la funcionalidad reducida. A lo largo del proyecto, se irán agregando funcionalidades adicionales. La decisión sobre cuándo y qué características añadir dependerá de los inconvenientes que surjan y del tiempo de desarrollo, que está limitado por la fecha de entrega.

### **1.4. Aplicaciones similares**

Aunque la mayoría de requisitos y casos de uso se han desarrollado sin basarse en aplicaciones ya creadas, es imprescindible comentar que el Ayuntamiento de Madrid dispone de una aplicación web que ya muestra información del conjunto de datos de diversas maneras [1].

#### **1.4.1. Visualiza Madrid con datos abiertos**

Esta plataforma ofrece una enorme cantidad de información. Este trabajo comparte entidades como los accidentes o los estacionamientos como se puede ver en la Figura 1.

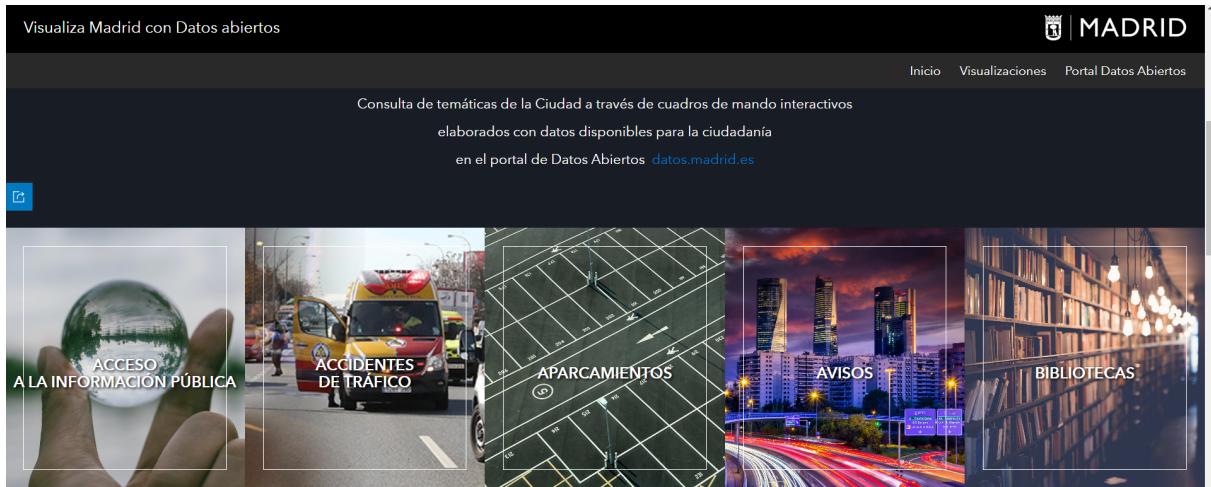


Figura 1: Página inicial de Visualizadatos Madrid.

Podemos ver que la información sobre los accidentes está detallada y estructurada de manera excelente. Hay puntos calientes en el mapa que indican el nivel de peligro en cada zona, y las gráficas están organizadas por años y distritos. Además se puede filtrar por diferentes características. Una decisión de diseño muy acertada ha sido dividir la información sobre los accidentes en diferentes páginas. Como se puede observar en el menú de navegación, se han separado los detalles específicos de los accidentes de los relacionados con las personas implicadas, tal y como se muestra en la Figura 2.

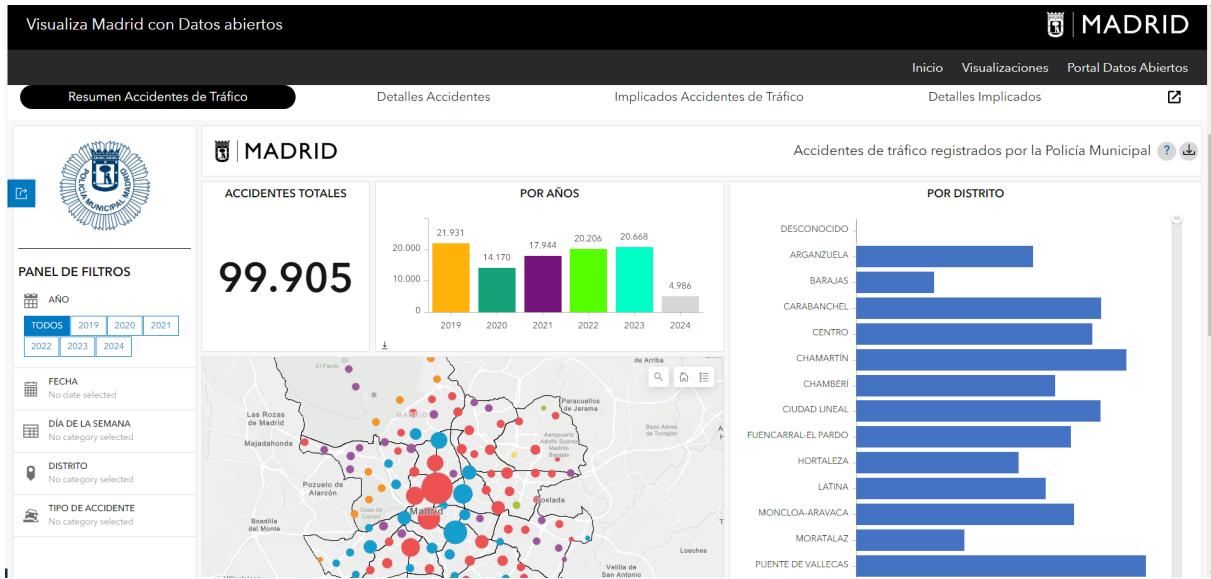


Figura 2: Apartado accidentes de Visualizadatos Madrid.

Respecto a los estacionamientos sucede algo similar: Esta vez la aplicación se centra más en la información del mapa y, al seleccionar una zona, se muestra una tabla con información específica como el número de plazas y la localización (figura 3).

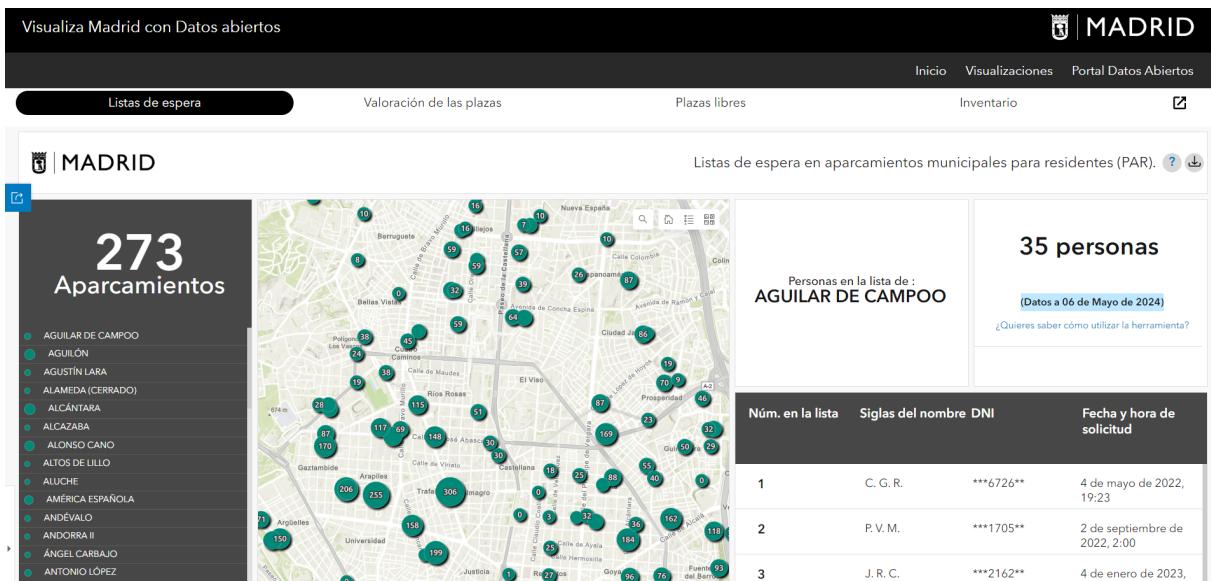


Figura 3: Apartado estacionamientos de Visualizadatos Madrid.

## 1.5. Estructura del documento

1. **Introducción:** Este capítulo proporciona una breve explicación de la motivación detrás del proyecto. También se establecen los objetivos iniciales, así como la metodología aplicada. Además, se hace referencia a otros trabajos que traten una temática similar.
2. **Tecnologías usadas:** Se describen los motivos detrás de las decisiones sobre qué tecnologías se han usado. Además se explica brevemente las características de cada una.
3. **Especificación y análisis:** Proceso inicial de captura de requisitos y análisis de los datos. Además se esboza el propósito que se les dará a estos datos.
4. **Diseño del sistema:** En este capítulo se explica el diseño del modelo relacional para la base de datos, de la interfaz de usuario y la arquitectura de la aplicación.
5. **Implementación y pruebas:** Este capítulo se centran en describir todo el proceso de implementación a lo largo de las iteraciones. También incluye capturas de algunas pruebas realizadas.
6. **Conclusiones y líneas futuras:** Se trata de una breve reflexión sobre el resultado del proyecto. Además se incluyen posibles mejoras y funcionalidades nuevas que no se han realizado por falta de tiempo.
7. **Apéndices:** Esta parte final engloba una serie de documentos técnicos adicionales como pueden ser el Manual de Usuario o el de Instalación.

# 2

# Tecnologías y Herramientas

En este capítulo se hablará resumidamente de las tecnologías (lenguajes de programación, frameworks, sistemas gestores de bases de datos, etc.) usadas en el desarrollo del proyecto.

## 2.1. Back-end

A continuación se mencionarán las principales tecnologías utilizadas tanto en el servidor como en la base de datos.

El servidor se ha desarrollado utilizando Express.js [2], un framework flexible que proporciona una capa de enrutamiento y manejo de peticiones y respuestas. Este framework emplea JavaScript como lenguaje de programación nativo.

Para la conexión con la base de datos y la creación de modelos, se ha utilizado la librería Sequelize [3]. Sequelize es un ORM (Object-Relational Mapping) robusto para Node.js, que facilita la interacción con bases de datos relaciones como MySQL al permitir la definición de modelos y asociaciones y realizar consultas complejas.

Para el análisis y conversión de datos geoespaciales, han sido de gran ayuda las librerías PROJ [4] y Turf. PROJ es una biblioteca de software ampliamente utilizada para la transformación de coordenadas geoespaciales. Por otro lado, Turf es una potente biblioteca JavaScript que proporciona una amplia gama de funciones para operaciones geoespaciales.

Desde el inicio del proyecto, se optó por implementar una base de datos relacional como MySQL [5], con el fin de optimizar la eficiencia en el manejo de las peticiones y aprovechar las relaciones entre las entidades. Para la gestión y manipulación de la base de datos, se ha utilizado el IDE MySQL Workbench, que ofrece una interfaz gráfica intuitiva y herramientas avanzadas para el diseño, desarrollo y administración de bases de datos MySQL.

## 2.2. Front-end

Para el desarrollo del frontend, se optó por utilizar la biblioteca React [6] debido a su amplia comunidad y su capacidad para construir interfaces de usuario dinámicas y eficientes. React se basa en la programación orientada a componentes, lo que permite crear interfaces de usuario complejas a partir de componentes reutilizables más pequeños. Esto facilita el mantenimiento y la organización del código, ya que los componentes se pueden encapsular y reutilizar en diferentes partes de la aplicación. En cuanto al manejo del estado React utiliza un sistema de estado unidireccional para administrar los datos de la aplicación. Esto significa que los datos solo fluyen en una sola dirección, desde el componente padre hacia los componentes hijos, lo que facilita el seguimiento y la predicción del estado de la aplicación. Esta famosa librería proporciona herramientas como useState o useEffect para gestionar el estado de forma eficiente. Además, React ofrece una enorme variedad de componentes y librerías creadas por la comunidad, lo que facilita el desarrollo de la interfaz de usuario y la integración de características complejas de manera rápida y sencilla. Para terminar, una gran ventaja de usar React es la herramienta de inspección React Developer Tools, una extensión de navegador que permite visualizar el árbol de componentes y el estado para tareas de depuración.

En cuanto a la visualización de mapas, se ha utilizado React-Leaflet [?], una biblioteca que proporciona componentes de React para integrar mapas interactivos en aplicaciones web.

Para el diseño de la interfaz de usuario, se combinó React con React-Bootstrap [7], una biblioteca que integra el popular framework de diseño Bootstrap con React. Esta combinación ha facilitado la creación de una interfaz moderna y adaptativa (*responsive*), aprovechando los estilos y componentes predefinidos de Bootstrap y adaptándolos fácilmente a las necesidades específicas del proyecto.

En lo que respecta a la generación de gráficos, se ha utilizado la biblioteca Recharts [8], una librería de código abierto para la creación de gráficos y diagramas en aplicaciones React. Está construida sobre componentes React y utiliza submódulos de D3 para renderizar módulos SVG de alto rendimiento. Recharts está diseñada específicamente para trabajar con React. Se integra a la perfección con la sintaxis y los componentes de React, lo que facilita la creación de gráficos interactivos y dinámicos dentro de React. Además ofrece gran flexibilidad para personalizar los gráficos ajustando las propiedades de los componentes e insertando componentes

personalizados.

En las etapas iniciales del desarrollo de la aplicación, se utilizaron exclusivamente los hooks de React para la gestión del estado local. No obstante, a medida que el proyecto avanzaba y se hacía más complejo, se identificó la necesidad de refactorizar el código para administrar ciertas variables de estado a través de Redux, optimizando así la gestión del estado global. Redux [9] es una biblioteca de código abierto que facilita la gestión del estado en aplicaciones JavaScript. Inspirada en el patrón arquitectónico Flux, Redux promulga un flujo de datos unidireccional, lo que contribuye a una estructura más predecible y mantenible de las aplicaciones. La principal utilidad de Redux radica en su capacidad para manejar el estado global, es decir, aquellos datos que deben estar disponibles en cualquier componente de la aplicación, independientemente de la jerarquía.

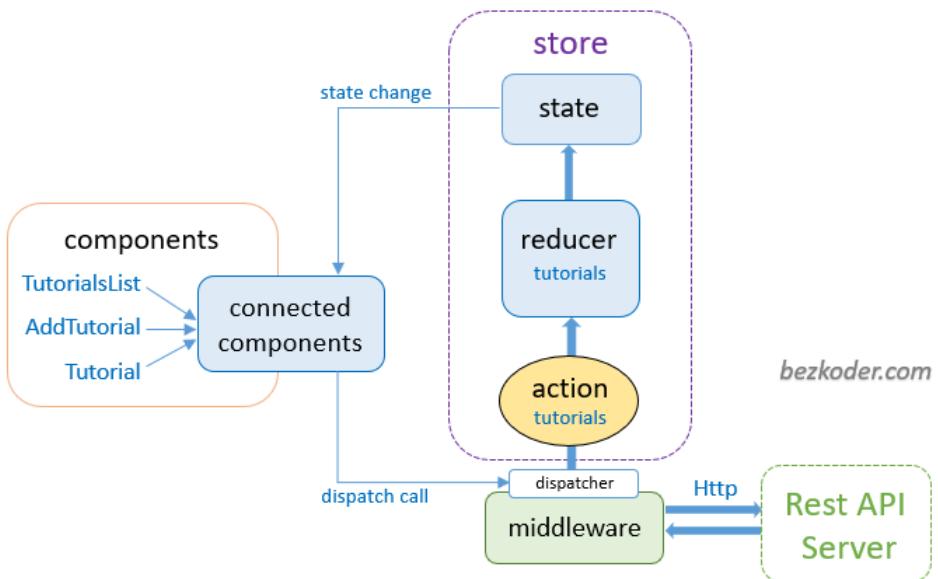


Figura 4: Elementos Redux

La centralización del estado global proporcionada por Redux simplifica la sincronización de datos y previene inconsistencias, resultando en un código más robusto y comprensible. La implementación de Redux se articula en torno a tres elementos fundamentales, como se ilustra en la Figura 4, obtenida de [10]:

- **Store:** Es el contenedor del estado global de la aplicación, actuando como la única fuente de verdad.

- Actions: Definen los eventos que describen las transiciones de estado requeridas, especificando el tipo de acción a realizar.
- Reducers: Son funciones puras que reciben el estado anterior y una acción, y devuelven un nuevo estado, sin mutar el original.

Para entender cómo funciona Redux nos hemos basado en la información proporcionada por la siguiente referencia [11], tal y como explicamos a continuación: Al procesar una acción, se invoca al reducer correspondiente, que actualiza el estado de manera inmutable. Los componentes suscritos al store son notificados automáticamente sobre el estado actualizado, garantizando así una reactividad eficiente. La librería react-redux proporciona una serie de hooks que facilitan la integración de Redux en aplicaciones React:

- useSelector: Permite a los componentes ‘suscribirse’ a cambios específicos en el estado global. La función selector se encarga de seleccionar y, si es necesario, transformar la porción del estado que el componente requiere.
- useDispatch: Este hook permite, como se ha mencionado antes, despachar una acción permitiendo así la actualización del estado global.

Para implementar Redux en la aplicación se ha hecho uso de la librería Redux Toolkit [12] que es un conjunto de herramientas que facilita la implementación de Redux escribiendo código más eficiente y acelerando el proceso de desarrollo.

En el desarrollo de la aplicación, se ha seleccionado Axios [13] para realizar peticiones HTTP al servidor de Express.js. Axios se destaca por su eficiente manejo de errores tanto de red como del lado del servidor. Además, Axios se basa en el uso de promesas para la gestión asíncrona de las peticiones HTTP, lo que permite escribir un código más limpio y fácil de seguir, alineado con las prácticas modernas de programación asíncrona.

Dado que ciertas acciones dentro de la aplicación requieren modificar el estado global tras la realización de operaciones asíncronas, se ha integrado Redux Thunk [14] como middleware en el ecosistema de Redux. Redux Thunk extiende las capacidades de Redux, permitiendo que las acciones puedan encapsular procesos asíncronos. Esto significa que Thunk puede interceptar las acciones despachadas, analizarlas y permitir que se realicen operaciones asíncronas si es necesario antes de que la acción se envíe al store.

### **2.3. Planificación**

Con el fin de alcanzar la meta de finalizar las funcionalidades principales para el plazo establecido, se ha optado por Google Calendar como instrumento clave de planificación. Esta herramienta ha sido esencial para programar tareas en el intervalo deseado y asignar una descripción detallada a cada una de ellas. Gracias a esta organización, se ha logrado una distribución efectiva del trabajo en iteraciones y se ha facilitado la implementación de una metodología iterativa incremental. El desarrollo del proyecto ha abarcado un total de cinco iteraciones, con una duración promedio de tres semanas.

### **2.4. Modelos y bocetos**

Para la elaboración del modelo relacional, se ha optado por Visual Paradigm, una herramienta integral de modelado UML que destaca por su versatilidad y funcionalidad completa. Visual Paradigm facilita la creación de una variedad de diagramas UML, incluyendo casos de uso, clases, actividades, componentes y secuencias. En particular, se ha empleado para desarrollar un detallado diagrama de clases.

En cuanto a los bocetos de la interfaz de usuario, se ha utilizado Balsamiq, una herramienta especializada en la creación de mockups o prototipos de interfaces de manera eficiente e intuitiva. Balsamiq proporciona una extensa biblioteca de elementos de interfaz de usuario predefinidos, lo que permite simular la experiencia del usuario final y facilitar la iteración del diseño de la interfaz.

Para el diagrama de la arquitectura de la aplicación web, se ha seleccionado Miro, una plataforma colaborativa que ofrece una gran flexibilidad para la visualización de ideas y la creación de diagramas de diversos tipos. Miro ha permitido diseñar un diagrama arquitectónico exhaustivo y descriptivo de la aplicación web, simplificando la representación de su estructura y componentes.

### **2.5. Control de versiones**

Para documentar y gestionar las distintas etapas del desarrollo del proyecto, se ha utilizado GitHub, una plataforma líder para el alojamiento de proyectos que emplea el sistema de control de versiones Git. Esta herramienta no solo facilita la colaboración y el seguimiento

to del progreso, sino que también permite asignar etiquetas a cada versión, lo que mejora la organización y la identificación de los distintos estados del proyecto.

La funcionalidad de reversión de versiones que ofrece GitHub ha sido fundamental para mantener la integridad del proyecto. En ocasiones donde se han introducido errores durante el desarrollo, esta característica ha permitido restaurar rápidamente el estado anterior del proyecto.

# 3

# Especificación y Análisis

## 3.1. Análisis de datos

Antes de adentrarnos en la descripción detallada de los datos, es esencial realizar un análisis de la plataforma que los alberga y las razones detrás de la selección de estos específicos conjuntos de datos.

El Portal de Datos Abiertos del Ayuntamiento de Madrid [15] ofrece un amplio espectro de información, abarcando desde datos relacionados con bicicletas eléctricas hasta detalles sobre paradas, horarios y líneas de los autobuses de la EMT. Además, incluye información sobre patinetes eléctricos y operaciones de retirada de vehículos por la grúa municipal.

Para el desarrollo de este proyecto, se han escogido aquellos datos que presentan una mayor relevancia y utilidad para los ciudadanos, como la ubicación de radares de tráfico y zonas de estacionamiento público. Este enfoque no solo busca facilitar la vida cotidiana de los madrileños, sino también proporcionar un análisis profundo de la movilidad urbana, poniendo especial énfasis en aspectos críticos como los accidentes y el aforo del tráfico.

El objetivo es mostrar los distritos y barrios que presentan mayores desafíos en términos de seguridad y congestión vial. En una etapa avanzada del proyecto, se tomó la decisión de incorporar la entidad de las multas de tráfico, vinculándolas directamente con los radares correspondientes. De esta forma, se logra una asociación directa entre las infracciones por exceso de velocidad y los dispositivos que las detectan.

### 3.1.1. Accidentes

La entidad ‘Accidentes’ es una de las más completas y detalladas, proporcionando un registro individual para cada persona involucrada en un incidente vial, ya sean conductores, pasajeros, peatones o testigos. Esta riqueza de información permite no solo documentar el accidente en sí, sino también ofrecer una visión comprensiva sobre los afectados.

Los datos disponibles en el Open Data son provisionales y se consolidan seis meses después de finalizar el año correspondiente. Una limitación notable de la plataforma es la ausencia de datos desglosados por barrio, lo cual motiva el enfoque de nuestra aplicación, centrado en correlacionar las zonas geográficas (barrios y distritos) con las entidades de datos.

Existen dos conjuntos de datos principales sobre accidentes de tráfico en el Open Data: el general y otro específico de accidentes con bicicletas. Dado que uno de los atributos de un accidente es el tipo de vehículo implicado, se ha tomado la decisión de fusionar ambos conjuntos. En el caso de los accidentes de bicicletas, se especificará claramente que este es el vehículo involucrado. Ambos archivos disponen de una descripción similar de los accidentes.

La actualización de los datos es mensual, y cada archivo contiene la información de un año completo. Los formatos disponibles son CSV y XLSX, siendo el CSV el seleccionado para la lectura de los datos en el servidor. Los enlaces a los conjuntos de datos y los pdf que describen los mismos [16] se pueden encontrar en la sección de Referencias,

La profundidad de detalles ofrecidos por la entidad ‘Accidentes’ nos permite implementar un sistema de filtrado avanzado, capaz de mostrar conjuntos de datos altamente específicos y relevantes para los usuarios.

Los campos disponibles en el archivo de accidentes de tráfico son los siguientes:

- **fecha:** Fecha en formato dd/mm/aaaa.
- **hora:** La hora se establece en rangos horarios de una hora.
- **localizacion:** Calle/cruce, puede venir con un numero.
- **numero:** Número de la calle. En caso de que tenga sentido.
- **cod\_distrito:** Código del distrito.
- **distrito:** Nombre del distrito.

■ **tipo\_accidente:**

- Colisión doble: Accidente de tráfico ocurrido entre dos vehículos en movimiento, (colisión frontal, fronto lateral, lateral)
- Colisión múltiple: Accidente de tráfico ocurrido entre más de dos vehículos en movimiento.
- Alcance: Accidente que se produce cuando un vehículo circulando o detenido por las circunstancias del tráfico es golpeado en su parte posterior por otro vehículo.
- Choque contra obstáculo fijo: Accidente ocurrido entre un vehículo en movimiento con conductor y un objeto inmóvil que ocupa la vía o zona apartada de la misma, ya sea vehículo estacionado, árbol, farola, etc.
- Atropello a persona: Accidente ocurrido entre un vehículo y un peatón que ocupa la calzada o que transita por aceras, refugios, paseos o zonas de la vía pública no destinada a la circulación de vehículos.
- Vuelco: Accidente sufrido por un vehículo con más de dos ruedas y que por alguna circunstancia sus neumáticos pierden el contacto con la calzada quedando apoyado sobre un costado o sobre el techo.
- Caída: Se agrupan todas las caídas relacionadas con el desarrollo y las circunstancias del tráfico, (motocicleta, ciclomotor, bicicleta, viajero bus, etc.,)
- Otras causas: Recoge los accidentes por atropello a animal, despeñamiento, salida de la vía, y otros.

■ **estado\_meteorológico:** Estado del clima en el momento del accidente. Pudiendo ser: Despejado, Nublado, Lluvia débil, Lluvia intensa o Granizando.

■ **tipo\_vehiculo:** Todo terreno, Turismo, Motocicleta hasta 125cc, Furgoneta, Vehículo articulado, Autobús, Camión rígido, Ciclomotor, Tractocamión, Motocicleta >125cc, Bicicleta, Otros vehículos con motor, Bicicleta EPAC (pedaleo asistido), Maquinaria de obras, VMU eléctrico.

■ **tipo\_persona:** Conductor, Pasajero, Peatón, Testigo.

- **rango\_edad:** Cadena de caracteres con el formato Desde x a y años.
- **sexo:** Puede ser: Hombre, Mujer o no asignado.
- **cod\_lesividad:** Relacionado con el siguiente atributo. Identifica la lesividad mediante un código numérico.
- **lesividad:** Descripción de los daños sufridos por la persona.

Cuadro 1: Descripción de los niveles de lesividad en accidentes de tráfico.

Código	Descripción	Gravedad
01	Atención en urgencias sin posterior ingreso	LEVE
02	Ingreso inferior o igual a 24 horas	LEVE
03	Ingreso superior o igual a 24 horas	GRAVE
04	Fallecido 24 horas	FALLECIDO
05	Asistencia sanitaria ambulatoria con posterioridad	LEVE
06	Asistencia sanitaria inmediata en centro de salud o mutua	LEVE
07	Asistencia sanitaria sólo en el lugar del accidente	LEVE
14	Sin asistencia sanitaria	NINGUNA
77	Se desconoce	DESCONOCIDA

- **coordenada\_x\_utm:** Coordenada X de la ubicación en el sistema de proyección cartográfica UTM (*Universal Transverse Mercator*).
- **coordenada\_y\_utm:** Coordenada Y de la ubicación en el mismo sistema.
- **positiva\_alcohol:** Puede ser N (no/negativo) o S (si/positivo).
- **positiva\_droga:** Puede ser NULL (negativo) o 1 (positivo).

Se realizarán las siguientes transformaciones sobre estos datos:

1. **Rango edad:** Será un entero con el valor del mínimo del rango. Por ejemplo si el rango es 'De 55 a 59 años' tendrá el valor 55.
2. **Coordenadas:** Se transformarán a coordenadas geográficas.

3. **Se eliminarán los atributos:** Localizacion, numero, cod\_distrito, distrito. Esto es porque se asociarán los accidentes al barrio correspondiente.

### 3.1.2. Estacionamientos

La funcionalidad relacionada con los estacionamientos es percibida como una de las más valiosas dentro de la aplicación, especialmente en una metrópoli con la intensidad de tráfico de Madrid. La capacidad de identificar rápidamente áreas disponibles para el estacionamiento se convierte en una herramienta indispensable para los conductores. En un contexto urbano donde el espacio para aparcar es un recurso escaso, esta aplicación facilita la localización de aparcamientos públicos, aliviando la tediosa búsqueda a la que a menudo se enfrentan los usuarios.

La selección de estos datos específicos para la aplicación no es casual; responde a una necesidad palpable de eficiencia y optimización del tiempo en la vida cotidiana de los ciudadanos. Al proporcionar información actualizada y precisa sobre las opciones de estacionamiento, la aplicación no solo mejora la experiencia de conducción sino que también contribuye a una mejor gestión del flujo vehicular en la ciudad.

El **Open Data** del Ayuntamiento de Madrid proporciona acceso público a varios conjuntos de datos relacionados con los estacionamientos [17], que se actualizan con distintas frecuencias:

- **Servicio de Estacionamiento Regulado (SER)** : Actualización trimestral con archivos anuales.
- **Plazas para personas con movilidad reducida** : Actualización anual en un archivo único.
- **Reservas para motocicletas** : Actualización semestral en un archivo único.
- **Zonas de carga y descarga** : Actualización anual con archivos anuales.

Para este proyecto, se han seleccionado y codificado los siguientes datos:

- **gis\_x**: Coordenada X en proyección UTM.
- **gis\_y**: Coordenada Y en proyección UTM.

- **color:** Clasificación de estacionamientos en el conjunto de datos SER, extendida a otros conjuntos:
  - Verde: Uso residencial.
  - Azul: Uso rotacional.
  - Naranja: Larga distancia.
  - Rojo: Ámbito sanitario.
  - Amarillo: Personas con movilidad reducida.
  - Negro: Reserva para motos.
  - Morado: Carga y descarga.
- **linea\_bateria:** Orientación del estacionamiento según la calle.
- **num\_plazas:** Número de plazas disponibles.

### 3.1.3. Tráfico

La inclusión de datos de tráfico en la aplicación responde a la necesidad de efectuar un análisis exhaustivo de la densidad de tráfico. Esta información resulta esencial para los ciudadanos que buscan optimizar sus desplazamientos urbanos, identificando áreas de alta congestión para evitarlas en momentos críticos del año o durante horas pico, facilitando así una movilidad más eficiente. Además, el análisis detallado del flujo de tráfico proporciona una perspectiva valiosa sobre las variaciones temporales de la circulación, permitiendo a los usuarios anticipar y planificar sus rutas con base en las tendencias observadas. Se hace posible discernir las horas de menor actividad vehicular y los períodos del año con menor afluencia, contribuyendo a una mejor gestión del tiempo a la población urbana.

Dentro de la plataforma [18], se encuentran diversos conjuntos de datos asociados al tráfico vehicular. Se ha seleccionado el conjunto denominado '*Aforos de tráfico de la Ciudad de Madrid permanentes*', que recopila información de tráfico registrada en 60 estaciones permanentes distribuidas por la ciudad. Los datos se actualizan mensualmente y cada archivo comprende la información de tráfico de un año completo. Adicionalmente, se dispone de otro archivo en formatos CSV y XLSX titulado '*Ubicación de estaciones permanentes y sentido de calles*', que complementa la información de aforos.

## **Aforos de tráfico permanentes en la Ciudad de Madrid**

- **FDIA:** Fecha del registro.
- **FEST:** Identificador numérico de la estación.
- **FSEN:** Sentido del tráfico, con los siguientes posibles valores:
  - 1- Sentido 1 (de 1:00 a 12:00).
  - 1= Sentido 1 (de 13:00 a 24:00).
  - 2- Sentido 2 (de 1:00 a 12:00).
  - 2= Sentido 2 (de 13:00 a 24:00).
  - La ausencia de valor indica un único sentido de circulación.

- **HOR1, HOR2, ..., HOR12:** Datos horarios del tráfico.

## **Ubicación de estaciones permanentes y sentido de calles**

- **Estación:** Número de la estación.
- **Nombre:** Denominación de la vía donde se ubica la estación.
- **Latitud:** Coordenada geográfica X.
- **Longitud:** Coordenada geográfica Y.
- **Sentido:** Dirección de la circulación (1, 2 o vacío).
- **Orient:** Orientación de la calle (N-S, S-N, E-O, O-E o vacío).

## **Selección y Estructuración de Datos para la Entidad 'Tráfico'**

Se ha optado por integrar en una única entidad la información de las estaciones y los datos de tráfico. Alternativamente, se podría haber estructurado en dos entidades relacionadas mediante una clave foránea en una relación de uno a muchos.

- **Fecha:** Día del registro.
- **Lat:** Latitud de la estación.

- **Lon:** Longitud de la estación.
- **Nombre:** Nombre de la vía.
- **Trafico:** Arreglo de 24 elementos representando el tráfico por hora.
- **Orientación:** Dirección de la calle (Norte-Sur, Sur-Norte, Este-Oeste, Oeste-Este).

### 3.1.4. Radares

Este es un simple archivo en formato CSV, XLSX y GEO con información sobre los radares fijos de Madrid [19]. Esta ha sido la selección de datos para la aplicación:

- **Nº RADAR:** Un número que sirve como identificador del radar.
- **Ubicación:** Proporciona una descripción detallada de la carretera donde se encuentra el radar. Esta información incluye detalles específicos sobre la zona concreta, como intersecciones, puntos de referencia o características geográficas relevantes.
- **Sentido:** Indica el sentido de la carretera que está bajo la vigilancia del radar.
- **Tipo:** Especifica el carril de la carretera al que está dirigido el radar.
- **PK:** PK se ha asumido que significa Punto Kilométrico ya que no hay una descripción de este conjunto de datos en la plataforma. Aunque un PK puede abarcar varios radares que controlen un mismo tramo de carretera, su uso es crucial para asociar los radares a las multas correspondientes. En etapas posteriores del proceso, esta información será valiosa para gestionar las sanciones de manera eficiente.

### 3.1.5. Multas

En una fase avanzada del proyecto, se tomó la decisión estratégica de incorporar información detallada sobre las multas de tráfico. Esta adición confiere una mayor utilidad a los radares, ya que permite establecer una relación directa entre las infracciones por exceso de velocidad y los dispositivos que las capturaron. Además, la aplicación podría tener la capacidad de mostrar información relativa a otras infracciones, lo que facilita la búsqueda exhaustiva de detalles específicos por parte de los usuarios.

Dentro del conjunto de datos disponibles en el Open Data, se identificaron dos opciones relacionadas con las multas: *Multas de circulación* y *Multas de circulación: detalle*. El primero ofrece una visión general que abarca aspectos como la cantidad recaudada por el ayuntamiento, el número de denuncias tramitadas y pagadas, todo ello desglosado por meses durante los últimos 10 o 15 años. Además, proporciona información sobre los procedimientos y la gestión del proceso, incluyendo el tiempo medio de atención en ventanilla y el tiempo medio de espera.

Sin embargo, se optó por el segundo conjunto de datos [20], *Multas de circulación: detalle*, debido a su nivel de detalle. Este conjunto ofrece información específica sobre cada multa, incluyendo la naturaleza de la infracción, el importe de la sanción y si se aplicó algún descuento por pago temprano. Además, su estructura permite establecer una relación directa con los radares, como se mencionó previamente. Por todas estas razones, se consideró más apropiado utilizar *Multas de circulación: detalle*.

Cabe destacar que este conjunto de datos se actualiza mensualmente, y cada archivo CSV agrupa todas las multas correspondientes a un mes específico. Se han seleccionado todos los atributos disponibles para garantizar una visión completa y detallada de las infracciones.

- **Calificación:** Expresa la calificación de la infracción denunciada en función de su gravedad; leve, grave o muy grave, conforme determina el RDL 6/2015, de 30 de octubre, por el que se aprueba el Texto Refundido de la Ley sobre Tráfico, Circulación de Vehículos y Seguridad Vial (LSV).
- **Lugar:** Ubicación del viario público donde ha tenido lugar la infracción denunciada.
- **Mes:** Mes del año en que ha sido formulada la denuncia.
- **Anio:** Año en que ha sido formulada la denuncia.
- **Hora:** Hora y minutos en que ha sido formulada la denuncia.
- **Imp. Bol:** Refleja el importe en euros correspondiente a la multa por la infracción cometida en función de su calificación.
- **Descuento:** Posibles valores: SI/NO. Indica si el tipo de infracción permite el pago de la multa con reducción del 50 % conforme a la LSV para acogerse al procedimiento abreviado.

- **Puntos:** Si la infracción cometida implica, conforme a la LSV, la detacción de puntos en el permiso de conducción del infractor, aparecerán los puntos a detraer.
- **Denunciante:** Proporciona información sobre la procedencia de la denuncia (Servicio de Estacionamiento Regulado, Agentes de Movilidad, Policía Municipal, Servicio de Apoyo al Control de Estacionamiento, etc).
- **Hecho-Bol:** Descripción de la infracción cometida.
- **Vel. límite:** En aquellos casos en que la infracción es un exceso de velocidad, refleja el límite de la vía donde la infracción fue detectada.
- **Vel. Circula:** En aquellos casos en que la infracción es un exceso de velocidad, muestra el límite de velocidad medido por el radar.
- **Coordenada X:** En aquellos casos en que consta indicado muestra la coordenada X del lugar de la denuncia.
- **Coordenada Y:** En aquellos casos en que consta indicado muestra la coordenada Y del lugar de la denuncia.

### 3.1.6. Distritos y barrios

La información sobre los distritos y barrios fue extraída de las páginas web indicadas en las referencias [21] y [22]. A partir de estos datos, se han obtenido las coordenadas que definen los límites de cada área.

## 3.2. Requisitos funcionales

ID	Descripción
RF 1	<b>Visualización de accidentes</b>
RF 1.1	Mostrar mapa de accidentes con zonas marcadas: El sistema debe mostrar un mapa con zonas (tanto distritos como barrios) marcadas por colores, indicando el nivel de peligrosidad de la zona en función de las características de los accidentes registrados.

RF 1.2	Mostrar información sobre una zona: Al hacer clic en una zona del mapa, el sistema debe mostrar información detallada sobre dicha zona.
RF 1.3	Mostrar mapa con marcadores de accidentes: El sistema debe mostrar un marcador por cada accidente registrado.
RF 1.4	Mostrar información sobre un accidente: Al hacer clic en un marcador de accidente, el sistema debe mostrar información detallada sobre dicho accidente y las personas implicadas.
RF 1.5	Cambiar visualización del tipo de zona: El sistema debe permitir al usuario seleccionar el tipo de zona a visualizar en el mapa, pudiendo elegir entre barrios o distritos.
RF 1.6	Mostrar gráfica accidentes: El sistema debe mostrar una gráfica que clasifique los accidentes según diferentes atributos.
RF 1.7	Cambiar clasificación de la gráfica: El sistema debe permitir al usuario cambiar el atributo por el cual se clasifican los accidentes en la gráfica.
RF 1.8	Filtro accidentes: El sistema debe permitir al usuario filtrar los accidentes mostrados según diferentes atributos.
RF 1.9	Mostrar flujo de accidentes: El sistema debe permitir al usuario seleccionar una zona y un periodo de tiempo, y mostrar los accidentes ocurridos en dicha zona clasificados por unidades de tiempo, como horas o días.
<b>RF 2</b>	<b>Visualización de estacionamientos</b>
RF 2.1	Mostrar mapa de estacionamientos con zonas: El sistema debe mostrar un mapa con las zonas delimitadas en caso de que se muestren más de 500 estacionamientos.
RF 2.2	Mostrar estacionamientos de una zona: Al hacer clic sobre una zona, el sistema debe mostrar los estacionamientos de esa zona seleccionada.
RF 2.3	Mostrar mapa con marcadores de estacionamientos: El sistema debe mostrar un marcador por cada estacionamiento registrado.
RF 2.4	Clasificar estacionamientos por color: Cuando el sistema muestre los marcadores de los estacionamientos, el color del mismo dependerá del tipo de estacionamiento.

RF 2.5	Mostrar leyenda: El sistema debe mostrar una leyenda que explique la clasificación de estacionamientos por colores.
RF 2.6	Mostrar información sobre un estacionamiento: Al hacer clic en un marcador de estacionamiento, el sistema debe mostrar información detallada sobre dicho estacionamiento.
RF 2.7	Filtro de estacionamientos: El sistema debe permitir al usuario filtrar los estacionamientos mostrados según diferentes atributos.
<b>RF 3</b>	<b>Visualización del tráfico</b>
RF 3.1	Mostrar mapa de tráfico con zonas marcadas: El sistema debe mostrar un mapa con las zonas marcadas por colores, indicando la densidad del tráfico medio en función del periodo de tiempo seleccionado.
RF 3.2	Mostrar información sobre una zona: Al hacer clic o poner el ratón sobre una zona del mapa, el sistema debe mostrar información sobre la zona y la densidad del tráfico.
RF 3.3	Mostrar información sobre una estación: Al hacer clic sobre una estación de tráfico, el sistema debe mostrar información sobre la estación de tráfico y los datos que ha registrado.
RF 3.4	Mostrar gráfica de tráfico ordenado: El sistema debe mostrar una gráfica con las zonas clasificadas por su densidad de tráfico y ordenadas de mayor a menor.
RF 3.5	Seleccionar zona de la gráfica: Al hacer clic sobre una barra de la gráfica, el sistema debe mostrar un popup de la zona seleccionada en el mapa.
RF 3.6	Filtro de tráfico: El sistema debe permitir al usuario filtrar por períodos de tiempo para mostrar el tráfico medio de ese momento.
RF 3.7	Mostrar flujo de tráfico: El sistema debe permitir al usuario seleccionar una zona y un periodo de tiempo, y mostrar el tráfico de cada unidad de tiempo, pudiendo ser horas o días.
<b>RF 4</b>	<b>Visualización de radares</b>
RF 4.1	Mostrar mapa de radares con zonas delimitadas: El sistema debe mostrar un mapa con las zonas delimitadas en las que se encuentran los radares.

RF 4.2	Mostrar mapa con marcadores de radares: El sistema debe mostrar un marcador en el mapa por cada radar registrado.
RF 4.3	Mostrar información sobre un radar: Al hacer clic sobre un marcador de un radar, el sistema debe mostrar la opción de visualizar información detallada del radar.
RF 4.4	Mostrar gráfica de radares con multas: El sistema debe mostrar una gráfica que clasifique los radares según la cantidad de multas registradas.
RF 4.5	Seleccionar radar de la gráfica: Al hacer clic sobre una barra de la gráfica, el sistema debe mostrar un popup del radar seleccionado en el mapa y una tabla con todas las multas asociadas a ese radar.
RF 4.6	Mostrar tabla de multas de un radar: Al hacer clic sobre un marcador de un radar, el sistema debe mostrar la opción de visualizar una tabla con las multas asociadas a ese radar.
RF 4.7	Ordenar multas según los atributos: El sistema debe permitir al usuario ordenar las multas en la tabla en orden ascendente y descendente a partir de sus atributos.
RF 4.8	Filtro de multas asociadas a radares: El sistema debe permitir al usuario filtrar las multas mostradas según diferentes atributos.
<b>RF 5</b>	<b>Visualización de las multas</b>
RF 5.1	Mostrar tabla con multas: El sistema debe mostrar una tabla con todas las multas registradas.
RF 5.2	Filtrar multas: El sistema debe permitir al usuario filtrar las multas por diferentes atributos.
RF 5.3	Ordenar multas: El sistema debe permitir al usuario ordenar las multas por diferentes atributos.

### 3.3. Casos de uso

En las siguientes tablas de esta sección se muestran los casos de uso implementados en este TFG, relacionándolos a su vez con los requisitos correspondientes.

<b>ID</b>	CU1
<b>Caso de uso</b>	Ver información de accidentes de una zona
<b>Descripción</b>	Ver toda la información asociada a los accidentes ocurridos en una zona concreta.
<b>Precondición</b>	Estar en la sección de accidentes con los barrios o distritos seleccionados para que se muestren.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en una zona.</li> <li>2. El usuario selecciona la opción 'Ver más información'.</li> <li>3. Al usuario se le muestra una ventana modal con la información de la zona junto con algunos datos del conjunto de accidentes.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 1.1, RF 1.2, RF 1.5

<b>ID</b>	CU2
<b>Caso de uso</b>	Ver información de accidente concreto
<b>Descripción</b>	Ver todos los detalles de un accidente concreto y de las personas implicadas en él.
<b>Precondición</b>	Estar en la sección de accidentes y visualizar marcadores de accidentes.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en un marcador de accidente.</li> <li>2. El usuario selecciona la opción 'Ver más información'.</li> <li>3. Al usuario se le muestra una ventana modal con la información del accidente y de cada persona implicada en él.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 1.3, RF 1.4

<b>ID</b>	CU3
<b>Caso de uso</b>	Ver gráfica de accidentes
<b>Descripción</b>	Ver los accidentes clasificados por diferentes atributos.
<b>Precondición</b>	Estar en la sección de accidentes.
<b>Escenario principal</b>	<p>1. El usuario hace clic en el botón 'Mostrar Gráfica'.</p> <p>2. El sistema muestra una gráfica con los accidentes clasificados por un atributo concreto.</p> <p>3. El usuario selecciona un atributo diferente.</p> <p>4. El sistema muestra los accidentes clasificados por el nuevo atributo.</p> <p>5. El usuario pone el ratón sobre una barra de la gráfica y el sistema muestra el total de accidentes que cumplen esa característica concreta.</p>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 1.6, RF 1.7

<b>ID</b>	CU4
<b>Caso de uso</b>	Filtrar accidentes
<b>Descripción</b>	Filtrar los accidentes que se muestran en la gráfica y en el mapa por características concretas.
<b>Precondición</b>	Estar en la sección de accidentes.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón del filtro.</li> <li>2. El usuario selecciona los parámetros por los que quiere filtrar los accidentes y hace clic en el botón 'Filtrar'.</li> <li>3. El sistema muestra tanto en el mapa como en la gráfica los accidentes que cumplen el filtro aplicado.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>2. El usuario hace clic en el botón 'Filtrar' sin haber seleccionado ningún parámetro por el que filtrar.</li> <li>3. El sistema muestra un mensaje diciendo: 'Datos incompletos'.</li> </ol>
<b>ID requisitos asociados</b>	RF 1.8

<b>ID</b>	CU5
<b>Caso de uso</b>	Flujo de accidentes
<b>Descripción</b>	Visualizar el flujo de accidentes en una gráfica dentro de un periodo de tiempo específico.
<b>Precondición</b>	Estar en la sección de accidentes.
<b>Escenario principal</b>	<p>1. El usuario hace clic en una zona del mapa.</p> <p>2. El usuario selecciona la opción 'Mostrar flujo'.</p> <p>3. El sistema muestra una ventana modal con un formulario.</p> <p>4. El usuario rellena el formulario introduciendo un periodo de tiempo entre dos fechas y hace clic en el botón 'Enviar'.</p> <p>5. El usuario es redirigido a una ventana donde se muestra una gráfica con una barra por cada día que contiene la información de los accidentes ocurridos en ese día.</p>
<b>Escenario alternativo</b>	<p>4. El usuario rellena el formulario introduciendo un periodo de tiempo entre dos horas y hace clic en el botón 'Enviar'.</p> <p>5. El usuario es redirigido a una ventana donde se muestra una gráfica con una barra por cada hora que contiene la información de los accidentes ocurridos durante esa hora.</p>
<b>ID requisitos asociados</b>	RF 1.9

<b>ID</b>	CU6
<b>Caso de uso</b>	Ver estacionamientos de una zona
<b>Descripción</b>	Mostrar marcadores de estacionamientos de una zona concreta.
<b>Precondición</b>	Estar en la sección de estacionamientos con los barrios o distritos seleccionados para que se muestren.
<b>Escenario principal</b>	<p>1. El usuario hace clic en una zona.</p> <p>2. El usuario selecciona la opción 'Ver estacionamientos de este distrito/barrio'.</p> <p>3. El sistema muestra los marcadores de estacionamientos de la zona seleccionada. Cada marcador tendrá el color correspondiente al tipo de estacionamiento.</p> <p>4. Los colores de los iconos serán acordes a los colores mostrados en la leyenda.</p>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 2.1, RF 2.2, RF 2.3, RF 2.4, RF 2.5

<b>ID</b>	CU7
<b>Caso de uso</b>	Ver información de estacionamiento concreto
<b>Descripción</b>	Ver todos los detalles de un estacionamiento concreto.
<b>Precondición</b>	Estar en la sección de estacionamientos y que aparezcan en el mapa marcadores de estacionamiento.
<b>Escenario principal</b>	<p>1. El usuario hace clic en un marcador de estacionamiento.</p> <p>2. Al usuario se le muestra un popup en el marcador con información sobre el estacionamiento seleccionado.</p>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 2.6

<b>ID</b>	CU8
<b>Caso de uso</b>	Filtrar estacionamientos
<b>Descripción</b>	Filtrar los estacionamientos que se muestran en el mapa.
<b>Precondición</b>	Estar en la sección de estacionamientos.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón del filtro.</li> <li>2. El usuario selecciona los parámetros por los que quiere filtrar los estacionamientos y hace clic en el botón 'Filtrar'.</li> <li>3. El sistema muestra en el mapa los estacionamientos que cumplen el filtro aplicado.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>2. El usuario hace clic en el botón 'Filtrar' sin haber seleccionado ningún parámetro por el que filtrar.</li> <li>3. El sistema muestra un mensaje diciendo: 'Datos incompletos'.</li> </ol>
<b>ID requisitos asociados</b>	RF 2.7

<b>ID</b>	CU9
<b>Caso de uso</b>	Ver información de tráfico de una zona
<b>Descripción</b>	Ver toda la información asociada al tráfico durante un periodo de tiempo concreto en una zona específica.
<b>Precondición</b>	Estar en la sección de tráfico con los barrios o distritos seleccionados para que se muestren.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en una zona.</li> <li>2. El usuario selecciona la opción 'Ver más información'.</li> <li>3. Al usuario se le muestra una ventana modal con la información de la zona junto con datos relativos a la densidad de tráfico.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 3.1, RF 3.2

<b>ID</b>	CU10
<b>Caso de uso</b>	Ver información sobre una estación de tráfico
<b>Descripción</b>	Ver un resumen de los datos tomados por la estación de tráfico durante un periodo de tiempo concreto.
<b>Precondición</b>	Estar en la sección de tráfico con las estaciones seleccionadas para que se muestren.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el marcador de la estación de tráfico.</li> <li>2. El usuario selecciona la opción 'Ver más información'.</li> <li>3. Al usuario se le muestra una ventana modal con la información de los datos registrados por la estación de tráfico durante un periodo de tiempo concreto.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 3.3

<b>ID</b>	CU11
<b>Caso de uso</b>	Ver datos de tráfico en gráfica
<b>Descripción</b>	Visualizar la densidad de tráfico de cada zona en una gráfica ordenada de mayor a menor.
<b>Precondición</b>	Estar en la sección de tráfico.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón 'Mostrar Gráfica'.</li> <li>2. El sistema muestra una gráfica con la densidad de tráfico clasificada por distritos/barrios.</li> <li>3. El usuario hace clic en una barra de la gráfica.</li> <li>4. El sistema muestra un popup en el mapa marcando la zona que representa la barra de la gráfica con el nombre de la zona.</li> <li>5. El usuario pone el ratón sobre una barra de la gráfica y el sistema muestra el tráfico medio junto con el nombre de la zona.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 3.4, RF 3.5

<b>ID</b>	CU12
<b>Caso de uso</b>	Filtrar tráfico
<b>Descripción</b>	Filtrar el tráfico medio que se muestra en la gráfica y en el mapa para que abarque un periodo de tiempo concreto o un sentido de circulación específico.
<b>Precondición</b>	Estar en la sección de tráfico.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón del filtro.</li> <li>2. El usuario selecciona los parámetros por los que quiere filtrar el tráfico y hace clic en el botón 'Filtrar'.</li> <li>3. El sistema muestra tanto en el mapa como en la gráfica el tráfico medio durante el periodo de tiempo introducido.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>2. El usuario hace clic en el botón 'Filtrar' sin haber seleccionado ningún parámetro por el que filtrar.</li> <li>3. El sistema muestra un mensaje diciendo: 'Datos incompletos'.</li> </ol>
<b>ID requisitos asociados</b>	RF 3.6

<b>ID</b>	CU13
<b>Caso de uso</b>	Flujo de tráfico
<b>Descripción</b>	Visualizar el flujo de la densidad de tráfico de una zona o estación de tráfico en una gráfica dentro de un periodo de tiempo concreto.
<b>Precondición</b>	Estar en la sección de tráfico.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en una zona/estación del mapa.</li> <li>2. El usuario selecciona la opción 'Mostrar flujo'.</li> <li>3. El sistema muestra una ventana modal con un formulario.</li> <li>4. El usuario rellena el formulario introduciendo un periodo de tiempo entre dos fechas y hace clic en el botón 'Enviar'.</li> <li>5. El usuario es redirigido a una ventana donde se muestra una gráfica con una barra por cada día que contiene la información del tráfico de ese día.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>4. El usuario rellena el formulario introduciendo un periodo de tiempo entre dos horas y hace clic en el botón 'Enviar'.</li> <li>5. El usuario es redirigido a una ventana donde se muestra una gráfica con una barra por cada hora que contiene la información del tráfico durante esa hora.</li> </ol>
<b>ID requisitos asociados</b>	RF 3.7

<b>ID</b>	CU14
<b>Caso de uso</b>	Ver información de un radar de una zona
<b>Descripción</b>	Ver toda la información asociada a un radar y las multas que ha registrado en una zona específica.
<b>Precondición</b>	Estar en la sección de radares con los barrios o distritos seleccionados para que se muestren.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en una zona.</li> <li>2. El usuario selecciona la opción 'Ver más información del radar'.</li> <li>3. Al usuario se le muestra una ventana modal con la información del radar y las multas que ha registrado.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 4.1, RF 4.2, RF 4.3

<b>ID</b>	CU15
<b>Caso de uso</b>	Ver multas registradas por los radares en una gráfica
<b>Descripción</b>	Visualizar la cantidad de multas registradas por cada radar en una gráfica.
<b>Precondición</b>	Estar en la sección de radares.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón 'Mostrar Gráfica'.</li> <li>2. El sistema muestra una gráfica con la cantidad de multas asociadas a cada radar junto con la dirección del radar.</li> <li>3. El usuario hace clic en una barra de la gráfica.</li> <li>4. El sistema muestra un popup en el mapa marcando el radar que representa la barra de la gráfica.</li> <li>5. El sistema muestra una tabla con las multas registradas por ese radar.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 4.4, RF 4.5

<b>ID</b>	CU16
<b>Caso de uso</b>	Ver tabla de multas registradas por un radar
<b>Descripción</b>	Visualizar una tabla donde cada fila es una multa registrada por un radar.
<b>Precondición</b>	Estar en la sección de radares.
<b>Escenario principal</b>	<p>1. El usuario hace clic en un marcador de radar.</p> <p>2. El usuario selecciona la opción 'Mostrar multas'.</p> <p>3. Al usuario se le muestra una tabla con todas las características de cada multa asociada al radar que ha seleccionado.</p> <p>4. El usuario hace clic en la cabecera de una columna de la tabla.</p> <p>5. El sistema ordena en orden ascendente/descendente las multas por el atributo seleccionado.</p>
<b>Escenario alternativo</b>	2. El sistema no muestra al usuario la opción 'Mostrar multas' porque no hay multas registradas que estén asociadas a ese radar.
<b>ID requisitos asociados</b>	RF 4.6, RF 4.7

<b>ID</b>	CU17
<b>Caso de uso</b>	Filtrar multas registradas por los radares
<b>Descripción</b>	Filtrar las multas asociadas a los radares que se muestran en la gráfica y en la tabla.
<b>Precondición</b>	Estar en la sección de radares.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón del filtro.</li> <li>2. El usuario selecciona los parámetros por los que quiere filtrar las multas y hace clic en el botón 'Filtrar'.</li> <li>3. El sistema muestra tanto en el mapa como en la tabla las multas que cumplen el filtro aplicado.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>2. El usuario hace clic en el botón 'Filtrar' sin haber seleccionado ningún parámetro por el que filtrar.</li> <li>3. El sistema muestra un mensaje diciendo: 'Datos incompletos'.</li> </ol>
<b>ID requisitos asociados</b>	RF 4.8

<b>ID</b>	CU18
<b>Caso de uso</b>	Ordenar multas en la tabla
<b>Descripción</b>	Visualizar toda la información de cada multa registrada en la aplicación y ordenarlas en orden ascendente o descendente por atributos concretos.
<b>Precondición</b>	Estar en la sección de multas.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en la cabecera de una columna de la tabla para seleccionar una característica de las multas.</li> <li>2. El sistema ordena las multas de la tabla en orden ascendente/descendente por la característica seleccionada.</li> </ol>
<b>Escenario alternativo</b>	-
<b>ID requisitos asociados</b>	RF 5.1, RF 5.3

<b>ID</b>	CU19
<b>Caso de uso</b>	Filtrar multas
<b>Descripción</b>	Filtrar las multas que se muestran en la tabla por alguna característica concreta.
<b>Precondición</b>	Estar en la sección de multas.
<b>Escenario principal</b>	<ol style="list-style-type: none"> <li>1. El usuario hace clic en el botón del filtro.</li> <li>2. El usuario selecciona los parámetros por los que quiere filtrar las multas y hace clic en el botón 'Filtrar'.</li> <li>3. El sistema muestra en la tabla las multas que cumplen con el filtro aplicado.</li> </ol>
<b>Escenario alternativo</b>	<ol style="list-style-type: none"> <li>2. El usuario hace clic en el botón 'Filtrar' sin haber seleccionado ningún parámetro por el que filtrar.</li> <li>3. El sistema muestra un mensaje diciendo: 'Datos incompletos'.</li> </ol>
<b>ID requisitos asociados</b>	RF 5.2

# 4

# Diseño del Sistema

## 4.1. Modelo de datos

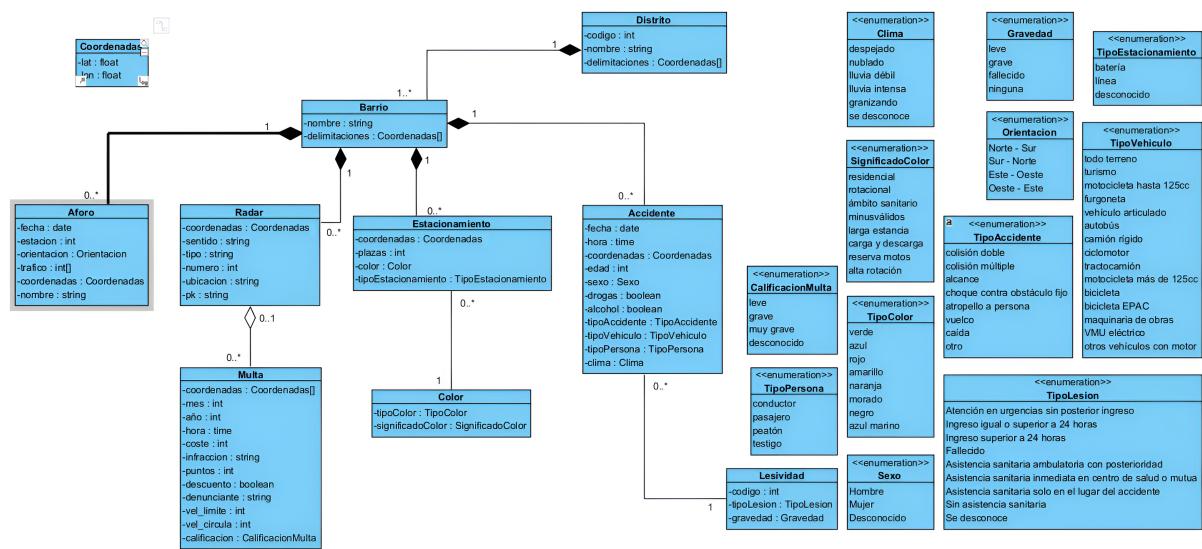


Figura 5: Modelo relacional

En la Figura 5 se muestra el diagrama UML del modelo relacional de la aplicación. El diseño de un modelo relacional robusto y flexible constituye la base sobre la cual se asienta la aplicación, permitiendo la realización de consultas eficientes sobre grandes volúmenes de datos y facilitando la integración de futuras modificaciones.

En este contexto, se ha optado por una estructura que define los atributos de valor constante como **enumeraciones**, lo que confiere al sistema una mayor coherencia y adaptabilidad. La vinculación de estos atributos a través de **claves foráneas** no solo refuerza la integridad referencial, sino que también es más eficiente en términos de memoria y comparación. Adicionalmente, de esta manera se minimizan los errores de entrada y se garantiza la validez de

los datos. Para abordar la aparición de datos desconocidos o nulos al importar información de fuentes externas, se ha incorporado un literal específico en la mayoría de las enumeraciones. Esta estrategia simplifica la lógica del código y preserva la integridad de la base de datos. Atributos complejos como ‘lesividad’ o ‘color’, que engloban múltiples conceptos como el tipo y la gravedad de las lesiones o el significado asociado a cada color, se han modelado como clases independientes.

En cuanto a las relaciones entre entidades, se ha establecido que cada instancia principal –ya sea ‘Accidente’, ‘Estacionamiento’, ‘Aforo’, ‘Radar’ o ‘Multas’– pertenezca exclusivamente a un único ‘Barrio’, aunque no todos los barrios contengan estas entidades. Esta característica justifica la elección de la **composición** como el tipo de relación más apropiado. Similarmente, cada ‘Distrito’ se compone de al menos un ‘Barrio’. Por otro lado, la relación entre ‘Multas’ y ‘Radar’ se ha modelado de manera que no todas las multas derivan de infracciones captadas por radares, reflejando así la naturaleza de los datos disponibles, que no muestran una correspondencia directa entre multas y radares.

Finalmente, el atributo ‘delimitaciones’ de las entidades ‘Barrio’ y ‘Distrito’, definido como un array de ‘Coordenadas’, establece los límites geográficos de cada zona. Esta característica resulta esencial para representar dichas áreas en un mapa y verificar la pertenencia de coordenadas específicas a las mismas.

Este enfoque metodológico asegura que el modelo relacional no solo responde a las necesidades actuales, sino que también está preparado para adaptarse a los requerimientos futuros, manteniendo la coherencia, la flexibilidad y la integridad de los datos en todo momento.

## 4.2. Diseño de la interfaz de usuario

A continuación se muestran bocetos de una fase temprana de desarrollo. En esta etapa del proyecto no se había incluido la característica de la visualización multas, resultando en una representación simplificada de la sección de radares. El diseño de la aplicación contempla una interfaz intuitiva y accesible, en la que se incluye una barra de navegación estratégicamente ubicada para facilitar el tránsito fluido entre las distintas secciones.

Las imágenes de la Figura 6 ilustran la interfaz conceptual para la sección de accidentes. En estas vistas, el mapa interactivo destaca las zonas, diferenciados por su nivel de riesgo. Se dispone de un filtro exhaustivo que facilita la búsqueda de accidentes según una amplia gama

de características. Además, se presenta una gráfica analítica que categoriza los accidentes por diferentes atributos como el sexo y la edad de los involucrados. Al seleccionar un marcador en el mapa, se accede a un desglose detallado de la información relacionada con cada accidente.

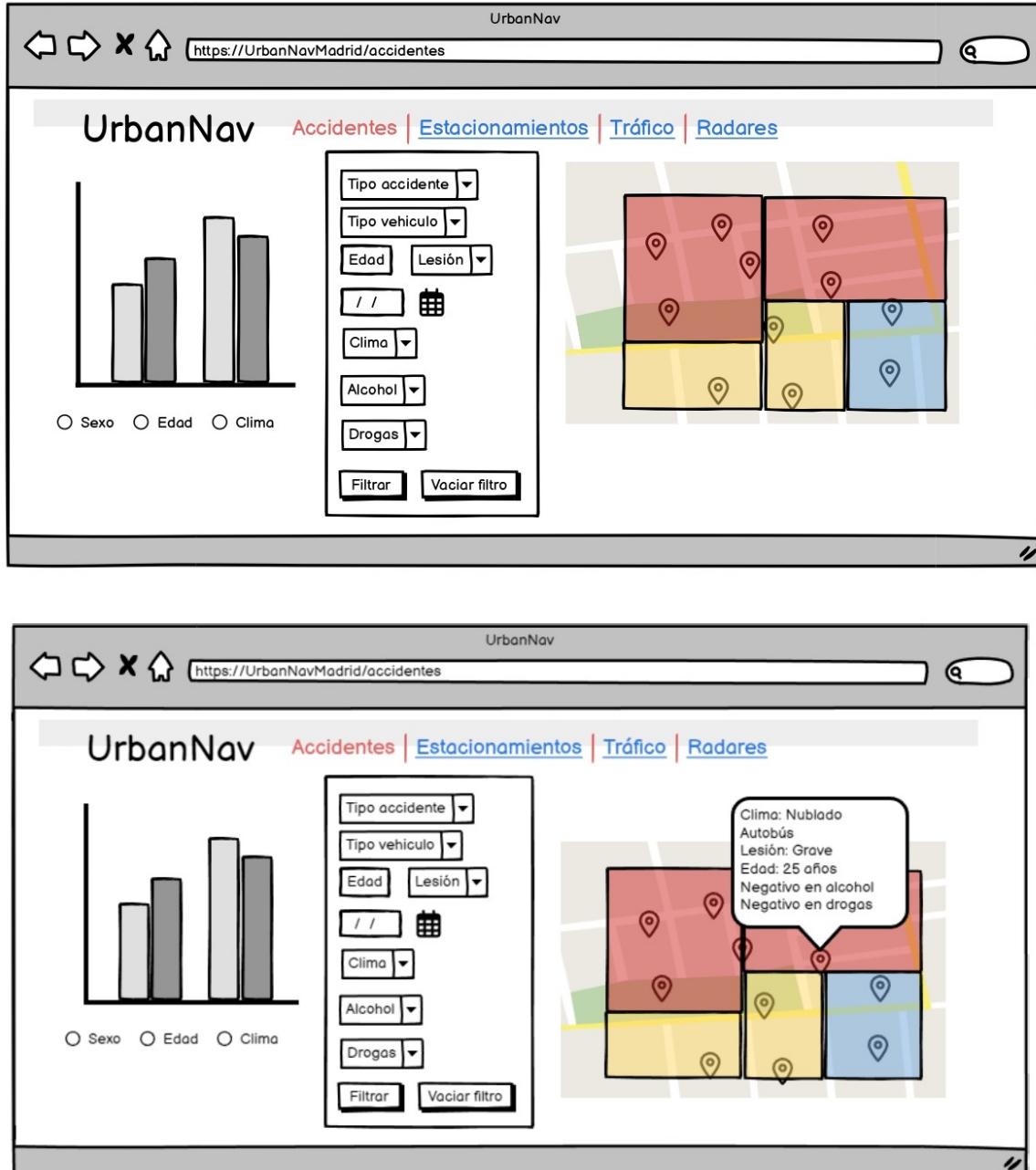


Figura 6: Boceto interfaz accidentes

La interfaz destinada a la visualización de estacionamientos se esboza en la Figura 7 . El

diseño propuesto incluye un sistema de filtrado y un mapa codificado por colores que refleja la clasificación de los estacionamientos, complementado con una leyenda explicativa. Similar a la sección de accidentes, al interactuar con un marcador de estacionamiento, se revelan detalles específicos, tales como el número de plazas, el tipo de estacionamiento y su disposición, ya sea en batería o en línea.

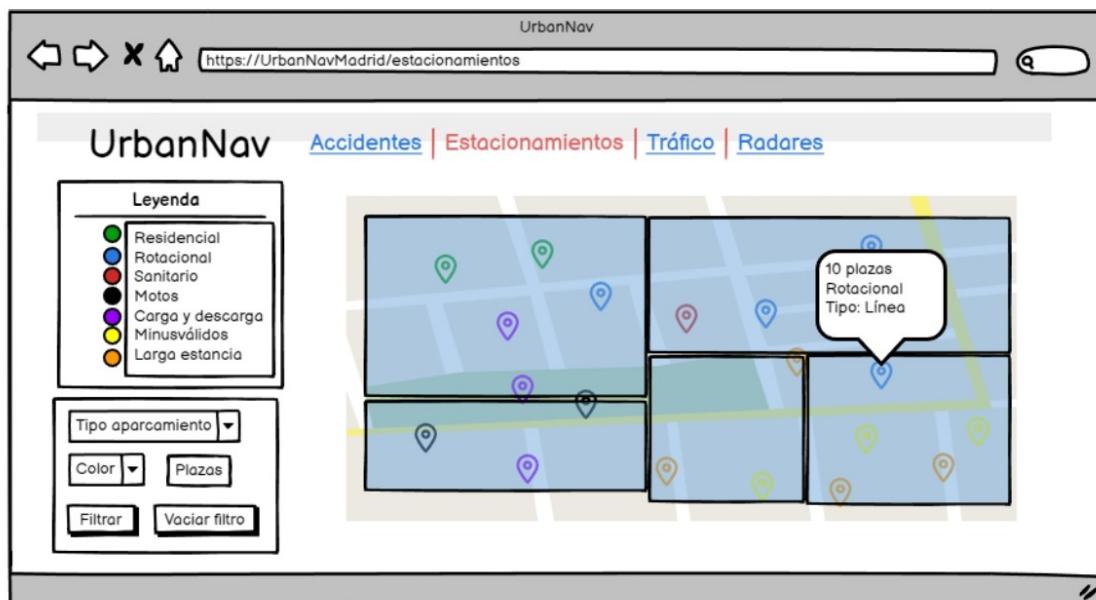
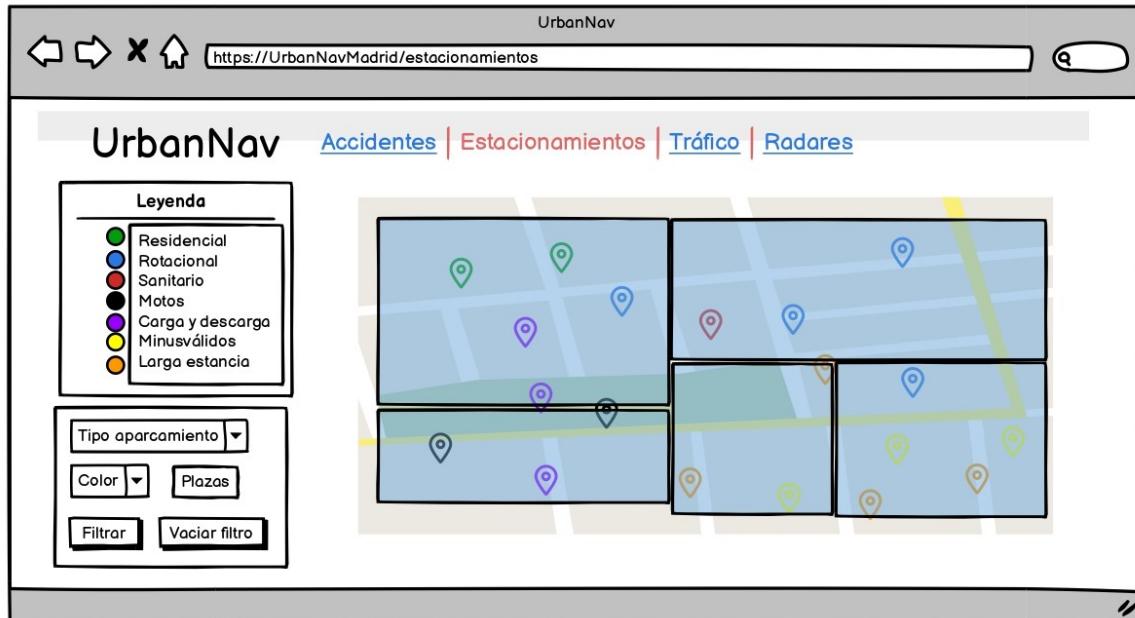


Figura 7: Boceto interfaz estacionamientos

La representación del tráfico, mostrada en la Figura 8, sigue una estructura análoga a la de la sección de accidentes. El objetivo es mostrar un análisis gráfico de la densidad del tráfico por zonas. El mapa clasifica visualmente las áreas por colores y señala las estaciones de monitorización como marcadores, proporcionando una interpretación intuitiva de los datos de tráfico.

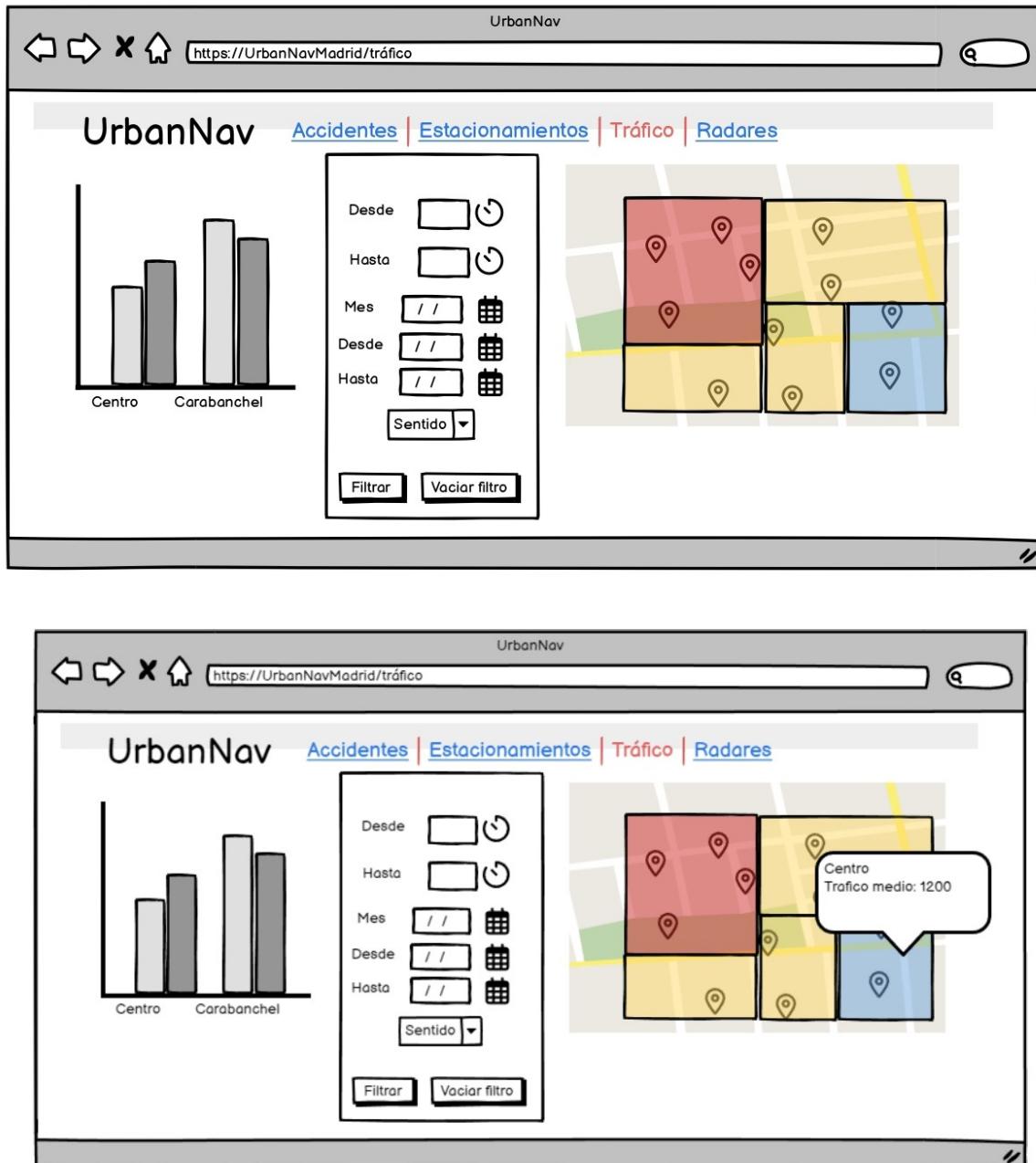


Figura 8: Boceto interfaz tráfico

Por último, la Figura 9 ofrece una perspectiva inicial de la interfaz para la sección de radares. En esta fase preliminar, la interfaz se centraba en mostrar las áreas delimitadas y la ubicación de los radares, junto con algunas de sus características esenciales.

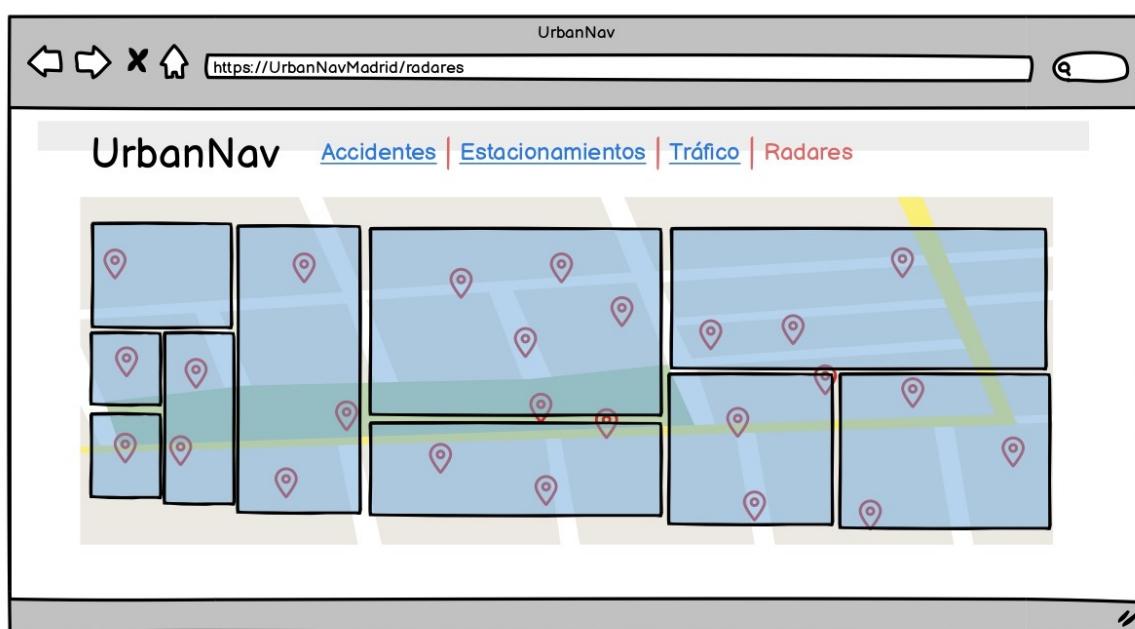


Figura 9: Boceto interfaz radares

### 4.3. Arquitectura de la aplicación

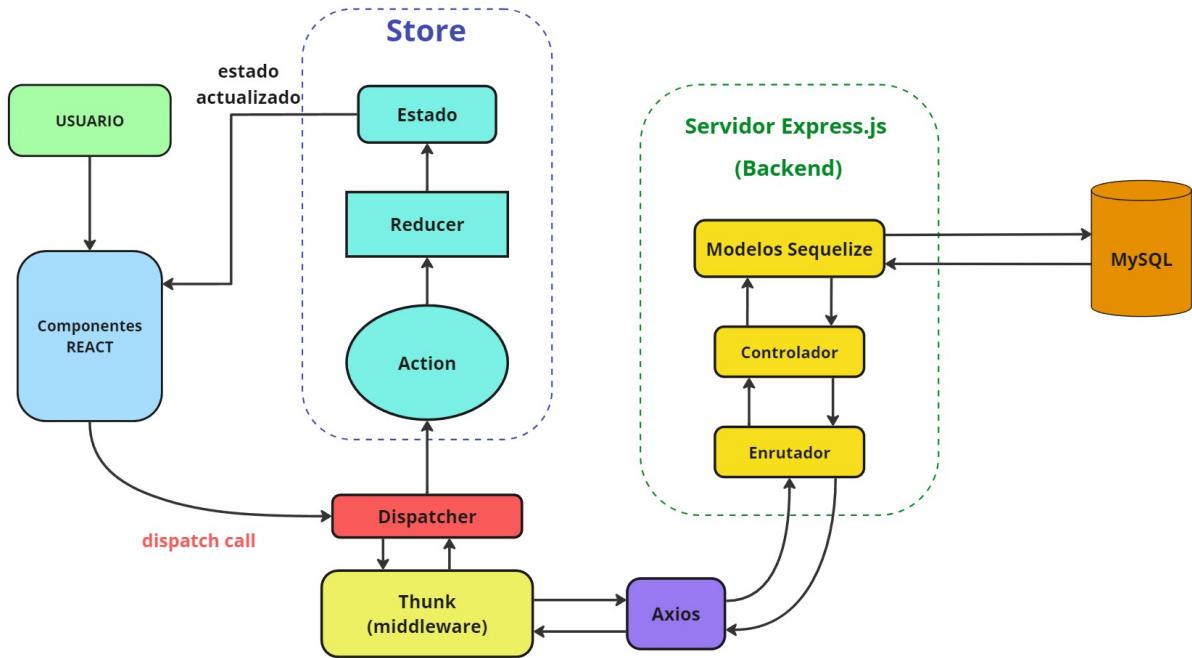


Figura 10: Arquitectura

La Figura 10 presenta una representación técnica de la arquitectura de la aplicación web, destacando la interacción entre sus componentes principales. La secuencia comienza cuando el usuario interactúa con la interfaz del **frontend**, compuesta por **Componentes React**. Las acciones generadas por elementos interactivos, como botones o formularios, son capturadas y despachadas a través de **Redux**, que actúa como un almacén centralizado para el estado de la aplicación.

**Redux Thunk**, como middleware, interviene en el proceso para gestionar peticiones asíncronas. Si una acción requiere una petición HTTP, Thunk delega esta tarea a **Axios**, que se encarga de enviar la solicitud al servidor **Express.js**. Dentro del servidor, el **Enrutador** asigna la petición a un Controlador específico basado en la ruta identificada.

El **Controlador** procesa la solicitud, utilizando **Sequelize** para interactuar con la base de datos **MySQL**. Sequelize Model abstracta las tablas de la base de datos, permitiendo al Controlador realizar operaciones CRUD. Sin embargo, en esta aplicación, las operaciones que podrá hacer el usuario están limitadas a peticiones GET.

Una vez que MySQL ejecuta la consulta, los resultados son devueltos en formato JSON a través del flujo inverso hasta llegar a Thunk, que despacha la respuesta como una nueva acción en Redux. El **Redux Store** recibe esta acción y actualiza el estado de la aplicación. Finalmente, el frontend refleja las actualizaciones del estado en la interfaz de usuario, completando así el ciclo de interacción.

Este diseño asegura un flujo de datos coherente y eficiente, facilitando la escalabilidad y el mantenimiento de la aplicación web.

# 5

# Implementación

## 5.1. Implementación

El proceso de implementación se ha estructurado en sprints o iteraciones que abarcan las distintas etapas del desarrollo del proyecto. A continuación, se detalla cada sprint y las actividades realizadas.

### 5.1.1. Iteración 0: 1 - 22 de febrero de 2024

Durante la primera iteración, se llevó a cabo un análisis exhaustivo del Open Data publicado por el ayuntamiento de Madrid. Se seleccionaron los conjuntos de datos pertinentes para su uso en la aplicación y se definieron posibles aplicaciones prácticas para los mismos. Además, se elaboraron los requisitos funcionales del sistema, los cuales ya han sido descritos en secciones previas de este documento. Y también se diseñaron los bocetos para la interfaz de usuario.

**Creación base de datos** Con la aprobación del tutor del proyecto sobre los requisitos y el modelo relacional propuestos, se inició la creación de la base de datos en un entorno local. Utilizando la IDE MySQL Workbench, se desarrolló un modelo relacional que permitió generar el esquema completo de la base de datos, facilitando la implementación de claves foráneas, entre otros elementos. El proceso seguido fue: ‘Database >Reverse Engineer >Next >Next >Seleccionar la base de datos >Next >Execute’. Posteriormente, se adaptó el modelo UML al diagrama de la base de datos, cuyas entidades principales se ilustran en la Figura 11.

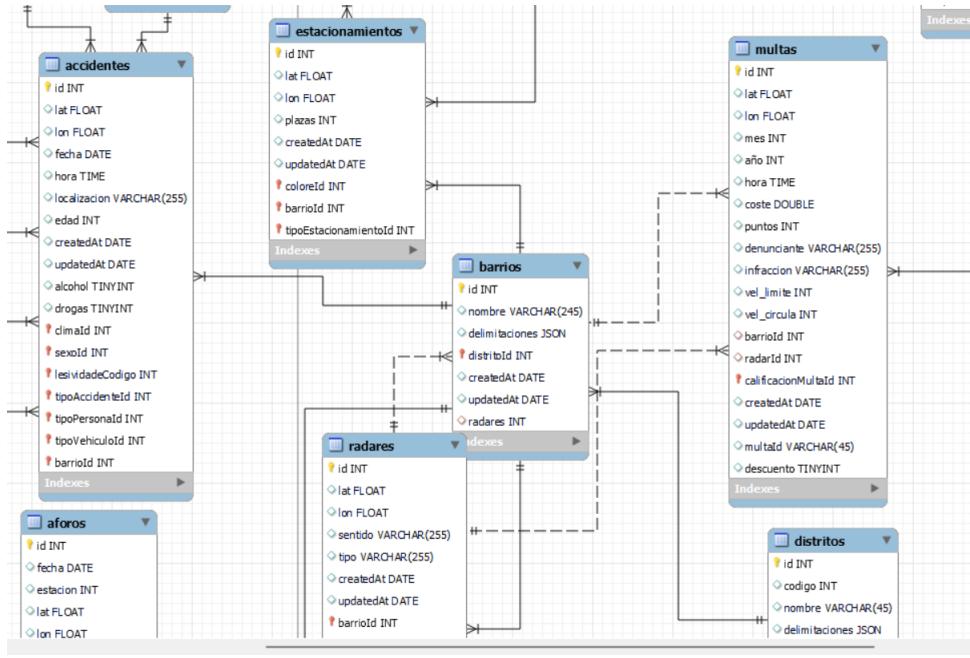


Figura 11: Modelo base de datos

Es importante mencionar que la biblioteca para el mapeo entidad-relacional (ORM), Sequelize, que se integrará más adelante, requiere la inclusión de dos campos en cada entidad: ‘CreatedAt’ y ‘UpdatedAt’. Por lo tanto, estos atributos se han añadido a todas las entidades de la base de datos.

**Creación del Backend** Para iniciar el proyecto, se ejecuta el comando `npm init -y`, el cual genera automáticamente un nuevo archivo `package.json`, estableciendo así la base para un nuevo proyecto de Node.js. Seguidamente, se procede a instalar las dependencias necesarias mediante el comando `npm i express nodemon -D morgan sequelize`. Este conjunto de instrucciones instala:

- **Express.js:** Un framework de Node.js seleccionado para el desarrollo de este proyecto.
- **Nodemon:** Una herramienta de utilidad que monitoriza los cambios en los archivos del proyecto y reinicia automáticamente el servidor, lo que resulta en una experiencia de desarrollo más eficiente.
- **Morgan:** Un middleware que registra las solicitudes HTTP realizadas a la aplicación, facilitando la visualización de las consultas a la base de datos en la consola.

- Sequelize: Un ORM (*Object-Relational Mapping*) que permite una gestión eficiente de la base de datos y facilita la definición de modelos.

El proyecto se estructura en dos componentes o directorios principales: ‘backend’ y ‘frontend’. La estructura dentro del directorio ‘backend’ se organiza de la siguiente manera (ver Figura 12):

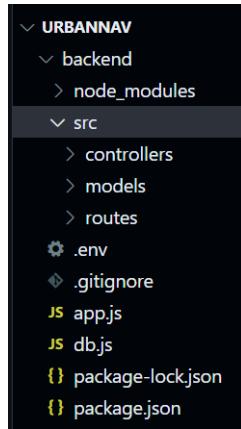


Figura 12: Estructura del módulo de backend.

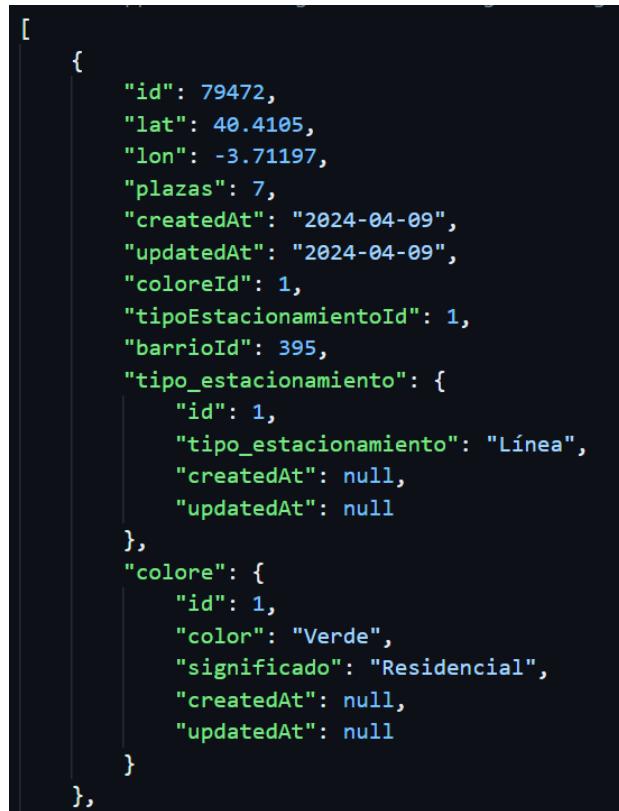
- `bd.js`: Gestiona la conexión con la base de datos.
- `app.js`: Se encarga del enrutamiento, establece la conexión con la base de datos y configura el servidor Express.js.
- `package.json`: Enumera las dependencias necesarias para el proyecto.
- `controllers`: Directorio que contiene los controladores de cada entidad.
- `models`: Directorio donde se definen los modelos y sus relaciones.
- `routes`: Directorio que establece las rutas para los métodos de los controladores.

Una vez instaladas las dependencias y creada la estructura del proyecto, se establece la conexión con la base de datos. Las credenciales de conexión se almacenan de forma segura en un archivo `.env` para proteger la información sensible. La librería `dotenv` [23] se utiliza para leer las variables de entorno definidas en este archivo. La configuración de la conexión [24] implica: el tiempo máximo de espera (`connectTimeout`) y el uso de un pool de conexiones. El pool permite gestionar múltiples conexiones simultáneas, en este caso hasta un máximo de 20,

y define un tiempo de espera de 30 segundos para adquirir una conexión, optimizando así el rendimiento.

**Modelos Sequelize** El paso siguiente implica la definición de todos los modelos [25] de datos que representan las tablas de la base de datos. Aquí se definen los atributos de cada modelo, incluyendo un enumerado y las relaciones entre modelos. Por ejemplo, los campos alcohol y drogas se representan en MySQL como tipo tinyint y en Sequelize como BOOLEAN, con Sequelize encargándose de la conversión entre estos tipos.

Tras la creación de los modelos, se realizan pruebas mediante consultas en los controladores para verificar la correcta conexión con la base de datos y la adecuada definición de los modelos. La Figura 13 muestra el resultado de una consulta GET a la entidad Estacionamiento. Para acceder a los atributos de entidades relacionadas, como Color y TipoEstacionamiento, se utiliza el parámetro include junto con el modelo correspondiente.



```
[{"id": 79472, "lat": 40.4105, "lon": -3.71197, "plazas": 7, "createdAt": "2024-04-09", "updatedAt": "2024-04-09", "coloreId": 1, "tipoEstacionamientoId": 1, "barrioId": 395, "tipo_estacionamiento": { "id": 1, "tipo_estacionamiento": "Línea", "createdAt": null, "updatedAt": null }, "colore": { "id": 1, "color": "Verde", "significado": "Residencial", "createdAt": null, "updatedAt": null }}, {"id": 79473, "lat": 40.4105, "lon": -3.71197, "plazas": 7, "createdAt": "2024-04-09", "updatedAt": "2024-04-09", "coloreId": 1, "tipoEstacionamientoId": 1, "barrioId": 395, "tipo_estacionamiento": { "id": 1, "tipo_estacionamiento": "Línea", "createdAt": null, "updatedAt": null }, "colore": { "id": 1, "color": "Verde", "significado": "Residencial", "createdAt": null, "updatedAt": null }}]
```

Figura 13: Resultado de llamar a getAllEstacionamientos.

Para el procesamiento de archivos CSV proporcionados por el Open Data, se emplea la librería csv-parser [26], que simplifica el análisis y manejo de estos archivos. Durante la lec-

tura, se han corregido caracteres especiales y codificaciones incorrectas con la ayuda de la librería iconv-lite [27]. En la lectura del CSV se analiza cada fila donde obtenemos un objeto que contiene los nombres de las columnas del CSV como claves y los datos específicos de cada columna para esa fila como valores. Métodos de JavaScript como trim, substring, split, parseInt, entre otros, se utilizan para adaptar los datos a las necesidades de la aplicación antes de su almacenamiento en la base de datos.

Otros pasos destacables en el procesamiento de datos ha sido el uso de técnicas avanzadas para la asignación precisa de puntos a áreas geográficas específicas. Utilizando la biblioteca turf.js, se ha empleado la función booleanPointInPolygon [28] para realizar comprobaciones geoespaciales críticas. Este método ha resultado ser fundamental para determinar la pertenencia de diversos elementos, como accidentes, estacionamientos y radares, a los barrios correspondientes dentro de la ciudad.

Además, se ha aplicado la misma técnica para establecer la inclusión de un barrio dentro de un distrito determinado. Esto se ha logrado calculando el punto medio geográfico del barrio y verificando su ubicación dentro de los límites del distrito.

### 5.1.2. Iteración 1: 23 de febrero - 1 de abril de 2024

A lo largo de la presente iteración, se ha llevado a cabo la creación del frontend de la aplicación, paralelamente al desarrollo continuo de consultas especializadas en el backend, con énfasis en las consultas de filtrado.

**Creación del Frontend** Una de las principales ventajas de utilizar React es la capacidad de instalar todas las dependencias necesarias y configurar una aplicación base con un solo comando: npx create-react-app. Adicionalmente, se han integrado librerías complementarias como react-router-dom para la gestión de rutas, react-leaflet para la implementación de mapas interactivos y react-bootstrap para el diseño de la interfaz de usuario.

Posteriormente, se estableció la conexión entre el backend y el frontend mediante la librería Axios. Axios es una herramienta de código abierto que simplifica la realización de solicitudes HTTP, permitiendo un manejo asíncrono de respuestas y errores a través de promesas, lo cual contribuye a una mayor legibilidad y mantenimiento del código.

La estructura de directorios del frontend se detalla brevemente a continuación, que a su vez se muestran en la Figura 14:



Figura 14: Estructura de carpetas del frontend.

- **App.js**: Define la estructura principal de la aplicación web y gestiona el enrutamiento.
- **index.js**: Configura el entorno de desarrollo y renderiza la aplicación React, integrando el store de Redux mediante el componente Provider.
- **App.css**: Contiene los estilos CSS que definen la apariencia visual de la aplicación.

La organización de los archivos se realiza creando una carpeta específica para cada entidad o grupo de componentes relacionados. Se ha logrado una reutilización eficiente de componentes como los gráficos (charts), ventanas modales (modal), tablas (table) y la barra de navegación (navbar), gracias a la filosofía de composición de React, que promueve la creación de componentes complejos a partir de otros más simples y fomenta la encapsulación de funcionalidad y presentación en bloques de código independientes.

**Implementación de Mapas en el Frontend** El primer componente implementado en el frontend fueron los mapas, que permiten la visualización de marcadores y zonas geográficas. Utilizando la librería react-leaflet, se han incorporado componentes como <Marker>para los marcadores y <Polygon>para las zonas, cada uno con sus respectivas propiedades: position o positions.

La propiedad position de <Marker>se define como un array de dos coordenadas, mientras que positions de <Polygon>es un array de coordenadas que delimitan el área a representar en el mapa. Se prevé añadir interactividad a estos elementos, como eventos al pasar el ratón o al hacer clic. Inicialmente, se planteó mostrar en el mapa los barrios y distritos como <Polygon>y los accidentes, radares, estacionamientos y estaciones de tráfico como <Marker>.

Debido a la gran cantidad de datos procesados, se ha establecido una estrategia para la visualización de marcadores en los mapas. Si se superan los 500 marcadores, estos no se mostrarán directamente. En su lugar, al hacer clic en una zona específica, se mostrarán los marcadores correspondientes. Además, se podrá seleccionar la visualización por distritos o barrios mediante un checkbox.

**Mapas de accidentes** El controlador clasifica las zonas según dos parámetros principales: lesividad y riesgo. La lesividad es un valor entre 0 y 100 que indica el nivel medio de peligro de una zona, basado en factores como el tipo de accidente y la gravedad de las lesiones. El riesgo se calcula a partir de la lesividad y la frecuencia de accidentes. Las zonas se clasifican en cuatro categorías de riesgo, reflejadas en el mapa con diferentes colores. Al hacer clic sobre una zona sale un popup, o ventana emergente, con 2 opciones: 'Ver más información' y 'Ver accidentes de este distrito/barrio', tal y como se muestra en la Figura 15.

**Mapas de estacionamientos.** Este mapa interactivo ofrece funcionalidades similares al anterior como la de mostrar los marcadores de una zona específica. Además, se han clasificado los estacionamientos por color, representando diferentes tipos. Esto está explicado en la leyenda, a la izquierda del mapa. Al filtrar por un tipo específico, las zonas se pintarán del color correspondiente. Al hacer clic en un marcador, se muestra información detallada sobre el estacionamiento como el número de plazas, si es en línea o batería y el tipo que es (rotacional, minusválidos, larga estancia, motos, etc.). Todo lo explicado puede verse en la Figura 16.

**Mapa de tráfico** El mapa de tráfico muestra barrios, distritos y estaciones de tráfico. La clasificación de las zonas se basa en la densidad diaria media de tráfico y se refleja en el ma-

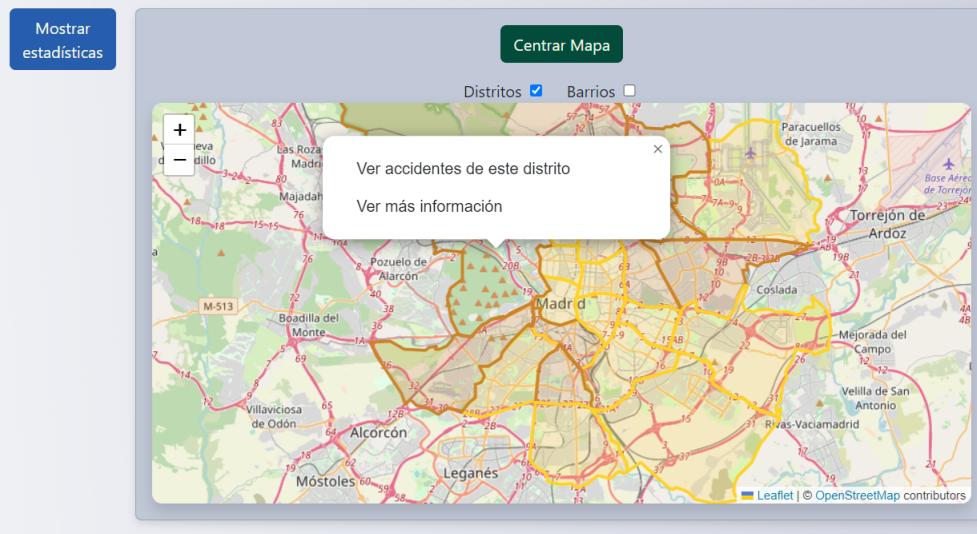
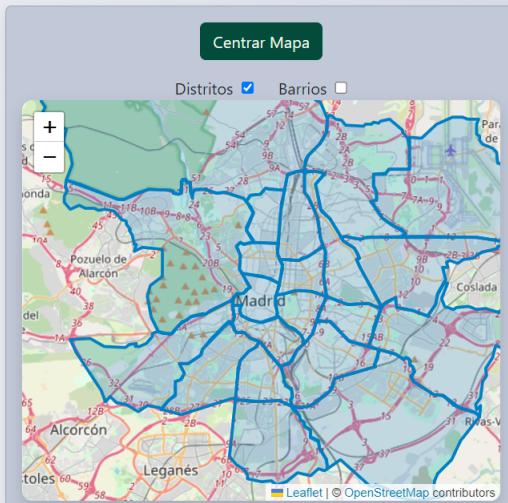


Figura 15: Mapa de accidentes.

## Estacionamientos: 1296

### Leyenda

- Tipo de estacionamiento (color)
- Residencial
  - Rotacional
  - Larga estancia
  - Ámbito sanitario
  - Minusválidos
  - Reserva de motos
  - Carga y descarga



Color: Sin especificar

Tipo de aparcamiento: Ninguna

Mínimo plazas:  Máximo plazas:

[Filtrar](#) [Limpiar filtro](#)

Figura 16: Mapa de estacionamientos.

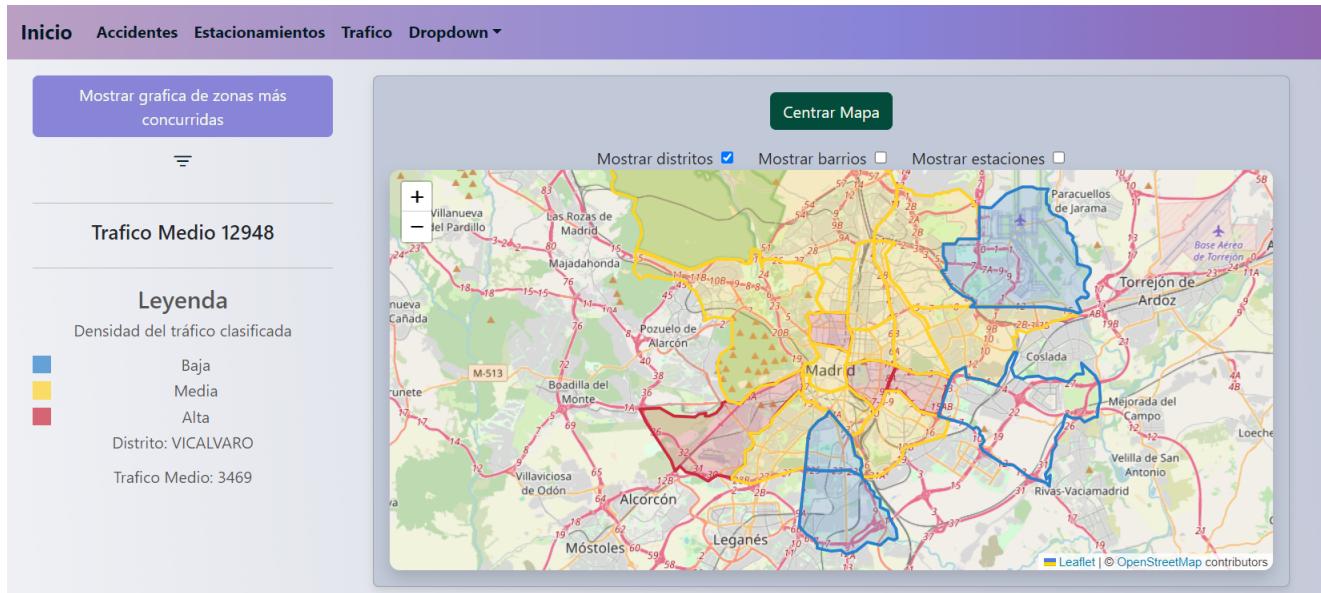


Figura 17: Mapa de tráfico.

pa con colores distintivos. La información detallada se muestra al hacer clic en una zona o estación, tal y como se muestra en la figura 17.

**Mapa de multas** Esta sección del mapa permite a los usuarios ocultar o visualizar radares. Al hacer clic en una zona, se mostrarán únicamente los radares de esa área. Tanto los radares como las zonas ofrecen información detallada al ser seleccionados. La funcionalidad de las multas se incorporó en etapas posteriores del desarrollo, mejorando la sección de radares con nuevas características. Figura 18.

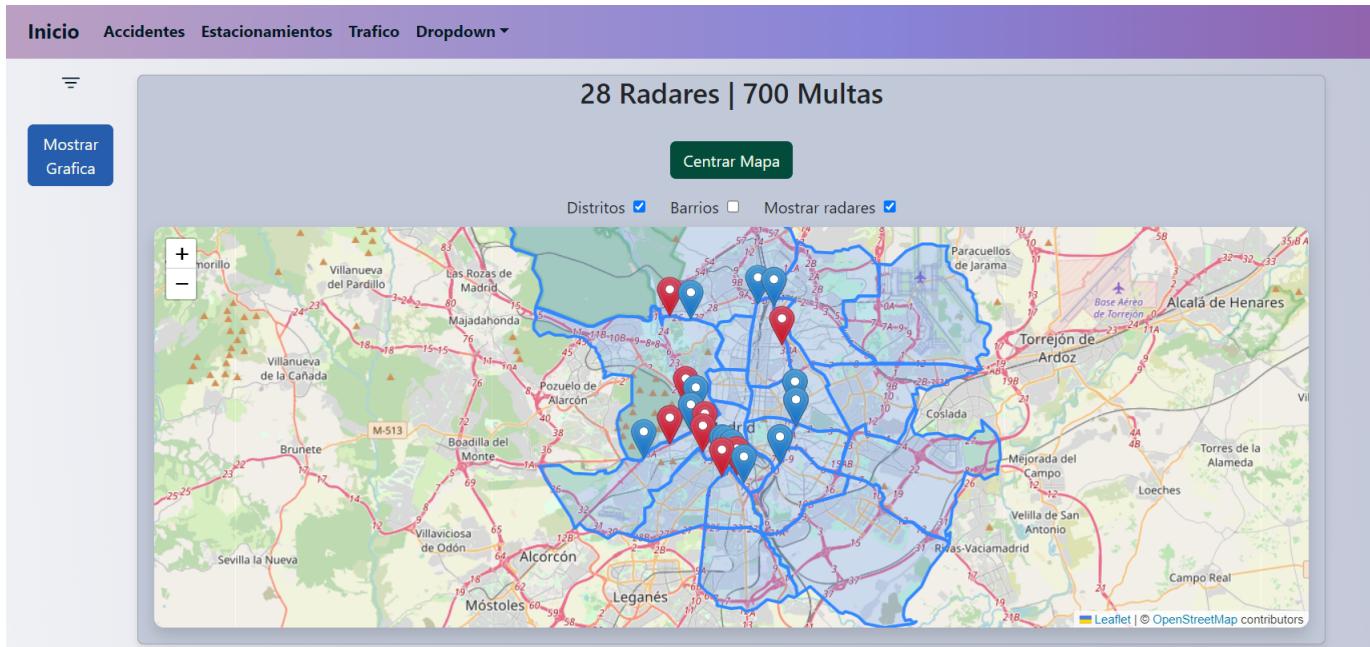


Figura 18: Mapa radares

**Implementación de los filtros.** El sistema de filtrado constituye un componente crítico de la aplicación, brindando a los usuarios la capacidad de realizar búsquedas precisas y eficientes basadas en criterios específicos. Cada entidad dentro de la aplicación está equipada con un mecanismo de filtro dedicado, que facilita la exploración de sus atributos. Los métodos desarrollados retornan información detallada de los distritos y barrios, incluyendo las entidades asociadas a cada área.

**Filtro accidentes.** Este filtro es el más completo, permitiendo búsquedas avanzadas basadas en múltiples características. La clasificación de zonas por nivel de riesgo se realiza mediante un algoritmo en el backend. El filtro admite consultas por tipo de vehículo, tipo de accidente, fecha, hora y características de los implicados como la edad, lesión y si fueron positivos en alcohol entre otros.

**Filtro estacionamientos.** Esta búsqueda no se hace en base a tantos atributos. Pero aun así permite realizar una búsqueda bastante específica ayudándose de la funcionalidad del mapa de buscar los aparcamientos de una zona concreta.

**Filtro tráfico.** En este método también se hace un cálculo por cada distrito y barrio, similar a los accidentes. Esta vez se calcula la media de tráfico diario de cada zona. En caso de que se

introduzcan en el filtro unas horas específicas solo se tendrán en cuenta esos momentos del día a la hora de hacer la media. Una vez calculada la media de cada zona y la media general se clasifica cada barrio y distrito por su densidad de tráfico. También se puede filtrar por el sentido de la circulación.

**Filtro frontend.** Para proporcionar una experiencia de usuario óptima, se ha integrado la funcionalidad de reiniciar el formulario de filtrado, permitiendo a los usuarios restablecer los campos a su estado inicial con facilidad. Esta característica es esencial para facilitar nuevas búsquedas sin la necesidad de ajustes manuales por parte del usuario. Además se le da la opción al usuario de hacer la carga inicial. También se ha añadido la posibilidad de ver el filtro aplicado en la búsqueda.

Al acceder a una sección por primera vez, el sistema realiza una consulta inicial que recupera un conjunto limitado de datos. Este enfoque previene la sobrecarga del sistema y mejora el rendimiento al evitar la carga masiva de datos innecesarios. Los usuarios tienen la libertad de solicitar cargas de datos adicionales según sea necesario, aplicando criterios de búsqueda específicos en el sistema de filtrado. En la figura 36 se puede observar el filtro de los accidentes.

**Mostrar marcador arrastrable**

<b>Alcohol</b>	<b>Drogas</b>
<input type="button" value="Sin especificar"/>	<input type="button" value="Sin especificar"/>
<b>Vehículo</b>	<b>Accidente</b>
<input type="button" value="Sin especificar"/>	<input type="button" value="Sin especificar"/>
<b>Lesión:</b>	
<input checked="" type="radio"/> Cualquiera <input type="radio"/> Leve <input type="radio"/> Grave <input type="radio"/> Fallecido	
<b>Sexo</b>	<b>Clima</b>
<input type="button" value="Sin especificar"/>	<input type="button" value="Sin especificar"/>
<b>Edad</b>	
<input type="button" value="Mínima"/>	<input type="button" value="Máxima"/>
Desde <input type="button" value="--:--"/> <input type="button" value=""/>	Hasta <input type="button" value="--:--"/> <input type="button" value=""/>
Desde <input type="button" value="dd/mm/aaaa"/> <input type="button" value=""/>	Hasta <input type="button" value="dd/mm/aaaa"/> <input type="button" value=""/>
<input style="background-color: #007bff; color: white; border-radius: 5px; padding: 5px; margin-right: 10px;" type="button" value="Filtrar"/> <input style="background-color: red; color: white; border-radius: 5px; padding: 5px;" type="button" value="Limpiar filtro"/>	

Figura 19: Filtro de accidentes.

### 5.1.3. Iteración 2: 2 - 25 de abril de 2024

El enfoque principal de esta iteración fue el desarrollo y la integración de las gráficas en la sección de accidentes y tráfico para mostrar estas entidades clasificadas por características concretas. Se introdujo el concepto de flujo de tráfico a través de un gráfico interactivo y la ventana modal para mostrar información detallada de una entidad. Además, se realizaron mejoras significativas en los mapas, tales como la funcionalidad de centrado automático, la incorporación de un checkbox para seleccionar la visualización de barrios o distritos y la extensión del filtro de accidentes para incluir un radio de búsqueda.

**Ventana modal.** En esta iteración, se ha enriquecido la interfaz de usuario con la integración de una ventana modal proporcionada por react-bootstrap [29]. Este componente visualmente atractivo permite la presentación de contenido adicional sin la necesidad de redirigir al usuario fuera de la página en curso. La ventana modal se despliega sobre el contenido principal, atenuando el fondo para focalizar la atención en la información emergente. Dicha ventana

puede cerrarse mediante un botón situado en la esquina superior derecha o haciendo clic fuera de su área. La adopción de este componente mejora la experiencia de visualización de datos, superando la funcionalidad previa basada en pop-ups asociados al mapa. En la figura 20 se ilustra la ventana modal correspondiente a un distrito en la sección del tráfico, mostrando detalles del mismo. Adicionalmente, se presenta en el pie de página el formulario para el cálculo del flujo, que será explicado posteriormente.

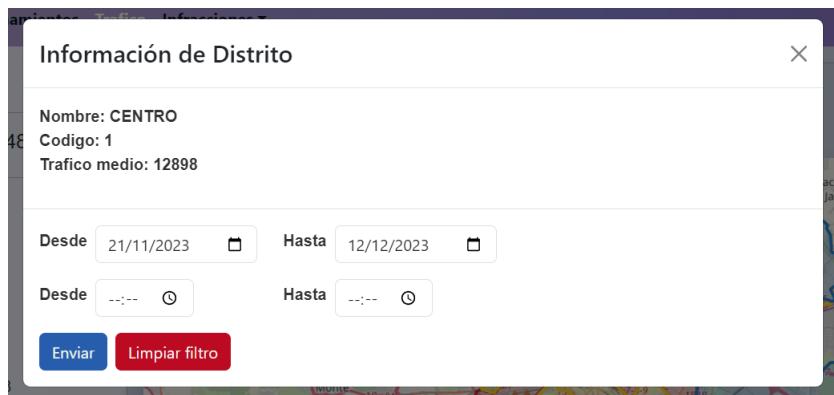


Figura 20: Ventana modal.

**Gráficas.** En el transcurso de la segunda iteración, se ha logrado un avance significativo en la visualización de datos mediante la implementación de gráficas interactivas. Utilizando la biblioteca Recharts, que facilita la integración de componentes gráficos con React, se ha enriquecido la sección de accidentes con herramientas analíticas visuales. La primera incorporación fue un BarChart o gráfico de barras, diseñado para representar distintos valores de las características de los accidentes. Se estableció un sistema que permite al usuario seleccionar entre varios atributos de los accidentes para clasificar y visualizar la frecuencia de accidentes correspondiente a cada categoría. Por ejemplo, en la figura 21, se ilustra la distribución de accidentes clasificados por género, revelando que de 1310 accidentes, 800 involucran a hombres. El algoritmo subyacente genera una lista de objetos, cada uno contenido los accidentes asociados a un valor específico de la categoría elegida.

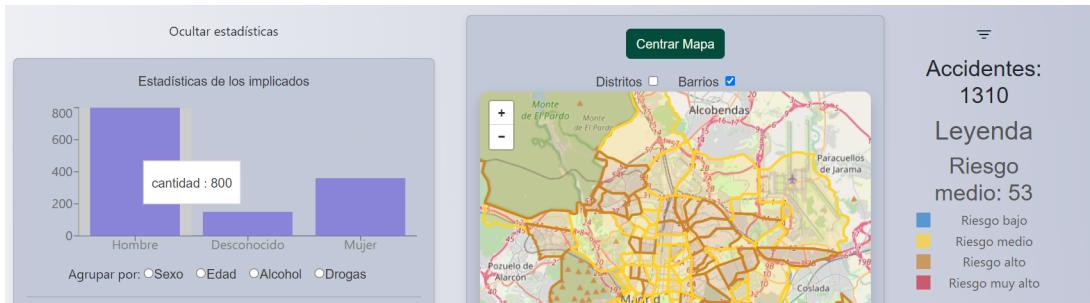


Figura 21: BarChart de accidentes.

Posteriormente, se decidió complementar esta visualización con un PieChart [30] o gráfico circular, con el objetivo de ofrecer diversidad en la presentación de datos. Esta segunda gráfica se centra en características intrínsecas del accidente, como las condiciones climáticas, el tipo de vehículo y la naturaleza del accidente. La implementación de ambas gráficas se basó en ejemplos proporcionados por la documentación de Recharts, adaptándolos cuidadosamente para satisfacer las necesidades específicas de la aplicación.

Como se ha mencionado anteriormente, las representaciones gráficas se han establecido como un recurso de gran utilidad, sobre todo en lo que respecta a la visualización del flujo. Esta funcionalidad, desplegada primeramente en la sección correspondiente al tráfico, capitaliza los datos obtenidos por las estaciones de monitorización. La información relativa a la circulación en barrios y distritos se deriva de los registros de estas estaciones, que también se visualizan en el mapa. El objetivo es destacar la dinámica del desplazamiento vehicular a través del tiempo. Se ha incorporado un componente denominado Brush [31], facilitado por Recharts, que permite a los usuarios seleccionar un rango específico de datos dentro de una gráfica, ideal para la inspección y análisis de grandes volúmenes de información. Se ha establecido que el componente Brush se muestre cuando el intervalo temporal seleccionado por el usuario comprenda diez unidades de tiempo o más. Al interactuar con una zona o estación y activar la ventana modal (Figura 20), se despliega un formulario que admite la introducción de dos fechas y dos horas, definiendo así el período para el análisis del flujo de tráfico. El usuario podrá completar solo uno de los dos campos, fechas o horas. Si se introducen dos fechas, se generará una gráfica donde cada barra simboliza el tráfico de cada día. En cambio, si se especifican dos horas, cada barra reflejará la media de tráfico horaria acumulada de todos los días registrados. Junto a la gráfica, se muestra un mapa que señala la zona o estación

en cuestión. Para los distritos, se visualiza la zona junto con las estaciones que pertenecen a ella. Finalmente, se ha proporcionado la opción de que, al seleccionar una estación de tráfico específica, se exhiba el flujo basado exclusivamente en los datos de esa estación, permitiendo así un análisis comparativo del tráfico entre diferentes áreas de un mismo distrito. La figura 22 muestra el flujo de tráfico de la estación calle Princesa en el distrito Centro, desde el 21 de noviembre hasta el 10 de diciembre de 2023, destacando que el 1 de diciembre se registró un total de 34,432 vehículos.



Figura 22: Ejemplo del flujo de tráfico.

**Mejoras en los mapas.** Respecto a los mapas se han implementado mejoras significativas utilizando como referencia ejemplos de la biblioteca react-leaflet. Se ha creado un módulo específico, denominado 'MapFunctions.js', que centraliza el desarrollo de componentes de mapa reutilizables, facilitando su integración en distintas secciones de la plataforma.

- **Centrado automático del mapa:** Una de las funcionalidades clave incorporadas es el centrado automático, conocido en la documentación de react-leaflet como 'External state' [32]. Esta herramienta permite a los usuarios establecer un punto focal y nivel de acercamiento predeterminados en el mapa con la simple acción de pulsar un botón ubicado en la parte superior de la interfaz.
- **Control de capas:** Además, se ha adaptado la funcionalidad de 'Layers control' [33] para

ofrecer una gestión dinámica de la visibilidad de los elementos del mapa. Con inspiración de la implementación original, se ha diseñado un sistema que permite alternar la visualización de ciertos elementos, como barrios y distritos, mejorando así la claridad y la experiencia de usuario.

- Marcador desplazable: Finalmente, se ha integrado la funcionalidad de ‘Draggable Marker’ [34], un marcador interactivo que puede desplazarse por el mapa. Al almacenar las coordenadas de este marcador y una distancia especificada por el usuario, el sistema es capaz de identificar y retornar los elementos que se encuentran dentro del radio definido. Esta característica ha sido particularmente útil en el filtro de accidentes, mejorando la precisión de las búsquedas realizadas por los usuarios. En la figura 23 se muestra una búsqueda por un radio de 1300 metros y positivos en alcohol que devuelve 20 accidentes.

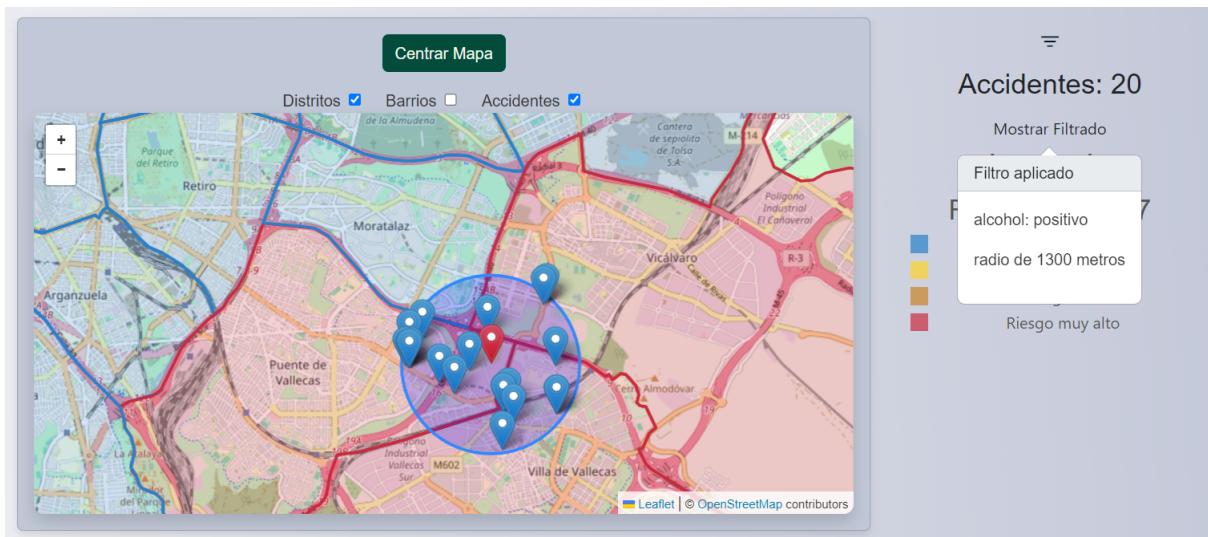


Figura 23: Ejemplo de radio de búsqueda.

#### 5.1.4. Iteración 3: 26 de abril - 26 de mayo de 2024

Durante la tercera iteración, se implementaron avances en el diseño visual de la aplicación y se optimizó la gestión del estado de la misma mediante Redux. Se replicó la funcionalidad desarrollada para el flujo de tráfico en la sección de accidentes. Se corrigieron errores menores y se llevó a cabo un análisis detallado de la entidad de multas con el fin de integrarla en la aplicación, lo que resultó en mejoras sustanciales en la sección de radares.

**Mejoras en la interfaz de usuario.** A medida que avanzaba el desarrollo de la aplicación, se satisfacían numerosos casos de uso. No obstante, se identificó la necesidad de mejorar el aspecto visual de la herramienta. Gracias a la integración de Bootstrap y React-Bootstrap, así como la aplicación de estilos CSS personalizados, se logró una notable mejora en la interfaz de usuario.

Una de las adiciones más significativas fue el componente Offcanvas [35], accesible desde la sección ‘Components’ de React-Bootstrap. Este componente permite la creación de un menú lateral que alberga el contenido de los filtros, el cual puede mostrarse u ocultarse a voluntad desde el borde de la pantalla. Esta implementación ha resultado ser una solución eficiente para optimizar el espacio, especialmente en secciones con elementos extensos como los filtros de accidentes. Además, se ha configurado el ancho del menú para que sea responsive y se adapte a diferentes tamaños de pantalla mediante los breakpoints de Bootstrap. En la figura 24 se puede ver el filtro de los accidentes en el componente Offcanvas.

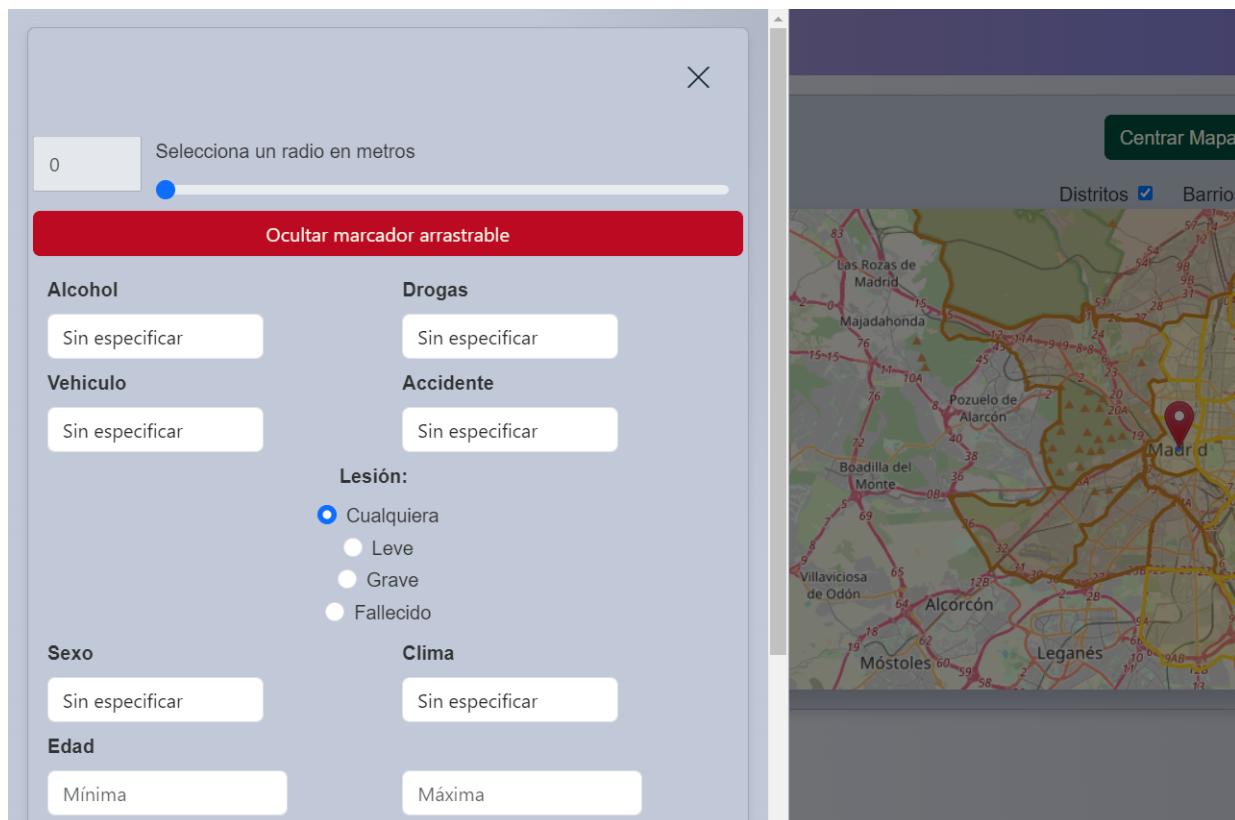


Figura 24: Offcanvas filtro

Para la visualización de la información del filtro aplicado, se han utilizado los componentes OverlayTrigger [36] y Popover de React-Bootstrap. Con OverlayTrigger, se activa el componente Popover, que muestra los atributos seleccionados en el filtro. Figura 25.



Figura 25: Filtro aplicado Popover

Uno de los principales objetivos era garantizar la accesibilidad de la aplicación web desde cualquier dispositivo, independientemente del tamaño de su pantalla. Los breakpoints [37] de Bootstrap han sido fundamentales en este aspecto, permitiendo una adaptación fluida del diseño a distintas resoluciones de pantalla.

En cuanto a las mejoras visuales específicas, se han efectuado ajustes en el componente Navbar [38] de React-Bootstrap. Se ha implementado un cambio sutil en el color de las secciones al pasar el cursor sobre ellas, y un cambio permanente en la sección del Navbar correspondiente a la página activa. Estas modificaciones han mejorado la usabilidad de la aplicación, facilitando la identificación de la sección en uso, proporcionando al usuario una mayor sensación de control y reforzando la jerarquía visual para una mejor comprensión de la estructura y organización del sitio web. En la figura 26 se muestra el diseño *responsive* de la aplicación en formato de dispositivo móvil y los cambios mencionados en el Navbar.



Figura 26: Diseño *responsive* de la barra de navegación.

**Manejo del estado con Redux.** En esta etapa avanzada del desarrollo del proyecto, se identificó la necesidad de gestionar el estado de la aplicación de manera más eficiente y con un código más claro. Por ello, se dedicó una parte considerable de esta iteración a la refactorización del código para integrar Redux, una decisión que simplifica la gestión del estado y mejora la mantenibilidad del código.

La implementación de Redux se ha realizado siguiendo una estrategia de modularización del estado global. Utilizando los slices de Redux, se ha dividido el estado en segmentos más pequeños y manejables, cada uno con su propio reducer y acciones asociadas. Esta técnica no solo incrementa la escalabilidad del proyecto, sino que también facilita la reutilización del código. Se ha optado por crear un slice específico para cada sección de la aplicación, lo que permite almacenar de forma organizada tanto los datos de las entidades como los filtros aplicados.

Para las acciones que requieren realizar consultas a la base de datos, se ha hecho uso de thunks y axios, proporcionando así una solución asíncrona y eficaz. Las acciones que no implican interacción con la base de datos, como el manejo del estado del filtro o la selección de elementos específicos en el mapa, se gestionan de manera directa.

Una de las principales ventajas de la adopción de Redux es la disponibilidad de Redux Dev-

Tools, una extensión para navegadores que ofrece herramientas robustas para la depuración y el testing de aplicaciones React que utilizan Redux. Como se ilustra en la figura 27, Redux DevTools permite:

- Inspeccionar el estado global de la aplicación en cualquier momento, facilitando la identificación de cambios en el estado y la detección de posibles problemas.
- Registrar un historial completo de las acciones despachadas, mostrando su impacto en el estado de la aplicación.
- Navegar hacia atrás y adelante en el historial de estados, lo cual es muy útil para la depuración de comportamientos que ocurren en condiciones específicas.

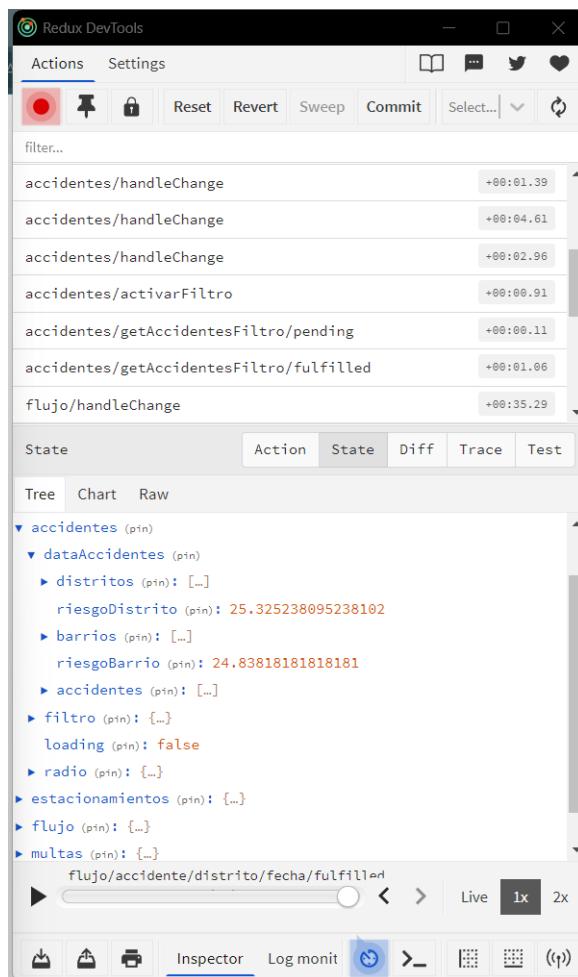


Figura 27: Redux DevTools

**Flujo accidentes.** La incorporación del análisis de flujos en la sección de accidentes se realizó con relativa facilidad, aprovechando la infraestructura previamente establecida para la sección de tráfico. Esta funcionalidad permite visualizar la frecuencia de accidentes durante un intervalo de tiempo específico, tanto a nivel de distritos como de barrios. Las barras en el gráfico representan el número de accidentes ocurridos en cada unidad de tiempo, que puede ser por horas o días. Asimismo, los usuarios tienen la capacidad de examinar el flujo de un barrio en particular dentro de un distrito. Un ejemplo de esto se muestra en la figura 28, donde se ilustra el flujo de accidentes en el barrio Adelfas del distrito Retiro, en el rango horario de 19:30 a 23:44. En este caso, cada barra del gráfico acumula la totalidad de accidentes registrados en ese intervalo horario a lo largo de todos los días. Es importante destacar que, al igual que en la sección de tráfico, la primera barra incluye los datos desde la hora de inicio hasta el comienzo de la hora siguiente, y la última barra abarca desde el inicio de la última hora hasta el final del periodo seleccionado. En la visualización proporcionada (Figura 28), la primera barra comprende los accidentes desde las 19:30 hasta las 20:00, y la última, los sucesos entre las 23:00 y las 23:44.



Figura 28: Ejemplo de flujo de accidentes.

**Pequeñas mejoras implementadas en esta fase:**

- **Sección Accidentes:** Al seleccionar una zona en el mapa (distrito o barrio), el gráfico se actualiza para mostrar únicamente los accidentes correspondientes a la zona elegida.
- **Sección Tráfico:** La leyenda ahora refleja el promedio de tráfico de la zona que se está inspeccionando con el cursor o mediante selección.
- **Sección Tráfico:** Se ha añadido un botón en el filtro que permite visualizar el promedio de tráfico de todos los datos almacenados en la base de datos.

#### **Errores corregidos en esta iteración:**

- **Sección Accidentes:** Se corrigió un error donde la ubicación del marcador arrastrable se reiniciaba al centro del mapa al aplicar el filtro de radio. Ahora, tanto el círculo que representa el radio como el marcador mantienen su posición tras la búsqueda.
- **Sección Estacionamientos:** Se solucionó un problema donde, tras mostrar los estacionamientos de una zona específica y luego volver a visualizar todas las zonas, el conteo total de estacionamientos no se actualizaba correctamente.
- **Sección Estacionamientos:** Se rectificó un error en el algoritmo que procesaba el CSV de estacionamientos para discapacitados, el cual no registraba adecuadamente el número de plazas disponibles. Los datos han sido reinsertados correctamente en la base de datos.

**Análisis de las multas.** Se evaluó el potencial de agregar valor a la aplicación mediante la inclusión de datos sobre multas, estableciendo una relación entre las sanciones por exceso de velocidad y los radares correspondientes. A pesar de que solo una fracción de las multas se debe a infracciones de velocidad, se propusieron nuevas funcionalidades en la sección de radares para aprovechar esta información. Además, se decidió crear una sección dedicada exclusivamente a las multas, que contará con una tabla detallada de infracciones y un filtro para búsquedas específicas. Dado que las multas por exceso de velocidad no incluían coordenadas, la asociación con los radares se realizó utilizando el punto kilométrico (PK) y la carretera especificados en el atributo de lugar de las multas, como se describió en la sección de análisis de datos de este documento. No obstante, algunos radares no se asociaron debido a que solo se

han incluido en la base de datos las multas correspondientes a diciembre de 2023, que ascienden a un total de 200 mil. Esta fase del proyecto terminó con la inserción de las multas en la base de datos.

#### **5.1.5. Iteración 4: 27 de mayo - 17 de junio de 2024**

La iteración final estuvo dedicada a la elaboración de la documentación del proyecto y a la ejecución de pruebas exhaustivas. Se completó la implementación de todas las funcionalidades relacionadas con las multas y se realizaron mejoras en la sección de radares. Se elaboró un manual de usuario detallado, se perfeccionó el manejo de errores en los formularios y, como culminación del proyecto, se procedió al despliegue de la aplicación en un entorno de nube.

**Memoria.** Durante el desarrollo del proyecto, se realizó documentación que incluyó la toma de apuntes y un seguimiento diario de las actividades realizadas. La mayoría de los diagramas y esquemas presentados en esta memoria fueron diseñados en etapas tempranas. No obstante, la redacción formal de la memoria comenzó en la última iteración del proyecto. A lo largo de esta fase, se consultó al tutor académico para resolver dudas específicas, alternando la redacción con la implementación de los últimos casos de uso y las correcciones finales. Los detalles sobre las pruebas realizadas se discutirán en la sección subsiguiente a la implementación.

**Mejoras en la sección de radares.** La introducción de la entidad de multas ha permitido una explotación más eficiente de la sección de radares. Esta mejora ha integrado elementos desarrollados en iteraciones previas, como gráficos y filtros, y ha incorporado un nuevo componente de tabla dinámica. Este componente, que recibe las propiedades data y columns, facilita la presentación de datos en la tabla, donde data contiene los registros a mostrar y columns enumera las cabeceras y atributos correspondientes. Esta abstracción de datos posibilita la reutilización del componente para diferentes conjuntos de datos. La tabla permite la ordenación ascendente o descendente de los registros al seleccionar la cabecera correspondiente al atributo deseado.

Con la carga de radares, se seleccionan automáticamente las multas por exceso de velocidad asociadas desde la base de datos, con un criterio inicial de 50 multas por radar. Al interactuar con un radar, el usuario puede visualizar una ventana modal con detalles específicos y acceder

a una tabla que lista todas las multas detectadas por ese radar. Adicionalmente, se presenta un gráfico que clasifica los radares según el número de multas registradas, con la posibilidad de mostrar un popup en el mapa al seleccionar una barra del gráfico.

Se ha implementado un filtro avanzado que permite búsquedas basadas en criterios como puntos perdidos por infracción, coste de la multa, hora, entre otros.

**Sección de Multas.** La sección dedicada a las multas cuenta con una tabla completa de infracciones y un filtro más complejo que el de la sección de radares. Esta tabla, que también utiliza el componente de tabla dinámica, ofrece funcionalidades de ordenación y paginación [39], esta última gracias a las capacidades de React-Bootstrap.

**Manejo de errores.** La gestión de errores en los formularios se centró principalmente en la validación de entradas numéricas, asegurando que los números negativos no fueran aceptados donde se requerían valores mayores o iguales a cero. Además, se implementaron comprobaciones para rangos de valores, ya sean temporales o numéricos, como en el caso de las plazas de estacionamiento, los períodos de tiempo para flujos de datos, o los costes mínimos y máximos de las infracciones. Se estableció un sistema de alertas para notificar al usuario inmediatamente si el valor inferior de un rango excedía al superior. En cuanto a la entrada de fechas para el análisis de tráfico, se diseñó una validación que impide la selección de una fecha específica si previamente se ha seleccionado un mes completo, y viceversa. Los mensajes de error se presentaron mediante el elemento Feedback [40] del componente Form de React-Bootstrap, proporcionando descripciones detalladas de los errores para que los usuarios comprendan claramente la naturaleza del problema y cómo resolverlo. Un ejemplo de estos mensajes de error se ilustra en la figura 29. La lógica para el manejo de estos errores se implementó dentro de las acciones de Redux, donde se verificaba la presencia de errores tras cualquier modificación en el estado del filtro por parte del usuario. Si se detectaba un error, el mensaje correspondiente se almacenaba en el estado para su posterior visualización en pantalla. No solo se muestran los errores por pantalla sino que no se puede enviar el formulario.

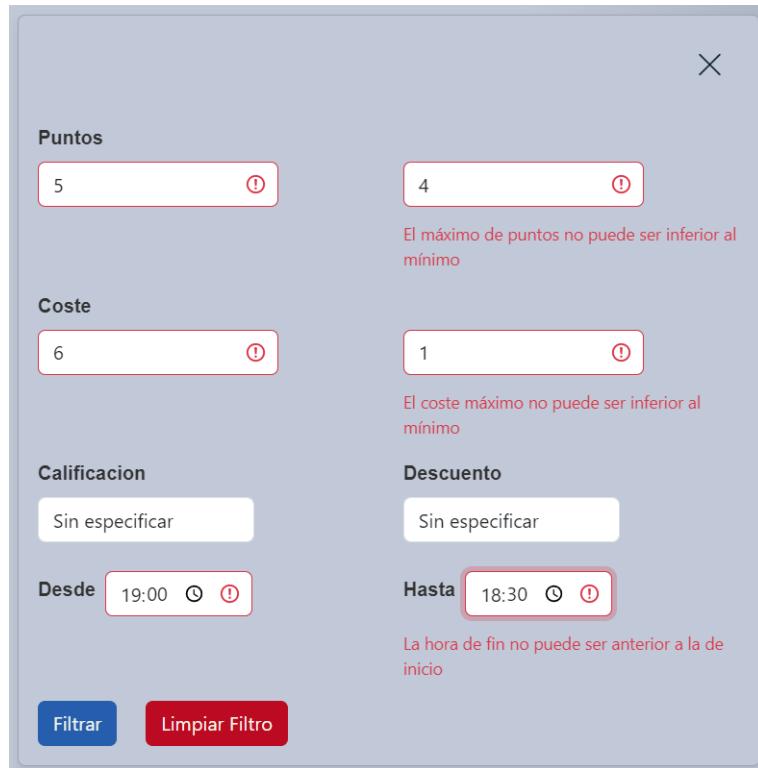


Figura 29: Manejo de errores

**Despliegue de la aplicación.** El proceso de despliegue comenzó con la base de datos, utilizando Heroku y JawsDB MySQL en su plan gratuito. JawsDB MySQL es un servicio de gestión de bases de datos que ofrece alto rendimiento y facilidad de uso. Tras reconfigurar la conexión con las credenciales de despliegue, se procedió con el despliegue del backend.

Inicialmente, se consideró utilizar Vercel para desplegar tanto el backend como el frontend. Sin embargo, se encontraron problemas al configurar el servidor de Express.js, específicamente errores relacionados con la sintaxis de módulos ES (ECMAScript modules). Esto llevó a optar por Render, otra plataforma de alojamiento en la nube, que facilitó el proceso de despliegue. El frontend fue la última parte de la aplicación en ser desplegada, tarea que se realizó sin complicaciones en Render. Una vez actualizados todos los endpoints de conexión a los del entorno de despliegue, la aplicación quedó completamente operativa y accesible para el público.



# 6

# Conclusiones y Líneas Futuras

## 6.1. Conclusiones

Al concluir este proyecto, es pertinente reflexionar sobre los logros alcanzados y el crecimiento personal experimentado durante su desarrollo.

El objetivo principal de desarrollar un proyecto de software completo se ha cumplido con éxito. No solo se han satisfecho los requisitos iniciales establecidos en el anteproyecto, sino que también se han incorporado mejoras adicionales que han enriquecido la funcionalidad de la aplicación. Este proceso ha servido para consolidar los conocimientos adquiridos a lo largo de la formación académica y ha propiciado el aprendizaje de nuevas competencias a través de la investigación y la práctica.

A nivel personal, este proyecto ha significado un avance sustancial en mi experiencia en el ámbito del desarrollo de software. Ha sido una oportunidad para fortalecer mis habilidades de gestión y autodisciplina, especialmente al ser la primera vez que afronto un desafío de esta envergadura de manera individual. La planificación y el cumplimiento de horarios y plazos han sido aspectos cruciales para la culminación exitosa del proyecto.

Se han alcanzado y, en algunos casos, superado los objetivos propuestos, gracias a una eficiente gestión del tiempo y al apoyo constante del tutor. El resultado es una herramienta valiosa que cumple con su propósito de informar a los ciudadanos sobre la movilidad en Madrid, facilitando el análisis de la misma. A pesar de los logros, se reconoce que hay espacio para mejoras y extensiones, las cuales se explorarán en la siguiente sección del documento.

## **6.2. Líneas Futuras**

Al contemplar el futuro desarrollo de la aplicación, se identifican varias vías de mejora que podrían enriquecer significativamente su funcionalidad y precisión.

La incorporación de nuevas secciones basadas en distintas entidades permitiría un análisis más amplio de la movilidad en Madrid. Explorar dimensiones adicionales como el transporte público o las rutas peatonales podrían ofrecer nuevas perspectivas de este análisis de la movilidad.

Además, la implementación de una funcionalidad para la lectura automática de archivos de Open Data potenciaría la veracidad y actualización constante de la información. Una base de datos enriquecida con datos más variados y actualizados garantizaría una herramienta más robusta y confiable.

Finalmente, se sugiere mejorar la capacidad analítica de la aplicación mediante la introducción de gráficos más sofisticados que permitan clasificar y comparar datos por una gama más amplia de atributos. Esto podría incluir análisis predictivos, tendencias a largo plazo y correlaciones entre diferentes variables.

# Referencias

- [1] Visualiza Madrid con Datos Abiertos. <https://visualizadatos.madrid.es/>.
- [2] Express 4.x - API Reference. <https://expressjs.com/en/4x/api.html>.
- [3] Core Concepts | Sequelize. <https://sequelize.org/docs/v6/category/core-concepts/>.
- [4] Geodetic transformation – PROJ 9.5.0-dev documentation. <https://proj.org/en/latest/usage/transformation.html>.
- [5] MySQL :: MySQL 8.4 Reference Manual. <https://dev.mysql.com/doc/refman/8.4/en/>.
- [6] Describir la UI – React. <https://es.react.dev/learn/describing-the-ui>, .
- [7] React Bootstrap | React Bootstrap. <https://react-bootstrap.netlify.app/>, .
- [8] BarChart | Recharts. <https://recharts.org/en-US/api/BarChart>, .
- [9] Getting Started with Redux | Redux. <https://redux.js.org/introduction/getting-started>, March 2024.
- [10] bezkoder. Redux-Toolkit example with CRUD Application, July 2021.
- [11] Qué es Redux. <https://desarrolloweb.com/articulos/que-es-redux.html>.
- [12] Usage Guide | Redux Toolkit. <https://redux-toolkit.js.org/usage/usage-guide>, January 2024.
- [13] API de Axios | Axios Docs. [https://axios-http.com/es/docs/api\\_intro](https://axios-http.com/es/docs/api_intro).
- [14] Writing Logic with Thunks | Redux. <https://redux.js.org/usage/writing-logic-thunks>, December 2022.
- [15] Conjuntos de datos - Portal de datos abiertos del Ayuntamiento de Madrid. <https://datos.madrid.es/sites/v/index.jsp?vgnnextoid=374512b9ace9f310VgnVCM100000171f5a0aRCRD&>
- [16] Accidentes de tráfico - Portal de datos abiertos del Ayuntamiento de Madrid. <https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnnextoid=7c2>

- [17] Servicio de Estacionamiento Regulado (SER) - Portal de datos abiertos del Ayuntamiento de Madrid.  
<https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=497>
- [18] Aforos de tráfico permanentes - Portal de datos abiertos del Ayuntamiento de Madrid.  
<https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=fab>
- [19] Radares - Portal de datos abiertos del Ayuntamiento de Madrid.  
<https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=467>
- [20] Multas de circulación: detalle - Portal de datos abiertos del Ayuntamiento de Madrid.  
<https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=fb9>
- [21] Distritos municipales de Madrid - Geoportal del Ayuntamiento de Madrid.  
[https://geoportal.madrid.es/IDEAM\\_WBGEOPORTAL/dataset.iam?id=541f4ef6-762b-11e9-861d-ecb1d753f6e8](https://geoportal.madrid.es/IDEAM_WBGEOPORTAL/dataset.iam?id=541f4ef6-762b-11e9-861d-ecb1d753f6e8).
- [22] Barrios municipales de Madrid - Geoportal del Ayuntamiento de Madrid.  
[https://geoportal.madrid.es/IDEAM\\_WBGEOPORTAL/dataset.iam?id=422fa235-762b-11e9-861d-ecb1d753f6e8](https://geoportal.madrid.es/IDEAM_WBGEOPORTAL/dataset.iam?id=422fa235-762b-11e9-861d-ecb1d753f6e8).
- [23] Dotenv. <https://www.npmjs.com/package/dotenv>, February 2024.
- [24] Connection Pool | Sequelize. <https://sequelize.org/docs/v7/other-topics/connection-pool/>, June 2024.
- [25] Defining a Model | Sequelize. <https://sequelize.org/docs/v7/models/defining-models/>, June 2024.
- [26] Csv-parser. <https://www.npmjs.com/package/csv-parser>, December 2020.
- [27] Iconv-lite. <https://www.npmjs.com/package/iconv-lite>, May 2021.
- [28] booleanPointInPolygon | Turf.js. <https://turfjs.org/docs/api/booleanPointInPolygon>, .
- [29] Modals | React Bootstrap. <https://react-bootstrap.netlify.app/docs/components/modal>, .
- [30] CustomActiveShapePieChart | Recharts. <https://recharts.org/en-US/examples/CustomActiveShapePieChart>.

- [31] BrushBarChart | Recharts. <https://recharts.org/en-US/examples/BrushBarChart>, .
- [32] External state | React Leaflet. <https://react-leaflet.js.org/docs/example-external-state/>.
- [33] Layers control | React Leaflet. <https://react-leaflet.js.org/docs/example-layers-control/>.
- [34] Draggable Marker | React Leaflet. <https://react-leaflet.js.org/docs/example-draggable-marker/>.
- [35] Offcanvas | React Bootstrap. <https://react-bootstrap.netlify.app/docs/components/offcanvas>,
- .
- [36] Overlay | React Bootstrap. <https://react-bootstrap.netlify.app/docs/components/overlays>,
- .
- [37] Breakpoints. <https://getbootstrap.com/docs/5.3/layout/breakpoints/>, .
- [38] Navbars | React Bootstrap. <https://react-bootstrap.netlify.app/docs/components/navbar>,
- .
- [39] Pagination | React Bootstrap. <https://react-bootstrap.netlify.app/docs/components/pagination>,
- .
- [40] Validation | React Bootstrap. [https://react-bootstrap.netlify.app/docs/forms/validation/](https://react-bootstrap.netlify.app/docs/forms/validation), .



# Apéndice A

# Manual de Usuario

## A.1. Introducción

Bienvenido a UrbanNav Madrid, la herramienta diseñada para facilitar la comprensión de las dinámicas de movimiento urbanas y la identificación de zonas de riesgo en la ciudad. A través de gráficos, tablas y mapas interactivos, esta aplicación proporciona información detallada sobre accidentes, estacionamientos, tráfico, radares y multas.

## A.2. Navegación

La barra de navegación (Figura 30) sirve como guía a través de la aplicación, facilitando el acceso a todas las secciones y proporcionando una indicación visual activa que permite identificar en qué página se encuentra el usuario.



Figura 30: Barra de navegación

## A.3. Inicio

La página de inicio ofrece una visión general de las herramientas disponibles y una explicación de los datos que se encuentran en cada sección.

## A.4. Accidentes

En esta página se tratarán todos los datos asociados a los accidentes de tráfico. Estos datos se mostrarán en un mapa y en dos gráficos, complementados con una leyenda que proporciona información adicional sobre las zonas representadas. Al visitar esta página por primera vez o tras una actualización, se realizará una carga inicial mostrando 10 accidentes por barrio.

#### A.4.1. Mapa

Todos los mapas cuentan con un botón ‘Centrar Mapa’ que sitúa la vista en el centro de Madrid con un zoom estándar. Las zonas coloreadas en el mapa reflejan el nivel de riesgo basado en los accidentes. Se puede alternar la visualización entre ‘Barrios’ y ‘Distritos’ y, si hay menos de 500 accidentes, también se podrán visualizar marcadores individuales.

Al pulsar sobre una zona aparece un pop-up del cual puede que se muestren dos opciones: ‘Ver accidentes de este distrito/barrio’ y ‘Ver más información’. La primera opción solo aparecerá si la zona contiene menos de 100 accidentes. Al hacer clic sobre la opción ‘Ver accidentes de este distrito/barrio’ se ocultarán todas las zonas dibujadas del mapa y solo aparecerán marcadores de accidentes sobre la zona que hemos seleccionado. Debajo del botón ‘Centrar Mapa’ aparecerá uno nuevo que dirá ‘Volver a mostrar distritos/barrios’. Al hacer clic sobre el botón desaparecerán los marcadores y volverán a aparecer las zonas en el mapa. Al hacer clic en el botón ‘Ver más información’ al seleccionar una zona, se desplegará una ventana emergente que mostrará información sobre la zona como el nombre, el riesgo y el número de accidentes en base a la búsqueda aplicada y debajo aparecerá un formulario. Este formulario se explicará más adelante. La ventana emergente puede cerrarse al hacer clic en un botón en forma de X posicionado en la esquina superior derecha o al pulsar fuera de la propia ventana. Al hacer clic sobre el marcador de un accidente aparecerá la opción de ‘Ver más información’ y al seleccionarla se verá una ventana modal con información sobre el accidente y todas las personas implicadas en él. En este caso no aparecerá el formulario en la parte inferior de la ventana modal. Figura 31.

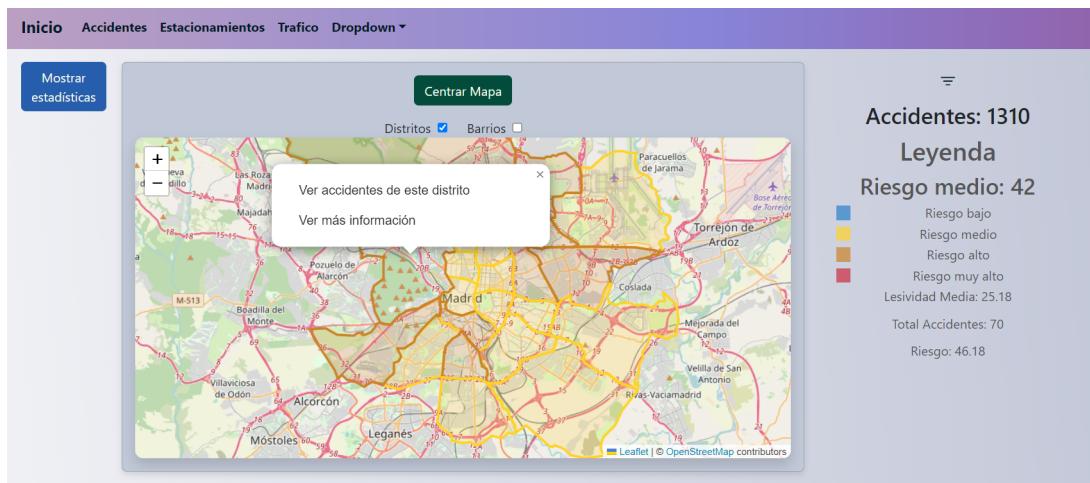


Figura 31: Mapa Accidentes

#### A.4.2. Flujo de accidentes

El formulario en la ventana emergente permite introducir dos fechas o dos horas para analizar el flujo de accidentes. Se debe cumplir que la primera fecha/hora sea anterior a la segunda para evitar errores. La gráfica resultante y el mapa adjunto permite examinar la distribución de accidentes por día o por hora. En caso contrario saldrá un mensaje de error por pantalla. Si se introducen 2 fechas correctas se mostrará una gráfica donde cada barra agrupa los accidentes ocurridos durante ese día. Además a la derecha de la gráfica se observa un mapa que muestra la zona dibujada en él. En el caso de haber seleccionado un distrito se mostrará dentro de él los barrios por los que está compuesto. Y si se hace clic sobre un barrio la gráfica se actualiza mostrando solo los datos de ese barrio. De esta manera se permite analizar el flujo de tráfico de las distintas zonas de un distrito. Al introducir 2 horas se muestra una gráfica donde cada barra representa todos los accidentes transcurridos durante esa hora todos los días registrados en la base de datos. Un ejemplo de esto se muestra en la figura 32, donde se ilustra el flujo de accidentes en el barrio Adelafas del distrito Retiro, en el rango horario de 19:30 a 23:44. La primera barra incluye los datos desde la hora de inicio hasta el comienzo de la hora siguiente, y la última barra abarca desde el inicio de la última hora hasta el final del periodo seleccionado. En la visualización proporcionada, la primera barra comprende los accidentes desde las 19:30 hasta las 20:00, y la última, los sucesos entre las 23:00 y las 23:44. Al colocar el cursor sobre una barra se observa el día/hora y la cantidad de accidentes registrados.

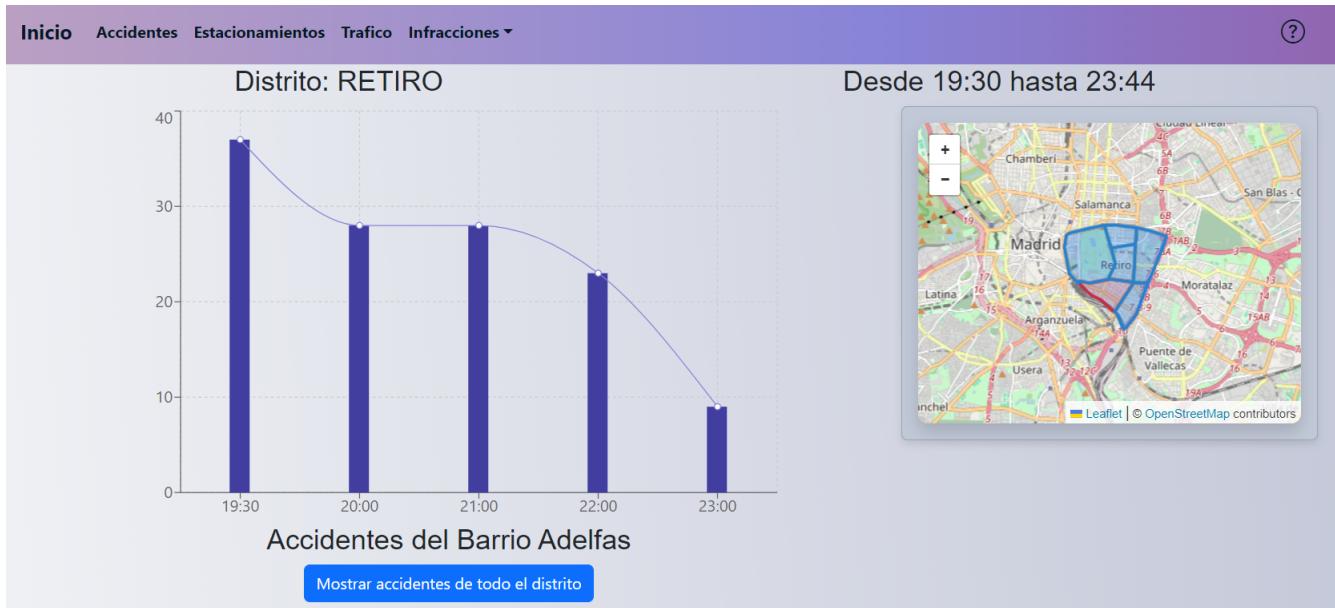


Figura 32: Flujo accidentes

#### A.4.3. Leyenda

La leyenda, ubicada adyacente al mapa, sirve como guía para interpretar la paleta de colores utilizada en la representación de las zonas. Los tonos más cálidos señalan áreas de mayor peligrosidad. Al posicionar el cursor sobre una zona específica o al hacer clic en ella, la leyenda proporcionará información detallada, incluyendo el número total de accidentes, el nivel de riesgo y la lesividad media. La lesividad media se mide en una escala de 0 a 100, reflejando el promedio de gravedad de los accidentes ocurridos. El valor del riesgo se incrementa proporcionalmente con la frecuencia de accidentes en la zona. Figura 33.

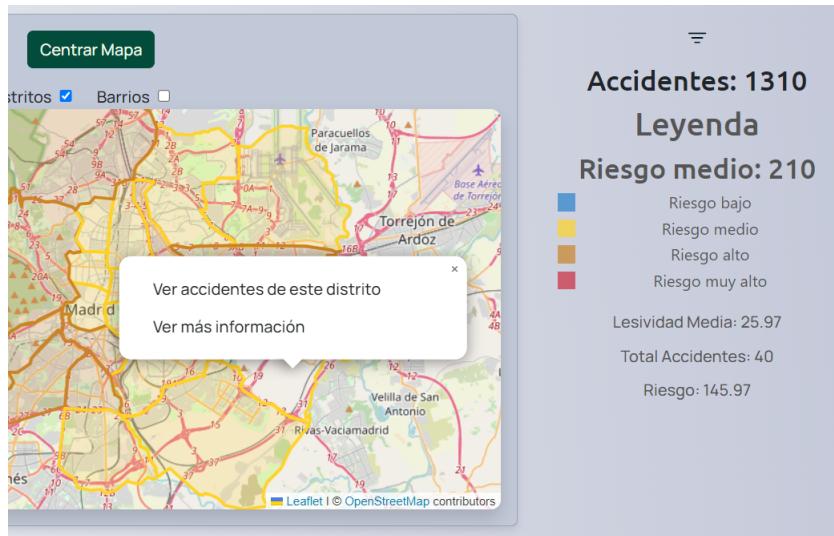


Figura 33: Leyenda Accidentes

#### A.4.4. Gráfica

Al seleccionar el ícono correspondiente a las gráficas, se desplegarán dos tipos distintos que categorizan los accidentes según diversas características. La gráfica de barras se enfoca en los atributos de las personas involucradas en los accidentes, mientras que la gráfica circular analiza aspectos inherentes a los propios accidentes. Al interactuar con una zona específica del mapa, ambas gráficas se actualizarán para reflejar los datos de accidentes pertinentes a esa área seleccionada. Ícono gráfica figura 34.



Figura 34: Gráfica Logo

#### A.4.5. Filtro

Al activar el ícono del filtro (figura 35), se desplegará un panel lateral que contiene un formulario detallado. Este panel puede cerrarse seleccionando el botón con forma de 'X' situado en la esquina superior derecha o haciendo clic fuera del área del menú. En la parte inferior del panel de filtro se encuentran dos opciones: 'Filtrar' y 'Limpiar filtro'. Al seleccionar 'Filtrar' sin completar los campos requeridos, se muestra una notificación indicando 'Datos incompletos'. Si los campos están correctamente llenados, se procederá con la búsqueda según los criterios establecidos.

rios especificados. Al elegir 'Limpiar filtro', aparecerá un mensaje (Figura 37) preguntando si se desea volver a la carga inicial de datos. Aceptar este mensaje reiniciará la carga de datos y limpiará el filtro. Si se selecciona cancelar, simplemente se borrarán los campos del formulario sin iniciar una nueva búsqueda. Figura 36.



Figura 35: Filtro Logo

Mostrar marcador arrastrable

Alcohol	Drogas	
Sin especificar	Sin especificar	
Vehículo	Accidente	
Sin especificar	Sin especificar	
Lesión:		
<input checked="" type="radio"/> Cualquiera		
<input type="radio"/> Leve		
<input type="radio"/> Grave		
<input type="radio"/> Fallecido		
Sexo	Clima	
Sin especificar	Sin especificar	
Edad		
Mínima	Máxima	
Desde	--:--	○
Desde	dd/mm/aaaa	□
Hasta	--:--	○
Hasta	dd/mm/aaaa	□
<b>Filtrar</b>		
<b>Limpiar filtro</b>		

Figura 36: Filtro Accidentes

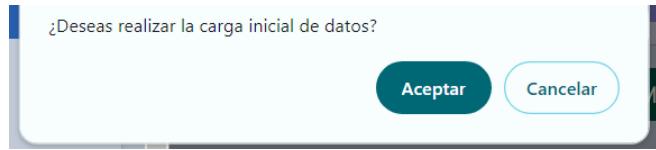


Figura 37: Mensaje Filtro

## A.5. Estacionamientos

La sección de Estacionamientos de UrbanNav Madrid permite explorar las áreas de aparcamiento de la ciudad, clasificadas por su propósito: residencial, rotacional, sanitario, para discapacitados, motocicletas, carga y descarga, y larga estancia. Al igual que en la sección de accidentes, la visualización inicial muestra 10 estacionamientos por barrio.

### A.5.1. Mapa

El mapa de estacionamientos comparte funcionalidades con otros mapas interactivos de la aplicación, como centrar la vista en Madrid o alternar entre la visualización de barrios y distritos. Si se muestran menos de 500 estacionamientos, aparecerá una opción para visualizar todos los aparcamientos disponibles. Al seleccionar una zona, se mostrarán todos los estacionamientos de esa área. Al hacer clic en un marcador, se desplegará un pop-up con detalles como el número de plazas disponibles, el tipo de uso y si el estacionamiento es en línea o en batería. Los marcadores se diferencian por colores según su tipo de uso. Figura 38.

### A.5.2. Leyenda

Ubicada junto al mapa, la leyenda facilita la interpretación de la clasificación de los estacionamientos según el color asignado a cada tipo de uso. Figura 38.



Figura 38: Leyenda y Mapa Estacionamientos

### A.5.3. Filtro

El filtro permite especificar el tipo de aparcamiento (en línea o batería) y el uso (rotacional, residencial, sanitario, para discapacitados, motocicletas, larga estancia, carga y descarga). También se puede filtrar por el número de plazas estableciendo un mínimo, un máximo o ambos. Si se desea buscar por un número de plazas concreto basta con introducir el mismo número como mínimo y máximo. En caso de que el valor mínimo ingresado sea mayor que el máximo, se mostrará un mensaje de error. El botón ‘Limpiar filtro’ permite la opción de vaciar los campos del filtro o realizar una nueva carga inicial de datos.

## A.6. Tráfico

La página de Tráfico ofrece una visión detallada de la densidad de tráfico en Madrid, con datos recopilados por estaciones distribuidas por toda la ciudad. Estos datos son esenciales para

comprender los patrones de movilidad y las áreas más transitadas. Al acceder a esta sección, la visualización inicial presentará el tráfico correspondiente a un mes específico.

#### A.6.1. Mapa

El mapa de tráfico incluye funcionalidades estándar y la opción adicional de visualizar las estaciones de tráfico. Las estaciones y las zonas se diferencian por colores que indican la densidad del tráfico. Al seleccionar una estación o zona, se abrirá una ventana modal con detalles relevantes y un formulario que permite definir un intervalo de tiempo para analizar el flujo de tráfico en la zona seleccionada. Figura 39.

#### A.6.2. Leyenda

La leyenda, situada junto al mapa, asigna un color a cada nivel de densidad de tráfico y proporciona información adicional sobre la zona seleccionada o señalada, como el nombre y el tráfico medio basado en los criterios de búsqueda aplicados. Figura 39.

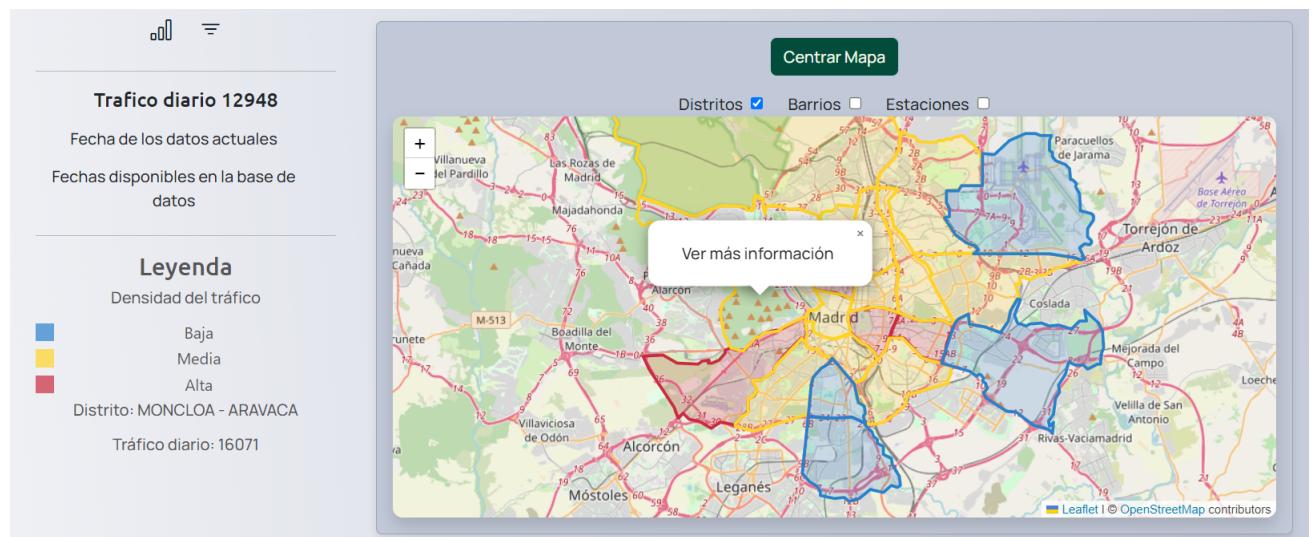


Figura 39: Leyenda y Mapa Tráfico

#### A.6.3. Flujo de tráfico

El análisis del flujo de tráfico funciona de manera similar al de los accidentes. En el formulario de la ventana modal, solo se podrá introducir dos fechas o dos horas. A continuación,

se mostrará una gráfica con barras que representan el tráfico registrado en la zona o estación durante el periodo especificado. Un mapa adicional ilustrará la ubicación exacta de la zona o estación. Si se trata de un distrito, se puede seleccionar una estación específica para obtener datos detallados de tráfico recopilados por esa estación.

#### A.6.4. Gráfica

Al hacer clic en el icono de la gráfica, se mostrará un gráfico que ordena distritos o barrios por densidad de tráfico, de mayor a menor. Al seleccionar una barra del gráfico, se activará un pop-up en el mapa que indica el nombre de la zona correspondiente. Figura 40.

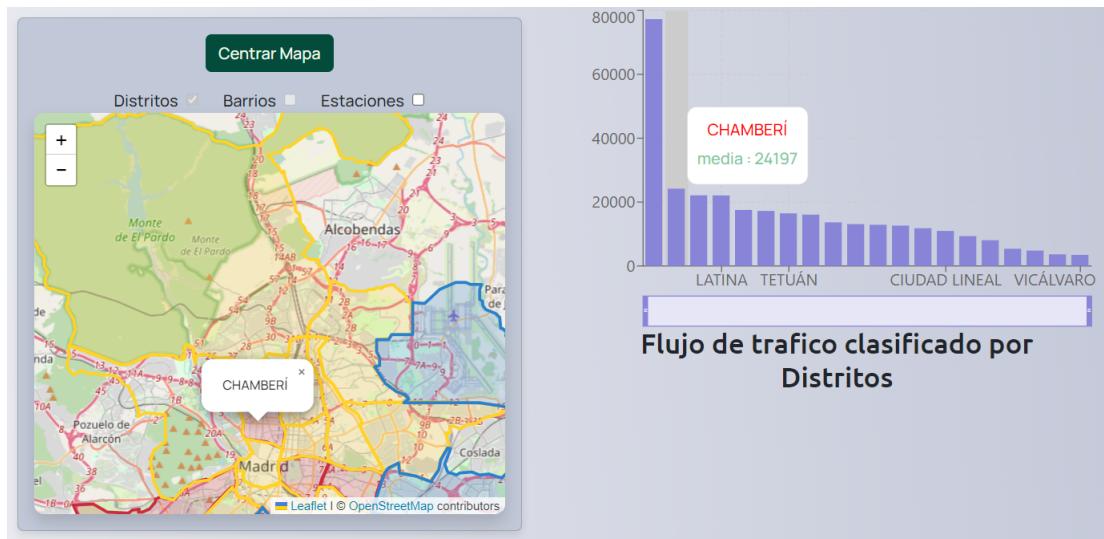


Figura 40: Gráfica Tráfico

#### A.6.5. Filtro

Al pulsar el botón del filtro saldrá un menú lateral similar al de accidentes con un filtro. En el filtro se puede introducir una fecha concreta, un periodo comprendido entre dos fechas o un mes. Al hacer clic sobre el botón de filtrado se realizará una búsqueda del tráfico registrado durante el periodo de tiempo seleccionado. Además se podrá seleccionar un sentido de tráfico concreto. Y al darle al botón 'Mostrar todo el tráfico' se mostrará el tráfico de cada zona con todos los datos registrados en el sistema. El botón 'Limpiar filtro' funciona igual que en el resto de filtros.

## A.7. Infracciones

La sección de Infracciones se divide en dos páginas interconectadas: Radares y Multas, que proporcionan información detallada sobre las infracciones de tráfico en Madrid.

### A.7.1. Radares

Esta página visualiza la ubicación de los radares fijos en Madrid y las multas registradas por cada uno.

- **Mapa:** El mapa resulta ser una herramienta útil para conocer dónde están situados los radares y cuáles capturan más infracciones. La visualización inicial muestra 50 multas asociadas a cada radar.
- **Gráfica:** Al hacer clic en el icono de gráfica, se muestra un gráfico que clasifica los radares por el número de multas registradas. Pasar el cursor sobre una barra proporciona información rápida del radar, y al hacer clic, se resalta el radar en el mapa con un marcador verde y se muestra una tabla con las multas asociadas. Figura 41.

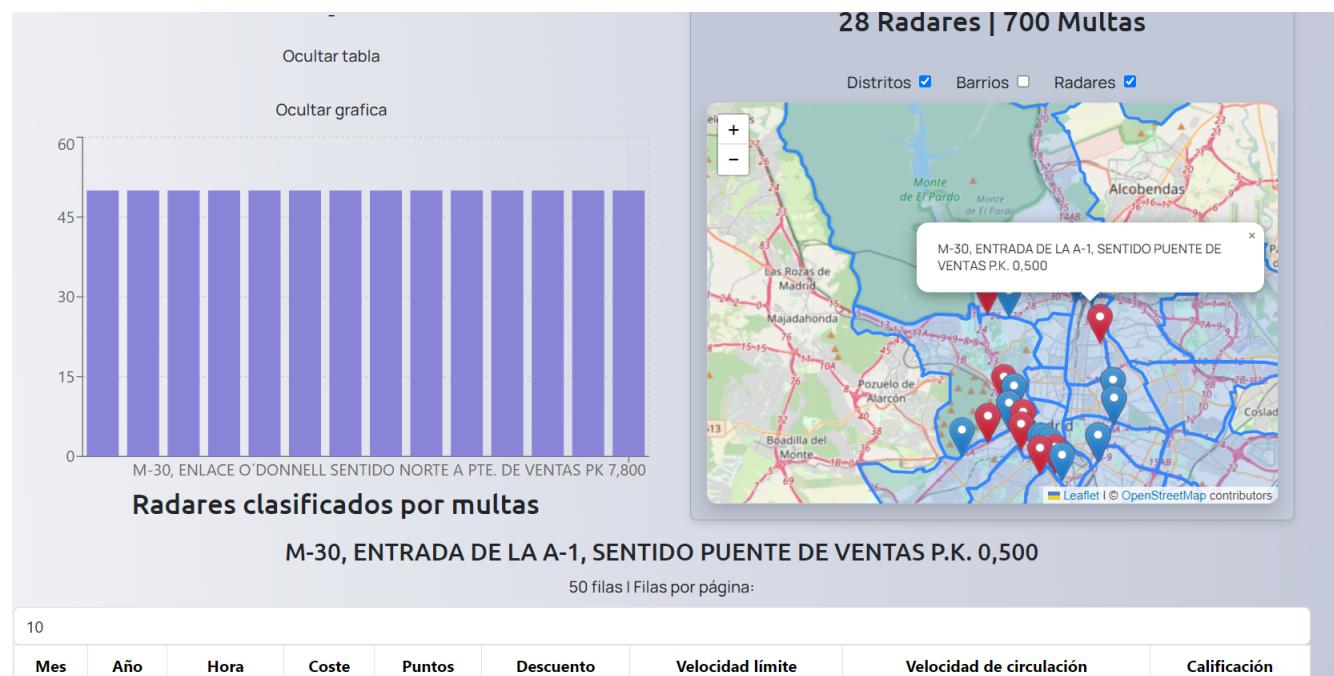


Figura 41: Gráfica Radares

- **Filtro:** El filtro permite buscar multas por criterios específicos como puntos infraccionados, aplicación de descuentos, coste de la multa, clasificación y hora de la multa. Este filtro también incluye la función ‘Limpiar filtro’ para restablecer los campos o realizar una nueva carga de datos y el manejo de errores de entrada.
- **Tabla:** La tabla detalla cada multa y permite ordenar las infracciones ascendente o descendente por cualquier atributo simplemente haciendo clic en la cabecera correspondiente. Figura 42

M-30, ENTRADA DE LA A-1, SENTIDO PUENTE DE VENTAS P.K. 0,500								
50 filas   Filas por página:								
10								
Mes	Año	Hora ↓	Coste	Puntos	Descuento	Velocidad límite	Velocidad de circulación	Calificación
12	2023	00:03:00	100	0	SI	90	96	GRAVE
12	2023	00:27:00	100	0	SI	90	97	GRAVE
12	2023	00:32:00	100	0	SI	90	97	GRAVE
12	2023	00:41:00	100	0	SI	90	117	GRAVE
12	2023	00:45:00	100	0	SI	90	96	GRAVE
12	2023	00:59:00	100	0	SI	90	99	GRAVE
12	2023	01:22:00	100	0	SI	90	107	GRAVE
12	2023	02:06:00	100	0	SI	90	95	GRAVE
12	2023	02:29:00	100	0	SI	90	96	GRAVE
12	2023	05:08:00	100	0	SI	90	101	GRAVE

« < 1 2 3 4 5 > »

Figura 42: Tabla Radares

### A.7.2. Multas

La sección de Multas ofrece acceso a una base de datos completa de todas las infracciones, permitiendo no solo ver las multas por exceso de velocidad sino también una variedad de otras infracciones.

- **Tabla:** Similar a la sección de Radares, esta tabla proporciona información adicional como el barrio, el motivo de la infracción y el denunciante. Se puede ordenar las multas según cualquier columna, ajustar el número de multas mostradas por página y navegar entre las distintas páginas de la tabla.

- **Filtro:** El filtro se ha enriquecido con la capacidad de buscar multas por motivo de infracción. Un botón adyacente al campo de búsqueda despliega ejemplos de infracciones para facilitar tu selección. Además, se puede filtrar las multas por barrio y denunciante para una búsqueda más específica.



# Apéndice B

# Manual de

# Instalación

Para ejecutar la aplicación en un entorno local, sigue estos pasos. Alternativamente, puedes acceder a la versión desplegada en la nube a través del enlace proporcionado.

**Requisitos previos.** Asegúrate de tener instalados Node.js y npm en tu ordenador. Si aún no los tienes, sigue estas instrucciones:

- **Descarga de Node.js:** Visita el sitio web oficial Node.js y descarga el instalador adecuado para tu sistema operativo.
- **Instalación de Node.js:** Ejecuta el instalador descargado y sigue las instrucciones en pantalla para completar la instalación.
- **Verificación de Node.js:** Abre un terminal y ejecuta node -v. Deberías ver la versión de Node.js instalada.
- **Verificación de npm:** En el mismo terminal, ejecuta npm -v para confirmar la versión de npm instalada junto con Node.js.

**Instalación de la aplicación.** Con Node.js y npm ya instalados, necesitarás el directorio que contiene la aplicación, que incluye dos carpetas principales: backend (servidor) y frontend (cliente), así como dos archivos .bat.

- **Instalación del Backend:** Abre el archivo installBack.bat. Se iniciará una ventana de terminal y comenzará la instalación de las dependencias. Una vez completada, deberías ver un mensaje indicando que el servidor se ha iniciado correctamente en <http://localhost:8000/>.
- **Instalación del Frontend:** Ejecuta el archivo installFront.bat. Tras la instalación, verás el mensaje 'Starting the development server...' indicando que el servidor del cliente se está ejecutando.

Para acceder a la aplicación en local, introduce `http://localhost:3000/` en la barra de direcciones de tu navegador.



UNIVERSIDAD  
DE MÁLAGA | **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática  
Bulevar Louis Pasteur, 35  
Campus de Teatinos  
29071 Málaga