

Documentación

Proyecto Trenes

2ºEVA



- Ciclo formativo: 2ºA.S.I.R-A
- Módulo: SGBD
- Unidad de trabajo: ANGULAR
- Fecha de entrega de la práctica:
- Nombre y apellidos: Alvaro Lucio-Villegas de Cea

Índice

API RESP	3
Get	3
Get con ID	8
POST	10
Delete	15
Update	17
Rutas	22
Angular	23
Códigos a resaltar	24
Formularios dinámicos	24
Gráficos	26
Mejor cliente	26
Tren más rentable	27
Sueldos de empleados	29
Actualizar	31
Método de clases	36

API RESP

Get

Le pide a la Base de Datos todos los datos de las colecciones de Empleados y Clientes

```
private getEmpleados = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Empleado.find({})
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })

  db.desconectarBD()
}

private getClientes = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Clientes.find({})
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })

  db.desconectarBD()
}
```

Colecciones Viajes y Billetes

```
private getViajes = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Viaje.find({})
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })

  db.desconectarBD()
}

private getBilletes = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Billetes.find({})
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })

  db.desconectarBD()
}
```

y Registros

```
private getRegistros = async (req: Request, res: Response) => {
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Registros.find({})
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })

  db.desconectarBD()
}
```

esta es la función más importante de la res Api ya que calcula el salario de los empleados dependiendo de qué tipo de empleado es y donde está trabajando.

```
private calcularSalario = async (req:Request, res:Response) => {
  try {
    await db.conectarBD()
    const id =req.params._dni
    let tmpOperario :Operario
    let tmpRevisor :Revisor
    let tmpLimpiador :Limpiador
    //let prueba:ERecepcion
    const query = await Empleado.findOne({_dni:id})

    if(query._tipoObjeto == "operario"){
      tmpOperario=new Operario (

        query._dni,
        query._nombre,
        query._telefono,
        query._sueldo,
        query._tren,
        query._viajes
      )
      let salario =Promise.resolve(tmpOperario.salario())
      res.send((await salario).toString())
    }else if(query._tipoObjeto == "revisor"){
      tmpRevisor=new Revisor (

        query._dni,
        query._nombre,
        query._telefono,
        query._sueldo,
        query._horas,
        query._viajes
      )
      let salario =tmpRevisor.salario().toString()
      res.send(salario)
    }else if(query._tipoObjeto == "limpiador"){
```

en los empleados "limpiador y operario varía su sueldo en base en que tren están trabajando

Para ello me ha sido necesario crear una promesa y que se resuelva en las rutas pero en la Función de las clases hago lo siguiente.

```
async salario(){
  let sueldocalculado:any
  sueldocalculado= await calcularSueldo(this.dni,this.sueldo)
  let sueldofinal=sueldocalculado*1

  return sueldofinal;
}
```

en la función de las clases no puede realizar el cálculo que quería para ello cree una función externa que si la hace .

```
export let calcularSueldo =async(dni:string,salariobase:number)=>{
  let salariocalculado=0

  let query: any = await Empleado.findOne({_dni:dni})
  let query1 = await Viaje.findOne({_id:query._tren})

  if(query1._tipoObjeto == "pasajeros"){
    salariocalculado=salariobase*1.5
  }else{
    salariocalculado=salariobase*1.25
  }

  if(query._viajes>=20){
    salariocalculado=salariocalculado*1.15
  }else if(query._viajes>=50){
    salariocalculado=salariocalculado*1.25
  }

  return salariocalculado
}
```

esta funcion tambien la usa el empleado limpiador y en ese caso tienen un campo que en número de viajes y en esta función se le aplican esos cálculos

función de limpiador

```
async salario(){  
  
    let sueldocalculado:any  
    sueldocalculado= await calcularSueldo(this.dni,this.sueldo)  
  
    if (this._horas>=200) {  
        sueldocalculado=this.sueldo*1.20  
    }else if(this._horas>=100){  
        sueldocalculado=this.sueldo*1.10  
    }else{  
        sueldocalculado=this.sueldo*1.02  
    }  
    let sueldofinal=sueldocalculado  
    return sueldofinal;  
}
```

Get con ID

En este caso he creado una serie de llamadas que son por una ID definida para que me devuelvan los datos enlazados con esa id en la base de datos.

El primero es el de buscar empleados por DNI

```
//Consulta selecta
private getempleadoDNI = async (req: Request, res: Response) => {
  const valor = req.params._dni
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Empleado.aggregate([
      {
        $match:{"_dni" : valor}
      }
    ])
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })
}
```

clientes por dni

```
private getclienteDNI = async (req: Request, res: Response) => {
  const valor = req.params._dni
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Clientes.aggregate([
      {
        $match:{"_dni" : valor}
      }
    ])
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })
}
```


y por último buscamos viajes por ID

```
private getviajeID = async (req: Request, res: Response) => {
  const valor = req.params._id
  await db.conectarBD()
  .then( async (mensaje) => {
    console.log(mensaje)
    const query = await Viaje.aggregate([
      {
        $match: {"_id" : valor}
      }
    ])
    res.json(query)
  })
  .catch((mensaje) => {
    res.send(mensaje)
  })
}
```

POST

Ahora vamos a ver las creaciones en este caso de los empleados

Operario

```
//POST
private crearOperario = async (req: Request, res: Response) => {

  const {_dni,_nombre,_telefono,_sueldo,_tren,_viajes} = req.body
  await db.conectarBD()
  const dSchema = {
    _tipoObjeto: "operario",
    _dni: _dni,
    _nombre: _nombre,
    _telefono: _telefono,
    _sueldo: _sueldo,
    _tren: _tren,
    _viajes: _viajes
  }
  const oSchema = new Empleado(dSchema)
  await oSchema.save()
  //res.send(nombre)
  .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Limpiador

```
private crearLimpiador = async (req: Request, res: Response) => {

  const {_dni,_nombre,_telefono,_sueldo,_horas,_tren} = req.body
  await db.conectarBD()
  const dSchema = {
    _tipoObjeto: "limpiador",
    _dni: _dni,
    _nombre: _nombre,
    _telefono: _telefono,
    _sueldo: _sueldo,
    _horas: _horas,
    _tren: _tren
  }
  const oSchema = new Empleado(dSchema)
  await oSchema.save()
  //res.send(nombre)
  .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Revisor

```
private crearRevisor = async (req: Request, res: Response) => {

    const {_dni, _nombre, _telefono, _sueldo, _horas, _viajes} = req.body
    await db.conectarBD()
    const dSchema = {
        _tipoObjeto: "revisor",
        _dni: _dni,
        _nombre: _nombre,
        _telefono: _telefono,
        _sueldo: _sueldo,
        _horas: _horas,
        _viajes: _viajes
    }
    const oSchema = new Empleado(dSchema)
    await oSchema.save()
    //res.send(nombre)
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

    await db.desconectarBD()
}
```

Ahora los clientes

```
private crearCliente = async (req: Request, res: Response) => {

    const {_dni, _nombre, _telefono, _email} = req.body
    await db.conectarBD()
    const dSchema = {
        _dni: _dni,
        _nombre: _nombre,
        _telefono: _telefono,
        _email: _email,
    }
    const oSchema = new Clientes(dSchema)
    await oSchema.save()
    //res.send(nombre)
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

    await db.desconectarBD()
}
```

A La hora de crear los trenes es necesario crearlos ya definiendo su tipo al igual que con los empleados.

Viaje de Pasajeros

```
//Viaje
private crearTrenPasajeros = async (req: Request, res: Response) => {

    const {_id, _origen, _destino, _nPasajeros, _nPlazas, _precio} = req.body
    await db.conectarBD()
    const dSchema = {
        _tipoObjeto: "pasajeros",
        _id: _id,
        _origen: _origen,
        _destino: _destino,
        _nPasajeros: _nPasajeros,
        _nPlazas: _nPlazas,
        _precio: _precio
    }

    const oSchema = new Viaje(dSchema)
    await oSchema.save()
    //res.send(nombre)
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

    await db.desconectarBD()
}
```

Viaje de Mercancías

```
private crearTrenMercancias = async (req: Request, res: Response) => {

    const {_id, _origen, _destino, _tipoCarga, _kilosCarga} = req.body
    await db.conectarBD()
    const dSchema = {
        _tipoObjeto: "mercancias",
        _id: _id,
        _origen: _origen,
        _destino: _destino,
        _tipoCarga: _tipoCarga,
        _kilosCarga: _kilosCarga
    }

    const oSchema = new Viaje(dSchema)
    await oSchema.save()
    //res.send(nombre)
    .then((doc: any) => res.send('Has guardado el archivo:\n' + doc))
    .catch((err: any) => res.send('Error: ' + err))

    await db.desconectarBD()
}
```

Creación de registro de mercancías y compra de billetes de cada tren

Registros mercancías

```
private crearRegistro = async (req: Request, res: Response) => {
  const { _id, tren_id, kilometros } = req.body
  await db.conectarBD()
  let dSchema:any

  let busquedatren =await Viaje.findOne({_id:tren_id})
  if(busquedatren._tipoObjeto=="mercancias"){

    dSchema={
      _id: _id,
      _tren_id: _tren_id,
      _origen:busquedatren._origen,
      _destino:busquedatren._destino,
      _fecha:new Date,
      _kilometros:_kilometros
    }

  }else{
    res.send("Ese tren es de pasajeros")
  }

  const oSchema = new Registros(dSchema)
  await oSchema.save()
    .then( (doc: any) => res.send(doc))
    .catch( (err: any) => res.send('Error: ' + err))
  await db.desconectarBD()
}
```

Billetes

```
private crearBillete = async (req: Request, res: Response) => {
  const { _dni, _idTrenPasajeros, _asiento } = req.body
  await db.conectarBD()
  let dSchema:any

  let busquedatren =await Viaje.findOne({_id:_idTrenPasajeros})
  //res.send(busquedatren)
  if(busquedatren._tipoObjeto=="pasajeros"){
    let busquedacli =await Clientes.findOne({_dni:_dni})
    if(busquedacli){
      dSchema={
        _dni:_dni,
        _idTrenPasajeros:_idTrenPasajeros,
        _origen:busquedatren._origen,
        _destino:busquedatren._destino,
        _asiento:_asiento,
        _precio:busquedatren._precio,
        _fecha:new Date
      }
    }
  }else{
    res.send("Ese tren no es de pasajeros")
  }

  const oSchema = new Billetes(dSchema)
  await oSchema.save()
    .then( (doc: any) => res.send(doc))
    .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}
```

Delete

Ahora vamos a crear la opción de borrar los distintos elementos.

Eliminación de Clientes y empleados

```
//DELETE
//Cliente
private deleteCliente = async (req: Request, res: Response) => {
  const { _dni } = req.params
  await db.conectarBD()
  await Clientes.findOneAndDelete(
    { _dni: _dni }
  )
  .then( (doc: any) => {
    if (doc == null) {
      res.send(`No encontrado`)
    }else {
      res.send('Borrado correcto: '+ doc)
    }
  })
  .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}

//Empleados
private deleteEmpleados = async (req: Request, res: Response) => {
  const { _dni } = req.params
  await db.conectarBD()
  await Empleado.findOneAndDelete(
    { _dni: _dni }
  )
  .then( (doc: any) => {
    if (doc == null) {
      res.send(`No encontrado`)
    }else {
      res.send('Borrado correcto: '+ doc)
    }
  })
  .catch( (err: any) => res.send('Error: '+ err))
  db.desconectarBD()
}
```

Trenes (viaje) y billetes

```
//Trenes
private deleteTrenes = async (req: Request, res: Response) => {
  const { _id } = req.params
  await db.conectarBD()
  await Viaje.findOneAndDelete(
    { _id: _id }
  )
  .then( (doc: any) => {
    if (doc == null) {
      res.send(`No encontrado`)
    }else {
      res.send('Borrado correcto: '+ doc)
    }
  })
  .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}

//Billetes
private deleteBilletes = async (req: Request, res: Response) => {
  const { _dni } = req.params
  await db.conectarBD()
  await Billetes.findOneAndDelete(
    { _dni: _dni }
  )
  .then( (doc: any) => {
    if (doc == null) {
      res.send(`No encontrado`)
    }else {
      res.send('Borrado correcto: '+ doc)
    }
  })
  .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}
```

por último el registro de mercancías.

```
private deleteRegistros = async (req: Request, res: Response) => {
  const { id } = req.params
  await db.conectarBD()
  await Registros.findOneAndDelete(
    { _id: id }
  )
  .then( (doc: any) => {
    if (doc == null) {
      res.send(`No encontrado`)
    }else {
      res.send('Borrado correcto: '+ doc)
    }
  })
  .catch( (err: any) => res.send('Error: '+ err))
  await db.desconectarBD()
}
```


Update

En este apartado he preferido limitar los distintos campos para que no se puedan cambiar como la PRIMARY KEY de las colecciones

Caso de Clientes

```
//UPDATE
private actualizarTlfCliente = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const telefono = req.params._telefono
  await Clientes.findOneAndUpdate(
    { _dni: dni },
    {
      _telefono: telefono
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))
  await db.desconectarBD()
}
```

Caso de Empleados

En este caso es especial ya que existen veces que solo queremos cambiar un campo dependiendo del tipo de empleado pero la función de salario es igual para todos los empleados.

```
//Empleados
private actualizarSalarioEmpleado = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const salario = req.params._sueldo
  await Empleado.findOneAndUpdate(
    { _dni: dni },
    {
      _sueldo: salario
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))
  await db.desconectarBD()
}
```

Cambiar el tren en el que está trabajando.

```
//Operario
private actualizarTrenOperario = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const idTren = req.params._tren
  await Empleado.findOneAndUpdate(
    { _dni: dni,
      _tipoObjeto: "operario"
    },
    {
      _tren: idTren
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))
  await db.desconectarBD()
}
```

Actualizar número de viajes

```
private actualizarViajesOperario = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const nViajes = req.params._viajes
  await Empleado.findOneAndUpdate(
    { _dni: dni,
      _tipoObjeto: "operario"
    },
    {
      _viajes: nViajes
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))
  await db.desconectarBD()
}
```

Actualización de Horas del empleado revisor

```
//Revisor
private actualizarViajesHorasRevisor = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const nViajes = req.params._viajes
  const horas = req.params._horas

  await Empleado.findOneAndUpdate(
    { _dni: dni,
      _tipoObjeto: "revisor"
    },
    {
      _horas: horas,
      _viajes: nViajes
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Actualización de horas y tren asignado

```
//Limpiador
private actualizarHorasTrenLimpiador = async (req: Request, res: Response) => {
  await db.conectarBD()
  const dni = req.params._dni
  const idTren = req.params._tren
  const horas = req.params._horas

  await Empleado.findOneAndUpdate(
    { _dni: dni,
      _tipoObjeto: "limpiador"
    },
    {
      _horas: horas,
      _tren: idTren
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Ahora los cambios que podemos hacer en los viajes , que son el destino y el origen.

```
//Viajes
private actualizarOrigenViajes = async (req: Request, res: Response) => {
  await db.conectarBD()
  const id = req.params._id
  const origen = req.params._origen
  await Viaje.findOneAndUpdate(
    { _id: id },
    {
      _origen: origen
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}

private actualizarDestinoViajes = async (req: Request, res: Response) => {
  await db.conectarBD()
  const id = req.params._id
  const destino = req.params._destino
  await Viaje.findOneAndUpdate(
    { _id: id },
    {
      _destino: destino
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Ahora los cambios dependiendo del tipo de viaje

Mercancías -Carga y cantidad

```
//Mercancias
private actualizarCarga = async (req: Request, res: Response) => {
  await db.conectarBD()
  const id = req.params._id
  const tipo = req.params._tipoCarga
  const cantidad = req.params._kilosCarga
  await Viaje.findOneAndUpdate(
    { _id: id,
      _tipoObjeto: "mercancias"
    },
    {
      _tipoCarga: tipo,
      _kilosCarga: cantidad
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Pasajeros Precio del Viaje/Billete

```
//Pasajeros
private actualizarPrecio = async (req: Request, res: Response) => {
  await db.conectarBD()
  const id = req.params._id
  const precio = req.params._precio

  await Viaje.findOneAndUpdate(
    { _id: id,
      _tipoObjeto: "pasajeros"
    },
    {
      _precio: precio,
    }
  )
  .then((doc: any) => res.send('Se han actualizado los datos:\n' + doc))
  .catch((err: any) => res.send('Error: ' + err))

  await db.desconectarBD()
}
```

Rutas

```
misRutas() {
  this._router.get('/', this.index)

  //GET

  this._router.get('/clientes', this.getClientes)
  this._router.get('/empleados', this.getEmpleados)
  this._router.get('/viajes', this.getViajes)
  this._router.get('/billetes', this.getBilletes)
  this._router.get('/registro', this.getRegistros)

  //POST

  this._router.post('/crearOperario', this.crearOperario)
  this._router.post('/crearLimpiador', this.crearLimpiador)
  this._router.post('/crearRevisor', this.crearRevisor)
  this._router.post('/crearRegistro', this.crearRegistro)
  this._router.post('/crearBilletes', this.crearBillete)
  this._router.post('/crearCliente', this.crearCliente)
  this._router.post('/crearTrenPasajeros', this.crearTrenPasajeros)
  this._router.post('/crearTrenMercancias', this.crearTrenMercancias)

  //GET con ID

  this._router.get('/salarios/:_dni', this.calcularSalario)
  this._router.get('/viaje/:_id', this.getviajeID)
  this._router.get('/clientes/:_dni', this.getclienteDNI)
  this._router.get('/empleado/:_dni', this.getempleadoDNI)
```

```
//DELETE
this._router.delete('/deleteCliente/:_dni', this.deleteCliente)
this._router.delete('/deleteEmpleado/:_dni', this.deleteEmpleados)
this._router.delete('/deleteTrenes/:_id', this.deleteTrenes)
this._router.delete('/deleteBilletes/:_dni', this.deleteBilletes)
this._router.delete('/deleteRegistros/:_id', this.deleteRegistros)

//UPDATE
//Cliente
this._router.put('/actualizarTlfCli/:_dni/:_telefono', this.actualizarTlfCliente)

//Empleado
this._router.put('/actualizarSalEmp/:_dni/:_sueldo', this.actualizarSalarioEmpleado)

//Operario
this._router.put('/actualizarTrenOpe/:_dni/:_tren', this.actualizarTrenOperario)
this._router.put('/actualizarViajesOpe/:_dni/:_viajes', this.actualizarViajesOperario)

//Revisor
this._router.put('/actualizarViajeHoras/:_dni/:_viajes/:_horas', this.actualizarViajesHorasRevisor)

//Limpiador
this._router.put('/actualizarHorasTren/:_dni/:_horas/:_tren', this.actualizarHorasTrenLimpiador)

//Viajes
this._router.put('/actualizarOrigenViaje/:_id/:_origen', this.actualizarOrigenViajes)
this._router.put('/actualizarDestinoViaje/:_id/:_destino', this.actualizarDestinoViajes)

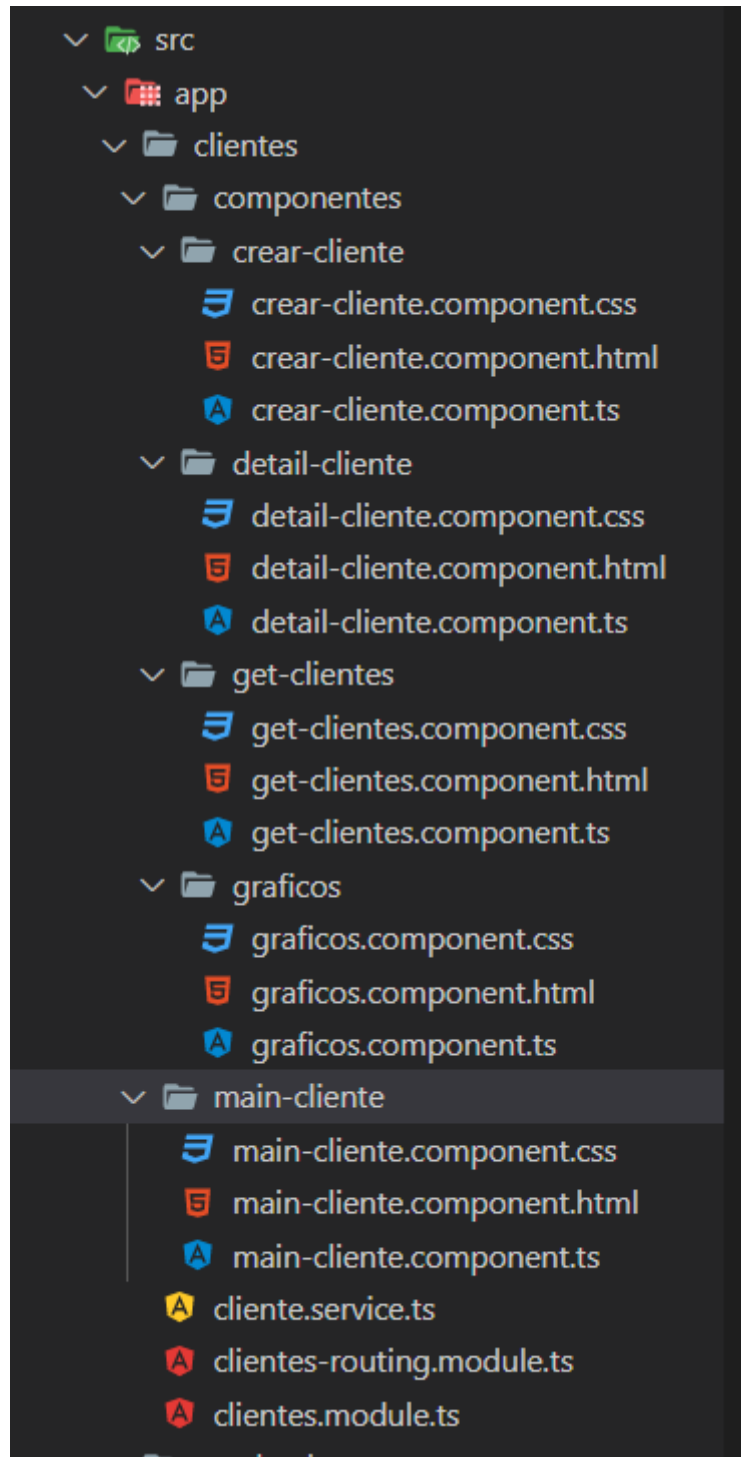
//Mercancias
this._router.put('/actualizarCarga/:_id/:_tipoCarga/:_kilosCarga', this.actualizarCarga)

//Pasajeros
this._router.put('/actualizarPrecio/:_id/:_precio', this.actualizarPrecio)
```

Angular

[Ir a la aplicación](#)

En esta parte del proyecto he seleccionado la estructura de encapsulamiento esto conlleva que cada componente está en una carpeta con todo lo que atañe a esta, por ejemplo services



Esto facilita a mi parecer a la hora de gestionar un código ya que todo está enlazado

Códigos a resaltar

Formularios dinámicos

Vista del usuario:

Administración de Trenes

Clientes Empleados Viajes

Consultar datos Gráficos

Tipo de empleado

Nombre

DNI

Teléfono

Sueldo

Crear

Administración de Trenes

Clientes Empleados Viajes

Consultar datos Gráficos

Tipo de empleado

Nombre

DNI

Teléfono

Sueldo

Viajes

Crear

Codigo

```

1 <div class="container-form">
2   <div class="formulario d-flex w-75 flex-wrap mx-auto mt-5 justify-content-around">
3     <div class="input">
4       <label>Tipo de empleado</label>
5       <select name="tipoObjeto" ngDefaultControl [(ngModel)]="empleado.tipo">
6         <option value="limpiador">Limpiador</option>
7         <option value="revisor">Revisor</option>
8         <option value="operario">Operario</option>
9       </select>
10    </div>
11    <div class="input">
12      <label>Nombre</label>
13      <input type="text" [(ngModel)]="empleado.nombre" placeholder="Nombre">
14    </div>
15    <div class="input">
16      <label>DNI</label>
17      <input type="text" [(ngModel)]="empleado.dni" placeholder="DNI">
18    </div>
19    <div class="input">
20      <label>Teléfono</label>
21      <input type="tel" [(ngModel)]="empleado.telefono" placeholder="Teléfono">
22    </div>
23    <div class="input">
24      <label>Sueldo</label>
25      <input type="number" [(ngModel)]="empleado.sueldo" placeholder="Sueldo">
26    </div>
27    <div class="input" *ngIf="empleado?.tipo == 'limpiador' || empleado?.tipo == 'revisor'">
28      <label>Horas</label>
29      <input type="number"
30        placeholder="Horas" [(ngModel)]="empleado.horas">
31    </div>
32    <div class="input" *ngIf="empleado?.tipo == 'operario' || empleado?.tipo == 'limpiador'" [(ngModel)]="empleado.tren">
33      <label>Elige un tren</label>
34      <select
35        *ngFor="let viaje of viajes" [value]="viaje_id">{{(viaje_origen)}} - {{(viaje_destino)}}</option>
36      </select>
37    </div>
38    <div class="input" *ngIf="empleado?.tipo == 'operario' || empleado?.tipo == 'revisor'">
39      <label>Viajes</label>
40      <input type="number"

```

```

36 ngOnInit(): void {
37 }
38
39 crearEmpleado(): void {
40   switch(this.empleado.tipo) {
41     case "limpiador":
42       let limpiador = new Limpiador(this.empleado.tipo, this.empleado.dni, this.empleado.nombre, this.empleado.telefono, this.empleado.sueldo);
43       this.empleadoService.crearLimpiador(limpiador).subscribe((respuesta) => {
44         this._snackBar.open("Limpiador creado correctamente", "", {
45           duration: 1000,
46           verticalPosition: "top"
47         });
48         this.router.navigateByUrl("/cliente")
49       });
50       break;
51     case "revisor":
52       let revisor = new Revisor(this.empleado.tipo, this.empleado.dni, this.empleado.nombre, this.empleado.telefono, this.empleado.sueldo);
53       this.empleadoService.crearRevisor(revisor).subscribe((respuesta) => {
54         this._snackBar.open("Revisor creado correctamente", "", {
55           duration: 1000,
56           verticalPosition: "top"
57         });
58         this.router.navigateByUrl("/cliente")
59       });
60       break;
61     case "operario":
62       let operario = new Operario(this.empleado.tipo, this.empleado.dni, this.empleado.nombre, this.empleado.telefono, this.empleado.sueldo);
63       this.empleadoService.crearOperario(operario).subscribe((respuesta) => {
64         this._snackBar.open("Operario creado correctamente", "", {
65           duration: 1000,
66           verticalPosition: "top"
67         });
68         this.router.navigateByUrl("/cliente")
69       });
70       break;
71   }
72 }

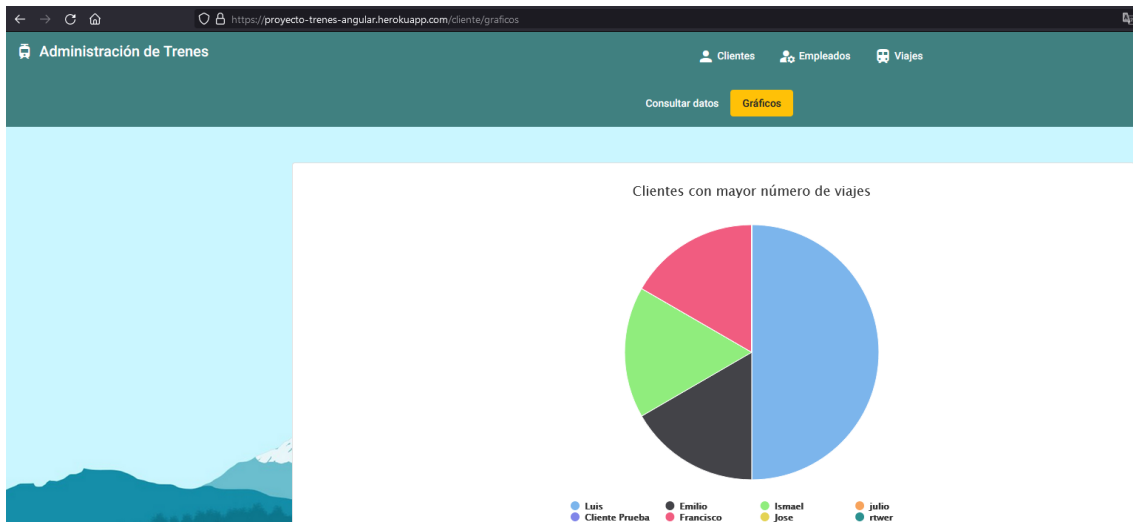
```

Gráficos

Mejor cliente

Cuenta la cantidad de Billetes comprados

Vista usuario:



Código:

```

11 export class GraficosComponent implements OnInit {
12   total_viajes!: number;
13   viajes!: any[];
14   clientes!: any[];
15
16   chart;
17   updateFromInput = false;
18   chartConstructor = "chart";
19   chartCallback;
20   Highcharts: typeof Highcharts = Highcharts;
21   chartOptions: any = {
22     chart: {
23       plotBackgroundColor: null,
24       plotBorderWidth: null,
25       plotShadow: false,
26       type: 'pie',
27       height: "500px"
28     },
29
30     title: {
31       text: 'Clientes con mayor número de viajes'
32     },
33     tooltip: {
34       pointFormat: '{series.name}: <b>{point.percentage:.1f}%</b>'
35     },
36     accessibility: {
37       point: {
38         valueSuffix: '%'
39       }
40     },
41     plotOptions: {
42       pie: {
43         allowPointSelect: true,
44         cursor: 'pointer',
45         dataLabels: {
46           enabled: false
47         },
48         showInLegend: true
49       }
50     },
51     series: [{
52       name: 'Cuota de billetes',
53       colorByPoint: true,
54       data: []
55     }],
56     exporting: {
57

```

```

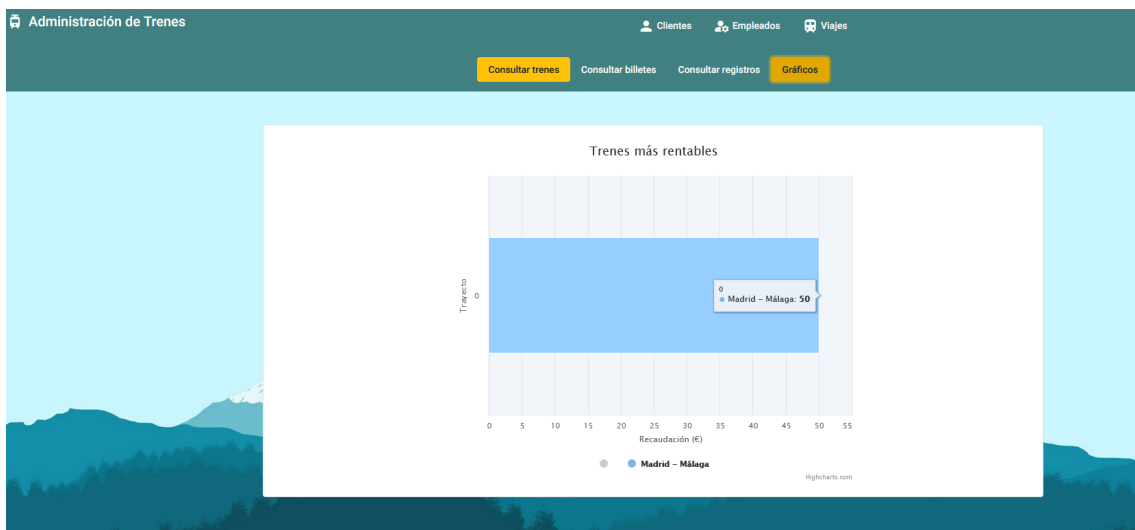
1  ngOnInit(): void {
2      this.getViajesPorCliente()
3      setTimeout(()=>{
4          this.getTotalViajes()
5      }, 500);
6  }
7
8  getTotalViajes() {
9      this.billeteService.getBilletes().subscribe((respuesta) => {
10         this.total_viajes = respuesta.length;
11         this.viajes = respuesta;
12         this.clientes.forEach((cliente) => {
13             let counter = 0;
14             this.viajes.forEach((viaje) => {
15                 cliente._dni == viaje._dni ? counter++ : true;
16             })
17             this.chartOptions.series[0].data.push({
18                 name: cliente._nombre,
19                 y: (counter/this.total_viajes) * 100
20             })
21             this.updateFromInput = true;
22             this.chart.hideLoading();
23         })
24     })
25 }
26
27 getViajesPorCliente() {
28     this.clienteService.getClientes().subscribe((clientes) => {
29         this.clientes = clientes;
30     })
31 }
32 }
33
34

```

Tren más rentable

Cuenta la cantidad de billetes vendidos y el valor de cada uno

Vista usuario:



Código

```

EDITORES ABIERTOS
PROYECTO TRENESCLIENTE-MAIN
> .vscode
> doc
  asdf
  src
    app
      clientes
      empleados
      login
      models
      viajes
        componentes
          crear-billete
          crear-registro
          crear-viaje
          detail-viaje
          get-billetes
          get-registros
          get-viajes
          graficos
            graficos.component.css
            graficos.component.html
            graficos.component.ts 3
          main-viaje
            billetes.service.ts
            registros.service.ts
            viaje.service.ts
            viajes-routing.module.ts
            viajes.module.ts
            app-routing.module.ts
            app.component.css
            app.component.html
            app.component.spec.ts
            app.component.ts
            app.module.ts
            is-logged.guard.ts
            is-not-logged.guard.ts
            login.service.ts
          assets
          environments
        ESQUEMA
        LÍNEA DE TIEMPO
src > app > viajes > componentes > graficos > graficos.component.ts > GraficosComponent > chartOptions
14 export class GraficosComponent implements OnInit {
15   todos_viajes!: any[];
16
17   chart;
18   updateFromInput = false;
19   chartConstructor = "chart";
20   chartCallback;
21   Highcharts: typeof Highcharts = Highcharts;
22   chartOptions: any = {
23     chart: {
24       type: 'bar',
25       height: "500px"
26     },
27     title: {
28       text: 'Trenes más rentables'
29     },
30     yAxis: {
31       min: 0,
32       title: {
33         text: 'Recaudación (€)'
34       }
35     },
36     xAxis: {
37       title: {
38         text: "Trayecto"
39       },
40       categories: [
41
42       ],
43       crosshair: true
44     },
45     accessibility: {
46       point: {
47         valueSuffix: '€'
48       }
49     },
50     plotOptions: {
51       column: {
52         depth: 25
53       }
54     },
55     series: [{
56       type: "column",
57       name: '',
58       colorByPoint: true,
59       data: []
60     }],
61     exporting: {
62       enabled: true

```

```

EXPLORADOR
... ficos.component.css  graficos.component.html ...viajes...  graficos.component.ts ...viajes... 3  graficos.component.ts ...empleados... 3  graf
EDITORES ABIERTOS
PROYECTOTRENESCLIENTE-MAIN
  .vscode
  doc
  asdf
  src
    app
      clientes
      empleados
      login
      models
      viajes
        componentes
          crear-billete
          crear-registro
          crear-viaje
          detail-viaje
          get-billetes
          get-registros
          get-viajes
          graficos
            graficos.component.css
            graficos.component.html
            graficos.component.ts 3
          main-viaje
            billetes.service.ts
            registros.service.ts
            viaje.service.ts
            viajes-routing.module.ts
            viajes.module.ts
            app-routing.module.ts
            app.component.css
            app.component.html
            app.component.spec.ts
            app.component.ts
            app.module.ts
            is-logged.guard.ts
            is-not-logged.guard.ts
            login.service.ts
          assets
          environments
  ESQUEMA
src > app > viajes > componentes > graficos > graficos.component.ts > GraficosComponent > chartOptions
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
constructor(private viajeService: ViajeService, private billeteService: BilletesService) {
  const self = this;

  this.chartCallback = chart => {
    self.chart = chart;
  };
}

ngOnInit(): void {
  this.viajeService.getViajes().subscribe((viajes) => {
    this.todos_viajes = viajes;
  })

  setTimeout(() => {
    this.billeteService.getBilletes().subscribe((billetes) => {
      let data = [];
      let categories = [];

      this.todos_viajes.forEach(viaje => {
        let pagado = 0;
        billetes.forEach(billete => {
          if(billete_idTrenPasajeros == viaje_id) {
            pagado += billete_precio;
          }
        })

        if(viaje_tipoObjeto=="pasajeros") {
          categories.push(viaje_origen + " - " + viaje_destino);

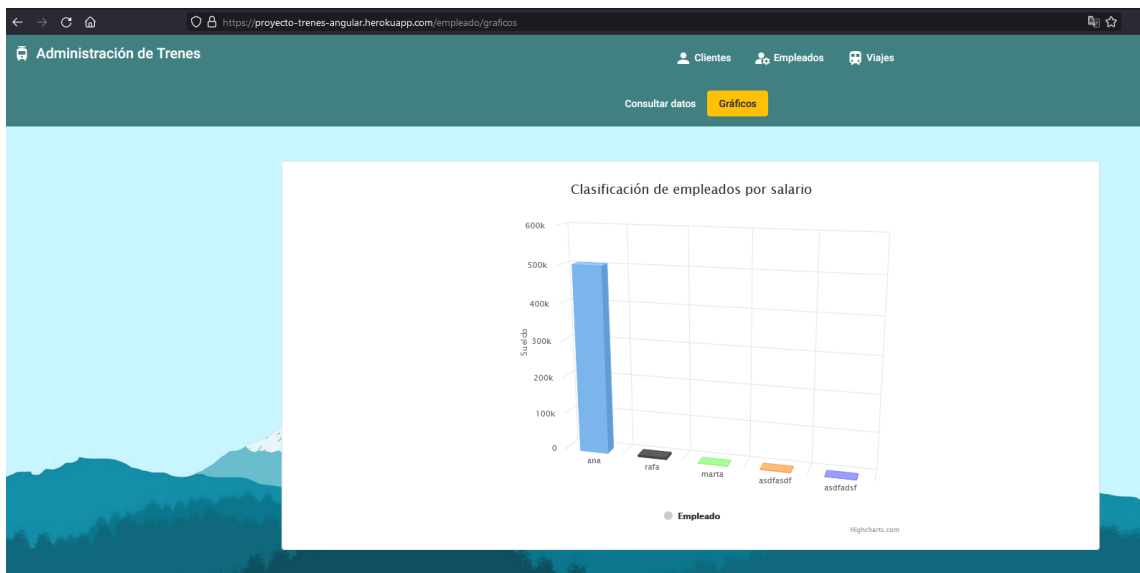
          this.chartOptions.series.push({
            name: viaje_origen + " - " + viaje_destino,
            data: [pagado]
          });
        }
      })

      this.updateFromInput = true;
      this.chart.hideLoading();
    })
  }, 500);
}

```

Sueldos de empleados

Vista usuarios





Código

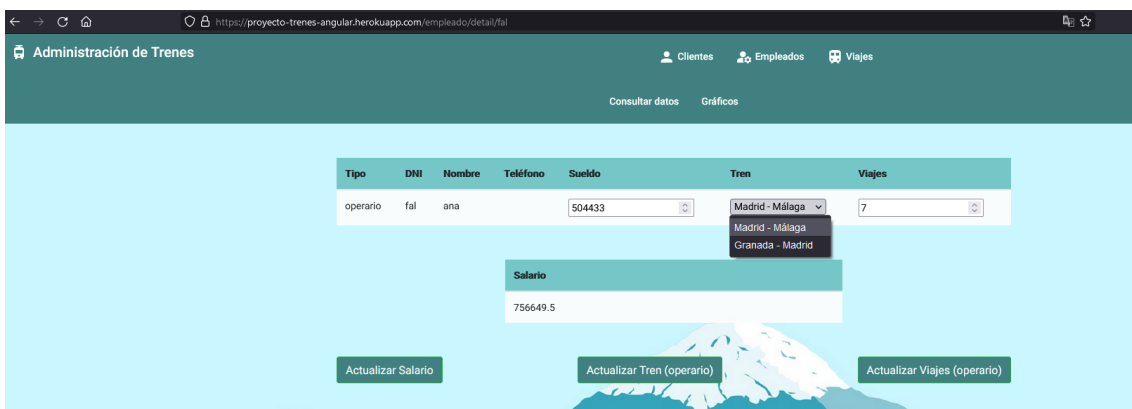
The screenshot shows a code editor with a file explorer on the left and a main editor on the right. The file explorer displays a project structure with folders like 'src', 'components', 'services', and 'utils'. The main editor displays the content of 'graficos.component.ts', which includes a TypeScript class 'EmpleadoService' and a chart configuration for 'Highcharts'.

```
constructor(private empleadoService: EmpleadoService) {  
  const self = this;  
  
  this.chartCallback = chart => {  
    self.chart = chart;  
  };  
}  
  
ngOnInit(): void {  
  this.empleadoService.getEmpleados().subscribe((empleados) => {  
    console.log(empleados)  
    this.todos_empleados = empleados;  
    let data = [];  
    let categories = [];  
  
    this.todos_empleados.forEach((empleado) => {  
      data.push({  
        name: empleado._nombre,  
        y: empleado._sueldo  
      });  
      categories.push(empleado._nombre);  
    })  
    this.chartOptions.series[0].data = data;  
    this.chartOptions.xAxis.categories = categories;  
  
    this.updateFromInput = true;  
    this.chart.hideLoading();  
  });  
}
```

Actualizar

Empleados

Vista usuario



Administración de Trenes

Cientes Empleados Viajes

Consultar datos Gráficos

Tipo	DNI	Nombre	Teléfono	Sueldo	Tren	Viajes
operario	fal	ana		504433	Madrid - Málaga Madrid - Málaga Granada - Madrid	7

Salario

756649.5

Actualizar Salario Actualizar Tren (operario) Actualizar Viajes (operario)

Código

```

EXPLORADOR
src > app > empleados > componentes > detail-empleado > detail-empleado.component.html
1 <table class="table table-active w-50 mx-auto mt-5">
2 <thead style="background-color: #185a75; color: white;">
3 <tr>
4 <th>Tipo</th>
5 <th>DNI</th>
6 <th>Nombre</th>
7 <th>Telefono</th>
8 <th>Suelo</th>
9 <th *ngIf="empleado._tipoObjeto == 'limpiador' || empleado._tipoObjeto == 'revisor'">Horas</th>
10 <th *ngIf="empleado._tipoObjeto == 'operario' || empleado._tipoObjeto == 'limpiador'">Tren</th>
11 <th *ngIf="empleado._tipoObjeto == 'operario' || empleado._tipoObjeto == 'revisor'">Viajes</th>
12 </tr>
13 </thead>
14 <tbody style="background-color: #f2f2f2;">
15 <tr>
16 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._suelo}}</td>
17 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._dni}}</td>
18 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._nombre}}</td>
19 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._telefono}}</td>
20 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._suelo}}</td>
21 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._tren}}</td>
22 <td *ngIf="empleado._tipoObjeto == 'limpiador'">{{empleado._viajes}}</td>
23 </tr>
24 <tr>
25 <td *ngIf="empleado._tipoObjeto == 'operario' || empleado._tipoObjeto == 'limpiador'">
26 <select [(ngModel)]="empleado._tren">
27 <option *ngFor="let tren of trenes" [value]="tren.id" [selected]="tren.id === empleado._tren">{{tren.destino}}
28 </select>
29 </td>
30 <td *ngIf="empleado._tipoObjeto == 'operario' || empleado._tipoObjeto == 'limpiador'">
31 <input type="text" [(ngModel)]="empleado._viajes">
32 </td>
33 </tr>
34 </tbody>
35 </table>
36 <table class="table table-active w-25 mx-auto mt-5">
37 <thead style="background-color: #185a75; color: white;">
38 <tr>
39 <th>Salario</th>
40 </tr>
41 </thead>
42 <tbody style="background-color: #f2f2f2;">
43 <tr>
44 <td *ngIf="empleado._tipoObjeto == 'operario' || empleado._tipoObjeto == 'limpiador'">
45 <input type="text" [(ngModel)]="empleado._salario">
46 </td>
47 </tr>
48 </tbody>
49 </table>
50 <div class="w-50 mx-auto d-flex justify-content-between mt-5">
51 <button (click)="actualizarSalario()" class="btn btn-success" style="background-color: #28a745; color: white;">Actualizar Salario</button>
52 <button (click)="actualizarTrenOperario()" class="btn btn-success" style="background-color: #28a745; color: white;" *ngIf="empleado._tipoObjeto == 'operario'">Actualizar

```

```

src > app > empleados > componentes > detail-empleado > detail-empleado.component.ts
46
47
48 actualizarSalario() {
49   this.empleadoService.actualizarSalarioEmpleado(this.empleado_dni, this.empleado_suelo).subscribe((respuesta) => {
50     this._snackBar.open("Salario actualizado correctamente", "", {
51       duration: 1000,
52       verticalPosition: "top"
53     });
54     this.router.navigateByUrl("/empleado")
55   });
56 }
57
58 actualizarTrenOperario() {
59   this.empleadoService.actualizarTrenOperario(this.empleado_dni, this.empleado_tren).subscribe((respuesta) => {
60     this._snackBar.open("Tren de operario actualizado correctamente", "", {
61       duration: 1000,
62       verticalPosition: "top"
63     });
64     this.router.navigateByUrl("/empleado")
65   });
66 }
67
68 actualizarViajesOperario() {
69   this.empleadoService.actualizarViajesOperario(this.empleado_dni, this.empleado_viajes).subscribe((respuesta) => {
70     this._snackBar.open("Viajes del operario actualizados correctamente", "", {
71       duration: 1000,
72       verticalPosition: "top"
73     });
74     this.router.navigateByUrl("/empleado")
75   });
76 }
77
78 actualizarViajeHoras() {
79   this.empleadoService.actualizarViajeHoras(this.empleado_dni, this.empleado_viajes, parseInt(this.empleado_horas)).subscribe((respuesta) => {
80     this._snackBar.open("Horas y viajes actualizados correctamente", "", {
81       duration: 1000,
82       verticalPosition: "top"
83     });
84     this.router.navigateByUrl("/empleado")
85   });
86 }
87
88 actualizarHorasTren() {
89   this.empleadoService.actualizarHorasTren(this.empleado_dni, this.empleado_horas, this.empleado_tren.toString()).subscribe((respuesta) => {
90     this._snackBar.open("Horas y tren actualizados correctamente", "", {
91       duration: 1000,
92       verticalPosition: "top"
93     });
94     this.router.navigateByUrl("/empleado")
95   });
96 }

```


Viajes

Vista usuario

Tipo	Origen	Destino	Pasajeros	Plazas	Precio
pasajeros	Madrid	Málaga	10000	60000	10

Actualizar Origen Actualizar Destino Actualizar Precio

Código

```

src > app > viajes > componentes > detail-viaje > detail-viaje.component.html
1 <table class="table table-active w-50 mx-auto mt-5">
2   <thead style="background-color: #1a3d54; color: white;">
3     <tr>
4       <th>Tipo</th>
5       <th>Origen</th>
6       <th>Destino</th>
7       <th>Pasajeros</th>
8       <th>Plazas</th>
9       <th>Precio</th>
10    </tr>
11    <tr>
12      <th>Tipo</th>
13      <th>Origen</th>
14      <th>Destino</th>
15      <th>Pasajeros</th>
16      <th>Plazas</th>
17      <th>Precio</th>
18    </tr>
19  </thead>
20  <tbody style="background-color: #f2f2f2;">
21    <tr>
22      <td>{{(viaje._tipoObjeto) | tipoObjeto}}</td>
23      <td>{{(viaje._origen) | origen}}</td>
24      <td>{{(viaje._destino) | destino}}</td>
25      <td>{{(viaje._pasajeros) | pasajeros}}</td>
26      <td>{{(viaje._plazas) | plazas}}</td>
27      <td>{{(viaje._precio) | precio}}</td>
28    </tr>
29  </tbody>
30 </table>
31
32 <div class="mt-5 d-flex justify-content-between">
33   <button (click)="actualizarOrigen()" class="btn btn-success">Actualizar Origen</button>
34   <button (click)="actualizarDestino()" class="btn btn-success">Actualizar Destino</button>
35   <button (click)="actualizarCarga()" class="btn btn-success">Actualizar Carga</button>
36   <button (click)="actualizarPrecio()" class="btn btn-success">Actualizar Precio</button>
37 </div>

```

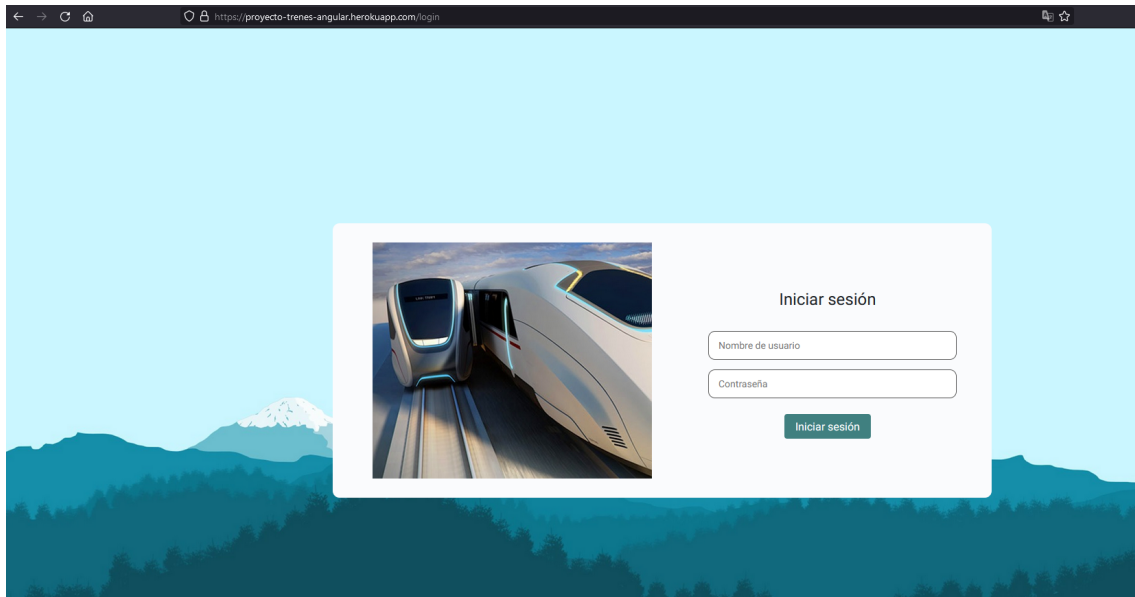
```

src > app > viajes > componentes > detail-viaje > detail-viaje.component.ts
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Login

Vista del usuario



Código

```

EDITORES ABIERTOS
PROYECTOTRENESCLIENTE-MAIN
.vscode
doc
asfd
src
app
  clientes
  empleados
  login
    login.component.css
    login.component.html
    login.component.ts
  models
  viajes
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts

src > app > login > login.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2  import { LoginService } from '../login.service';
3
4  @Component({
5    selector: 'app-login',
6    templateUrl: './login.component.html',
7    styleUrls: ['./login.component.css']
8  })
9  export class LoginComponent implements OnInit {
10
11    user!: string;
12    passwd!: string;
13    constructor(private loginService: LoginService) { }
14
15    ngOnInit(): void {
16
17    }
18
19    login(): void {
20      this.loginService.login(this.user, this.passwd)
21    }
22
23  }

```

```

EDITORES ABIERTOS
PROYECTOTRENESCLIENTE-MAIN
.vscode
doc
asfd
src
app
  clientes
  empleados
  login
    login.component.css
    login.component.html
    login.component.ts
  models
  viajes
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts

src > app > login > login.component.html > ...
1  <div class="mx-auto w-50 login-container">
2  <div class="d-flex justify-content-around align-items-center">
3  
4  <div class="form-part">
5  <h1>Iniciar sesión</h1>
6  <input type="text" [(ngModel)]="user" placeholder="Nombre de usuario">
7  <input type="password" [(ngModel)]="passwd" placeholder="Contraseña">
8  <button mat-button class="btn btn-success" (click)="login()" style="background-color: #408080">Iniciar sesión</button>
9  </div>
10 </div>
11 </div>
12

```

```

EDITORES ABIERTOS
PROYECTOTRENESCLIENTE-MAIN
> .vscode
> doc
> asdf
> src
  > app
    > clientes
    > empleados
    > login
      login.component.css
      login.component.html
      login.component.ts
    > models
    > viajes
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      is-logged.guard.ts
      is-not-logged.guard.ts
      login.service.ts
    > assets

src > app > login.service.ts > LoginService > login
1 import { Injectable } from '@angular/core';
2 import { Router } from '@angular/router';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class LoginService {
8   private _is_admin;
9   constructor(private router: Router) {}
10  get is_admin() {
11    return this._is_admin;
12  }
13
14  set is_admin(value) {
15    this._is_admin = value;
16  }
17
18  login(user: string, pass: string) {
19    if(user === "admin" && pass === "admin") {
20      this.is_admin = true;
21      this.router.navigateByUrl("/viaje");
22    }
23    else {
24      this.is_admin = false;
25    }
26  }
27 }
28

```

Wards es una función intermediaria entre el código y la vista del usuario

```

PROYECTOTRENESCLIENTE-MAIN
> .vscode
> doc
> asdf
> src
  > app
    > clientes
    > empleados
    > login
      login.component.css
      login.component.html
      login.component.ts
    > models
    > viajes
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      is-logged.guard.ts
      is-not-logged.guard.ts

1 import { Injectable } from '@angular/core';
2 import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';
3 import { Observable } from 'rxjs';
4 import { LoginService } from './login.service';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class IsLoggedGuard implements CanActivate {
10   constructor(private loginService: LoginService, private router: Router) {}
11 }
12
13 canActivate(
14   route: ActivatedRouteSnapshot,
15   state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
16
17   if(this.loginService.is_admin){
18     return true;
19   }
20   else {
21     return this.router.navigateByUrl("/login").then(() => false);
22   }
23 }
24
25

```

```

EDITORES ABIERTOS
PROYECTOTRENESCLIENTE-MAIN
> .vscode
> doc
> asdf
> src
  > app
    > clientes
    > empleados
    > login
      login.component.css
      login.component.html
      login.component.ts
    > models
    > viajes
      app-routing.module.ts
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      is-logged.guard.ts
      is-not-logged.guard.ts
      login.service.ts

src > app > is-not-logged.guard.ts > ...
1 import { Injectable } from '@angular/core';
2 import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';
3 import { Observable } from 'rxjs';
4 import { LoginService } from './login.service';
5
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class IsNotLoggedGuard implements CanActivate {
10   constructor(private loginService: LoginService, private router: Router) {}
11 }
12
13 canActivate(
14   route: ActivatedRouteSnapshot,
15   state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
16
17   if(!this.loginService.is_admin){
18     return true;
19   }
20   else {
21     return this.router.navigateByUrl("/viaje").then(() => false);
22   }
23 }
24
25

```

Método de clases

Formato de fecha

```

src > app > models > registro.ts > Registro
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84

set origen(value: string) {
  this._origen = value;
}

get destino(): string {
  return this._destino;
}

set destino(value: string) {
  this._destino = value;
}

get fecha(): Date {
  return this._fecha;
}

set fecha(value: Date) {
  this._fecha = value;
}

getFechaFormat() {
  let fecha = new Date(this._fecha);
  return fecha.getDate() + "/" + fecha.getMonth() + 1 + "/" + fecha.getFullYear();
}

```