



# DOCUMENTACIÓN PROYECTO 2 EVALUACIÓN

---

## INDICE

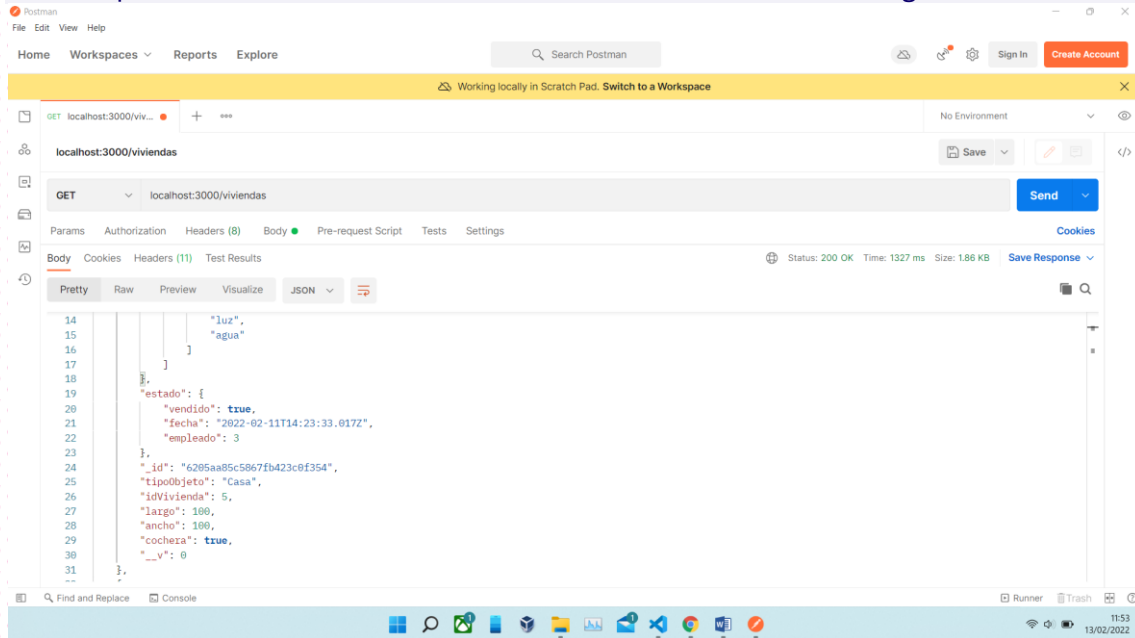
1. RUTAS
2. COLECCIONES
3. CLASES
4. VISTA DEL USUARIO FINAL
5. ESTRUCTURA ANGULAR
6. MÓDULOS Y ROUTING
7. POLIMORFISMO Y HERENCIAS



## 1. RUTAS

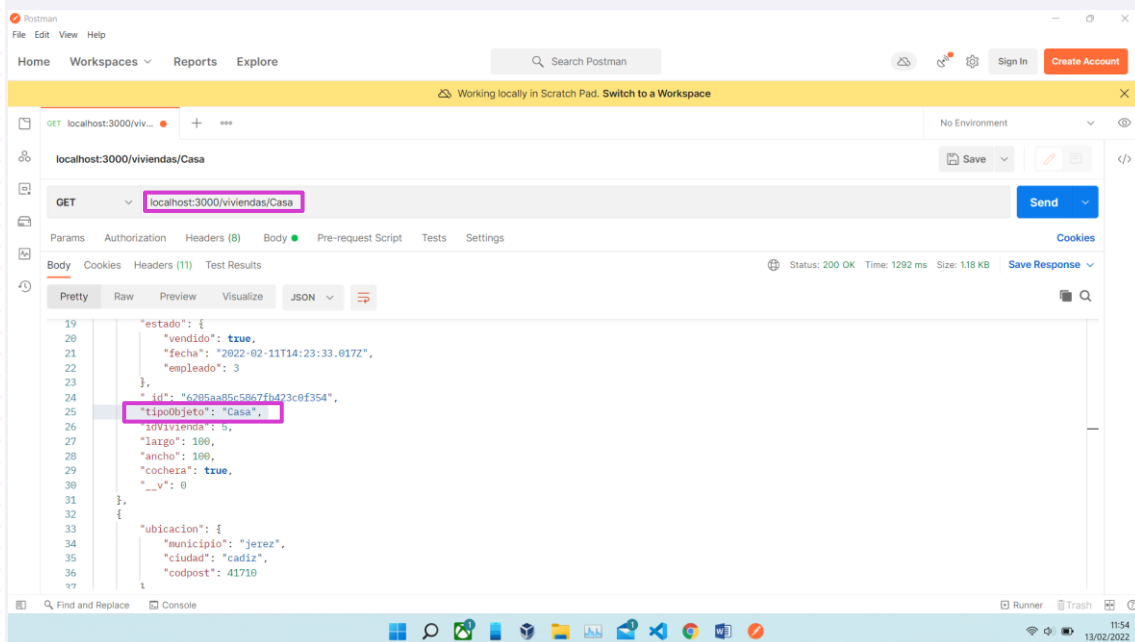
```
this._router.get('/viviendas', this.getViviendas)
```

Se usa para obtener todas las viviendas almacenadas en MongoDB



```
this._router.get('/viviendas/:type', this.getTypes)
```

Se usa para obtener todas las viviendas filtradas por el tipo de objeto (Casa o Chalet)





`this._router.get('/vivienda/:idVivienda', this.getVivienda)`

Se usa para obtener una vivienda con el ID dado

The screenshot shows the Postman application interface. At the top, a yellow banner indicates "Working locally in Scratch Pad. Switch to a Workspace". The main area displays a GET request to `localhost:3000/vivienda/5`, which has been successfully executed. The response is a JSON object with the following structure:

```
{
  "agua": true,
  "estado": {
    "vendido": true,
    "fecha": "2022-02-11T14:23:33.017Z",
    "empleado": 3
  },
  "id": "6205aa85c5867fb423c8f354",
  "idVivienda": 5,
  "largo": 100,
  "ancho": 100,
  "cocheza": true,
  "v": 0
}
```

The `idVivienda: 5` field in the response is highlighted with a red box. The status bar at the bottom shows the request was successful (Status: 200 OK) and the response size is 872 B.



```
this._router.get('/empleados', this.getEmpleados)
```

Se usa para obtener todos los empleados de MongoDB

Postman interface showing a GET request to `localhost:3000/empleados`. The response is a JSON array of two employee objects.

```
1 {
2   "_id": "6287cc3fdd7e17e9f7a713ad",
3   "idEmpleado": 2,
4   "nombre": "Luis",
5   "email": "luis@gmail.com",
6   "telefono": "23123",
7   "sueldo base": 12231,
8   "comision venta": 21131,
9   "numeroVentas": 0
10 },
11 {
12   "_id": "6287e8e6377167523e78cc2c",
13   "idEmpleado": 1,
14   "nombre": "Lucia",
15   "email": "lucia@gmail.com",
16   "telefono": "11111",
17   "sueldo base": 2800,
18   "comision venta": 20,
19   "numeroVentas": 0
20 }
```

```
this._router.get('/empleado/:idEmpleado', this.getEmpleado)
```

Se usa para obtener un empleado dada su ID

Postman interface showing a GET request to `localhost:3000/empleado/1`. The response is a JSON object for the employee with `idEmpleado: 1`.

```
1 {
2   "_id": "6287e8e6377167523e78cc2c",
3   "idEmpleado": 1,
4   "nombre": "Lucia",
5   "email": "lucia@gmail.com",
6   "telefono": "11111",
7   "sueldo base": 2800,
8   "comision venta": 20,
9   "numeroVentas": 0
10 }
```



```
this._router.post('/vivienda', this.postVivienda)
```

Se usa para añadir viviendas a MongoDB tanto casas como chalets

Postman interface showing a POST request to `localhost:3000/vivienda`. The request body is a JSON object:

```
{  "tipoObjeto": "Casa",  "idVivienda": 10,  "largo": 12,  "ancho": 10,  "municipio": "utreza",  "ciudad": "sevilla",  "codpost": 41710,  "habitaciones": 4,  "baños": 3,  "ascensor": false,  "enunamiento": false}
```

The response status is 200 OK, Time: 1389 ms, Size: 933 B.

```
this._router.post('/empleados', this.postEmpleado)
```

Se usa para añadir un nuevo empleado a MongoDB

Postman interface showing a GET request to `localhost:3000/empleados`. The response is a 200 OK status, Time: 1241 ms, Size: 116 KB.

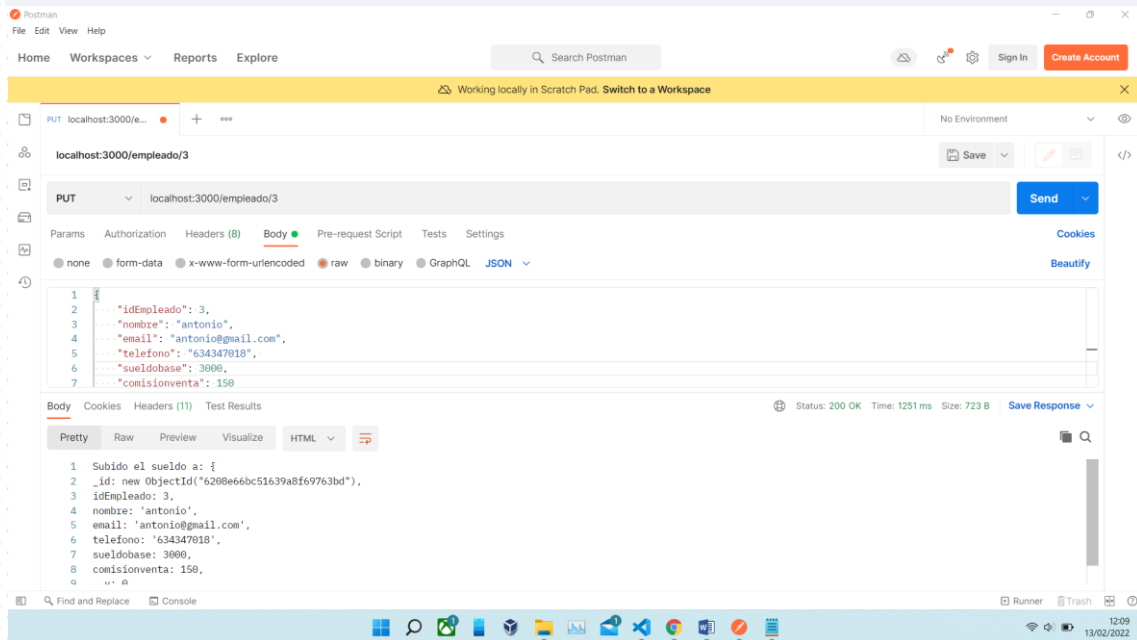
The response body is a JSON object:

```
{  "_id": "6208e631c51639a8f69763bb",  "nombre": "antonio",  "email": "antonio@gmail.com",  "telefono": "634347018",  "sueldobase": 1200,  "comisionventa": 150}
```



```
this._router.put('/empleado/:idEmpleado', this.modificarEmpleado)
```

Se usa para modificar empleados, en este caso se le modifica el sueldo base de 1200 a 300



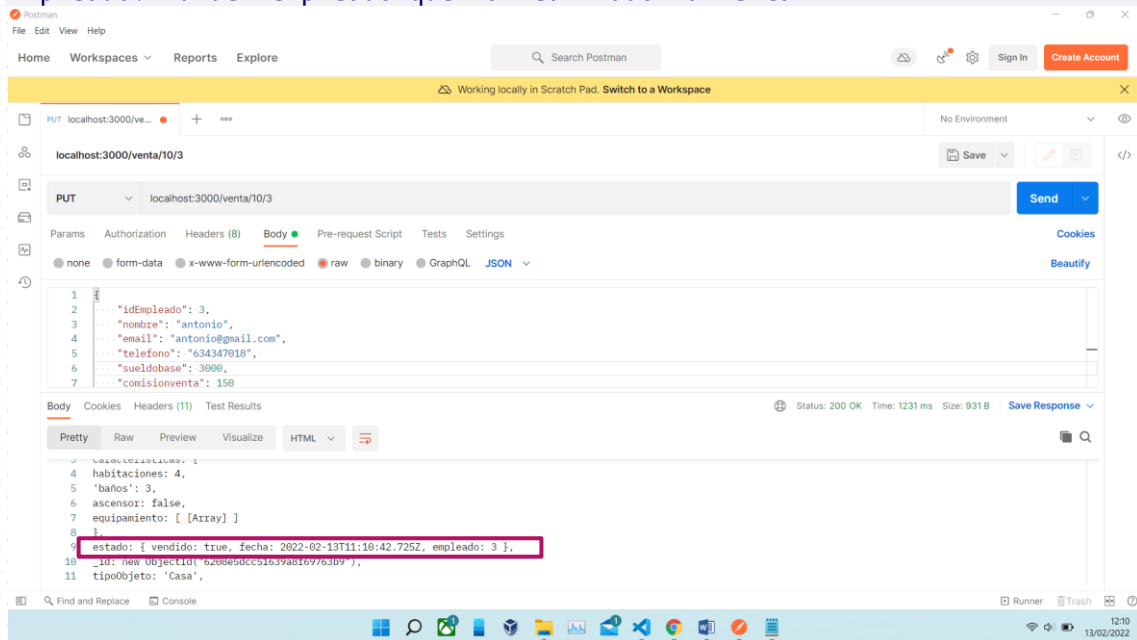
```
this._router.put('/venta/:idVivienda/:idEmpleado', this.updateEstado)
```

Esta función es la más interesante, se usa para realizar una venta. Añadiremos al campo estado de la colección vivienda los siguientes valores:

Vendido: pasa de ser false a true

Fecha: fecha y hora en la que se realiza la venta

Empleado: id del empleado que ha realizado la venta





```
1 Vivienda vendida {
2   ubicacion: { municipio: 'utrera', ciudad: 'sevilla', codpost: 41710 },
3   características: {
4     habitaciones: 4,
5     'baños': 3,
6     ascensor: false,
7     equipamiento: [ [Array] ]
8   },
9   estado: { vendido: true, fecha: 2022-02-13T11:10:42.725Z, empleado: 3 },
10  _id: new ObjectId("6208e5dcc51639a8f69763b9"),
11  tipoObjeto: 'Casa',
12  idVivienda: 10,
13  largo: 12,
14  ancho: 10,
15  cochera: true,
16  __v: 0
17 }
```

`this._router.delete('/deleteEmpleado/:idEmpleado', this.deleteEmpleado)`

Se usa para borrar un empleado

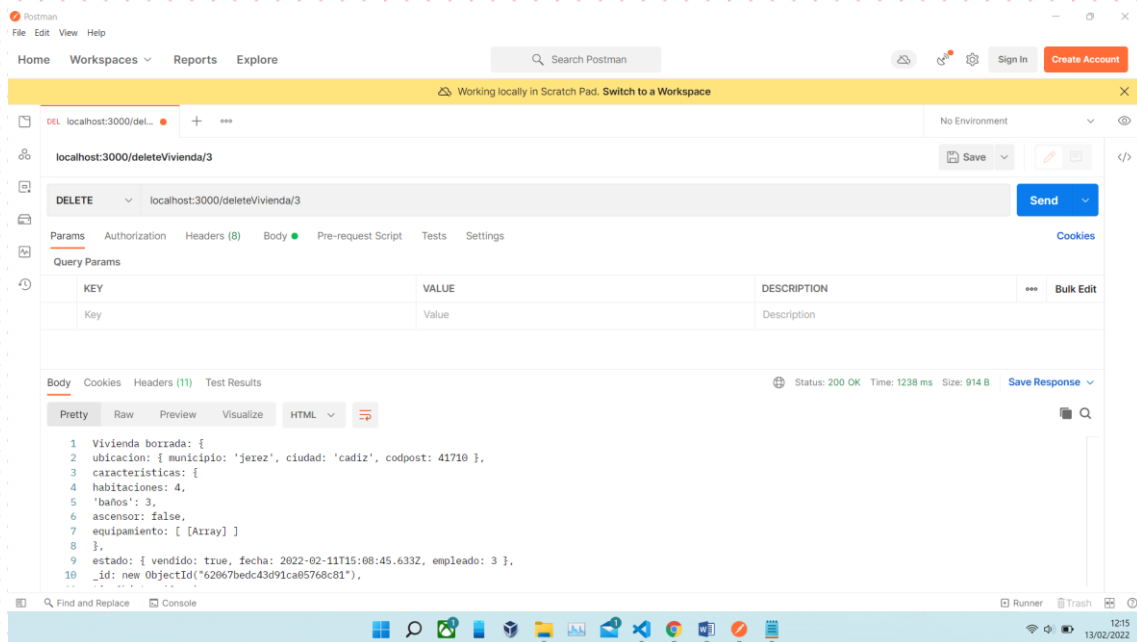
The screenshot shows the Postman interface with a DELETE request to `localhost:3000/deleteEmpleado/1`. The response is a JSON object with the following structure:

```
1 {
2   "_id": "6208e5dcc51639a8f69763b9",
3   "idEmpleado": 1,
4   "nombre": "Lucia",
5   "email": "lucia@gmail.com",
6   "telefono": "11111",
7   "sueldoBase": 2000,
8   "comisionVenta": 20,
9   "__v": 0
10 }
```

The status of the response is 200 OK, with a time of 1322 ms and a size of 688 B.



`this._router.delete('/deleteVivienda/:idVivienda', this.deleteVivienda)`  
Se usa para eliminar una vivienda

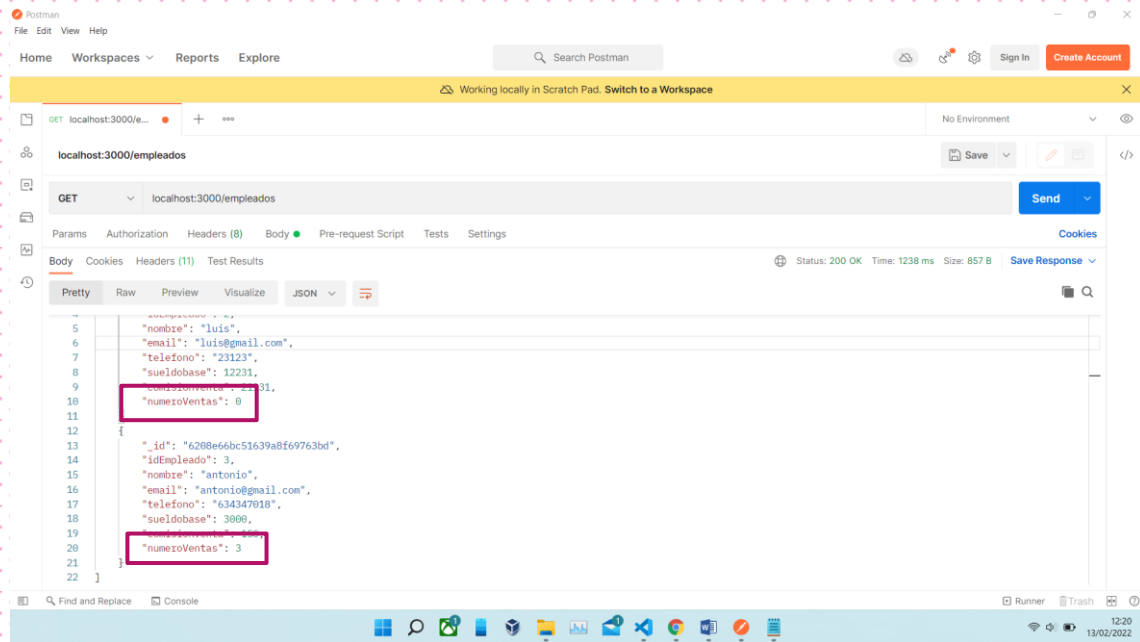


Por último, remarcar esta función:

`this._router.get('/empleados', this.getEmpleado)`

```
private getEmpleados = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then(async () => {
    const query = await modeloEmpleado.aggregate([
      {
        '$lookup': {
          'from': 'viviendas',
          'localField': 'idEmpleado',
          'foreignField': 'estado.empleado',
          'as': 'numeroVentas'
        }
      }, {
        '$project': {
          'idEmpleado': 1,
          'nombre': 1,
          'email': 1,
          'telefono': 1,
          'sueldobase': 1,
          'comisionventa': 1,
          'numeroVentas': {
            '$size': '$numeroVentas'
          }
        }
      }
    ])
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })
  await db.desconectarBD()
}
```





Añade un nuevo campo al empleado, que nos muestra cuantas viviendas ha vendido.



## 2. COLECCIONES

En primer lugar, veremos la colección viviendas.

En ella hacemos uso de las validaciones de mongoose:

Required: agrega un validador requerido para el campo

Unique: Es un ayudante para construir índices únicos de MongoDB

Lowercase: convierte las cadenas de string en minúscula.

Default: establece un valor predeterminado para el campo

Mensajes de errores personalizados.

```
const viviendaSchema = new Schema({
  tipoObjeto: {
    type: String,
    required: 'Que no se te olvide, a ver como lo identificas luego'
  },
  idVivienda: {
    type: Number,
    unique: true,
  },
  largo: Number,
  ancho: Number,
  ubicacion: {
    municipio: String,
    ciudad: {
      type: String,
      lowercase: true,
      required: [true, 'Se te olvida la ciudad!']
    },
    codpost: Number,
  },
  características: {
    habitaciones: Number,
    baños: Number,
    ascensor: Boolean,
    equipamiento: Array
  },
  estado: {
    vendido: {
      type: Boolean,
      default: false
    },
    fecha: Date,
    empleado: Number,
  },
  piscina: Boolean,
```



```
    largojardin: Number,  
    anchojardin: Number,  
    cochera: Boolean  
  })
```

Para continuar veremos la colección de los empleados:

```
const empleadoSchema = new Schema({  
  idEmpleado: {  
    type: Number,  
    unique: true  
  },  
  nombre: String,  
  email: {  
    type: String,  
    required: true  
  },  
  telefono: {  
    type: String,  
    required: true  
  },  
  sueldobase: {  
    type: Number,  
    default: 950  
  },  
  comisionventa: Number,  
})
```



### 3. CLASES

#### 3.1 CLASE VIVIENDA (PADRE)

La clase Vivienda tiene algo particular, para hacerlo más "profesional" he recurrido a las interfaces para ahorrar líneas de código, las podemos ver en los campos Ubicación y Características.

```
export abstract class Vivienda {  
  private _idVivienda: number;  
  private _largo: number;  
  private _ancho: number;  
  private _ubicacion: Ubicacion;  
  private _caracteristicas: Caracteristicas;  
  private _estado: {  
    vendido: boolean;  
    fecha: Date | null;  
    empleado: number  
  };  
};
```

Las interfaces están formuladas en otro directorio llamados interfaces.ts y serían de tal forma:

```
export interface Ubicacion {  
  municipio: string;  
  ciudad: string;  
  codpost: number;  
}  
  
export interface Caracteristicas {  
  habitaciones: number;  
  baños: number;  
  ascensor: boolean;  
  equipamiento: Array<string>;  
}
```

Los métodos de la clase vivienda son:



```
ubi() {
  return (
    'Municipio: ' +
    this._ubicacion.municipio +
    ', Ciudad: ' +
    this._ubicacion.ciudad +
    ',Codigo postal: ' +
    this._ubicacion.codpost
  );
}

est() {
  return (
    'Vendido: ' + this._estado.vendido +
    ', Fecha: ' + this._estado.fecha +
    ', Empleado: ' + this._estado.empleado
  );
}

carac() {
  return (
    '\nHabitaciones: ' +
    this._caracteristicas.habitaciones +
    ', Baños: ' +
    this._caracteristicas.baños +
    ', ¿Tiene ascensor?: ' +
    this._caracteristicas.ascensor +
    '\nEquipamiento: ' +
    this._caracteristicas.equipamiento
  );
}
```

```
//Calcular el precio de la vivienda, según la ubicación el precio del m2 varía.
preciom2() {
  let preciom2: any;
  if (this.ubicacion.ciudad == 'sevilla') {
    preciom2 = 1386 * this.superficie();
  } else if (this.ubicacion.ciudad == 'almeria') {
    preciom2 = 1088 * this.superficie();
  } else if (this.ubicacion.ciudad == 'jaen') {
    preciom2 = 823 * this.superficie();
  } else if (this.ubicacion.ciudad == 'malaga') {
    preciom2 = 2442 * this.superficie();
  } else if (this.ubicacion.ciudad == 'granada') {
    preciom2 = 1375 * this.superficie();
  } else if (this.ubicacion.ciudad == 'cadiz') {
    preciom2 = 1555 * this.superficie();
  } else if (this.ubicacion.ciudad == 'cordoba') {
    preciom2 = 1220 * this.superficie();
  } else if (this.ubicacion.ciudad == 'huelva') {
    preciom2 = 1253 * this.superficie();
  }
  return Math.round(preciom2);
}

superficie(): number {
  let superficie: number;
  superficie = this._ancho * this.largo;
  return superficie;
}
```



```
todo() {  
  return `Superficie: ${this.superficie()},  
  Precio: ${this.preciom2()},  
  Ubicación: [ ${this.ubi()} ],  
  Características: [ ${this.carac()} ],  
  Estado: [ ${this.est()} ]`;  
}
```

## 3.2 CLASE CHALET Y CASA (HIJAS)

```
export class Chalet extends Vivienda {  
  private _piscina: boolean;  
  private _largojardin: number;  
  private _anchojardin: number;
```

```
export class Casa extends Vivienda {  
  private _cochera: boolean
```

Polimorfismo en la clase Casa:

```
preciom2() {  
  let preciom2 = super.preciom2();  
  if (this._cochera == true) {  
    preciom2 += 1.000;  
  }  
  return Math.round(preciom2)  
}
```

```
todo() {  
  return super.todo() + "Dispone de cochera: " + this._cochera  
}
```

Polimorfismo en la clase Chalet:

```
preciom2() {  
  let preciom2 = super.preciom2();  
  let preciojardin = this.m2jardin();  
  preciom2 = preciom2 + preciojardin;  
  if (this._piscina == true) {  
    preciom2 += 200;  
  }  
  return Math.round(preciom2);  
}
```

```
todo() {  
  let resultado: string;  
  resultado = `${super.todo()}, ¿Tiene piscina?: ${  
    this._piscina  
  } , Superficie del jardin(m2): ${this.sjardin()}, `;  
  return resultado;  
}
```











## 4. VISTA DEL USUARIO FINAL

Hablaremos en primer lugar del menú CRUD (CREATE, READ, UPDATE AND DELETE).

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Lola	lolita@hotmail.com	662137012	\$1,400.00	\$70.00	1	 

## AGREGAR UN EMPLEADO

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART





### CREAR EMPLEADO

Agregar

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Lola	lolita@hotmail.com	662137012	\$1,400.00	\$70.00	1	 
5	pepe	pepe@outlook.es	662795913	\$1,000.00	\$200.00	0	 



Si dejamos los campos vacíos nos saltará un error de validación:

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### CREAR EMPLEADO

Pro favor, ingrese todos los campos

ID

Nombre

Email Telefono

Sueldo Base Comision Venta

Agregar

¿Y como se consigue esto?

Pues bien, hay dos partes imprescindibles:

La primera en el documento HTML:

Añadimos la directiva \*ngIf, es decir se activará si...

Submitted: significa que hayamos dado click al boton de submit

crearEmpleado.invalid: significa que el formulario de crear empleado tiene que ser invalido

```
<div class="container">
  <div class="row">
    <div class="col-lg-6 offset-lg-3">
      <div class="card text-center">
        <div class="card-body">
          <h3>{{titulo}}</h3>
          <span *ngIf="submitted && crearEmpleado.invalid" class="badge badge-danger">Pro favor, ingrese todos los campos</span>
          <form [formGroup]="crearEmpleado" (ngSubmit)="agregarEmpleado()">
            <div class="row">
              <div class="col">
                <input type="number" formControlName="idEmpleado" class="mt-2 form-control form-control-lg" placeholder="ID">
              </div>
            </div>
            <div class="row">
              <div class="col">
                <input type="text" formControlName="nombre" class="form-control form-control-lg mt-3" placeholder="Nombre">
              </div>
            </div>
            <div class="row">
              <div class="col">
                <input type="text" formControlName="email" class="form-control form-control-lg mt-3" placeholder="Email">
              </div>
              <div class="col">
                <input type="text" formControlName="telefono" class="form-control form-control-lg mt-3" placeholder="Telefono">
              </div>
            </div>
            <div class="row">
              <div class="col">
                <input type="number" formControlName="sueldobase" class="form-control form-control-lg mt-3" placeholder="Sueldo Base">
              </div>
              <div class="col">
                <input type="number" formControlName="comisionventa" class="form-control form-control-lg mt-3" placeholder="Comision Venta">
              </div>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```





¿Qué quiere decir que el formulario sea invalido?

Pues significa que el formulario no cumple estos requisitos que se encuentran en el archivo ts del componente:

```
export class CrearEmpleadoComponent implements OnInit {  
  crearEmpleado: FormGroup;  
  submitted = false;  
  id: string | null  
  titulo = 'CREAR EMPLEADO'  
  constructor(  
    private fb: FormBuilder,  
    private empleadoService: EmpleadoService,  
    private activeRoute: ActivatedRoute) {  
    this.crearEmpleado = this.fb.group({  
      idEmpleado: ['', Validators.required],  
      nombre: ['', Validators.required],  
      email: ['', [Validators.required, Validators.email]],  
      telefono: ['', Validators.required],  
      sueldobase: ['', Validators.required],  
      comisionventa: ['', Validators.required]  
    })  
    this.id = this.activeRoute.snapshot.paramMap.get("idEmpleado")  
  }  
}
```













## BORRAR EMPLEADO

IDEALISTA

CASASCHALETSEMPLEADOSABOUT MEBUBBLE CHART

Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Lola	lolita@hotmail.com	662137012	\$1,400.00	\$70.00	1	 
5	pepe	pepe@outlook.es	662795913	\$1,000.00	\$200.00	0	 









Como vemos en la siguiente foto, pepe ha sido eliminado.

IDEALISTA

CASASCHALETSEMPLEADOSABOUT MEBUBBLE CHART

Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Lola	lolita@hotmail.com	662137012	\$1,400.00	\$70.00	1	 



## EDITAR EMPLEADO

Vamos a editar a Lola:

IDEALISTA

CASAS

CHALETs









EMPLEADOS

ABOUT ME

BUBBLE CHART

Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Lola	lolita@hotmail.com	662137012	\$1,400.00	\$70.00	1	 

Como podemos ver, al haber hecho click en LOLA su ID viene automaticamente al formulario de edici3n.

IDEALISTA

CASAS

CHALETs

EMPLEADOS

ABOUT ME

BUBBLE CHART

EDITAR EMPLEADO

4

Nombre

Email

Telefono

Sueldo Base

Comision Venta

Editar

Cambiaremos el email:

IDEALISTA

CASAS

CHALETs

EMPLEADOS

ABOUT ME

BUBBLE CHART

EDITAR EMPLEADO

4

lola

lolita@hotmail.es

662137012

1400

70

Editar



Como podemos ver, el email de LOLA ha cambiado.

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### Listado de Empleados

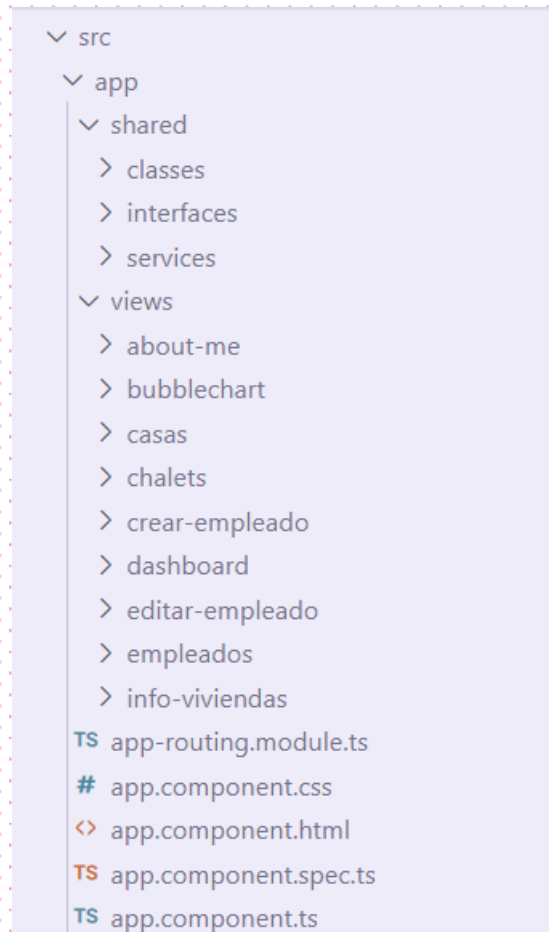
Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	lola	lolita@hotmail.es	662137012	\$1,400.00	\$70.00	1	 



## 5. ESTRUCTURA ANGULAR

El proyecto está estructurado de la siguiente forma:



Como podemos ver, dentro del directorio APP existen dos ramas,

SHARED: donde encontraremos toda la parte "lógica"

VIEWS: donde encontraremos toda la parte "vista"

Es una estructura pensada en el MVC. Separando las partes vistas de los "controladores".



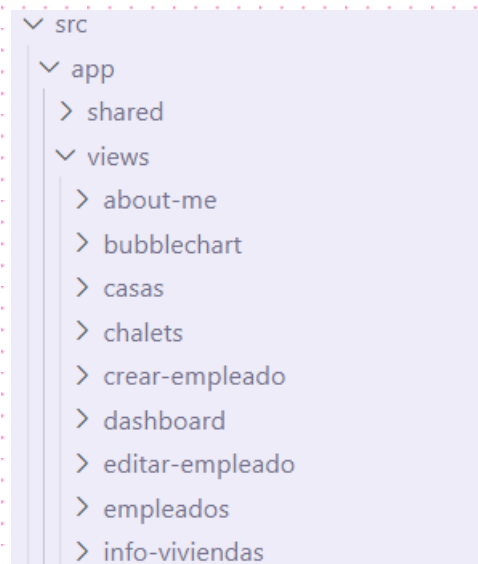
Dentro del directorio SHARED  
encontraremos:

- Clases
- Interfaces
- Servicios



Dentro del directorio VIEWS  
encontraremos:

- Componentes





## ¿Qué hace cada uno de los componentes?

### DASHBOARD

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART



### CASAS

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

#### Listado de Casas

ID	Superficie	Ubicacion	Habitaciones	Precio	
5	10000	Municipio: utrera, Ciudad: sevilla, Codigo postal: 41710	4	\$13,860,001.00	VENDER
10	120	Municipio: utrera, Ciudad: sevilla, Codigo postal: 41710	4	\$166,321.00	VENDER
1	120	Municipio: chipiona, Ciudad: cadiz, Codigo postal: 41710	4	\$186,600.00	VENDER
2	120	Municipio: fuengirola, Ciudad: malaga, Codigo postal: 23452	2	\$293,041.00	VENDER
3	400	Municipio: alcaracejos, Ciudad: cordoba, Codigo postal: 23452	2	\$488,001.00	VENDER

### CHALETS

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

#### Listado de Chalets

ID	Superficie	Ubicacion	Habitaciones	Precio	Superficie Jardin	
1	10000	Municipio: utrera, Ciudad: sevilla, Codigo postal: 41710	2	\$13,998,800.00	100	VENDER
2	10000	Municipio: utrera, Ciudad: sevilla, Codigo postal: 41710	2	\$13,998,600.00	100	VENDER
3	400	Municipio: baena, Ciudad: cordoba, Codigo postal: 23452	2	\$530,900.00	35	VENDER
4	400	Municipio: alcaudete, Ciudad: jaen, Codigo postal: 11111	4	\$395,240.00	80	VENDER
5	96	Municipio: almonte, Ciudad: huelva, Codigo postal: 12341	1	\$135,324.00	12	VENDER











## EMPLEADOS

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### Listado de Empleados

Agregar

Id	Nombre	Email	Telefono	Sueldo Base	Comision Venta	Numero Ventas	
2	Luis	luis@gmail.com	23123	\$12,231.00	\$21,131.00	0	 
3	antonio	antonio@gmail.com	634347018	\$3,000.00	\$150.00	2	 
1	Francisco	francisco@gmail.com	955486752	\$1,100.00	\$200.00	0	 
4	Iola	lolita@hotmail.es	662137012	\$1,400.00	\$70.00	1	 

## ABOUT ME

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART

### PROYECTO 2 EVALUACION ANGULAR NODEJS Y TYPESCRIPT

Este proyecto está hecho por Lucía Ramírez Monje

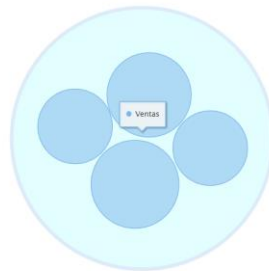
Proyecto orientado al sector inmobiliario

Contruido con



## BUBBLE CHART

IDEALISTA CASAS CHALETS EMPLEADOS ABOUT ME BUBBLE CHART







## 6. MODULOS Y ROUTING

### APP.MODULE.TS

```
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';
import { AppRoutingModule } from './app-routing.module';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
import { HighchartsChartModule } from 'highcharts-angular';

import { AppComponent } from './app.component';
import { EmpleadosComponent } from './views/empleados/empleados.component';
import { AboutMeComponent } from './views/about-me/about-me.component';
import { DashboardComponent } from './views/dashboard/dashboard.component';
import { InfoViviendasComponent } from './views/info-viviendas/info-viviendas.component';
import { CrearEmpleadoComponent } from './views/crear-empleado/crear-empleado.component';
import { EditarEmpleadoComponent } from './views/editar-empleado/editar-empleado.component';
import { ChaletsComponent } from './views/chalets/chalets.component';
import { CasasComponent } from './views/casas/casas.component';
import { BubblechartComponent } from './views/bubblechart/bubblechart.component';
```

### APP-ROUTING

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { AboutMeComponent } from './views/about-me/about-me.component';
import { BubblechartComponent } from './views/bubblechart/bubblechart.component';
import { CasasComponent } from './views/casas/casas.component';
import { ChaletsComponent } from './views/chalets/chalets.component';
import { CrearEmpleadoComponent } from './views/crear-empleado/crear-empleado.component';
import { DashboardComponent } from './views/dashboard/dashboard.component';
import { EditarEmpleadoComponent } from './views/editar-empleado/editar-empleado.component';
import { EmpleadosComponent } from './views/empleados/empleados.component';

const routes: Routes = [
  { path: 'listadocasa', component: CasasComponent },
  { path: 'aboutME', component: AboutMeComponent },
  { path: 'empleados', component: EmpleadosComponent },
  { path: 'dashboard', component: DashboardComponent },
  { path: 'listadochalet', component: ChaletsComponent },
  { path: 'grafico', component: BubblechartComponent },

  { path: 'crear-empleado', component: CrearEmpleadoComponent },
  { path: 'editar-empleado/:idEmpleado', component: EditarEmpleadoComponent },

  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
];
```



## 7. POLIMORFISMO Y HERENCIAS

```
<div class="container mt-4">
  <div class="card">
    <div class="card-body">
      <span class="h3">Listado de Casas</span>
      <table class="table table-striped mt-5">
        <thead>
          <tr>
            <th>ID</th>
            <th>Superficie</th>
            <th>Ubicacion</th>
            <th>Habitaciones</th>
            <th>Precio</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let vivienda of viviendas">
            <td>{{vivienda.idVivienda}}</td>
            <td>{{vivienda.superficie()}}</td>
            <td>{{vivienda.ubi()}}</td>
            <td>{{vivienda.caracteristicas.habitaciones}}</td>
            <td>{{vivienda.preciom2() | currency}}</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

```
<div class="container mt-4">
  <div class="card">
    <div class="card-body">
      <span class="h3">Listado de Chalets</span>
      <table class="table table-striped mt-5">
        <thead>
          <tr>
            <th>ID</th>
            <th>Superficie</th>
            <th>Ubicacion</th>
            <th>Habitaciones</th>
            <th>Precio</th>
            <th>Superficie Jardin</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let vivienda of viviendas">
            <td>{{vivienda.idVivienda}}</td>
            <td>{{vivienda.superficie()}}</td>
            <td>{{vivienda.ubi()}}</td>
            <td>{{vivienda.caracteristicas.habitaciones}}</td>
            <td>{{vivienda.preciom2() | currency}}</td>
            <td>{{vivienda.sjardin()}}</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```