# DOCUMENTACIÓN PROYECTO 2 EVALUACIÓN

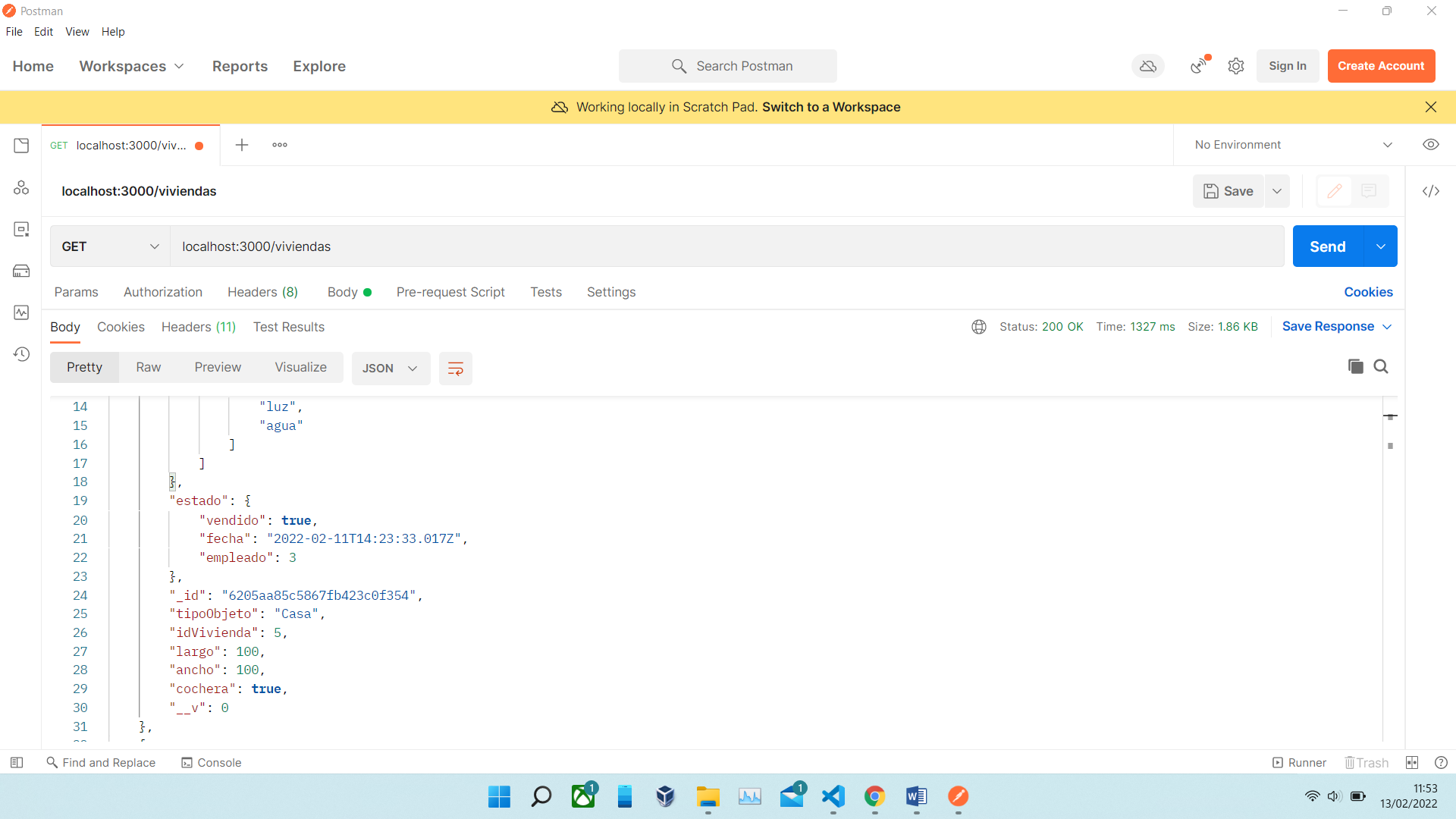
## INDICE

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

## RUTAS

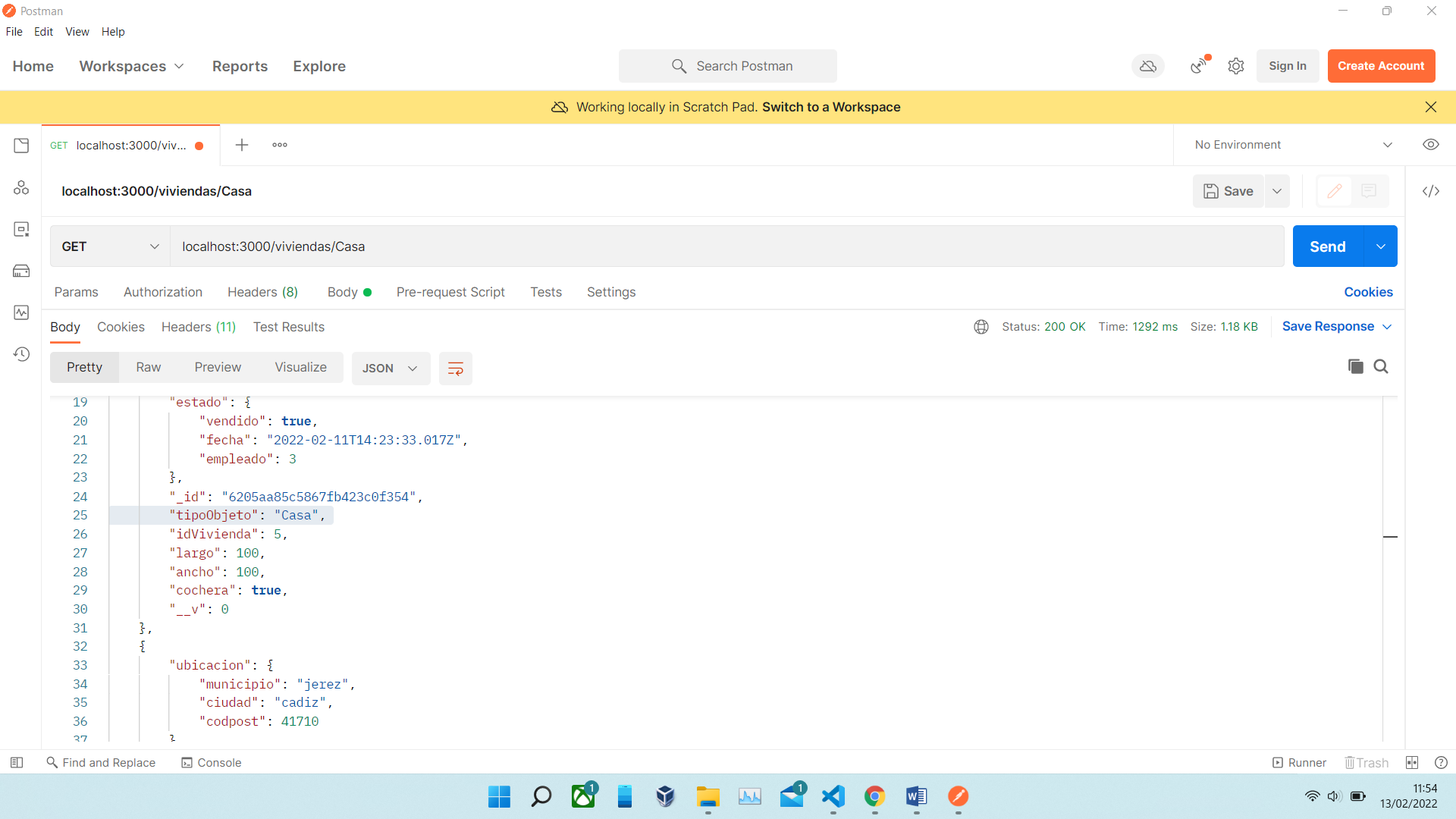
*this***.***\_router***.**get('/viviendas', *this***.***getViviendas*)

Se usa para obtener todas las viviendas almacenadas en MongoDB



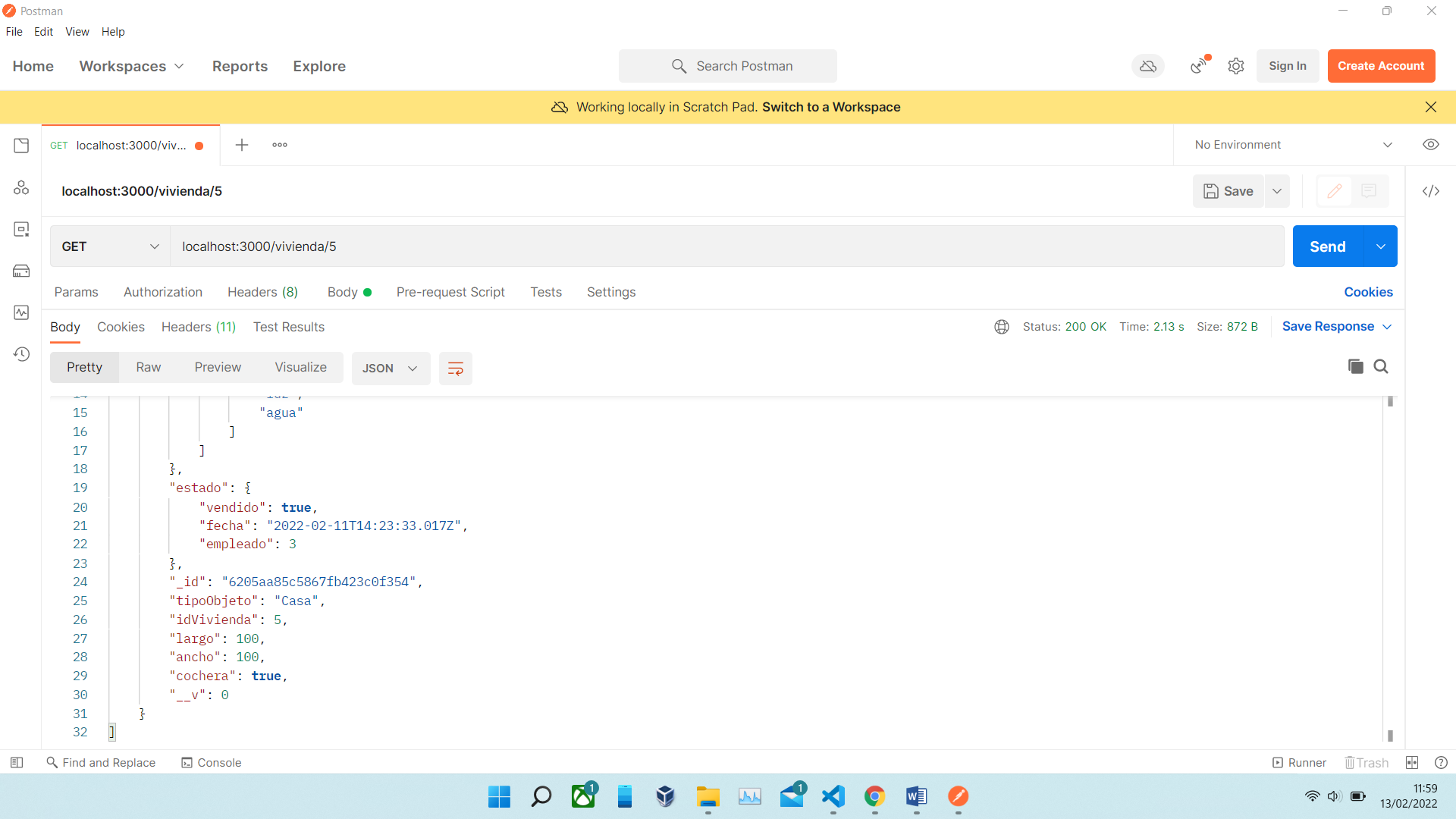
*this***.***\_router***.**get('/viviendas/:type', *this***.***getTypes*)

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



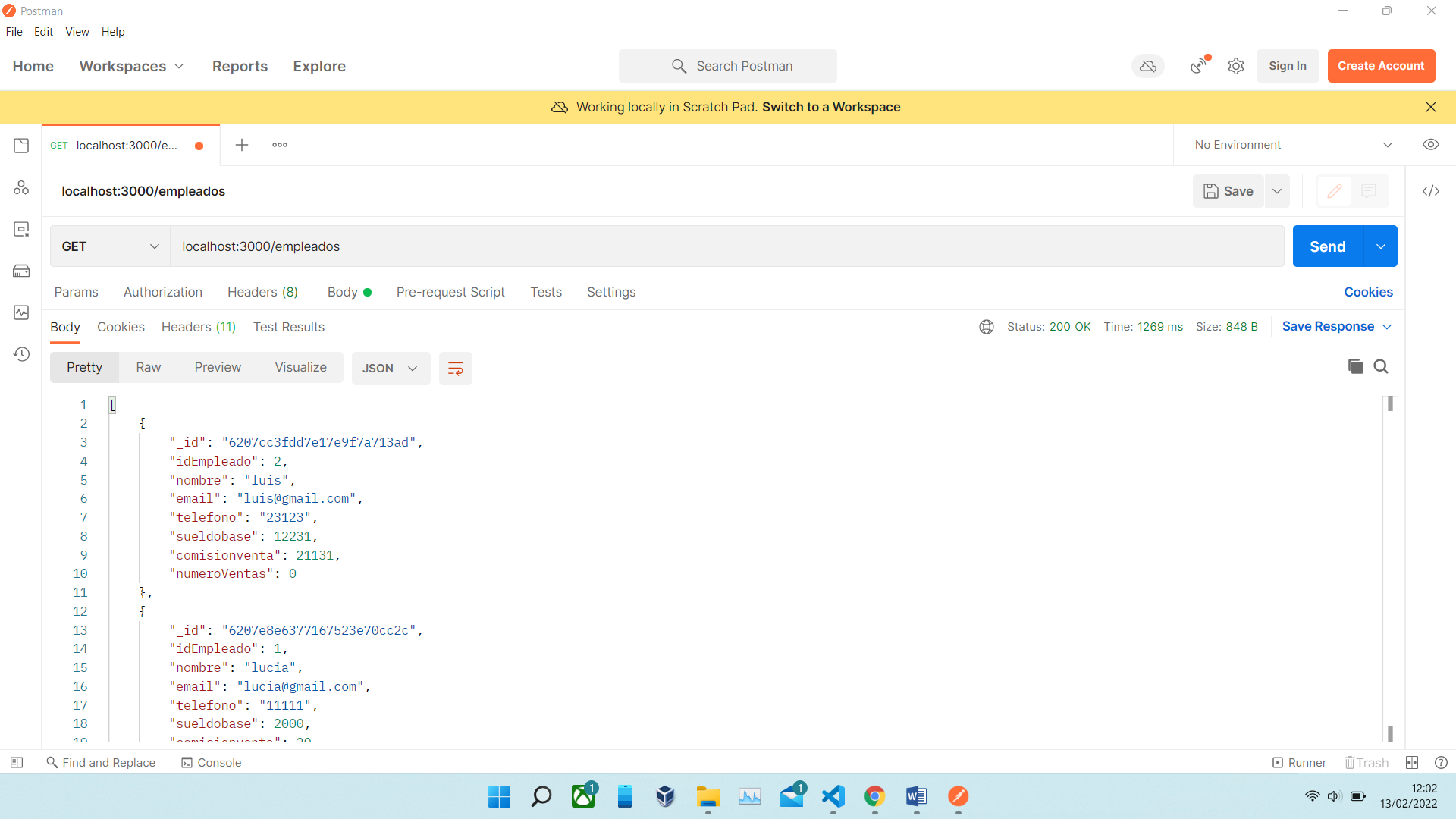
*this***.***\_router***.**get('/vivienda/:idVivienda', *this***.***getVivienda*)

Se usa para obtener una vivienda con el ID dado



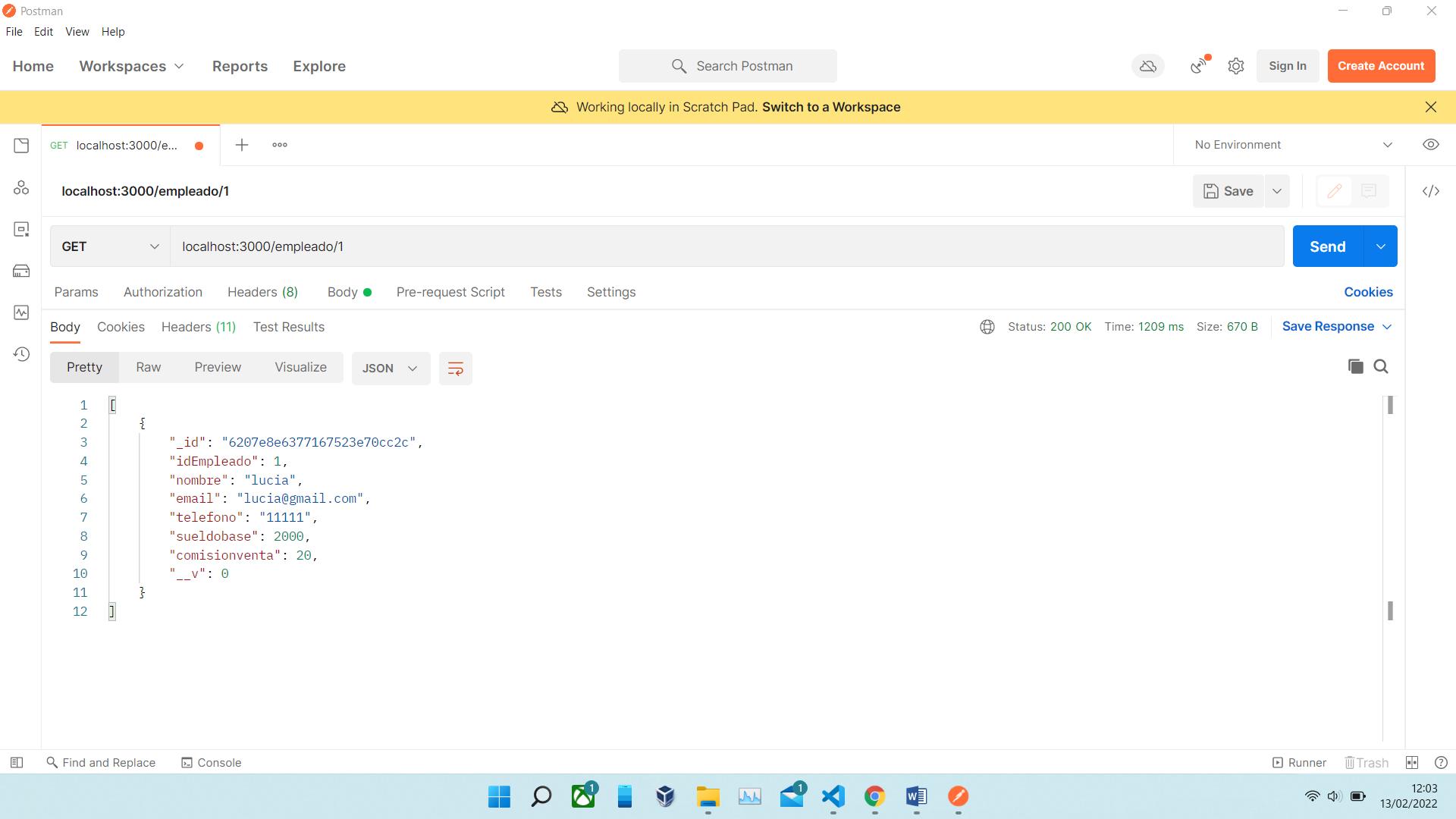
*this***.***\_router***.**get('/empleados', *this***.***getEmpleados*)

Se usa para obtener todos los empleados de MongoDB



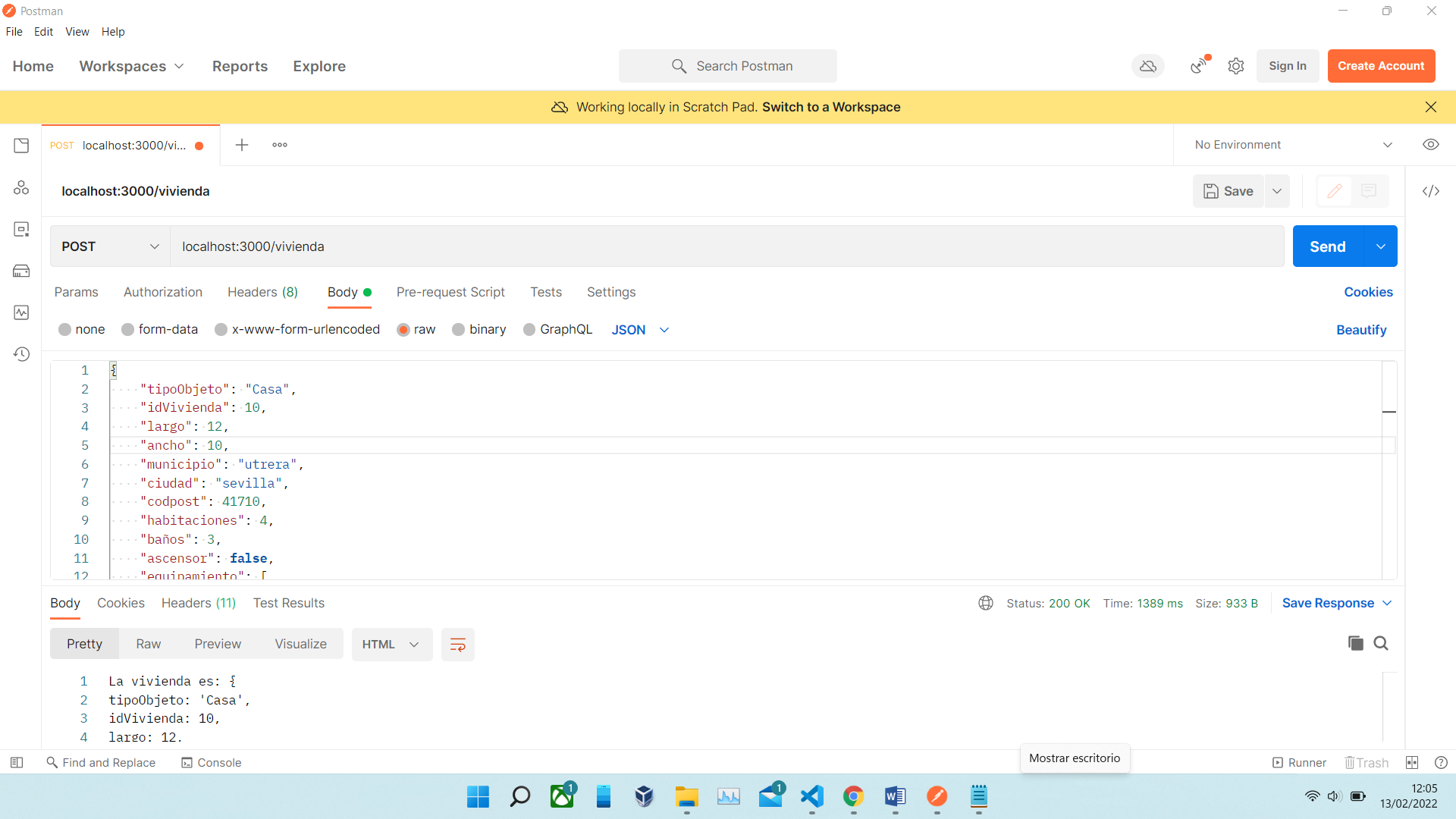
*this***.***\_router***.**get('/empleado/:idEmpleado', *this***.***getEmpleado*)

Se usa para obtener un empleado dada su ID



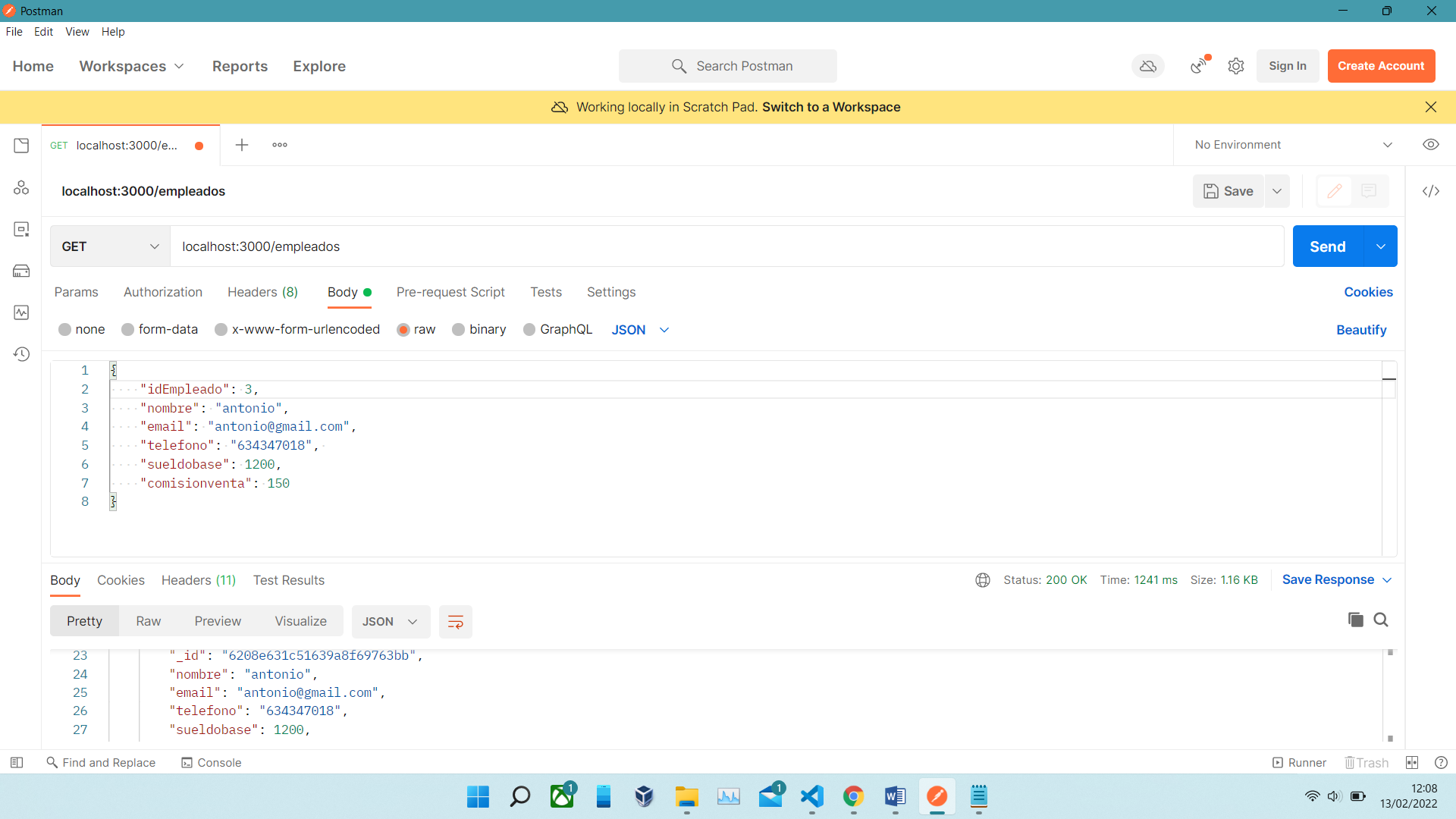
*this***.***\_router***.**post('/vivienda', *this***.***postVivienda*)

Se usa para añadir viviendas a MongoDB tanto casas como chalets



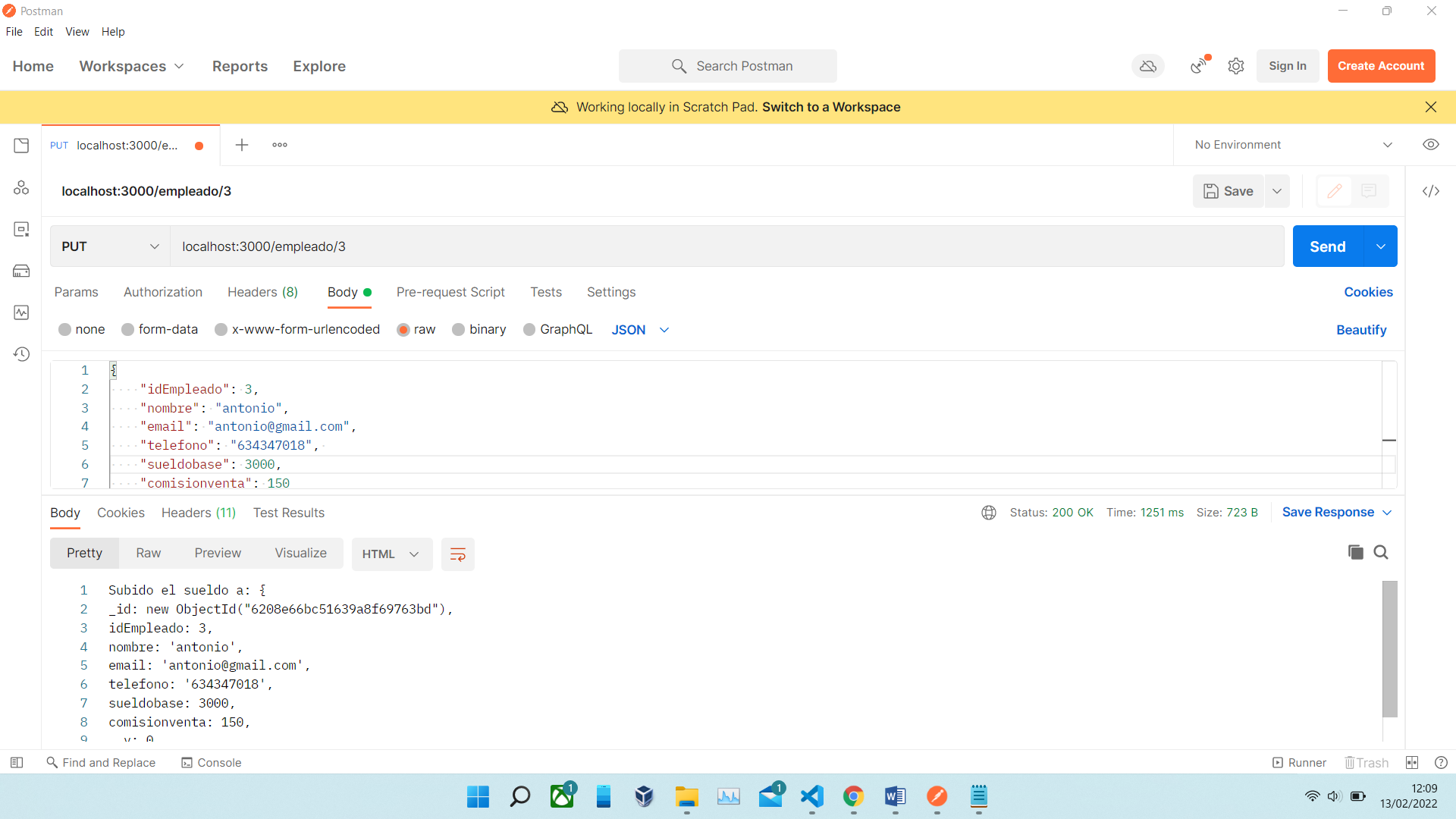
*this***.***\_router***.**post('/empleados', *this***.***postEmpleado*)

Se usa para añadir un nuevo empleado a MongoDB



*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*)

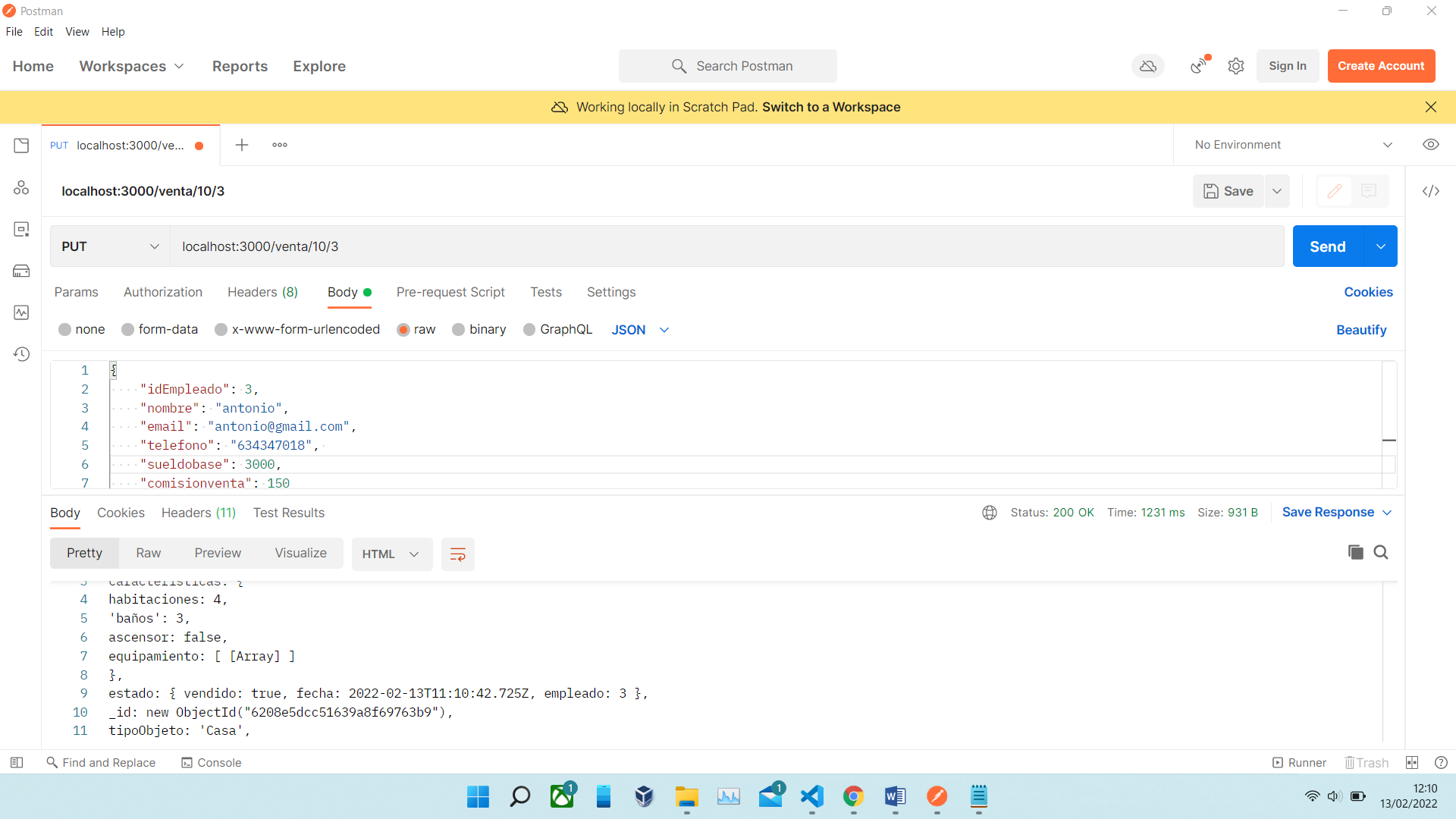
Esta función es la más interesante, se usa para realizar una venta.

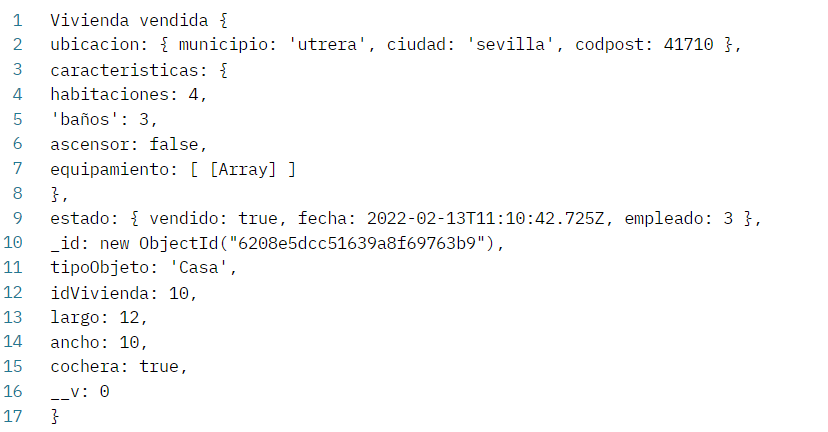
Añadiremos al campo estado de la colección vivienda los siguientes valores:

Vendido: pasa de ser false a true

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

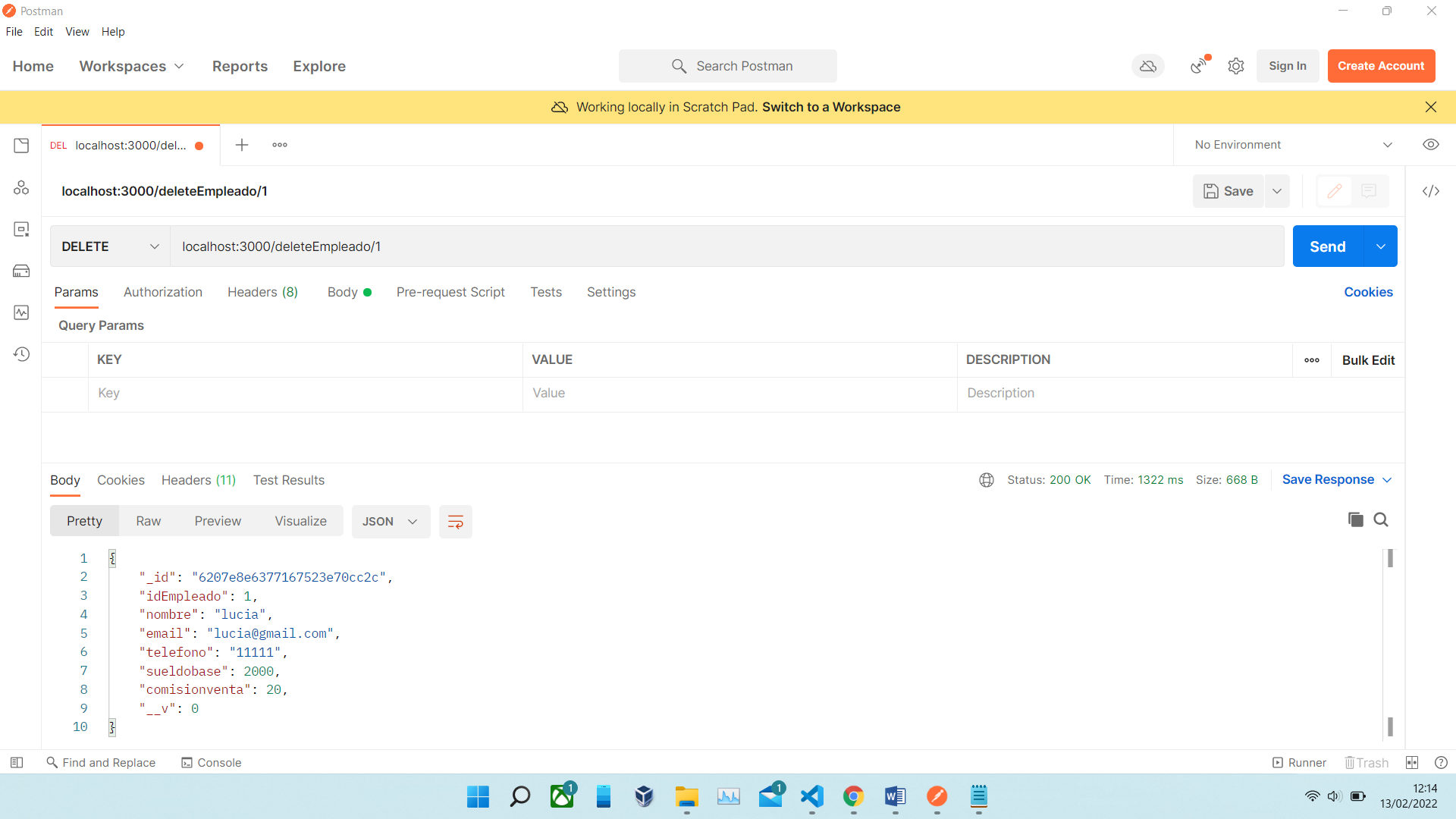
Empleado: id del empleado que ha realizado la venta





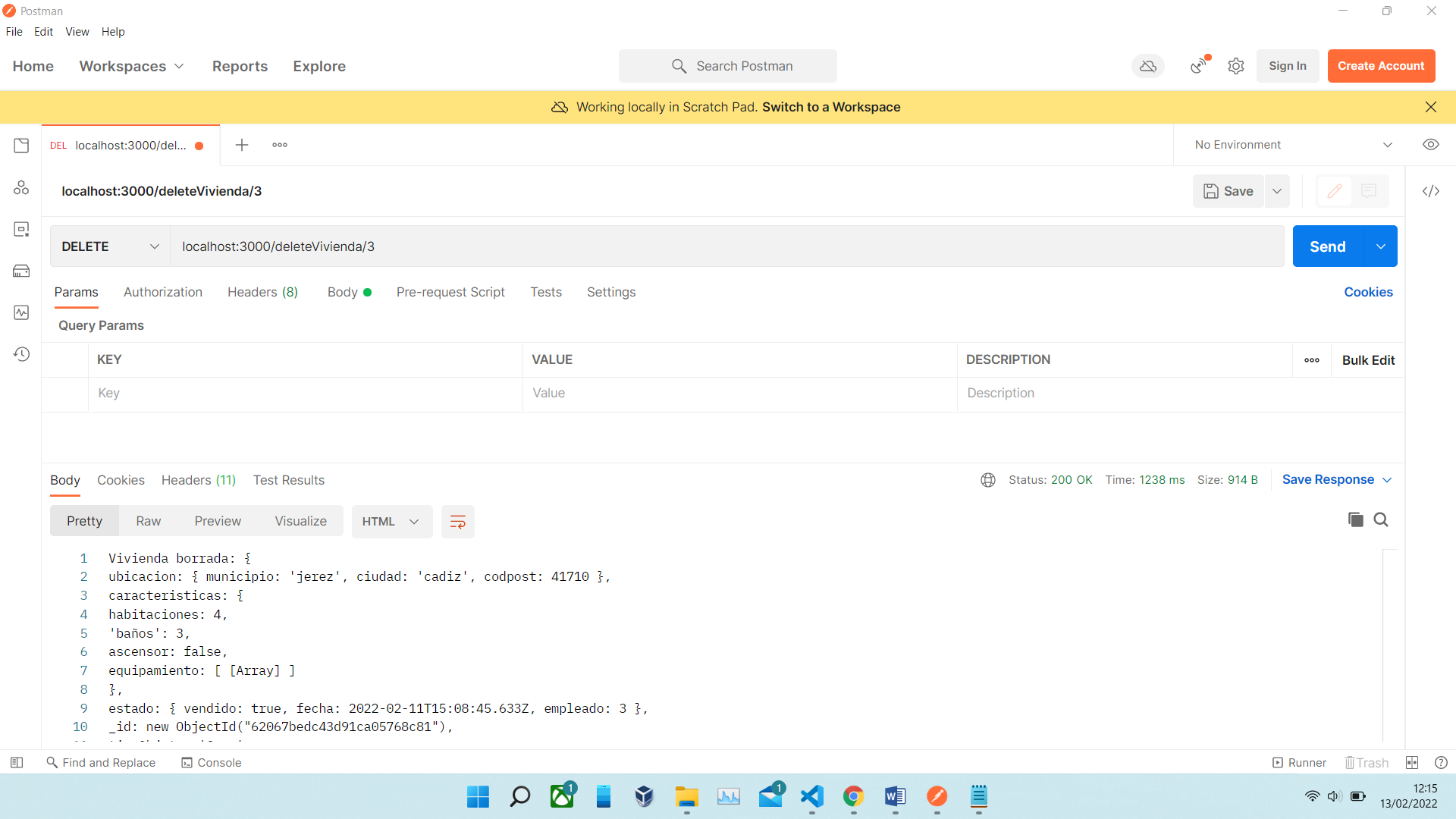
*this***.***\_router***.**delete('/deleteEmpleado/:idEmpleado', *this***.***deleteEmpleado*)

Se usa para borrar un empleado



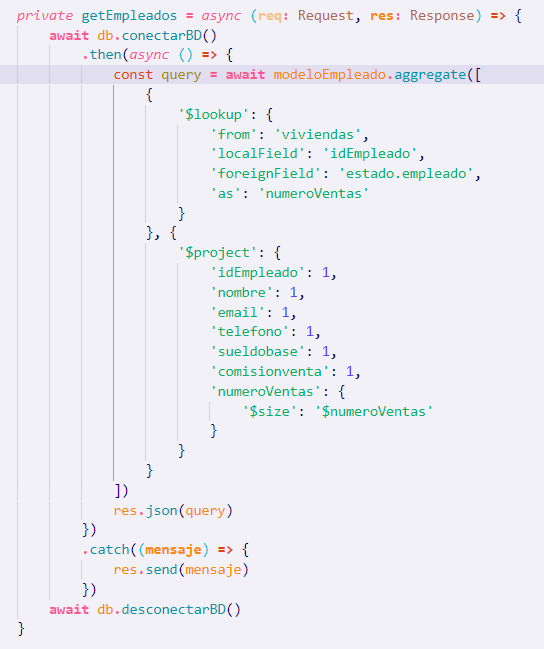
*this***.***\_router***.**delete('/deleteVivienda/:idVivienda', *this***.***deleteVivienda*)

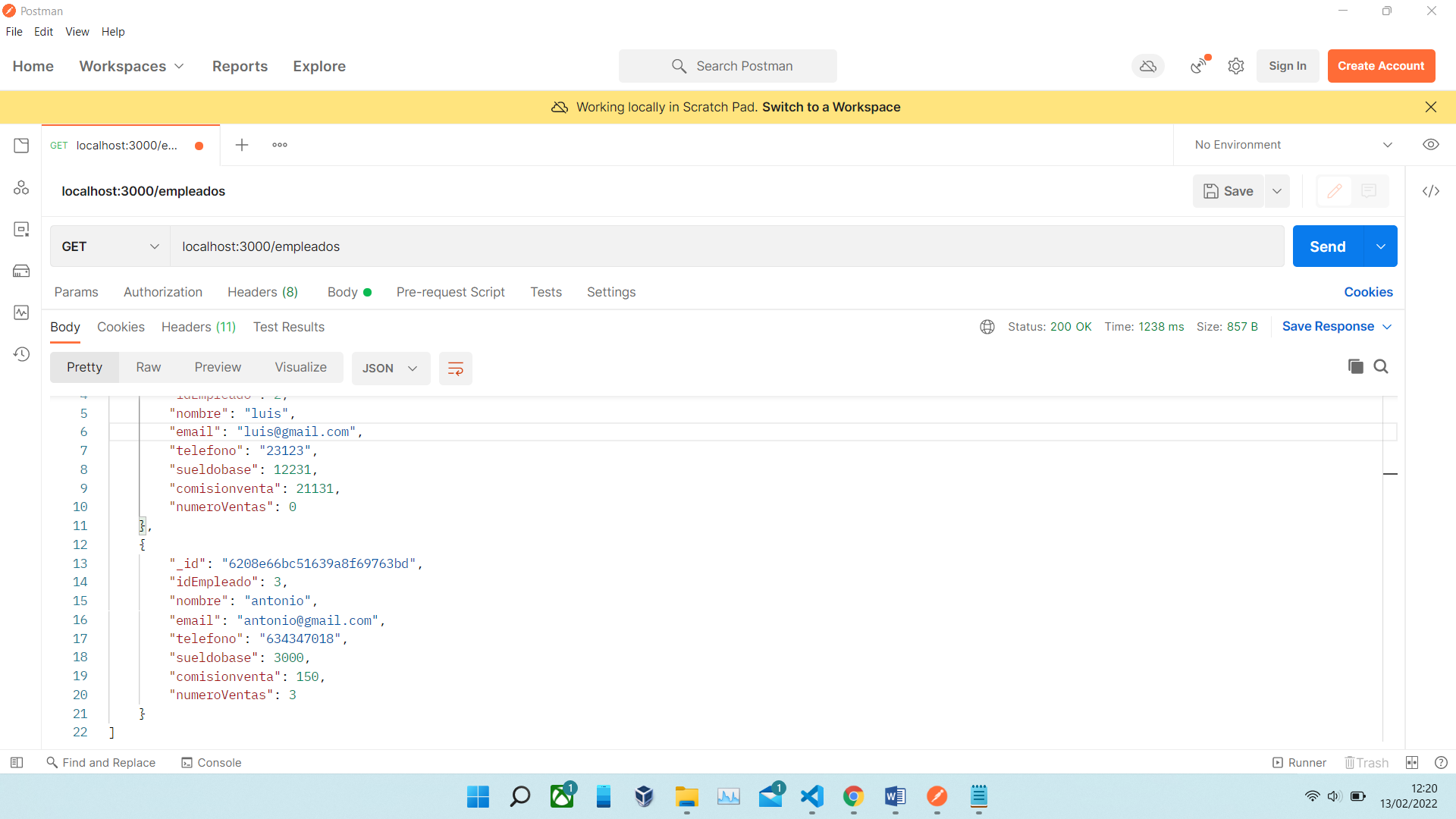
Se usa para eliminar una vivienda



Por último, remarcar esta función:

*this***.***\_router***.**get('/empleados', *this***.***getEmpleado*)





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

## 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.

Dafault: 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*,

    },

    caracteristicas: {

        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

    },

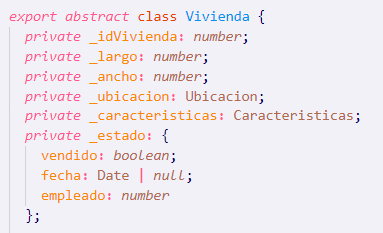
    comisionventa: *Number*,

})

## CLASES

## 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.



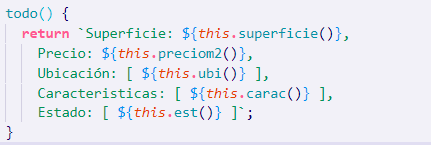
Las interfaces están formuladas en otro directorio llamados interfaces.ts y serían de tal forma:



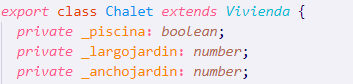
Los métodos de la clase vivienda son:



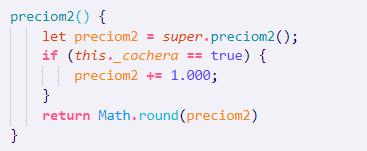




## CLASE CHALET Y CASA (HIJAS)

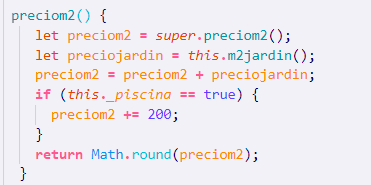
 

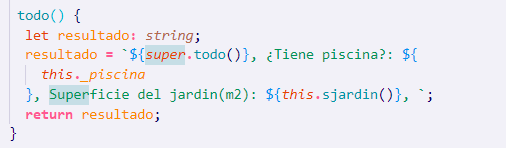
Polimorfismo en la clase Casa:





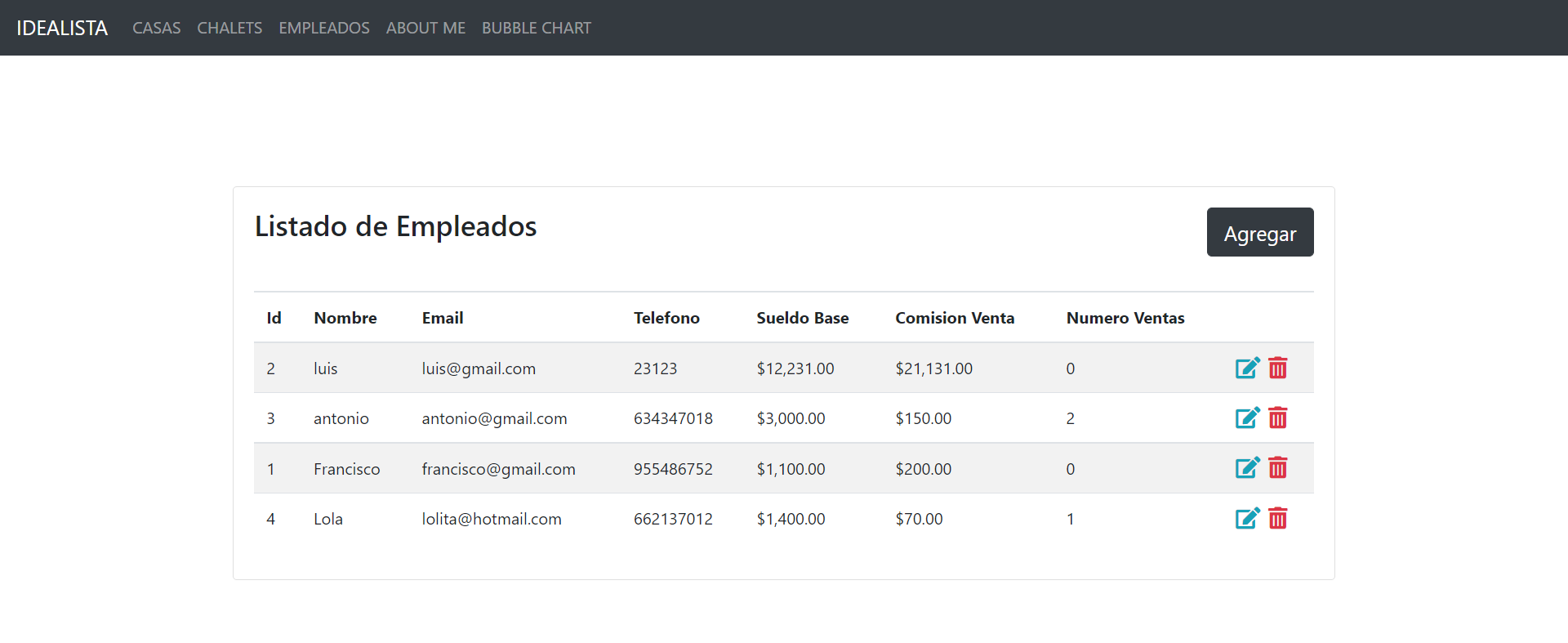
Polimorfismo en la clase Chalet:



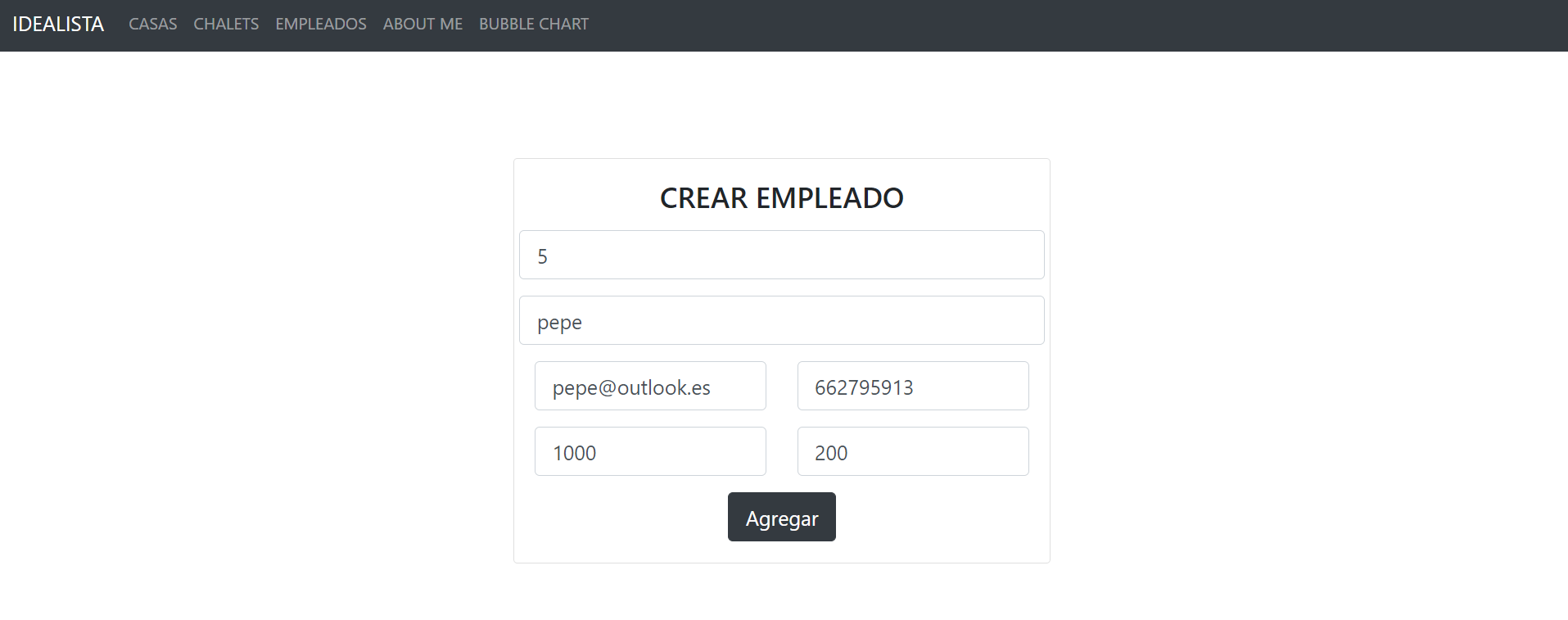


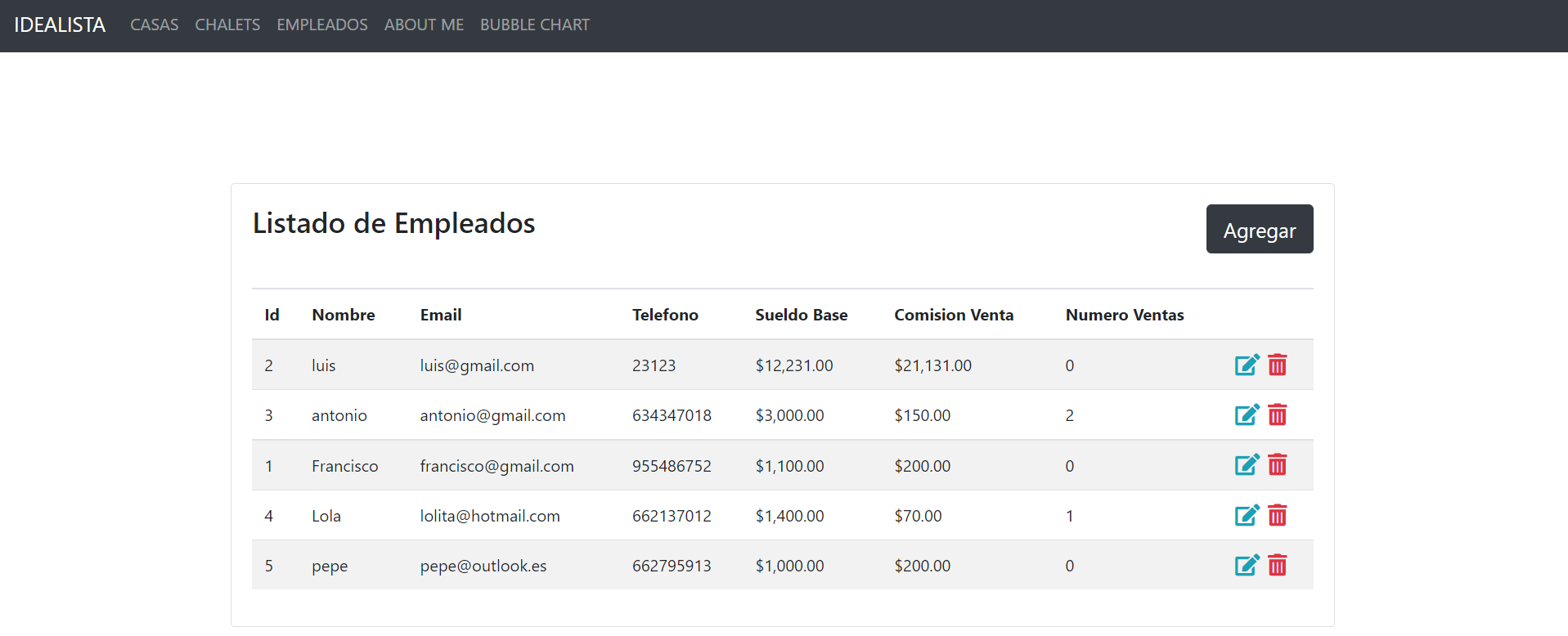
## VISTA DEL USUARIO FINAL

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

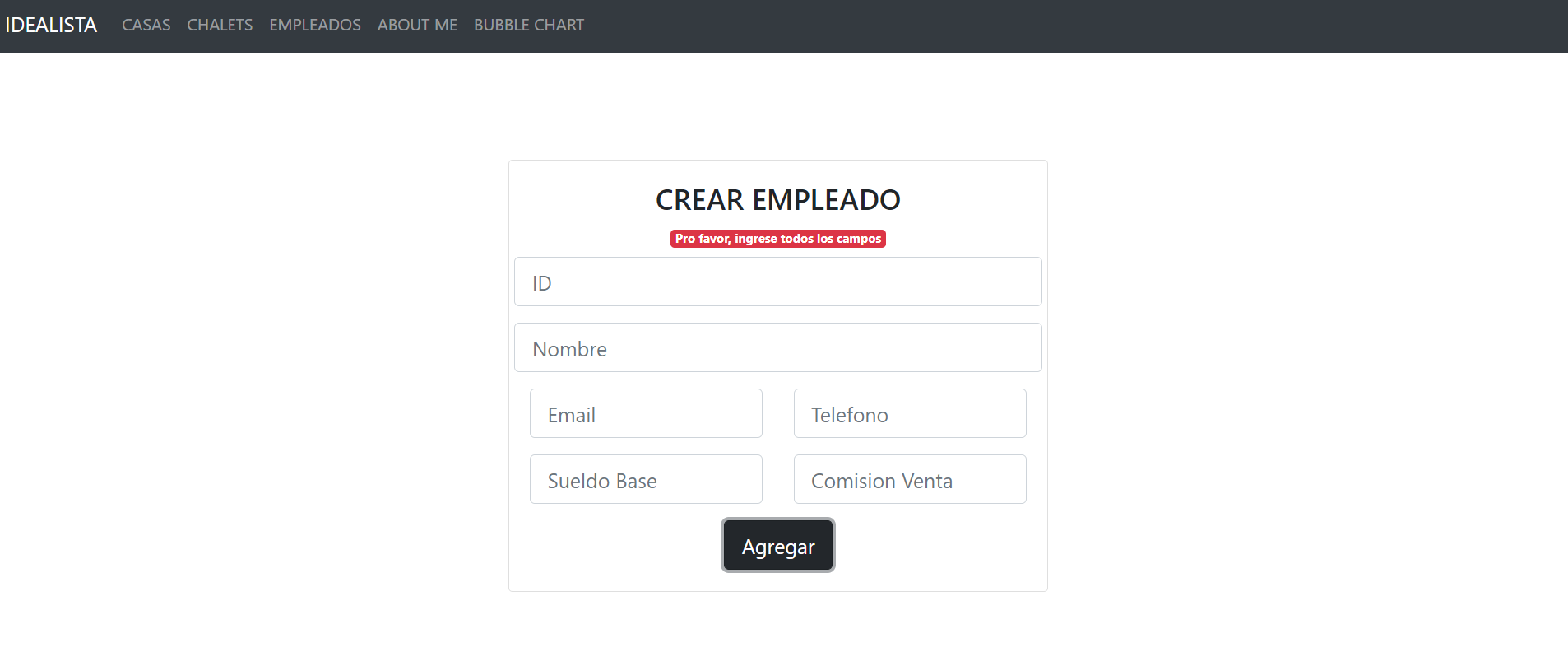


AGREGAR UN EMPLEADO





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



¿Y como se consigue esto?

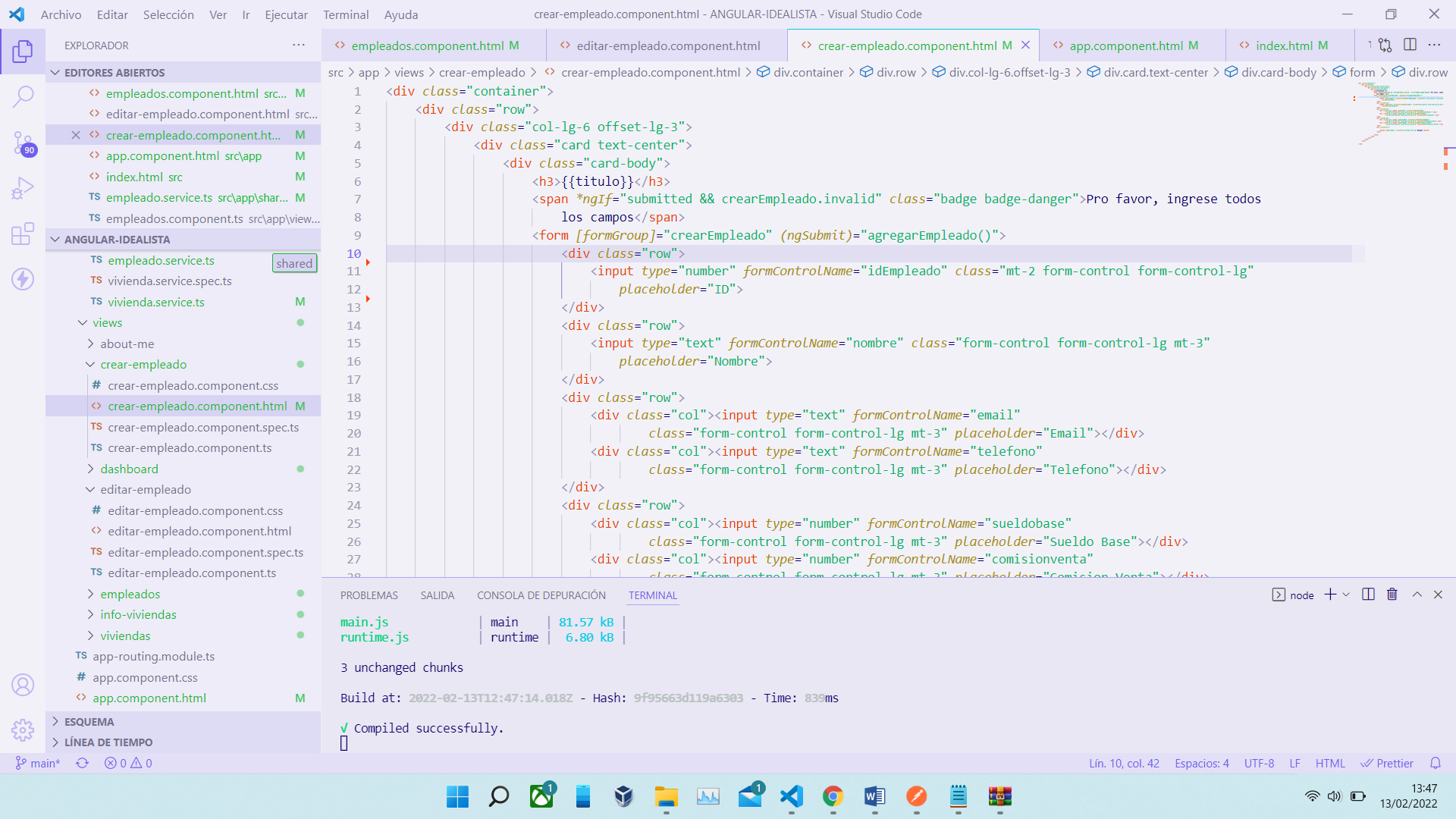
Pues bien, hay dos partes imprescindibles:

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

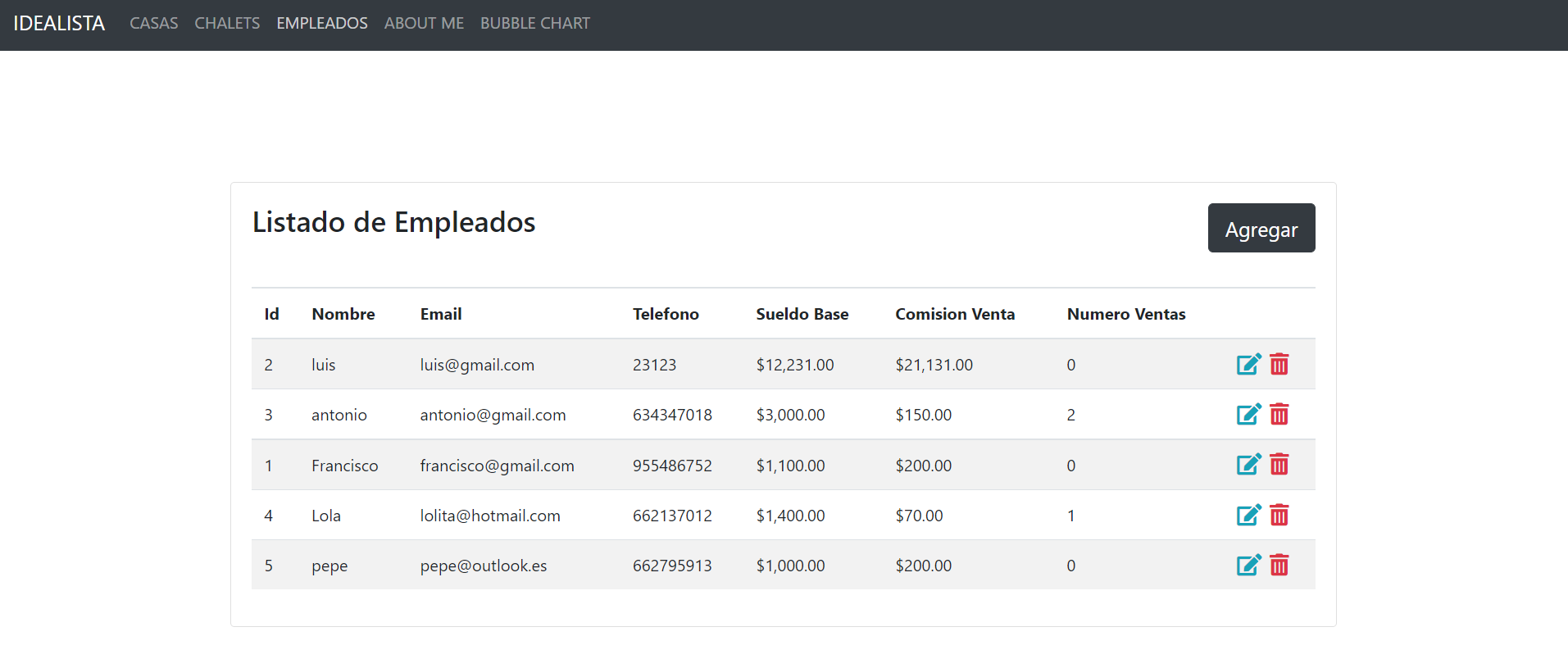


¿Qué quiere decir que el formulario sea invalido?

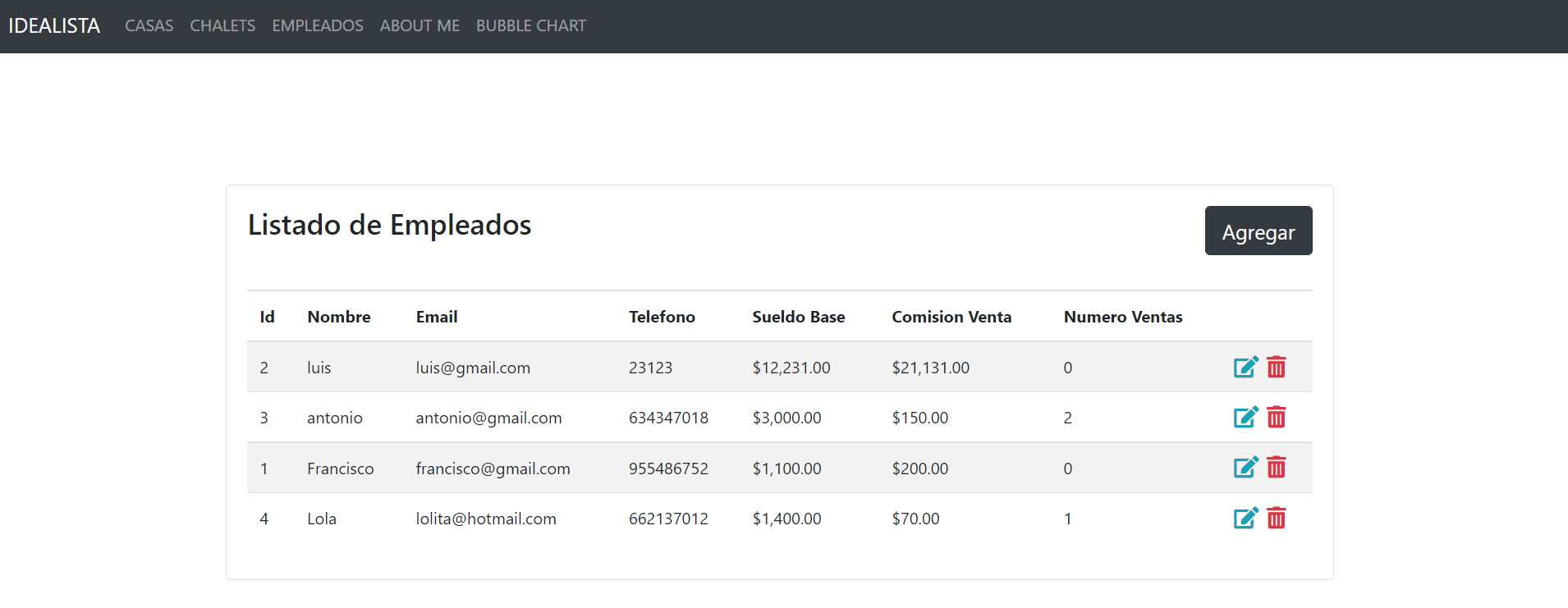
Pues significa que el formulario no cumple estos requisitos que se encuentran en el archivo ts del componente:



BORRAR EMPLEADO

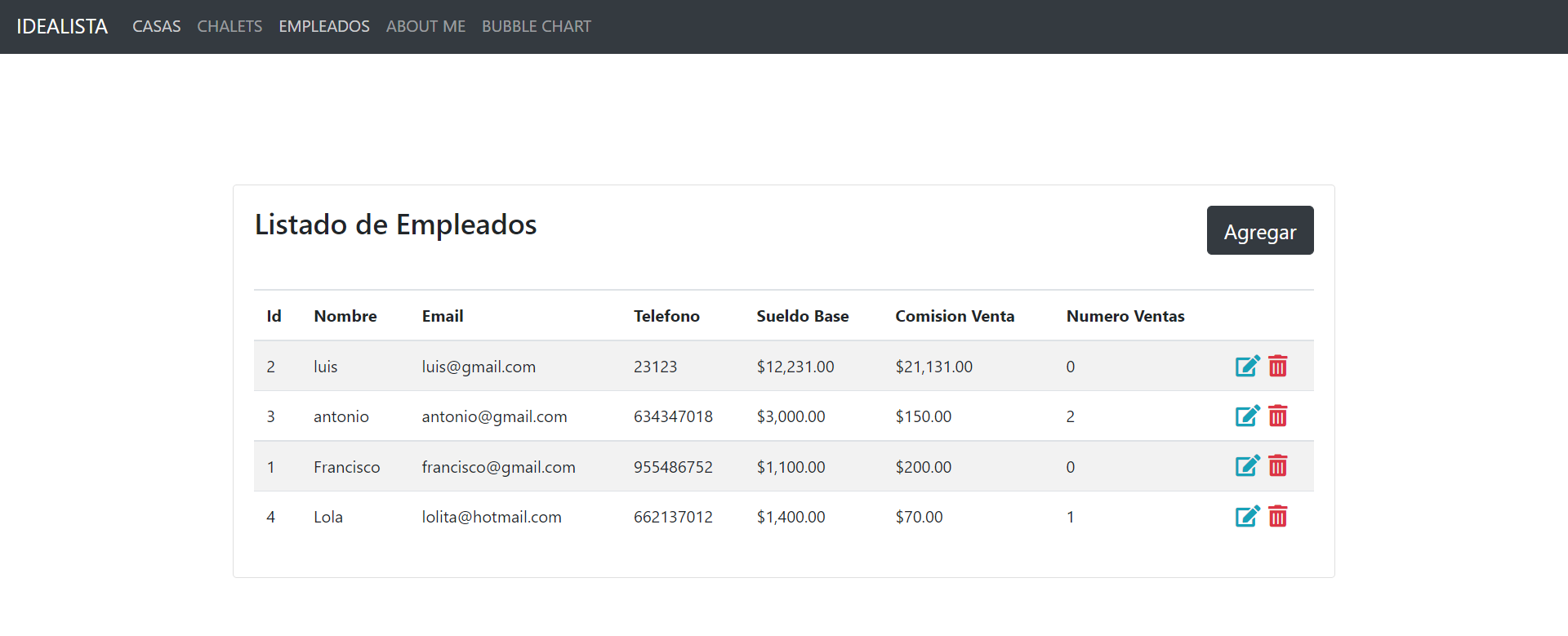


Como vemos en la siguiente foto, pepe ha sido eliminado.



EDITAR EMPLEADO

Vamos a editar a Lola:



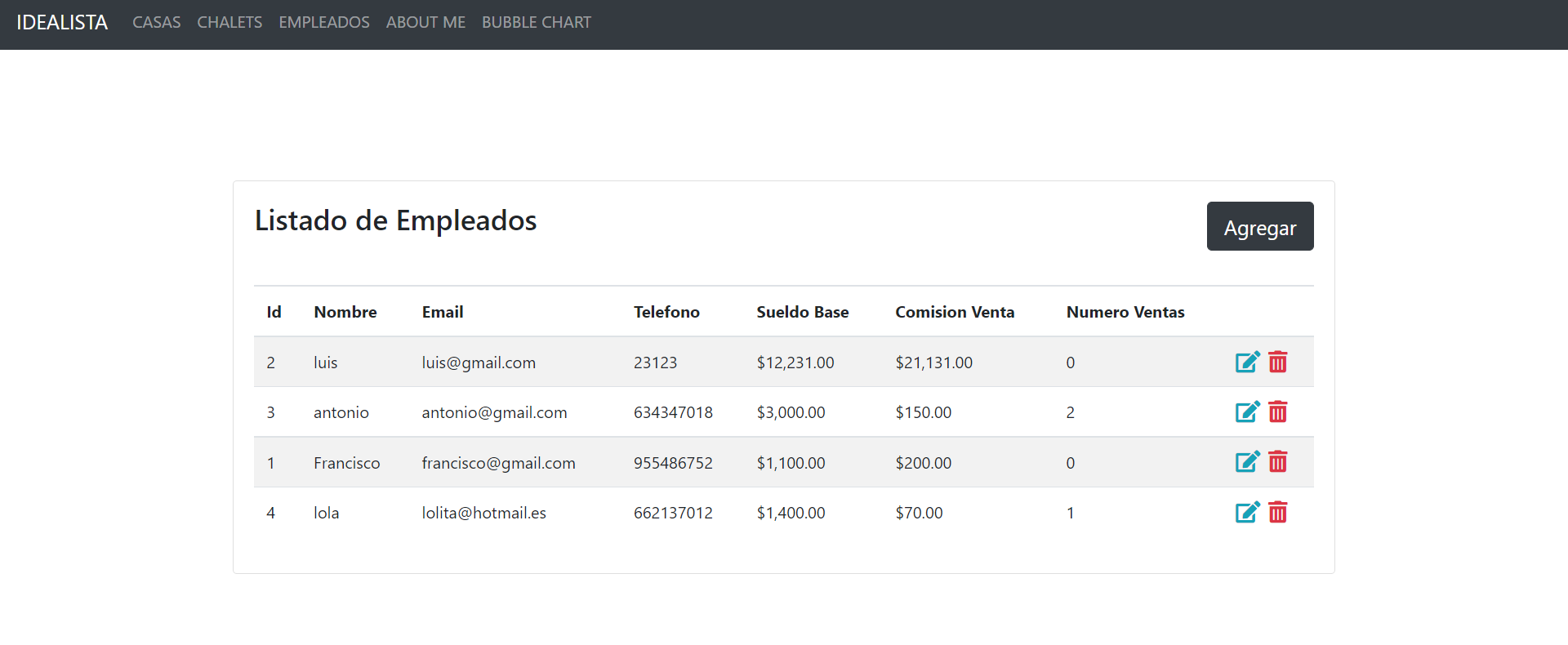
Como podemos ver, al haber hecho click en LOLA su ID viene automaticamente al formulario de edición.



Cambiaremos el email:

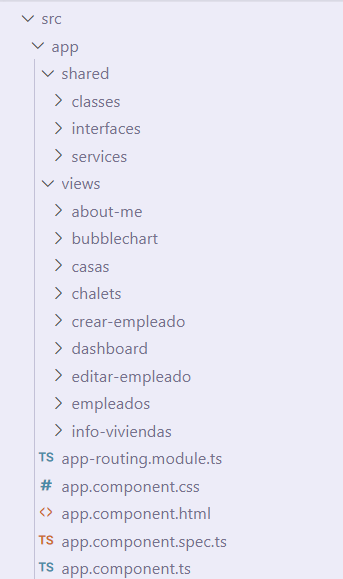


Como podemos ver, el email de LOLA ha cambiado.



## ESTRUCTURA ANGULAR

El proyecto está estructurado de la siguiente forma:



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

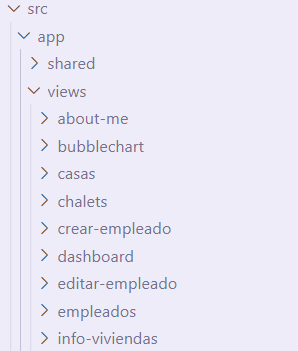
SHARED: donde encontraremos toda la parte ‘’logica’’

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:

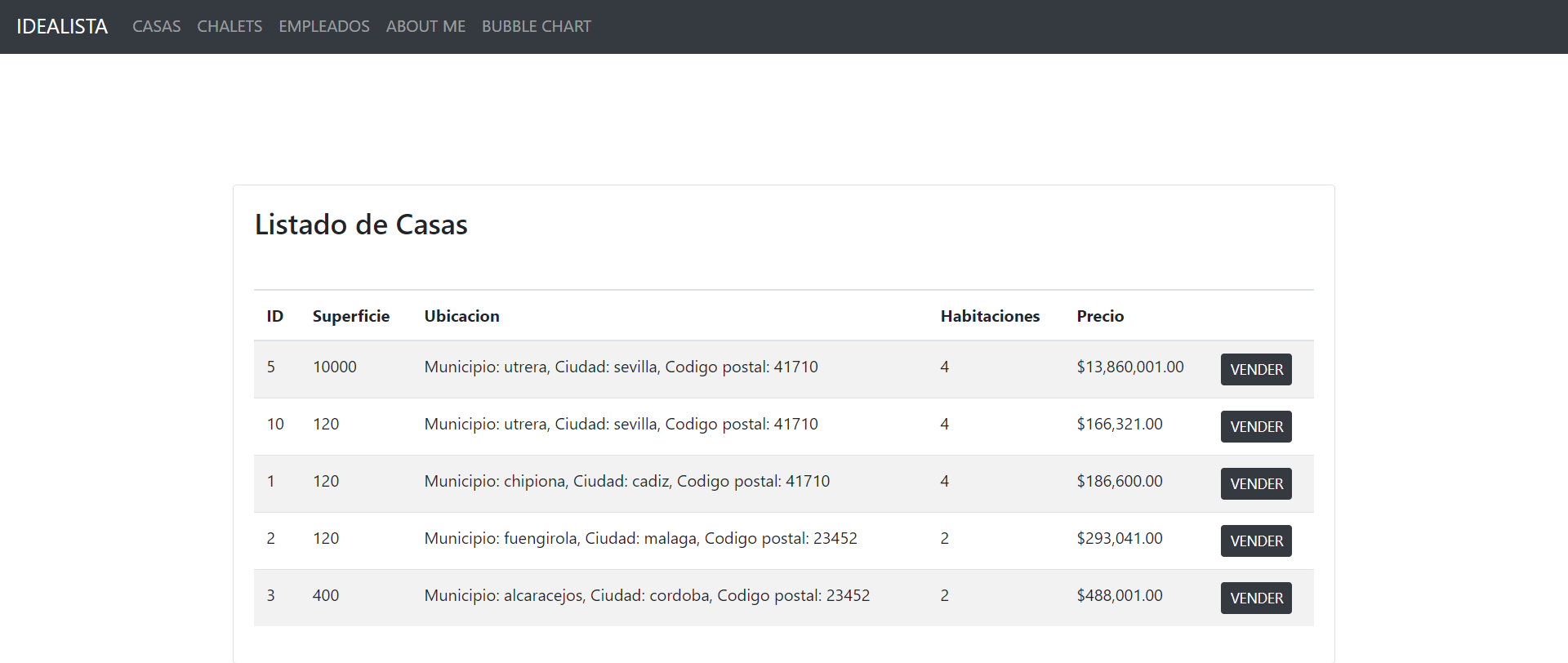
* Componentes

## ¿Qué hace cada uno de los componentes?

### DASHBOARD



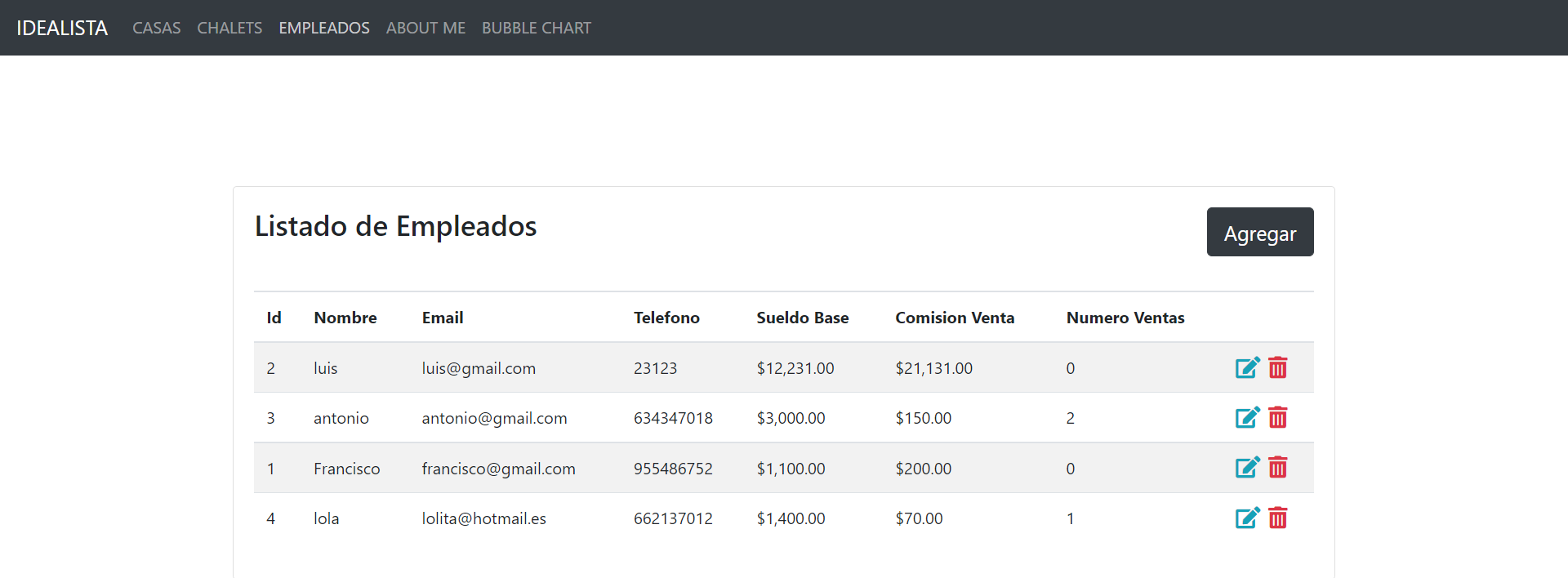
### CASAS



### CHALETS

### 

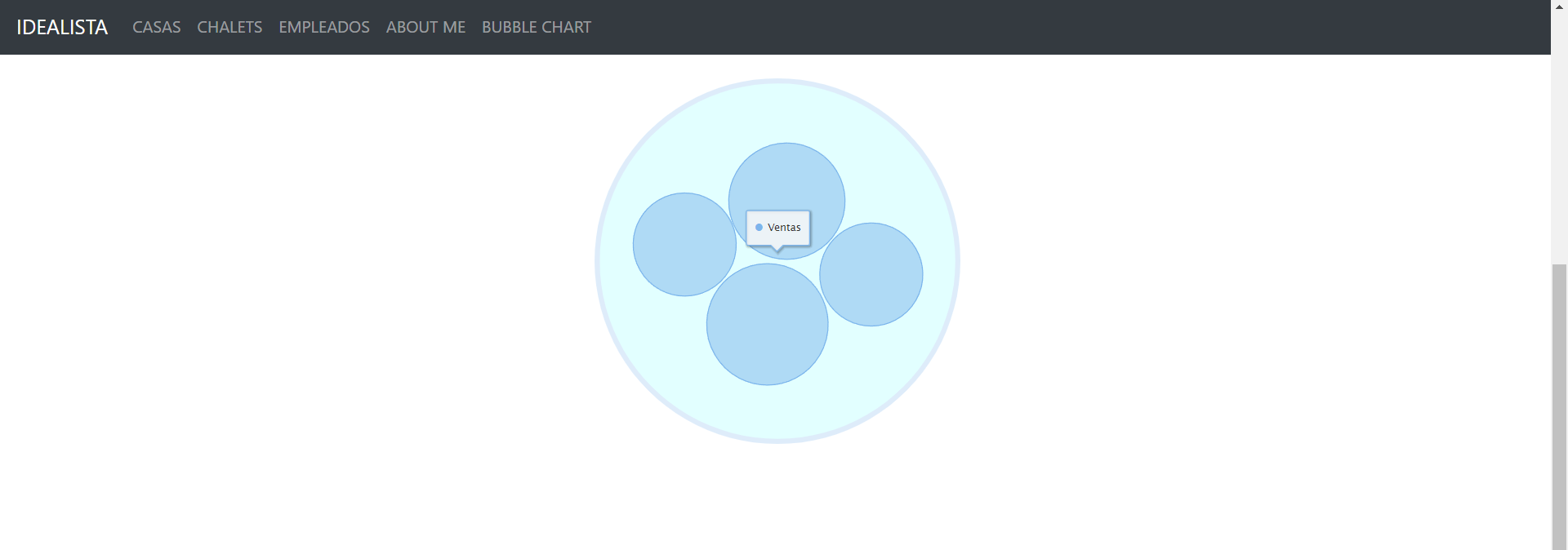
### EMPLEADOS



### ABOUT ME



### BUBBLE CHART



## MODULOS Y ROUTING

### APP.MODULE.TS



### APP-ROUTING



## POLIMORFISMO Y HERENCIAS

