

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



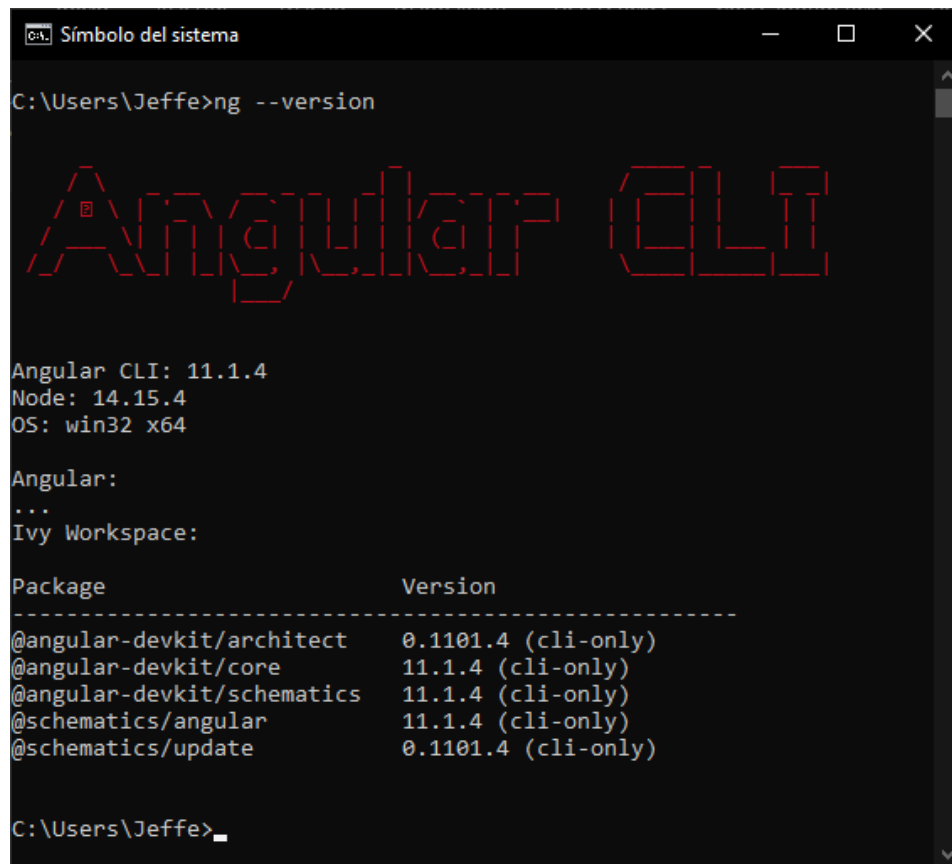
Integrantes	Carne
<b><i>Mario Roberto Cojolon Shoc</i></b>	201314359
<b><i>José Alejandro Grande Marín</i></b>	201602855
<b><i>Cristian Manances Juárez Juárez</i></b>	201700529
<b><i>Jefferson Geovanny Moreno Perez</i></b>	201603047
<b><i>Hayrton Omar Ixpata Coloch</i></b>	201313875
<b><i>Miguel Angel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

Guatemala 27 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.



```
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>
```

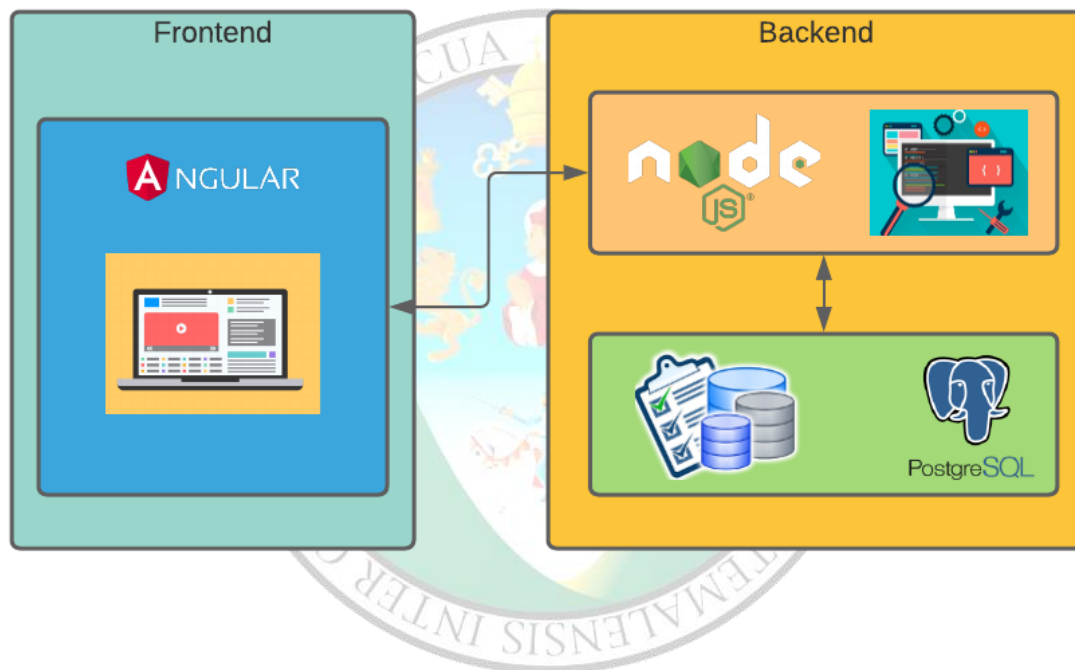
Para un correcto funcionamiento de las librerías, se recomienda aplicar *npm install* antes de inicializar los servidores de Angular y Nodejs (Frontend y Backend respectivamente):

## 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.



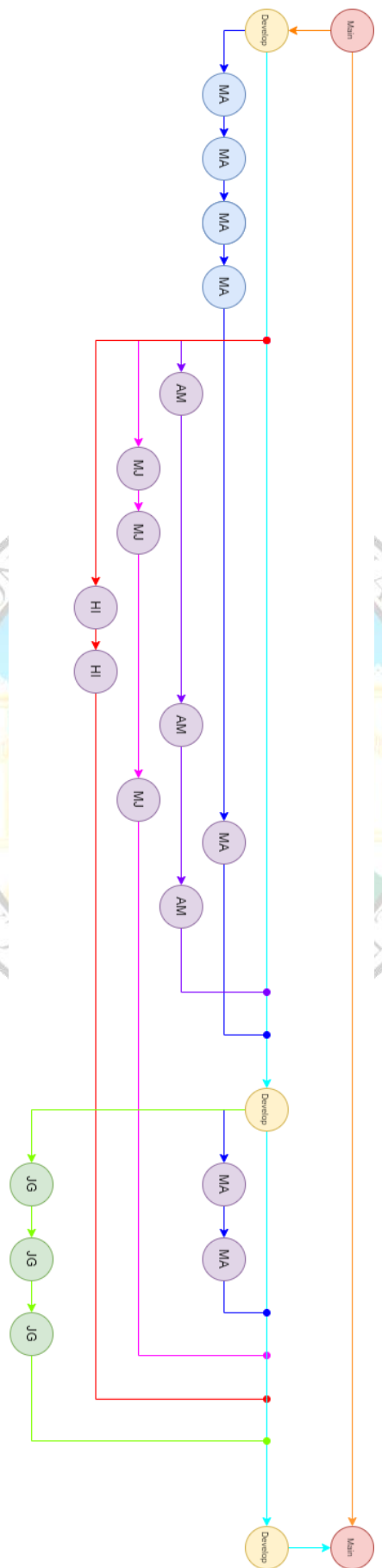
## REPOSITORIO

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.



## CASOS DE PRUEBA

<b>Objetivo del Caso de Prueba</b>	Validar un usuario existen en la plataforma para tener acceso a las diferentes funcionalidades del sistema
<b>Identificador</b>	CP-P3-001
<b>Nombre del Caso</b>	Inicio de Sesión
<b>Precondiciones</b>	<ul style="list-style-type: none"> <li>Ninguno</li> </ul>
<b>Paso</b>	<b>Resultado Esperado</b>
<ul style="list-style-type: none"> <li>Ir a la opción iniciar sesión</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de inicio de sesión</li> </ul>
<ul style="list-style-type: none"> <li>Llenar campos usuario y contraseña</li> </ul>	
<ul style="list-style-type: none"> <li>Click en “<i>Iniciar Sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida los campos ingresados en la base de datos y se redirecciona a su perfil.</li> </ul>

<b>Objetivo del Caso de Prueba</b>	Registrar un nuevo usuario en la plataforma
<b>Identificador</b>	CP-P3-002
<b>Nombre del Caso</b>	Registro de Usuario
<b>Precondiciones</b>	<ul style="list-style-type: none"> <li>Ninguno</li> </ul>
<b>Paso</b>	<b>Resultado Esperado</b>
<ul style="list-style-type: none"> <li>Ir a la opción Registrar Usuario</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de registro de usuario</li> </ul>
<ul style="list-style-type: none"> <li>Llenar los campos correspondientes</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar una alerta si falta algún campo importante</li> </ul>
<ul style="list-style-type: none"> <li>Click en “<i>Registrar</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida la información ingresada, se añade a la base de datos y se redirecciona al login de la aplicación.</li> </ul>

<b>Objetivo del Caso de Prueba</b>	Ver el saldo actual de la cuenta de un usuario en la plataforma
<b>Identificador</b>	CP-P3-003
<b>Nombre del Caso</b>	Consulta de Saldo
<b>Precondiciones</b>	<ul style="list-style-type: none"> <li>Se debe tener una cuenta activa</li> <li>Se debe estar logueado en una cuenta</li> </ul>
<b>Paso</b>	<b>Resultado Esperado</b>
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de inicio de sesión</li> </ul>
<ul style="list-style-type: none"> <li>Llenar Campos Usuario y Contraseña</li> </ul>	

<ul style="list-style-type: none"> <li>Click en “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida los campos ingresados en la base de datos y se redirecciona a su perfil.</li> </ul>
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Ver Saldo</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe visualizar el saldo actual del usuario</li> </ul>

<b>Objetivo del Caso de Prueba</b>	Transferir un monto ingresado de una cuenta a otra cuenta
<b>Identificador</b>	CP-P3-004
<b>Nombre del Caso</b>	Transferencia Monetaria
<b>Precondiciones</b>	<ul style="list-style-type: none"> <li>Se debe tener una cuenta activa</li> <li>Se debe estar logueado en una cuenta</li> <li>Se debe tener el monto ingresado en la cuenta a transferir</li> <li>La cuenta destino debe ser una cuenta activa</li> </ul>
<b>Paso</b>	<b>Resultado Esperado</b>
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de inicio de sesión</li> </ul>
<ul style="list-style-type: none"> <li>Llenar Campos Usuario y Contraseña</li> </ul>	
<ul style="list-style-type: none"> <li>Click en “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida los campos ingresados en la base de datos y se redirecciona a su perfil.</li> </ul>
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Transferir</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se muestra la interfaz para transferir un monto</li> </ul>
<ul style="list-style-type: none"> <li>Llenar los campos (Monto y cuenta destino)</li> </ul>	
<ul style="list-style-type: none"> <li>Click en “<i>Transferir</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida que exista el monto ingresado en la cuenta origen, si esto es valido se transfiere el monto a la cuenta destino, a continuación, se muestra una alerta de que se ha completado la transferencia.</li> </ul>

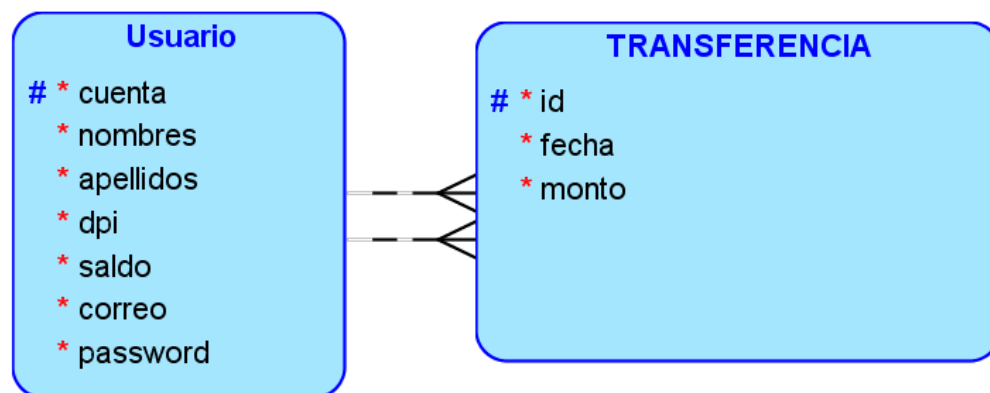
<b>Objetivo del Caso de Prueba</b>	Descargar las transferencias bancarias realizadas por un usuario en un archivo pdf
<b>Identificador</b>	CP-P3-005
<b>Nombre del Caso</b>	Reportes de Transferencias
<b>Precondiciones</b>	<ul style="list-style-type: none"> <li>Se debe tener una cuenta activa</li> <li>Se debe estar logueado en una cuenta</li> <li>Se debe haber realizado al menos una transferencia</li> </ul>

Paso	Resultado Esperado
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de inicio de sesión</li> </ul>
<ul style="list-style-type: none"> <li>Llenar Campos Usuario y Contraseña</li> </ul>	
<ul style="list-style-type: none"> <li>Click en “<i>Iniciar sesión</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se valida los campos ingresados en la base de datos y se redirecciona a su perfil.</li> </ul>
<ul style="list-style-type: none"> <li>Ir a la opción “<i>Reporte</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe mostrar la interfaz de reportes del usuario.</li> </ul>
<ul style="list-style-type: none"> <li>Click en “<i>Descargar Reporte</i>”</li> </ul>	<ul style="list-style-type: none"> <li>Se debe comenzar a descargar un archivo pdf, donde se detallan las transferencias realizadas por el usuario.</li> </ul>

## APLICACIÓN

### Base de Datos:

Diagrama de la base de datos utilizada por la aplicación.



### Servidor del Backend:

Servidor usado para la aplicación.

```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  index.js - Banca_Virtual - Visual Studio Code

JS index.js X
Backend > JS index.js > ...
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.

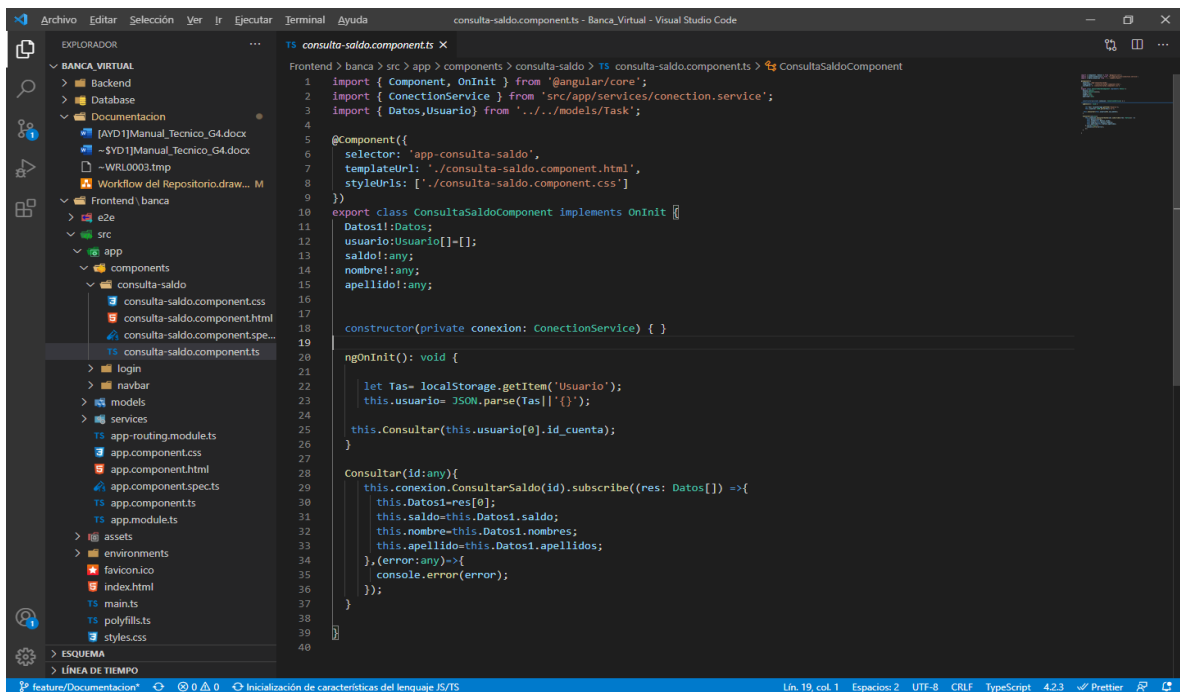
```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  Usuarios.js - Banca_Virtual - Visual Studio Code

JS Usuarios.js X
Backend > src > Routes > JS Usuarios.js > ...
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/:idUsu', 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      }
45    } catch (error) {
46      res.status(400).json(null);
47    }
48  });
```



## Componentes del Frontend:

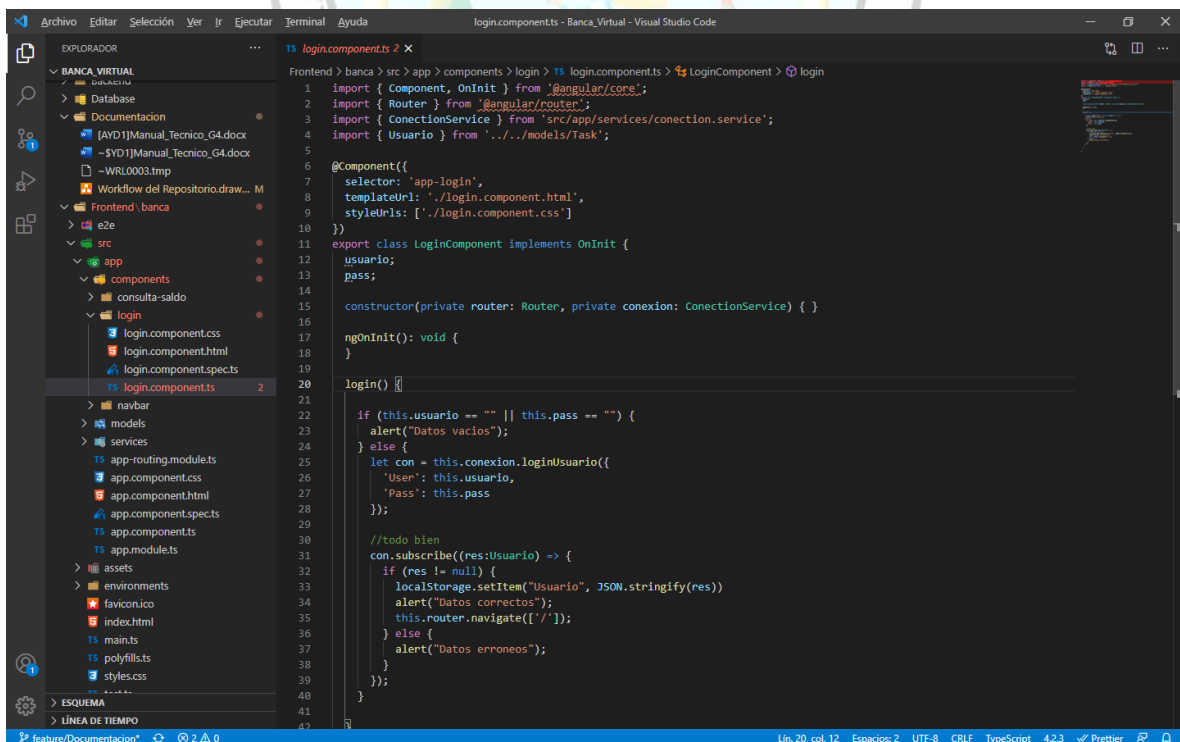
### Componente para Consulta de saldo



```
Frontend > banca > src > app > components > consulta-saldo > TS consulta-saldo.component.ts > ConsultaSaldoComponent

1 import { Component, OnInit } from '@angular/core';
2 import { ConnectionService } from 'src/app/services/connection.service';
3 import { Datos, Usuario } from 'src/app/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
21     let Tas = localStorage.getItem('Usuario');
22     this.usuario = JSON.parse(Tas || '{}');
23
24     this.Consultar(this.usuario[0].id_cuenta);
25   }
26
27   Consultar(id: any) {
28     this.conexion.ConsultarSaldo(id).subscribe((res: Datos[]) => {
29       this.Datos1 = res[0];
30       this.saldo = this.Datos1.saldo;
31       this.nombre = this.Datos1.nombres;
32       this.apellido = this.Datos1.apellidos;
33     }, (error: any) => {
34       console.error(error);
35     });
36   }
37 }
38
39
40
```

### 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 { ConnectionService } from 'src/app/services/connection.service';
4 import { Usuario } from 'src/app/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: ConnectionService) {}
16
17   ngOnInit(): void {}
18
19   login() {
20
21     if (this.usuario == "" || this.pass == "") {
22       alert("Datos vacios");
23     } else {
24       let con = this.conexion.loginUsuario({
25         'User': this.usuario,
26         'Pass': this.pass
27       });
28
29       //todo bien
30       con.subscribe((res: Usuario) => {
31         if (res != null) {
32           localStorage.setItem("Usuario", JSON.stringify(res));
33           alert("Datos correctos");
34           this.router.navigate(['/']);
35         } else {
36           alert("Datos erroneos");
37         }
38       });
39     }
40   }
41 }
42
43
```

## Componente para navbar

```
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
```

## Code Coverage de los Componentes

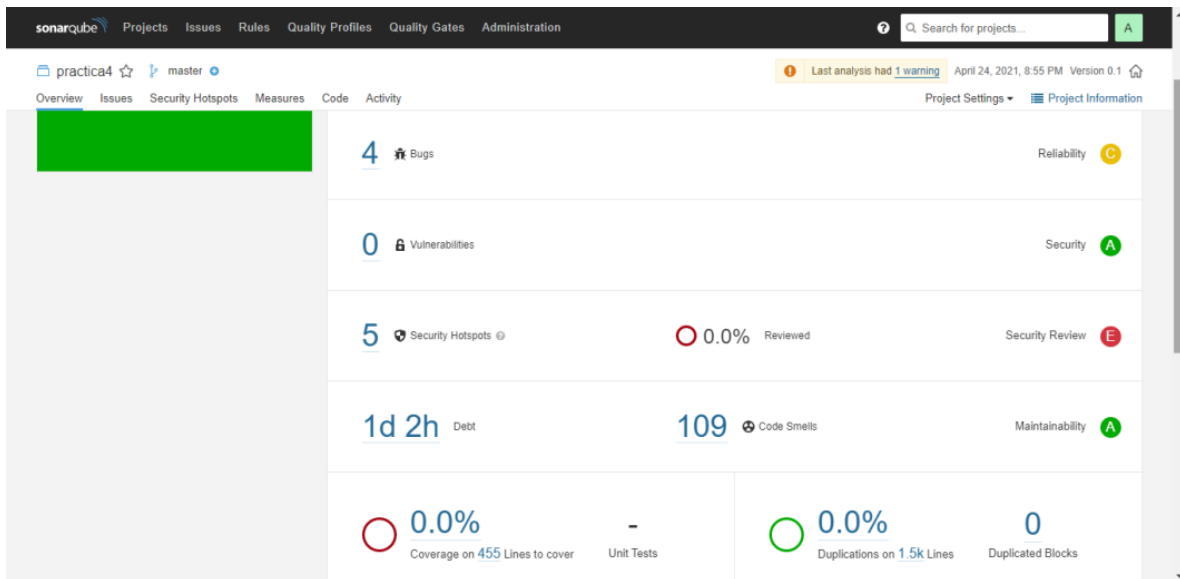
### All files

94.33% Statements 133/141 62.5% Branches 28/32 92.68% Functions 38/41 93.89% Lines 123/131

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File	Statements	Branches	Functions	Lines
src	100%	3/3	100%	0/0
src/app/components/consulta-saldo	100%	12/12	100%	0/0
src/app/components/login	100%	14/14	87.5%	7/8
src/app/components/navbar	100%	7/7	100%	2/2
src/app/components/perfil	100%	9/9	100%	0/0
src/app/components/registro	82.35%	28/34	50%	10/20
src/app/components/reportes	93.33%	28/30	100%	0/0
src/app/components/transferencia	100%	15/15	50%	1/2
src/app/services	100%	17/17	100%	0/0

## SONARQUBE



## 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](#)