



Proyecto Angular
VapeHub

Índice

1. Introducción	2
2. Clases	3
2.1. Persona	3
2.1.1. Persona	3
2.1.2. Cliente	4
2.1.3. Empleado	4
2.2. Producto	5
2.2.1. Producto	5
2.2.2. Líquido	5
2.2.3. Dispositivo	6
3. Compra	7
4. Tipos	8
4.1. Persona	8
4.2. Producto	8
4.3. Compra	9
5. Rutas	10
6. Aplicación	17
6.1. Servicios	17
6.1.1. Cliente	17
6.1.2. Empleado	17
6.1.3. Líquido	18
6.1.4. Dispositivo	18
6.1.5. Compra	18
6.1.6. Menú dinámico	18
6.2. Explicación menú	19
6.3. Routing	20
6.3.1. app-routing	20
6.3.2. dashboard-routing	20
6.4. Módulos	21
6.4.1. DashboardModule	21
6.4.2. SharedModule	21
6.5. Componentes	22
6.5.1. ClientesComponent	22
6.5.2. EmpleadosComponent	23
6.5.3. CrearClienteComponent	24
6.5.4. CrearEmpleadoComponent	25
6.5.5. LiquidosComponent	26
6.5.6. DispositivosComponent	26
6.5.7. CompraComponent	27
6.5.8. ReportesComponent	28
6.5.9. LoginComponent	29

1. Introducción

Proyecto de una aplicación que trata datos mostrados de forma visual gracias a Angular usando la librería de componentes de Angular Material. Es una aplicación destinada a un empleado el cuál es administrador o es encargado de un punto de venta.

Esta alojada en Heroku desplegada por el repositorio de github. Usa una API la cuál conecta con una base de datos alojada en Mongo Atlas, desplegada en Heroku al igual que la APP.

Primero veremos las clases y modelos usados, y ya luego pasaremos a la aplicación.



2. Clases

2.1. Persona

2.1.1. Persona

```

1   export class Persona {
2   protected _id: string;
3   protected _nombre: string;
4   protected _direccion: { calle: string; numero: number };
5   protected _telefono: number;
6   protected _email: string;
7   constructor(
8     id: string,
9     nombre: string,
10    direccion: { calle: string; numero: number },
11    telefono: number,
12    email: string,
13  ) {
14    this._id = id;
15    this._nombre = nombre;
16    this._direccion = direccion;
17    this._telefono = telefono;
18    this._email = email;
19  }
20  get id() {
21    return this._id;
22  }
23  get nombre() {
24    return this._nombre;
25  }
26  get direccion() {
27    return this._direccion;
28  }
29  get telefono() {
30    return this._telefono;
31  }
32  get email() {
33    return this._email;
34  }
35
36  salario(){
37    let salario: number;
38    return Math.round(salario)
39  }
40
41  todo() {
42    return `ID: ${this._id},
43    Nombre: ${this._nombre},
44    Direccion: ${this._direccion},
45    Telefono: ${this._telefono},
46    Email : ${this._email}`;
47  }
48 }

```

2.1.2. Cliente

```

1  import { Persona } from "../persona";
2  export class Cliente extends Persona {
3      protected _socio: boolean;
4      constructor(
5          id: string,
6          nombre: string,
7          direccion: { calle: string; numero: number },
8          telefono: number,
9          email: string,
10         socio: boolean,
11     ) {
12         super(id, nombre, direccion, telefono, email);
13         this._socio = socio;
14     }
15     get socio() {
16         return this._socio;
17     }
18 }

```

2.1.3. Empleado

```

1  import { Persona } from "../persona";
2  export class Empleado extends Persona {
3      protected _ventas: number;
4      protected _horas: number;
5      constructor(
6          id: string,
7          nombre: string,
8          direccion: { calle: string; numero: number },
9          telefono: number,
10         email: string,
11         ventas: number,
12         horas: number,
13     ) {
14         super(id, nombre, direccion, telefono, email);
15         this._ventas = ventas;
16         this._horas = horas;
17     }
18     get ventas() {
19         return this._ventas;
20     }
21     salario(): number {
22         let salario: number;
23         let base: number = this._horas * 8;
24         if (this._ventas > 5) { salario = base * 1.02 }
25         else if (this._ventas > 10) {
26             salario = base * 1.03
27         } else if (this._ventas > 20) {
28             salario = base * 1.05
29         } else {
30             salario = base
31         }
32         return Math.round(salario)
33     }
34 }

```

2.2. Producto

2.2.1. Producto

```

1  export class Producto {
2      protected _id: string;
3      protected _nombreProd: string;
4      protected _marca: string;
5      protected _coste: number;
6
7      constructor(id: string, nombreProd: string, marca: string, coste: number) {
8          this._id = id;
9          this._nombreProd = nombreProd;
10         this._marca = marca;
11         this._coste = coste;
12     }
13
14     get id() {
15         return this._id;
16     }
17     get nombreProd() {
18         return this._nombreProd;
19     }
20     get marca() {
21         return this._marca;
22     }
23     get coste() {
24         return this._coste;
25     }
26
27     todoProd() {
28         return `ID: ${this._id},
29         Nombre de producto: ${this._nombreProd},
30         Marca: ${this._marca}
31         coste ${this._coste}`;
32     }
33 }

```

2.2.2. Líquido

```

1  export class Liquido extends Producto {
2
3      protected _sabor: string;
4      protected _nicotina: number;
5
6      constructor(
7          id: string,
8          nombreProd: string,
9          marca: string,
10         sabor: string,
11         nicotina: number,
12         coste: number
13     ) {
14         super(id, nombreProd, marca, coste);
15
16         this._sabor = sabor;
17         this._nicotina = nicotina;
18     }
19     get sabor() {
20         return this._sabor;
21     }
22     get nicotina() {
23         return this._nicotina;
24     }
25 }

```



2.2.3. Dispositivo

```
1  export class Dispositivo extends Producto {
2      protected _potencia: number;
3      protected _bateria: number;
4
5      constructor(
6          id: string,
7          nombreProd: string,
8          marca: string,
9          potencia: number,
10         bateria: number,
11         coste: number
12     ) {
13         super(id, nombreProd, marca, coste);
14
15         this._potencia = potencia;
16         this._bateria = bateria;
17     }
18     get potencia() {
19         return this._potencia;
20     }
21     get bateria() {
22         return this._bateria;
23     }
24 }
```

3. Compra

```

1  export class Compra {
2      protected _id: string;
3      protected _idCliente: string;
4      protected _coste: number;
5      protected _idProducto: string;
6
7      constructor(id: string, idCliente: string, coste: number, idProducto: string) {
8          this._id = id;
9          this._idCliente = idCliente;
10         this._coste = coste;
11         this._idProducto = idProducto;
12     }
13     get getId() {
14         return this._id;
15     }
16     get getCliente() {
17         return this._idCliente;
18     }
19     get getProductos() {
20         return this._idProducto;
21     }
22
23     todoCompra() {
24         return `ID: ${this._id},
25             : nombreCliente: ${this._idCliente},
26             coste: ${this._coste},
27             productos:  ${this._idProducto}`;
28     }
29 }

```


4. Tipos

4.1. Persona

```

1 export type tCliente = {
2   id: string | null;
3   nombre: string | null;
4   calle: string;
5   numero: number;
6   telefono: number | null;
7   email: string | null;
8   socio: Boolean | null;
9 };
10
11 export type tEmpleado = {
12   id: string | null;
13   nombre: string | null;
14   calle: string;
15   numero: number;
16   telefono: number | null;
17   email: string | null;
18   ventas: number | null;
19   horas: number | null;
20 };
21
22 export type tEmpleado2 = {
23   _id: string;
24   _nombre: string;
25   _direccion: {
26     calle: string;
27     numero: number;
28   };
29   _telefono: number;
30   _email: string;
31   _ventas: number;
32   _horas: number;
33 };
34
35 export type tSalario = {
36   _id: string | null;
37   _nombre: string | null;
38   _salario: number | null;
39 };

```

4.2. Producto

```

1 export type tDispositivo = {
2   id: string | null;
3   nombreProd: string | null;
4   marca: string | null;
5   potencia: number | null;
6   bateria: number | null;
7   coste: number | null;
8 };
9 export type tLiquido = {
10  id: string | null;
11  nombreProd: string | null;
12  marca: string | null;
13  sabor: string | null;
14  nicotina: number | null;
15  coste: number | null;
16 };

```



4.3. Compra

```
1 export type tCompra = {  
2   id: string | null;  
3   idCliente: string | null;  
4   idProducto: string | null;  
5   coste: number | null;  
6   };
```

5. Rutas

Funciones usadas en el routing:

■ Listar:

```

1  // Listar empleados
2  private getEmpleados = async (req: Request, res: Response) => {
3      await db.conectarBD()
4      .then(async (mensaje) => {
5          const valor = req.params.id
6          console.log(mensaje)
7          const query = await EmpleadoDB.find();
8          res.json(query)
9      })
10     .catch((mensaje) => {
11         res.send(mensaje)
12     })
13     db.desconectarBD()
14 }
15
16 // Listar clientes
17 private getClientes = async (req: Request, res: Response) => {
18     await db.conectarBD()
19     .then(async (mensaje) => {
20         const valor = req.params.id
21         console.log(mensaje)
22         const query = await ClienteDB.find();
23         res.json(query)
24     })
25     .catch((mensaje) => {
26         res.send(mensaje)
27     })
28     db.desconectarBD()
29 }
30
31 // Listar dispositivos
32 private getDispositivos = async (req: Request, res: Response) => {
33     await db.conectarBD()
34     .then(async (mensaje) => {
35         const valor = req.params.id
36         console.log(mensaje)
37         const query = await DispositivoDB.find();
38         res.json(query)
39     })
40     .catch((mensaje) => {
41         res.send(mensaje)
42     })
43     db.desconectarBD()
44 }
45
46 // Listar liquidos
47 private getLiquidos = async (req: Request, res: Response) => {
48     await db.conectarBD()
49     .then(async (mensaje) => {
50         const valor = req.params.id
51         console.log(mensaje)
52         const query = await LiquidoDB.find();
53         res.json(query)
54     })
55     .catch((mensaje) => {
56         res.send(mensaje)
57     })
58     db.desconectarBD()
59 }

```

■ Añadir:

```

1  // Add empleado
2  private addEmpleado = async (req: Request, res: Response) => {
3      const { id, nombre, calle, numero, telefono, email, ventas, horas } = req.body
4      await db.conectarBD()
5      const dSchema = {
6          _id: id,
7          _nombre: nombre,
8          _direccion: { calle: calle, numero: numero },
9          _telefono: telefono,
10         _email: email,
11         _ventas: ventas,
12         _horas: horas
13     }
14     const oSchema = new EmpleadoDB(dSchema)
15     await oSchema.save()
16     .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
17     .catch((err: any) => res.send('Error: ' + err))
18
19     db.desconectarBD()
20 }
21
22 // Add cliente:
23 private addCliente = async (req: Request, res: Response) => {
24     const { id, nombre, calle, numero, telefono, email, socio } = req.body
25     await db.conectarBD()
26     const dSchema = {
27         _id: id,
28         _nombre: nombre,
29         _direccion: { calle: calle, numero: numero },
30         _telefono: telefono,
31         _email: email,
32         _socio: socio
33     }
34     const oSchema = new ClienteDB(dSchema)
35     await oSchema.save()
36     .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
37     .catch((err: any) => res.send('Error: ' + err))
38
39     db.desconectarBD()
40 }
41
42 // Add compra
43 private addCompra = async (req: Request, res: Response) => {
44     const { id, idCliente, idProducto, coste } = req.body
45     let fecha: Date = new Date()
46     await db.conectarBD()
47     const dSchema = {
48         _id: id,
49         _idCliente: idCliente,
50         _idProducto: idProducto,
51         _coste: coste,
52         _fecha: fecha
53     }
54     const oSchema = new CompraDB(dSchema)
55     await oSchema.save()
56     .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
57     .catch((err: any) => res.send('Error: ' + err))
58
59     db.desconectarBD()
60 }

```

■ Actualizar:

```

1  // Actualizar empleado
2  private updateEmpleado = async (req: Request, res: Response) => {
3      await db.conectarBD()
4      const id = req.params.id
5      const { nombre, calle, numero, telefono, email, ventas, horas } = req.body
6      await EmpleadoDB.findOneAndUpdate(
7          { _id: id },
8          {
9              _id: id,
10             _nombre: nombre,
11             _direccion: { calle: calle, numero: numero },
12             _telefono: telefono,
13             _email: email,
14             _ventas: ventas,
15             _horas: horas
16         },
17         {
18             new: true,
19             runValidators: true
20         }
21     )
22     .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
23     .catch((err: any) => res.send('Error: ' + err))
24
25     await db.desconectarBD()
26 }
27
28 // Actualizar cliente
29 private updateCliente = async (req: Request, res: Response) => {
30     await db.conectarBD()
31     const id = req.params.id
32     const { nombre, calle, numero, telefono, email, socio } = req.body
33     await ClienteDB.findOneAndUpdate(
34         { _id: id },
35         {
36             _id: id,
37             _nombre: nombre,
38             _direccion: { calle: calle, numero: numero },
39             _telefono: telefono,
40             _email: email,
41             _socio: socio,
42         },
43         {
44             new: true,
45             runValidators: true
46         }
47     )
48     .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
49     .catch((err: any) => res.send('Error: ' + err))
50
51     await db.desconectarBD()
52 }

```

■ Eliminar:

```

1  // Eliminar empleado
2  private delEmpleado = async (req: Request, res: Response) => {
3      await db.conectarBD()
4
5      const id = req.params.id
6      await EmpleadoDB.findOneAndDelete({ _id: id })
7      .then((doc: any) => res.send('Eliminado correctamente.'))
8      .catch((err: any) => res.send('Error: ' + err))
9
10     await db.desconectarBD()
11 }
12
13 // Eliminar cliente
14 private delCliente = async (req: Request, res: Response) => {
15     await db.conectarBD()
16
17     const id = req.params.id
18     await ClienteDB.findOneAndDelete({ _id: id })
19     .then((doc: any) => res.send('Eliminado correctamente.'))
20     .catch((err: any) => res.send('Error: ' + err))
21
22     await db.desconectarBD()
23 }

```

■ Buscar ID:

```

1  // Buscar empleado
2  private getEmpleadoId = async (req: Request, res: Response) => {
3  await db.conectarBD()
4  .then(async (mensaje) => {
5    const id = req.params.id
6    console.log(mensaje)
7    const query = await EmpleadoDB.findOne({ _id: id });
8    res.json(query)
9  })
10 .catch((mensaje) => {
11   res.send(mensaje)
12 })
13 await db.desconectarBD()
14 }
15
16 // Buscar cliente
17 private getClienteId = async (req: Request, res: Response) => {
18 await db.conectarBD()
19 .then(async (mensaje) => {
20   const id = req.params.id
21   console.log(mensaje)
22   const query = await ClienteDB.findOne({ _id: id });
23   res.json(query)
24 })
25 .catch((mensaje) => {
26   res.send(mensaje)
27 })
28 await db.desconectarBD()
29 }
30
31 // Buscar dispositivo
32 private getDispositivoId = async (req: Request, res: Response) => {
33 await db.conectarBD()
34 .then(async (mensaje) => {
35   const id = req.params.id
36   console.log(mensaje)
37   const query = await DispositivoDB.findOne({ _id: id });
38   res.json(query)
39 })
40 .catch((mensaje) => {
41   res.send(mensaje)
42 })
43 await db.desconectarBD()
44 }
45
46 // Buscar liquido
47 private getLiquidoId = async (req: Request, res: Response) => {
48 await db.conectarBD()
49 .then(async (mensaje) => {
50   const id = req.params.id
51   console.log(mensaje)
52   const query = await LiquidoDB.findOne({ _id: id });
53   res.json(query)
54 })
55 .catch((mensaje) => {
56   res.send(mensaje)
57 })
58 await db.desconectarBD()
59 }

```

■ Calcular salario:

```

1  // Calcular salario
2  private getSalario = async (req: Request, res: Response) => {
3      let tmpEmpleado: Empleado
4      let dEmpleado: tEmpleado2
5      let arraySalario: Array<tSalario> = []
6      console.log('hola');
7
8      await db.conectarBD()
9          .then(async (mensaje) => {
10         console.log(mensaje)
11         const query = await EmpleadoDB.find({});
12         for (dEmpleado of query) {
13             tmpEmpleado = new Empleado(
14                 dEmpleado._id,
15                 dEmpleado._nombre,
16                 dEmpleado._direccion,
17                 dEmpleado._telefono,
18                 dEmpleado._email,
19                 dEmpleado._ventas,
20                 dEmpleado._horas)
21
22             let salario = tmpEmpleado.salario()
23
24             let oSalario: tSalario = {
25                 _id: null,
26                 _nombre: null,
27                 _salario: null
28             }
29             oSalario._id = tmpEmpleado.id
30             oSalario._nombre = tmpEmpleado.nombre
31             oSalario._salario = salario
32             arraySalario.push(oSalario)
33         }
34
35         res.json(arraySalario)
36     })
37     .catch((mensaje) => {
38         res.send(mensaje)
39     })
40     await db.desconectarBD()
41 }

```


Rutas usadas:

```

1  // RUTAS
2  misRutas() {
3    // Listar
4    this._router.get('/empleados/salario', this.getSalario)
5    this._router.get('/empleados', this.getEmpleados)
6    this._router.get('/clientes', this.getClientes)
7    this._router.get('/empleados/:id', this.getEmpleadoId)
8    this._router.get('/clientes/:id', this.getClientId)
9    this._router.get('/dispositivos', this.getDispositivos)
10   this._router.get('/liquidados', this.getLiquidados)
11   this._router.get('/dispositivos/:id', this.getDispositivoId)
12   this._router.get('/liquidados/:id', this.getLiquidadoId)
13   // Crear
14   this._router.post('/empleados/addEmpleado', this.addEmpleado)
15   this._router.post('/clientes/addCliente', this.addCliente)
16   this._router.post('/addCompra', this.addCompra)
17   // Actualizar
18   this._router.put('/empleados/update/:id', this.updateEmpleado)
19   this._router.put('/clientes/update/:id', this.updateCliente)
20   // Eliminar
21   this._router.delete('/empleados/delete/:id', this.delEmpleado)
22   this._router.delete('/clientes/delete/:id', this.delCliente)
23 }
24 }
```

6. Aplicación

6.1. Servicios

6.1.1. Cliente

```

1  export class ClienteService {
2    baseUrl = 'https://api-vapehub.herokuapp.com/clientes'
3
4    constructor(private http: HttpClient) { }
5
6    getCliente(): Observable<any> {
7      return this.http.get(this.baseUrl);
8    }
9
10   getClienteId(id: string): Observable<any> {
11     return this.http.get(this.baseUrl + '/' + id);
12   }
13
14   deleteCliente(id: any): Observable<any> {
15     return this.http.delete(this.baseUrl + '/delete/' + id, {responseType: 'text'})
16   }
17
18   addCliente(cliente: tCliente): Observable<any> {
19     return this.http.post(this.baseUrl + '/addCliente', cliente)
20   }
21
22   editCliente(id: string, cliente: tCliente): Observable<any> {
23     return this.http.put(this.baseUrl + '/update/' + id, cliente)
24   }
25 }

```

6.1.2. Empleado

```

1  export class EmpleadoService {
2    baseUrl = 'https://api-vapehub.herokuapp.com/empleados'
3
4    constructor(private http: HttpClient) { }
5
6    getEmpleado(): Observable<any> {
7      return this.http.get(this.baseUrl);
8    }
9
10   getSalario(): Observable<any> {
11     return this.http.get(this.baseUrl + '/salario');
12   }
13
14   getEmpleadoId(id: string): Observable<any> {
15     return this.http.get(this.baseUrl + '/' + id);
16   }
17
18   deleteEmpleado(id: any): Observable<any> {
19     return this.http.delete(this.baseUrl + '/delete/' + id, {responseType: 'text'})
20   }
21
22   addEmpleado(empleado: tEmpleado): Observable<any> {
23     return this.http.post(this.baseUrl + '/addEmpleado', empleado)
24   }
25
26   editEmpleado(id: string, empleado: tEmpleado): Observable<any> {
27     return this.http.put(this.baseUrl + '/update/' + id, empleado)
28   }
29 }

```

6.1.3. Líquido

```

1  export class LiquidoService {
2      baseUrl = 'https://api-vapehub.herokuapp.com/liquidos'
3
4      constructor(private http: HttpClient) { }
5
6      getLiquido(): Observable<any> {
7          return this.http.get(this.baseUrl);
8      }
9
10     getLiquidoId(id: string): Observable<any> {
11         return this.http.get(this.baseUrl + '/' + id);
12     }
13 }

```

6.1.4. Dispositivo

```

1  export class DispositivoService {
2      baseUrl = 'https://api-vapehub.herokuapp.com/dispositivos'
3
4      constructor(private http: HttpClient) { }
5
6      getDispositivo(): Observable<any> {
7          return this.http.get(this.baseUrl);
8      }
9
10     getDispositivoId(id: string): Observable<any> {
11         return this.http.get(this.baseUrl + '/' + id);
12     }
13 }

```

6.1.5. Compra

```

1  export class CompraService {
2      baseUrl = 'https://api-vapehub.herokuapp.com'
3
4      constructor(private http: HttpClient) { }
5
6      addCompra(compra: tCompra): Observable<any> {
7          return this.http.post(this.baseUrl + '/addCompra', compra)
8      }
9  }

```

6.1.6. Menú dinámico

```

1  export class MenuService {
2
3      constructor(private http: HttpClient) { }
4
5      getMenu(): Observable<Menu[]> {
6          return this.http.get<Menu[]>('./assets/data/menu.json');
7      }
8  }

```

6.2. Explicación menú

Cómo se ve en el servicio llama a un documento .json el cuál hace que la barra de navegación sea dinámica y de fácil extensión.

Ese documento incluye lo siguiente:

```

1  [
2    {
3      "nombre": "Dashboard",
4      "redirect": "/dashboard"
5    },
6    {
7      "nombre": "Liquidados",
8      "redirect": "/dashboard/liquidados"
9    },
10   {
11     "nombre": "Dispositivos",
12     "redirect": "/dashboard/dispositivos"
13   },
14   {
15     "nombre": "Clientes",
16     "redirect": "/dashboard/clientes"
17   },
18   {
19     "nombre": "Empleados",
20     "redirect": "/dashboard/empleados"
21   },
22   {
23     "nombre": "Reportes",
24     "redirect": "/dashboard/reportes"
25   }
26 ]

```

Y esos campos son importados de una interfaz menu:

```

1  export interface Menu {
2    nombre: string,
3    redirect: string
4  }

```

En el componente de la navbar importamos el servicio y la interfaz y cargamos el menu:

```

1  export class NavbarComponent implements OnInit {
2    menu: Menu[] = [];
3
4    constructor(private _menuService: MenuService) { }
5
6    ngOnInit(): void {
7      this.cargarMenu();
8    }
9
10   cargarMenu(){
11     this._menuService.getMenu().subscribe(data => {
12       this.menu = data;
13     })
14   }
15 }

```

Por último en el html de la navbar recorremos los datos y los insertamos en botones:

```

1  <button mat-button *ngFor="let item of menu" [routerLink]="item.redirect">
2    {{ item.nombre }}
3  </button>

```

6.3. Routing

6.3.1. app-routing

Redirige al login directamente, y si pones cualquier ruta inexistente.

Se crea la ruta padre llamada dashboard.

```
1  const routes: Routes = [
2  { path: '', redirectTo: 'login', pathMatch: 'full' },
3  { path: 'login', component: LoginComponent},
4  { path: 'dashboard', loadChildren: () => import('./components/dashboard/dashboard.module')
    .then(x => x.DashboardModule) },
5  { path: '**', redirectTo: 'login', pathMatch: 'full' }
6  ];
```

6.3.2. dashboard-routing

Rutas hijas:

```
1  const routes: Routes = [
2  {
3    path: '', component: DashboardComponent, children: [
4      { path: '', component: InicioComponent },
5      { path: 'clientes', component: ClientesComponent },
6      { path: 'liquidados', component: LiquidadosComponent },
7      { path: 'dispositivos', component: DispositivosComponent },
8      { path: 'empleados', component: EmpleadosComponent },
9      { path: 'reportes', component: ReportesComponent },
10     { path: 'crear-cliente', component: CrearClienteComponent },
11     { path: 'crear-cliente/:id', component: CrearClienteComponent },
12     { path: 'crear-empleado', component: CrearEmpleadoComponent },
13     { path: 'crear-empleado/:id', component: CrearEmpleadoComponent },
14     { path: 'compra', component: CompraComponent },
15   ]
16 }
17 ];
```

6.4. Módulos

6.4.1. DashboardModule

```

1  @NgModule({
2    declarations: [
3      DashboardComponent,
4      InicioComponent,
5      NavbarComponent,
6      ReportesComponent,
7      ClientesComponent,
8      EmpleadosComponent,
9      CrearClienteComponent,
10     CrearEmpleadoComponent,
11     LiquidosComponent,
12     DispositivosComponent,
13     CompraComponent,
14   ],
15   imports: [
16     CommonModule,
17     DashboardRoutingModule,
18     SharedModule
19   ]
20 })

```

6.4.2. SharedModule

Este SharedModule es para separar los módulos que son importados de librerías de componentes, como los de Angular Material y los de HighCharts.

```

1  @NgModule({
2    declarations: [],
3    imports: [ CommonModule,
4      ReactiveFormsModule,
5      MatFormFieldModule,
6      MatInputModule,
7      MatButtonModule,
8      MatSnackBarModule,
9      MatProgressSpinnerModule,
10     MatToolbarModule,
11     MatIconModule,
12     HttpClientModule,
13     MatTableModule,
14     MatTooltipModule,
15     MatPaginatorModule,
16     MatSortModule,
17     MatCardModule,
18     MatGridListModule,
19     MatSelectModule,
20     MatTabsModule,
21     HighchartsChartModule ],
22    exports : [ ReactiveFormsModule,
23      MatFormFieldModule,
24      MatInputModule,
25      MatButtonModule,
26      MatSnackBarModule,
27      MatProgressSpinnerModule,
28      MatToolbarModule,
29      MatIconModule,
30      HttpClientModule,
31      MatTableModule,
32      MatTooltipModule,
33      MatPaginatorModule,
34      MatSortModule,
35      MatCardModule,
36      MatGridListModule,
37      MatSelectModule,
38      MatTabsModule,
39      HighchartsChartModule ]
40 })

```

6.5. Componentes

6.5.1. ClientesComponent

```

1  export class ClientesComponent implements OnInit {
2
3  loading = true;
4  listClientes: Cliente[] = [];
5
6  displayedColumns = ['id', 'nombre', 'calle', 'numero', 'telefono', 'email', 'socio', '
    acciones'];
7  dataSource!: MatTableDataSource<any>;
8
9  constructor(private _clienteService: ClienteService,
10 private _snackBar: MatSnackBar,
11 private _router: Router) { }
12
13 ngOnInit(): void {
14 this._clienteService.getCliente()
15   .pipe(first())
16   .subscribe(data => {
17     this.listClientes = data,
18     this.dataSource = new MatTableDataSource(this.listClientes);
19     this.loading = false;
20   })
21 }
22
23 applyFilter(event: Event) {
24   const filterValue = (event.target as HTMLInputElement).value;
25   this.dataSource.filter = filterValue.trim().toLowerCase();
26 }
27
28 deleteCliente(id: string) {
29   this._clienteService.deleteCliente(id)
30     .pipe(first())
31     .subscribe(data => {
32       console.log(data);
33       this.ngOnInit();
34       this._snackBar.open('Cliente eliminado correctamente', '', {
35         duration: 1500,
36         horizontalPosition: 'center',
37         verticalPosition: 'bottom'
38       })
39     },
40     error => console.log(error));
41 }
42
43 addCliente() {
44   this._router.navigate(['/dashboard/crear-cliente']);
45 }
46
47 editCliente(id: number) {
48   this._router.navigate(['/dashboard/crear-cliente', id]);
49   console.log(id);
50 }
51 }

```

6.5.2. EmpleadosComponent

```

1  export class EmpleadosComponent implements OnInit {
2
3      loading = true;
4      listEmpleados: Empleado[] = [];
5
6      displayedColumns = ['id', 'nombre', 'calle', 'numero', 'telefono', 'email', 'ventas', '
          horas', 'acciones'];
7      dataSource!: MatTableDataSource<any>;
8
9      constructor(private _empleadoService: EmpleadoService,
10                 private _snackBar: MatSnackBar,
11                 private _router: Router) { }
12
13     ngOnInit(): void {
14         this._empleadoService.getEmpleado().subscribe(data => {
15             this.listEmpleados = data,
16             this.dataSource = new MatTableDataSource(this.listEmpleados);
17             this.loading = false;
18         })
19     }
20
21     applyFilter(event: Event) {
22         const filterValue = (event.target as HTMLInputElement).value;
23         this.dataSource.filter = filterValue.trim().toLowerCase();
24     }
25
26     deleteEmpleado(id: number) {
27         this._empleadoService.deleteEmpleado(id). subscribe(data => {
28             console.log(data);
29             this.ngOnInit();
30         },
31         error => console.log(error));
32     }
33
34     addEmpleado() {
35         this._router.navigate(['/dashboard/crear-empleado']);
36     }
37
38     editEmpleado(id: number) {
39         this._router.navigate(['/dashboard/crear-empleado', id]);
40         console.log(id);
41     }
42 }

```


6.5.3. CrearClienteComponent

```

1  export class CrearClienteComponent implements OnInit {
2  clienteForm!: FormGroup;
3  id!: string;
4  isAddMode!: boolean;
5
6  constructor(
7    private _fb: FormBuilder,
8    private _clienteService: ClienteService,
9    private _aRouter: ActivatedRoute,
10   private _snackBar: MatSnackBar
11 ) { }
12
13 ngOnInit(): void {
14   this.id = this._aRouter.snapshot.params['id'];
15   this.isAddMode = !this.id;
16
17   this.clienteForm = this._fb.group({
18     id: ['', Validators.required],
19     nombre: ['', Validators.required],
20     calle: ['', Validators.required],
21     numero: ['', Validators.required],
22     telefono: ['', Validators.required],
23     email: ['', Validators.required],
24     socio: ['', Validators.required]
25   });
26
27   if (!this.isAddMode) {
28     this.inputCliente();
29   }
30 }
31
32 onSubmit() {
33   if (this.isAddMode) {
34     this.addCliente();
35   } else {
36     this.editCliente();
37   }
38 }
39
40 addCliente() {
41   this._clienteService.addCliente(this.clienteForm.value)
42     .pipe(first())
43     .subscribe();
44 }
45
46 inputCliente() {
47   this._clienteService.getClientId(this.id).subscribe(data => {
48     console.log(data);
49     this.clienteForm.setValue({
50       id: data._id,
51       nombre: data._nombre,
52       calle: data._direccion.calle,
53       numero: data._direccion.numero,
54       telefono: data._telefono,
55       email: data._email,
56       socio: data._socio,
57     })
58   })
59 }
60
61 editCliente() {
62   this._clienteService.editCliente(this.id!, this.clienteForm.value)
63     .pipe(first())
64     .subscribe();
65 }
66 }

```

6.5.4. CrearEmpleadoComponent

```

1  export class CrearEmpleadoComponent implements OnInit {
2      empleadoForm!: FormGroup;
3      id!: string;
4      isAddMode!: boolean;
5
6      constructor(
7          private _fb: FormBuilder,
8          private _empleadoService: EmpleadoService,
9          private _aRouter: ActivatedRoute,
10         private _snackBar: MatSnackBar
11     ) { }
12
13     ngOnInit(): void {
14         this.id = this._aRouter.snapshot.params['id'];
15         this.isAddMode = !this.id;
16
17         this.empleadoForm = this._fb.group({
18             id: ['', Validators.required],
19             nombre: ['', Validators.required],
20             calle: ['', Validators.required],
21             numero: ['', Validators.required],
22             telefono: ['', Validators.required],
23             email: ['', Validators.required],
24             ventas: ['', Validators.required],
25             horas: ['', Validators.required]
26         });
27
28         if (!this.isAddMode) {
29             this.inputEmpleado();
30         }
31     }
32
33     onSubmit() {
34         if (this.isAddMode) {
35             this.addEmpleado();
36         } else {
37             this.editEmpleado();
38         }
39     }
40
41     addEmpleado() {
42         this._empleadoService.addEmpleado(this.empleadoForm.value)
43             .pipe(first())
44             .subscribe();
45     }
46
47     inputEmpleado() {
48         this._empleadoService.getEmpleadoId(this.id).subscribe(data => {
49             console.log(data);
50             this.empleadoForm.setValue({
51                 id: data._id,
52                 nombre: data._nombre,
53                 calle: data._direccion.calle,
54                 numero: data._direccion.numero,
55                 telefono: data._telefono,
56                 email: data._email,
57                 ventas: data._ventas,
58                 horas: data._horas,
59             });
60         });
61     }
62
63     editEmpleado() {
64         this._empleadoService.editEmpleado(this.id!, this.empleadoForm.value)
65             .pipe(first())
66             .subscribe();
67     }
68 }

```

6.5.5. LiquidosComponent

```

1  export class LiquidosComponent implements OnInit {
2      loading = true;
3      listLiquidos: Liquido[] = [];
4      displayedColumns = ['id', 'nombre', 'marca', 'sabor', 'nicotina', 'coste'];
5      dataSource!: MatTableDataSource<any>;
6
7      @ViewChild(MatPaginator) paginator!: MatPaginator;
8      @ViewChild(MatSort) sort!: MatSort;
9
10     constructor(private _liquidoService: LiquidoService,
11                 private _snackBar: MatSnackBar) { }
12
13     ngOnInit(): void {
14         this._liquidoService.getLiquido()
15             .pipe((first()))
16             .subscribe(data => {
17                 this.listLiquidos = data,
18                 this.dataSource = new MatTableDataSource(this.listLiquidos);
19                 this.loading = false;
20             })
21     }
22
23     ngAfterViewInit() {
24         //this.dataSource.paginator = this.paginator;
25         //this.dataSource.sort = this.sort;
26     }
27
28     applyFilter(event: Event) {
29         const filterValue = (event.target as HTMLInputElement).value;
30         this.dataSource.filter = filterValue.trim().toLowerCase();
31     }
32 }

```

6.5.6. DispositivosComponent

```

1  export class DispositivosComponent implements OnInit {
2      loading = true;
3      listDispositivos: Dispositivo[] = [];
4      displayedColumns = ['id', 'nombre', 'marca', 'bateria', 'potencia', 'coste'];
5      dataSource!: MatTableDataSource<any>;
6
7      @ViewChild(MatPaginator) paginator!: MatPaginator;
8      @ViewChild(MatSort) sort!: MatSort;
9
10     constructor(private _dispositivoService: DispositivoService,
11                 private _snackBar: MatSnackBar) { }
12
13     ngOnInit(): void {
14         this._dispositivoService.getDispositivo()
15             .pipe((first()))
16             .subscribe(data => {
17                 this.listDispositivos = data,
18                 this.dataSource = new MatTableDataSource(this.listDispositivos);
19                 this.loading = false;
20             })
21     }
22
23     ngAfterViewInit() {
24         //this.dataSource.paginator = this.paginator;
25         //this.dataSource.sort = this.sort;
26     }
27
28     applyFilter(event: Event) {
29         const filterValue = (event.target as HTMLInputElement).value;
30         this.dataSource.filter = filterValue.trim().toLowerCase();
31     }
32 }

```

6.5.7. CompraComponent

```

1  export class CompraComponent implements OnInit {
2      compraForm!: FormGroup;
3      id!: string;
4      idCliente: string = "";
5      listCompra: Compra[] = [];
6      coste!: number;
7
8      constructor(
9          private _fb: FormBuilder,
10         private _compraService: CompraService,
11         private _liquidoService: LiquidoService,
12         private _dispositivoService: DispositivoService,
13         private _snackBar: MatSnackBar
14     ) { }
15
16     ngOnInit(): void {
17         this.compraForm = this._fb.group({
18             id: ['', Validators.required],
19             dni: ['', Validators.required],
20             idp: ['', Validators.required],
21         })
22     }
23
24     onSubmit() {
25         this.addCompra();
26     }
27
28     addCompra() {
29         this.id = this.compraForm.get('idp')?.value
30
31         if (this.id >= '100') {
32             this._liquidoService.getLiquidoId(this.id)
33                 .subscribe(data => {
34                     console.log(data);
35                     this.coste = data._coste
36
37                     const COMPRA: tCompra = {
38                         id: this.compraForm.get('id')?.value,
39                         idCliente: this.compraForm.get('dni')?.value,
40                         idProducto: this.compraForm.get('idp')?.value,
41                         coste: this.coste
42                     }
43                     this._compraService.addCompra(COMPRA)
44                         .subscribe()
45                     this.compraForm.reset()
46                 })
47             if (this.id < '100') {
48                 this._dispositivoService.getDispositivoId(this.id)
49                     .subscribe(data => {
50                         console.log(data);
51                         this.coste = data._coste
52
53                         const COMPRA: tCompra = {
54                             id: this.compraForm.get('id')?.value,
55                             idCliente: this.compraForm.get('dni')?.value,
56                             idProducto: this.compraForm.get('idp')?.value,
57                             coste: this.coste
58                         }
59                         this._compraService.addCompra(COMPRA)
60                             .subscribe()
61                         this.compraForm.reset()
62                     })
63             }
64         }
65     }

```

6.5.8. ReportesComponent

```

1  export class ReportesComponent implements OnInit {
2      Highcharts: typeof Highcharts = Highcharts;
3      listSalario: Salario[] = []
4
5      chartOptions: any = {
6          chart:
7          {
8              backgroundColor: {
9                  linearGradient: [500, 500, 500, 500],
10                 stops: [
11                     [0, 'rgb(255, 255, 255)'],
12                 ]
13             },
14             type: 'column',
15             title: {
16                 text: '',
17             },
18             xAxis: {
19                 categories: []
20             },
21             credits: {
22                 enabled: false
23             },
24             series: [{
25                 name: '',
26                 data: []
27             }]
28         };
29
30         constructor(private _empleadoService: EmpleadoService) { }
31
32         ngOnInit(): void {
33             this.salarioEmpleado();
34         }
35
36         salarioEmpleado(){
37             this._empleadoService.getSalario().subscribe((data) => {
38                 this.listSalario = data
39                 this.listSalario.map((salario: any) => {
40                     return new Salario(salario._id, salario._nombre, salario._salario)
41                 })
42                 const dataSeries = this.listSalario.map((x: Salario) => x._salario)
43                 const dataCategorias = this.listSalario.map((x: Salario) => x._nombre)
44
45                 this.chartOptions.title["text"] = "Salario de empleados"
46                 this.chartOptions.series[0]["data"] = dataSeries
47                 this.chartOptions.xAxis["categories"] = dataCategorias
48                 this.chartOptions.series["name"] = "Empleados"
49
50                 Highcharts.chart("salario", this.chartOptions)
51             })
52         }
53     }
54 }

```

6.5.9. LoginComponent

```

1  export class LoginComponent implements OnInit {
2      form: FormGroup;
3      loading = false;
4
5      constructor(private fb: FormBuilder, private _snackBar: MatSnackBar, private router:
6          Router) {
7          this.form = this.fb.group({
8              user: ['', Validators.required],
9              password: ['', Validators.required]
10         })
11     }
12
13     ngOnInit(): void {
14     }
15
16     ingresar() {
17         const user = this.form.value.user;
18         const password = this.form.value.password;
19
20         if(user == 'ismael' && password == '1234') {
21             // Redireccion dashboard
22             this.fakeLoading();
23         } else {
24             // Mostrar error
25             this.error();
26             this.form.reset()
27         }
28     }
29
30     error() {
31         this._snackBar.open('Usuario o password incorrecta', '', {
32             duration: 3000,
33             horizontalPosition: 'center',
34             verticalPosition: 'bottom'
35         })
36     }
37
38     fakeLoading() {
39         this.loading = true;
40         setTimeout(() => {
41             // Redireccion dashboard
42             this.router.navigate(['dashboard'])
43         }, 1500);
44     }
45 }

```