UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Webapp para la monitorización de conjuntos de datos abiertos geolocalizados publicados en tiempo real

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Webapp para la monitorización de conjuntos de datos abiertos geolocalizados publicados en tiempo real

Autor: Judit Jiménez Jiménez

Tutor: Álvaro Prieto Ramos

Índice general de contenido

Índice de tablas

Índice de ilustraciones

Resumen

Ante los problemas de tráfico y polución que se dan en grandes ciudades como Madrid, es cada vez más común que muchas personas, para hacer frente a dicha situación, tomen como medio de transporte la bicicleta. Gracias a esto no solo ayudan a que se contamine menos, sino que los viajes se suelen hacer más rápido debido a que no hacen frente a los atascos, sin olvidar que es más económico que el transporte motorizado convencional.

En el caso concreto de Madrid, el servicio BiciMad presta bicicletas eléctricas en la ciudad de Madrid a aquellos ciudadanos suscritos en su plataforma. Se estima que en julio de 2019 el sistema tenía más de 65.000 usuarios con abono anual, superando los 14.000 usos diarios. Estos datos sugieren que este servicio ya es interesante para muchos ciudadanos de Madrid y que en el futuro lo puede ser para aún más ciudadanos, no solo en Madrid, sino en otras ciudades si este servicio se ofreciera en otras localidades.

Para ayudar a los ciudadanos interesados en el uso de este servicio, en este trabajo se presenta la aplicación web ***Muévete***. Esta aplicación hace uso de distintas APIs Open Data para proporcionar a sus usuarios información útil que facilite la planificación de sus recorridos.

Así, ***Muévete*** provee información sobre el estado y el nivel de ocupación de las estaciones de bicicletas que hay disponibles en Madrid junto con información meteorológica. Además, esta web también permite ver de forma geolocalizada actividades culturales y deportivas como forma de promover que la bicicleta no es solo un medio de transporte ecológico para ir al trabajo, sino que puede ser usado para desplazarse a cualquier tipo de actividad.

En las siguientes páginas recorreremos todo el proceso de desarrollo de esta aplicación web, desde los duros inicios de investigación hasta el resultado final. Espero que les guste este paseo, cojan el casco y el cortaviento y adentrémonos en esta aventura.

Capítulo 1.- Introducción

Motivación

Empezamos este recorrido, hablando de los inicios del TFG, antes incluso de que hubiese una propuesta encima de la mesa... qué hizo que este proyecto estuviese centrado en **web**, en objetos **geolocalizados** y por qué todo debía ser **Open Data**.

Durante todo el grado se estudian muchos campos y desde varias perspectivas. Al estudiar "Programación en Internet" fui más consciente de que el desarrollo web es un campo muy versátil, ya que es el más utilizado en el día a día de todos nosotros. ¿Quién, hoy en día, no dedica una hora de su vida a echar un vistazo a internet? Puesto que seguramente esta pregunta tenga muy pocas respuestas negativas, creo que seguir aprendiendo y hacer un proyecto desde cero utilizando nuevas tecnologías, podría ser muy enriquecedor.

Una vez que sabía que debía ser web, los únicos requisitos que tenía claros eran que debían ser objetos geolocalizados y en tiempo real. Mostrar y tratar objetos de esta índole siempre me ha parecido muy interesante, gracias en parte a proyectos desarrollados en la universidad, desde hacer unos ejercicios simples con PG PSQL en la asignatura "Programación en Bases de Datos" hasta desarrollar un pequeño proyecto simulando en tiempo real, el recorrido de un pájaro en la asignatura "Recuperación de la información y búsqueda en la Web".

Por último, dado el interés en trabajar con datos geolocalizados, parecía un paso natural hacer uso de Open Data dado que es la fuente más accesible a este tipo de datos. Además, uno de los objetivos también era hacer una aplicación que fuera capaz de hacer uso de distintas APIs de terceros. Al final hay mucho software en este mundo y si podemos contribuir para que cualquier persona pueda hacer uso de este y aprenda de esto por poco que sea, ya estaremos haciendo mucho bien.

Si juntamos todos estos ingredientes aparece Muévete, una web que hace uso de distintas APIs de Open Data para ofrecer datos en tiempo real (estado de estaciones de bicicletas, información meteorológica y actividades culturales y deportivas).

Alcance

Teniendo claro qué pretendemos hacer en este desarrollo, continuamos el camino aclarando con más detalle la magnitud que se pretende abarcar.

Como queremos que nuestro portal reúna mucha información de diferentes APIs, lo primero que estudiamos es si estas eran Open Data y si ofrecían los datos que nosotros necesitábamos para la idea que teníamos en mente.

En primer lugar, se pensó hacer un portal global en el que mostrar la flota de transportes públicos de una ciudad. Al ser pensada de forma global, revisamos exhaustivamente diferentes datasets de diferentes ciudades (Francia, Italia, Alemania, Estados Unidos, Zaragoza y Cáceres). El problema que vimos en esto es que cada ciudad ofrece los datos de forma distinta e incluso la forma de acceder a los mismos era diferente. Hay diversas variaciones entre ellas para lo que en una ciudad era acceder a un JSON y dentro del mismo al primer atributo, para otra era hacer varias peticiones a su servidor. Además de esto, cada ciudad presentaba estos datos de forma distinta, unos en formato json, otros en formato XML… y lo más importante de todo, una ofrecía toda la información en un mismo archivo y en otra había que hacer varias peticiones para encontrar el mismo dato. No había una sinergia y, sin un estándar específico, hacía complicado el desarrollo de esta primera idea.

Una vez visto esto, decidimos centrarnos en un tema menos explotado por otras aplicaciones que el transporte público y decidimos centrarnos en el uso de servicios de alquiler de bicicletas y toda la información complementaria que podía fomentar su uso. Así, se revisaron diferentes portales que exponen este servicio en distintas ciudades y se analizaron qué fortalezas y debilidades observábamos en ellas, siempre tratando de pensar qué valor podríamos aportar para hacer más completos y atractivos estos

servicios. De esta forma fuimos aclarando qué información queríamos ofrecer en nuestro servicio para que fuese interesante a los usuarios que utilizaban las mismas.

Nos decantamos por hacer más innovadora la web ofrecida por Madrid, por ser una ciudad referencia en este tipo de servicios y que, por su tamaño, ofrece la posibilidad de que cada vez más gente se anime a utilizar este medio de transporte. Vimos por tanto una oportunidad para que gente de todo el mundo pudiera verla (turismo) y también a promulgar que este medio se use debido a la contaminación que padece.

Para llevar a cabo la aplicación necesitábamos un desarrollo front en el que mostrar todo lo comentado anteriormente y también un desarrollo back con el que hacer las diferentes peticiones a las diferentes APIs que atacamos y obtener los datos necesarios.

Este desarrollo no dispone de una Base de Datos, ya que la mayoría de los datos son volátiles al tratarse de información en tiempo real obtenidas de las APIs de terceros. Tampoco veíamos necesario el almacenamiento de estos ya que no se iban a ofrecer servicios como la consulta de históricos de la información obtenida.

Objetivos

El objetivo de este proyecto es proporcionar de forma simple, clara y amigable los datos de las bicicletas disponibles en cada estación a los usuarios, aportando también datos meteorológicos y sobre actividades culturales y deportivas.

Se necesitará, por tanto, investigar el entorno con el que las diferentes APIs proporcionan la información que necesitamos. Para ello se debe:

- Buscar información acerca de las estaciones de bicicletas para ver cómo y qué información ofrecen.
- Buscar información del tiempo en aquella ciudad que encontremos datos de bicicletas.

- Buscar información de actividades que puedan ser interesante publicar en el portal para la ciudad que cumpla los requisitos anteriores.
- Toda esta información debe ser Open Data.
- Estudiar otras webs que aporten un valor similar y ver sus fortalezas y debilidades.
- Investigar herramientas para el desarrollo de este.

Una vez se obtengan los datos que se quieren mostrar, nos centraremos en los objetivos de nuestra aplicación web:

- Ofrecer toda la información de forma clara e intuitiva, priorizando su accesibilidad y usabilidad.
- Tomar como referencia las mejores características de las aplicaciones existentes para ofrecerlas en nuestra aplicación web estudiando si es posible añadirles aún más valor.
- Implementar todas las características finales haciendo código de calidad mediante la aplicación de buenas prácticas.

Capítulo 2. Estudio del arte

En esta sección, empezaremos introduciendo Open Data. A continuación, veremos diferentes herramientas que muestran datos geolocalizados, centrándonos en aquellas que muestran bicicletas para poder identificar sus fortalezas y debilidades. Después veremos cómo diferentes aplicaciones incluyen datos climatológicos en sus aplicaciones. Por último, investigaremos cómo diferentes portales ofrecen información sobre actividades.

Una vez que hayamos terminado este análisis, podremos ver cómo encajar todas estas piezas en nuestro portal.

Open Data

¿Qué es Open Data?

Open Data tiene como objetivo poner a disposición de los ciudadanos los datos de los que disponen las administraciones públicas y las instituciones.

Open Data significa “Datos Abiertos”, pero realmente los datos abiertos son datos que pueden ser utilizados, reutilizados y redistribuidos libremente por cualquier persona, y que se encuentran sujetos, cuando más, al requerimiento de atribución y de compartirse de la misma manera en que aparecen [1].

Los datos deben tener una serie de características:

- **Disponibilidad y acceso**: los datos deben estar accesibles, preferiblemente a través de Internet. El acceso debe poder hacerse a través un protocolo estándar que no suponga un coste adicional y los datos deben volcarse en un formato ampliamente extendido y que faciliten su procesamiento.

- **Reutilización y redistribución**: los datos deben ser distribuidos bajo unos términos que permitan su uso y explotación de forma libre y no pongan restricciones a cruzarlos con otros datos.

- **Participación universal**: todos los interesados, sin excepción, deben poder acceder a los datos y usarlos de la manera que deseen, sin ningún tipo de restricción comercial o de propósito acotado.

Existen otros movimientos que fomentan una filosofía similar, como el Open Source o el Open Knowledge. En última instancia lo que todas ellas pretenden es potenciar al máximo la innovación a través de la libertad y la colaboración.

Principios Open Data

El fin de semana del 7-8 de diciembre de 2007, treinta defensores del gobierno abierto se reunieron para desarrollar un conjunto de principios de datos de gobierno abierto. La reunión, celebrada en Sebastopol, California, fue diseñada para desarrollar una comprensión más sólida de por qué los datos abiertos del gobierno son esenciales para la democracia. [2]

El grupo ofrece un conjunto de principios fundamentales para los datos abiertos del gobierno. Al abrazar los ocho principios, los gobiernos del mundo pueden volverse más efectivos, transparentes y relevantes para nuestras vidas.

- **Completo**. Todos los datos públicos están disponibles. Los datos públicos son datos que no están sujetos a limitaciones válidas de privacidad, seguridad o privilegios.
- **Primaria**. Los datos se recopilan en la fuente, con el mayor nivel de granularidad posible, no en forma agregada o modificada.
- **Oportuna**. Los datos se ponen a disposición tan rápido como sea necesario para preservar el valor de los datos.
- **Accesible**. Los datos están disponibles para la más amplia gama de usuarios para la más amplia gama de propósitos.
- **Máquina procesable**. Los datos están razonablemente estructurados para permitir el procesamiento automatizado.

- **No discriminatorio.** Los datos están disponibles para cualquier persona, sin necesidad de registrarse.
- **No propietario.** Los datos están disponibles en un formato sobre el cual ninguna entidad tiene control exclusivo.
- **Sin licencia.** Los datos no están sujetos a ningún reglamento de derechos de autor, patente, marca registrada o secreto comercial. Se pueden permitir restricciones razonables de privacidad, seguridad y privilegios.

Tipos de datos

Un conjunto de datos se organiza en datasets para que sean más sencillo de localizar y consultar. Al conjunto de datos se le conoce como datasets. Dentro del open data tenemos varios tipos de datasets, los cuales se pueden organizar en diferentes conjuntos haciendo referencia el tema que tratan. En la figura 2.1 podemos encontrar los datasets más comunes.

Figura 2.1. Tipos de Datasets

Los datasets también se encuentran clasificados dependiendo de la facilidad con la que se puedan reutilizar. Este sistema de calificación fue propuesto por Tim Berners-Lee, fundador de la Word Wide Web [3].

Esta clasificación expone lo siguiente [4]:

- **Una estrella**: Publica datos en la Web en cualquier formato (por ejemplo, PDF, JPEG) acompañado de una licencia abierta explícita (expresión de derechos).
- **Dos estrellas**: Publica datos estructurados en la Web en un formato legible por máquina (por ejemplo, XML).
- **Tres estrellas**: Publica datos estructurados en la Web en un formato de datos documentado y no patentado (por ejemplo, CSV, KML).
- **Cuatro estrellas**: Publica datos estructurados en la Web como RDF (por ejemplo, Turtle, RDFa, JSON-LD, SPARQL)
- **Cinco estrellas**: En su RDF, haga que los identificadores sean enlaces (URL) a fuentes de datos útiles.

Estudio de mercado

Seguimos nuestra ruta analizando las diferentes herramientas que podemos encontrar sobre estaciones de bicicletas, sobre cómo integran datos climatológicos en las webs y el listado de actividades. A partir de este análisis, obtendremos las conclusiones y objetivos propuestos para la aplicación con el fin de introducir funcionalidades innovadoras.

Primero buscaremos portales web que faciliten información acerca de las estaciones de bicicletas de sus ciudades, para poder ver qué datos muestran y cómo lo hacen. Los sitios que se han estudiado son BiciMad [5], BiziZaragoza y DublinBikes.

BiciMad

La aplicación que se encarga de mostrar los datos de las estaciones en la ciudad de Madrid es BiciMad. Esta aplicación muestra un mapa y a simple vista hace una diferencia por nivel de ocupación de las estaciones. A continuación, en la figura 2.2, podemos ver cómo aporta esta información el portal BiciMad.

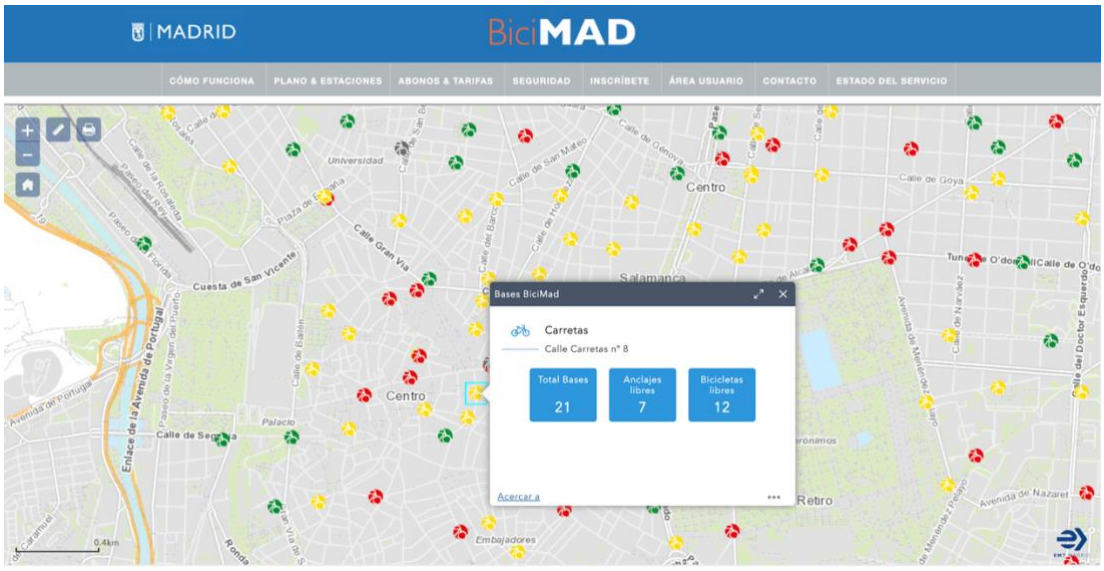

Figura 2.2. Portal BiciMad

Una vez que hacemos clic en una de las estaciones, podemos ver más información detallada del estado de la estación. Dónde se encuentra ubicada, cuántas bicicletas libres quedan, cuántos anclajes libres tenemos y el total de bases que hay.

Como **fortalezas** en esta web destacamos:

- Distinción de las estaciones por colores.
- Información más detallada al hacer clic en el marcador.
- Sencilla.
- Ofrece Open Data.

Como **debilidades** a su vez vemos:

- No queda claro a qué corresponden los colores, si a aquellos que tienen muchos anclajes libres o si hay muchas bicicletas.
- No nos da la opción de cambiar la vista de los colores si lo que queremos es dejar una bicicleta en vez de cogerlas.

Vemos que esta web no integra datos de tiempo ni tampoco de actividades, por lo que queda claro que podemos aportar valor añadiendo esto.

BiziZaragoza

Esta aplicación nos muestra la información de las bicicletas en la ciudad de Zaragoza [6]. Vemos que esta web es bastante completa, a simple vista vemos un mapa y en él las estaciones, también tenemos una breve información del color de las estaciones. Se adjunta una imagen en la que podemos ver el portal de BiziZaragoza, figura 2.3.

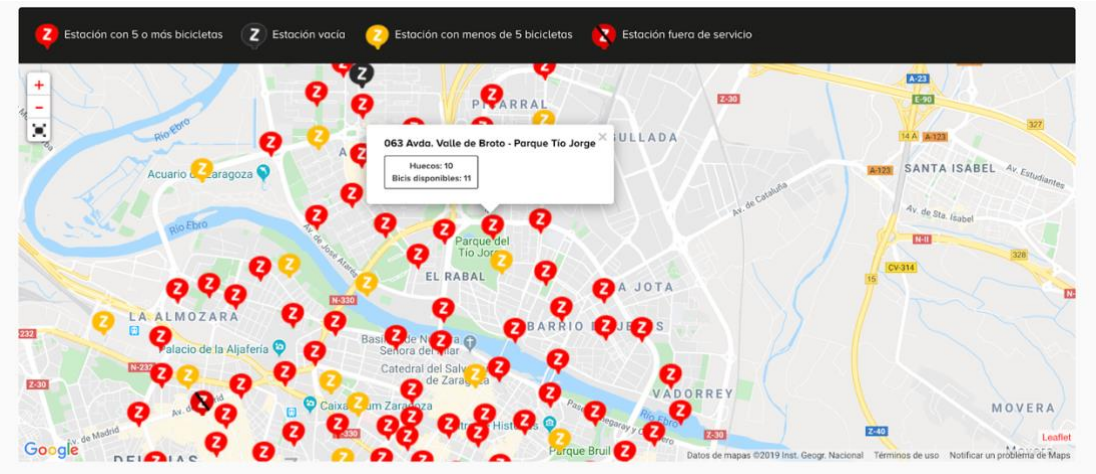

Figura 2.3. Portal BiziZaragoza

Una vez hacemos clic sobre uno de los marcadores, podemos ver más información de la estación en concreto.

Como **fortalezas**:

- Pequeña leyenda indicando el por qué del color de cada marcador.
- Información de la estación al hacer clic sobre ella.
- Ofrece Open Data.

Como **debilidades**:

- El diseño de este no es el más adecuado, los colores elegidos no quedan claros. Parece que el rojo son los no disponibles cuando es justo lo contrario.
- La información dentro del marcador se presenta de una forma tosca.

En esta web tampoco disponen de datos relativos al tiempo, ni de actividades a las que te puedas apuntar.

DublinBikes

Esta web se hace cargo de todas las estaciones de bicicletas que se encuentran en la ciudad de Dublín [7].

Al meternos en ella vemos que hay un pequeño mapa en el centro con información acerca de las mismas. También hace diferencia entre las estaciones por colores. En la figura 2.4 podemos ver la página web que expone DublinBikes para mostrar dicha información.

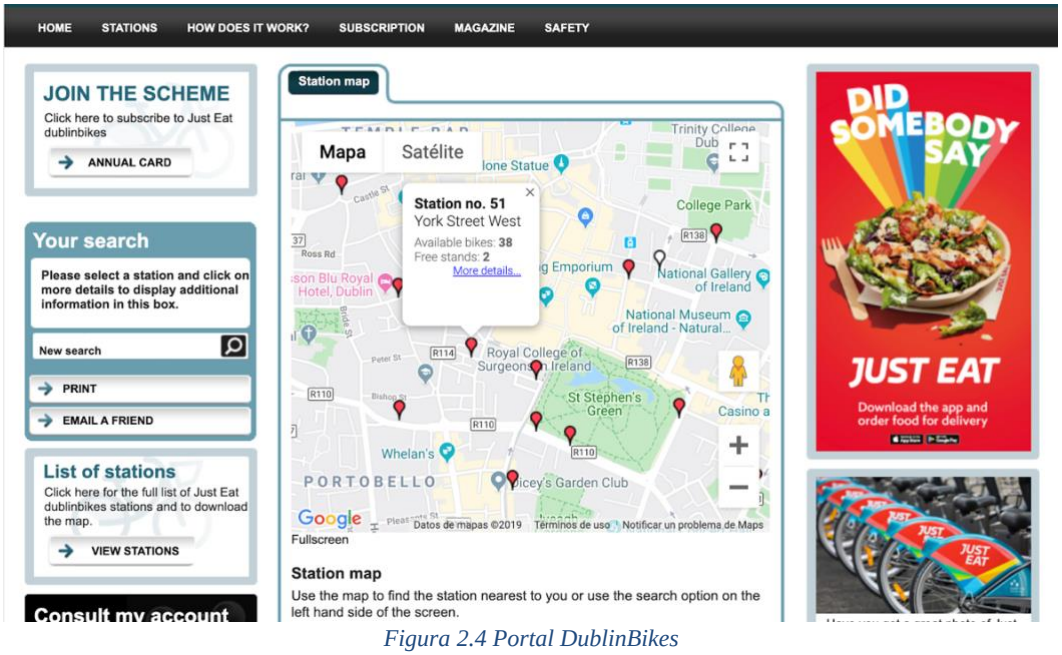

Figura 2.4 Portal DublinBikes

Una vez que hacemos clic sobre una de las estaciones y al igual que en las anteriores webs, podemos ver más información acerca de las mismas.

Como **fortalezas** en esta web destacamos:

- Información acerca de las estaciones.

Como **debilidades** en esta web destacamos:

- Mucha publicidad.
- No se centra en el mapa.
- Diseño pobre.

Tras ver las anteriores, parece que debemos centrarnos más en las otras páginas analizadas que en la web de Dublín.

Conclusiones

Con este análisis hemos podido revisar varias webs dedicadas a visualizar datos de bicicletas, viendo qué fortalezas y qué debilidades tienen. De esta forma, vemos qué sinergias debe tener la web que queremos hacer. Nos hemos dado cuenta de que al no incorporar ninguna de ellas datos sobre el tiempo o de actividades, creemos que vamos por buen camino y que la web resultará de interés.

Por otro lado, tanto Madrid como Zaragoza exponen sus datos de forma abierta utilizando la filosofía Open Data. Por lo que también vemos que está en pleno auge y que cada vez más ciudades se apuntan a publicar sus datos para que la ciudadanía pueda hacer uso de ella.

Vemos que todas las webs tienen un diseño un tanto tosco, la información que se detalla en los marcadores no está muy refinada y que los colores que suelen utilizar no son los más apropiados. Esto nos anima a hacer una web mucho más intuitiva y siguiendo las últimas tendencias en cuanto a estilo, para que el usuario se sienta más cómodo usándola.

Además, ninguna de ellas añade información del tiempo, algo que consideramos muy útil para cualquier persona que vaya a usar una bicicleta. Probablemente esta información puede ser de interés para los usuarios incluso antes de consultar si hay bicicletas disponibles en la estación.

Si a esto le unimos la posibilidad de ver información de actividades disponibles en el entorno donde van a usar la bicicleta, estaremos ofreciendo un portal más completo y que contribuya a animar a los ciudadanos a participar en las mismas yendo a ellas de un modo sano, económico y ecológico.

Teniendo tanto datos abiertos de Zaragoza como Madrid a nuestra disposición, vemos que Madrid nos aporta más información sobre las actividades. Contamos con el plus de que es la capital de España y, por lo tanto, será más utilizada. Nos centraremos en mejorar esta aplicación web, para que más ciudadanos puedan beneficiarse de la nueva información que les daremos.

Como conclusión podríamos decir que la web aportará valor al ciudadano. Hemos comprobado que no muchas ciudades disponen de esta información, por lo que podemos decir que será una aplicación novedosa. Igualmente, nos apoyaremos en los puntos fuertes de todas las webs que hemos analizado y trataremos de evitar caer en las debilidades de estas.

Capítulo 3. Muévete

La aplicación web Muévete promueve el transporte limpio y ecológico en la ciudad de Madrid. Además de promover el deporte y las actividades culturales y de ocio propias de la ciudad.

Muévete surge gracias a que varios organismos siguen la filosofía Open Data, tales como AEMet, EMT y el Ayuntamiento de Madrid. Estos organismos ponen a disposición de los ciudadanos datos geolocalizados y de interés para muévete. Estos datos serán pertenecientes al sector de transporte, medio ambiental y de cultura.

Muévete solicitará datos a las diferentes APIs para luego poder mostrarlos de forma clara e intuitiva.

Lo primero que hará será pedir la información de las bicicletas. Al ser en tiempo real, no se almacenará ningún dato. Una vez que se recibe, se muestra en el mapa clasificando los marcadores, permitiendo que el usuario elija cómo lo quiere ver. Por defecto verá las estaciones clasificadas por disponibilidad de bicicletas (roja baja disponibilidad, amarillo media disponibilidad, verde alta disponibilidad), pero se dará la opción de que también pueda ver estos colores por disponibilidad de anclajes.

Además, en todo momento se podrá ver el tiempo que hace actualmente en la ciudad, la dirección del aire y la velocidad de este. Esta información también será mostrada para la hora siguiente, ofreciendo una previsión de esta. De esta forma nos aseguramos de que, de una simple consulta, el usuario disponga de toda la información posible para que pueda viajar en este medio de transporte tan dependiente de la meteorología como es la de las bicicletas.

Por otro lado, el usuario también podrá ver una agenda deportiva, que mostrará un listado de actividades de esta índole, también clasificadas por deporte. En esta vista tendrá información del nombre de la actividad y también de su hora de inicio y fin,

además se dará la opción de que a través de un enlace poder ir a la web oficial para poder inscribirse y ver más información.

Para hacer que la aplicación sea más completa, se ofrecerán también actividades culturales geolocalizadas en todo Madrid. Estas actividades se dibujarán en un mapa que, al hacer clic sobre el marcador, permitirá visualizar información más detallada de la actividad. Del mismo modo que con las actividades deportivas, también se mostrará el enlace que permite ir al sitio oficial para inscribirse en la misma.

En resumen, Muévete es un portal web que principalmente se centra en poder ver información sobre estaciones de bicicletas junto con datos meteorológicos de un simple vistazo. Además, tratando de promover las actividades culturales y deportivas que se llevan a cabo en la ciudad, fomentando de esta manera el deporte y la salud medioambiental de la ciudad de Madrid. Todo esto de una forma sencilla, clara e intuitiva.

Capítulo 4. Análisis y diseño

En este apartado hablaremos de la fase de Análisis y Diseño del desarrollo software. Antes de empezar a codificar la aplicación software, es de suma importancia que se tenga una completa y plena comprensión de los requisitos y del comportamiento de esta.

Pressman establece que la tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado y especificación [8].

Requisitos

Empezaremos definiendo qué requisitos queremos que tenga nuestra aplicación.

Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer [9].

Para Muévete, los requisitos funcionales serían los siguientes:

- El sistema deberá permitir obtener información Open Data de los portales del tiempo (AEmet), del servicio de transporte de Madrid (EMT) y de las actividades de Madrid (portal Open Data del Ayuntamiento de Madrid).
- Poder visualizar el mapa de la ciudad y en él las estaciones de bicicletas de Madrid en tiempo real.

- Poder tener información acerca del estado de las estaciones, poder ver cuántas bicicletas quedan libres y ver cuántos anclajes están libres al hacer clic sobre cualquiera de ellas.
- Poder seleccionar si lo que queremos ver son los marcadores por anclajes libres o por bicicletas.
- Poder diferenciar a simple vista qué estaciones están más llenas de bicicletas y cuáles menos si tenemos seleccionado que queremos ver bicicletas (escala de colores).
- Poder diferenciar a simple vista qué estaciones tienen más anclajes libres y cuáles menos si tenemos seleccionado que queremos ver anclajes (escala de colores).
- Poder visualizar información relativa a datos climatológicos en tiempo real.
- Poder ver una predicción del tiempo a una hora vista haciendo clic sobre la información del tiempo.
- Tener varias vistas, una con un mapa para ver información de las bicicletas, otra para ver actividades globales y otra con una agenda deportiva.
- Poder ver un listado de todas las actividades deportivas que se llevan a cabo en la ciudad. También debemos poder ofrecer información para cada una de ellas y tener un enlace que nos lleve a la actividad en particular.
- Poder ver sobre un mapa las actividades de otra índole que se llevan a cabo en Madrid. También debemos poder ofrecer información para cada una de ellas y tener un enlace que nos lleve a la actividad en particular.

Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad [10].

Para Muévete, los requisitos no funcionales serían los siguientes:

- El sistema deberá ser multiplataforma, es decir, deberá funcionar correctamente en los navegadores de las distintas plataformas: ordenador, Tablet, dispositivo móvil.
- El sistema deberá funcionar en la mayoría de los navegadores web: Google Chrome, Mozilla Firefox, Opera…
- La interfaz deberá ser clara, amigable y sencilla. La curva de aprendizaje del usuario con la aplicación deber ser lo más amena posible.
- El tiempo de respuesta será breve y en todo momento se le dará un feedback al usuario en caso de que pueda llevar más tiempo.
- La información que se presenta en toda la web debe ser limpia y correcta, respetando en todo momento al usuario.
- Priorizar el uso de APIs evitando la descarga en cualquier formato y posterior tratamiento de los datasets abiertos por parte de la aplicación.

Casos de uso

Un caso de uso es una secuencia de interacciones entre un sistema y alguien o algo que usa alguno de sus servicios. Un caso de uso es una forma de expresar cómo alguien o algo externo a un sistema lo usa. Cuando decimos “alguien o algo” hacemos referencia a que los sistemas son usados no sólo por personas, sino también por otros sistemas de hardware y software [9].

Descripción de actores

Los actores representan el papel de los futuros usuarios del sistema. Los actores modelan la perspectiva del usuario para saber cómo podría interactuar con el software desarrollado.

En nuestra aplicación tendremos únicamente usuarios anónimos. Estos usuarios querrán conocer la información que ofrecemos de forma pública y sin

necesidad de que se registren, por esto decimos que son anónimos. Este usuario podrá acceder a toda la información, ver la información de las estaciones de bicicletas, ver las actividades (tanto de ocio como las deportivas) y ver el tiempo.

También tendremos un administrador del sistema, el cual será el encargado de que, si alguna API cambia la forma de ofrecer sus datos, haga el cambio pertinente.

Diagrama de casos de usos

En este apartado se presentan los diferentes casos de uso que se pueden dar para los actores descritos con anterioridad. En el dibujo (figura 4.1) se incluye la funcionalidad de lo que el sistema deberá hacer.

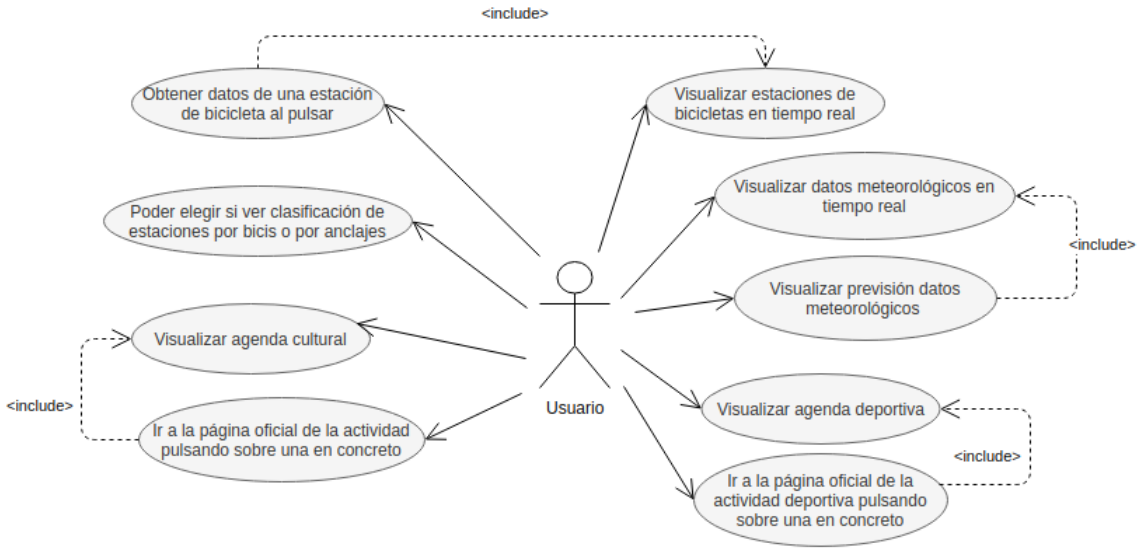

Figura 4.1: Casos de uso para un usuario

Fichas de los casos de uso

Tras el diagrama de casos de uso, en esta nueva parada de nuestro camino por Muévete, mostramos una especificación de cada caso de uso en forma de tablas. En estas podremos encontrar una descripción y los pasos que debe seguir nuestro usuario para darla por completada.

Visualizar las estaciones de bicicletas

<table>
<tr><td>Caso de Uso</td><td>Visualizar estaciones de bicicletas</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver en tiempo real el estado de las estaciones de bicicletas de Madrid de forma geolocalizada en un mapa. Esta información será presentada por colores, siendo verde alta disponibilidad de bicicletas, amarillo medio y de color roja indicando disponibilidad baja.</td></tr>
<tr><td>Precondición</td><td>Carga de la información referente a las estaciones disponible en el API del EMT.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>Se verá en un mapa todas las estaciones geolocalizadas que hay en Madrid. Esta información estará clasificada por colores.</td></tr>
</table>

Tabla 4.1. Caso de Uso: Visualizar estaciones de bicicletas

Visualizar las estaciones de bicicletas clasificadas por anclajes

<table>
<tr><td>Caso de Uso</td><td>Visualizar estaciones de bicicletas clasificadas por disponibilidad de anclajes.</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver en tiempo real el estado de las estaciones de bicicletas de Madrid de forma geolocalizada en un mapa. Esta información será presentada por colores, siendo verde alta disponibilidad de anclajes, amarillo medio y de color roja indicando disponibilidad baja.</td></tr>
<tr><td>Precondición</td><td>Carga de la información referente a las estaciones disponible en el API del EMT.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr><tr><td>2</td><td>Hacer clic en “Ver Anclajes”</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>Se verá en un mapa todas las estaciones geolocalizadas que hay en Madrid. Esta información estará clasificada por colores dependiendo de la disponibilidad que se tenga por anclajes.</td></tr>
</table>

Tabla 4.2. Caso de Uso: Visualizar estaciones de bicicletas clasificadas por disponibilidad de anclajes

Obtener información de una estación en particular

<table>
<tr><td>Caso de Uso</td><td>Obtener información de una estación en particular.</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver en tiempo real el estado de la estación de bicicletas de Madrid que haya seleccionado pulsando sobre la estación interesada.</td></tr>
<tr><td>Precondición</td><td>Carga de la información referente a las estaciones disponible en el API del EMT. Se muestra información de las estaciones.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr><tr><td>2</td><td>Hacer clic en un marcador</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>Aparecerá información relativa a la estación que se ha seleccionado. Se especificará: ubicación de la estación (calle dónde se encuentra), nombre de la estación, número de bicicletas libres y número de anclajes libres.</td></tr>
</table>

Tabla 4.3. Caso de Uso: Obtener información de una estación en particular

Visualizar datos meteorológicos

<table>
<tr><td>Caso de Uso</td><td>Visualizar datos meteorológicos</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver el tiempo actual (en grados Celsius), ver un icono indicando el estado del cielo y también datos referentes al viento (velocidad y orientación).</td></tr>
<tr><td>Precondición</td><td>Llama al API de AEMet para obtener los datos relativos al tiempo.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>El usuario podrá ver a la derecha de la barra de navegación un icono indicando el estado del cielo, seguido del a temperatura en tiempo real de la ciudad en grados Celsius. A continuación, verá información del viento, orientación y velocidad.</td></tr>
<tr><td>Excepciones</td><td>En caso de entrar en la web de 1:00 de la mañana hasta las 6:00 de la mañana, se verá una luna en el icono y no se dispondrá de esta información, ya que AEMet solo ofrece la información de 7 de la mañana a 12 de la noche.</td></tr>
</table>

Tabla 4.4. Caso de Uso: Visualizar datos meteorológicos

Visualizar previsión de los datos meteorológicos

<table>
<tr><td>Caso de Uso</td><td>Visualizar la previsión de los datos meteorológicos</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver el tiempo (en grados Celsius) estimado por AEMet en la siguiente hora a la que se encuentre, ver un icono indicando el estado del cielo en la hora siguiente y también datos referentes al viento (velocidad y orientación).</td></tr>
<tr><td>Precondición</td><td>Llama al API de AEMet para obtener los datos relativos al tiempo.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr><tr><td>2</td><td>Pulsar sobre cualquier información del tiempo de la barra de navegación.</td></tr></table></td></tr>
<tr><td>Postcondición</td><td>El usuario podrá ver a la derecha de la barra de navegación un icono indicando una previsión del tiempo, se verá el estado del cielo, seguido de la temperatura de la ciudad en grados Celsius. A continuación, verá información del viento, orientación y velocidad.</td></tr>
<tr><td>Excepciones</td><td>En caso de entrar en la web de 1:00 de la mañana hasta las 6:00 de la mañana, se verá una luna en el icono y no se dispondrá de esta información, ya que AEMet solo ofrece la información de 7 de la mañana a 12 de la noche.</td></tr>
</table>

Tabla 4.5. Caso de Uso: Visualizar la previsión de los datos meteorológicos

Visualizar agenda deportiva

<table>
<tr><td>Caso de Uso</td><td>Visualizar agenda deportiva</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder ver un listado con todas las actividades deportivas que se llevan a cabo a lo largo del año actual en Madrid. Estas actividades estarán clasificadas por temática. //AÑADIR LEYENDA</td></tr>
<tr><td>Precondición</td><td>Llama al API del ayuntamiento de Madrid para obtener las actividades.</td></tr>
<tr><td>Secuencia normal</td><td><table><tr><td>PASO</td><td>ACCIÓN</td></tr><tr><td>1</td><td>Acceder a la web de Muévete.</td></tr><tr><td>2</td><td>Ir a la sección de “Agenda Deportiva”</td></tr></table></td></tr>
</table>

Postcondición	El usuario podrá ver un listado con las actividades deportivas. Estas actividades incluyen nombre, dónde se llevará a cabo y la hora de inicio y fin de esta. El listado tendrá una clasificación por colores y por iconos.
Excepciones	En caso de que el API del ayuntamiento tarde mucho en mandar los datos y nuestra aplicación en cargarlos, se verá un spinner indicando la carga de las actividades.

Tabla 4.6. Caso de Uso: Visualizar agenda deportiva

Acceder a la página oficial de la actividad deportiva

Caso de Uso	Acceder a la página oficial de la actividad deportiva
Descripción	Permite a cualquier usuario que acceda a la web poder acceder a la página oficial de una actividad para poder apuntarse o ver más información.
Precondición	Llama al API del ayuntamiento de Madrid para obtener las actividades.
Secuencia normal	

PASO	ACCIÓN
1	Acceder a la web de Muévete.
2	Ir a la sección de “Agenda Deportiva”.
3	Pulsar sobre una de ellas.

Postcondición	El usuario será redirigido a la página oficial del Ayuntamiento para que pueda ver más información de la actividad.

Tabla 4.7. Caso de Uso: Acceder a la página oficial de la actividad deportiva

Visualizar actividades culturales

Caso de Uso	Visualizar actividades culturales
Descripción	Permite a cualquier usuario que acceda a la web poder ver un listado con todas las actividades relacionadas con el ocio y la cultura que se llevan a cabo a lo largo del año actual en Madrid. Estas actividades serán mostradas en un mapa y estarán geolocalizadas.
Precondición	Llama al API del ayuntamiento de Madrid para obtener las actividades.

<table>
<tr><td>Secuencia normal</td><td><table>
<tr><td>PASO</td><td>ACCIÓN</td></tr>
<tr><td>1</td><td>Acceder a la web de Muévete.</td></tr>
<tr><td>2</td><td>Ir a la sección de “Actividades”</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td>El usuario podrá ver en un mapa todas las actividades geolocalizadas.</td></tr>
<tr><td>Excepciones</td><td>En caso de que el API del ayuntamiento tarde mucho en mandar los datos y nuestra aplicación en cargarlos, se verá un spinner indicando la carga de las actividades.</td></tr>
</table>

Tabla 4.8. Caso de Uso: Visualizar actividades culturales

Acceder a la página oficial de la actividad cultural

<table>
<tr><td>Caso de Uso</td><td>Acceder a la página oficial de la actividad cultural.</td></tr>
<tr><td>Descripción</td><td>Permite a cualquier usuario que acceda a la web poder acceder a la página oficial de una actividad para poder apuntarse o ver más información.</td></tr>
<tr><td>Precondición</td><td>Llama al API del ayuntamiento de Madrid para obtener las actividades.</td></tr>
<tr><td>Secuencia normal</td><td><table>
<tr><td>PASO</td><td>ACCIÓN</td></tr>
<tr><td>1</td><td>Acceder a la web de Muévete.</td></tr>
<tr><td>2</td><td>Ir a la sección de “Actividades”.</td></tr>
<tr><td>3</td><td>Pulsar sobre una de ellas.</td></tr>
<tr><td>4</td><td>Pulsar en “saber más”.</td></tr>
</table></td></tr>
<tr><td>Postcondición</td><td>El usuario será redirigido a la página oficial del Ayuntamiento para que pueda ver más información de la actividad.</td></tr>
</table>

Tabla 4.9. Caso de Uso: Acceder a la página oficial de la actividad cultural.

Diagramas UML

Antes de mostrar el diagrama UML de nuestra aplicación, explicaremos brevemente qué es UML.

Como hemos visto en la asignatura de "Ingeniería Software", UML es un lenguaje de modelado que consiste en un conjunto de vistas, diagramas y símbolos, así como una serie de reglas que indican cómo utilizar los elementos. El objetivo de UML es establecer un lenguaje visual de modelado, expresivo y sencillo.

Hay varios tipos de diagramas UML clasificados según lo que muestran, en este documento, con anterioridad hemos mostrado los diagramas de casos de uso (apartado anterior), a continuación, mostraremos los diagramas de clases y de actividad.

Diagrama de clases

En este apartado se muestra el diagrama de clases (figura 4.2) que tiene nuestra aplicación y las relaciones entre ellas.

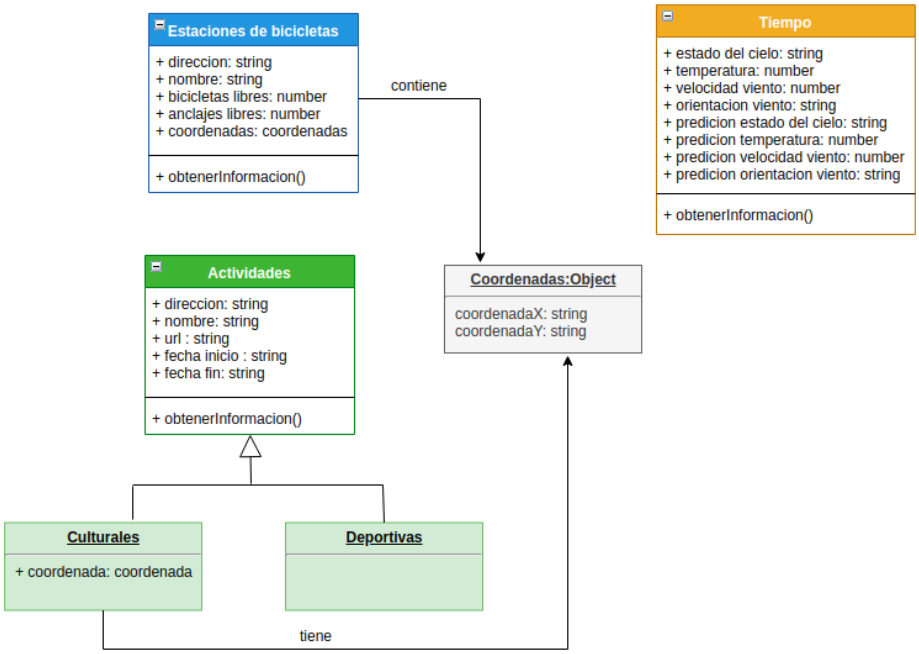

Figura 4.2. Diagrama de clases Muévete

Diagrama de actividad

Los diagramas de actividad muestran actividades ejecutadas por el sistema, detallando el flujo y los posibles caminos que podemos ir encontrando. Nos centraremos en tres flujos que tenemos en nuestra aplicación, se han seleccionado estas tres porque engloban flujos mucho más sencillos y también por semejanza con otras.

Diagrama de actividad - Ver estaciones de bicicletas geolocalizadas

En este subapartado, mostraremos en la figura 4.3, el diagrama de actividad que corresponde con el caso de uso “Visualizar las estaciones de bicicletas”.

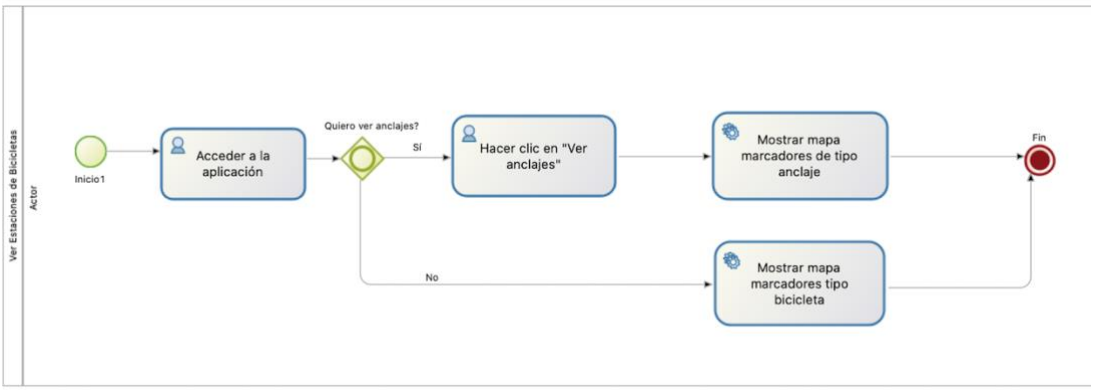

Figura 4.3. Diagrama de actividad - Ver estaciones de bicicletas geolocalizadas

Para que un usuario pueda ver el mapa con las estaciones de bicicletas, el usuario solamente tendrá que acceder a la aplicación web. La página principal será el mapa con los diferentes marcadores de las estaciones que hay en Madrid.

Estos marcadores serán de diferente color indicando el nivel de ocupación de la estación (ya sea para mostrar dicho nivel por bicicletas como para mostrarlos por anclajes). Si queremos que esa clasificación se muestre por anclajes, simplemente tenemos que dar al botón “Ver Anclajes”.

De esta forma el usuario podrá ver un mapa con las diferentes estaciones de bicicletas.

Diagrama de actividad - Obtener información de una estación en particular

El diagrama de actividad que corresponde con la figura 4.4, hace referencia al caso de uso "Obtener información de una estación en particular", el cual es el siguiente:

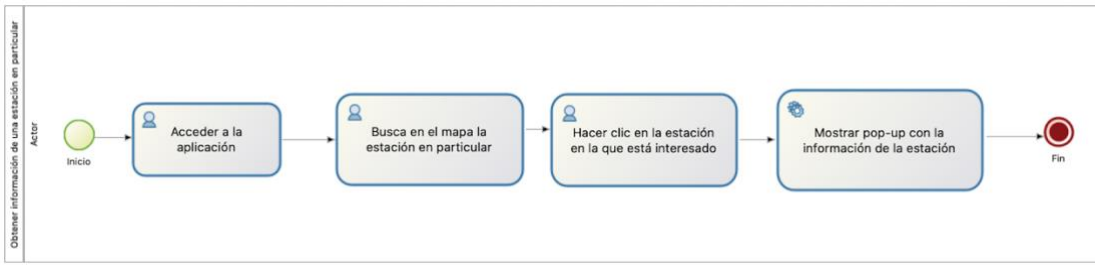

Figura 4.4. Diagrama de actividad - Obtener información de una estación en particular

Para que un usuario pueda ver información de una estación de bicicletas en particular, el usuario tendrá que acceder a la aplicación web y hacer clic en la estación de bicicletas en la que está interesado.

Al hacer clic le aparecerá el nombre de la estación, su dirección, el número de anclajes libres que tiene la misma y también el número de bicicletas disponible para el alquiler en tiempo real.

Diagrama de actividad - Visualizar previsión de los datos meteorológicos.

El diagrama de actividad que corresponde con el caso de uso de "Visualizar previsión de los datos meteorológicos" es el siguiente:

Figura 4.5. Diagrama de actividad - Visualizar previsión de los datos meteorológicos

Para que un usuario pueda ver información de los datos meteorológicos, deberá hacer clic en la información del tiempo que nos aparece en la barra de marcadores (en la esquina derecha), aparecerá un pop-up mostrándole la misma información que dispone en tiempo real (estado del cielo, temperatura en grados Celsius y velocidad y orientación del viento.

Diagrama de actividad - Acceder a la página oficial de una actividad deportiva

A continuación mostraremos el diagrama de actividad que corresponde con el caso de uso "Acceder a la página oficial de una actividad deportiva".

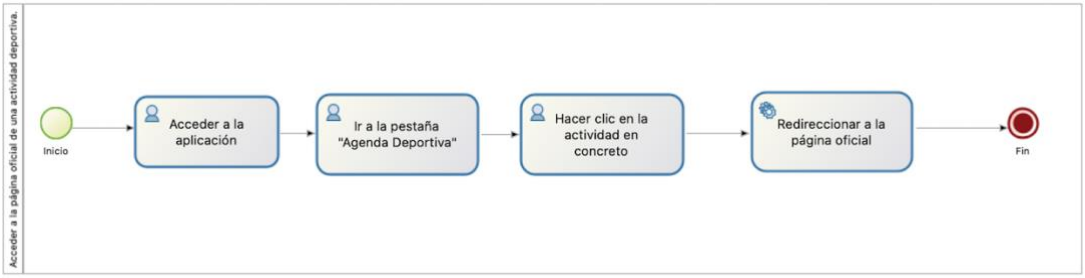

Figura 4.6. Diagrama de actividad - Acceder a la página oficial de una actividad deportiva.

Para que un usuario pueda ver más información de una actividad deportiva en particular, deberá acceder a la aplicación, dirigirse a la barra de navegación e ir a la pestaña "Agenda Deportiva" y hacer clic en la actividad en la que está interesado. Al pulsar sobre la misma, será redirigido al portal oficial de la aplicación en el cual podrá ver una descripción más detallada y también apuntarse a la misma (en caso de que sea necesario).

Este flujo será muy parecido al que se sigue al ir a la página oficial de una actividad de tipo Ocio.

Búsqueda y tratamiento de datos

En esta sección, hablaremos de la búsqueda de datos en portales Open Data y de su posterior tratamiento para poder utilizarlos en Muévete.

Búsqueda de datos abiertos

La búsqueda de datos abiertos comenzó centrándonos en España y, en particular, en las ciudades de mayor tamaño que dispusieran de un portal Open Data con alguno de los datasets necesarios para Muévete. De este modo se examinaron los portales de Madrid, Barcelona, Zaragoza, Valencia y Sevilla.

Como queremos que nuestra aplicación trabaje con diferentes datasets, haremos una división en este apartado entre datos que se refieren a estaciones de bicicletas, datos que se refieren al tiempo y datos que se refieren a actividades.

Datos relacionados con las estaciones de bicicletas.

En el portal open data de Barcelona ciudad se puede consultar información de las estaciones [11]. Esta información la disponen en csv y en formato json. No tienen un API por el que poder consultarlas directamente. La información que facilita este csv lo podemos encontrar en la tabla 4.10.

Nombre campo	Tipo campo	Descripción campo
id	Number	Identificador
station_id	String	Código de la estación
name	String	Nombre de la estación
physical_configuration	String	Electrónica o mecánica
lat	Number	Latitud
lon	Number	Longitud
altitude	Number	Altitud
address	String	Dirección de la estación

post_code	String	Código postal de la estación
capacity	Number	Capacidad de la estación
last_updated	String	Información de la última actualización

Tabla 4.10. Datos del dataset de Barcelona

El portal Open Data del Ayuntamiento de Madrid tiene el conjunto de datos para las bicicletas, además dispone de un API sobre el cual poder consultar la información acerca de las estaciones [12]. Para ello es necesario estar autenticado en su portal. La tabla 4.11 recoge dicha información.

Nombre campo	Tipo campo	Descripción campo
code	String	Resultado de la operación
description	String	Descripción
datetime	String	Instante de la operación en el lado del servidor
data	Array	Estructura principal de valores, si la operación ha ido bien contiene los siguientes campos de la tabla
id	String	Identificador de la estación
name	String	Nombre de la estación
geometry	String	Posición Geográfica
stop	String	Identificador de la parada
light	Object	Nivel de ocupación (0. bajo, 1.- medio, 2.- alto)
number	String	Identificador lógico de la estación
activate	String	Situación (0.- Inactivo 1.- Activo)
no_available	String	0.- No disponible 1.- Disponible

total_bases	String	Total bases
dock_bikes	String	Bicicletas en los anclajes
free_bases	String	Anclajes libres
reservations_count	String	Bicicletas reservadas
Number	String	Número activo de bicicletas reservadas

Tabla 4.11. Datos del dataset de Madrid

En Zaragoza, el ayuntamiento tiene un portal Open Data muy completo [13], en él vemos que los atributos del datasets que hace referencia a las bicicletas contiene la información que recoge la tabla 4.12.

Nombre campo	Tipo campo	Descripción campo
id	String	Id estación
uri	String	URI estación
title	String	Nombre de la estación
language	String	Lenguaje en el que podemos encontrar la información de la estación.
category	String	Categoría de la estación
estado_t	String	Estado de la estación, si está operativa su valor es "OPN"
bicisdisponibles_i	Number	Número de bicis disponibles en la estación
anclajesdisponibles_i	Number	Número de anclajes disponibles
icon_t	String	Icono según el estado de la estación
x_coordinate	Number	Coordenadas UTM30
y_coordinate	Number	
coordenadas_p	Number	Coordenadas WGs84
tipocontenido_s	String	Contenido de los datos

start	Number	Indicar el registro a partir del cual se muestran los resultados
rows	Number	Indicar el nº de registros a mostrar
sort	Number	Ordenar según un criterio
last_modified	Number	Fecha de la última modificación

Tabla 4.12. Datos del dataset de Zaragoza

En Valencia, el ayuntamiento ofrece en su portal Open Data [14] un conjunto de datos en diferentes formatos. Estos datos los mostramos en la tabla 4.13.

Nombre campo	Tipo campo	Descripción campo
Aproximaci	String	Valenbisi estación
Number	Number	Número de la estación
Address	String	Dirección.
Open	String	SI es operativa o no
Available	String	Bicicletas disponibles
Free	Number	Anclajes disponibles
Total	Number	Número total de anclajes
Ticket	String	Si puedes pagar con tarjeta o no
Updated_at	String	Información de la última actualización

Tabla 4.13. Datos del dataset de Valencia

Finalizamos esta fase con Sevilla. Esta ciudad dispone en su página oficial open data [15] un servicio por el cual puedes consultar una estación de bicicletas, pero no nos informa de los huecos libres que puede haber en ella ni tampoco de cuántas bicicletas hay. Los datos que ofrece son los que se exponen en la tabla 4.14. No tiene un API por el que poder consultar estos datos.

Nombre campo	Tipo campo	Descripción campo
FID	String	Id.
id_estacio	String	Identificador de la estación.
direccion	String	Dirección de la estación
aproximaci	String	Sitios que están cerca a la estación
Lat_Y	Number	Latitud
Lon_x	Number	Longitud
CreationDate	String	Fecha de creación
Creator	String	Información del creador de la estación
EditDate	String	Información de la última vez que se editó

Tabla 4.14. Datos del dataset de Sevilla

Además de los diferentes conjuntos de datos para las ciudades, también hemos tenido en cuenta si pudiéramos consultarlos en un API directamente para que la comunicación fuese más sencilla y rápida. Os exponemos los pros y contras de cada ciudad y acabaremos indicando el por qué de quedarnos con la ciudad elegida en la tabla 4.15.

Ciudad	API	Geolocalizada	Info. Estación	Comentario
Barcelona	NO	SI	SI	Se haría una petición a una URL la cual nos descarga un fichero, no es un API.
Madrid	SI	SI	SI	API formal, hay que pedir credenciales.
Zaragoza	SI	SI	SI	API formal, hay que pedir credenciales.
Valencia	NO	SI	SI	Se haría una petición a una URL la cual nos

				descarga un fichero, no es un API.
Sevilla	NO	SI	NO	No ofrece información relevante de las estaciones.

Tabla 4.15. Conclusión Datasets

Tras este estudio pudimos hacer una gran criba de qué ciudades nos facilitaban los datos de una manera más formal (especificando una cabecera y obteniendo una respuesta estructurada por su parte) y cuáles de ellas no. Sevilla quedó rechaza en primer lugar al no disponer de la información que necesitábamos acerca de los anclajes o de las bicicletas disponibles en cada estación. Barcelona y Valencia, pese a que nos proporcionaban datos geolocalizados y también información relevante, decidimos no cogerlas como opción al no disponer de un API formal. Por lo tanto, los principales datasets que tenemos son los correspondientes a Zaragoza y a Madrid.

Datos relacionados con valores meteorológicos

Haciendo una búsqueda, el principal medio por el cual se puede consultar el tiempo, es a través de la página oficial de la Administración Estatal de Meteorología, sus siglas AEMET [16].

Cuenta con un API [17] en el cual tras hacer un login y obtener un API_KEY puedes consultar toda la información relativa a datos meteorológicos por comunidad autónoma de España. En nuestro caso, como queremos valores climatológicos, el conjunto de datos que obtiene la información que necesitamos es “Predicción por municipios diaria. Último elaborado”.

Con esta llamada obtendremos un array, por cada celda de este array obtendremos la siguiente información:

- uvMax
- fecha
- probabilidad de precipitación

- probabilidad de cota de Nieve
- estado del cielo
- viento: dirección - velocidad
- racha Máx:
- temperatura
- sensación térmica
- humedad relativa

Como hemos comentado antes, podíamos obtener esta información tanto para Zaragoza como para Madrid, por lo que encaja con lo que necesitamos.

También teníamos la opción de añadir algún widget a nuestra aplicación web.

HotelMix [18] es una página web que de forma gratuita facilita diferentes tipos de widgets que se pueden añadir al desarrollo web de forma sencilla. Incluso permite personalizar el widget cambiando el color o la medida de temperatura (Celsius por Fahrenheit). En las figuras 4.7 y 4.8 podemos ver cómo esta página nos ofrece diferentes tipos de widgets de gran tamaño o menor tamaño con información climatológica.

Figura 4.7. Widgets tiempo grandes

Figura 4.8. Widgets tiempo sencillos

No obstante, al no disponer de información del viento decidimos no contar con esta herramienta para nuestro desarrollo.

Tiempo.es [19] también facilita datos del tiempo en forma de widget para que simplemente tengas que incrustarlo en el desarrollo. Lo hace de una forma muy parecida a la herramienta que hemos visto anteriormente. Permite una amplia gama de personalización, incluso existe la opción de que añada información del tiempo. A continuación, podemos ver el portal en la figura 4.9.

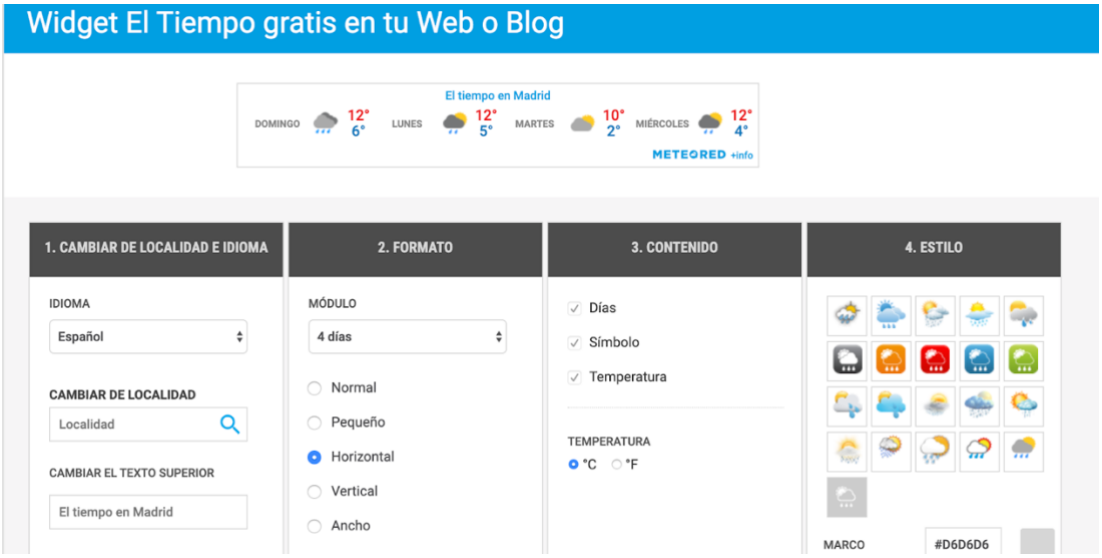

Figura 4.9. Widget del tiempo

No obstante, el resultado final con los widgets no nos parecía del todo apropiado y vimos más interesante poder llamar nosotros a un API y traernos los datos en tiempo real por cada hora, por lo que nos quedamos con la primera opción que hemos expuesto.

Datos relacionados con actividades

Puesto que las ciudades principales que cuentan con información sobre bicicletas son Madrid y Zaragoza, nos centramos en estas para la búsqueda de un datasets que nos de información de actividades de la ciudad.

En Madrid, el Ayuntamiento ofrece un portal de datos [20] y en particular un API [21] con información detallada acerca de las diferentes actividades que se pueden disfrutar en la ciudad. Este API dispone de dos endpoints, uno para las actividades de ocio y aventuras y otro para las actividades deportivas.

Estos datasets incluyen la siguiente información para las actividades de ocio y aventuras y comparten la misma estructura para aquellas actividades que son de tipo deportivas. En la tabla 4.16 recogemos la información que nos proporciona dicho dataset.

Nombre campo	Tipo campo	Descripción campo
uid	String	Identificador
dtend	String	Fecha de obtención
location. longitude	Number	Coordenada correspondiente a la longitud
location. latitude	Number	Coordenada correspondiente a la latitud
event-location	String	Dirección de dónde se encuentra el evento
link	String	Link directo al evento
organization	String	Organización del evento
title	String	Título del evento
dtstart	String	Cuando comienza
references	String	Referencias
recurrence	String	Cada cuanto tiempo se realiza el evento
price	Number	Precio del evento
address	String	Dirección de dónde se organiza el evento
description	String	Descripción del evento
excluded-days	String	Días que no se hace el evento

Tabla 4.16. Dataset actividades Madrid

Con esta información podremos mostrar las diferentes actividades en un mapa o hacer una lista en nuestra aplicación.

En Zaragoza, su API también ofrece un dataset para la obtención de actividades que se realizan en la ciudad. Este dataset [22] contiene, entre otras, la información que podemos ver en la tabla 4.17.

Nombre campo	Tipo campo	Descripción campo
geometry	String	Objeto que contiene información acerca de la geolocalización del evento.
id	String	Id del evento
location	String	Localización del evento
startDate	String	Fecha inicio
endDate	String	Fecha fin
openingHours	String	Horas de apertura
title	String	Título
description	String	Descripción
destacada	Boolean	Si es un evento destacado o no
RR SS	String	Twitter, Facebook...
imagen	String	Imagen del evento
url	String	Url del evento

Tabla 4.17. Datasets actividades Zaragoza

Una vez que hemos hecho el estudio de los diferentes atributos que nos proporcionan ambas ciudades, tendremos que sopesar con cuál de ellas nos quedamos para el desarrollo de nuestra aplicación.

En la siguiente tabla 4.18 podemos ver qué características importantes tienen cada una de ellas y así poder decantarnos por la más apropiada.

Ciudad	API	Datos geolocalizados	Volumetría estaciones	Volumetría bicicletas
Madrid	SI	SI	165 [23]	2028 [23]
Zaragoza	SI	SI	130 [24]	1300 [24]

Tabla 4.18. Conclusión Datasets Actividades

Como podemos comprobar ambas ciudades disponen casi del mismo conjunto de datos, la información que ofrecen es muy parecida y esto hace que la elección sea difícil. No obstante, pensando en el reto que puede suponer tener una gran volumetría de datos (estaciones y bicicletas), vemos un proyecto más interesante hacerlo para **la ciudad de Madrid**.

Arquitectura

Podemos definir arquitectura software como el conjunto de patrones que proporcionan el guion a seguir para la construcción del software, permitiendo a todo el conjunto de personas que forman parte del proceso (programadores, analistas, diseñadores…) compartir una misma forma de trabajo y cubrir objetivos y restricciones de la aplicación.

Esta parte del proceso software es considerada de las más importantes, ya que en ella se definirá el marco de referencia necesario para el desarrollo de la aplicación.

En nuestra aplicación no queremos que se almacene ninguna información, toda aquella que se pide será informada al cliente en tiempo real, por lo que el componente de “base de datos” nosotros no lo necesitaremos.

Este proyecto se ha desarrollado siguiendo varios patrones de arquitectura. Para el desarrollo de la aplicación se ha seguido el patrón Cliente-Servidor. El Cliente se ha desarrollado siguiendo el patrón de diseño de Single Page Application (SPA). La

SPA internamente se ha estructurado siguiendo el patrón de arquitectura de capas Model-View-View-Model (MVVM). En la parte del Servidor, las llamadas a las APIs externas han sido realizadas usando el protocolo REST.

En los siguientes apartados explicaremos más detenidamente cada uno de los patrones utilizados. La arquitectura de la aplicación queda definida en el diagrama que podemos ver en la figura 4.10:

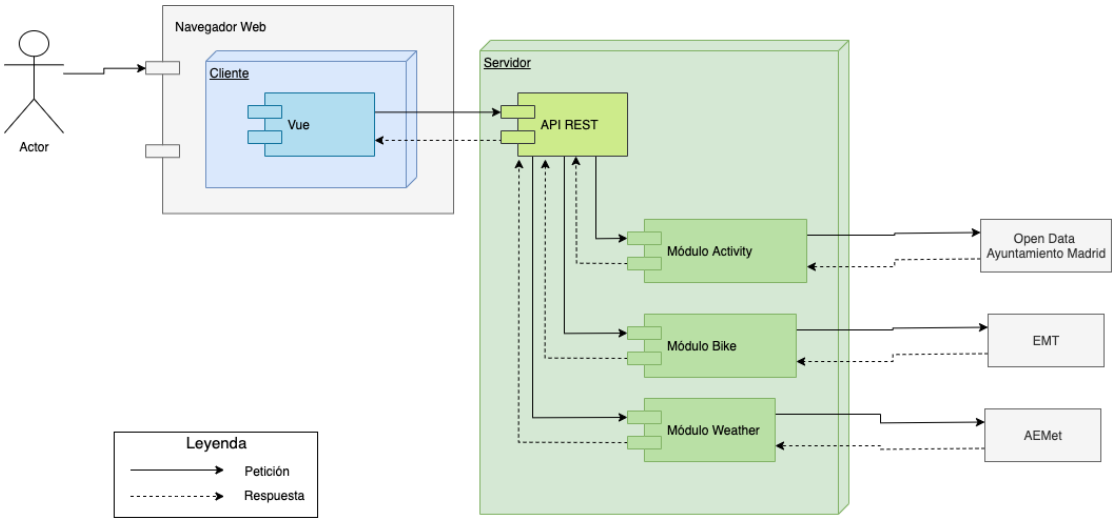

Figura 4.10. Arquitectura global Muévete

Patrón de arquitectura Cliente-Servidor

En nuestro caso, puesto que la aplicación será web, nuestra arquitectura estará orientada a ese tipo de desarrollo. El modelo de arquitectura que utilizaremos será **arquitectura cliente-servidor**.

Parafraseando los apuntes de la asignatura de "Programación en Internet", podemos decir que la arquitectura cliente-servidor es un modelo de aplicación distribuida la cual divide el trabajo entre el proveedor de los recursos o servicios (servidor) y aquellos consumidores de estos recursos (clientes). La comunicación se hace a través de la red siguiendo unos protocolos. En la figura 4.11 se muestra cómo tenemos implementado este patrón junto con el SPA y Facade en la aplicación de Muévete.

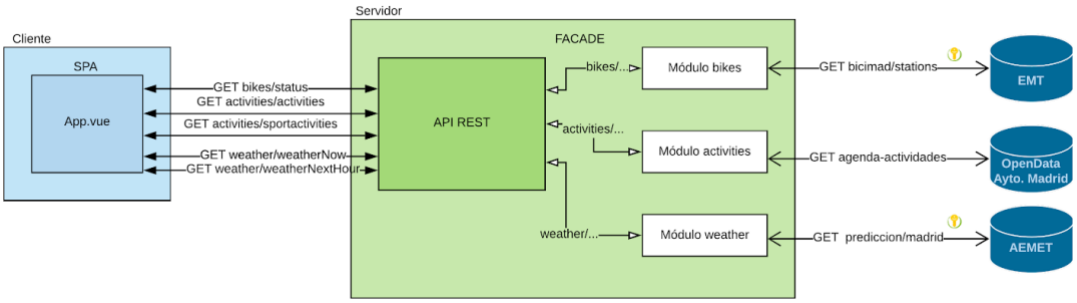

Figura 4.11. Arquitectura cliente-servidor de Muévete

Implementación del patrón en nuestra aplicación:

- **Cliente**: Para la implementación del lado cliente, se ha escogido implementar una aplicación web desarrollada en JavaScript usando el framework Vue, usando el patrón de arquitectura SPA. Esta aplicación contiene toda la lógica de presentación implementada en el proyecto. El cliente llamará al servidor para obtener los datos que quiere mostrar.

  En el caso de Muévete, esta parte se mantiene lo más desacoplada posible a la ciudad, esto quiere decir que simplemente pinta los datos que el servidor le provee. De esta forma, si al día de mañana queremos mostrar otra ciudad, esta parte no sufrirá cambios. Para conseguirlo, el servidor le tendrá que proporcionar los datos siempre con el mismo formato. Esto está definido con más detalle, en el siguiente punto de la documentación.

- **Servidor**: Del lado del servidor tenemos un servicio que nos proveerá de los datos que van a ser pintados en el cliente. Este servidor está implementado en Javascript también, usando NodeJS y el framework Express. Este servicio implementa esencialmente el patrón Facade (Fachada), exponiendo una interfaz sencilla y permitiendo esconder la complejidad de conocer y llamar a distintos servicios públicos para obtener los datos necesarios.

  De esta forma es como conseguimos que el front esté lo más desacoplado posible, siendo la parte del servidor la encargada de hacer las

peticiones más concretas a los diferentes servicios. En el apartado cinco podemos encontrar las rutas que implementa esta parte y cómo son los datos devueltos. En la implementación actual, los servicios públicos a los que se acceden pertenecen a AEMET, EMT y al Ayuntamiento de Madrid.

- **Protocolos de conexión**: El cliente y el servidor se comunican entre sí usando conexiones HTTP, en particular usando una interfaz REST. Los 3 servicios públicos que contienen los datos usados en la aplicación también utilizan el patrón REST para organizar sus interfaces. Todas estas interfaces exponen sus datos en el formato de texto JSON. Además, las peticiones a la API REST de la Empresa Municipal de Transportes de Madrid (EMT) y la del tiempo (AEMet) siguen el protocolo HTTPS, es decir, utilizan el modo seguro de transferencia de hipertexto. Esto implica que para hacer las peticiones, necesitaremos un API KEY que obtenemos al registrarnos en sus servicios y el cual mandamos en la cabecera de las peticiones.

- **Base de datos:** Los 3 servicios públicos comentados tienen almacenados los datos necesarios para el funcionamiento de Muévete, a los que accedemos a través de las distintas peticiones HTTP/HTTPS que realizamos desde el servidor, por lo que no necesitamos tener una base de datos que cumpla esas funciones.

Patrón de diseño SPA

Para definir la estructura principal de el cliente web de Muévete, se ha decidido usar el patrón Single Page Application (SPA). VueJS nos va a ayudar a agrupar todo el código de nuestro cliente para que funcione cumpliendo con las características de este patrón.

Una de las características más importantes a mencionar de Vue.js es que el núcleo de esta librería está enfocado en la capa de la Vista (View) en el patrón MVVM, sin embargo, también se pueden construir aplicaciones robustas de tipo Single-Page Application.

Una **Single Page Application** o aplicación de página única es una aplicación web que carga todo el contenido de ésta en la misma página. Es decir, la página en si no se refresca cada vez que pulsamos en un enlace dentro de la misma.

A diferencia de la web tradicional, en una Single Page Application, no tenemos varias páginas, es decir, no tendremos una página para la sección "contacto" de nuestra web, otra para la sección "sobre nosotros", etc. Si no que en la misma página se cargará todo el contenido que el usuario desee ver. De ahí el nombre de "Single Page Application" (aplicación de página única). Lo que sí tenemos son distintas vistas que podemos personalizar a voluntad. Todas estas vistas se irán cargando según el usuario haga clic en uno u otro enlace.

VueJS organiza estas vistas con lo que llama "Componentes". Estos componentes son independientes entre ellos y agrupan la lógica de funcionamiento y de presentación de fragmentos de la interfaz. Estos componentes se pueden colocar por separado o anidarse, pudiendo estructurar páginas realmente complejas.

En el caso de Muévete tenemos una aplicación principal y cinco componentes diferenciados por la funcionalidad de cada uno. Tendremos uno para los datos de la barra de Navegación (común en todas las vistas), otro en el que se muestra lo principal de cada página y que variará en función de dónde nos encontremos. Esto cargará el mapa de bicicletas o el listado de actividades deportivas o el mapa con el listado de las actividades de ocio. Por último, al igual que la barra de navegación, tendremos el footer el cual no se recargará por cada vista. Por lo tanto, nuestra SPA estará compuesta de la siguiente forma:

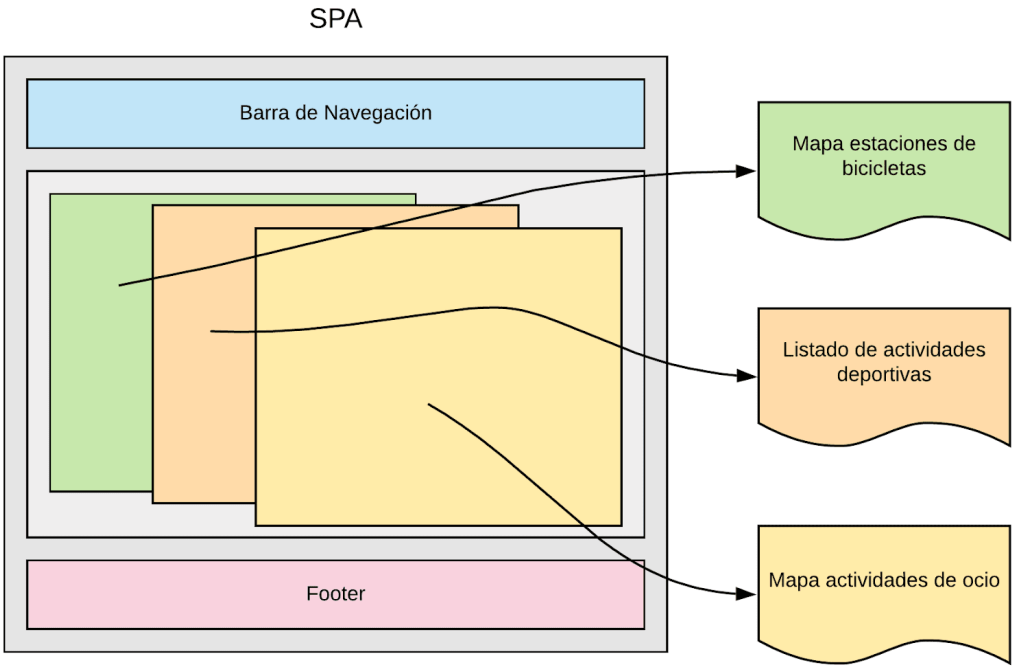

Figura 4.12. Arquitectura SPA Muévete

Patrón de arquitectura MVVM

Técnicamente, Vue.js está definido como un entorno de trabajo hecho en JavaScript para desarrollar aplicaciones web usando el patrón de arquitectura de software MVVM (Model–View–ViewModel).

En Muévete, el **patrón MVVM** ayuda a separar la lógica de negocios de la interfaz de usuarios facilitando las pruebas, mantenimiento y la escalabilidad de los proyectos. Mantener una separación entre la lógica de aplicación y la interfaz de usuario ayuda a abordar numerosos problemas de desarrollo y puede hacer que probar una aplicación, mantenerla y desarrollarla sea más fácil. [27] También puede mejorar enormemente las oportunidades de reutilización de código y permite a los desarrolladores y diseñadores de interfaz de usuario colaborar con mayor facilidad al desarrollar sus respectivos elementos de una aplicación.

En este patrón hay tres componentes principales y cada uno de ellos con un propósito diferente: el modelo, la vista y el modelo de vista. En la siguiente figura 4.13 mostramos este patrón de una forma gráfica.

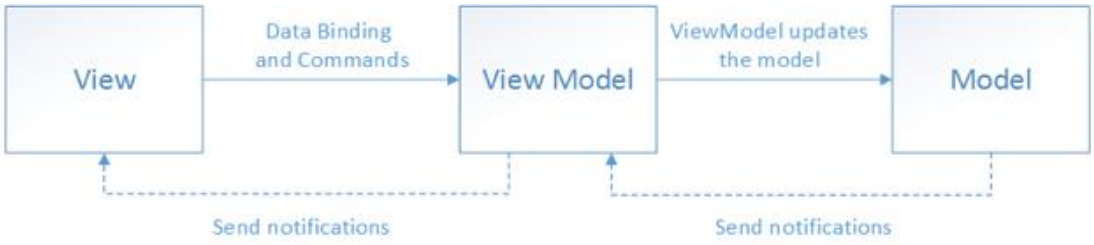

Figura 5. Esquema patrón MVVM

El framework VueJS provee de varias primitivas que permiten adaptar este patrón en la aplicación de una forma muy sencilla. A través de sus ficheros ".vue", los distintos elementos de este patrón pueden ser fácilmente identificados y estructurados.

Aplicado en Muévete, podemos encontrar:

- **Modelo**: En la aplicación, podemos comprobar en cada uno de los ficheros .vue como el modelo se actualiza con las peticiones que se realizan al Servidor con los datos que se tienen que renderizar para ser pintados. Estas peticiones están agrupadas en el fichero "request.js".

- **Vista**: Podemos encontrar la lógica de las vistas en la sección de HTML ubicada en los distintos ficheros .vue. Estas vistas permiten representar visualmente tanto la propia interfaz de la aplicación como los datos que se representan en esta, siendo estos últimos dinámicos, es decir, actualizados según se van solicitando al Servidor.

- **Modelo de vista**: El modelo de vista es un actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. La comunicación entre la vista y el viewmodel se

realiza por medio los enlaces de dato. En Muévete esta lógica se implementa en la sección “script” dentro de cada uno de los ficheros “.vue” .

Patrón Facade (Fachada)

Muévete necesita realizar llamadas a distintos servicios públicos para obtener los datos que necesita. Disponer de una capa de abstracción sobre la realización de estas llamadas nos permite desacoplar la capa de presentación (situada en el cliente) de las tareas de conocer todos los datos relativos a la realización de estas peticiones. El patrón que podemos encontrar a la hora de realizar esta abstracción puede ser el patrón conocido como “Facade”. Nos hemos basado en este patrón para estructurar el código y las responsabilidades encontradas en el Servidor.

En el Servidor podemos ver que la interfaz a la que hace referencia el patrón “Facade” se encuentra implementada como una API REST. Esta expone de una forma sencilla y sin dar detalles de ni de su implementación ni de las peticiones necesarias los datos que podemos encontrar en los tres servicios públicos usados. Esto nos va a facilitar no solo poder estructurar mejor nuestra aplicación, dividiéndola en componentes desacoplados, sino también poder hacer cambios en las URLs de los servicios o otros cambios a la hora de realizar las peticiones sin que haya que modificar nada en el Cliente web de Muévete.

API Rest

Muévete necesitaba un protocolo apropiado para conectar la parte Cliente con la parte Servidor. El protocolo que más se ajusta a los requisitos de la aplicación es el conocido como API REST. Este protocolo además se ajusta especialmente bien al ser una comunicación que se va a realizar entre un navegador que ejecutará el Cliente y el Servidor que va a servir estos datos. Este apartado lo podemos encontrar más en detalle en el apartado “Listado de rutas Backend” en el capítulo cinco, sección cuarta.

El término API (Application Programming Interface) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. [28] El término REST (Representational State Transfer) se originó en el

año 2000, descrito en la tesis de Roy Fielding. REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

Las restricciones definidas por Roy Fielding y aplicadas a Muévete, para que cumpla con un sistema RESTful, aplicarían de la siguiente manera:

- **Cliente-servidor**: esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor y el servidor se "despreocupa" de cómo son usados los datos que envía al cliente. La utilización del patrón Facade dentro del servidor nos ha permitido poder generar una interfaz REST sencilla y apropiada.

- **Sin estado:** aquí decimos que cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones.

- **Cacheable**: debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso.

- **Interfaz uniforme**: define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio REST debe tener una única dirección, "URI".

- **Sistema de capas:** el servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

Capítulo 5.- Implementación y desarrollo

Una vez tenemos clara la arquitectura y el alcance del proyecto, podemos seguir explicando la implementación de Muévete.

La fase de implementación se lleva a cabo una vez tenemos claros los requisitos y restricciones que debe cumplir la aplicación y, por otro lado, la arquitectura que se llevará a cabo en el mismo. En esta etapa se llevará a cabo la construcción de los elementos software para llegar a tener la solución propuesta y analizada previamente.

No obstante, antes de empezar con la propia implementación, debemos saber qué herramientas y tecnologías son las más adecuadas para abordar el proyecto.

A continuación, se detallarán las herramientas que se han utilizado para llevar a cabo el proyecto, también las tecnologías utilizadas y el por qué de elegirlas y por último hablaremos de la propia implementación del software para llegar a construir Muévete.

Herramientas

En esta sección se exponen ciertas herramientas que han ayudado de forma secundaría durante el desarrollo del proyecto. Decimos que de forma secundaría ya que no son imprescindibles para la elaboración de este, pero son de mencionar por facilitar las tareas para las que se han usado.

Lucidchart

Esta herramienta se ha utilizado para la creación de los diagramas UML especificados en la fase de análisis y diseño. Es una herramienta cómoda de utilizar ya que es 100% online y cuenta con una versión gratuita que te provee de varias plantillas para hacer diferentes tipos de diagramas. En la figura 5.1 podemos encontrar el portal de la aplicación, el cual llama la atención por la gran cantidad de diagramas que permite realizar.

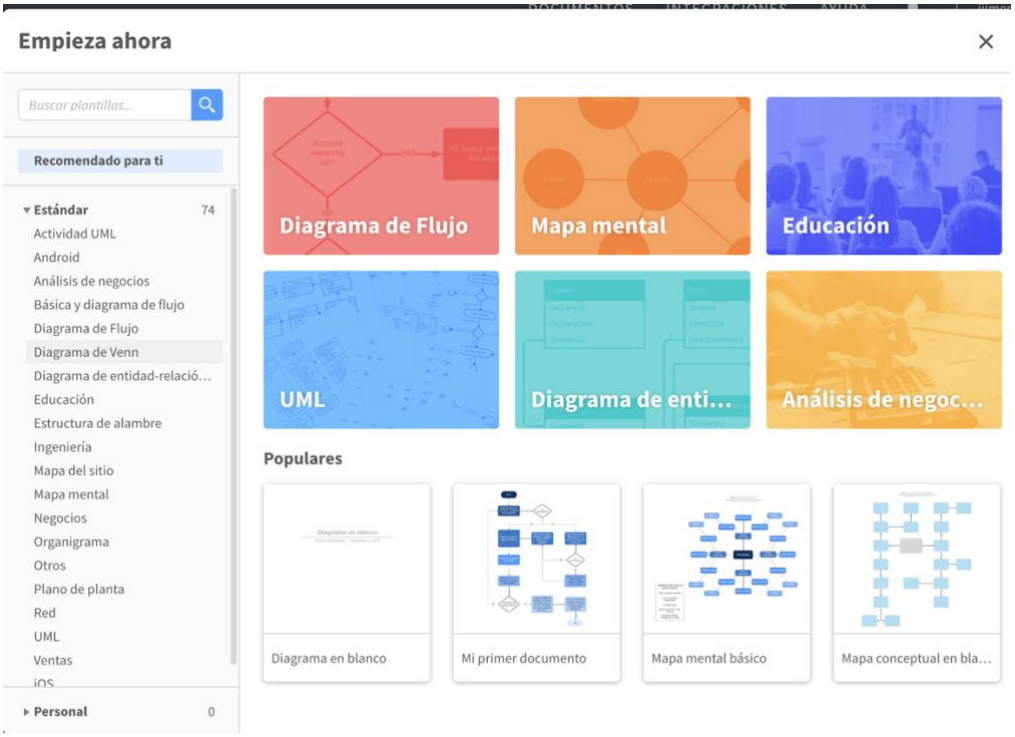

Figura 5.6. Lucidchart: plantillas de diagramas

Además, a estos diagramas se les puede cambiar los colores, formas y tamaños de forma sencilla y permite la exportación de estos en varios formatos (png, pdf, svg…) Se pueden compartir los diagramas con otras personas e incluso permite importar los diagramas para utilizarlos en otras herramientas (Visio, Draw.io, Gliffy…).

BonitaSoft

Bonita es una plataforma código abierto de aplicaciones de flujo de trabajo y gestión de procesos de negocio (BPM) creada en 2001.

En este caso, se ha utilizado Bonita Studio para la creación de los diagramas de actividad o diagramas de flujo. Esta herramienta es muy potente y pese a que su objetivo es diseñar los procesos BPM usando BPMN (Business Process Management Notation), ha sido la elegida para dicho objetivo por su facilidad y presentación a la hora de la creación de dichos diagramas. En la siguiente imagen 5.2 podemos ver el portal.

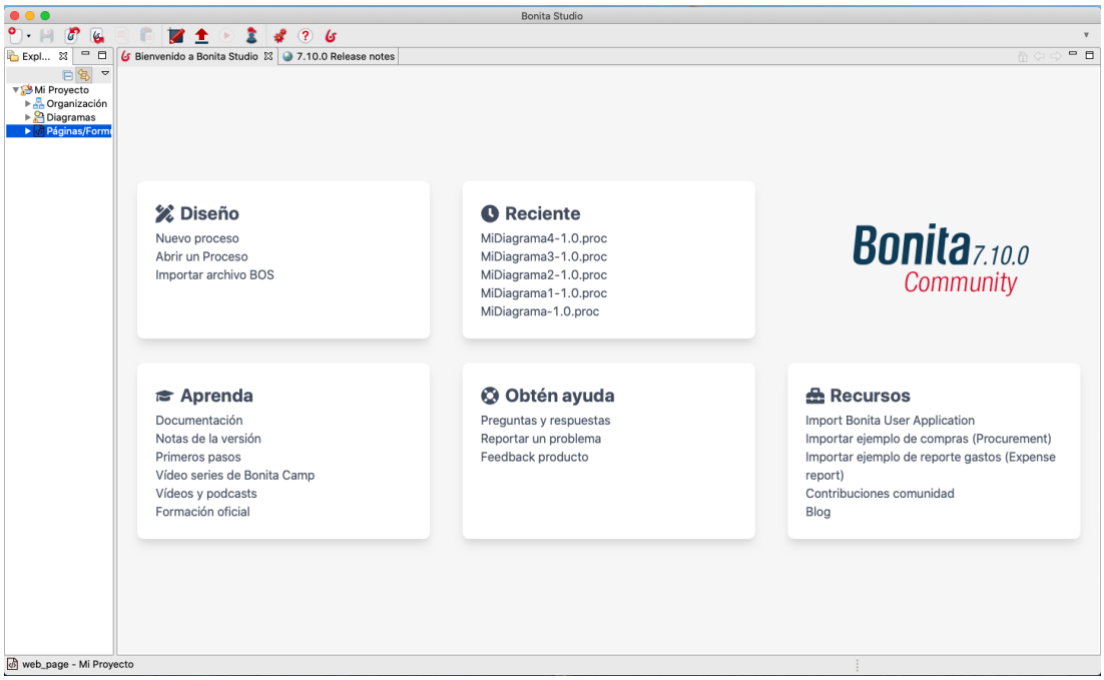

Figura 5.7. Portal BonitaSoft

JSONFormatter

JSONFormatter es una herramienta online que sirve para formatear y validar un JSON de forma online. En cuanto al formateo, permite elegir la tabulación y los espacios que queremos dejar entre atributos. Nos presenta una interfaz sencilla, como podemos ver en la figura 5.3.

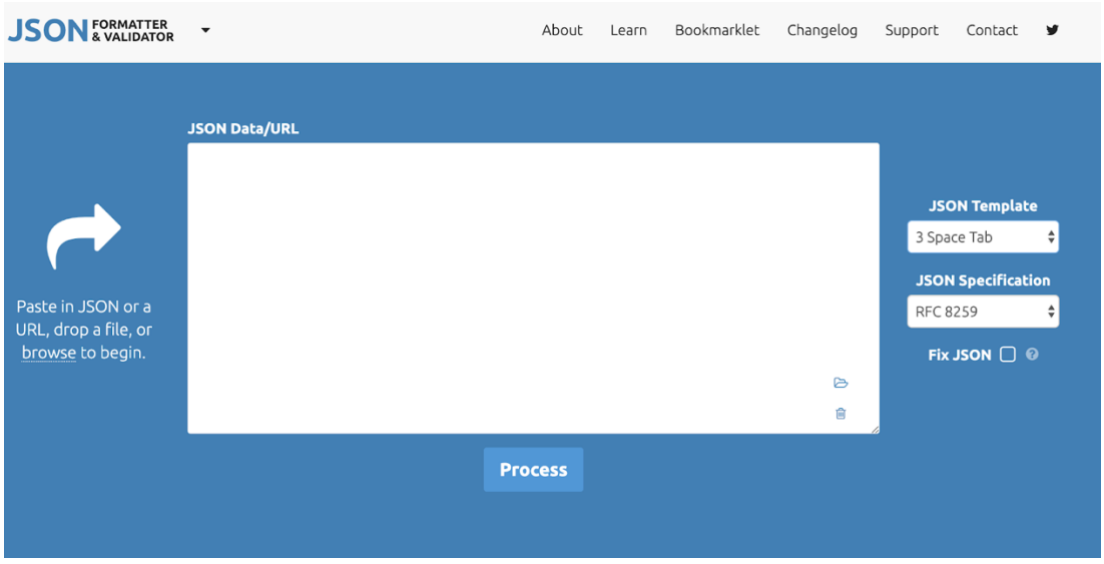

Figura 5.8. Portal JSONFormatter & Validator

Puesto que este proyecto integra varias llamadas a varias APIs diferentes y cuyos datos proporcionados en formato JSON son bastantes grandes (listas grandes

con varios atributos por cada elemento) y generalmente las llamadas a las APIs nos devuelven respuestas sin saltos de línea para ahorrar espacio, puede ser extremadamente difícil leerlo y darle sentido, por lo que se requería una aplicación que validara, formateara y permitiera la navegación de dichos datos. De esta forma, era más sencillo ver qué atributo dentro del JSON era el que necesitábamos y también poder ver la clave que era necesaria para obtener el dato en concreto en nuestro desarrollo.

Canva

Canva es un sitio web de herramientas de diseño gráfico simplificado. Esta herramienta utiliza un formato de arrastrar y soltar y proporciona acceso a más de un millón de fotografías, vectores, gráficos y fuentes para la creación de diseños de forma gratuita, a continuación, podemos ver una imagen del portal.

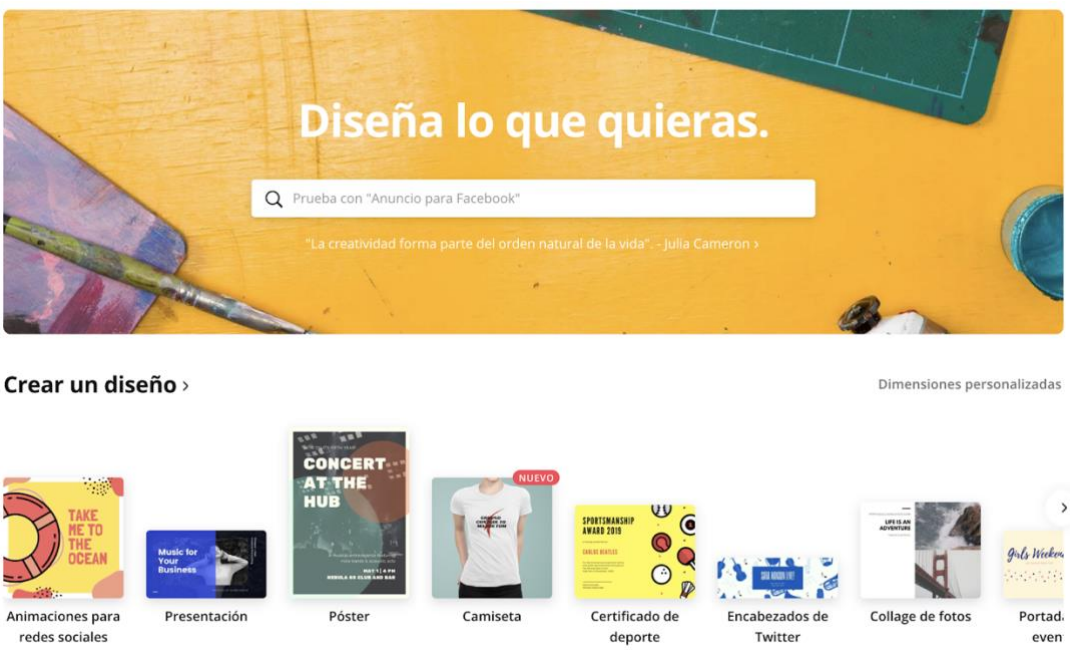

Figura 5.4. Canva: inicio

Canva se ha utilizado para el diseño del logo de la aplicación. Este es el que hace referencia la figura 5.5:

Figura 9. Logo Muévete

Iconos8

Otra de las herramientas que ha sido indispensable a la hora de desarrollar la aplicación ha sido Iconos8, podemos ver el portal en la figura 5.6. Esta página provee de innumerables iconos y para cada icono diferentes temas para que encontremos el que más se adecua a nuestra aplicación. Lo positivo de la misma es que son de forma gratuita. Todos los iconos de la aplicación han sido sacados de aquí. Los marcadores tanto de las bases como de las bicicletas han sido una mezcla de dos de los iconos que nos podemos encontrar en la página, al superponerlos, creamos los que podemos ver en Muévete.

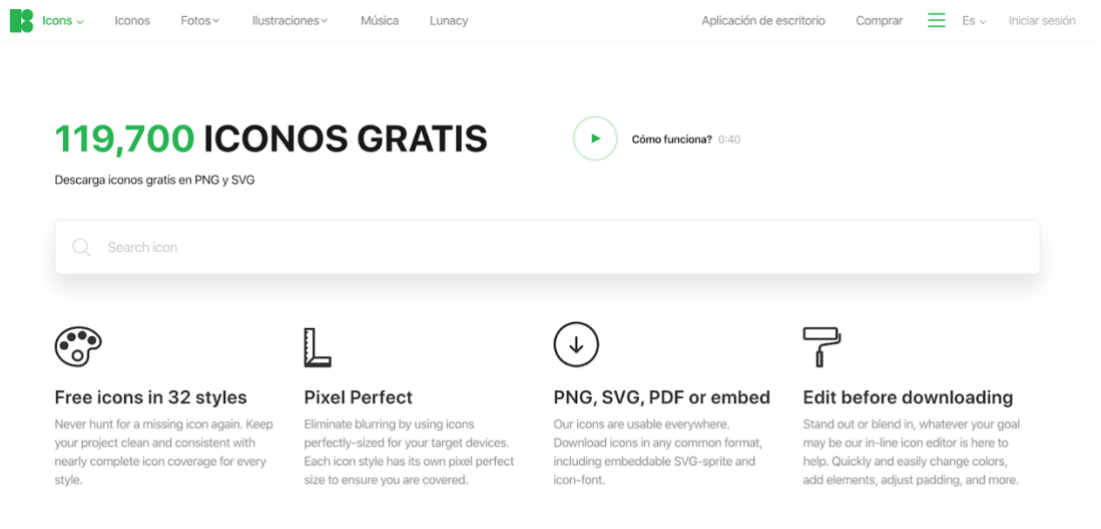

Figura 10. Página web Iconos8

En el caso de que no encontremos el icono que queremos, la web nos permite crear un comentario que posteriormente verán los diseñadores para añadirlo a la biblioteca de iconos que tienen.

Los marcadores creados para Muévete son los siguientes:

- Ver los anclajes de bicicletas.

Figura 5.7. Marcadores bicicletas por anclaje

- Ver las bicicletas

Figura 5.8. Marcadores bicicletas

Control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos (código software) a lo largo del tiempo de tal manera que sea posible recuperar versiones del software específicas más adelante.

Para realizar dichas acciones en este proyecto, se han utilizado Bitbucket y Git.

Bitbucket

Bitbucket Cloud es una herramienta de colaboración y alojamiento de código basada en Git, creada para equipos.

Figura 5.9. Logo Bitbucket

Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con rapidez y eficiencia.

Figura 5.10. Logo Git

Framework

En este apartado veremos las tecnologías utilizadas para el proceso de desarrollo del proyecto y el motivo de su elección.

Separaremos estas tecnologías en función de la parte a la que van dedicadas. Tendremos unas que son más específicas del cliente (Front) y otras que van más orientadas a la parte del servidor (Back).

Cliente

Cuando hablamos de la parte cliente de una aplicación (frontend) nos referimos a la parte de un programa a la que un usuario puede acceder directamente, es lo primero que se encuentra y es con lo que el usuario interactúa. Esta capa es la que permite la interacción usuario - máquina. Por lo tanto, estas tecnologías son aquellas que tienen que ver con el diseño y desarrollo web que corren en el navegador y que se encargan de la interactividad con los usuarios.

En la actualidad existen varios marcos de desarrollo orientados a la parte front de una aplicación web. Los principales hoy en día son: angular, react y nodejs. Estos tres marcos son los que se tomaron en cuenta a la hora del desarrollo del proyecto, a continuación, pasamos a describirlos.

Angular

Figura 5.11. Logo Angular

Es un framework mantenido por google, se creó en 2009 e implementa el patrón MVC (Modelo Vista Controlador) para el desarrollo de aplicaciones web que permite crear aplicaciones SPA (Single Page Application).

Angular permite la creación de componentes a través de lo que llaman “directivas”. Estas directivas son muy útiles pero difíciles de usar. Los componentes son pequeñas partes lógicas de la aplicación, que van a estar representando a un trozo de la pantalla. Un componente es un bloque, que contiene un template, contiene estilos, contiene lógica [29].

El uso de esta herramienta es muy alto entre los desarrolladores webs, lo que hace que tenga gran cantidad de documentación y apoyo.

Como punto negativo, es un marco muy pesado (alrededor de 500 KB), esto es debido a que Angular tiene mucho que ofrecer a sus desarrolladores, desde plantillas hasta pruebas (test). Obviamente, esto necesita espacio, lo que lo hace pesado. Además, el uso real del DOM (Modelo de Objetos del Documento) por parte de Angular afecta a su rendimiento y capacidad para crear aplicaciones de software dinámicas.

React

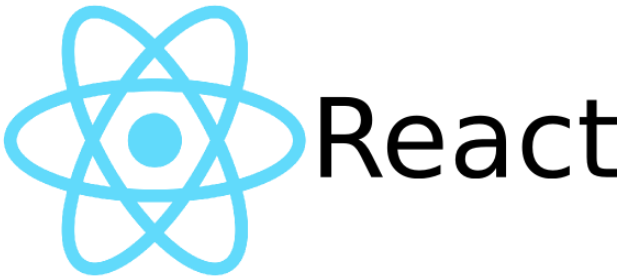

Figura 5.12. Logo React

React no es un framework, es una librería de JavaScript para construir interfaces de usuarios. Fue lanzado en 2013 por Facebook y es utilizado comúnmente en sitios webs de alto tráfico.

React también está basado en componentes, por lo que posibilita la reutilización de código. La lógica de los componentes está escrita en JavaScript y no en plantillas, por lo que puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM.

Es una librería y por lo tanto, también es más liviana que otros framework, llegando a pesar alrededor de 100 KB. Es adecuada para aplicaciones ligeras.

Esta librería tiene un gran apoyo en la comunidad debido a que está en constantes actualizaciones que hacen que cada vez esta librería sea más robusta, sin

embargo, este también es un punto débil. Debido a sus constantes actualizaciones no dispone de una gran documentación.

Vue.js

Figura 5.13. Logo Vue.js

Es un framework progresivo lanzado en 2014 y desarrollado por un exingeniero software de Google. Es de los framework de JavaScript más recientes. Decimos que es progresivo ya que podemos usarlo para algo muy básico o para algo más complejo como el desarrollo de una SPA, o para Server-Rendering. Siempre con un rendimiento y experiencia de desarrollo muy buena. Utiliza el patrón Modelo - Vista - Vista Modelo.

Vue ha tomado todos los buenos atributos de los framework lanzados antes. También está basado en componentes flexibles que permite su reutilización dentro y fuera del proyecto (componentes reutilizados en cualquier desarrollo).

Vue utiliza Virtual DOM como un concepto adoptado de React. Esto garantiza un rendimiento más rápido y sin errores. Vue proporciona una mayor personalización, por lo tanto, es más fácil de aprender que Angular o React.

Pese a que es un framework que también dispone de muchas facilidades como angular, es tremendamente menos pesado, llegando a tener un tamaño de 80 KB.

Está basado en open-source. Tiene una amplia documentación oficial, la cual se encuentra actualizada y muy bien detallada (contiene ejemplos de código, grandes

explicaciones…), lo que hace que esté cogiendo mucha popularidad entre los desarrolladores hoy en día.

Debido a que Vue ha sido creado más recientemente, a que es una herramienta de código abierto y a que reúne lo bueno de Angular y de React, **ha sido el framework escogido para el desarrollo frontend** de esta aplicación Web (Muévete).

Servidor

Cuando hablamos de la parte del servidor de una aplicación web, nos referimos a la parte que hace referencia a la lógica de toda página web. Es decir, a la arquitectura interna del sitio que asegura que todos elementos que se encuentran en la parte front desarrollen la función correcta y, por lo tanto, que estos funcionen de la forma en que se espera. No está visible a ojos del usuario y no incluye ningún tipo de elemento gráfico.

A la hora de desarrollar Backend para una aplicación web existen varios framework y lenguajes de programación para elegir. Hoy en día los más comunes son Java, Python, Ruby… En este proyecto se ha escogido Node.js para el desarrollo de este, a continuación, veremos por qué.

Node.js

Figura 5.14. Logo Node.js

Node proporciona un entorno de ejecución del lado del servidor que compila y ejecuta JavaScript de forma óptima. Es una librería dirigida basada en eventos y por

lo tanto asíncrona que se ejecuta sobre el intérprete de JavaScript en el lado del servidor.

Node sigue la filosofía Open Source por lo que cuenta con una gran cantidad de usuarios y de documentación sobre la misma. Esto ayuda a que la curva de aprendizaje de este sea sencilla.

Otra característica de Node es que tiene la capacidad de soportar una gran cantidad de conexiones simultáneas. Puesto que en el servidor muchas veces el cuello de botella son los procesos de entradas y salidas, Node.js emplea un único hilo y un bucle de eventos asíncrono. Las nuevas peticiones son tratadas como eventos en este bucle. Este es el motivo por el que las características asíncronas y los eventos de JavaScript encajan tan bien en la filosofía de Node.js. [30]

Esto permite que Node.js sea capaz de gestionar múltiples conexiones y peticiones de forma muy eficiente, lo que lo hace apropiado para desarrollo y aplicaciones con un gran número de conexiones simultáneas.

Es una herramienta muy popular para el desarrollo de aplicaciones en tiempo real, característica que nos interesa debido a que nuestra aplicación contará con varios datos de diferentes APIs en real time.

Por otro lado, tenemos que su gestor de paquetes npm nos permite acceder a una gran cantidad de librerías Open Source desarrolladas por la comunidad, cuya instalación es sencilla y rápida.

Express

Además de Node.js, también haremos uso de Express. Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Esto nos ayudará en el desarrollo.

Express [31] es el framework web más popular de Node, y es la librería subyacente para un gran número de otros framework web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

Centrándonos en las características que queríamos para nuestra aplicación, tiempo real, sencilla y amigable, vemos que tanto vue.js (parte frontend) como node.js (parte Backend) se complementan muy bien y se adaptan perfectamente a la solución propuesta en la parte de arquitectura.

Entorno de desarrollo

Tanto para la parte del cliente como para la parte del servidor se ha empleado el mismo IDE de desarrollo. Este ha sido WebStorm [32].

Figura 5.15. Logo WebStorm

JetBrains WebStorm es un IDE profesional JavaScript que es compatible con una amplia gama de tecnologías modernas relacionadas con el lenguaje de

programación JavaScript, HTML y CSS, además ofrece la experiencia completa para el desarrollo Web productivo.

Posee varias ventajas, entre ellas destacamos:

- Esta herramienta es compatible tanto con node.js como con vue.js. Esto hace que el desarrollo en el mismo sea mucho más sencillo.
- Facilita el hacer pruebas, pudiendo debuggear la aplicación rápidamente.
- Integra una consola, por lo que el hacer cualquier cosa desde la carpeta en la que se encuentra el proyecto es más sencillo (no tenemos que abrir una nueva ventana, lo podemos hacer desde la misma vista).
- Tiene una interfaz sencilla e intuitiva.
- Posee Asistencia inteligente a la codificación.
- Es sencilla la integración con otras herramientas (Git para el control de versiones, ESLint para calidad de código…).

Por lo tanto, es una herramienta bastante completa, facilita el desarrollo del proyecto y por tanto, esta ha sido la utilizada.

Implementación

Teniendo en cuenta que en los apartados anteriores hemos dejado claros los requisitos que queríamos que cumpliese nuestra aplicación, la arquitectura que queríamos cumplir y también las herramientas y marcos de desarrollo que queremos utilizar, nos adentramos ahora en la propia implementación de Muévete.

Empezaremos hablando un poco de la funcionalidad del backend (servidor) y lo siguiente será la funcionalidad del lado cliente.

Funcionalidad Servidor

En esta sección, detallaremos qué estructura sigue la parte del servidor de la aplicación para que la lógica de la aplicación funcione y por lo tanto, proporcione los datos a la parte frontend.

Creación del Backend

Para el desarrollo del backend se utilizará node.js, necesitaremos seguir unos sencillos pasos para tener la estructura necesaria para comenzar a codificar.

1. Instalar Node.js y npm
2. Crear estructura proyecto Node.js
3. Desarrollar la aplicación

Lo primero que tendremos que hacer será instalar tanto Node.js como NPM en nuestro ordenador.

Para tener instalado Node.js será tan sencillo como irnos a su página web *https://nodejs.org/en/* y descargarnos el archivo que nos corresponda en función a nuestro sistema operativo.

Una vez que lo tenemos instalado, procedemos a instalar su gestor de dependencias, en nuestro caso NPM, para ello, abriremos una consola y por comandos pondremos “npm install”.

Teniendo ya instalados tanto Node como NPM, podremos dar comienzo a las tareas de codificación. Empezaremos por crear la estructura de nuestro proyecto.

Una vez que tenemos las herramientas necesarias, podremos crear la estructura del proyecto a través del IDE de WebStorm. Este IDE nos facilita la creación de una amplia gama de “tipos” de proyectos, todos orientados al desarrollo web.

Crearemos un nuevo proyecto desde el IDE y seleccionaremos los framework que queremos utilizar, en nuestro caso Node.js con Express. Especificaremos el nombre de nuestro proyecto. La ventana deberá ser parecida a lo que mostramos en la figura 5.16.

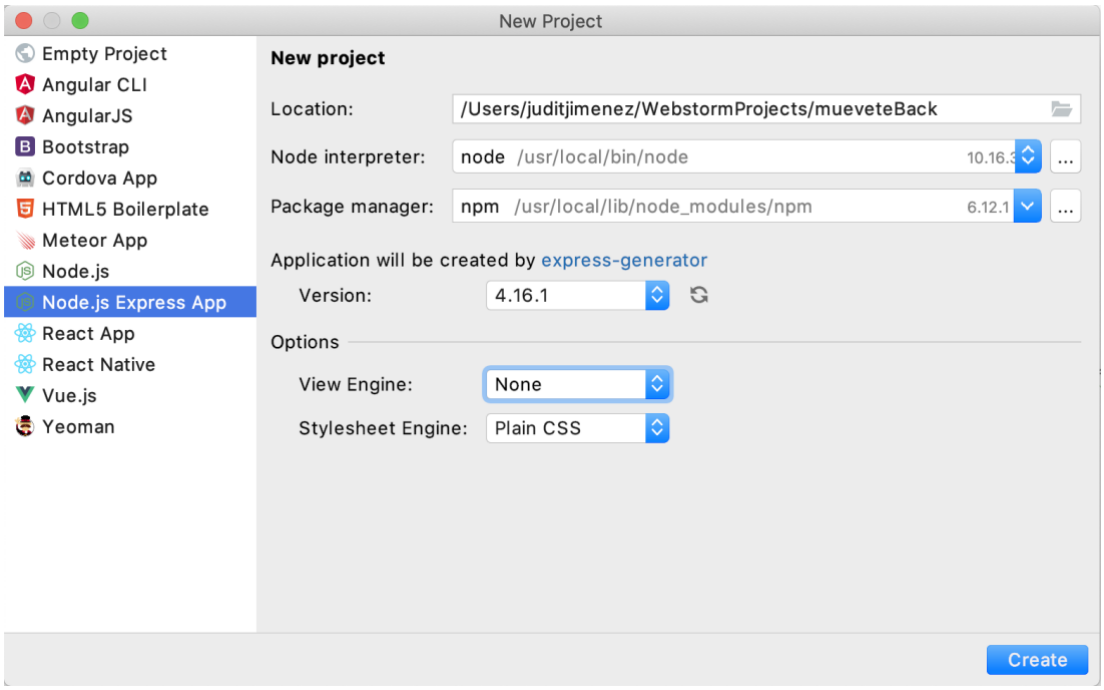

Figura 5.16. Creación estructura Backend

A continuación, al darle a "crear" nos habrá creado una estructura (la podemos ver en la figura 5.17) por defecto para poder empezar a desarrollar. Habrá archivos de esta estructura que eliminaremos, puesto que la aplicación crea un esqueleto general y que en nuestro proyecto no será necesario.

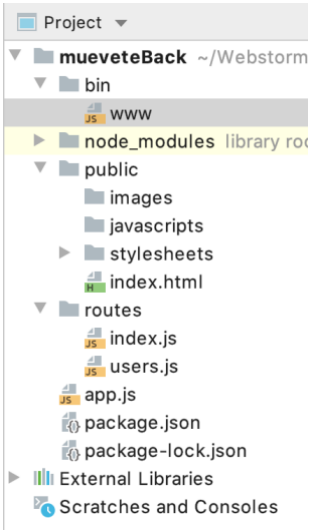

Figura 5.17. Estructura proyecto Node.js y Express

Una vez que tenemos la estructura global y como acabamos de comentar, la carpeta "public" la eliminamos, puesto que nuestro proyecto front será otro diferente.

Estructura Backend

Nuestro backend será aquel encargado de hacer las peticiones a las diferentes APIS para la obtención de los datos. Por lo tanto, tendremos tres archivos principales (los cuales hacen referencia a los controladores) dentro de la carpeta routes:

- **activities.js**: En este archivo se codificarán las llamadas necesarias al API de actividades del Ayuntamiento de Madrid. Tanto aquellas que son deportivas como las generales.
- **weather.js**: En este archivo se harán las llamadas al API del tiempo de AEMet.
- **bikes.js**: En este archivo se harán las llamadas necesarias para la obtención de las paradas de las bicicletas y del estado de estas.

En cada uno de estos archivos, además de las llamadas a los APIs, también se refinarán los datos para quedarnos con aquellos que necesitamos y enviárselos al front. De esta forma, hacemos que el front no tenga carga de datos y simplemente muestre aquello que el back le manda.

En la carpeta bin, tendremos todo lo relacionado con el servidor (número del puerto que queremos que disponibiliza…) que levanta Node para hacer frente a las peticiones y poder comunicar el front con el back.

En el archivo app.js tendremos especificadas las rutas de nuestro servidor y enlazadas a estas rutas, los controladores a los que tendrá que dirigirse para hacer las peticiones. La estructura del proyecto (figura 5.18) quedará de la siguiente forma:

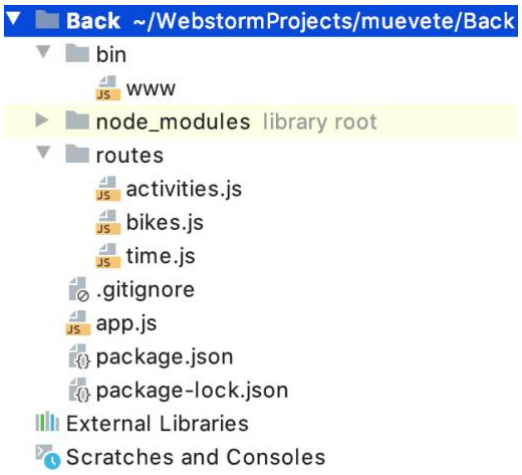

Figura 5.18. Estructura Servidor

Listado de rutas Backend

Controlador activities

Endpoints para la obtención de actividades.

VERBO	RUTA	DESCRIPCIÓN
GET	/activities/activities	Obtención de todas las actividades geolocalizadas que podemos encontrar en Madrid.
GET	/activities/sportactivities	Obtención de todas las actividades de tipo deportivas que podemos encontrar en Madrid.

Tabla 5.1. Rutas para el controlador de actividades

Controlador bikes

Endpoints para la obtención de la información de las paradas de bicicletas.

VERBO	RUTA	DESCRIPCIÓN
GET	/bikes/status	Obtención de toda la información disponible de las paradas de bicicletas de la Comunidad de Madrid.

Tabla 5.2. Rutas para el controlador de bicicletas

Controlador weather

Endpoints para la obtención de la información del tiempo.

VERBO	RUTA	DESCRIPCIÓN
GET	/weather/weatherNow	Obtención de los datos del tiempo relativos a ahora.
GET	/weather/weatherNextHour	Obtención de los datos del tiempo relativos a la hora siguiente.

Tabla 5.3. Rutas para el controlador del tiempo

Funciones más importantes

En nuestro backend, se realizan las tareas de recolección de datos, es decir, de pedir los datos que necesitamos a las diferentes APIs.

El hecho de que nos comuniquemos con tres APIs diferentes hace que pese a que las peticiones deberían ser las mismas, tengan sus diferencias. Con ser las mismas nos referimos a que en todas hacemos una petición GET, no obstante, la forma en la que debemos hacerlo varía.

Por ejemplo, detallaremos a continuación los pasos que hemos seguido para poder obtener la información del estado de todas las estaciones de bicicletas de Madrid.

1º. Registrarnos en sus servicios con un usuario y una contraseña. Este usuario y contraseña.

2º. Logarnos en sus servicios para que nos proporcionasen un API KEY. Este API KEY les servirá a ellos para identificarme y poder facilitarme sus datos. No obstante, este API KEY tiene una duración de un día, por lo que una vez al día deberemos hacer esta petición de obtención de API KEY.

3º. Añadir este API KEY en la cabecera de la petición get que hagamos desde nuestro back.

4º. Hacer la petición GET a los servicios de EMT.

5º. Esto nos devolverá todas las estaciones de bicicletas geolocalizadas y el estado de estas.

Una vez que tenemos todos los datos de estas, es el momento de procesarlos y sacar la información relevante.

Funcionalidad Cliente

En el punto anterior hemos explicado la funcionalidad de la parte del servidor de Muévete, en este apartado veremos la funcionalidad del lado del cliente.

Creación del FrontEnd

Como hemos comentado anteriormente, nuestra aplicación tendrá dos proyectos diferentes para el back y para el front. De esta forma, si en algún momento queremos cambiar una de las tecnologías, lo haremos sin afectar a la otra parte.

Antes de explicar la estructura en detalle, comentaremos los conceptos más importantes que encontramos en una aplicación front desarrollada con vue.

¿Qué son los componentes?

Los componentes son una de las características más poderosas de Vue. Te permiten extender elementos HTML básicos para encapsular código reutilizable. En un nivel alto, los componentes son elementos personalizados a los que el compilador de Vue les añade comportamiento. En ellos tenemos en un mismo archivo, el código HTML, el código CSS y también la lógica que queremos que tenga esa página. La estructura de un componente suele ser:

Figura 5.19. Estructura de un componente con Vue.js

La ventaja de crear componentes es que en un mismo fichero reunimos todo el comportamiento que queremos que tenga ese "trozo" de código front que será independiente del resto. Además, de esta forma, también logramos tener el código encapsulado e independiente entre sí, haciendo que la reutilización de estos componentes ya sea para este proyecto como para futuros, sea posible.

Una vez que tenemos claro qué es un componente, pasamos a la creación propia del proyecto Front. Lo haremos también a través del IDE de WebStorm, y escogeremos la opción de Vue.js. La ventana con la especificación del nombre del proyecto y de la versión de Node, quedará de la siguiente forma:

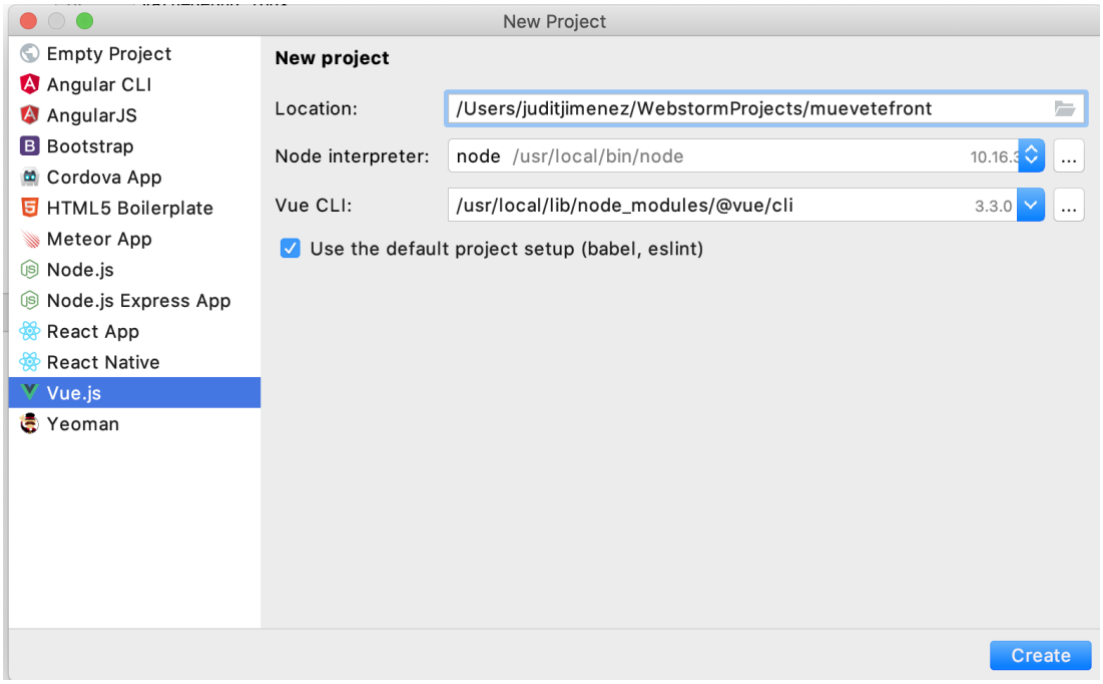

Figura 5.20. Creación proyecto Vue

Al darle a "Create" el IDE empezará a descargar y a preparar los paquetes y las librerías necesarios para que podamos empezar a desarrollar. El esqueleto de la aplicación Front quedaría:

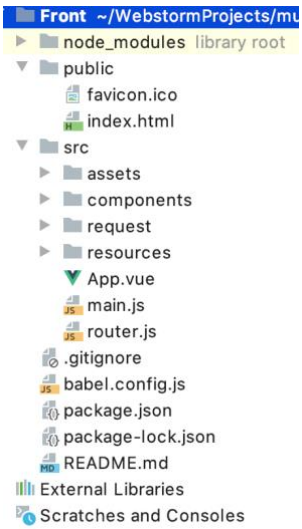

Figura 5.21. Estructura frontend

En esta estructura tendremos lo importante dentro de la carpeta src:

- En la carpeta assets tendremos todos los iconos de la aplicación, marcadores, iconos del tiempo...
- Los componentes creados en Vue estarán en la carpeta components.

- Las peticiones a nuestro backend las tendremos en request/request.js, será este archivo el encargado de hacerlas.
- En resources tenemos los recursos de la aplicación, en principio no hay nada.
- El archivo App encontramos al componente principal de Vue, a partir de este se llama a los demás componentes. Será el primer componente que cargue nuestra aplicación.
- Main.js nos encontramos con la inicialización del mapa que usamos en nuestra aplicación.
- El archivo router.js es el encargado de enrutar qué componente va con cada vista y con cada uri a la que se llama.

Funciones más importantes

Nuestra parte frontal de la aplicación, también se divide en tres vistas bien diferenciadas. En la página principal de la aplicación, podremos ver el estado en tiempo real de las estaciones de bicicletas (consultando no sólo la localización de las estaciones, si no también información relevante sobre las bicicletas disponibles) de La Comunidad de Madrid. Además, desde esta misma vista, tendremos acceso a información meteorológica en tiempo real tanto en la actualidad, como una previsión a una hora vista.

Quizás por todo el contenido que tiene la primera pantalla, creo que es de las más interesantes que nos encontramos en el proyecto.

Lo primero que hará nuestra aplicación será leer el archivo “Main.js”, en el mismo nos encontramos la inicialización del mapa. En nuestro caso, utilizamos un componente que ya existe para cargar el mapa de google llamado “GmapCluster”. Este componente, para inicializarse, necesita una KEY que obtendremos de la consola de Google. El acceso y la utilización es totalmente gratuita, lo único que haremos será entrar en el apartado de Maps y crear el API KEY.

1. Iremos a Google Cloud Platform Console.
2. Haremos clic en el botón del menú y seleccionaremos crear proyecto del tipo "Maps JavaScript API". Rellenamos los datos que nos piden.
3. Haremos clic en el botón desplegable (tres puntitos) y seleccionaremos APIs & Services -> Credentials.
4. En la página de Credentials, haremos clic en Crear Credentials -> API Key. Esto creará un string que será nuestro identificador. Será el que necesitaremos copiar y pasarle por parámetros al componente descrito anteriormente.

El componente inicializado en main.js tendrá la siguiente forma:

```
Vue.use(VueGoogleMaps, options: {
  load: {
    key: 'NUESTRA API KEY',
    libraries: 'places',
  },
});
```

Figura 5.22. Librería VueGoogleMaps

Una vez que tenemos esto especificado, ya le podremos decir a Vue que utilice GmapCluster como componente.

```
Vue.component('GmapCluster', GmapCluster);
```

Gracias a la importación de este componente, en nuestra página web principal, con tan solo utilizar los componentes "GmapMap" y "GmapMarker", tendríamos creado nuestro mapa principal.

Estos componentes admiten una serie de parámetros que facilitan el desarrollo y la creación del desarrollo web con mapas de google. En el caso de GmapMap, podemos indicarle dónde queremos que se centre el mapa y también el zoom que queremos que tenga al iniciarse la aplicación. En nuestro caso, especificamos latitud y longitud de Madrid y para que se vea bien, pondremos un zoom de 15.

En el caso de los marcadores, GmapMarker, también dispone de parámetros, como por ejemplo, la geolocalización en la que queremos que aparezca, el icono que queremos que tenga…

```
<GmapMap :center="{lat:40.416775, lng:-3.703790}" :zoom="15">
   <GmapMarker
       v-for="(keys, index) in biciStation" :key="index"
       :position="google && new google.maps.LatLng(keys.geometry.coordinates[1],keys.geometry.coordinates[0])"
       :clickable="true"
       :icon="{ url: colorMarker(keys.light)}"
       @click="toggleInfoWindow(keys)"
    >
    </GmapMarker>
</GmapMap>
```

Figura 5.23. Componente Mapa y Marcadores

Con esto ya logramos que la aplicación muestre un mapa centrado en Madrid y con la información de los marcadores que recogemos al hacer la llamada desde el back a la EMT. Pero como habíamos comentado, también mostramos información acerca del tiempo.

Para ello, hacemos la petición al back correspondiente para que llame a la AEMet y nos traiga toda la información que queremos (estado del cielo, orientación y velocidad del tiempo y la temperatura). Con toda esta información lo que hacemos es mostrarla en la barra de navegación.

Capítulo 6.- Manual de usuario

En este punto detallaremos el funcionamiento de la aplicación con el objetivo de facilitar su uso al usuario.

La aplicación

Muévete es una aplicación web cuyo objetivo es mostrar de forma sencilla el estado de las estaciones de bicicletas de la comunidad de Madrid. Por cada estación de bicicletas se mostrará en un mapa su localización y también información sobre cuántas bicicletas libres y cuántos anclajes libres quedan en la misma. En todo momento se mostrará datos relativos al tiempo, temperatura actual en la ciudad, dirección y velocidad del viento y el estado del cielo.

Requisitos

Para poder acceder a Muévete es necesario tener un dispositivo con conexión a internet y disponer de un navegador web para acceder a la dirección del sistema.

Manual de Usuario

Uno de los requisitos de Muévete es que fuese sencillo de usar, por lo que es una aplicación web que de un solo vistazo proporciona diferente información. Tiene tres vistas principales, en las que puedes navegar directamente de unas a otras. A continuación se muestra la aplicación.

Vista principal

El objetivo principal de Muévete es mostrar en tiempo real datos sobre las estaciones de bicicletas y a la vez, proporcionar datos meteorológicos. Por lo tanto, la vista principal es aquella que reúne estos dos requisitos.

Cuando se accede a la aplicación de Muévete, nos encontraremos con esta pantalla:

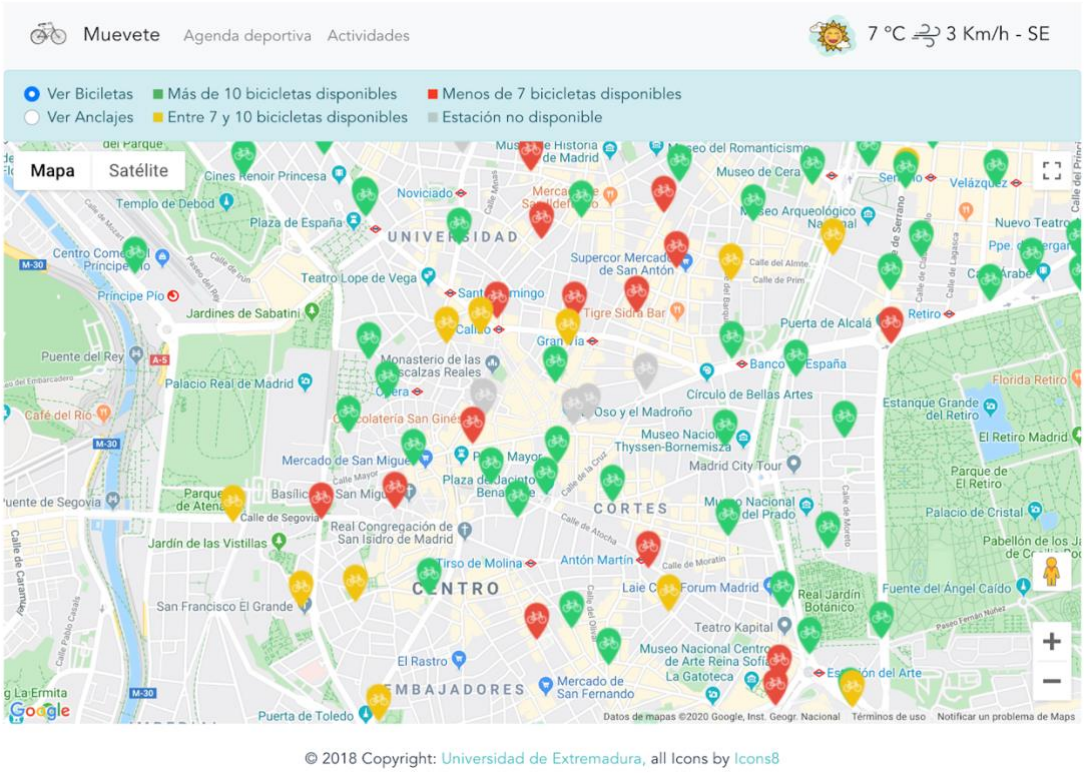

Figura 6.1. Muévete, pantalla principal

En esta vista podemos apreciar una barra de navegación en la parte superior. Desde ella tenemos acceso a dos pestañas “Agenda deportiva” y “Actividades”. Estas son las pantallas relativas a la información de las actividades que nos podemos encontrar en Madrid, las veremos más adelante.

En la parte final de la barra de navegación, podemos ver la información relativa al tiempo.

Tenemos un icono que representa el estado del cielo en tiempo real, vemos la temperatura actual en la ciudad y también la velocidad y dirección del viento. Además, clicando en cualquier elemento relativo al tiempo, nos aparecerá un pop-up indicándonos una previsión de estos mismos datos en la siguiente hora. Este pop-up no ocupará toda la pantalla para permitir seguir viendo los datos anteriores.

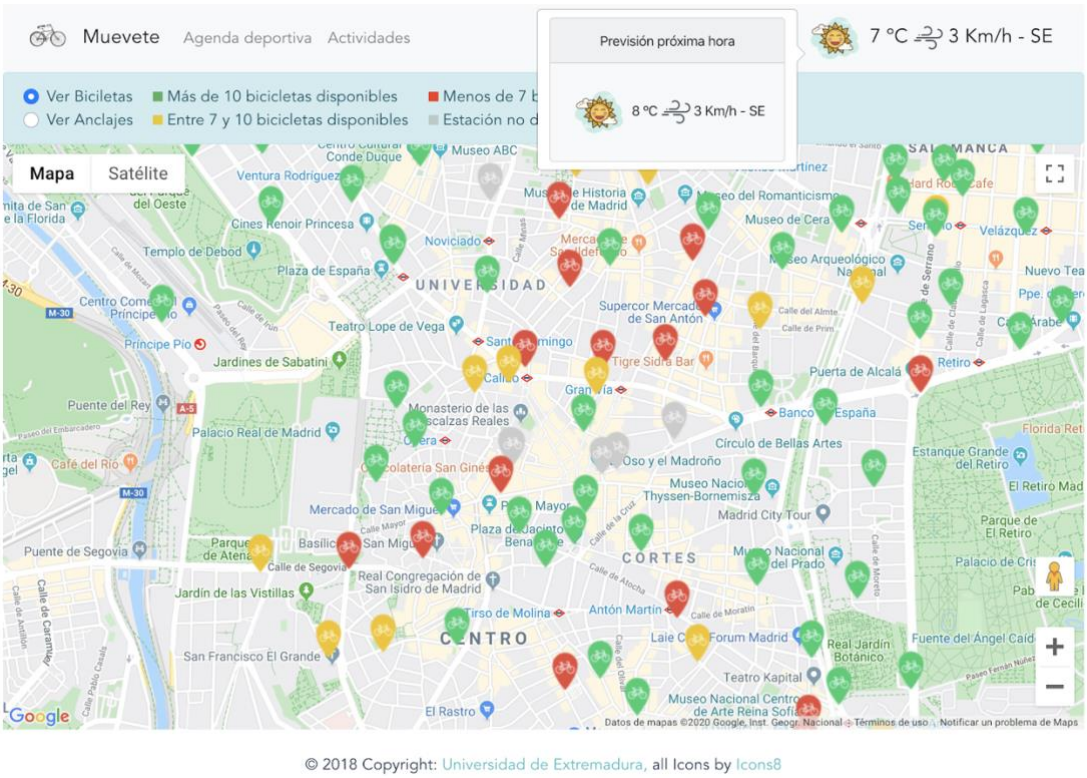

Figura 6.2. Muévete, información de previsión del tiempo

Como se puede ver, el estado del cielo aparece en forma de icono. Nos podremos encontrar varios en función del estado, estos iconos son:

- Cielo cubierto o nuboso, el icono que representa este estado es:

Figura 6.3. Cielos nubosos

- Chubascos leves, lluvia, el icono que representa este estado es:

Figura 6.4. Cielos con lluvia

- Si la lluvia es más fuerte e incluso hay granizo, el icono para representar dicho fenómeno será:

Figura 6.5. Tormenta

- Para el caso de que el estado del cielo sea nieve, el icono será:

Figura 6.6. Nieve

- Para el caso de que el cielo esté despejado, el icono que aparecerá será:

Figura 6.7. Despejado

- En el caso de que no esté despejado pero las nubes sean altas, el icono será:

Figura 6.8. Nubes altas, poco nuboso

- En el caso de que haya niebla, calima o bruma, el icono será:

Figura 6.9. Niebla

Lo que más destaca de esta pantalla es el mapa. Este mapa contiene unos marcadores de colores que son los que representan las estaciones de bicicletas. Estos colores dependen de la disponibilidad de bicicletas/anclajes (**por defecto serán bicicletas disponibles**) de la estación.

Estos colores representan:

- Verdes: Más de 10 bicicletas/anclajes disponibles.
- Amarillo: Entre 7 y 10 bicicletas/anclajes disponibles.
- Rojo: Menos de 7 bicicletas/anclajes disponibles.
- Gris: Estación cerrada, no disponible.

Toda esta información aparece a modo de leyenda en la pantalla, debajo de la barra de navegación, de color azul. Además, en esta parte también podremos elegir si queremos que los marcadores aparezcan coloreados por disponibilidad de bicicletas (opción Ver Bicicletas) o por anclajes (opción Ver Anclajes).

Si hacemos clic en un marcador la información disponible aparecerá en un pop-up justo donde se encontraba el marcador, por lo que permite ver el resto del mapa y la información del tiempo en todo momento.

La información para cada estación será siempre (tengamos seleccionado anclajes o bicicletas) la siguiente:

- Nombre de la estación
- Dirección de la estación
- Número de bicicletas disponibles
- Número de anclajes disponibles para dejar la bicicleta.

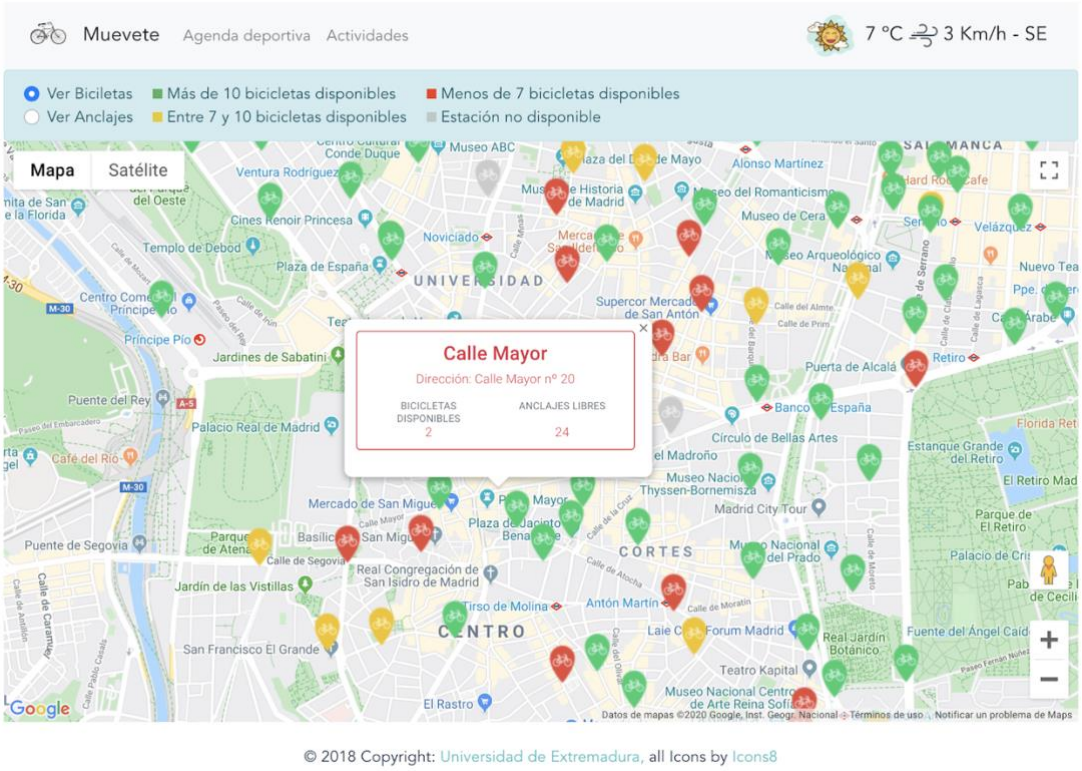

Figura 6.10. Muévete, selección de una estación de bicicletas

Como se puede comprobar, desaparece el marcador y aparece un pop-up con esta información. Para no olvidarnos del color de este, la información relevante aparecerá del mismo color que el marcador que representaba la estación.

En este caso, estábamos mostrando los marcadores por disponibilidad de bicicletas, podemos ver que en esta estación hay pocas bicicletas disponibles, por lo que el color es rojo.

En el caso de que queramos ver los marcadores coloreados en función de la disponibilidad de anclajes libres, haremos clic en la opción "Ver Anclajes" que aparece en la parte superior izquierda. En ese momento, se recarga el mapa con los marcadores del tipo anclajes (serán iguales que los anteriores pero con una "P" de "parking" encima de la bicicleta). Además, la leyenda cambiará y en vez de poner "bicicletas" aparecerá "anclajes".

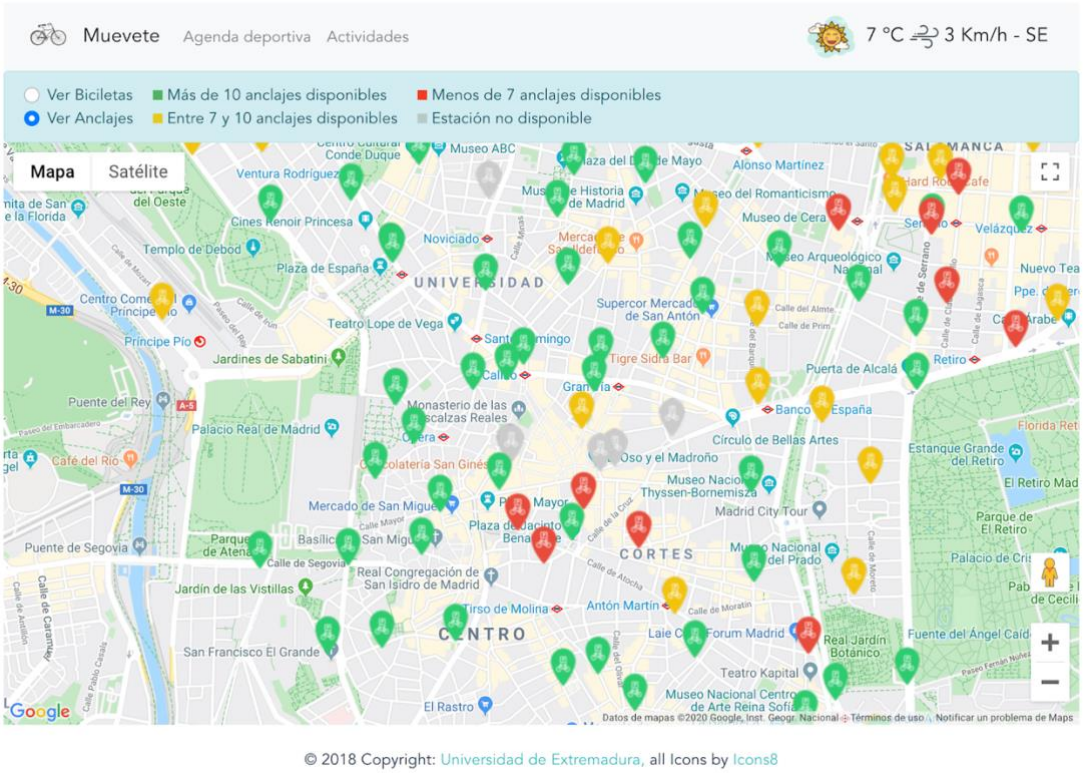

Figura 6.11. Muévete, visualización del mapa por anclajes

Como podemos comprobar, los marcadores se han modificado, no obstante, la información que se muestra es prácticamente la misma que en el caso anterior.

Ahora, la estación seleccionada en la fotografía anterior aparece en verde, puesto que había más de 10 anclajes disponibles. Como se puede comprobar, el pop-up que aparece también está de otro color, siempre haciendo referencia a los anclajes disponibles (para el caso de "ver anclajes").

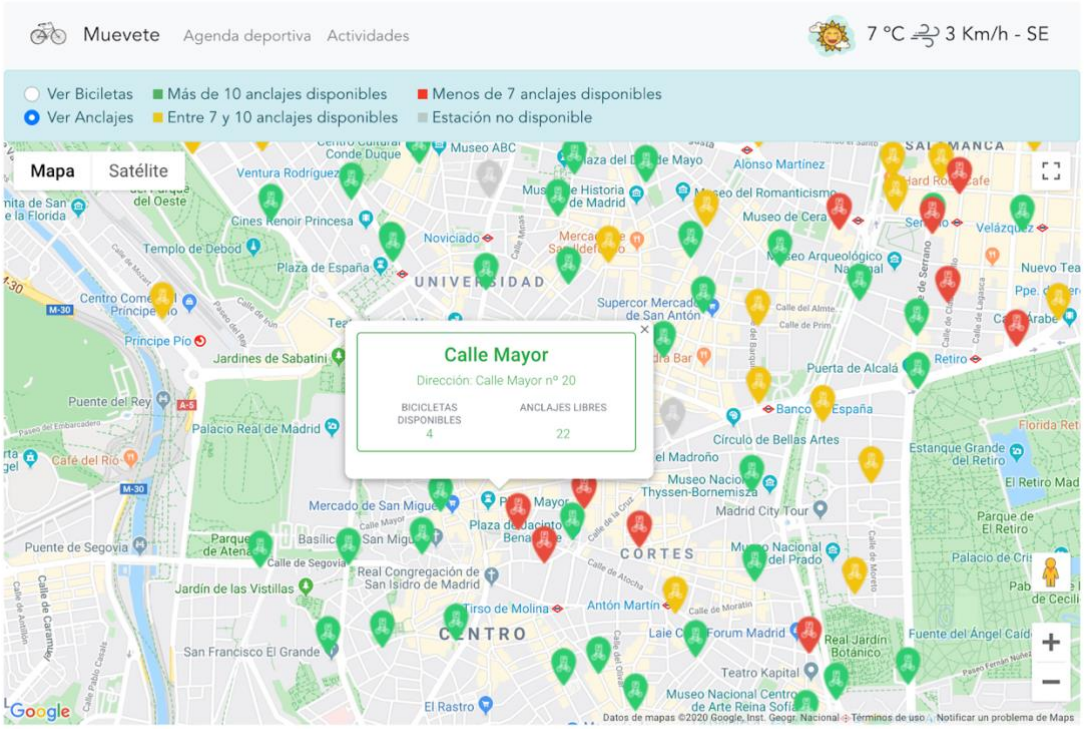

Figura 6.12. Muévete, selección de una estación para ver la información por anclajes

Vista, agenda deportiva

Por otro lado, en Muévete también podemos encontrar un listado de actividades deportivas que se llevan a cabo en la ciudad de Madrid. Para poder ver esta información nos dirigiremos a "Agenda deportiva", la cual nos muestra la siguiente pantalla:

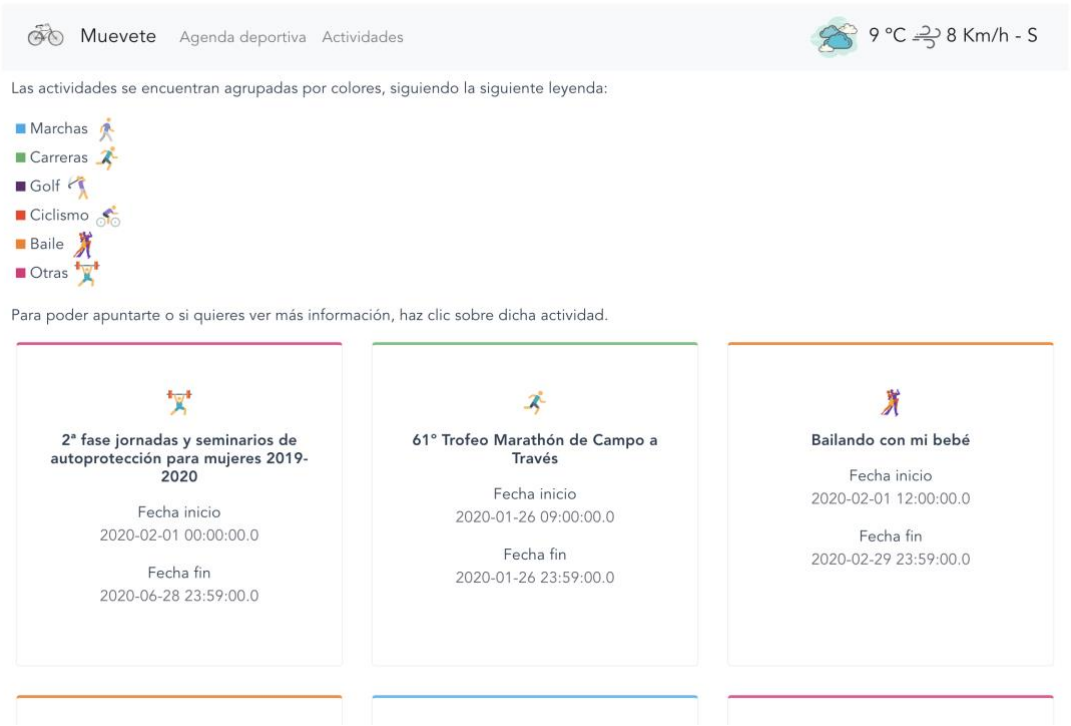

Figura 6.13. Muévete, vista Agenda deportiva

Como podemos ver, la barra de marcadores no ha cambiado, tendremos siempre disponible en todo momento la información del tiempo y si clicamos sobre ella, la información relativa a la previsión del tiempo. Desde la misma, podremos volver a la pantalla principal haciendo clic sobre "Muévete". En el caso de que queramos ver las actividades de diferente tipo a las deportivas, podremos ir a su pantalla correspondiente haciendo clic en "Actividades".

En la pantalla de agenda deportiva, tendremos una leyenda que nos indica el icono y el color de las diferentes actividades que podemos ver en Madrid. A continuación, tendremos un listado de estas en cuadrados, con un margen superior del color correspondiente a su tipo de deporte. Por cada actividad veremos:

- Nombre de la actividad
- Fecha inicio
- Fecha fin

Además, haciendo clic en cada una de ellas, accederemos a la página principal de la actividad seleccionada para poder apuntarnos y ver información más detallada. Este link será el que se dispone en el propio portal del ayuntamiento de Madrid.

Vista Actividades

Por último, la sección "Actividades" mostrará en un mapa las diferentes actividades de toda índole (salvo deportivas) que dispone el ayuntamiento de Madrid.

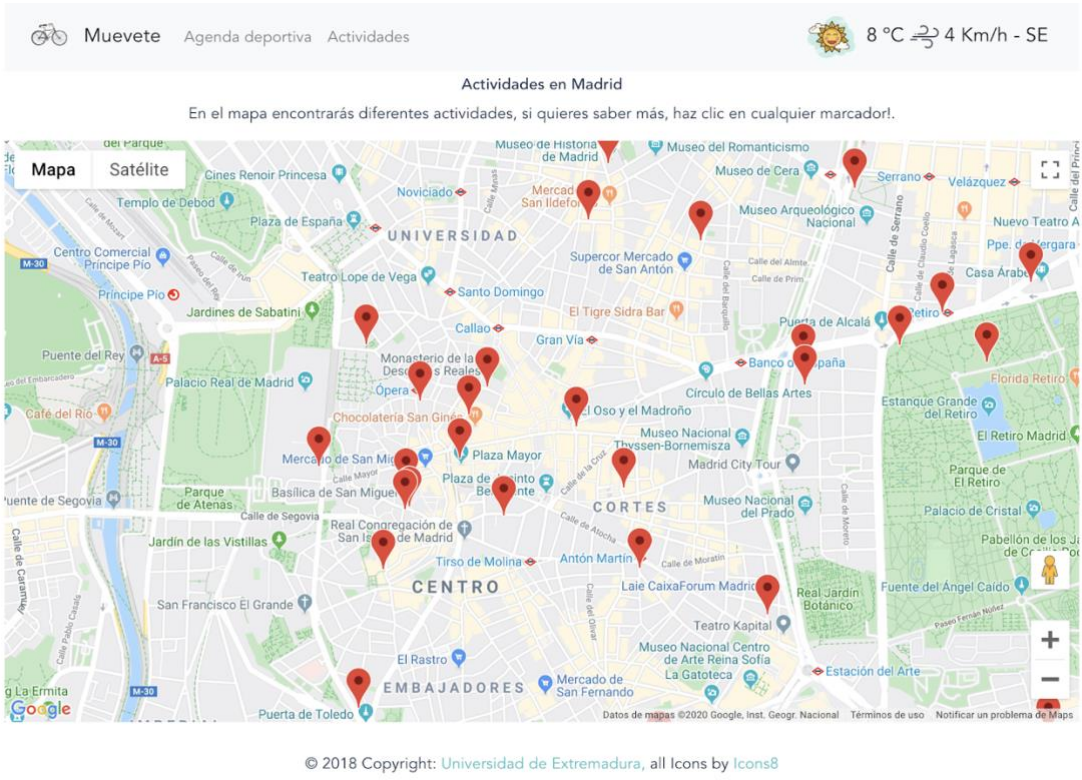

Figura 6.14. Mapa actividades

Si hacemos clic en el marcador, veremos información sobre la actividad en concreto, esta información será:

- Nombre evento
- Dirección
- Fecha inicio
- Fecha fin
- Link a la actividad

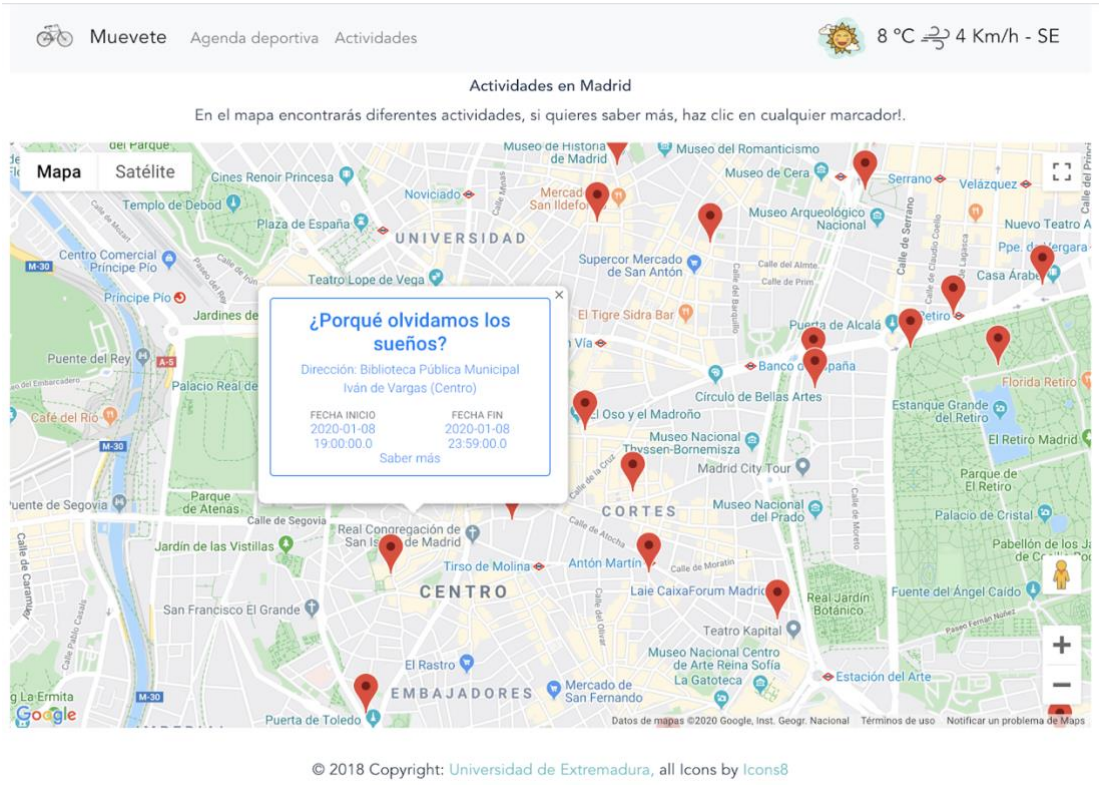

Figura 6.15. Muévete, Información de una actividad

Si queremos saber más información sobre el evento en concreto, tendremos un link en “Saber más” que nos llevará a la propia web del Ayuntamiento.

Capítulo 7.- Planificación

La planificación de este proyecto se divide en dos fases principales, la primera corresponde al estudio del arte realizado para llevar a cabo la primera idea (desarrollar una aplicación en tiempo real de geolocalización de cualquier medio de transporte para cualquier ciudad del mundo), la segunda fase corresponde con la idea desarrollada finalmente, Muévete (aplicación en tiempo real sobre bicicletas en Madrid, cohesionada con el tiempo y con el Ayuntamiento de Madrid para mostrar actividades).

- **Análisis de mercado**. En esta primera etapa buscamos herramientas que hiciesen algo parecido a lo que nosotros queríamos hacer. Necesitábamos saber cómo estaba el mercado en este aspecto, cuánto de desarrollado y también ver si aportaba valor al ciudadano. En esta etapa vimos que no había ninguna parecida a nuestra idea, que teníamos a nuestra disposición herramientas (APIs a las que llamar de forma gratuita) y por lo tanto, la idea era viable.

- **Definición de requisitos**. El siguiente paso fue definir qué queríamos que nuestra aplicación cumpliese. En esta etapa fue necesario la realización de diagramas de clases, de casos de uso y de actividad.

- **Elección de herramientas**. Una vez que teníamos claros qué requisitos tiene nuestra aplicación y antes de ponernos a codificar, hicimos un estudio con las herramientas más sonadas en el mundo del desarrollo. De esta forma, vimos cuál era la que mejor encajaría. En esta etapa además de terminar de elegir las tecnologías, también añadiría los tutoriales y la documentación leída para formarnos sobre dichas herramientas antes de programar con ellas.

- **Implementación**. Tras tener todos los requisitos claros y las herramientas a usar claras, el siguiente paso es empezar a codificar. Pese a que el proyecto se divide en parte Back y parte Front, se ha ido

programando por funcionalidad completa. Es decir, no se ha centrado todo el trabajo en hacer primero el back y luego en hacer el front, ambas partes se han ido haciendo a la par. En este apartado se incluyen las llamadas a todos los APIs necesarios (AEMeT, EMT y Ayuntamiento de Madrid), al procesado de sus datos y a mostrar dichos datos al cliente.

- **Documentación**. En la última parte se ha elaborado un documento en el que se ha recogido todo el proceso de creación del proyecto. Pese a que parece estar en último puesto, este documento se ha ido desarrollando a lo largo de todo el proyecto, no obstante, ha sido en la etapa final cuando se le ha dado más forma al mismo.

A continuación se muestran unos gráficos en el que se detallan las horas dedicadas a cada parte del proyecto.

Estudio de la idea original

Como hemos comentado en apartados anteriores, el proyecto comenzó con la idea de hacer una aplicación global que permitiese visualizar la flota de transportes de cualquier ciudad y cualquier empresa. Pese a que finalmente no se llevó a cabo, es tiempo que también debemos contar, ya que ha sido parte importante del estudio de la aplicación final. A esta idea, se le dedicaron las horas que representamos en la figura 7.1 con un diagrama de barras.

Figura 7.1. Horas dedicadas a la idea original

La tabla 7.1 muestra la correspondencia de la fase del proyecto con las horas de una forma más detallada.

	Enero	Febrero	Marzo	Abril	Julio	Agosto	Septiembre	Octubre
Análisis de mercado	8,5	10,5						
Elección de herramientas		3,15			6,32			
Documentación		2,3	2	1,23				
Definición de requisitos			3	1,43				
Implementación						19,34	22,25	28

Tabla 7.1. Horas dedicadas a la idea original

Estudio de la idea final

Para el análisis y desarrollo de Muévete, el tiempo dedicado es el que mostramos en la figura 7.2.

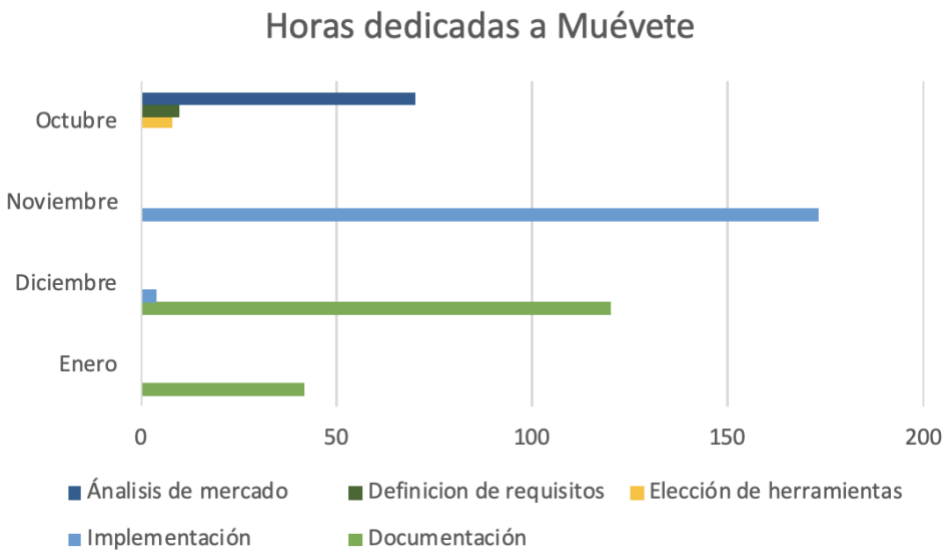

Figura 7.2. Horas dedicadas a la idea final por etapas

La tabla 7.2 muestra la correspondencia de la fase del proyecto con las horas más detallada.

	Octubre	Noviembre	Diciembre	Enero
Análisis de mercado	70			
Definición de requisitos	10			
Elección de herramientas	8			
Implementación		173	4	
Documentación			120	42

Tabla 7.2. Horas dedicadas a la idea final

Capítulo 8.- Conclusiones

Problemas encontrados

Durante todo el proceso de desarrollo se han encontrado numerosos problemas. El más destacable fue al comienzo. Teníamos claro que queríamos hacer una aplicación en tiempo real y que los datos fuesen geolocalizados. Fuimos ambiciosos y pensamos en hacer una aplicación global para controlar la flota de transporte de cualquier ciudad del mundo. Con esta idea en mente, estuvimos desarrollando una posible solución, haciendo un gran estudio del arte y recopilando datos para poder hacerlo. No obstante, nos dimos cuenta de que sin un estándar que seguir, no había sinergias entre los diferentes datasets que consultamos (teníamos un dataset de un tranvía de Italia, otro de autobuses localizado en Zaragoza y el de autobuses de Cáceres). Entre ellos no había ninguna similitud y la forma en la que se obtenían también eran completamente diferente, por lo que era muy costoso y casi imposible de desarrollar dicha idea.

En este tiempo, se hacía frustrante ver que no se avanzaba con la rapidez que se esperaba y que tampoco había muchas soluciones.

A la hora de hacer el proyecto actual, los problemas más difíciles han sido los de ponernos en contacto con las diferentes instituciones y empresas que nos proporcionaban los datos y entender cómo lo hacían, esto hacía que los tiempos se alargasen, ya que no contestaban de forma inmediata. No obstante, siempre han sido muy educados y han estado en plena disposición.

Trabajos futuros

Pese a que hemos conseguido que Muévete sea una aplicación web que funciona bien, que cumple los requisitos deseados y por lo tanto, un desarrollo hecho con éxito, creemos que hay algunas funcionalidades extra que son interesantes y que aportan valor al usuario que la utilizará.

- Posibilidad de guardar estaciones favoritas, de tal forma que se puedan filtrar por las mismas.
- Posibilidad de buscar las estaciones dando un distrito de Madrid.
- Añadir un chat entre usuarios para que puedan interactuar y hacer quedadas por la misma aplicación, para enterarse de nuevas actividades o para quedar e ir en grupo a las mismas.
- Marcar en qué estación quieres dejar la bicicleta que has alquilado.
    - A este extra, añadirle la opción de que salte una notificación en el caso de que no haya anclajes disponibles.

Enriquecimiento personal

Pese a que se han encontrado problemas, creo que el desarrollo de este proyecto me ha hecho crecer tanto profesional como personalmente.

Durante la carrera se hacen diferentes proyectos pero ya parten de una idea y de unos requisitos establecidos. Sin embargo, este ha sido de los primeros que absolutamente todo el proceso de desarrollo dependía de mí. Esto ha hecho que sea más consciente de todo el trabajo que hay detrás de un desarrollo y de un proyecto software, de las partes que hay implicadas y del esfuerzo que hay que hacer para que una idea salga adelante. Teniendo en cuenta el sector en el que nos encontramos, en el que es tan difícil ver y cuantificar el trabajo que conlleva un desarrollo, creo que es una experiencia que aporta mucho valor.

Por otro lado, también puedo decir que gracias a este proyecto me reafirmo en que me encanta desarrollar ideas dinámicas, en tiempo real y sobre mapas, por lo que creo que me ha despertado mucha curiosidad en este ámbito y que sin duda, intentaré perseguir en mi entorno laboral.

Me gusta el trabajo que he realizado, el resultado de este y todo el proceso que he seguido. Con esto pongo punto final a mi etapa de universidad… al menos, por ahora.

Anexo A

Guía de instalación

Para poder hacer uso de la aplicación, necesitaremos tener descargada e instalada en nuestros equipos la siguiente herramienta:

- Node.js con la versión v10.16.3
- Npm, herramienta de gestión de paquetes de Node con la versión 6.12.1

Una vez que tenemos el código en nuestro equipo, nos encontraremos dos carpetas. Una llamada "Back" en la que nos podremos encontrar la parte del servidor de la aplicación; y otra llamada "Front", será en la que nos encontremos la parte del cliente de la aplicación.

Para que la aplicación funcione, deberemos tener disponible el puerto 3000 ya que sobre él se arrancará la parte del servidor y por defecto la parte cliente estará en el 8080.

Necesitaremos abrir en ambas carpetas una terminal (preferiblemente, arrancaremos primero la parte del servidor) y ejecutar sobre ellas los siguientes comandos:

- `Npm install.` Con este comando nos descargaremos e instalaremos las dependencias que se necesitan para que el proyecto funcione, como por ejemplo Vue.
- `Npm run start.` Con este comando lo que haremos será inicializar la aplicación. Nos dispondrá el proyecto en un puerto específico de nuestro ordenador y dirigiéndonos a él ya podremos disfrutar de Muévete.

Una vez **que ambos proyectos** se encuentren inicializados, veremos que en la consola donde hemos ejecutado los comandos para inicializar la parte del **Front**, nos especifica dos URLs en las que disponemos la aplicación lista para nuestro disfrute.

Referencias bibliográficas

[1] https://opendatahandbook.org/guide/es/what-is-open-data/
[2] https://public.resource.org/8_principles.html
[3] https://opendatahandbook.org/glossary/en/terms/five-stars-of-open-data/
[4] https://dvcs.w3.org/hg/gld/raw-file/default/glossary/index.html#x5-star-linked-open-data
[5] https://u.bicimad.com/mapa
[6] https://www.bizizaragoza.com/es/availability_map
[7] http://www.dublinbikes.ie/All-Stations/Station-map
[8] Roger S.Pressman. *Ingeniería del Software. Un enfoque práctico. Editorial: McGraw-Hill Séptima edición 2010.*
[9]https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a
[10] https://es.scribd.com/document/79981138/Casos-de-Uso-Teoria
[11] https://opendata-ajuntament.barcelona.cat/data/en/dataset/informacio-estacions-bicing
[12] https://apidocs.emtmadrid.es/#api-Block_4_TRANSPORT_BICIMAD
[13] https://www.zaragoza.es/sede/portal/datos-abiertos/solr#bizi
[14] http://gobiernoabierto.valencia.es/en/dataset/?id=estaciones-valenbisi
[15] http://sevilla-idesevilla.opendata.arcgis.com/datasets/8979868482d84138b2ea236f104f3a7f_0
[16] http://www.aemet.es/es/datos_abiertos
[17] https://opendata.aemet.es/centrodedescargas/inicio
[18] https://hotelmix.es/widgets/weather
[19] https://www.tiempo.com/widget/
[20] https://datos.madrid.es/portal/site/egob/
[21] https://datos.madrid.es/portal/site/egob/menuitem.214413fe61bdd68a53318ba0a8a409a0/?vgnextoid=b07e0f7c5ff9e510VgnVCM1000008a4a900aRCRD&vgnextchannel=b07e0f7c5ff9e510VgnVCM1000008a4a900aRCRD&vgnextfmt=default
[22] https://www.zaragoza.es/sede/servicio/catalogo/282
[23] https://sede.madrid.es/portal/site/tramites/menuitem.62876cb64654a55e2dbd7003a8a409a0/?vgnextoid=bc68bb2b19146410VgnVCM1000000b205a0aRCRD&vgnextchannel=ed1aa38813180210VgnVCM100000c90da8c0RCRD&vgnextfmt=default
[24] https://www.soydezaragoza.es/mapa-estaciones-bizi-zaragoza/
[25] https://blog.infranetworking.com/modelo-cliente-servidor/
[26] https://concepto.de/http/
[27] https://medium.com/@reyes.leomaris/aplicando-el-patr%C3%B3n-de-dise%C3%B1o-mvvm-d4156e51bbe5
[28] https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces
[29] https://platzi.com/tutoriales/1153-angular/1619-que-son-los-componentes-en-angular/
[30] https://www.luisllamas.es/que-es-node-js/
[31] https://comprasoft.com/jetbrains/webstorm
[32] https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Herramientas externas

- Lucidchart:
    - https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml

- BonitaSoft:
    - https://es.bonitasoft.com/

- JSONFormatter & Validator:
    - https://jsonformatter.curiousconcept.com/

- Canva:
    - https://www.canva.com/

- Iconos8:
    - https://iconos8.es/icons

Enlaces al código

El proyecto se puede encontrar en la siguiente ruta:

- https://bitbucket.org/jjimenezjdm/muevete/