

DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL
DESARROLLO DE
SOFTWARE

TALLER UNIDAD 3 FRONTEND

DAVID JULIAN CRIOLLO GÓMEZ

UNIVERSIDAD DE NARIÑO
INGENIERIA DE SISTEMAS
SAN JUAN DE PASTO, 2023

1. Creamos nuestro proyecto que se llamará DonFruver, con la instrucción ng new DonFruver

The screenshot shows the Visual Studio Code interface with the title bar "Bienvenido - Taller3_DonFruver - Visual Studio Code". The left sidebar shows a folder structure for "TALLER3_DONFRUVER" containing files like "Angular", "vscode", "node_modules", "src", "app", "assets", "favicon.ico", "index.html", "main.ts", "style.css", and ".editorconfig". The right sidebar has sections for "Tutorial" (with "Introducción a VS Code" highlighted), "Recente" (listing "FruverFF", "Taller2_Backend", "Prueba", "BACKEND", and "Sistema_Semilleros_Udemy"), and "Terminal". The terminal tab shows the command "ng new DonFruver" being run, with options for Angular routing and stylesheet format. The status bar at the bottom indicates "2:29 p.m. 30/07/2023".

```
PS E:\0_Programas\DIPLOMADO\02\Taller3_DonFruver> ng new DonFruver
Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SASS [ https://sass-lang.com/documentation/syntax#css ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less [ http://lesscss.org ]
```

2. Ejecutamos la instrucción ng Serve --open para correr el servicio

The screenshot shows the Visual Studio Code interface with the title bar ".gitignore - Taller3_DonFruver - Visual Studio Code". The left sidebar shows a folder structure for "TALLER3_DONFRUVER" containing files like ".gitignore", "DonFruver", "angular", "vscode", "node_modules", "src", "app", "assets", "favicon.ico", "index.html", "main.ts", "style.css", and ".editorconfig". The right sidebar has sections for "Tutorial" (with "Introducción a VS Code" highlighted), "PROBLEMAS" (listing "warning: in the working copy of 'src/index.html'", "warning: in the working copy of 'src/main.ts'", etc.), and "Terminal". The terminal tab shows the command "ng serve --open" being run, with a warning about file endings and a note from the Angular Team. The status bar at the bottom indicates "2:31 p.m. 30/07/2023".

```
warning: in the working copy of 'src/index.html', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/main.ts', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/styles.css', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.app.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'tsconfig.spec.json', LF will be replaced by CRLF the next time Git touches it
warning: successfully initialized
PS E:\0_Programas\DIPLOMADO\02\Taller3_DonFruver> cd ..\DonFruver
PS E:\0_Programas\DIPLOMADO\02\Taller3_DonFruver> ng serve --open
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details on how to change this setting, see https://angular.io/analytics. No
Global setting: enabled
Local setting: disabled
Effective status: disabled
Generating browser application bundles (phase: building)...
```

3. Creamos los modelos uno por cada tabla en la base de datos:

```

1 export class UsuarioModel {
2     constructor(public idUsuario: number, public Nombre: string, public Email:string, public Contraseña:string,
3                 public Rol: 'Cliente' | 'Administrador', public Direccion:string, public Ciudad:string, public Telefono:string) {
4         ...
5     }
6 }

```

The screenshot shows the Visual Studio Code interface with the code editor open to the file "usuario.model.ts". The code defines a class "UsuarioModel" with a constructor taking several properties: idUsuario (number), Nombre (string), Email (string), Contraseña (string), Rol (a union type of 'Cliente' or 'Administrador'), Direccion (string), Ciudad (string), and Telefono (string). The code editor has syntax highlighting and a dark theme.

Usuario:

```

1 export class ProductoModel {
2     constructor(public idProducto: string, public Nombre: string, public Descripcion:string, public Precio:number,
3                 public Cantidad_Disponible:number, public Imagen:string, public Categoria:string) {
4         ...
5     }
6 }

```

The screenshot shows the Visual Studio Code interface with the code editor open to the file "producto.model.ts". The code defines a class "ProductoModel" with a constructor taking several properties: idProducto (string), Nombre (string), Descripcion (string), Precio (number), Cantidad_Disponible (number), Imagen (string), and Categoria (string). The code editor has syntax highlighting and a dark theme.

Producto:

```

1 export class PedidoModel {
2     constructor(public idPedido: string, public Usuario_ID: number, public Fecha:Date, public Confirmado:number,
3                 public Total:number) {
4         ...
5     }
6 }

```

The screenshot shows the Visual Studio Code interface with the code editor open to the file "pedido.model.ts". The code defines a class "PedidoModel" with a constructor taking several properties: idPedido (string), Usuario_ID (number), Fecha (Date), Confirmado (number), and Total (number). The code editor has syntax highlighting and a dark theme.

Pedido:

```

1 export class DetalleModel {
2     constructor(public idDetalle: string, public Producto_ID: number, public Cantidad:number, public Precio:number,
3                 public SubTotal:number) {
4         ...
5     }
6 }

```

The screenshot shows the Visual Studio Code interface with the code editor open to the file "detalle.model.ts". The code defines a class "DetalleModel" with a constructor taking several properties: idDetalle (string), Producto_ID (number), Cantidad (number), Precio (number), and SubTotal (number). The code editor has syntax highlighting and a dark theme.

Detalle:

```

src > app > shared > detalle.model.ts > DetalleModel
1  export class DetalleModel {
2    constructor(public idDetalle: string, public Pedido_ID: string, public Producto_ID:string, public Cantidad:number,
3               public Subtotal:number) {
4   }
5 }
6 You, anteayer * ModelosServicios

```

4. Creamos los servicios que permitirán conexión con la el backend y el trabajo con la base de datos, en un primer momento se crearon cuatro servicios:

- Producto
- Pedido
- Usuario
- Detalle-Pedido

Después se agregaron dos más:

- Correo
- Auth

```

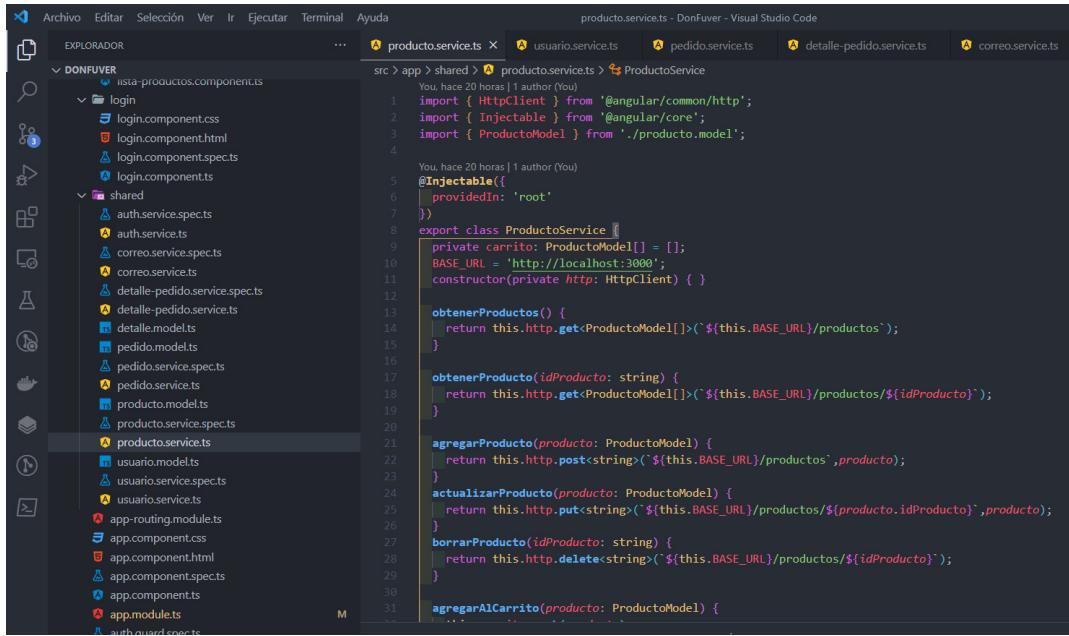
Explorador
TALLER-DONFRUVER
  > VSCode
  > node_modules
  > src
    > app
      > shared
        detalle-pedido.service.ts
        detalle-pedido.service.spec.ts
        detalle.models.ts
        pedido.models.ts
        pedido.service.ts
        pedido.service.spec.ts
        producto.models.ts
        producto.service.ts
        producto.service.spec.ts
        usuario.models.ts
        usuario.service.ts
        usuario.service.spec.ts
      app-routing.module.ts
      app.component.ts
      app.component.html
      app.component.spec.ts
      app.component.ts
      assets
      favicon.ico
      index.html
      main.ts
      styles.css
      editorconfig
      .gitignore
      angular.json
      package-lock.json
      narkane.json
      .esquema
      linea-de-tiempo
      Git Graph

TERMINAL
PS E:\0_Programas\0DPL0M400M2\Taller3_DonFruter> cd ..\DonFruter
CREATE src/app/shared/shared.producto.service.ts (367 bytes)
CREATE src/app/shared/shared.producto.service.spec.ts (137 bytes)
CREATE src/app/shared/shared.usuario.service.ts (162 bytes)
CREATE src/app/shared/shared.usuario.service.spec.ts (136 bytes)
CREATE src/app/shared/shared.pedido.service.ts (135 bytes)
CREATE src/app/shared/shared.pedido.service.spec.ts (123 bytes)
PS E:\0_Programas\0DPL0M400M2\Taller3_DonFruter> ng generate service shared/detalle_pedido
CREATE src/app/shared/shared.detalle-pedido.service.ts (393 bytes)
CREATE src/app/shared/shared.detalle-pedido.service.spec.ts (142 bytes)
PS E:\0_Programas\0DPL0M400M2\Taller3_DonFruter>

```

Se adjunta capturas de parte del contenido de los servicios para no extender el documento:

Producto

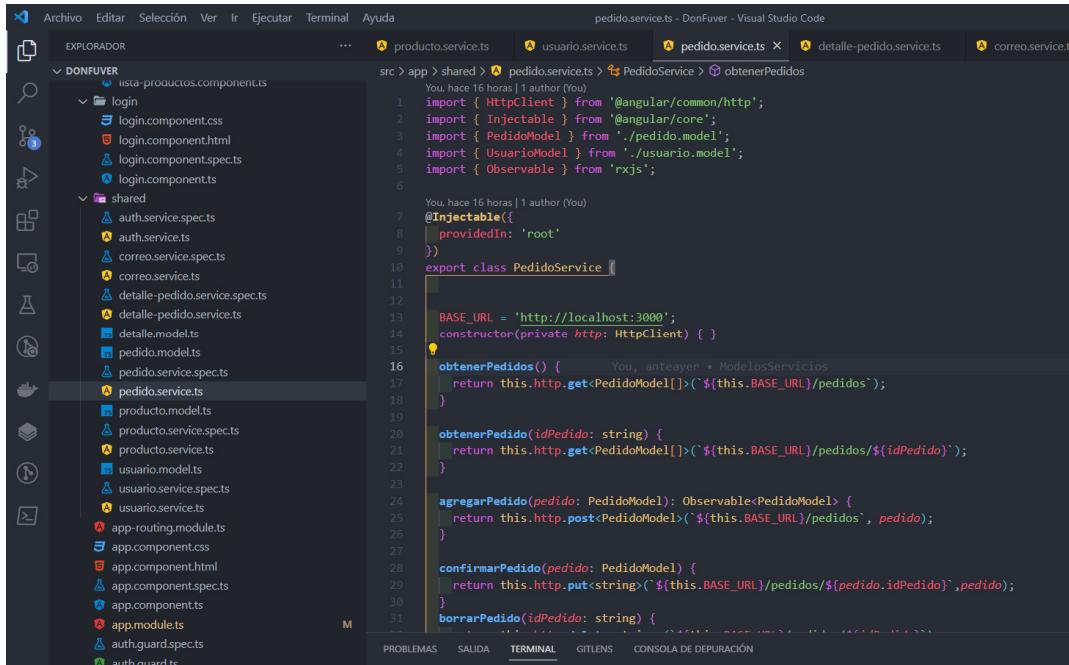


```

src > app > shared > producto.service.ts - DonFuber - Visual Studio Code
You, hace 20 horas | 1 author (You)
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { ProductoModel } from './producto.model';
4
5 You, hace 20 horas | 1 author (You)
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class ProductoService {
10   private carro: ProductoModel[] = [];
11   BASE_URL = 'http://localhost:3000';
12   constructor(private http: HttpClient) { }
13
14   obtenerProductos() {
15     return this.http.get<ProductoModel[]>(`${this.BASE_URL}/productos`);
16   }
17
18   obtenerProducto(idProducto: string) {
19     return this.http.get<ProductoModel>(`${this.BASE_URL}/productos/${idProducto}`);
20   }
21
22   agregarProducto(producto: ProductoModel) {
23     return this.http.post<string>(`${this.BASE_URL}/productos`, producto);
24   }
25   actualizarProducto(producto: ProductoModel) {
26     return this.http.put<string>(`${this.BASE_URL}/productos/${producto.idProducto}`, producto);
27   }
28   borrarProducto(idProducto: string) {
29     return this.http.delete<string>(`${this.BASE_URL}/productos/${idProducto}`);
30   }
31
32   agregarAlCarrito(producto: ProductoModel) {
33
34 }

```

Pedido



```

src > app > shared > pedido.service.ts - DonFuber - Visual Studio Code
You, hace 16 horas | 1 author (You)
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { PedidoModel } from './pedido.model';
4 import { UsuarioModel } from './usuario.model';
5 import { Observable } from 'rxjs';
6
7 You, hace 16 horas | 1 author (You)
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class PedidoService {
12
13   BASE_URL = 'http://localhost:3000';
14   constructor(private http: HttpClient) { }
15
16   obtenerPedidos() {
17     return this.http.get<PedidoModel[]>(`${this.BASE_URL}/pedidos`);
18   }
19
20   obtenerPedido(idPedido: string) {
21     return this.http.get<PedidoModel>(`${this.BASE_URL}/pedidos/${idPedido}`);
22   }
23
24   agregarPedido(pedido: PedidoModel): Observable<PedidoModel> {
25     return this.http.post<PedidoModel>(`${this.BASE_URL}/pedidos`, pedido);
26   }
27
28   confirmarPedido(pedido: PedidoModel) {
29     return this.http.put<string>(`${this.BASE_URL}/pedidos/${pedido.idPedido}`, pedido);
30   }
31
32   borrarPedido(idPedido: string) {
33
34 }

```

Usuario

The screenshot shows the 'Explorador' (Explorer) view on the left with files under 'DONFUVER' and 'src/app/shared'. The 'usuario.service.ts' file is open in the editor. It contains code for an 'UsuarioService' class with methods like 'obtenerUsuarios', 'obtenerUsuario', 'agregarUsuario', 'actualizarUsuario', and 'borrarUsuario'. It uses HttpClient to interact with a local host at port 3000.

```
You, anteyer | 1 author (You)
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { UsuarioModel } from './usuario.model';

@.Injectable({
  providedIn: 'root'
})
export class UsuarioService {

  BASE_URL = 'http://localhost:3000';
  constructor(private http: HttpClient) { }

  obtenerUsuarios() {
    return this.http.get<UsuarioModel[]>(`${this.BASE_URL}/usuarios`);
  }

  obtenerUsuario(idUsuario: string) {
    You, anteyer + ModelosServicios
    return this.http.get<UsuarioModel>(`${this.BASE_URL}/usuarios/${idUsuario}`);
  }

  agregarUsuario(usuario: UsuarioModel) {
    return this.http.post<string>(`${this.BASE_URL}/usuarios`, usuario);
  }
  actualizarUsuario(usuario: UsuarioModel) {
    return this.http.put<string>(`${this.BASE_URL}/usuarios/${usuario.idUsuario}`, usuario);
  }
  borrarUsuario(idUsuario: string) {
    return this.http.delete<string>(`${this.BASE_URL}/usuarios/${idUsuario}`);
  }
}
```

Detalle-Pedido

The screenshot shows the 'Explorador' (Explorer) view on the left with files under 'DONFUVER' and 'src/app/shared'. The 'detalle-pedido.service.ts' file is open in the editor. It contains code for a 'DetallePedidoService' class with methods like 'obtenerDetalles', 'obtenerDetalle', 'agregarDetalle', 'actualizarDetalle', and 'borrarDetalle'. It uses HttpClient to interact with a local host at port 3000.

```
You, hace 20 horas | 1 author (You)
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { DetalleModel } from './detalle.model';
import { ProductoModel } from './producto.model';
import { PedidoModel } from './pedido.model';

@.Injectable({
  providedIn: 'root'
})
export class DetallePedidoService {

  BASE_URL = 'http://localhost:3000';
  constructor(private http: HttpClient) { }

  obtenerDetalles() {
    return this.http.get<DetalleModel[]>(`${this.BASE_URL}/detallesP`);
  }

  obtenerDetalle(idDetalles: string) {
    return this.http.get<DetalleModel>(`${this.BASE_URL}/detallesP/${idDetalles}`);
  }

  agregarDetalle(detalle: DetalleModel) {
    return this.http.post<string>(`${this.BASE_URL}/detallesP`, detalle);
  }
  actualizarDetalle(detalle: DetalleModel) {
    return this.http.put<string>(`${this.BASE_URL}/detallesP/${detalle.idDetalles}`, detalle);
  }
  borrarDetalle(idDetalles: string) {
    return this.http.delete<string>(`${this.BASE_URL}/detallesP/${idDetalles}`);
  }
}
```

Correo, con este servicio se accede a la funcionalidad que permite enviar emails

The screenshot shows the 'Explorador' (Explorer) view on the left with files under 'DONFUVER' and 'src/app/shared'. The 'correo.service.ts' file is open in the editor. It contains code for a 'CorreoService' class with a single method 'enviarCorreo'. It uses HttpClient to interact with a local host at port 3000.

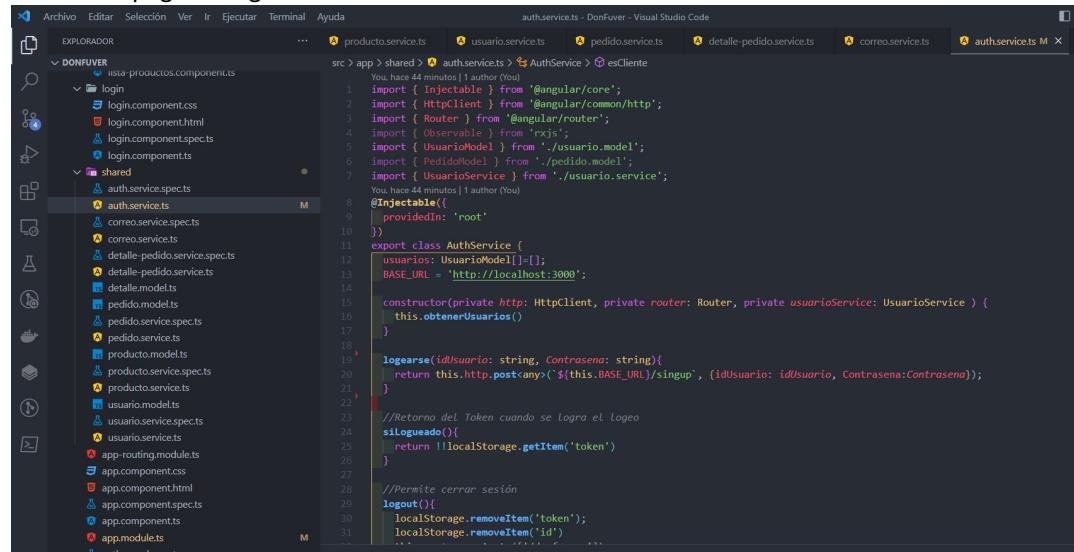
```
You, anteyer | 1 author (You)
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@.Injectable({
  providedIn: 'root'
})
export class CorreoService {
  BASE_URL = 'http://localhost:3000'; // Ruta del endpoint en el backend

  constructor(private http: HttpClient) {}

  enviarCorreo(email: string, subject: string, text: string) {
    console.log('Datos de correo electrónico:', { email, subject, text });
    You, anteyer + confirmacionCorreo
    return this.http.post<any>(`${this.BASE_URL}/correo/${email}`, { subject, text });
  }
}
```

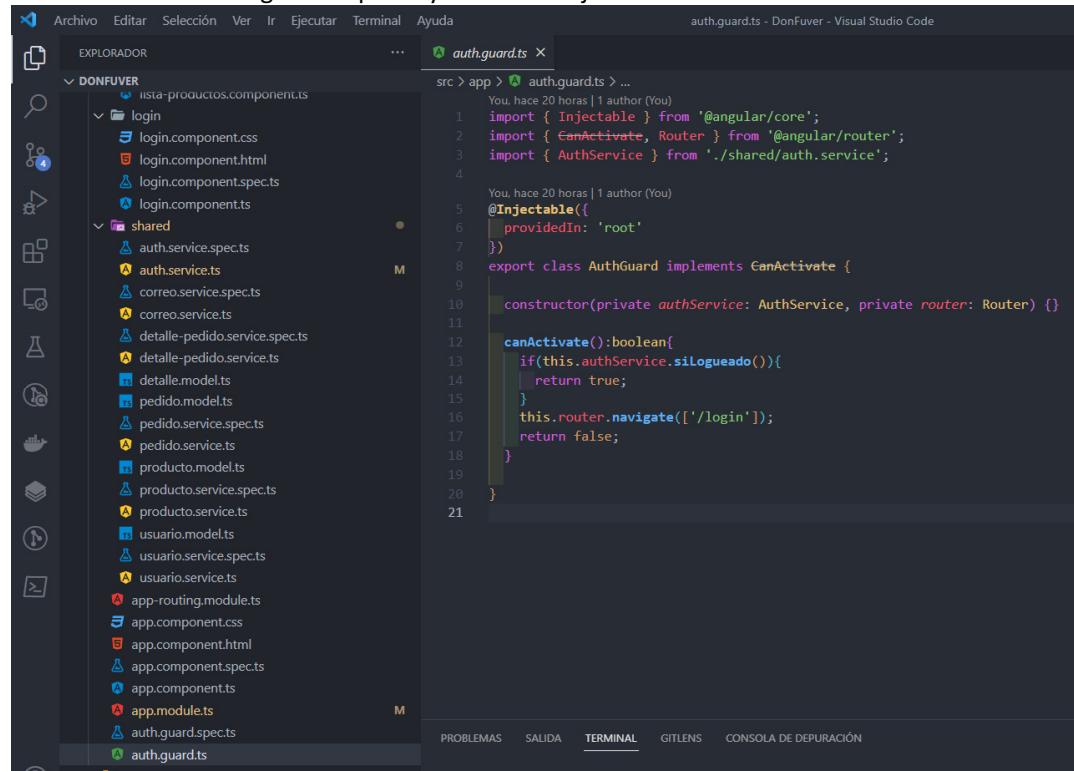
Auth, aquí se implementó toda la lógica para Autenticarse y los permisos para acceder a las diferentes páginas según el rol.



The screenshot shows the Visual Studio Code interface with the file 'auth.service.ts' open in the editor. The code implements an AuthService with methods for logging in, checking if a user is logged in, and logging out. It uses HttpClient, Router, and AuthService from '@angular/core' and '@angular/common/http'. It also imports Observable from rxjs and UsuarioModel, PedidoModel, and UsuarioService from './usuario.model', './pedido.model', and './usuario.service' respectively. The code includes logic to handle user authentication and session storage.

```
You hace 44 minutos | 1 autor (You)
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Router } from '@angular/router';
4 import { Observable } from 'rxjs';
5 import { UsuarioModel } from './usuario.model';
6 import { PedidoModel } from './pedido.model';
7 import { UsuarioService } from './usuario.service';
8
9 @Injectable({
10   providedIn: 'root'
11 })
12 export class AuthService {
13   usuarios: UsuarioModel[] = [];
14   BASE_URL = 'http://localhost:3000';
15
16   constructor(private http: HttpClient, private router: Router, private usuarioService: UsuarioService) {
17     this.obtenerUsuarios();
18   }
19
20   logearse(idUsuario: string, Contrasena: string){
21     return this.http.post<any>(`${this.BASE_URL}/singup`, {idUsuario: idUsuario, Contrasena:Contrasena});
22   }
23
24   //Retorno del Token cuando se Logra el Logeo
25   siLogueado(){
26     return !!localStorage.getItem('token')
27   }
28
29   logout(){
30     localStorage.removeItem('token');
31     localStorage.removeItem('id');
32   }
33 }
```

También se creo auth.guard.ts para ayudar a manejar la ruta de autenticación:



The screenshot shows the Visual Studio Code interface with the file 'auth.guard.ts' open in the editor. The code defines a AuthGuard class that implements CanActivate. It checks if the user is logged in using the authService. If they are not, it navigates to the login page. Otherwise, it returns true. The code imports Injectable, CanActivate, Router, and AuthService from '@angular/core', '@angular/router', and './shared/auth.service' respectively.

```
You hace 20 horas | 1 autor (You)
1 import { Injectable } from '@angular/core';
2 import { CanActivate, Router } from '@angular/router';
3 import { AuthService } from './shared/auth.service';
4
5 You hace 20 horas | 1 autor (You)
6 @Injectable({
7   providedIn: 'root'
8 })
9 export class AuthGuard implements CanActivate {
10   constructor(private authService: AuthService, private router: Router) {}
11
12   canActivate():boolean{
13     if(this.authService.siLogueado()){
14       return true;
15     }
16     this.router.navigate(['/login']);
17     return false;
18   }
19
20 }
```

5. En se configura app.module.ts, importando HttpClientModule y FormsModule, y en providers agregamos: los diferentes servicios

```

src > app > app.module.ts > AppModule
16   import { CarroComponent } from './carro/carro.component';
17   import { AuthGuard } from './auth.guard';
18   import { CorreoService } from './shared/correo.service';
19
20 You, hace 8 segundos | 1 author (You)
21 @NgModule({
22   declarations: [
23     AppComponent,
24     ListaProductosComponent,
25     EditarProductosComponent,
26     ListaPedidosComponent,
27     LoginComponent,
28     DonFruverComponent,
29     CarroComponent
30   ],
31   imports: [
32     BrowserModule,
33     AppRoutingModule,
34     HttpClientModule,
35     FormsModule
36   ],
37   providers: [
38     DetallePedidoService,
39     PedidoService,
40     UsuarioService,
41     ProductoService,
42     CorreoService,
43     AuthGuard
44   ],
45   bootstrap: [AppComponent]
46 }
47 export class AppModule { }

```

6. Con la instrucción ng generate component <nOMBRE-componente>, creamos los diferentes componentes.

```

src > app > app-routing.module.ts > app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [];
5
6 @NgModule({
7   imports: [RouterModule.forRoot(routes)],
8   exports: [RouterModule]
9 })
10 export class AppRoutingModule { }

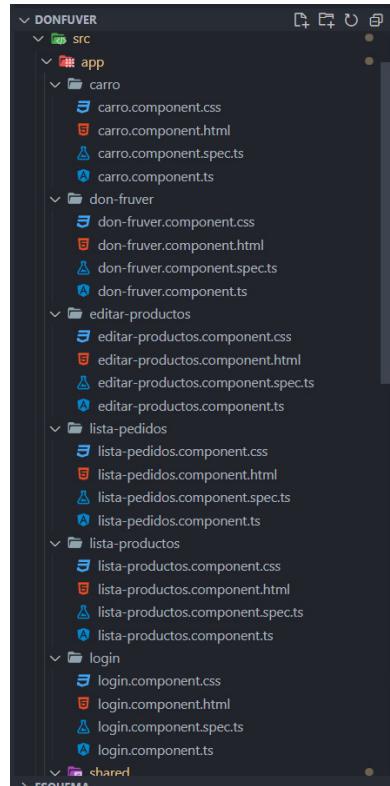
CREATE src/app/shared/detalle-pedido.service.spec.ts (933 bytes)
CREATE src/app/shared/detalle-pedido.service.ts (142 bytes)
PS E:\>_Programas\DESKTOP\Taller3_DonFruver\DonFruver> ng generate component lista-productos
CREATE src/app/lista-productos/lista-productos.component.ts (616 bytes)
CREATE src/app/lista-productos/lista-productos.component.html (237 bytes)
CREATE src/app/lista-productos/lista-productos.component.css (0 bytes)
UPDATE src/app/app.module.ts (988 bytes)
PS E:\>_Programas\DESKTOP\Taller3_DonFruver\DonFruver> ng generate component editar-productos
CREATE src/app/editar-productos/editar-productos.component.ts (211 bytes)
CREATE src/app/editar-productos/editar-productos.component.html (623 bytes)
CREATE src/app/editar-productos/editar-productos.component.spec.ts (241 bytes)
CREATE src/app/editar-productos/editar-productos.component.css (0 bytes)
UPDATE src/app/app.module.ts (1161 bytes)
PS E:\>_Programas\DESKTOP\Taller3_DonFruver\DonFruver>

```

la totalidad de componentes creados fueron:

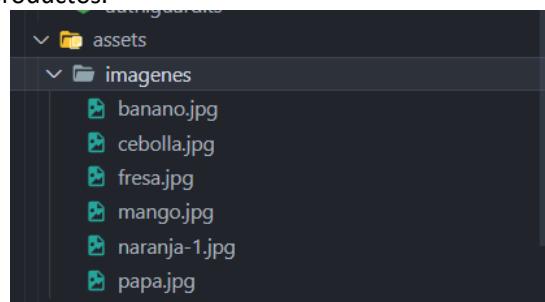
- Carro
- Don-fruver
- Editar-productos
- Lista-pedidos
- Lista-productos
- Login

Como se observa a continuación:



En nombre.component.html se diseño las vistas con las que interactúa el usuario, en nombre.component.ts se planteó la lógica para hacer funcionar las vistas y la interacción con el backend, y en nombre.component.css se puso los estilos para hacer amigable la aplicación web.

En el directorio assets se creó el directorio imágenes para almacenar ahí las imágenes de los diferentes productos.



La estructura completa del proyecto es la siguiente:



A continuación, se presentará parte del código cada archivo en el que se ha trabajado:

index.html: se incluyeron los CDN para trabajar con Bootstrap:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>DonFuber</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-aepP/Y94NfpVNVgjZdgIC5+VXNBRQNGCheKRQN+PtmoIDExupvnDjzQiu9" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
</head>
<body>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-HwwtgBNc3BZJLYd8oVXrBZt8cqSpE8NSn/C8IVinxGoxmnlMuBnhbgkrk" crossorigin="anonymous"></script>
<app-root></app-root>
</body>
</html>
```

app.component.html: se agregó un navbar para que este presente en todas las vistas

```

<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <a class="navbar-brand" href="/donfruver">
      
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarScroll" aria-controls="navbarScroll">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarScroll">
      <ul class="navbar-nav me-auto my-2 my-lg-0 navbar-nav-scroll" style="--bs-scroll-height: 100px;">
        <li class="nav-item me-4">
          <a class="nav-link" aria-current="page" href="/donfruver" routerLinkActive="active">
            <i class="fas fa-home"></i> Inicio
          </a>
        </li>
      </ul>
      <ul class="nav navbar-nav ml-auto">
        <li class="nav-item" ngIf="authService.esCliente()&& authService.silogueado()">
          <a class="nav-link" href="/carro" routerLinkActive="active">
            <i class="fas fa-shopping-cart"></i> Carrera
          </a>
        </li>
        <li class="nav-item dropdown" ngIf="authService.esAdministrador()&& authService.silogueado()">
          <a class="nav-link" href="/pedidos" routerLinkActive="active">
            <i class="fas fa-dashboard-list"></i> Pedidos
          </a>
        </li>
        <li class="nav-item dropdown" ngIf="authService.esAdministrador()&& authService.silogueado()">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            <i class="fas fa-box-open"></i> Productos
          </a>
          <ul class="dropdown-menu">

```

app-routing.module.ts: se agregaron todas la rutas para navegar con las que se trabaja

```

const routes: Routes = [
  { path: 'productos', component: ListaProductosComponent, canActivate:[AuthGuard] },
  { path: 'productos/editar/:idProducto', component: EditarProductosComponent, canActivate:[AuthGuard] },
  { path: 'productos/agregar', component: EditarProductosComponent, canActivate:[AuthGuard] },
  { path: 'pedidos', component: ListaPedidosComponent, canActivate:[AuthGuard] },
  { path: 'carro', component: CarreroComponent, canActivate:[AuthGuard] },
  { path: 'donfruver', component: DonFruverComponent },
  { path: 'login', component: LoginComponent },
  { path:'**',redirectTo:'donfruver',pathMatch:'full' }
];

```

```

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

login.component.ts: captura los datos para permitir la autenticación del usuario

```

@Directive({
  selector: '#app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  user: new UsuarioModel(' ', ' ', ' ', ' ', ' ', ' ');
  id: string | undefined;

  constructor(private authService: AuthService, private router: Router) {}

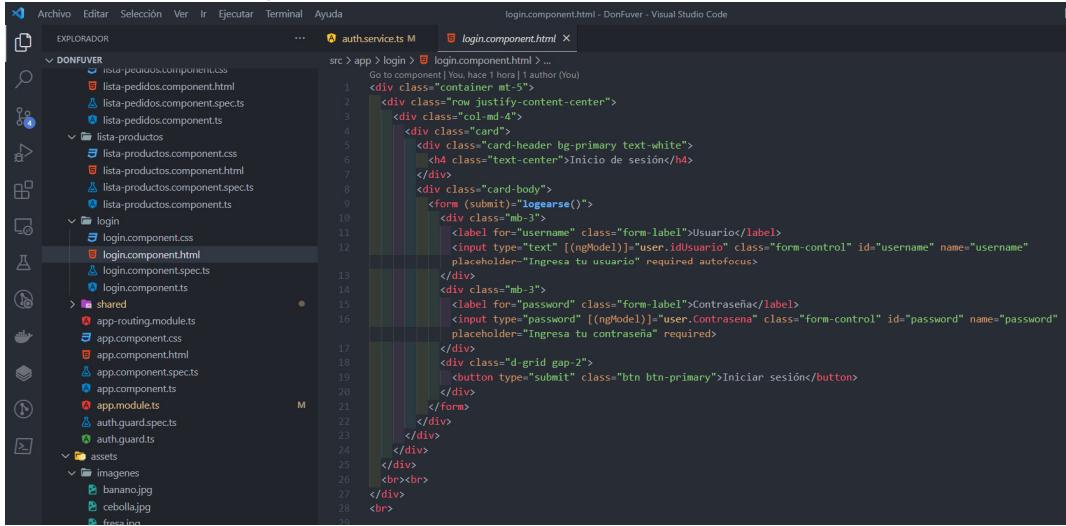
  ngOnInit(): void {
  }

  //Permite Logearse en la tienda, ademas se genera un token y se guarda el idUsuario en localStorage
  logearse() {
    console.log(this.user);
    this.authService.logearse(this.user.idUsuario, this.user.Contrasena).subscribe(
      res => {
        console.log(res);
        localStorage.setItem('token', res.token);
        localStorage.setItem('id', this.user.idUsuario);
        this.router.navigate(['/']);
      },
      err => console.log(err)
    );
  }
}

```

login.component.html:

- Vista accesible por todos los usuarios

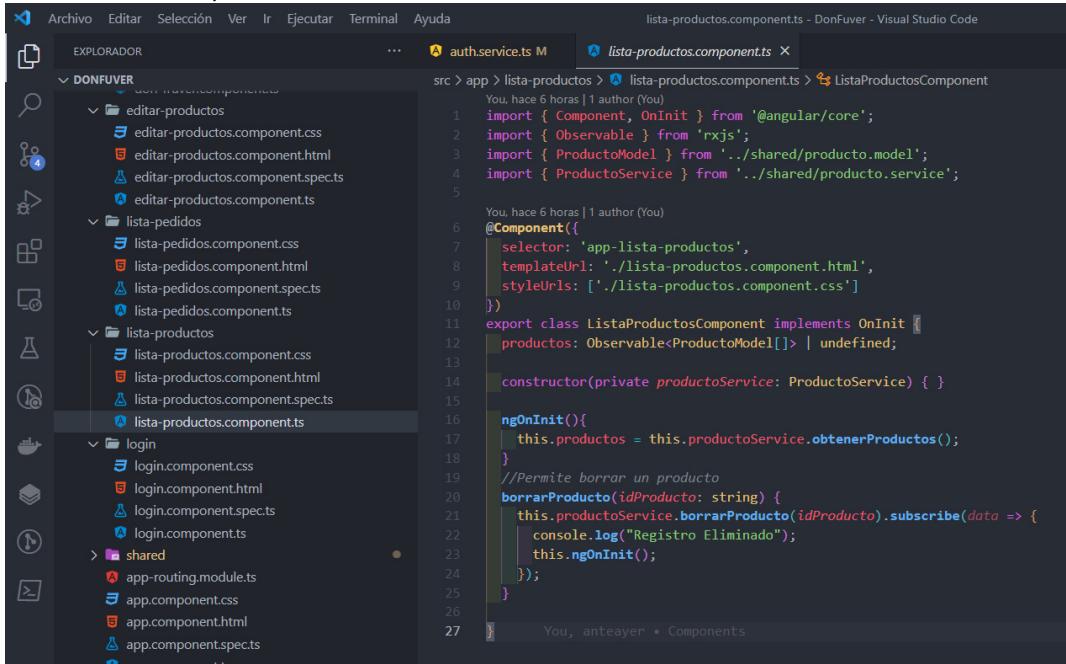


The screenshot shows the Visual Studio Code interface with the file 'login.component.html' open. The code is an Angular template for a login form. It includes an H4 header 'Inicio de sesión', a form with two input fields ('username' and 'password'), and a submit button labeled 'Iniciar sesión'. The code uses Bootstrap classes like 'container', 'row justify-content-center', 'card', and 'card-body'.

```
src > app > login > login.component.html ...  
1 <div class="container mt-5">  
2   <div class="row justify-content-center">  
3     <div class="col-md-4">  
4       <div class="card">  
5         <div class="card-header bg-primary text-white">  
6           <h4 class="text-center">Inicio de sesión</h4>  
7         </div>  
8         <div class="card-body">  
9           <form (submit)="loguearse()">  
10             <div class="mb-3">  
11               <label for="username" class="form-label">Usuario</label>  
12               <input type="text" [(ngModel)]="user.idUsuario" class="form-control" id="username" name="username" placeholder="Ingresá tu usuario" required autofocus>  
13             </div>  
14             <div class="mb-3">  
15               <label for="password" class="form-label">Contraseña</label>  
16               <input type="password" [(ngModel)]="user.Contraseña" class="form-control" id="password" name="password" placeholder="Ingresá tu contraseña" required>  
17             </div>  
18             <div class="d-grid gap-2">  
19               <button type="submit" class="btn btn-primary">Iniciar sesión</button>  
20             </div>  
21           </form>  
22         </div>  
23       </div>  
24     </div>  
25   </div>  
26   <br><br>  
27 </div>  
28 <br>
```

lista-productos.component.ts: permite

- Obtener todos los productos
- Borrar los productos



The screenshot shows the Visual Studio Code interface with the file 'lista-productos.component.ts' open. The code defines a component named 'ListaProductosComponent' that implements the 'OnInit' interface. It imports services from 'angular/core', 'rxjs', 'shared/producto.model', and 'shared/producto.service'. The component has a selector 'app-lista-productos', a template URL 'listaproductos.component.html', and a style URL 'listaproductos.component.css'. The 'ngOnInit' method initializes the 'productos' observable and handles the 'borrarProducto' event by logging a message and calling 'ngOnInit' again. The code also includes a comment about permitting product deletion.

```
src > app > lista-productos > lista-productos.component.ts ...  
1 import { Component, OnInit } from '@angular/core';  
2 import { Observable } from 'rxjs';  
3 import { ProductoModel } from './shared/producto.model';  
4 import { ProductoService } from './shared/producto.service';  
5  
5 You, hace 6 horas | 1 autor (You)  
6 @Component({  
7   selector: 'app-lista-productos',  
8   templateUrl: './listaproductos.component.html',  
9   styleUrls: ['./listaproductos.component.css']  
10 })  
11 export class ListaProductosComponent implements OnInit {  
12   productos: Observable<ProductoModel[]> | undefined;  
13  
14   constructor(private productoService: ProductoService) { }  
15  
16   ngOnInit():  
17     this.productos = this.productoService.obtenerProductos();  
18   }  
19   //Permite borrar un producto  
20   borrarProducto(idProducto: string) {  
21     this.productoService.borrarProducto(idProducto).subscribe(data => {  
22       console.log("Registro Eliminado");  
23       this.ngOnInit();  
24     });  
25   }  
26 }  
27 You, anteayer • Components
```

lista-productos.component.html:

- Muestra la lista de productos
- Redirecciona a agregar producto
- Redirecciona a editar producto
- Solo es accesible por el Administrador

```


<div class="row">
    <div class="col">
      <div class="d-flex justify-content-between align-items-center mb-3">
        <h2>Lista de Productos</h2>
        <a class="btn btn-success" [routerLink]="/productos/agregar">Nuevo Producto</a>
      </div>
      <table class="table table-striped">
        <thead>
          <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Descripción</th>
            <th>Categoría</th>
            <th>Precio</th>
            <th>Cantidad Disponible</th>
            <th>Imagen</th>
            <th>Acciones</th>
          </tr>
        </thead>
        <tbody>
          <tr> <!-- Celda vacía para separar los botones -->
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
            <td></td>
          </tr>
          <tr>
            <td>1</td>
            <td>Laptop 15</td>
            <td>Unidad de procesamiento portátil</td>
            <td>Electrónica</td>
            <td>$1200</td>
            <td>10</td>
            <td></td>
            <td><a href="#" class="btn btn-info mr-2" [routerLink]="/productos/editar/">Editar</a></td>
          </tr>
          <tr>
            <td>2</td>
            <td>Monitor 24</td>
            <td>Pantalla para computadora</td>
            <td>Electrónica</td>
            <td>$800</td>
            <td>5</td>
            <td></td>
            <td><a href="#" class="btn btn-info mr-2" [routerLink]="/productos/editar/">Editar</a></td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>


```

lista-pedidos.component.ts: permite

- Confirmar un pedido
- Rechazar un pedido
- Enviar correo informando la aceptación o rechazo del pedido
- Obtener la lista de pedidos

```

@Component({
  selector: 'app-lista-pedidos',
  templateUrl: './lista-pedidos.component.html',
  styleUrls: ['./lista-pedidos.component.css']
})
export class ListaPedidosComponent {
  pedidos: Observable<PedidoModel[]> | undefined;
  correoElectronico: string | undefined;
  // correoService: any;

  constructor(
    private pedidosService: PedidoService,
    private correoService: CorreoService
  ) {}

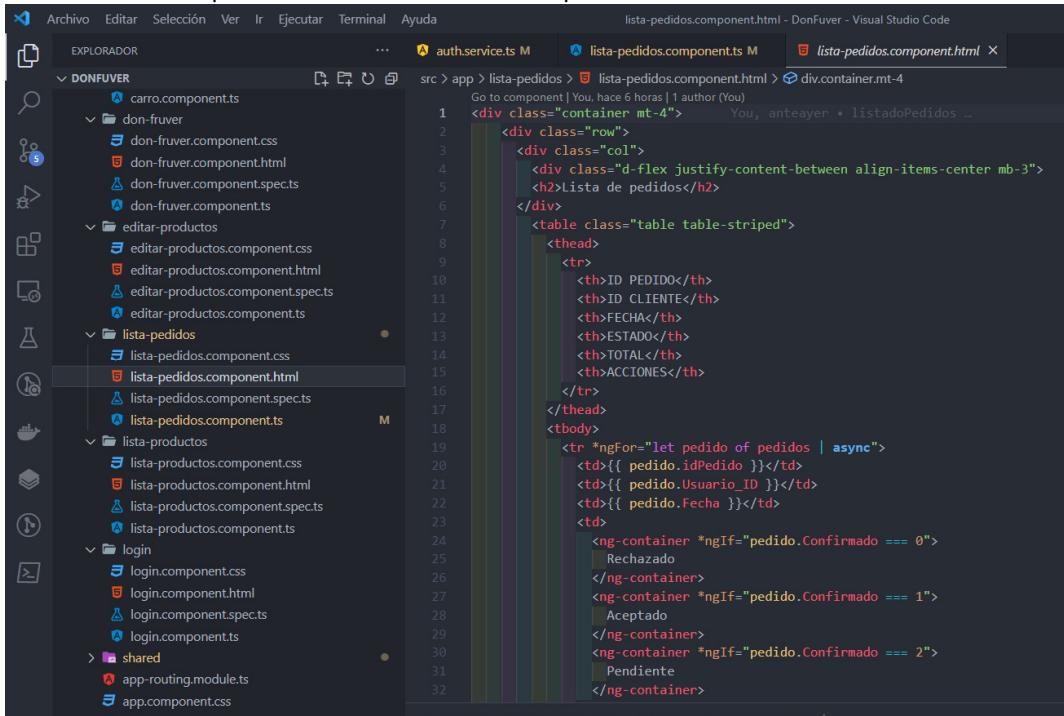
  ngOnInit() {
    this.pedidos = this.pedidosService.obtenerPedidos();
  }

  confirmarPedido(pedido: PedidoModel): void {
    pedido.Confirmado = 1; // Establecer el estado a "Aceptado" (valor 1)
    this.obtenerCorreoElectronico(pedido.Usuario_ID);
  }
}

```

lista-pedidos.component.html:

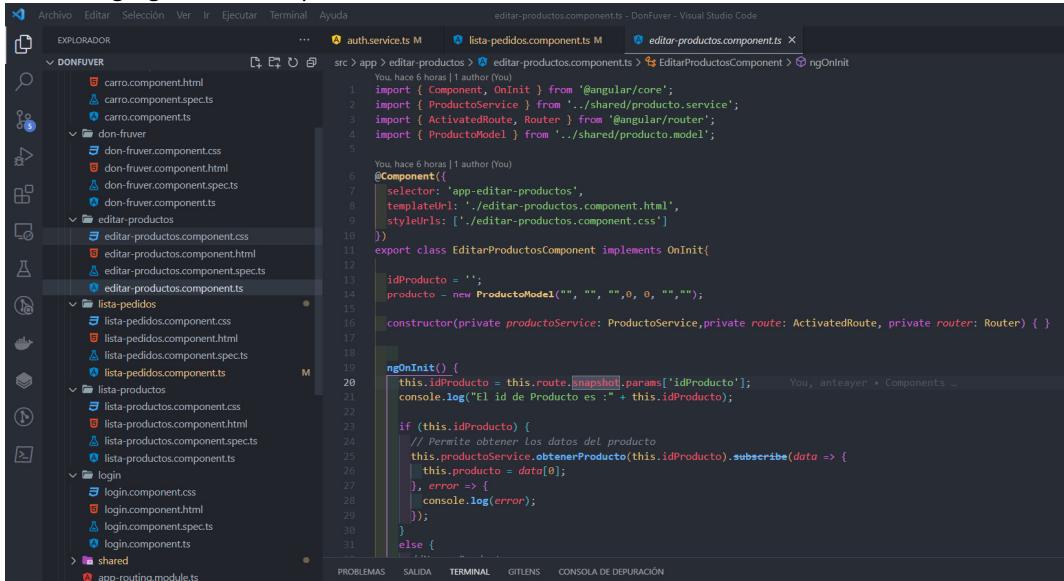
- Accesible solo por el Administrador
- Permite observar la lista de pedidos
- El usuario puede confirmar o rechazar un pedido



```
<div class="container mt-4">
  <div class="row">
    <div class="d-flex justify-content-between align-items-center mb-3">
      <h2>Lista de Pedidos</h2>
    </div>
    <table class="table table-striped">
      <thead>
        <tr>
          <th>ID Pedido</th>
          <th>ID Cliente</th>
          <th>Fecha</th>
          <th>Estado</th>
          <th>Total</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let pedido of pedidos | async">
          <td>{{ pedido.idPedido }}</td>
          <td>{{ pedido.Usuario_ID }}</td>
          <td>{{ pedido.Fecha }}</td>
          <td>
            <ng-container *ngIf="pedido.Confirmado === 0">
              Rechazado
            </ng-container>
            <ng-container *ngIf="pedido.Confirmado === 1">
              Aceptado
            </ng-container>
            <ng-container *ngIf="pedido.Confirmado === 2">
              Pendiente
            </ng-container>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

editar-productos.component.ts: permite

- Editar la información de un producto y actualizarla en la BD
- Agregar un nuevo producto



```
import { Component, OnInit } from '@angular/core';
import { ProductoService } from '../shared/producto.service';
import { ActivatedRoute, Router } from '@angular/router';
import { ProductoModel } from '../shared/producto.model';

@Component({
  selector: 'app-editar-productos',
  templateUrl: './editar-productos.component.html',
  styleUrls: ['./editar-productos.component.css']
})
export class EditarProductosComponent implements OnInit {

  idProducto = '';
  producto = new ProductoModel("", "", "", 0, 0, "", "");

  constructor(private productoService: ProductoService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {
    this.idProducto = this.route.snapshot.params['idProducto'];
    console.log("El id de Producto es :" + this.idProducto);

    if (this.idProducto) {
      // Permite obtener los datos del producto
      this.productoService.obtenerProducto(this.idProducto).subscribe(data => {
        this.producto = data[0];
      }, error => {
        console.log(error);
      });
    } else {
      // Permite crear un nuevo producto
      this.producto = new ProductoModel("", "", "", 0, 0, "", "");
    }
  }

  guardarProducto() {
    this.productoService.guardarProducto(this.producto).subscribe(data => {
      console.log("Producto guardado exitosamente");
      this.router.navigate(['/listar-productos']);
    }, error => {
      console.log(error);
    });
  }
}
```

editar-productos.component.html:

- Vista para ingresar información de un producto nuevo o uno antiguo
- Accesible solo por el administrador

```


<div class="row">
    <div class="col-md-6 offset-md-3">
      <form (ngSubmit)="onSubmit()" #productoForm="ngForm">
        <div class="mb-3">
          <label class="form-label" for="nombre">Nombre</label>
          <input type="text" class="form-control" required [(ngModel)]="producto.Nombre" name="nombre">
        </div>
        <div class="mb-3">
          <label class="form-label" for="descripcion">Descripción</label>
          <textarea class="form-control" required [(ngModel)]="producto.Descripcion" name="descripcion" rows="5"></textarea>
        </div>
        <div class="mb-3">
          <label class="form-label" for="categoria">Categoría</label>
          <select class="form-select" required [(ngModel)]="producto.Categoria" name="categoria">
            <option value="Frutas">Frutas</option>
            <option value="Vegetales">Vegetales</option>
          </select>
        </div>
        <div class="mb-3">
          <label class="form-label" for="precio">Precio</label>
          <input type="text" class="form-control" required [(ngModel)]="producto.Precio" name="precio" pattern="^\d+(\.\d{1,2})?$/>
        </div>
        <div class="mb-3">
          <label class="form-label" for="cantidad">Cantidad Disponible</label>
          <input type="text" class="form-control" required [(ngModel)]="producto.Cantidad_Disponible" name="cantidad" pattern="^\d+$"/>
        </div>
        <div class="mb-3">
          <label class="form-label" for="imagen">Imagen</label>
          <input type="text" class="form-control" required [(ngModel)]="producto.Imagen" name="imagen">
        </div>
      </form>
    </div>
  </div>


```

don-fruver.component.ts: permite

- Obtener todos los productos
- Hacer búsqueda de productos
- Filtrar por categoría
- Filtrar por precio
- Llamar a una ventana emergente
- Comprar un producto
- Obtener información de un producto

```

You have 1 hour | 1 author (You)
import { Component } from '@angular/core';
import { ProductoModel } from '../shared/producto.model';
import { Observable } from 'rxjs';
import { ProductoService } from '../shared/producto.service';
import { PedidoModel } from '../shared/pedido.model';
import { DetalleModel } from '../shared/detalle.model';
import { PedidoService } from '../shared/pedido.service';
import { DetallePedidoService } from '../shared/detalle-pedido.service';
import { map } from 'rxjs/operators';

You have 1 hour | 1 author (You)
@Component({
  selector: 'app-don-fruver',
  templateUrl: './don-fruver.component.html',
  styleUrls: ['./don-fruver.component.css']
})
export class DonFruverComponent {
  productos: Observable<ProductoModel[]> | undefined;
  showModal = false; // Variable para mostrar/ocultar el modal
  productoSeleccionado: ProductoModel | undefined;
  pedidoNuevo: PedidoModel | undefined;
  productosFiltrados: ProductoModel[] = [] // Agregar lista para almacenar productos filtrados
  filtro: any = {} // Objeto para almacenar los criterios de filtrado

  // detallesPedido: DetalleModel[] = [];
  // cantidadProductos: number = 0;

  constructor(
    private productoService: ProductoService,
    private pedidoService: PedidoService,
    private detallePedidoService: DetallePedidoService
  ) {}
}

```

don-fruver.component.html:

- Muestra todos los productos en tarjetas
- Muestra la barra de búsqueda
- Muestra los filtros
- Muestra la ventana emergente con la información de los productos
- Permite la interacción para hacer compras y búsquedas
- Accesible por todos los usuarios

```

src > app > don-fruver > don-fruver.component.html


<form class="form-inline d-flex justify-content-end ml-auto" (ngSubmit)="filtrarProductos()" #filtroForm="ngForm">
    <div class="input-group">
      <input type="text" class="form-control" placeholder="Buscar producto" name="buscar" [(ngModel)]="filtro.nombre">
      <button type="submit" class="btn btn-primary">Buscar</button>
    </div>
  </form>
  <br>
  <form class="form-inline d-flex" (ngSubmit)="filtrarProductos()" #filtroForm="ngForm">
    <div class="input-group">
      <label for="categoria" class="mr-2">Categoria:</label>
      <input type="text" class="form-control" name="categoria" [(ngModel)]="filtro.categoria">
    </div>
    <div class="input-group">
      <label for="precio" class="mr-2">Precio Máximo:</label>
      <input type="number" class="form-control mr-2" name="precio" [(ngModel)]="filtro.precio" min="0">
    </div>
    &nbsp;&nbsp;
    <div class="input-group">
      <button type="submit" class="btn btn-primary mr-2">Filtrar</button>
      <button type="button" class="btn btn-secondary mr-2" (click)="limpiarFiltro()>">Limpiar</button>
    </div>
  </form>
  <div class="row">
    <div class="col-md-12 mb-4">
      <div>
        <!-- lista de productos -->
        <div class="col-md-4" *ngFor="let producto of productosFiltrados">
          <img [src]="'assets/imagenes/' + producto.Imagen" class="card-img-top custom-img" alt="Imagen del producto">
          <div class="card-body d-flex flex-column">
            ...
          </div>
        </div>
      </div>
    </div>
  </div>


```

carro.component.ts: permite

- Obtener los pedidos de cada cliente
- Editar el pedido por cada producto
- Borrar productos del pedido

```

src > app > carro > carro.component.ts
CarroComponent > ngOnInit
You, hace 6 horas | 1 author (You)
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';
import { DetalleModel } from './shared/detalle.model';
import { DetallePedidoService } from './shared/detalle-pedido.service';
import { ProductoModel } from './shared/producto.model';
import { PedidoService } from './shared/pedido.service';
import { PedidoModel } from './shared/pedido.model';
import { UsuarioService } from './shared/usuario.service';
import { UsuarioModel } from './shared/usuario.model';

You, hace 6 horas | 1 author (You)
@Component({
  selector: 'app-carro',
  templateUrl: './carro.component.html',
  styleUrls: ['./carro.component.css']
})
export class CarroComponent implements OnInit {
  detalles: Observable<DetalleModel[]> | undefined;
  productos: ProductoModel[] = [];
  pedidos: PedidoModel[] = [];
  usuarios: UsuarioModel[] = [];
  pedidoConfirmado: Map<string, boolean> = new Map<string, boolean>();

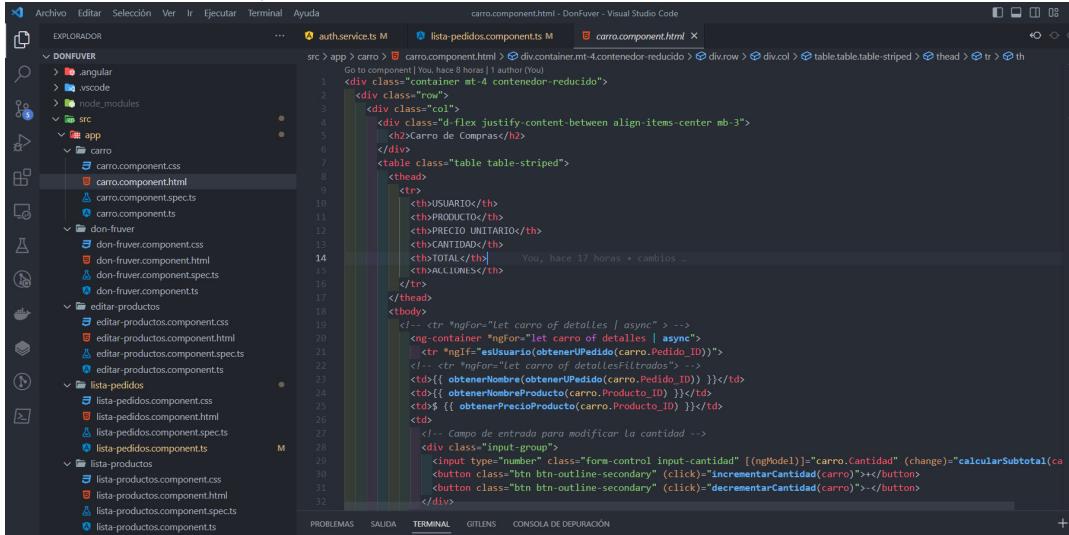
  constructor(private detalleService: DetallePedidoService,
    private pedidoService: PedidoService,
    private usuarioService: UsuarioService) { }

  ngOnInit() {
    You, anteyer * avance ...
    this.detalles = this.detalleService.obtenerDetalles();
    this.obtenerProductos();
    // Obtener los pedidos del usuario
    this.pedidoService.obtenerPedidos().subscribe((pedidos) => {
      ...
    });
  }
}

```

carro.component.html:

- Muestra la lista de los pedidos con su información
- Muestra opciones para cambiar numero de productos y eliminar productos del pedido
- Accesible solo por el usuario Cliente



```
src > app > carro > carro.component.html M lista-pedidos.component.ts carro.component.html
Go to component! You have 8 horas | author (You)
1 <div class="container mt-4 contenedor-reducido">
2   <div class="row">
3     <div class="col">
4       <div class="d-flex justify-content-between align-items-center mb-3">
5         <h2>Carro de Compras</h2>
6       </div>
7       <table class="table table-striped">
8         <thead>
9           <tr>
10             <th>USUARIO</th>
11             <th>PRODUCTO</th>
12             <th>PRECIO UNITARIO</th>
13             <th>CANTIDAD</th>
14             <th>TOTAL</th> You, hace 17 horas * cambios ...
15             <th>ACCIONES</th>
16           </tr>
17         </thead>
18         <tbody>
19           <!-- <tr *ngFor="let carro of detalles | async" -->
20           <ng-container *ngFor="let carro of detalles | async">
21             <tr *ngIf="!esusuario(obtenerUPedido(carro.Pedido_ID))">
22               <td>{{ obtenerNombre(obtenerUPedido(carro.Pedido_ID)) }}</td>
23               <td>{{ obtenerNombreProducto(carro.Producto_ID) }}</td>
24               <td>{{ obtenerPrecioProducto(carro.Producto_ID) }}</td>
25               <td>
26                 <!-- Campo de entrada para modificar la cantidad -->
27                 <div class="input-group">
28                   <input type="number" class="form-control input-cantidad" [(ngModel)]="carro.Cantidad" (change)="calcularSubtotal(carro)" ...
29                   <button class="btn btn-outline-secondary" (click)="incrementarCantidad(carro)"></button>
30                   <button class="btn btn-outline-secondary" (click)="decrementarCantidad(carro)"></button>
31                 </div>
32               </td>
33             </tr>
34           </ng-container>
35         </tbody>
36       </table>
37     </div>
38   </div>
39 </div>
```

Notas:

- Para el envío del correo se trabajo con nodemailer en el backend
- Para la autenticación se trabajo con jsonwebtoken en el backend

Vistas aplicación web Don Fruver funcionando:

1. Vista inicial

The screenshot shows the main landing page of the Don Fruver web application. At the top, there is a search bar labeled "Buscar producto" and a "Buscar" button. Below it, there are two filter sections: "Categoría:" and "Precio Máximo:". There are three product cards displayed:

- Naranja** (Frutas) - Precio: \$ 20.99
Stock: 15
Buy button: Comprar
Details button: Ver detalles
- Cebolla** (Vegetales) - Precio: \$ 35.50
Stock: 16
Buy button: Comprar
Details button: Ver detalles
- Papa** (Vegetales) - Precio: \$ 45.00
Stock: 20
Buy button: Comprar
Details button: Ver detalles

2. Vista autenticación

The screenshot shows the login page of the Don Fruver web application. The title of the form is "Inicio de sesión". It contains two input fields: "Usuario" and "Contraseña", and a "Iniciar sesión" button.

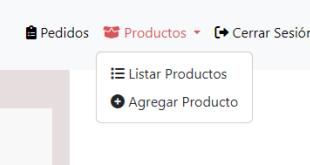
3. Vista autenticación como Administrador, se muestran las opciones pedidos y productos

The screenshot shows a web browser window for 'Don Fruter' at 'localhost:4200/donfruter'. The top navigation bar includes links for 'Pedidos', 'Productos' (with a dropdown arrow), and 'Cerrar Sesión'. A search bar at the top right contains the placeholder 'Buscar producto' and a 'Buscar' button. Below the search bar are filters for 'Categoría:' and 'Precio Máximo:', with 'Filtrar' and 'Limpiar' buttons. The main content area displays three product cards:

- Naranja** (Frutas) - Precio: \$ 20.99
Stock: 15
Buttons: Comprar, Ver detalles
- Cebolla** (Vegetales) - Precio: \$ 35.50
Stock: 16
Buttons: Comprar, Ver detalles
- Papa** (Vegetales) - Precio: \$ 45.00
Stock: 20
Buttons: Comprar, Ver detalles

The bottom of the screen shows a Windows taskbar with various icons and the system clock indicating '7:35 p.m.' on '1/09/2023'.

4. Menú desplegable Productos



5. Vista de pedidos

The screenshot shows a web browser window for 'Don Fruter' at 'localhost:4200/pedidos'. The top navigation bar includes links for 'Pedidos', 'Productos' (with a dropdown arrow), and 'Cerrar Sesión'. A title 'Lista de pedidos' is displayed above a table. The table has columns: ID PEDIDO, ID CLIENTE, FECHA, ESTADO, TOTAL, and ACCIONES. The data in the table is as follows:

ID PEDIDO	ID CLIENTE	FECHA	ESTADO	TOTAL	ACCIONES
1	123	2023-07-25T15:30:00.000Z	Aceptado	\$ 40.99	<button>Confirmar</button> <button>Rechazar</button>
2	456	2023-07-26T09:45:00.000Z	Pendiente	\$ 80.50	<button>Confirmar</button> <button>Rechazar</button>
11	456	2023-08-01T02:46:58.000Z	Pendiente	\$ 106.50	<button>Confirmar</button> <button>Rechazar</button>
30	456	2023-08-01T16:28:07.000Z	Pendiente	\$ 100.00	<button>Confirmar</button> <button>Rechazar</button>
31	456	2023-08-01T20:18:15.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>
32	456	2023-08-01T23:30:25.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>

The bottom of the screen shows a Windows taskbar with various icons and the system clock indicating '7:35 p.m.' on '1/09/2023'.

6. Vista de listado de productos

The screenshot shows a web browser window titled "Don Fruter" displaying a list of products. The table has the following data:

ID	NOMBRE	DESCRIPCIÓN	CATEGORÍA	PRECIO	CANTIDAD	IMAGEN	ACCIONES
1	Naranja	La naranja es una fruta cítrica de forma redonda o elipsoidal, conocida por su sabor dulce y refrescante. Es rica en vitamina C, lo que la convierte en una excelente opción para fortalecer el sistema inmunológico y mantener una piel saludable.	Frutas	\$ 20.99	15		<button>Editar</button> <button>Borrar</button>
2	Cebolla	La cebolla es una hortaliza bulbosa ampliamente utilizada en la cocina de diferentes culturas. Tiene un sabor característico y puede variar en tamaño y color. Es una fuente de antioxidantes y nutrientes beneficiosos para la salud.	Vegetales	\$ 35.50	16		<button>Editar</button> <button>Borrar</button>
3	Papa	La papa, también conocida como patata, es un tubérculo originario de América del Sur. Es un alimento básico en muchas dietas y se puede preparar de diversas formas, como hervida, frita, asada o al horno. Es rica en carbohidratos, vitaminas y minerales.	Vegetales	\$ 45.00	20		<button>Editar</button> <button>Borrar</button>
4	Mango	El mango es una fruta tropical de pulpa jugosa y dulce, con un sabor distintivo y refrescante. Es una excelente fuente de vitamina C, vitamina A y fibra, y se utiliza en diversas preparaciones culinarias, como ensaladas, saladas y postres.	Frutas	\$ 50.00	16		<button>Editar</button> <button>Borrar</button>
5	Fresa	La fresa es una fruta pequeña y roja con un sabor agrio y refrescante. Es conocida por su alto contenido de vitamina C, antioxidantes y fibra. Se consume fresca, en jugos, mermeladas o como complemento en postres y ensaladas.	Frutas	\$ 29.99	20		<button>Editar</button> <button>Borrar</button>
18	Banano	El banano, también conocido como plátano, es una fruta de pulpa suave y dulce que se presenta en racimos. Es una excelente fuente de potasio y carbohidratos, proporcionando energía rápida y ayudando a mantener el equilibrio hídrico del cuerpo. Es una de las frutas más populares y versátiles en la dieta diaria.	Frutas	\$ 1.00	20		<button>Editar</button> <button>Borrar</button>

7. Vista de agregar nuevo producto

The screenshot shows a web browser window titled "Don Fruter" displaying a form to add a new product. The form fields are:

- Nombre:
- Descripción:
- Categoría:
- Precio: 0
- Cantidad Disponible: 0
- Imagen:

At the bottom right of the form are two buttons: "Regresar" (Return) and "Enviar" (Send).

8. Vista de editar datos de un producto existente

A screenshot of a web browser window showing a product editing form. The URL is `localhost:4200/productos/editar/1`. The form fields include:

- Nombre:** Naranja
- Descripción:** La naranja es una fruta cítrica de forma redonda o elipsoidal, conocida por su sabor dulce y refrescante. Es rica en vitamina C, lo que la convierte en una excelente opción para fortalecer el sistema inmunológico y mantener una piel saludable.
- Categoría:** Frutas
- Precio:** 20.99
- Cantidad Disponible:** 15
- Imagen:** naranja-1.jpg

At the bottom are "Regresar" and "Enviar" buttons.

9. Vista como Cliente

A screenshot of a web browser window showing a client view of the product catalog. The URL is `localhost:4200/donfruver`. The interface includes a search bar and filters for category and price. The products listed are:

Imagen	Nombre	Categoría	Precio	Opciones
	Naranja	Frutas	Precio: \$ 20.99	Comprar Ver detalles
	Cebolla	Vegetales	Precio: \$ 35.50	Comprar Ver detalles
	Papa	Vegetales	Precio: \$ 45.00	Comprar Ver detalles

At the top right, there is a "Carro" (Cart) icon and a "Cerrar Sesión" (Logout) link. The status bar at the bottom right shows "7:37 p. m. 1/09/2023".

10. Vista carrito

The screenshot shows a shopping cart interface titled "Carro de Compras". The table lists items with columns: USUARIO, PRODUCTO, PRECIO UNITARIO, CANTIDAD, TOTAL, and ACCIONES. The data is as follows:

USUARIO	PRODUCTO	PRECIO UNITARIO	CANTIDAD	TOTAL	ACCIONES
Maria López	Cebolla	\$ 35.50	1	\$ 35.50	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Mango	\$ 50.00	2	\$ 100.00	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Fresa	\$ 29.99	1	\$ 29.99	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Cebolla	\$ 35.50	4	\$ 142.00	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Mango	\$ 50.00	2	\$ 100.00	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Cebolla	\$ 35.50	2	\$ 71.00	<button>Actualizar</button> <button>Eliminar</button>
Maria López	Cebolla	\$ 35.50	2	\$ 71.00	<button>Actualizar</button> <button>Eliminar</button>



11. Búsqueda de producto (naranja)

The screenshot shows a search results page for "naranja". The search bar contains "naranja". Below it are filters for "Categoría:" and "Precio Máximo:", and buttons for "Filtrar" and "Limpiar". A large image of oranges is displayed, followed by a product card for "Naranja" under the category "Frutas". The card shows a price of "\$ 20.99" and a quantity input field set to "15". Buttons for "Comprar" and "Ver detalles" are present.



12. Filtro activado (categoría: frutas, precio máximo: 20.99)

The screenshot shows a web browser window for 'Don Fruter' with the URL 'localhost:4200/donfruter'. The page displays a search bar and a filter section. In the filter section, 'Categoría:' is set to 'frutas' and 'Precio Máximo:' is set to '20.99'. Below the filter are two product cards:

- Naranja**
Frutas
Precio: \$ 20.99
Quantity: 15
Buttons: Comprar (green), Ver detalles (light green)
- Banano**
Frutas
Precio: \$ 1.00
Quantity: 20
Buttons: Comprar (green), Ver detalles (light green)

The status bar at the bottom right indicates '7:31 p. m.' and '1/09/2023'.

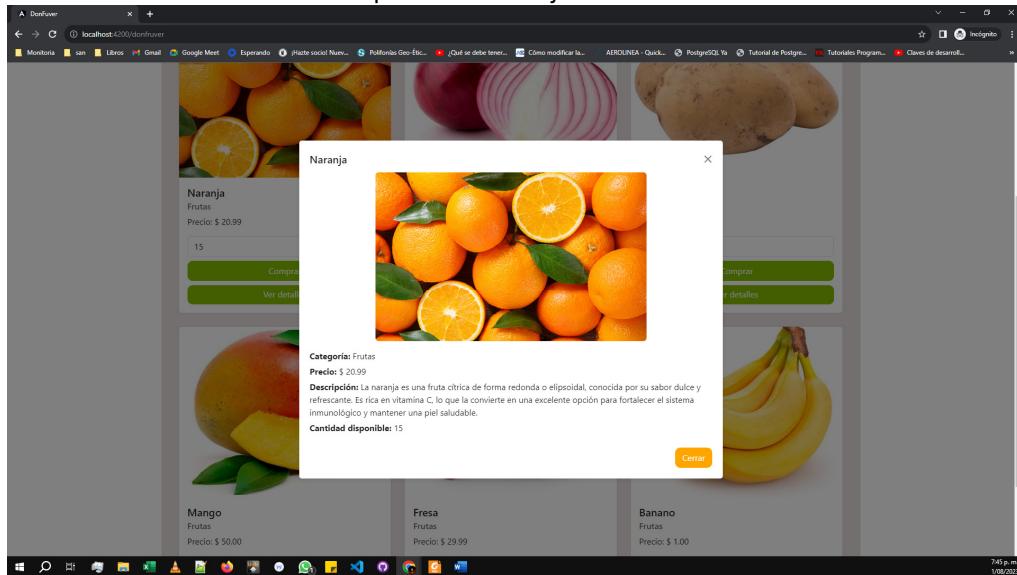
13. Filtro activado (categoría: vegetales)

The screenshot shows a web browser window for 'Don Fruter' with the URL 'localhost:4200/donfruter'. The page displays a search bar and a filter section. In the filter section, 'Categoría:' is set to 'vegetales' and 'Precio Máximo:' is set to '20'. Below the filter are two product cards:

- Cebolla**
Vegetales
Precio: \$ 35.50
Quantity: 16
Buttons: Comprar (green), Ver detalles (light green)
- Papa**
Vegetales
Precio: \$ 45.00
Quantity: 20
Buttons: Comprar (green), Ver detalles (light green)

The status bar at the bottom right indicates '7:31 p. m.' and '1/09/2023'.

14. Vista de Ver detalles del producto Naranja



Datos tabla de Usuarios para ver Rol y Email

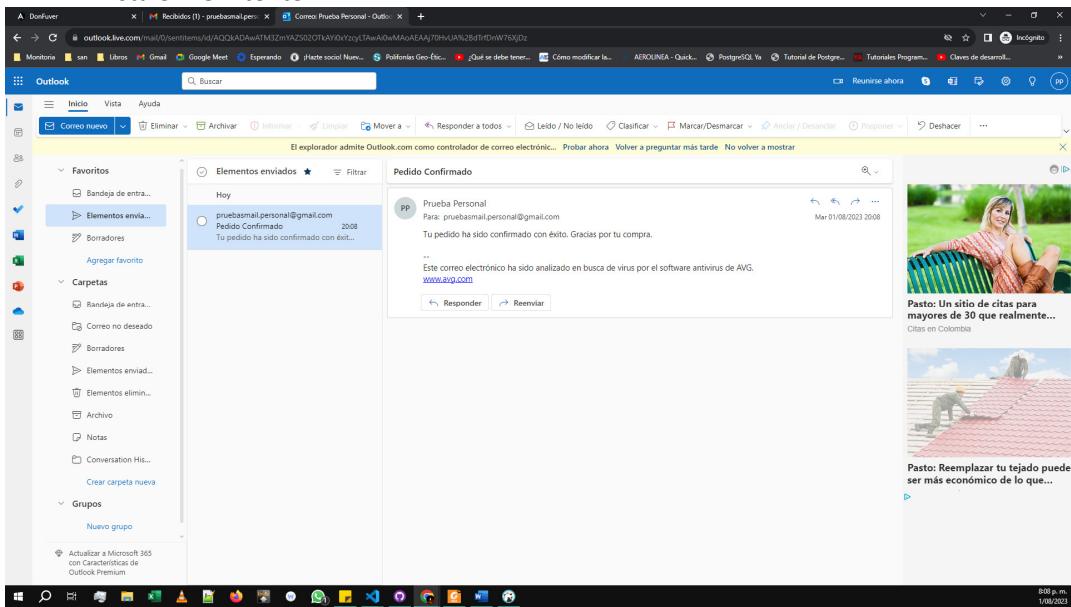
	idUsuario	Nombre	Email	Contrasena	Rol	Direccion	Ciudad	Telefono
1	0	Admin	pruebacorreo@outlook.com	123	Administrador	[NULL]	[NULL]	[NULL]
2	123	Juan Dominguez	pruebasmail.personal@gmail.com	123	Cliente	Calle 123, Colonia Centro	Ciudad A	555-123-4567
3	456	Maria Lopez	pruebasmail.personal@gmail.com	123	Cliente	Avenida Principal, Barrio X	Ciudad B	444-987-6543

el correo será enviado desde: pruebacorreo@outlook.com
el correo será enviado a: pruebasmail.personal@gmail.com

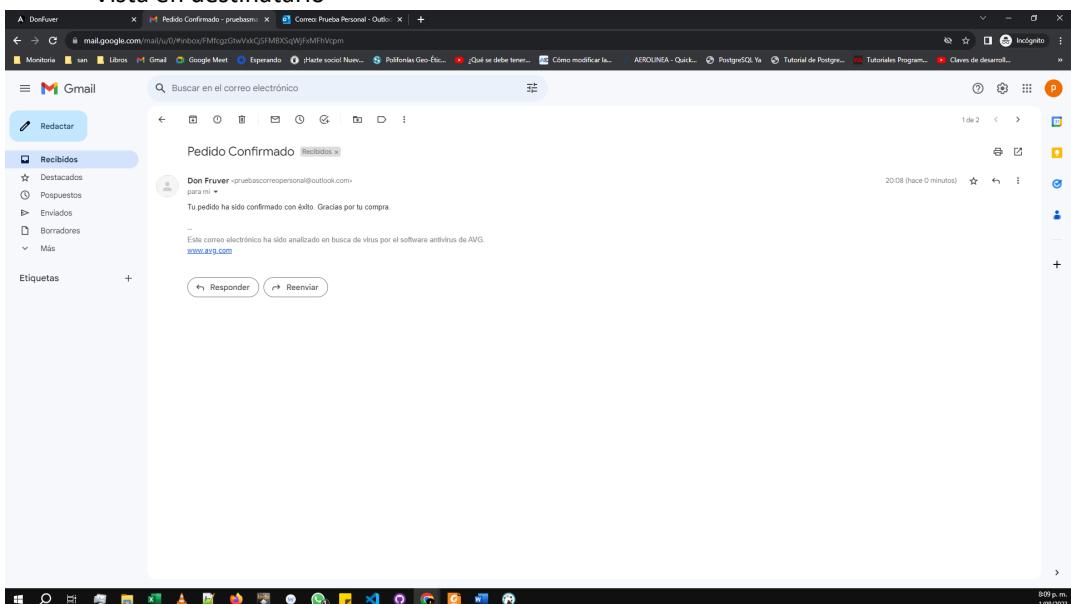
15. Correo de confirmación de pedido:

ID PEDIDO	ID CLIENTE	FECHA	ESTADO	TOTAL	ACCIONES
1	123	2023-07-25T15:30:00.000Z	Aceptado	\$ 40.99	<button>Confirmar</button> <button>Rechazar</button>
2	456	2023-07-26T09:45:00.000Z	Aceptado	\$ 80.50	<button>Confirmar</button> <button>Rechazar</button>
11	456	2023-08-01T02:46:58.000Z	Aceptado	\$ 106.50	<button>Confirmar</button> <button>Rechazar</button>
30	456	2023-08-01T16:28:07.000Z	Pendiente	\$ 100.00	<button>Confirmar</button> <button>Rechazar</button>
31	456	2023-08-01T20:18:15.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>
32	456	2023-08-01T23:30:25.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>

Vista en remitente



Vista en destinatario



16. Correo de rechazo de pedido

ID PEDIDO	ID CLIENTE	FECHA	ESTADO	TOTAL	ACCIONES
1	123	2023-07-25T15:30:00.000Z	Aceptado	\$ 40.99	<button>Confirmar</button> <button>Rechazar</button>
2	456	2023-07-26T09:45:00.000Z	Aceptado	\$ 80.50	<button>Confirmar</button> <button>Rechazar</button>
11	456	2023-08-01T02:46:58.000Z	Aceptado	\$ 106.50	<button>Confirmar</button> <button>Rechazar</button>
30	456	2023-08-01T16:28:07.000Z	Rechazado	\$ 100.00	<button>Confirmar</button> <button>Rechazar</button>
31	456	2023-08-01T20:18:15.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>
32	456	2023-08-01T23:30:25.000Z	Pendiente	\$ 71.00	<button>Confirmar</button> <button>Rechazar</button>

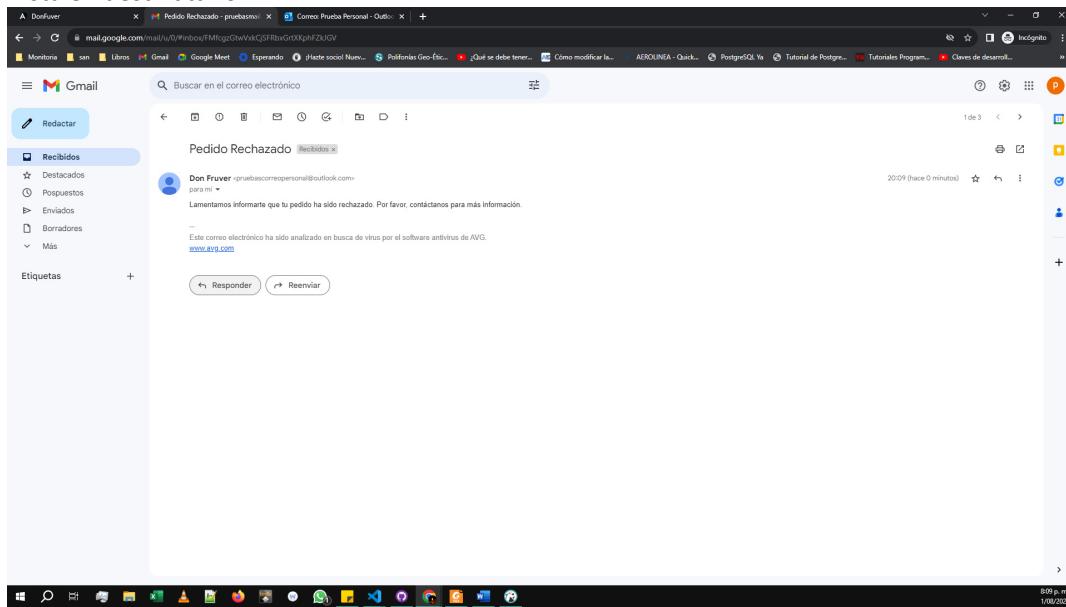
8:09 p. m.
1/08/2023

Vista en remitente

Prueba Personal
Para: pruebasmall.personal@gmail.com
Mar 01/08/2023 2009
Lamentamos informarte que tu pedido ...
Este correo electrónico ha sido analizado en busca de virus por el software antivirus de AVG.
www.avg.com

8:09 p. m.
1/08/2023

Vista en destinatario



17. Proyecto en repositorio remoto

Creamos el repositorio local y lo sincronizamos al proyecto remoto DonFruverFE.git

```
djuli@DAVIDCGZ MINGW64 /e/0_Programas/DIPLOMADO/M2/Taller3_DonFruver/DonFuver (main)
$ git init
$ git add .
warning: in the working copy of 'angular.json', LF will be replaced by CRLF the next time Git touches it
djuli@DAVIDCGZ MINGW64 /e/0_Programas/DIPLOMADO/M2/Taller3_DonFruver/DonFuver (main)
$ git commit -m "first commit"
[master 65f2ff1] first commit
 1 file changed, 3 insertions(+)
djuli@DAVIDCGZ MINGW64 /e/0_Programas/DIPLOMADO/M2/Taller3_DonFruver/DonFuver (main)
$ git remote add origin https://github.com/djulioz/DonFruverFE.git
djuli@DAVIDCGZ MINGW64 /e/0_Programas/DIPLOMADO/M2/Taller3_DonFruver/DonFuver (main)
$ git push -u origin main
Enumerating objects: 31, done.
Counting objects: 100% (31/31), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (29/29), done.
Writing objects: 100% (31/31) 112.98 KiB | 4.91 MiB/s, done.
Total 31 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/djulioz/DonFruverFE.git
 * [new branch]  .main -> main
branch 'main' set up to track 'origin/main'.
```