Proyecto Angular Taller Dils

Realizado por Miguel García
2º ASIR

1. Explicación del proyecto

En este proyecto se pretende crear una aplicación para un taller de mecánica donde se recogerán los siguientes datos:

Los vehículos que se reparan y las reparaciones.

Cada vehículo tiene sus reparaciones y estas estarán gestionadas por un tratamiento CRUD por los empleados del taller.

Los empleados podrán listar, borrar, crear y listar los vehiculos y las reparaciones ya sea globalmente o uno en concreto.

También se tiene un login para proteger la api de internet ya que si entran y nos eliminan o ponen datos erróneos pues estaríamos ante una vulnerabilidad.

2. Clasess y subclases

Mis clases son las siguientes:

Clase Vehiculo:

```
export  class Vehiculo {
    private _matricula: string;
    private _marca: string;
    private _color: string;
    private _tipoVehiculo: string;
```

Esta clase se usará para los vehiculos del taller, tiene dos herencias que son dos tipos de coches, deportivos y todoTerrenos:

Subclase Deportivo:

```
export class Deportivo extends Vehiculo {
    private _potenciaM: string;

    constructor(matricula: string, marca: string, color: string, precio: string,

        super(matricula, marca, color, tipoVehiculo);
        this._potenciaM = pMotor;
    }
```

Donde la potencia del motor es el campo de esta.

Subclase todoTerreno:

```
import { Vehiculo } from "./vehiculo";

export class todoTerreno extends Vehiculo {
    private _traccion: string;

    constructor(matricula: string, marca: string, color: string, precio: string,

        super(matricula, marca, color, tipoVehiculo);
        this._traccion = traccion;
    }
```

Donde la tracción será su campo identificativo.

3. Rutas y sus metodos

Las principales rutas que usaré son las de listarVehiculos:

```
this._router.get("/verVehiculos", this.listarVehiculos);
```

Con ella cargamos en la pantalla inicial de la aplicación los vehículos de la base de datos.

Lo hacemos de la siguiente manera:

```
private listarVehiculos = async (req: Request, res: Response) => {
  await db
    .conectarBD()
    .then(async (mensaje) => {
      const query = await Vehiculos.find({});
      res.json(query);
    })
    .catch((mensaje) => {
      res.send(mensaje);
    });
  await db.desconectarBD();
};
```

La siguiente ruta nos sirve para añadir vehiculos:

```
this._router.post("/addVehiculo", this.agregarVehiculo);
```

Esta ruta será la que nos añadirá los documentos a la base de datos, su estructura:

```
private agregarVehiculo = async (req: Request, res: Response) => {
  const { matricula, marca, color, tipoVehiculo } = req.body;
  await db.conectarBD();
  const dSchema = {
    _matricula: matricula,
    _marca: marca,
    _color: color,
    _tipoVehiculo: tipoVehiculo,
  };
  const oSchema = new Vehiculos(dSchema);
  await oSchema
    .save()
    .then((doc: any) => res.send(doc))
    .catch((err: any) => res.send("Error: " + err));
  await db.desconectarBD();
};
```

Ahora veremos la ruta de modificar los documentos de Vehiculos:

```
this._router.put("/updateVehiculo/:matricula", this.modificarVehiculo);
```

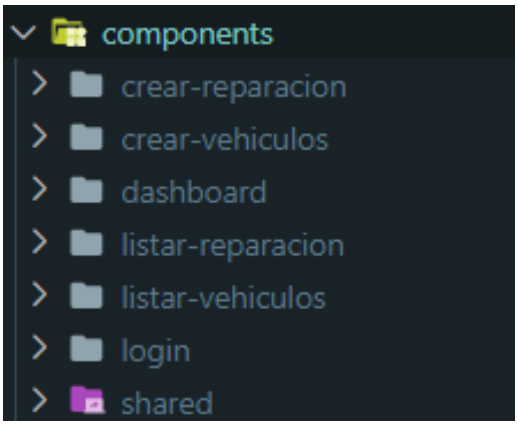
Su estructura es la siguiente:

```
private modificarVehiculo = async (req: Request, res: Response) => {
  await db.conectarBD();
  const mat = req.params.matricula;
  const {  matricula,marca, color, tipoVehiculo } = req.body;
  await Vehiculos.findOneAndUpdate(
    { _matricula: mat },
    { _matricula: matricula,_marca: marca, _color: color,  _tipoVehiculo: ti
    { new: true }
  )
    .then((doc: any) => res.send(doc))
    .catch((err: any) => res.send("Error: " + err));

  await db.desconectarBD();
};
```

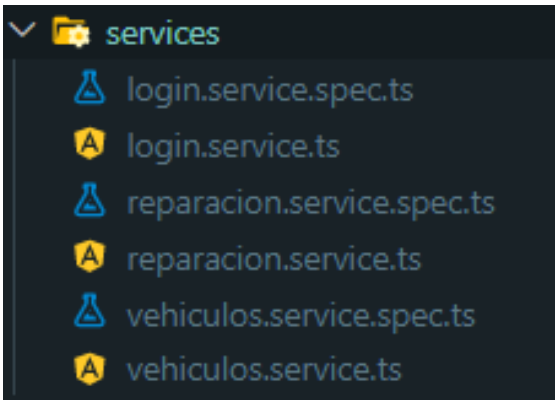
4. Componentes y servicios en angular

Los componentes usados en angular son:



- Crear reparacion: Formulario para crear una reparaciones
- Crear vehiculos: Formulario para crear un vehiculo
- dashboard: Componente menu principales
- Listar reparacion: Nos lista en una tabla los datos recogidos de la base de datos de reparaciones
- Listar vehiculos: Nos lista en una tabla los datos recogidos de la base de datos de vehiculos
- Login: Componente para logarte en la aplicación

Los servicios utilizados son:

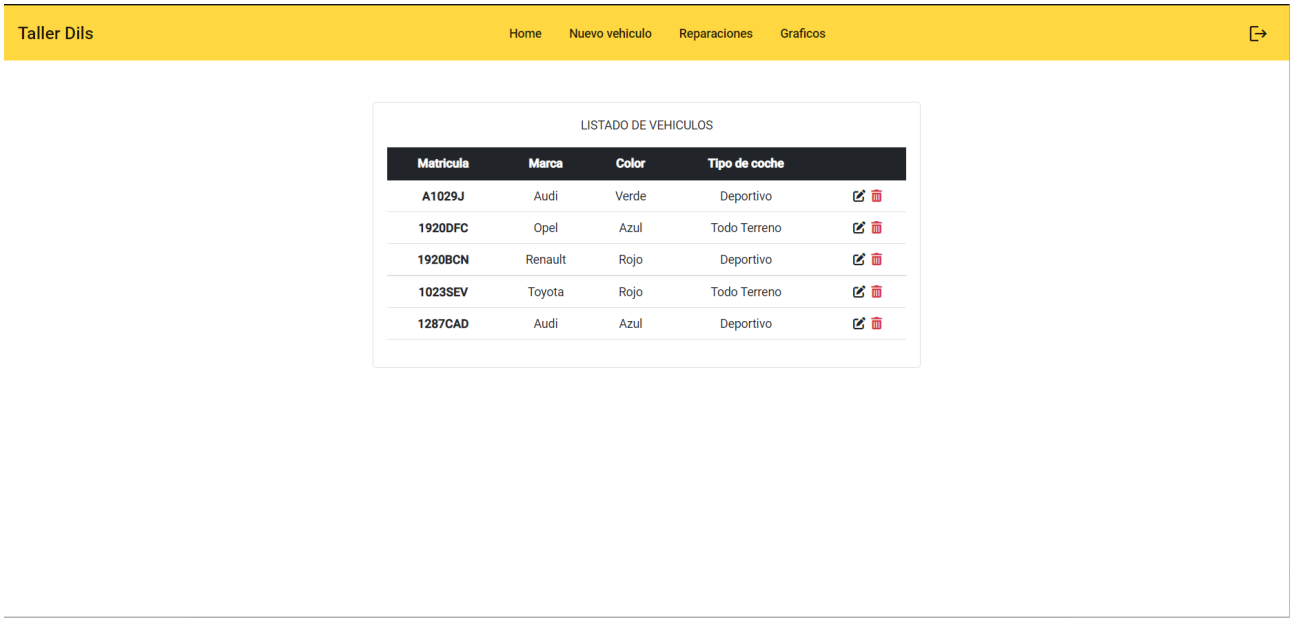


- reparacion.service: Servicio que se comunica con la rest-api para todo lo relacionado con reparaciones.

- vehiculos.service: Servicio que se comunica con la rest-api para todo lo relacionado con vehiculos.

Imagenes de la aplicación:

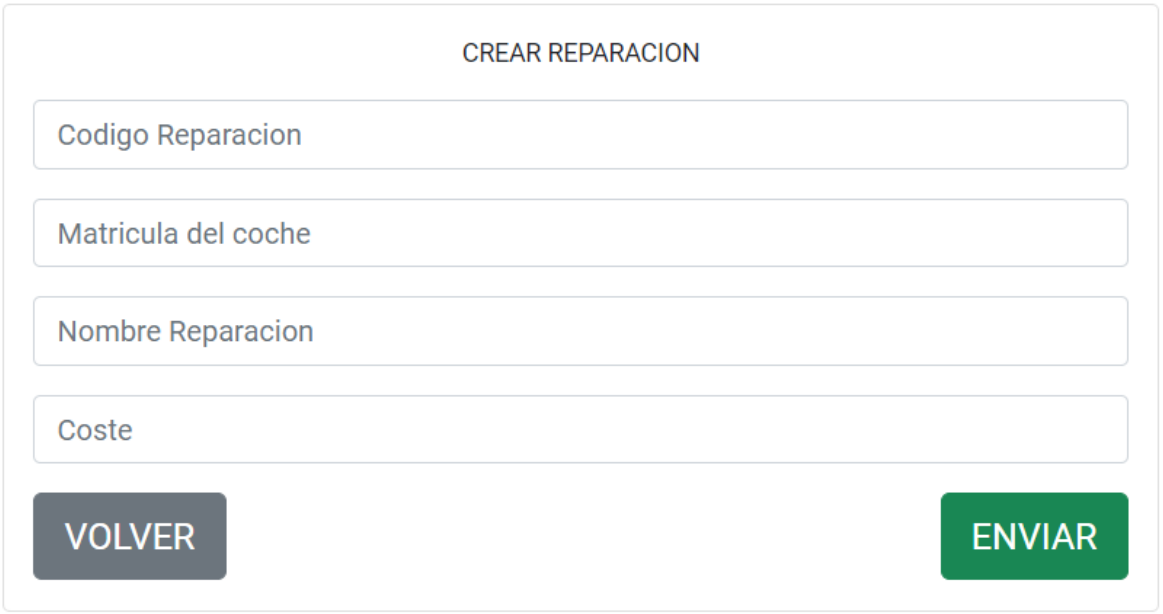
PANTALLA INICIAL



CREACION DE VEHICULOS

LISTADO DE REPARACIONES

CREAR REPARACIONES



ENLACES:

https://mechanic-frontend-v1.herokuapp.com

https://github.com/mgarciaholgado/PROYECTO-FRONTEND

API-REST

https://mechanic-api-rest.herokuapp.com

https://github.com/mgarciaholgado/PROYECTO_BACKEND