

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Análisis y Diseño de Sistemas 1  
Ing. Ivonne Aldana  
Aux. Brandon Pedroza  
Sección A-



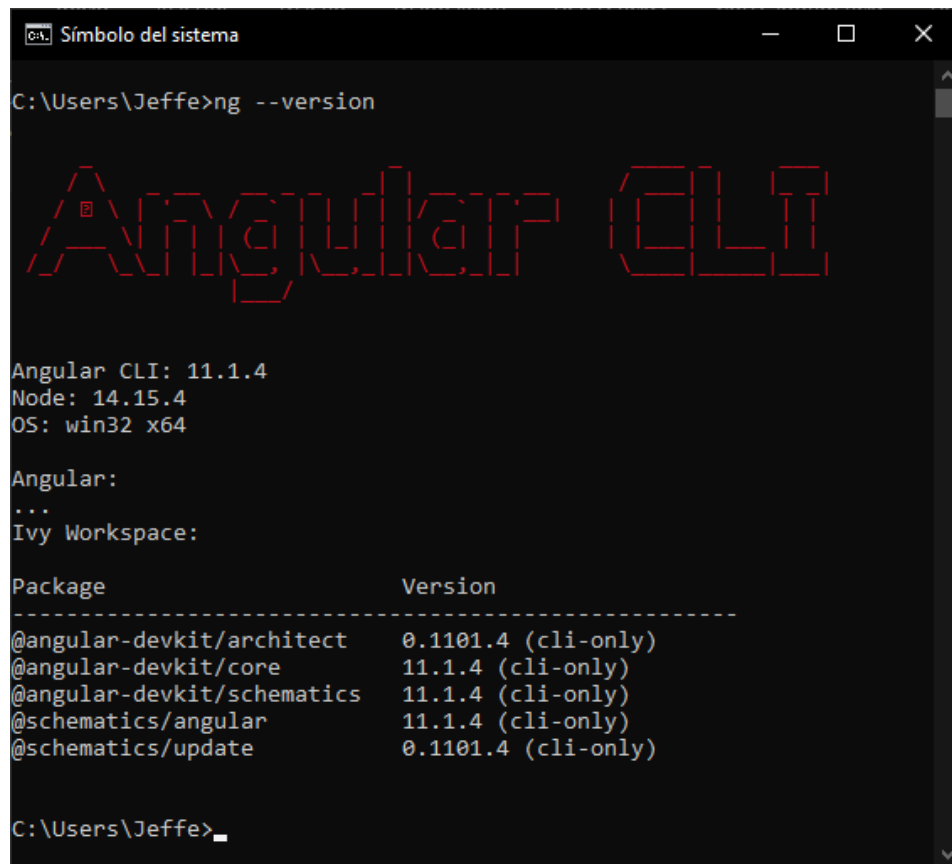
Integrantes	Carne
<i>Mario Roberto Cojolon Shoc</i>	201314359
<i>José Alejandro Grande Marín</i>	201602855
<i>Cristian Manances Juárez Juárez</i>	201700529
<i>Jefferson Geovanny Moreno Perez</i>	201603047
<i>Hayrton Omar Ixpata Coloch</i>	201313875
<i>Miguel Angel Solis Yantuche</i>	201700543

Guatemala 12 de abril del 2021

## REQUERIMIENTOS DEL SISTEMA

Para el correcto funcionamiento de la aplicación *Banca Virtual* se requieren ciertos programas y librerías.

- Sistema Operativo Windows 10.
- Typescript instalado de manera global V 4.2.4 o superior.
- Angular instalado V 10.0 o superior.
- Nodejs Instalado V 14.12 o superior.
- Gestor de Base de Datos PostgreSQL 13 o superior.



```
Símbolo del sistema
C:\Users\Jeffe>ng --version

Angular CLI
Angular CLI: 11.1.4
Node: 14.15.4
OS: win32 x64

Angular:
...
Ivy Workspace:

Package                                  Version
-----
@angular-devkit/architect               0.1101.4 (cli-only)
@angular-devkit/core                    11.1.4 (cli-only)
@angular-devkit/schematics              11.1.4 (cli-only)
@schematics/angular                     11.1.4 (cli-only)
@schematics/update                       0.1101.4 (cli-only)

C:\Users\Jeffe>
```

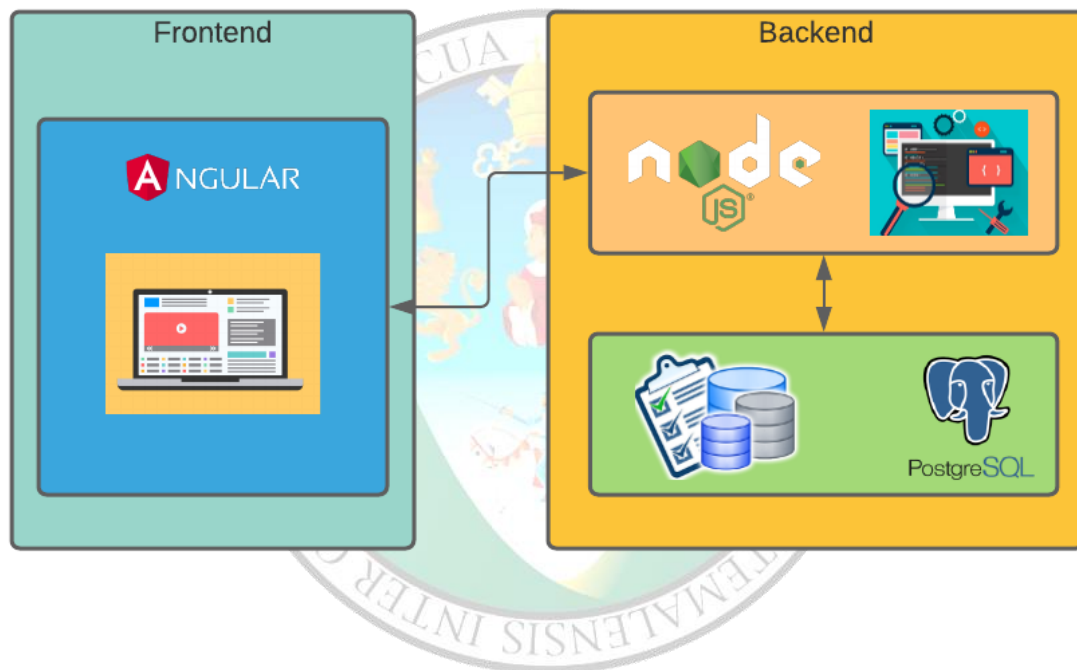
Para un correcto funcionamiento de las librerías, se recomienda aplicar *npm install* antes de inicializar los servidores de Angular y Nodejs (Frontend y Backend respectivamente):

## ARQUITECTURA

Se hizo uso de una arquitectura de 3 capas, esto con el fin de obtener una mejor comunicación entre las diferentes tecnologías.

- Para el Frontend se hizo uso del Framework Angular 10.0
- Para el Backend se hizo uso de Nodejs con el Framework Express para el servidor.
- Para la Base de datos se hizo uso de Postegresql 13.

Además de todo se hizo uso de conceptos de API REST para un tráfico de información óptimo.



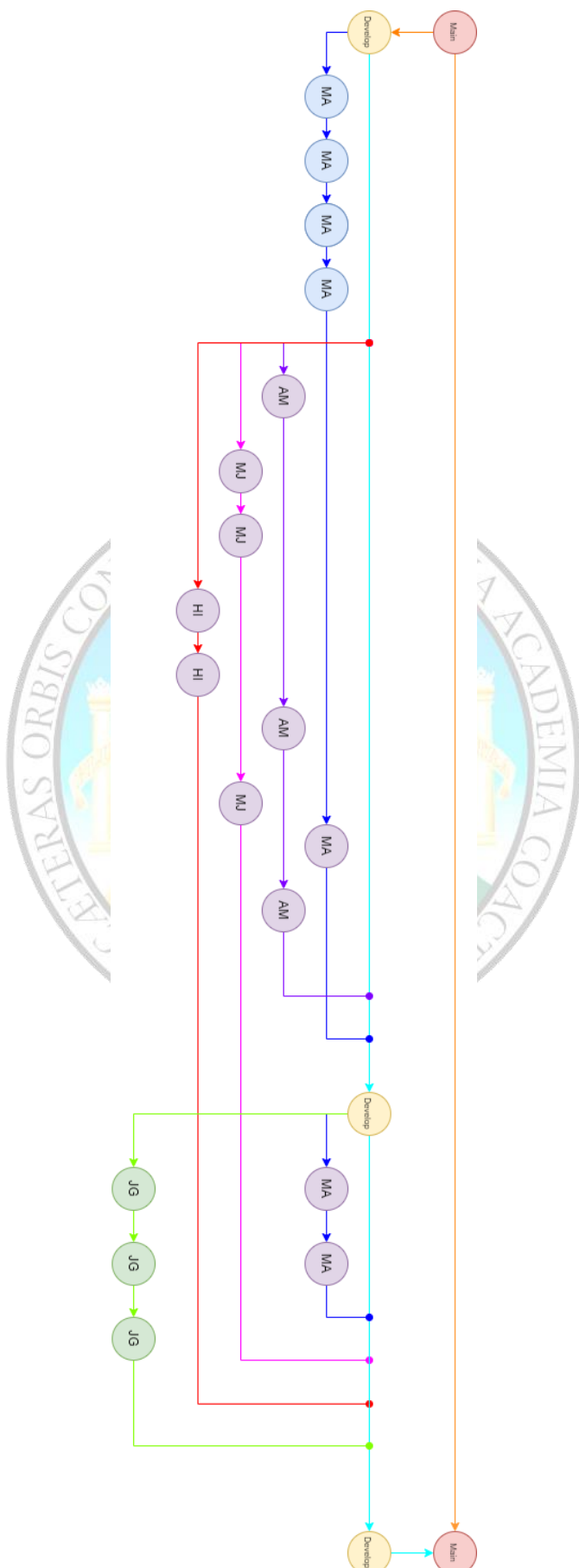
## REPOSITORIO

Para un óptimo manejo de las versiones, se hizo uso de GitHub con el flujo de trabajo Git Flow, lo cual permite la paralelización del desarrollo mediante ramas independientes para la preparación, mantenimiento y publicación de versiones del proyecto, así como soporta la reparación de errores en cualquier momento.

Link del Proyecto: [https://github.com/JeffGeoMP/Banca\\_Virtual.git](https://github.com/JeffGeoMP/Banca_Virtual.git)

El cual es un proyecto público para cualquier mejora que se presente.

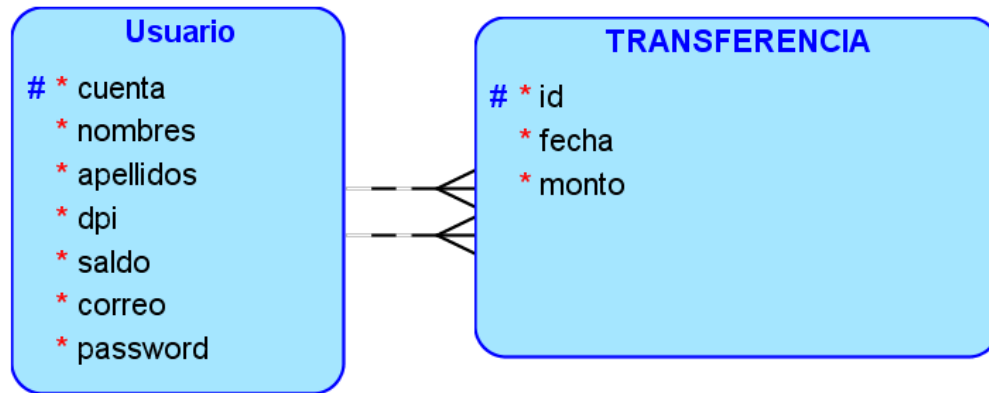
A continuación, se presenta un Workflow de las ramificaciones del repositorio antes citado.



## APLICACIÓN

### Base de Datos:

Diagrama de la base de datos utilizada por la aplicación.

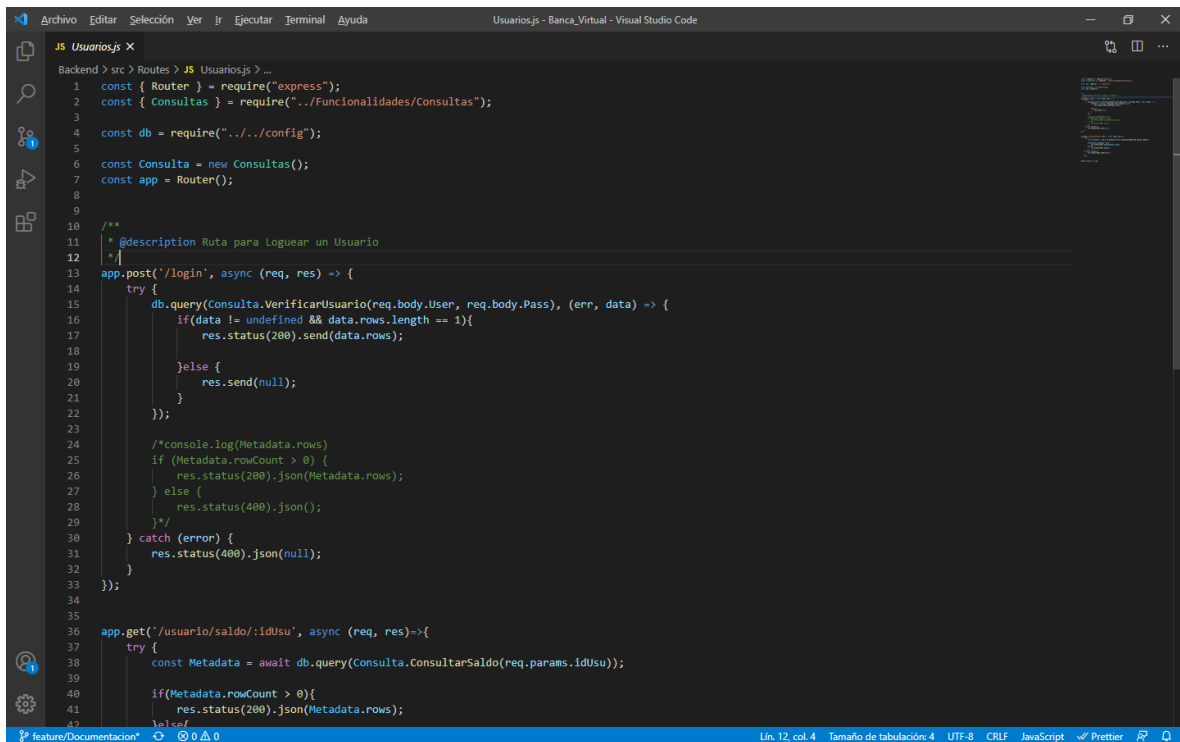


### Servidor del Backend:

Servidor usado para la aplicación.

```
1 const db = require('./config');
2 const express = require('express');
3 const morgan = require('morgan');
4 const parser = require('body-parser');
5 const cors = require('cors');
6
7
8 /**
9  * Configuraciones del Servidor
10  */
11 const app = express();
12 const port = 3000;
13 app.set('port', port);
14
15
16 /**
17  * Middlewares
18  */
19 app.use(morgan('dev')); //Para correr el server desde npm run dev
20 app.use(parser.json()); //Valida que el intercambio sea de tipo json
21 app.use(parser.urlencoded({ extended:false})); //Subida de Imagenes al server
22 app.use(cors());
23
24 /**
25  * Rutas
26  */
27 app.use(require('./src/Routes/Usuarios'));
28
29 /**
30  * Archivos Estaticos
31  */
32 app.use(express.static('public'));
33
34 /**
35  * Ejecucion del Servidor
36  */
37 app.listen(app.get('port'),()=>{
38   console.log('Api REST corriendo en http://localhost:' + port);
39 });
```

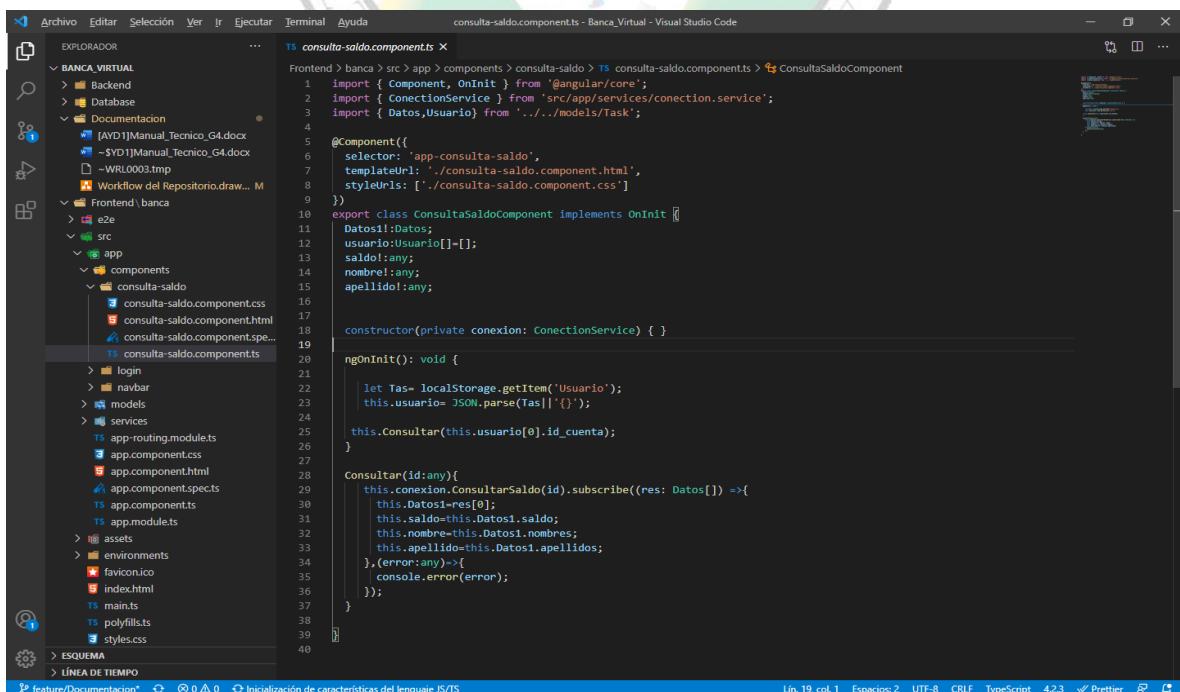
Rutas utilizadas por el servidor.



```
1 const { Router } = require("express");
2 const { Consultas } = require("../Funcionalidades/Consultas");
3
4 const db = require("../config");
5
6 const Consulta = new Consultas();
7 const app = Router();
8
9
10 /**
11  * @description Ruta para Loguear un Usuario
12  */
13 app.post('/login', async (req, res) => {
14   try {
15     db.query(Consulta.VerificarUsuario(req.body.User, req.body.Pass), (err, data) => {
16       if(data != undefined && data.rows.length == 1){
17         res.status(200).send(data.rows);
18       } else {
19         res.send(null);
20       }
21     });
22   } catch (error) {
23     res.status(400).json(null);
24   }
25 });
26
27 /**console.log(Metadata.rows)
28 if (Metadata.rowCount > 0) {
29   res.status(200).json(Metadata.rows);
30 } else {
31   res.status(400).json();
32 }
33 */
34 } catch (error) {
35   res.status(400).json(null);
36 }
37 });
38
39 app.get('/usuario/saldo/:idUser', async (req, res)=>{
40   try {
41     const Metadata = await db.query(Consulta.ConsultarSaldo(req.params.idUsu));
42     if(Metadata.rowCount > 0){
43       res.status(200).json(Metadata.rows);
44     } else {
45       res.status(400).json();
46     }
47   } catch (error) {
48     res.status(400).json(null);
49   }
50 });
```

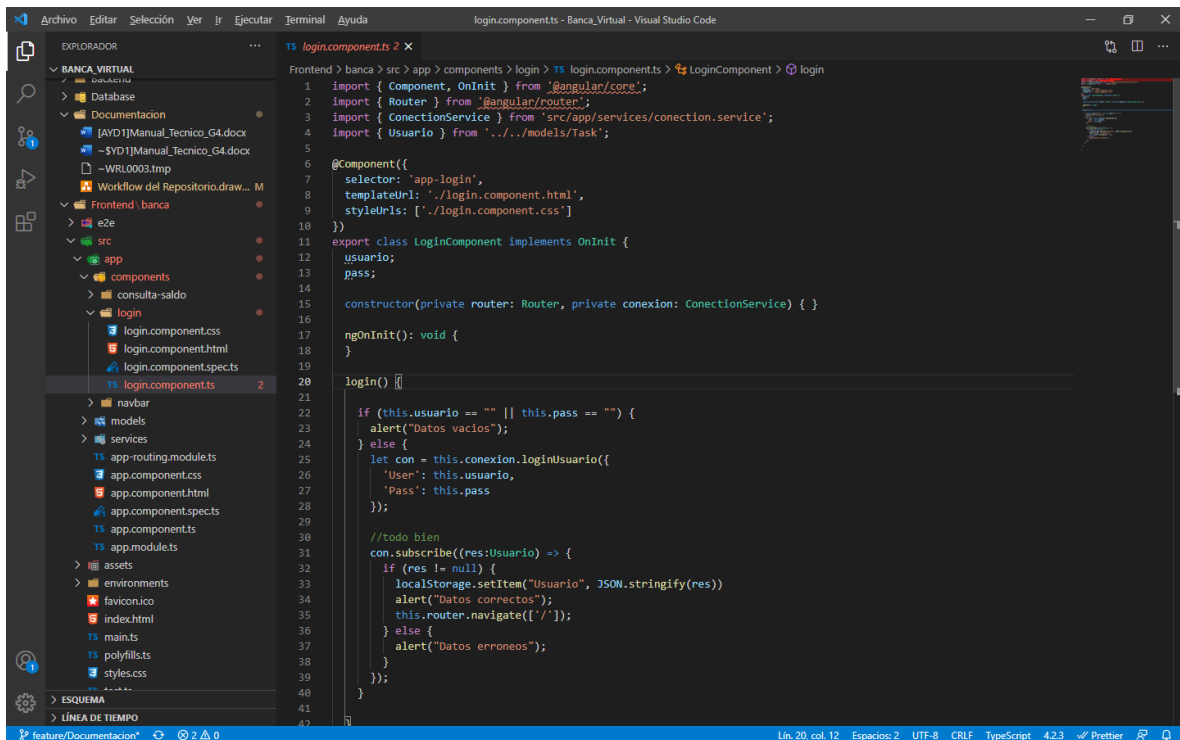
## Componentes del Frontend:

Componente para Consulta de saldo



```
1 import { Component, OnInit } from '@angular/core';
2 import { ConnectionService } from 'src/app/services/connection.service';
3 import { Datos, Usuario } from '../models/Task';
4
5 @Component({
6   selector: 'app-consulta-saldo',
7   templateUrl: './consulta-saldo.component.html',
8   styleUrls: ['./consulta-saldo.component.css']
9 })
10 export class ConsultaSaldoComponent implements OnInit {
11   Datos1: Datos;
12   usuario: Usuario[] = [];
13   saldo: any;
14   nombre: any;
15   apellido: any;
16
17   constructor(private conexion: ConnectionService) {}
18
19   ngOnInit(): void {
20     let Tas = localStorage.getItem('Usuario');
21     this.usuario = JSON.parse(Tas) || [];
22     this.Consultar(this.usuario[0].id_cuenta);
23   }
24
25   Consultar(id: any) {
26     this.conexion.ConsultarSaldo(id).subscribe((res: Datos[]) => {
27       this.Datos1 = res[0];
28       this.saldo = this.Datos1.saldo;
29       this.nombre = this.Datos1.nombre;
30       this.apellido = this.Datos1.apellidos;
31     }, (error: any) => {
32       console.error(error);
33     });
34   }
35 }
```

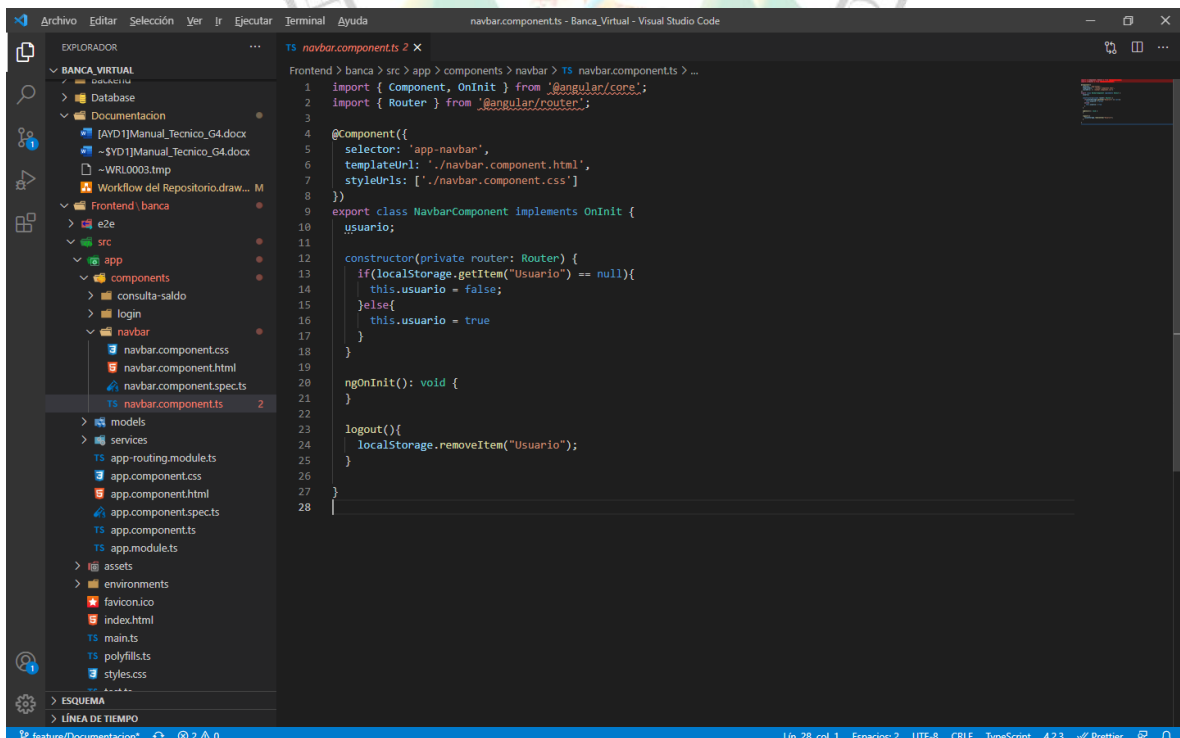
## Componente para el login



```
Frontend > banca > src > app > components > login > TS login.component.ts > LoginComponent > login

1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3 import { ConexionService } from 'src/app/services/conexion.service';
4 import { Usuario } from '../models/Task';
5
6 @Component({
7   selector: 'app-login',
8   templateUrl: './login.component.html',
9   styleUrls: ['./login.component.css']
10 })
11 export class LoginComponent implements OnInit {
12   usuario;
13   pass;
14
15   constructor(private router: Router, private conexion: ConexionService) {}
16
17   ngOnInit(): void {
18   }
19
20   login() {
21
22     if (this.usuario == "" || this.pass == "") {
23       alert("Datos vacios");
24     } else {
25       let con = this.conexion.loginUsuario({
26         'User': this.usuario,
27         'Pass': this.pass
28       });
29
30       //todo bien
31       con.subscribe((res:Usuario) => {
32         if (res != null) {
33           localStorage.setItem("Usuario", JSON.stringify(res));
34           alert("Datos correctos");
35           this.router.navigate(['/']);
36         } else {
37           alert("Datos erroneos");
38         }
39       });
40     }
41   }
42 }
```

## Componente para navbar



```
Frontend > banca > src > app > components > navbar > TS navbar.component.ts > ...

1 import { Component, OnInit } from '@angular/core';
2 import { Router } from '@angular/router';
3
4 @Component({
5   selector: 'app-navbar',
6   templateUrl: './navbar.component.html',
7   styleUrls: ['./navbar.component.css']
8 })
9 export class NavbarComponent implements OnInit {
10   usuario;
11
12   constructor(private router: Router) {
13     if(localStorage.getItem("Usuario") == null){
14       this.usuario = false;
15     }else{
16       this.usuario = true
17     }
18   }
19
20   ngOnInit(): void {
21   }
22
23   logout(){
24     localStorage.removeItem("Usuario");
25   }
26
27 }
28 }
```

## BIBLIOGRAFÍA

- Admin. A. (2016). *Un Caso de Prueba*. [Un Caso de Prueba de ejemplo! - Testing Colombia](#)
- Perez. Antonio (2020). *Como Usar Testing en Angular con Jasmine y Karma*. [Cómo usar Testing en Angular con Jasmine y Karma \(digital55.com\)](#)
- Moreno Jimenez. Yone (2018). *Haciendo test en Angular, con Jasmine y Karma*. [Haciendo tests en Angular, con Jasmine y Karma | by Yone Moreno Jiménez | Medium](#)

