Proyecto Angular

VapeHub

Índice

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

```
export class Persona {
protected _id: string;
protected _nombre: string;
protected _direccion: { calle: string; numero: number };
protected _telefono: number;
protected _email: string;
constructor(
  id: string,
  nombre: string,
  direccion: { calle: string; numero: number },
  telefono: number,
  email: string,
) {
  this._id = id;
  this._nombre = nombre;
  this._direccion = direccion;
  this._telefono = telefono;
  this._email = email;
}
get id() {
  return this._id;
}
get nombre() {
  return this._nombre;
}
get direccion() {
  return this._direccion;
}
get telefono() {
  return this._telefono;
}
get email() {
  return this._email;
}

todo() {
  return `ID: ${this._id},
  Nombre: ${this._nombre},
  Direccion: ${this._direccion},
  Telefono: ${this._telefono},
  Email : ${this._email}`;
}
}
```

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

```
export class Producto {
  protected _id: string;
  protected _nombreProd: string;
  protected _marca: string;
  protected _coste: number;

  constructor(id: string, nombreProd: string, marca: string, coste: number) {
    this._id = id;
    this._nombreProd = nombreProd;
    this._marca = marca;
    this._coste = coste;
  }

  get id() {
    return this._id;
  }
  get nombreProd() {
    return this._nombreProd;
  }
  get marca() {
    return this._marca;
  }
  get coste() {
    return this._coste;
  }

  todoProd() {
    return `ID: ${this._id},
    Nombre de producto: ${this._nombreProd},
    Marca: ${this._marca}
    coste ${this._coste}`;
  }
  }
```

2.2.2. Líquido

```
export class Liquido extends Producto {

protected _sabor: string;
protected _nicotina: number;

constructor(
  id: string,
  nombreProd: string,
  marca: string,
  sabor: string,
  nicotina: number,
  coste: number
) {
  super(id, nombreProd, marca, coste);

  this._sabor = sabor;
  this._nicotina = nicotina;
}
get sabor() {
  return this._sabor;
}
get nicotina() {
  return this._nicotina;
}
}
```

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

```
export class Compra {
  protected _id: string;
  protected _idCliente: string;
  protected _coste: number;
  protected _idProducto: string;

  constructor(id: string, idCliente: string, coste: number, idProducto: string) {
    this._id = id;
    this._idCliente = idCliente;
    this._coste = coste;
    this._idProducto = idProducto;
  }
  get getId() {
    return this._id;
  }
  get getCliente() {
    return this._idCliente;
  }
  get getProductos() {
    return this._idProducto;
  }

  todoCompra() {
    return `ID: ${this._id},
      : nombreCliente: ${this._idCliente},
      coste: ${this._coste},
      productos:  ${this._idProducto}`;
  }
}
```

4. Tipos

4.1. Persona

```
export type tCliente = {
  id: string | null;
  nombre: string | null;
  calle: string;
  numero: number;
  telefono: number | null;
  email: string | null;
  socio: Boolean | null;
};

export type tEmpleado = {
  id: string | null;
  nombre: string | null;
  calle: string;
  numero: number;
  telefono: number | null;
  email: string | null;
  ventas: number | null;
  horas: number | null;
};

export type tEmpleado2 = {
  _id: string;
  _nombre: string;
  _direccion: {
    calle: string;
    numero: number;
  };
  _telefono: number;
  _email: string;
  _ventas: number;
  _horas: number;
};

export type tSalario = {
  _id: string | null;
  _nombre: string | null;
  _salario: number | null;
};
```

4.2. Producto

```
export type tDispositivo = {
  id: string | null;
  nombreProd: string | null;
  marca: string | null;
  potencia: number | null;
  bateria: number | null;
  coste: number | null;
};
export type tLiquido = {
  id: string | null;
  nombreProd: string | null;
  marca: string | null;
  sabor: string | null;
  nicotina: number | null;
  coste: number | null;
};
```

4.3. Compra

```
export type tCompra = {
  id: string | null;
  idCliente: string | null;
  idProducto: string | null;
  coste: number | null;
  };
```

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:

```
// Add empleado
private addEmpleado = async (req: Request, res: Response) => {
  const { id, nombre, calle, numero, telefono, email, ventas, horas } = req.body
  await db.conectarBD()
  const dSchema = {
    _id: id,
    _nombre: nombre,
    _direccion: { calle: calle, numero: numero },
    _telefono: telefono,
    _email: email,
    _ventas: ventas,
    _horas: horas
  }
  const oSchema = new EmpleadoDB(dSchema)
  await oSchema.save()
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

  db.desconectarBD()
}

// Add cliente:
private addCliente = async (req: Request, res: Response) => {
  const { id, nombre, calle, numero, telefono, email, socio } = req.body
  await db.conectarBD()
  const dSchema = {
    _id: id,
    _nombre: nombre,
    _direccion: { calle: calle, numero: numero },
    _telefono: telefono,
    _email: email,
    _socio: socio
  }
  const oSchema = new ClienteDB(dSchema)
  await oSchema.save()
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

  db.desconectarBD()
}

// Add compra
private addCompra = async (req: Request, res: Response) => {
  const { id, idCliente, idProducto, coste } = req.body
  let fecha: Date = new Date()
  await db.conectarBD()
  const dSchema = {
    _id: id,
    _idCliente: idCliente,
    _idProducto: idProducto,
    _coste: coste,
    _fecha: fecha
  }
  const oSchema = new CompraDB(dSchema)
  await oSchema.save()
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

  db.desconectarBD()
}
```

- 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:

```
// Buscar empleado
private getEmpleadoId = async (req: Request, res: Response) => {
await db.conectarBD()
  .then(async (mensaje) => {
  const id = req.params.id
  console.log(mensaje)
  const query = await EmpleadoDB.findOne({ _id: id });
  res.json(query)
  })
  .catch((mensaje) => {
  res.send(mensaje)
  })
await db.desconectarBD()
}

// Buscar cliente
private getClienteId = async (req: Request, res: Response) => {
await db.conectarBD()
  .then(async (mensaje) => {
  const id = req.params.id
  console.log(mensaje)
  const query = await ClienteDB.findOne({ _id: id });
  res.json(query)
  })
  .catch((mensaje) => {
  res.send(mensaje)
  })
await db.desconectarBD()
}

// Buscar dispositivo
private getDispositivoId = async (req: Request, res: Response) => {
await db.conectarBD()
  .then(async (mensaje) => {
  const id = req.params.id
  console.log(mensaje)
  const query = await DispositivoDB.findOne({ _id: id });
  res.json(query)
  })
  .catch((mensaje) => {
  res.send(mensaje)
  })
await db.desconectarBD()
}

// Buscar liquido
private getLiquidoId = async (req: Request, res: Response) => {
await db.conectarBD()
  .then(async (mensaje) => {
  const id = req.params.id
  console.log(mensaje)
  const query = await LiquidoDB.findOne({ _id: id });
  res.json(query)
  })
  .catch((mensaje) => {
  res.send(mensaje)
  })
await db.desconectarBD()
}
```

- 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:

```
  // RUTAS
  misRutas() {
    // Listar
    this._router.get('/empleados/salario', this.getSalario)
    this._router.get('/empleados', this.getEmpleados)
    this._router.get('/clientes', this.getClientes)
    this._router.get('/empleados/:id', this.getEmpleadoId)
    this._router.get('/clientes/:id', this.getClienteId)
    this._router.get('/dispositivos', this.getDispositivos)
    this._router.get('/liquidos', this.getLiquidos)
    this._router.get('/dispositivos/:id', this.getDispositivoId)
    this._router.get('/liquidos/:id', this.getLiquidoId)
    // Crear
    this._router.post('/empleados/addEmpleado', this.addEmpleado)
    this._router.post('/clientes/addCliente', this.addCliente)
    this._router.post('/addCompra', this.addCompra)
    // Actualizar
    this._router.put('/empleados/update/:id', this.updateEmpleado)
    this._router.put('/clientes/update/:id', this.updateCliente)
    // Eliminar
    this._router.delete('/empleados/delete/:id', this.delEmpleado)
    this._router.delete('/clientes/delete/:id', this.delCliente)
  }
  }
```

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

```
export class LiquidoService {
  baseUrl = 'https://api-vapehub.herokuapp.com/liquidos'

  constructor(private http: HttpClient) { }

  getLiquido(): Observable<any> {
    return this.http.get(this.baseUrl);
  }

  getLiquidoId(id: string): Observable<any> {
    return this.http.get(this.baseUrl + '/' + id);
  }
}
```

6.1.4. Dispositivo

```
export class DispositivoService {
baseUrl = 'https://api-vapehub.herokuapp.com/dispositivos'

constructor(private http: HttpClient) { }

getDispositivo(): Observable<any> {
  return this.http.get(this.baseUrl);
}

getDispositivoId(id: string): Observable<any> {
  return this.http.get(this.baseUrl + '/' + id);
}
}
```

6.1.5. Compra

```
export class CompraService {
  baseUrl = 'https://api-vapehub.herokuapp.com'

  constructor(private http: HttpClient) { }

  addCompra(compra: tCompra): Observable<any> {
    return this.http.post(this.baseUrl + '/addCompra', compra)
  }
}
```

6.1.6. Menú dinámico

```
export class MenuService {

  constructor(private http: HttpClient) { }

  getMenu(): Observable<Menu[]> {
    return this.http.get<Menu[]>('./assets/data/menu.json');
  }
}
```

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:

```
[
  {
      "nombre": "Dashboard",
      "redirect": "/dashboard"
  },
  {
      "nombre": "Liquidos",
      "redirect": "/dashboard/liquidos"
  },
  {
      "nombre": "Dispositivos",
      "redirect": "/dashboard/dispositivos"
  },
  {
      "nombre": "Clientes",
      "redirect": "/dashboard/clientes"
  },
  {
      "nombre": "Empleados",
      "redirect": "/dashboard/empleados"
  },
  {
      "nombre": "Reportes",
      "redirect": "/dashboard/reportes"
  }
]
```

Y esos campos son importados de una interfaz menu:

```
export interface Menu {
  nombre: string,
  redirect: string
}
```

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

```
export class NavbarComponent implements OnInit {
  menu: Menu[] = [];

  constructor(private _menuService: MenuService) { }

  ngOnInit(): void {
  this.cargarMenu();
  }

  cargarMenu(){
  this._menuService.getMenu().subscribe(data => {
    this.menu = data;
  })
  }
}
```

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

```
<button mat-button *ngFor="let item of menu" [routerLink]="item.redirect">
{{ item.nombre }}
</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: 'liquidos', component: LiquidosComponent },
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

```
  @NgModule({
  declarations: [
    DashboardComponent,
    InicioComponent,
    NavbarComponent,
    ReportesComponent,
    ClientesComponent,
    EmpleadosComponent,
    CrearClienteComponent,
    CrearEmpleadoComponent,
    LiquidosComponent,
    DispositivosComponent,
    CompraComponent,
],
  imports: [
    CommonModule,
    DashboardRoutingModule,
    SharedModule
  ]
})
```

6.4.2. SharedModule

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

```
  @NgModule({
  declarations: [],
  imports: [ CommonModule,
    ReactiveFormsModule,
    MatFormFieldModule,
    MatInputModule,
    MatButtonModule,
    MatSnackBarModule,
    MatProgressSpinnerModule,
    MatToolbarModule,
    MatIconModule,
    HttpClientModule,
    MatTableModule,
    MatTooltipModule,
    MatPaginatorModule,
    MatSortModule,
    MatCardModule,
    MatGridListModule,
    MatSelectModule,
    MatTabsModule,
    HighchartsChartModule ],
  exports : [ ReactiveFormsModule,
    MatFormFieldModule,
    MatInputModule,
    MatButtonModule,
    MatSnackBarModule,
    MatProgressSpinnerModule,
    MatToolbarModule,
    MatIconModule,
    HttpClientModule,
    MatTableModule,
    MatTooltipModule,
    MatPaginatorModule,
    MatSortModule,
    MatCardModule,
    MatGridListModule,
    MatSelectModule,
    MatTabsModule,
    HighchartsChartModule ]
})
```

6.5. Componentes

6.5.1. ClientesComponent

```
export class ClientesComponent implements OnInit {

loading = true;
listClientes: Cliente[] = [];

displayedColumns = ['id', 'nombre', 'calle', 'numero', 'telefono', 'email', 'socio', '
    acciones'];
dataSource!: MatTableDataSource<any>;

constructor(private _clienteService: ClienteService,
  private _snackBar: MatSnackBar,
  private _router: Router) { }

ngOnInit(): void {
  this._clienteService.getCliente()
    .pipe((first()))
    .subscribe(data => {
      this.listClientes = data,
        this.dataSource = new MatTableDataSource(this.listClientes);
      this.loading = false;
    })
}

applyFilter(event: Event) {
  const filterValue = (event.target as HTMLInputElement).value;
  this.dataSource.filter = filterValue.trim().toLowerCase();
}

deleteCliente(id: string) {
  this._clienteService.deleteCliente(id)
    .pipe((first()))
    .subscribe(data => {
      console.log(data);
      this.ngOnInit();
      this._snackBar.open('Cliente eliminado correctamente', '', {
        duration: 1500,
        horizontalPosition: 'center',
        verticalPosition: 'bottom'
      })
    },
      error => console.log(error));
}

addCliente() {
  this._router.navigate(['/dashboard/crear-cliente']);
}

editCliente(id: number) {
  this._router.navigate(['/dashboard/crear-cliente', id]);
  console.log(id);
}
}
```

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

```
export class CrearClienteComponent implements OnInit {
clienteForm!: FormGroup;
id!: string;
isAddMode!: boolean;

constructor(
  private _fb: FormBuilder,
  private _clienteService: ClienteService,
  private _aRouter: ActivatedRoute,
  private _snackBar: MatSnackBar
) { }

ngOnInit(): void {
  this.id = this._aRouter.snapshot.params['id'];
  this.isAddMode = !this.id;

  this.clienteForm = this._fb.group({
    id: ['', Validators.required],
    nombre: ['', Validators.required],
    calle: ['', Validators.required],
    numero: ['', Validators.required],
    telefono: ['', Validators.required],
    email: ['', Validators.required],
    socio: ['', Validators.required]
  });

  if (!this.isAddMode) {
    this.inputCliente();
  }
}

onSubmit() {
  if (this.isAddMode) {
    this.addCliente();
  } else {
    this.editCliente();
  }
}

addCliente() {
  this._clienteService.addCliente(this.clienteForm.value)
    .pipe(first())
    .subscribe();
}

inputCliente() {
  this._clienteService.getClienteId(this.id).subscribe(data => {
    console.log(data);
    this.clienteForm.setValue({
      id: data._id,
      nombre: data._nombre,
      calle: data._direccion.calle,
      numero: data._direccion.numero,
      telefono: data._telefono,
      email: data._email,
      socio: data._socio,
    })
  })
}

editCliente() {
  this._clienteService.editCliente(this.id!, this.clienteForm.value)
    .pipe(first())
    .subscribe();
}
}
```

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

```
export class LiquidosComponent implements OnInit {
  loading = true;
  listLiquidos: Liquido[] = [];
  displayedColumns = ['id', 'nombre', 'marca', 'sabor', 'nicotina', 'coste'];
  dataSource!: MatTableDataSource<any>;

  @ViewChild(MatPaginator) paginator!: MatPaginator;
  @ViewChild(MatSort) sort!: MatSort;

  constructor(private _liquidoService: LiquidoService,
    private _snackBar: MatSnackBar) { }

  ngOnInit(): void {
    this._liquidoService.getLiquido()
    .pipe((first()))
    .subscribe(data => {
      this.listLiquidos = data,
      this.dataSource = new MatTableDataSource(this.listLiquidos);
      this.loading = false;
    })
  }

  ngAfterViewInit() {
    //this.dataSource.paginator = this.paginator;
    //this.dataSource.sort = this.sort;
  }

  applyFilter(event: Event) {
    const filterValue = (event.target as HTMLInputElement).value;
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }
}
```

6.5.6. DispositivosComponent

```
export class DispositivosComponent implements OnInit {
  loading = true;
  listDispositivos: Dispositivo[] = [];
  displayedColumns = ['id', 'nombre', 'marca', 'bateria', 'potencia', 'coste'];
  dataSource!: MatTableDataSource<any>;

  @ViewChild(MatPaginator) paginator!: MatPaginator;
  @ViewChild(MatSort) sort!: MatSort;

  constructor(private _dispositivoService: DispositivoService,
    private _snackBar: MatSnackBar) { }

  ngOnInit(): void {
    this._dispositivoService.getDispositivo()
    .pipe((first()))
    .subscribe(data => {
      this.listDispositivos = data,
      this.dataSource = new MatTableDataSource(this.listDispositivos);
      this.loading = false;
    })
  }

  ngAfterViewInit() {
    //this.dataSource.paginator = this.paginator;
    //this.dataSource.sort = this.sort;
  }

  applyFilter(event: Event) {
    const filterValue = (event.target as HTMLInputElement).value;
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }
}
```

6.5.7. CompraComponent

```
export class CompraComponent implements OnInit {
  compraForm!: FormGroup;
  id!: string;
  idCliente: string = "";
  listCompra: Compra[] = [];
  coste!: number;

  constructor(
    private _fb: FormBuilder,
    private _compraService: CompraService,
    private _liquidoService: LiquidoService,
    private _dispositivoService: DispositivoService,
    private _snackBar: MatSnackBar
  ) { }

  ngOnInit(): void {
    this.compraForm = this._fb.group({
    id: ['', Validators.required],
    dni: ['', Validators.required],
    idp: ['', Validators.required],
    })
  }

  onSubmit() {
    this.addCompra();
  }

  addCompra() {
    this.id = this.compraForm.get('idp')?.value

    if (this.id >= '100') {
    this._liquidoService.getLiquidoId(this.id)
      .subscribe(data => {
      console.log(data);
      this.coste = data._coste

      const COMPRA: tCompra = {
        id: this.compraForm.get('id')?.value,
        idCliente: this.compraForm.get('dni')?.value,
        idProducto: this.compraForm.get('idp')?.value,
        coste: this.coste
      }
      this._compraService.addCompra(COMPRA)
        .subscribe()
      this.compraForm.reset()
      })
    } if (this.id < '100') {
    this._dispositivoService.getDispositivoId(this.id)
      .subscribe(data => {
      console.log(data);
      this.coste = data._coste

      const COMPRA: tCompra = {
        id: this.compraForm.get('id')?.value,
        idCliente: this.compraForm.get('dni')?.value,
        idProducto: this.compraForm.get('idp')?.value,
        coste: this.coste
      }
      this._compraService.addCompra(COMPRA)
        .subscribe()
      this.compraForm.reset()
      })
    }
  }
}
```

6.5.8. ReportesComponent

```
export class ReportesComponent implements OnInit {
  Highcharts: typeof Highcharts = Highcharts;
  listSalario: Salario[] = []

  chartOptions: any = {
    chart:
    {
    backgroundColor: {
      linearGradient: [500, 500, 500, 500],
      stops: [
      [0, 'rgb(255, 255, 255)'],
      ]
    },
    type: 'column'
    },
    title: {
    text: ''
    },
    xAxis: {
    categories: []
    },
    credits: {
    enabled: false
    },
    series: [{
    name: '',
    data: []
    }]
  };

  constructor(private _empleadoService: EmpleadoService) { }

  ngOnInit(): void {
    this.salarioEmpleado();
  }

  salarioEmpleado(){
    this._empleadoService.getSalario().subscribe((data) => {
    this.listSalario = data
    this.listSalario.map((salario: any) => {
      return new Salario(salario._id, salario._nombre, salario._salario)
    })
    const dataSeries = this.listSalario.map((x: Salario) => x._salario)
    const dataCategorias = this.listSalario.map((x: Salario) => x._nombre)

    this.chartOptions.title["text"] = "Salario de empleados"
    this.chartOptions.series[0]["data"] = dataSeries
    this.chartOptions.xAxis["categories"] = dataCategorias
    this.chartOptions.series["name"] = "Empleados"

    Highcharts.chart("salario", this.chartOptions)
    })
  }
}
```

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:
          Router) {
6       this.form = this.fb.group({
7       user: ['', Validators.required],
8       password: ['', Validators.required]
9       })
10    }
11
12    ngOnInit(): void {
13    }
14
15    ingresar()  {
16
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    error() {
30      this._snackBar.open('Usuario o password incorrecta', '', {
31      duration: 3000,
32      horizontalPosition: 'center',
33      verticalPosition: 'bottom'
34      })
35    }
36    fakeLoading() {
37      this.loading = true;
38      setTimeout(() => {
39      // Redireccion dashboard
40      this.router.navigate(['dashboard'])
41      }, 1500);
42    }
43    }
```