Creación de WebApp para automatizar los procesos de la organización EnBiciSenseEdat

Jonathan De La Cruz Carrillo

Resumen—La organización EnBiciSenseEdat lleva a cabo un proyecto de voluntariado en el que quiere mejorar la vida de las personas de avanzada edad y personas con movilidad reducida, mediante la organización de paseos en triciclos eléctricos especialmente adaptados, conducidos por personas voluntarias.
Este proyecto se lleva a cabo desde el año 2016, aunque la gestión manual del mismo (contacto con voluntarios, gestionar la disponibilidad de estos, organizar los paseos…), impide que el proyecto crezca y se pueda gestionar un número mayor de triciclos a la vez.
Para ello, la organización quiere implementar un sistema capaz de gestionar automáticamente todo el proceso que va desde el momento en que una persona se apunta a este programa, hasta que realiza el paseo en el triciclo, así como la comunicación entre la asociación y los voluntarios. El objetivo final, es poder adquirir un mayor número de triciclos y poder realizar más paseos sin que la gestión de estos sea un impedimento.

Paraules clau—Single-page Application, front-end, back-end, Angular, Node.js

Abstract—The EnBiciSenseEdat organization carries out a volunteer project in which it wants to improve the lives of the old people and people with reduced mobility, by organizing rides on a specially adapted electric tricycles, driven by volunteers.
This project has been carried out since 2016, although manual management of it (contact with volunteers, manage their availability, organize rides…), prevents the project from growing and can manage a greater number of tricycles at the same time.
For this reason, the organization wants to implement a system capable of automatically managing the entire process that goes from the moment when a person signs up for this program, until they take the ride with the tricycle, as well as the communication between the association and the volunteers. The final objective is to be able to acquire a greater number of tricycles and to be able to take more rides without the management of these being an impediment.

Index Terms— Single-page Application, front-end, back-end, Angular, Node.js

1 Introducción

ESTE documento explica el proceso de desarrollo de un sistema del estilo WebApp, para la organización EnBiciSenseEdat. El objetivo final es automatizar los procesos requeridos para gestionar las actividades que realiza, los cuales, actualmente, se hacen de forma manual.

Durante el desarrollo de este informe se hablará sobre la organización EnBiciSenseEdat, los frameworks Angular y Nodejs, los objetivos del proyecto, el estado del arte, la metodología que se ha seguido, la planificación, los resultados obtenidos y una serie de conclusiones y agradecimientos.

1.1 EnBiciSenseEdat

En esta sección se introducirá la organización para la cual se ha desarrollado el sistema, las necesidades que la han llevado a realizar el proyecto y los beneficios que éste aportará.

La organización mencionada, EnBiciSenseEdat [1], nace sin ánimo de lucro y con finalidad de estimular la participación activa de la sociedad en la mejora de la calidad de vida y el bienestar de las personas grandes y de movilidad reducida, a partir de la organización de paseos en unos triciclos eléctricos especialmente adaptados, conducidos por personas voluntarias.

El nombre "EnBiciSenseEdat", representa a la comunidad que representa a esta organización en Cataluña. El origen, "Cycling Without Age" se sitúa en el 2012 en Copenhagen y a partir de ese inicio, se ha expandido hasta 50 países diferentes y más de 2200 ubicaciones diferentes. Además de las ubicaciones, también hay que mencionar toda la gente implicada, obteniendo más de 33.000 pilotos de triciclo y gestionando más de 3.000 triciclos.

1.2 Angular

Angular [2], es un framework de código abierto para aplicaciones web desarrollado en TypeScript. Esta desarrollado por Google y tiene como objetivo la creación de aplicaciones del estilo "Single-page application" [3].

En el proyecto, este framework se utiliza para desarrollar el front-end de la aplicación, lo cual ofrece una amplia gama de componentes fáciles de integrar, personalizar según las especificaciones requeridas y testear una vez añadidos al proyecto.

Se ha escogido este framework debido a las facilidades que ofrece para controlar la comunicación con el back-end y para el diseño de la interfaz de usuario.

1.3 Nodejs

Node.js [4], es un controlador de eventos asíncrono desarrollado en Javascript. Está diseñado para crear aplicaciones web escalables. Funciona a partir de un bucle de eventos al que ingresa después de ejecutar el scipt de entrada y sale cuando no hay más devoluciones de llamada para realizar, el resto del tiempo permanece dormido. Es capaz de recibir múltiples llamadas simultáneas y pemite responder a estas de manera individual. Como no ejecuta casi ninguna función de Entrada / Salida directamente, el proceso nunca se bloquea y puede seguir recogiendo eventos.

En el proyecto, se utiliza Node.js para crear el servidor que actua como back-end, respodiendo las llamadas del cliente o front-end, accediendo a la base de datos y transformando los datos necesarios para dar la respuesta necesaria.

Se ha escogido esta herramienta debido a las facilidades que ofrece a la hora de desarrollar un servicio de API REST, que es el que se ha escogido para el proyecto.

1.4 MySQL

MySQL [5], es un sistema de gestión de bases de datos relacional, desarrollado por Oracle Corporation y de código abierto. Está considerada como una de las bases de datos más populares del mundo. Se utiliza mucho en aplicaciones web, en plataformas como Linux y en herramientas de seguimiento de errores como Bugzilla.

En el proyecto, se utiliza MySQL para guardar los datos o información que se va a gestionar en el proyecto, como las personas que van a realizar un paseo, la fecha o el vehículo que se va a utilizar.

Se ha escogido esta herramienta debido a que es una base de datos muy rápida en la lectura y, principalmente, el proyecto va a realizar operaciones de lectura para mostar la información solicitada por el usuario.

1.5 Servicio REST

Como último apartado tecnológico, se va a explicar el funcionamiento de un servicio REST ya que es la metodología que se ha escogido para este proyecto.

Un servicio REST [6] es un conjunto de restricciones con las que se puede crear un estilo de arquitectura software, que se usará para crear aplicaciones web respetando el protocolo HTTP.

En el proyecto se utiliza el servicio REST como método de comunicación entre el cliente o front-end y el servidor o back-end.

Se ha escogido esta metodología por la facilidad que ofrece a la hora de comunicar cliente y servidor y porque las necesidades del sistema se complementan a la perfección con este método de trabajo.

2 OBJETIVOS

Para satisfacer las necesidades del cliente, se ha marcado un objetivo principal, el de ser capaces de automatizar todo el proceso de la gestión de los paseos que organizará la organización. Para poder abordar este problema tan grande, hay que dividirlo en subobjetivos, que serán los siguientes.

- El primer objetivo es el diseño y la creación de una estructura de base de datos que sea capaz de almacenar toda la información necesaria para el correcto funcionamiento del sistema, así como una posible futura expansión de éste.
- Una vez creada la base de datos es necesario un back-end o servidor que reciba las peticiones del front-end o cliente y acceda a la base de datos, devolviendo al cliente la información solicitada.
- El siguiente paso es desarrollar la parte de front-end o cliente que utilizará el equipo técnico para coordinar la gestión de los paseos, teniendo tareas que no tendrán el resto de usuarios como añadir y eliminar paseos, vehículos o usuarios.
- Cuando esté creada la parte del equipo técnico, será necesario desarrollar la parte de front-end o cliente donde los voluntarios accederán a ver los paseos que tienen asignados y puedan aceptar o rechazar un paseo.
- A continuación, se debe distinguir el acceso a las diferentes secciones del sistema creando dos roles diferentes; el rol 'admin' (equipo técnico) y el rol 'regular' (voluntariado).
- Para poder acceder desde cualquiera de los usuarios, hay que crear un sistema de login y registro que permita, a partir de un correo electrónico y una contraseña, acceder al sistema con el usuario en cuestión y su respectivo rol.
- Con todas las funcionalidades básicas ya implementadas, se plantea un sistema capaz de realizar la comunicación automáticamente con los voluntarios (a través de email).
- En relación con el objetivo anterior, es necesario crear un proceso programado o automático, que dos días antes de que un paseo esté a punto de realizarse, envíe un mail al voluntario en cuestión a modo de recordatorio.

Según se han ido concretando los objetivos durante el desarrollo de la aplicación, los mencionados anteriormente són los que se requieren en la versión final del proyecto, dejando los siguiente como un añadido en un futuro.

- Sistema gestor de turnos que automatice en gran parte los procesos de asignación de paseos que anteriormente se hacían manualmente relacionados con qué voluntario realiza cada paseo.
- Interfaz que permita a un voluntario rellenar un formulario de inscripción que llegará al equipo técnico y este podrá validar.

El resumen de los objetivos queda reflejado en la figura Tabla 1.

Objetivos del proyecto		
ID	**Objetivo**	**Prioridad**
1	Basede datos para gestionar la información.	Alta
2	Back-end que responda a las peticiones del cliente.	Alta
3	Front-end para que el equipo técnico gestione los paseos.	Alta
4	Front-end para que los voluntarios accedan a la información	Alta
5	Creación de dos roles que diferencien al equipo técnico de los voluntarios.	Alta
6	Sistema de login y registro que permita acceder con el usuario en cuestión	Alta
7	Automatización de la comunicación con los voluntarios a partir de emails.	Media
8	Proceso programado o automático que avise al voluntario 2 días antes del paseo.	Media
9	Sistema que automatice la gestión de los turnos de paseos.	Baja
10	Interfaz que permita a los voluntarios rellenar un formulario de ingreso.	Baja

Tabla 1: Objetivos del proyecto

3 Estado del arte

El sistema a desarrollar está en un estado inexistente ya que toda la gestión se realiza de manera totalmente manual, por tanto, no hay ninguna tecnología que se pueda reutilizar.

Dentro de la gestión manual (no automatizada), se utilizan diversas herramientas ya existentes como hojas de Excel para llevar el control de una serie de datos como los turnos de paseos, Google Forms para las inscripciones de los nuevos voluntarios o Gmail para realizar la comunicación entre todas las partes que forman parte del desarrollo de estas actividades.

La información procesada con estas herramientas, debería, de alguna forma, introducirse en el nuevo sistema a desarrollar, aunque se vaya a gestionar de manera totalmente diferente. El objetivo de esto es no perder los datos anteriores a la creación del aplicativo.

También cabe la posibilidad de que el sistema a crear recurra a las herramientas mencionadas anteriormente para ejecutar ciertos procesos.

4 Metodo de desarrollo

Para abarcar la cantidad de trabajo de manera organizada y lógica, éste se dividirá en bloques de entre 2 y 1 mes a los que se le llamará sprints. Estas unidades de tiempo que dividirán el proyecto tendrán asignadas ciertas tareas que deberán realizarse dentro del sprint en cuestión. El objetivo es poder planificar y dividir el trabajo total a realizar y al final de cada sprinte ver el trabajo que se ha hecho y el que no, además de hacer un informe con todos los avances para poder mostrarlos el cliente.

En cada iteración, el sistema tiene que haber obtenido nuevas funcionalidades, lo que sirve para demostrar al cliente que se está avanzando y puede dar lugar a nuevas ideas o modificaciones que se deben tomar en el desarrollo del proyecto.

Para finalizar el sprint e iniciar el siguiente, se debe hacer una reflexión sobre si se han realizado todas las tareas correctamente y si es necesario modificar elementos del siguiente sprint antes de iniciarlo.

A esta metodología de trabajo se la conoce como desarrollo iterativo y creciente [7].

Las ventajas que ofrece esta metodología a nuestro proyecto frente a otras son:

- Los riesgos en tiempo que puede aparecer, se mitigan en gran escala, ya que en el caso de que un requisito ocupe demasiado tiempo, se detecta rápidamente en las primeras iteraciones. Este punto es muy importante ya que se han tenido que definir unos requisitos como base necesaria a entregar en la finalización del tiempo que dura el TFG.
- Al anterior beneficio se le puede añadir que esta metodología permite conocer el progreso real del proyecto desde el inicio hasta el punto actual y en base a eso determinar si la finalización es viable en la fecha prevista.
- Una ventaja que presenta teniendo en cuenta que el producto es para un cliente, es que este método de trabajo permite obtener pequeñas versiones funcionales que se pueden mostrar y el cliente puede decidir como continuar orientando el proyecto en base a lo que ha visto, ya que, en este caso, el cliente no sabía exactamente las necesidades del producto.
- Permite detectar errores, o inconsistencias en el diseño del sistema, que se pueden corregir en futuras iteraciones.
- Permite organizar las tareas en el tiempo, lo cual obliga a pensar qué tareas se realizarán antes y cuales se realizarán después para poder optimizar al máximo el desarrollo en tiempo y calidad.

Una vez decidida la metodología a seguir, se aplica al proyecto, obteniendo una serie de sprints a realizar en el tiempo establecido para finalizar el proyecto. De manera que los sprints obtenidos han durado 2 semanas cada uno y son los siguientes:

Sprints		
Nº	**Tarea**	**Inicio**
1	Captación de requisitos y elaboración de la planificación.	20/09
2	Creación de la Base de datos	04/10
3	Servidor Back-end que reciba las peticiones de la interfaz de usuario y realice las consultas a la base de datos	18/10

4	Interfaz que muestre la información de la Base de Datos	01/11
5	Sistema de registro y Login al sistema	16/11
6	Funcionalidades en la interfaz de usuario para completar la gestión de las actividades.	30/11
7	Funcionalidades en el servidor para completar la gestión de las actividades.	14/12
8	Interfaz del usuario 'regular' con accesos limitados al sistema y sistema de comunicación automático.	30/12
9	Realización del informe final y presentación del proyecto.	15/01

Tabla 2: Sprints del proyecto

En el punto final del proyecto, los sprints difieren respecto a como estaban pensados en un principio, esto se debe al cambio de necesidades del cliente según ha ido avanzando el proyecto.

5 PLANIFICACIÓN DEL PROYECTO

Aunque el trabajo ya está dividido en sprints, cada uno de estos es demasiado grande para abarcarlo directamente, por lo que se tiene que dividir en tareas más concretas. De manera que se han planificado una serie de tareas que se tendrán que realizar durante el transcurso del sprint y que tendrán como deadline el día final del sprint en cuestión.

Tareas		
Tarea	**Sprint**	**Inicio**
Reunión de toma de requisitos con la organización EnBiciSenseEdat	1	25/09
Planificación de tareas y requisitos	1	26/09
Diseño de las tablas en la base de datos para recoger toda la información necesaria.	2	05/10
Implementación del diseño realizado en el punto anterior.	2	09/10
Diseño de servidor Back-end capaz de responder las llamadas de la interfaz del equipo técnico.	3	01/11
Implementación del diseño realizado en el punto anterior	3	02/11
Reunión con la organización para mostrar el progreso del proyecto y redefinir los objetivos en base a las necesidades.	3	06/11
Diseño de la interfaz del equipo técnico para ver y modificar la base de datos.	4	19/10
Implementación del diseño realizado en el punto anterior.	4	20/10
Funcionalidad que permita al servidor gestionar el login y registro de usuarios a partir de unos parámetros de entrada.	5	18/11
Pantallas de Login y Registro en la interfaz de usuario que envíen al servidor los datos necesarios (introducidos por el usuario) para realizar un login.	5	29/11
Revisión del estilo de la interfaz del usuario para hacerlo mas intuïtivo y profesional.	6	02/12
Creación de funcionalidades en la interfaz para relacionar las tablas y crear objetos en una tabla a partir de otra.	6	06/12
Creación de funcionalidades en el back-end para complementar la interfaz mencionada en el punto anterior.	7	14/12
Otorgar diferente acceso a los usuarios según sus permisos tanto en el front-end como en el back-end	7	25/12
Interfaz completa para el usuario 'regular', que le permita aceptar o declinar paseos y ver todos los relacionados con él/ella.	8	02/01
Sistema de comunicación por email automatizado.	8	05/01
Proceso programado que envíe emails automáticamente cuando queden 2 días para realizar el paseo	8	07/01
Corrección de errores y creación de mensajes de error en la interfaz.	8	10/01
Realización del informe final y presentación del proyecto	9	15/01

Tabla 3: Tareas del proyecto relacionadas con los Sprints.

6 DESARROLLO DEL PROYECTO

En esta sección se explicarán en detalle y a nivel técnico los sprints y sus respectivas tareas, que han ido realizandose a lo largo del proyecto.

6.0 Aprendizaje

Para desarrollar el proyecto se han pensado muchos lenguajes distintos, pero al poder crearse desde cero era una buena oportunidad para aprender lenguajes de programación que puedan servir en un futuro. Por tanto, se hizo una investigación de los lenguajes que más se están utilizando y se decidió desarrollar el proyecto en Angular para el Front-End y Nodejs para el Back-End. Al tener un conocimiento casi nulo de ámbas tecnologías, fue necesario dedicar una gran cantidad de tiempo a cursos y formaciones de ambos lenguajes. Una vez acabado el proyecto, el conocimiento de estas tecnologías ha aumentado en gran medida, hasta el punto de poder desarrollar tanto webs como aplicaciones.

6.1 Captación de requisitos y elaboración de la planificación

Para la captación de requisitos del sistema, ha sido

necesario concretar una serie de reuniones con la organización que hacía de cliente. La primera reunión y más importante fue el día 25 de Setiembre del 2020. En ella se concretaron las necesidades del proyecto y su situación en ese momento.

A partir de las conclusiones extraidas de esa reunión fue posible elaborar una serie de requisitos, funcionales y no funcionales, que hicieran posible la implementación de un sistema que cumpliera todas las necesidades demandadas por la organización.

Una vez entendidas las necesidades y extraidos los requisitos, se ideó una planificación respecto al tiempo total del proyecto y se tuvieron que eliminar las funcionalidades menos importantes, ya que, en tiempo, no era posible asumir la carga de trabajo. Con lo que se realizó otra reunión para, entre las dos partes, llegar a un acuerdo en el que se implementara un sistema como base que fuera posible acabar en el tiempo disponible y, a la vez, tuviera una funcionalidad real para el cliente.

6.2 Creación de la Base de Datos

Con los requisitos entendidos, se planteó el primer problema, crear una base de datos que fuera capaz de almacenar toda la información utilizada en el proyecto. Además, era necesario hacerla de una manera general, para que, en un futuro, se pudiera ampliar el sistema sin necesidad de tocar la base de datos.

De manera que las tablas pensadas fueron las siguientes:

- **Usuarios:** Tabla en la que se guarda la información de las personas que tienen relación con las actividades de la organización (Voluntarios, equipo técnico, beneficiarios…). De éstos se guardan datos básicos como el nombre, el email o el número de teléfono.
- **Entidades:** Esta tabla guarda la información de cada entidad relacionada con las actividades que se realizan. Por ejemplo, las residencias o centros que hacen la petición de los paseos o la propia organización. Se guardan datos como el nombre, la dirección o el tipo de entidad.
- **TiposEntidad:** Esta tabla guarda los diferentes tipos de entidad que puede haber y esta relacionada con la tabla Entidades. Por el momento, según las necesidades hay 3 tipos: centro, organización y residencia. Se ha pensado en crear una tabla para facilitar el añadir entidades.
- **Vehiculos:** En esta tabla se guarda información relacionada con los vehículos que se usan para realizar los paseos. La información que se guarda puede ser el estado del vehículo, su localización o el número de paseos que se han realizado con él.
- **Paseos:** Esta es la tabla más importante de todo el sistema. En ella se guardan los paseos realizados y por realizar. Esta relacionada con la tabla usuarios, para saber quien realiza el paseo, con la tabla vehículos, para saber qué vehículo se utilizará y con la tabla entidades, para saber a que entidad pertenece el beneficiario o si es un particular. Guarda datos como la fecha del paseo, personas involucradas, vehículo que se va a utilizar…
- **Roles:** Tabla en la que se definen los roles que existen para las personas relacionadas con las actividades de la organización. De momento hay 3 roles que son coordinador, voluntario y beneficiario. Se ha pensado en crear esta tabla porque si en un futuro es necesario añadir roles, se podrá hacer fácilmente.
- **Privilegios:** Esta tabla esta pensada para gestionar los privilegios que puede tener cada usuario (ver ciertas pantallas, realizar ciertas acciones, modificar ciertos datos…). Dado que el estado demandado por el cliente solo tiene dos tipos de usuario, esta en desuso, pero al igual que la anterior esta pensada para facilitar futuras modificaciones.
- **Roles-Privilegios:** Esta tabla unirá las tablas Roles y Privilegios, de manera que para cada rol se indicará qué privilegios tendrá.

La arquitectura que se ha escogido para implementar la base de datos pensada ha sido un servidor xampp con un motor *MySQL* y la interfaz gráfica *phpMyAdmin*.

6.3 Back-End que gestione las peticiones a Base de Datos

El siguiente paso a seguir era implementar un Back-End que fuera capaz de extraer, a partir de las peticiones que enviaba un cliente, la información necesaria de la Base de Datos. Para ello se utilizó un programa llamado *Postman* [8], que simula el Front-End o aplicación cliente, con el objetivo de poder desarrollar la parte de servidor sin tener peticiones reales.

La arquitectura escogida para este Back-End ha sido el entorno de ejecución de JavaScript, *Node.js*. Junto a él, se ha utilizado el framework *Express.js*, que proporciona una serie de funcionalidades como el enrutamiento, opciones de gestión de sesión, manejo de cookies… A su vez se ha utilizado otro framework para Nodejs llamado *MySQL*, que sirve para conectar el servidor a la base de datos.

Estos frameworks se han importado en un archivo del servidor, donde también se han configurado para que hicieran su función correctamente.

El último extra que se ha añadido al servidor ha sido una herramienta llamada nodemon, que permite desarrollar más fácilmente el servidor, reiniciándolo automáticamente cuando detecta cualquier cambio en los archivos.

A lo largo del desarrollo del servidor, se ha programado de diferentes maneras. En un principio, se creó un archivo *JavaScript* con las llamadas del servidor y al ejecutarse funcionaba correctamente, pero era un método muy desordenado. En consecuencia y a raíz del tamaño que estaba adquiriendo el proyecto, se decidió trabajar con una metodología *Ruta-Controlador-Modelo*.

Esta forma de trabajo separa los archivos de nodejs en 3 partes distintas:

- **Rutas:** El archivo que detecta la url a la que se ha accedido y, dependiendo de eso, redirecciona la petición a un controlador o a otro.
- **Controlador:** El archivo que implementa toda la lógica. Envía peticiones a la base de datos, transforma la información que recibe, hace las operaciones necesarias y las envía como respuesta.
- **Modelo:** Archivo encargado de acceder a la base de datos. Su única función es hacer eso cuando recibe una petición de un controlador. Una vez recuperada la información de la base de datos, la devuelve al controlador para que la transforme.

Siguiendo este patrón de diseño. La estructura de carpetas queda de la siguiente manera:

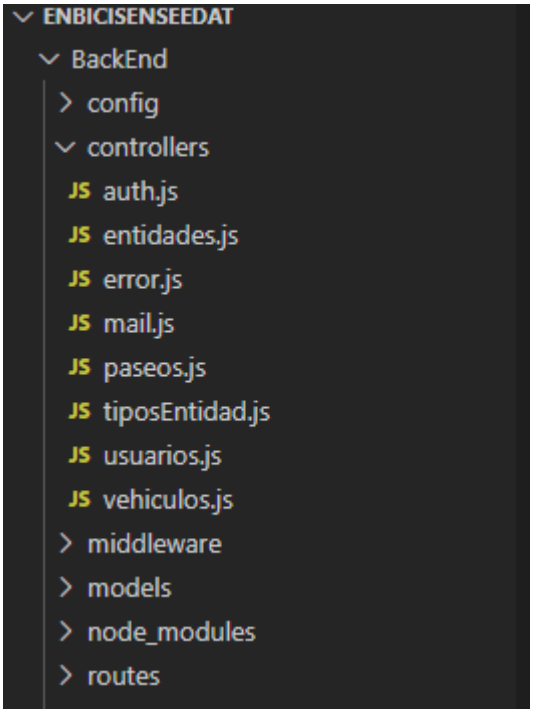

Imagen 1: Estructura de carpetas del back-end.

El Back-End tiene la función de *API-REST*. Esto significa que recibe peticiones HTTP como (GET, PUT, POST, DELETE…) y según la petición debe actuar de una forma u otra.

Las rutas están pensadas de la siguiente manera. Existe una ruta base, que en el caso del desarrollo sería *'localhost:8200'* pero una vez pasado a un entorno real cambiará según el dominio.

Detrás de esta ruta base, seguido por un '/', se encuentra el archivo de ruta al que va a acceder. Se ha pensado un archivo de ruta por cada tabla de la base de datos. Un ejemplo de como se accedería a la tabla de vehículos sería *'localhost:8200/vehículos'*.

Por último, seguido de otro '/', se encuentra la funcionalidad a la que accede. Ésta puede ser muy cambiante dependiendo de la petición que se realiza y se define en el archivo de ruta de cada tabla. Un mismo archivo de ruta tendrá varias funciones. Un ejemplo puede ser el archivo de ruta de la tabla vehículos, que contiene un GET con la ruta /:id, donde se recuperará la información de el vehículo con esa id, un DELETE con la ruta /:id, que eliminará de la base de datos el vehículo con esa id, un PUT con un objeto JSON asociado y la ruta /modificarLocalizacion, que cambiará el atributo localización de la tabla vehículos en la base de datos…

6.4 Front-End orientado al equipo técnico

La arquitectura escogida para la aplicación cliente ha sido Angular. Este framework se basa en componentes que se van uniendo a otros formando una aplicación *responsive*, que sirve como interfaz al usuario.

Tras muchas versiones de la interfaz de usuario, se ha decidido utilizar la librería *PrimeNG* [9]. Ésta ofrece una serie de componentes como botones, tablas, inputs… fácilmente personalizables y con un estilo my profesional que han permitido desarrollar todas las funcionalidades requeridas.

Una vez creada la interfaz, se implementa la lógica que hay detrás de esta, que se basa en realizar llamadas al servidor utilizando la información introducida por el usuario combinada con la información de la base de datos. Para ello, se crean peticiones HTTP que apuntan a la url donde esta el Back-End y añadiendo la ruta donde se encuentra cada petición, tal y como se ha mostrado en el punto 7.3 de este documento. Como ejemplo, si se quiere recuperar el paseo con id 24 se realizará una llamada del estilo '*localhost:8200/paseos/24*' donde el formato es '*urlServidor/archivoDeRuta/funcionDelArchivo*'.

Para cada petición hay que decidir el método HTTP que se usa (¨GET, POST, DELETE…) dependiendo de como se requiera en el servidor. Si la petición que se realiza no existe en el servidor, este devolverá un error indicando que no se ha encontrado la petición y los datos de respuesta estarán vacíos.

En el caso del equipo técnico, para cada tabla se ha creado una pantalla y para agregar un elemento a cada tabla, se ha creado otra.

Cada una de las pantallas tiene diferentes elementos relacionados con la información que muestran, pero todas ellas son muy similares y tienen la función de mostrar de manera intuitiva la información de la base de datos para que el equipo técnico la pueda gestionar.

6.5 Front-End orientado a los voluntarios

En el caso del Front-End al que acceden los voluntarios, se muestra únicamente una pantalla que muestra el historial de paseos y los próximos paseos que tienen como voluntario al usuario que accede al sistema. Además, aparece una opción que permite que estos acepten o rechacen una solicitud para ser voluntario de cada paseo. Esta decisión llega a la sección de paseos del equipo técnico, de manera que tienen monitorizada la asistencia a los paseos.

6.6 Elementos comunes en el Front-End para todos

los roles

Independientemente del rol que tenga el usuario que accede al sistema, el front-end tendrá los siguientes elementos en común:

- Una pantalla para acceder al sistema a partir de un Login.
- Una pantalla para registrarse en el sistema.
- Una pantalla para cambiar la contraseña de la cuenta del usuario.
- Un botón para salir del sistema haciendo Logout.
- Un menú lateral para navegar por las pantallas que se le muestran al usuario en cuestión.
- Un panel superior que muestra en que nivel de jerarquía (dentro del sistema) se encuentra la pantalla que se le está mostrando al usuario.

6.7 Sistema de Registro y Login

Para obtener seguridad y una distinción de roles se ha creado un sistema que permite, a partir de un usuario y una contraseña, hacer un login y un registro en el sistema. Si los datos son correctos, el sistema permite acceder a las pantallas en las que el rol del usuario tiene permisos. Si son incorrectos, el sistema muestra un mensaje de error y no permite que el usuario haga login.

Para lograr esto, en el back-end se crea un archivo de ruta *auth* que implementa 3 peticiones.

- **Registro:** Recibe un nombre, email y contraseña. Comprueba que el email no exista en el sistema y si no existe, encripta la contraseña con el módulo *bcrypt* y guarda la información en la base de datos.
- **Login:** Recibe un email y una contraseña. Comprueba si el email existe y para ese email compara la contraseña encriptada con la introducida por el usuario. Si es correcta, permite al usuario iniciar sesión y le otorga un token utilizando el módulo *JWToken* de Nodejs.
- **Cambiar Contraseña:** Recibe la anterior contraseña y una nueva. Si la anterior contraseña coincide con la que hay en la base de datos, modifica la contraseña antigua por la nueva.

Además, en la parte del front-end se ha creado una pantalla para cada una de las 3 peticiones que recibe el back-end. Desde estas pantallas, el usuario puede introducir los datos necesarios y activar estas funcionalidades.

6.8 Sistema de comunicación automatizada

Uno de los requisitos demandados por el cliente era un sistema que enviara notificaciones push a los usuarios. Después de analizar las necesidades, se ha decidido implementar un sistema que enviara mensajes a través de correo electrónico para evitar que el equipo técnico tuviera que hacerlo manualmente y así cumplir uno de los principales objetivos que es ahorrar una gran cantidad de costes en tiempo.

Para ello se ha utilizado una librería de node.js que se llama *nodemailer*, la cual permite configurar un servidor que envía (desde una cuenta de Gmail introducida), mensajes automáticos a los usuarios en cuestión. Se ha decidido que se envíen correos en los siguientes casos:

- Cuando el usuario se crea una cuenta.
- Cuando al usuario se le asigna un paseo.
- Cuando quedan 2 días para realizar el paseo, como recordatorio.

6.9 Proceso programado para enviar mensajes

Para poder cumplir el punto final del anterior apartado, es necesario crear un proceso programado o automático, que cada día analice los siguientes paseos y envíe, para los usuarios responsables del paseo, un mensaje recordatorio. Es ideal concretar la hora conveniente para enviar el recordatorio de manera que no pueda molestar a ningún usuario.

Este proceso es posible gracias al modulo node-cron, el cual, a partir de unos parámetros de configuración, procesa la petición solicitada en un determinado momento de tiempo en el futuro.

7 Test

Para comprobar que el sistema funciona correctamente se han realizado dos juegos de pruebas.

El primero ha consistido en ir componente a componente probando todos los casos en los que se puede usar. Como ejemplo, en un formulario se introducen campos que no deberían ser admitidos y se intenta continuar. El sistema no debe permitir que el flujo de ejecución avance. Además, debe indicar al usuario que ha cometido un error y dónde lo ha cometido. En un campo "teléfono" no están permitidos los caracteres que no sean números.

El segundo ha consistido en reproducir los casos de uso de la aplicación uno a uno y validar que no aparece ningún error en ninguno de los flujos. Además, se han testeado los casos en que el sistema debería devolver un error, asegurando que se devuelve el error correctamente.

Gracias a estas pruebas, se puede entergar una versión estable al usuario sin que se produzcan problemas.

8 Resultados

Los resultados obtenidos de todo el trabajo descrito en este informe son los siguientes:

- Una base de datos en MySQL, capaz de recoger toda la información necesaria para que el proyecto deseado por el cliente funcione correctamente y se pueda expandir en futuras versiones.
- Un servidor back-end en Node.js, capaz de hacer peticiones a todo el contenido de la base de datos y de devolverlo al cliente Front-End. Además, es capaz de procesar toda la información y transformarla en los datos necesarios para que el sistema funcione como el cliente espera.
- Un cliente front-end en Angular, capaz de realizar las peticiones necesarias al servidor Back-End y mostrar al usuario la información obtenida. Además, otorga al cliente una interfaz de usuario intuitiva para poder interactuar con el

sistema.

- Un sistema automàtico de envío de emails para poder comunicarse con el equipo de voluntarios y personas relacionadas con las actividades de la organización.

El resultado de la interfaz gráfica que va a utilizar el equipo técnico se puede apreciar en la Imagen 2.

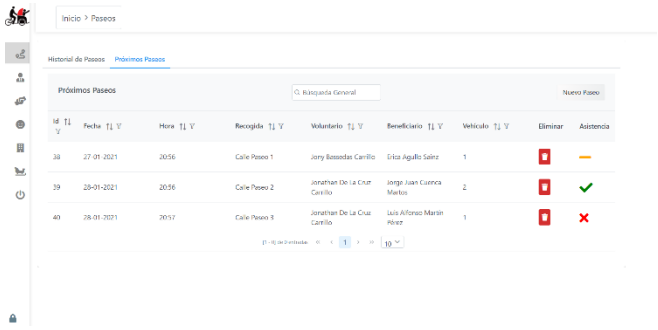

Imagen 2: Ejemplo de la interfaz gráfica desarrollada.

El resultado final del proyecto, es una versión inicial del sistema que quiere desarrollar la organización. A partir del plazo de tiempo del que se disponía para realizarlo, se acordaron unas necesidades básicas, las cuales debían entregarse como base para escalar el proyecto en un futuro.

Dichas necesidades son las siguientes:

- Automatización de la gestión de paseos.
- Notificaciones a las personas relacionadas con la organización.
- Almacenamiento de los datos relacionados con la organización.

Como se ha mencionado anteriormente, en el sistema actual estas necesidades se cumplen correctamente y, además, a partir de varias iteraciones en el proyecto, se gestionan de una manera cómoda para la organización según su estilo de trabajo.

En relación a los aspectos técnicos del sistema, el desarrollo del Front-End en Angular, lo convierte en una aplicación a la cual se puede acceder tanto desde ordenadores como dispositivos móviles o tablets. Lo cual facilita las necesidades de la organización ya que, si no fuera posible esta ejecución, no se podría consultar el sistema mientras se realizan las actividades en cuestión.

El desarrollo en Node.js del Back-End, permite acceder a un gran número de módulos ya desarrollados, que implementan funcionalidades que, si se tuvieran que implementar manualmente, supondrían un gran coste en tiempo que sería imposible de asumir según el tiempo estimado para el proyecto.

Tanto el Front-End, el Back-End y la base de datos, se han desplegado en un servidor que las soporta y ejecuta. Gracias a eso, es posible acceder al sistema desde cualquier lugar y dispositivo.

9 Conclusiones

Valoro muy positivamente la experiencia de desarrollar un sistema para un cliente real, ya que me ha hecho entender como funciona el proceso de desarrollo de un proyecto, desde el principio (la toma de requisitos y comprensión de las necesidades del cliente), hasta el final (el despliegue del proyecto y la entrega al cliente).

Además de la comprensión de todo el flujo de desarrollo, me ha obligado a aprender los lenguajes de Angular y Nodejs de manera profesional, ya que, al ser para un cliente real, la presión de entregar un proyecto profesional era mayor.

Otro punto interesante, ha sido el ver con mis propios ojos cómo las necesidades del proyecto iban cambiando según se iba desarrollando el sistema, cómo no podía obtener una respuesta inmediata del cliente y cómo tenía que organizarme para adaptarme a los constantes cambios.

10 Agradecimientos

Agradezco a mi tutor Marc Talló Sendra, la ayuda que me ha prestado, orientándome sobre como enfocar el proyecto y colaborando en la comprensión de las necesidades del cliente.

También quiero agradecer a la organización EnBiciSenseEdat su confianza en mí, como alumno recién graduado, para desarrollar una herramienta tan importante para ellos. Además, la simpatía que han mostrado a la hora de reunirnos para decidir el avance del proyecto.

11 Bibliografía

[1]"EN BICI SENSE EDAT - Cycling Without Age", *Cycling Without Age*, 2021. [Online]. Available: https://cyclingwithoutage.org/catalonia/. [Accessed: 24- Sep- 2020].
[2]"Angular", *Angular.io*, 2021. [Online]. Available: https://angular.io/. [Accessed: 28- Sep- 2020].
[3]"Single Page Application: Un viaje a las SPA a través de Angular y Javascript", *Medium*, 2021. [Online]. Available: https://medium.com/@davidjguru/single-page-application-un-viaje-a-las-spa-a-trav%C3%A9s-de-angular-y-javascript-337a2d18532. [Accessed: 28- Sep- 2020].
[4]"Node.js", *Node.js*, 2021. [Online]. Available: https://nodejs.org/es/. [Accessed: 30- Sep- 2020].
[5]"MySQL", *Mysql.com*, 2021. [Online]. Available: https://www.mysql.com/. [Accessed: 26- Sep- 2020].
[6]"¿Qué es REST? Conoce su potencia", *OpenWebinars.net*, 2021. [Online]. Available: https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/. [Accessed: 08- Oct- 2020].
[7]"Desarrollo iterativo e incremental", *Proyectos Ágiles*, 2021. [Online]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/. [Accessed: 27- Sep- 2020].
[8]*Postman.com*, 2021. [Online]. Available: https://www.postman.com/. [Accessed: 18- Oct- 2020].
[9]"PrimeNG | Angular UI Component Library", *Primefaces.org*, 2021. [Online]. Available: https://www.primefaces.org/primeng/. [Accessed: 17- Nov- 2020].

- *E-mail de contacte: jonathan.delacruz@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Marc Talló Sendra (Ciències de la Computació)*
- *Curs 2020/21*