Proyecto MEAN Stack / eCommerce

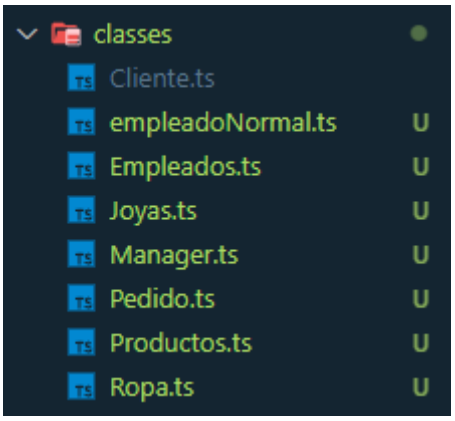
Realizado por: Gonzalo Pazos Sardella

APIREST

Esta se compone de:

- Clases
- Base de datos
- Esquemas
- Rutas

Las clases creadas son las siguientes:



Subclase Manager con su respectivo método.

```ts
import { Empleado } from "./Empleados";

export class Manager extends Empleado {
    protected _estudios: string


    constructor(dni: string, tipoEmpleado: string, nombre: string, tlf:number, departamento: string, estudios:string, sueldo:number){
        super(dni, nombre, tlf, sueldo, departamento, tipoEmpleado)
        this._estudios = estudios
    }

    get estudios(){
        return this._estudios
    }

    sueldoFinal(): number{
        let sueldoFinal: number = super.sueldoFinal()

        if(this._estudios == "Universidad"){
            sueldoFinal = sueldoFinal + 20
        }

        return Math.trunc(sueldoFinal)
    }
}
```

Esquema empleado:

```
import { model } from 'mongoose'

var mongoose = require('mongoose');
var Schema = mongoose.Schema;

const empleadoSchema = new Schema({
    dni: String,
    tipoEmpleado: String,
    nombre: String,
    tlf: Number,
    departamento: String,
    jornada: String,
    horario: String,
    sueldo: Number,
    estudios: String
})

export type iEmpleadoNormal = {
    dni: string | null
    tipoEmpleado: string | null
    nombre: string | null
    tlf: number | null
    departamento: string | null
    horario: string | null
    jornada: string | null
    sueldo: number | null
}

export type iManager = {
    dni: string | null
    tipoEmpleado: string | null
    nombre: string | null
    tlf: number | null
    departamento: string | null
    estudios: string | null
    sueldo: number | null
}

export type iManager2 = {
    dni: string
    tipoEmpleado: string
    nombre: string
    tlf: number
    departamento: string
    estudios: string
    sueldo: number
}

export type iSueldo = {
    dni: string | null,
    nombre: string | null,
    sueldoFinal: number | null
}

export const Empleados = model('empleados', empleadoSchema)
```

Conexión base de datos:

```
import mongoose from 'mongoose';      You, 5 days ago • First commit

class DataBase {

    // private _cadenaConexion: string = 'mongodb://localhost/test'
    private _cadenaConexion: string = `mongodb+srv://gonzalo:abc@proyecto.mtaer.mongodb.net/ProyectoAngular?retryWrites=true&w=majority`
    //private _cadenaConexion2:string= 'mongodb+srv://***:****@cluster0.sentg.mongodb.net/test?retryWrites=true&w=majority'
    constructor(){

    }
    set cadenaConexion(_cadenaConexion: string){
        this._cadenaConexion = _cadenaConexion
    }

    conectarBD = async () => {
        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.connect(this._cadenaConexion, {
            })
            .then( () => resolve(`Conectado a ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error conectando a ${this._cadenaConexion}: ${error}`) )
        })
        return promise

    }

    desconectarBD = async () => {

        const promise = new Promise<string>( async (resolve, reject) => {
            await mongoose.disconnect()
            .then( () => resolve(`Desconectado de ${this._cadenaConexion}`) )
            .catch( (error) => reject(`Error desconectando de ${this._cadenaConexion}: ${error}`) )
        })
        return promise
    }

}
export const db = new DataBase()
```

La api rest contiene las principales características, es decir, los métodos put, post, get y delete.

Delete:

```
private deletePedido = async (req: Request, res: Response) => {
    const dniCliente  = req.params.dniCliente
    await db.conectarBD()
    await Pedidos.findOneAndDelete({ dniCliente: dniCliente })
        .then( (doc: any) => res.send('Borrado correcto'))
        .catch( (err: any) => res.send('Error: ' + err))
    await db.desconectarBD()
}
```

Put:

```
private putJoya = async (req: Request, res: Response) => {
    await db
      .conectarBD()
      .then(async (mensaje) => {
        const {cBarra} = req.params
        const { tipoProducto, marca, descripcion, material, piedraPreciosa, peso, precio } = req.body
        await Productos.findOneAndUpdate(
          {
            cBarra: cBarra,
          },
          {
            cBarra: cBarra,
            tipoProducto: tipoProducto,
            marca: marca,
            descripcion: descripcion,
            material: material,
            piedraPreciosa: piedraPreciosa,
            peso: peso,
            precio: precio
          },
          {
            new: true,
          }
        )
          .then((docu: any) => res.send(docu))
          .catch((fail: any) => res.send(fail));
      })
      .catch((mensaje) => {
        res.send(mensaje);
      });

    db.desconectarBD();
}
```

Post:

```
private postLogin = async (req: Request, res: Response) => {
  const { user, password } = req.body
  await db.conectarBD()
  const dSchema={
        user: user,
        password: password
  }
  const oSchema = new Login(dSchema)
  await oSchema.save()
        .then( (doc: any) => res.send(doc))
        .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}
```

Get:

```
private getRopas = async (req:Request, res: Response) => {
  await db.conectarBD()
  .then(async (mensaje) => {
        const cBarra = req.params.cBarra
        const query = await Productos.find({ tipoProducto: { $eq: "Ropa" } })
        res.json(query)
  })
  .catch((mensaje) => {
        res.send(mensaje)
  })
  await db.desconectarBD()
}
```

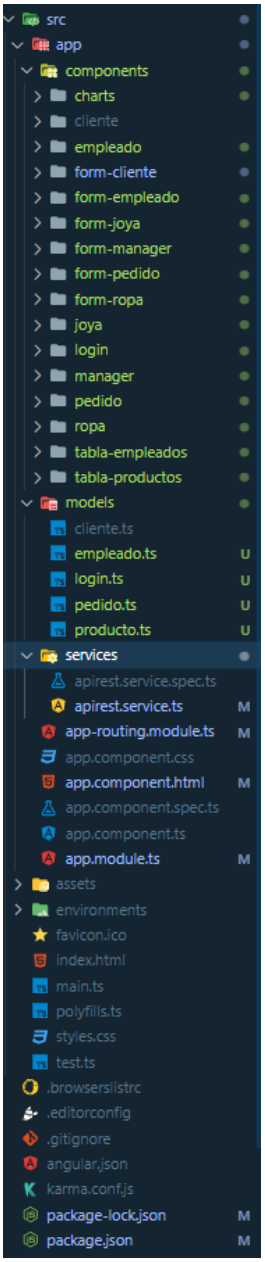
Rutas:

```
misRutas() {
    // Clientes
    this._router.get('/getCliente/:dni', this.getCliente),
    this._router.get('/getClientes', this.getClientes),
    this._router.post('/postCliente', this.postCliente),
    this._router.put("/putCliente/:dni", this.putCliente),
    this._router.delete("/delCliente/:dni", this.deleteCliente),

    // Empleados
    this._router.get('/getEmpleado/:dni', this.getEmpleado),
    this._router.get('/getManagers', this.getManagers),
    this._router.get('/getEmpleados', this.getEmpleados),
    this._router.post('/postEmpleado', this.postEmpleado),
    this._router.post('/postManager', this.postManager),
    this._router.put('/putEmpleado/:dni', this.putEmpleado),
    this._router.put('/putManager/:dni', this.putManager),
    this._router.delete("/delEmpleado/:dni", this.deleteEmpleado),

    // Productos
    this._router.get('/getProducto/:cBarra', this.getProducto),
    this._router.get('/getJoyas', this.getJoyas),
    this._router.get('/getRopas', this.getRopas),
    this._router.post('/postJoya', this.postJoya),
    this._router.post('/postRopa', this.postRopa),
    this._router.put('/putJoya/:cBarra', this.putJoya),
    this._router.put('/putRopa/:cBarra', this.putRopa),
    this._router.delete("/delProducto/:cBarra", this.deleteProducto),

    // Pedidos
    this._router.get('/getPedido/:dniCliente', this.getPedido),
    this._router.get('/getPedidos', this.getPedidos),
    this._router.post('/postPedido', this.postPedido),
    this._router.put("/putPedido/:dniCliente", this.putPedido),
    this._router.delete("/delPedido/:dniCliente", this.deletePedido)

    // Login
    this._router.get('/getLogin/:user/:password', this.getLogin),
    this._router.post('/postLogin', this.postLogin)
  }
```

Angular

El proyecto de angular está compuesto de las siguientes secciones:

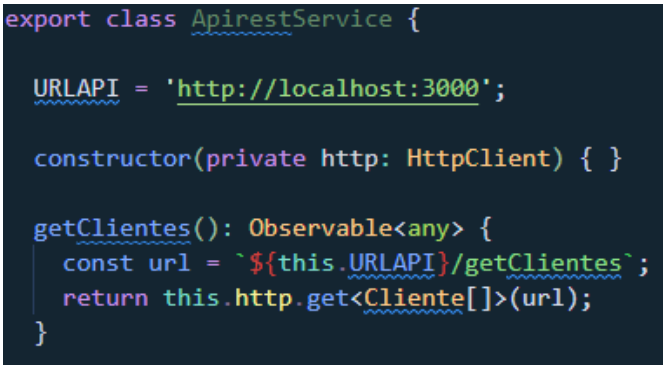
- Componentes
- Rutas
- Servicios
- Modelos



Rutas de navegación:

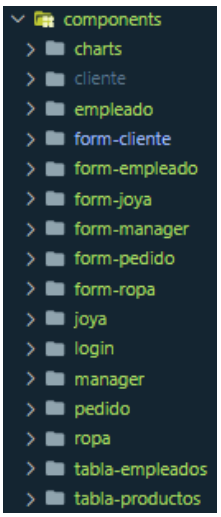
```
const routes: Routes = [
  { path: "", component: CEmpleadosComponent },      You, seconds ago • Uncommitted changes
  { path: "Clientes", component: ClienteComponent },
  { path: "Clientes/formCliente", component: FormClienteComponent },
  { path: "Clientes/formCliente/:dni", component: FormClienteComponent },
  { path: "tablaEmpleados", component: TablaEmpleadosComponent },
  { path: "tablaEmpleados/Empleados", component: EmpleadoComponent },
  { path: "tablaEmpleados/Empleados/formEmpleado", component: FormEmpleadoComponent },
  { path: "tablaEmpleados/Empleados/formEmpleado/:dni", component: FormEmpleadoComponent },
  { path: "tablaEmpleados/loginForm", component: LoginComponent },
  { path: "tablaEmpleados/loginForm/Managers", component: ManagerComponent },
  { path: "tablaEmpleados/loginForm/Managers/formManager", component: FormManagerComponent },
  { path: "tablaEmpleados/loginForm/Managers/formManager/:dni", component: FormManagerComponent },
  { path: "tablaProductos", component: TablaProductosComponent },
  { path: "tablaProductos/Joyas", component: JoyaComponent },
  { path: "tablaProductos/Joyas/formJoya", component: FormJoyaComponent },
  { path: "tablaProductos/Joyas/formJoya/:cBarra", component: FormJoyaComponent },
  { path: "tablaProductos/Ropas", component: RopaComponent },
  { path: "tablaProductos/Ropas/formRopa", component: FormRopaComponent },
  { path: "tablaProductos/Ropas/formRopa/:cBarra", component: FormRopaComponent },
  { path: "Pedidos", component: PedidoComponent },
  { path: "Pedidos/formPedido", component: FormPedidoComponent },
  { path: "Pedidos/formPedido/:dniCliente", component: FormPedidoComponent },
  { path: "Estadisticas", component: CEmpleadosComponent },
];
```

El servicio contendrá la conexión entre nuestra Api Rest con nuestra base de datos, directamente a nuestro Angular. Pudiendo así hacer uso de esta.



```
export class ApirestService {

  URLAPI = 'http://localhost:3000';

  constructor(private http: HttpClient) { }

  getClientes(): Observable<any> {
    const url = `${this.URLAPI}/getClientes`;
    return this.http.get<Cliente[]>(url);
  }
```

Los componentes cumplen la función de través de nuestra Api Rest, conseguir información y desplegarla de forma visual.



Componente chart:

Usaremos HighChart para mostrar estadísticas en nuestra pagina.

Coger información de la Api Rest.

```
salarioEmpleado() {
  this.apirestService.getEmpleados().subscribe(
    (result: any) => {
      this.lEmpleado = result.map((empleado: any) => {
        return new empleadoNormal(empleado.horario, empleado.dni, empleado.nombre, empleado.tlf, empleado.sueldo, empleado.departamento, empleado.jornada, empleado.tipoEmpleado);
      });

      const dataSeries = this.lEmpleado.map((x: empleadoNormal) => x.dni);
      const dataCategorias = this.lEmpleado.map((x: empleadoNormal) => x.sueldo);
      if(dataSeries!=undefined && dataCategorias !=undefined && this.chartOptions.series!=undefined && this.chartOptions.xAxis!=undefined) {
        this.chartOptions.series[0]["data"] = dataCategorias;
        this.chartOptions.xAxis["categories"] = dataSeries;
        this.chartOptions.title["text"] = "Sueldos";
        this.chartOptions.series["name"] = "Personas"
      Highcharts.chart("miGrafico01", this.chartOptions);
      }
    },
    error => console.log(error)
  );
}
```

HTML de nuestra gráfica.

```
<mat-toolbar>
  <span class="titulos">Comparativa</span>
</mat-toolbar>

<div style="text-align:center">
  <br>
  <h1>
    Sueldos empleados
  </h1>
</div>

<div id="miGrafico01">
  <highcharts-chart
    [Highcharts]="Highcharts"
    [options]="chartOptions"
    style="width: 100%; height: 400px; display: block;"
  ></highcharts-chart>
</div>
```

Login:

Recogida de datos desde la ApiRest.

```
constructor(
  public apirestService: ApirestService,
  private fb: FormBuilder,
  private router: Router,
  private toastr: ToastrService,
  private aRouter: ActivatedRoute
) {
  this.loginForm = this.fb.group({
    user: ['', Validators.required],
    password: ['', Validators.required],
  });
  this.user = this.aRouter.snapshot.paramMap.get('user');
  this.password = this.aRouter.snapshot.paramMap.get('password');
  console.log('Rellena el campo password');
  console.log(this.password);
}

ngOnInit(): void {}

login() {
  const log: iLogin = {
    user: this.loginForm.get('user')?.value,
    password: this.loginForm.get('password')?.value,
  };

  if (log.user !== null && log.password !== null) {
    this.apirestService.getLogin(log.user, log.password).subscribe((data) => {
      if(data !== null){
        this.router.navigate([
          'tablaEmpleados/loginForm/Managers',
        ]);
        this.toastr.success('Login correcto', 'Acceso permitido');
      }else{
        this.toastr.warning('Login incorrecto', 'Acceso denegado');
      }
    })
  }
}
```

Formulario para loguearse con comprobaciones de error.

```
<div class="container col-md-6">
    <div class="card" style="margin-top: 10%;">
        <div class="card-body">
            <form [formGroup]="loginForm">
                <div class="form-group">
                    <input type="text" formControlName="user" class="form-control" placeholder="Usuario">
                    <div class="text-danger"
                    *ngIf="loginForm.get('user')?.hasError('required') && loginForm.get('user')?.touched">
                        <span>Campo usuario <strong>obligatorio</strong></span>
                    </div>
                </div>
                <div class="form-group">
                    <input type="password" formControlName="password" class="form-control" placeholder="Contraseña">
                    <div class="text-danger"
                    *ngIf="loginForm.get('password')?.hasError('required') && loginForm.get('password')?.touched">
                        <span>Campo contraseña <strong>obligatorio</strong></span>
                    </div>
                </div>
                <button (click)="login()" type="submit" [disabled]="loginForm.invalid" class="btn btn-success">
                    Login
                </button>
            </form>
        </div>
    </div>
</div>
<a mat-raised-button color="primary" style="margin-left:35%;margin-right:35%;display:block;margin-top:5%;margin-bottom:5%" routerLink="/tablaEmpleados">Volver</a>
```

Componente de objeto:

Aquí tenemos el html que nos muestra la información de nuestro objeto.

```html
<button mat-raised-button color="primary" style="margin-left:45%;margin-right:35%;display:block;margin-top:5%;margin-bottom:5%" routerLink="formJoya">Agregar joya</button>
<h1 style="font-size: 35px;margin-bottom: 2%;">Joya:</h1>
<table class="table table-striped">
    <thead>
        <tr>
            <th>Codigo barra:</th>
            <th>Tipo producto:</th>
            <th>Marca:</th>
            <th>Descripcion:</th>
            <th>Material:</th>
            <th>Piedra preciosa:</th>
            <th>Peso:</th>
            <th>Precio:</th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let joyas of joya">
            <td>{{ joyas.cBarra }}</td>
            <td>{{ joyas.tipoProducto }}</td>
            <td>{{ joyas.marca }}</td>
            <td>{{ joyas.descripcion }}</td>
            <td>{{ joyas.material }}</td>
            <td>{{ joyas.piedraPreciosa }}</td>
            <td>{{ joyas.peso }}</td>
            <td>{{ joyas.precio }}</td>
            <td>
                <button [routerLink]="['formJoya', joyas.cBarra]" class="btn btn-secondary btn-sm">
                    <i class="material-icons">edit</i>
                </button>
                <button class="btn btn-danger btn-sm" (click)="deleteProducto(joyas.cBarra)">
                    <i class="material-icons">delete</i>
                </button>
            </td>
        </tr>
    </tbody>
</table>
<router-outlet></router-outlet>
```

Este contiene funciones para hacer un get y listar el producto y un delete del mismo.

```typescript
export class JoyaComponent implements OnInit {
  joya: Joyas[] = []

  constructor(public apirestService: ApirestService, private toastr: ToastrService) { }

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

  getJoyas(){
    this.apirestService.getJoyas().subscribe(joya => {
      this.joya = joya;
    })
  }

  deleteProducto(cBarra: string){
    if(confirm('¿Seguro que quieres borrar el producto?')) {
    this.apirestService.deleteProducto(cBarra).subscribe()
  }
    this.joya = this.joya.filter((encontrar) => encontrar.cBarra !== cBarra)
  }
```