

**Universidad de Nariño**  
**Ingeniería de Sistemas**

**Diplomado de actualización en nuevas tecnologías para el desarrollo de  
Software**

**Taller Unidad 3 Frontend**

**Estudiante: Juan José Revelo Jojoa**

Para la creación de la aplicación Frontend en angular, se hizo uso de una plantilla construida en **Angula Material**, por tal razón, se continuó trabajando con dicha biblioteca para crear los demás componentes necesarios en la aplicación. Angular Material es una biblioteca de componentes de interfaz de usuario (UI) para angular, la cual proporciona una colección de componentes predefinidos y estilizados.

Para la construcción del aplicativo, también se utilizó la biblioteca de **Primeng**, la cual proporciona una amplia gama de componentes predefinidos y estilizados que hacen de la interfaz de usuario más atractiva e intuitiva.

Una vez instaladas las dos bibliotecas, se configura los estilos respectivos añadiéndolos directamente en el archivo angular.json, específicamente en la sección “styles” como se muestra en la siguiente imagen.

```
{ "projects": { "fruver-fe": { "architect": { "build": { "main": "src/main.ts", "polyfills": [ "zone.js" ], "tsConfig": "tsconfig.app.json", "inlineStyleLanguage": "scss", "assets": [ "src/favicon.ico", "src/assets" ], "styles": [ "node_modules/@angular/material/prebuilt-themes/purple-green.css", "src/styles.scss", "src/assets/styles/style.scss", "node_modules/material-icons/iconfont/material-icons.css", "primeicons/primeicons.css", "primeng/resources/themes/saga-blue/theme.css", "primeng/resources/primeng.min.css", "primeflex/primeflex.css" ], "scripts": [] }, "configurations": { "production": { "optimization": true, "outputHashing": "all", "sourceMap": false, "extractCss": true, "namedChunks": true, "cacheStaleCheckInterval": 0, "fileReplacements": [ { "replace": "src/environments/environment.ts", "with": "src/environments/environment.prod.ts" } ] } } } } }
```

Por otro lado, en el archivo **environments.ts** se define la propiedad “apiUrl”, la cuál se utilizará para consumir los respectivos endpoints del backend.

The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "environment.ts - fruver-fe - Visual Studio Code". The code editor displays the content of the environment.ts file:

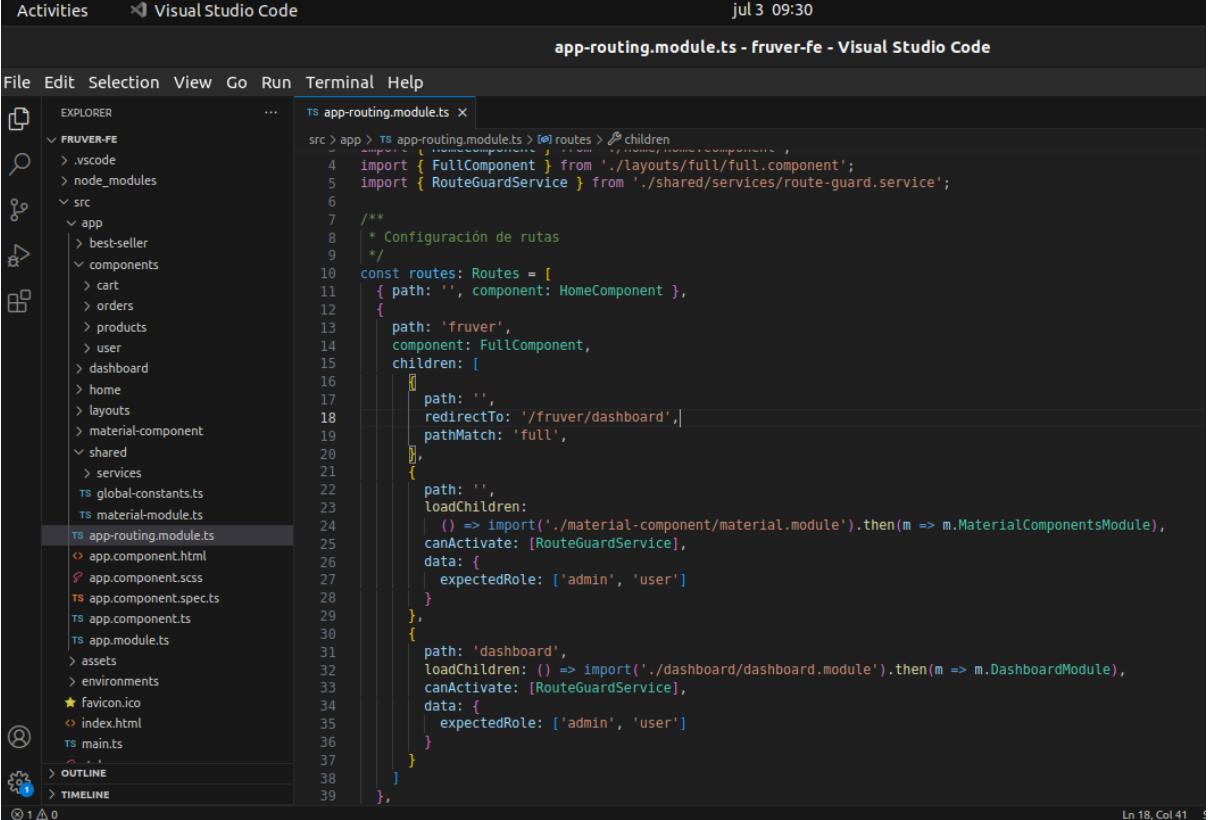
```
src > environments > TS environment.ts > ...
1 // This file can be replaced during build by using the 'fileReplacements' array.
2 // 'ng build' replaces 'environment.ts' with 'environment.prod.ts'.
3 // The list of file replacements can be found in 'angular.json'.
4
5 export const environment = {
6   production: false,
7   apiUrl: 'http://localhost:3000'
8 };
9
10 /*
11  * For easier debugging in development mode, you can import the following file
12  * to ignore zone related error stack frames such as 'zone.run', 'zoneDelegate.invokeTask'.
13  *
14  * This import should be commented out in production mode because it will have a negative impact
15  * on performance if an error is thrown.
16  */
17 // import 'zone.js/plugins/zone-error'; // Included with Angular CLI.
18
```

El archivo **app.module.ts** es un archivo clave en la aplicación Angular, ya que es el módulo principal de la aplicación, es decir, el punto de entrada para configurar y definir los componentes, servicios y otras dependencias utilizadas en la aplicación como se muestra en la siguiente imagen.

The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "app.module.ts - fruver-fe - Visual Studio Code". The code editor displays the content of the app.module.ts file:

```
src > app > ts app.module.ts > (e) ngxUiLoaderConfig
6   import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
7   import { FormsModule, ReactiveFormsModule } from '@angular/forms';
8   import { FlexLayoutModule } from '@angular/flex-layout';
9   import { MaterialModule } from './shared/material-module';
10  import { HomeComponent } from './home/home.component';
11  import { BestSellerComponent } from './best-seller/best-seller.component';
12  import { FullComponent } from './layouts/full/full.component';
13  import { HeaderComponent } from './layouts/full/header/header.component';
14  import { SidebarComponent } from './layouts/full/sidebar/sidebar.component';
15  import { HTTP_INTERCEPTORS, HttpClientModule } from '@angular/common/http';
16  import { NgxUiLoaderModule, NgxUiLoaderConfig, PB_DIRECTION } from 'ngx-ui-loader';
17  import { LoginComponent } from './components/user/login/login.component';
18  import { TokenInterceptor } from './shared/services/token.interceptor';
19  import { MatIconModule } from '@angular/material/icon';
20  import { ManageProductComponent } from './components/products/manage-product/manage-product.component';
21  import { ProductComponent } from './components/products/product/product.component';
22
23 /**
24  * Configuración para agregar un indicador de carga (loader) a la aplicación
25  * durante la carga de contenido asíncrono
26 */
27 const ngxUiLoaderConfig: NgxUiLoaderConfig = [
28   text: "Cargando...",
29   textColor: "#FFFFFF",
30   textPosition: 'center-center',
31   pbColor: "red",
32   bgsColor: "red",
33   fgsColor: "red",
34   fgsType: SPINNER.ballSpinClockwise,
35   fgsSize: 100,
36   pbDirection: PB_DIRECTION.leftToRight,
37   pbThickness: 5
38 ];
39
40 @NgModule({
41   declarations: [
42     AppComponent,
43     HomeComponent,
44     BestSellerComponent,
45     FullComponent,
46     HeaderComponent,
47     SidebarComponent,
48     LoginComponent,
49     ManageProductComponent,
50     ProductComponent
51   ],
52   imports: [
53     BrowserModule,
54     BrowserAnimationsModule,
55     FormsModule,
56     ReactiveFormsModule,
57     FlexLayoutModule,
58     MaterialModule,
59     HttpClientModule,
60     NgxUiLoaderModule,
61     AppRoutingModule
62   ],
63   providers: [
64     { provide: HTTP_INTERCEPTORS, useClass: TokenInterceptor, multi: true },
65     { provide: NgxUiLoaderConfig, useValue: ngxUiLoaderConfig }
66   ]
67 })
68 export class AppModule {}
```

Otro archivo muy importante es el **app-routing.module.ts**, ya que se utiliza para configurar las rutas y la navegación en la aplicación. El enrutamiento es el proceso de definir las rutas y los componentes asociados a esas rutas en una aplicación Angular, en tal sentido, permite navegar entre diferentes vistas o componentes según la URL actual en el navegador. En dicho archivo se configuran las rutas como se muestra en la siguiente imagen.

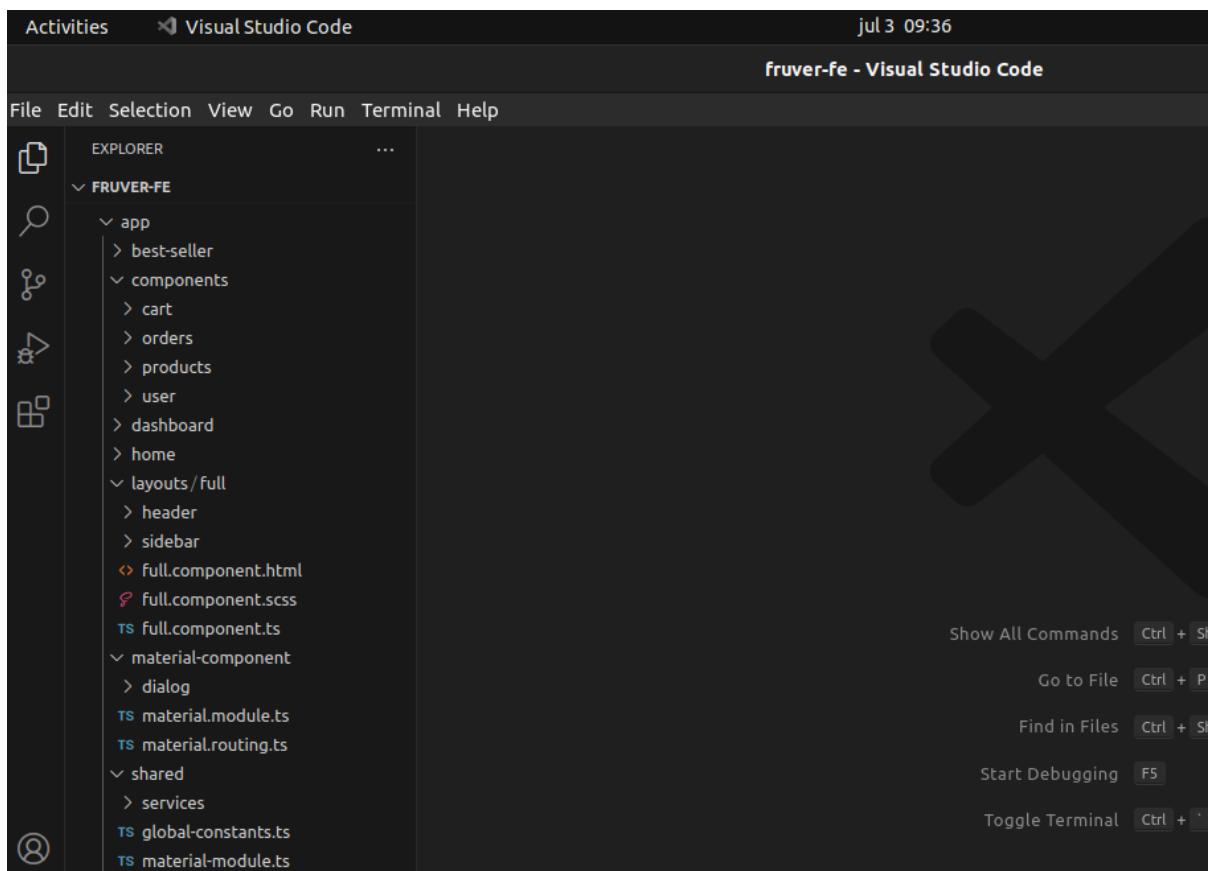


The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities > Visual Studio Code, jul 3 09:30, app-routing.module.ts - fruver-fe - Visual Studio Code
- Menu Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer View:** Shows the project structure:
  - FRUVER-FE
  - .vscode
  - node\_modules
  - src
    - app
      - best-seller
      - cart
      - orders
      - products
      - user
      - dashboard
      - home
      - layouts
      - material-component
    - shared
      - services
      - global-constants.ts
      - material-module.ts
      - app-routing.module.ts
      - app.component.html
      - app.component.scss
      - app.component.spec.ts
      - app.component.ts
      - app.module.ts
  - assets
  - environments
  - favicon.ico
  - index.html
  - main.ts
- Terminal View:** Not visible in the screenshot.
- Status Bar:** Line 18, Col 41

Como se puede visualizar en la anterior imagen, algunos paths hacen uso de la propiedad “canActivate”, la cual se utiliza para aplicar lógica de autorización o control de acceso a una ruta específica, permitiendo definir una función o un guardia (guard) que determina si un usuario tiene permiso para acceder a una ruta determinada.

Una vez concluido lo anterior, se muestra a continuación, la estructura del proyecto en las siguientes carpetas y componentes:



**best-seller:** es un componente de contenido estático, cuya función es presentar al inicio de la aplicación, los productos mejor vendidos, esto para que la aplicación se vea más interesante. El contenido html de dicha página es el siguiente.

Activities Visual Studio Code jul 3 09:41 best-seller.component.html - fruver-fe - Visual Studio Code

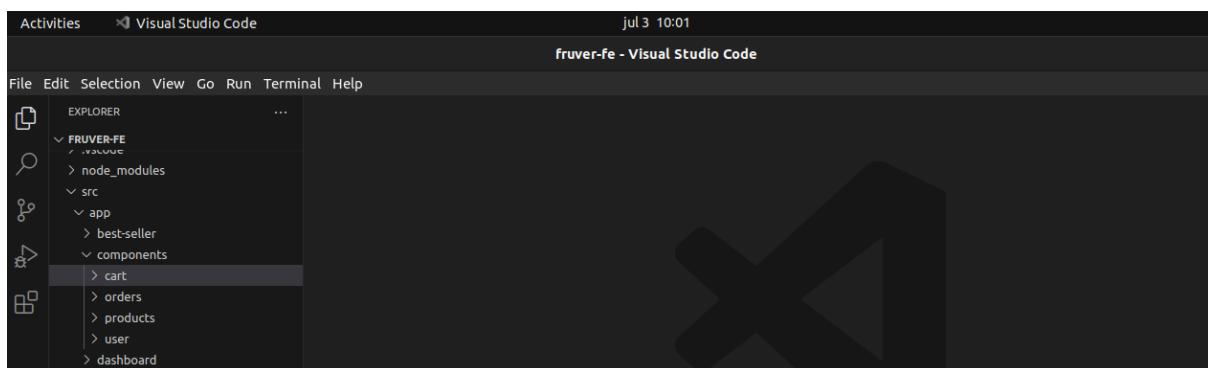
```

File Edit Selection View Go Run Terminal Help
EXPLORER ... > app > best-seller > best-seller.component.html > section#about.page-section > div.container > ul.timeline > li > div
    <section class="page-section" id="about">
        <div class="container">
            <div class="text-center">
                <h2 class="section-heading text-uppercase">Más Vendidos</h2>
            </div>
            <ul class="timeline">
                <li><a href="../../assets/img/mango.png" target="_blank">
                    <div class="timeline-image">
                </div>
                    <div class="timeline-panel">
                        <div class="timeline-heading">
                            <h4>Mango</h4>
                        </div>
                        <div class="timeline-body">
                            <p class="text-muted">Los mangos contienen enzimas digestivas que ayudan a desintoxicar el organismo.</p>
                        </div>
                    </div>
                </li>
                <li class="timeline-inverted">
                    <a href="../../assets/img/pera.png" target="_blank">
                        <div class="timeline-image">
                    </div>
                    <div class="timeline-panel">
                        <div class="timeline-heading">
                            <h4>Pera</h4>
                        </div>
                        <div class="timeline-body">
                            <p class="text-muted">Las peras son los frutos provenientes de los perales. Contienen una gran cantidad de agua y fibra.</p>
                        </div>
                    </div>
                </li>
            </ul>
        </div>
    </section>

```

**components:** es una carpeta contenedora de otras carpetas, donde irán todos los componentes que se utilizarán para gestionar el backend, es decir, dicha carpeta contendrá las siguientes carpetas:

- **cart:** en dicha carpeta irá el componente del carrito de compras (shopping-cart), y el componente para que el cliente complete el proceso de compra ingresando sus respectivos datos (complete-purchase).
- **orders:** en dicha carpeta irá el componente para ver todas las solicitudes de pedido, poderlas consolidar y enviar cuando un administrador lo requiera (view-order). También irá el componente para ver el detalle de un pedido, es decir, para mostrar los productos asociados a ese pedido (order-detail).
- **products:** en dicha carpeta irá el componente para gestionar los productos, es decir, para visualizar los productos en una tabla y poder agregar, eliminar o editar un producto (manage-product). Por otro lado, irá el componente para poder añadir o editar un respectivo producto (product), y también el componente para mostrar los productos en un DataView, los cuales serán visualizados por los clientes.
- **user:** en dicha carpeta irá el componente para poder iniciar sesión en la aplicación, y redirigir al usuario administrador al dashboard del aplicativo.



**dashboard:** es el componente que se visualiza inicialmente cuando un usuario inicia sesión en el aplicativo, el cual mostrará el usuario que está en sesión, el número de productos registrados y el número de pedidos que no se han atendido hasta el momento en tres cards.

El archivo .ts se muestra a continuación.

Activities Visual Studio Code jul 3 13:20

File Edit Selection View Go Run Terminal Help

EXPLORER dashboard.component.ts

```
src > app > dashboard > ts dashboard.component.ts > DashboardComponent
```

```
1 /**
2  * Variable para contar los productos registrados hasta el momento
3  */
4 countProducts: number = 0;
```

```
5 /**
6  * Variable para contar los pedidos que aún no se han atendido
7  */
8 countOrdersAttend: number = 0;
```

```
9 /**
10  * Variable para obtener el usuario en sesión
11  */
12 user:any;
```

```
13 /**
14  * Constructor de la clase
15  * @param productService - Inyección del servicio de productos
16  * @param orderservice - Inyección del servicio de pedidos
17  */
18 constructor(private productService: ProductService, private orderservice: OrderService) {}

19 /**
20  * Inicializador de la clase, donde se obtiene el usuario en sesión,
21  * el número de productos y el número de pedidos que no se han atendido hasta el momento
22  */
23 ngOnInit() {
24     const token:any = localStorage.getItem('token');
25     try{
26         this.user = jwt_decode(token);
27     } catch(err){
28         this.user = {};
29     }
30     this.productService.getproducts().subscribe((products:any) => {
31         this.countProducts = products.length;
32     });
33     this.orderservice.getOrders().subscribe((orders:any) => {
34         this.countOrdersAttend = orders.length;
35     });
36 }
```

OUTLINE

y su respectivo html:

Activities Visual Studio Code jul 3 10:30

File Edit Selection View Go Run Terminal Help

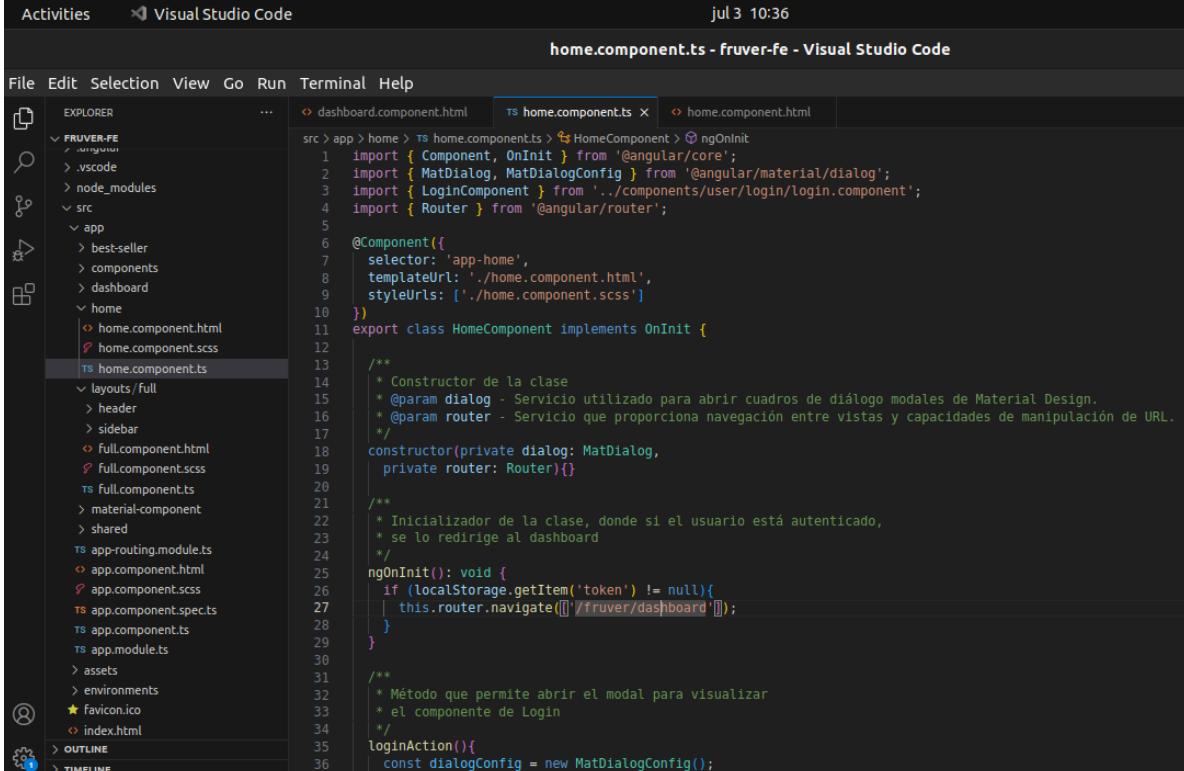
EXPLORER dashboard.component.html

```
src > app > dashboard > dashboard.component.html > body > div.row > div.column > div.card > div.container
```

```
<body>
<mat-card>
    <b><span>Dashboard</span></b>
</mat-card>
<br>
<div class="row">
    <div class="column">
        <div class="card">
            <div class="container">
                <h2 class="title">Bienvenido</h2>
                <h1 class="title">{{user.userName}}</h1>
                <b class="user">
                    <b style="margin-right: 3px;">Rol:</b>{{user.role}}
                </b>
            </div>
        </div>
    </div>
    <div class="column">
        <div class="card">
            <div class="container">
                <h2 class="title">Total de Productos:</h2>
                <h1 class="title">{{countProducts}}</h1>
                <p>
                    <button class="button" [routerLink]="/fruver/products">
                        Gestiónar productos
                    </button>
                </p>
            </div>
        </div>
    </div>
    <div class="column">
        <div class="card">
            <div class="container">
                <h2 class="title">Pedidos por Atender:</h2>
                <h1 class="title">{{countOrdersAttend}}</h1>
                <p>
                    <button class="button" [routerLink]="/fruver/orders">
                        Gestiónar pedidos
                    </button>
                </p>
            </div>
        </div>
    </div>
</div>
```

**home:** es el componente que se visualiza cuando se ingresa a la aplicación y aún no se ha iniciado sesión, es decir, este componente muestra la lista de productos y el carrito de compras para que un cliente pueda realizar un respectivo pedido.

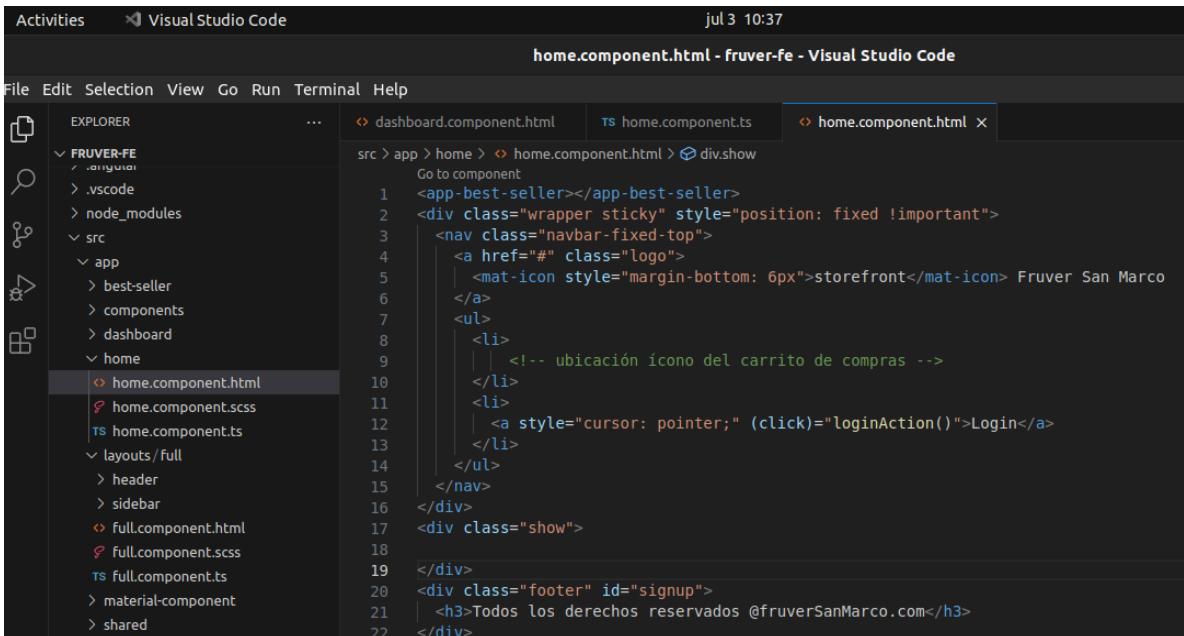
El archivo .ts se muestra a continuación.



The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and the date "jul 3 10:36". The active tab is "home.component.ts - fruver-fe - Visual Studio Code". The code editor displays the following TypeScript code for the HomeComponent:

```
src > app > home > ts home.component.ts > HomeComponent > ngOnInit
1 import { Component, OnInit } from '@angular/core';
2 import { MatDialog, MatDialogConfig } from '@angular/material/dialog';
3 import { LoginComponent } from './components/user/login/Login.component';
4 import { Router } from '@angular/router';
5
6 @Component({
7   selector: 'app-home',
8   templateUrl: './home.component.html',
9   styleUrls: ['./home.component.scss']
10 })
11 export class HomeComponent implements OnInit {
12
13 /**
14 * Constructor de la clase
15 * @param dialog - Servicio utilizado para abrir cuadros de diálogo modales de Material Design.
16 * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación de URL.
17 */
18 constructor(private dialog: MatDialog,
19             private router: Router){}
20
21 /**
22 * Inicializador de la clase, donde si el usuario está autenticado,
23 * se lo redirige al dashboard
24 */
25 ngOnInit(): void {
26   if (localStorage.getItem('token') != null){
27     this.router.navigate(['/fruver/dashboard']);
28   }
29
30 /**
31 * Método que permite abrir el modal para visualizar
32 * el componente de Login
33 */
34 loginAction(){
35   const dialogConfig = new MatDialogConfig();
36   dialogConfig
37 }
```

y su respectivo html:



The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and the date "jul 3 10:37". The active tab is "home.component.html - fruver-fe - Visual Studio Code". The code editor displays the following HTML code for the HomeComponent:

```
src > app > home > home.component.html > Go to component
1 <app-best-seller></app-best-seller>
2 <div class="wrapper sticky" style="position: fixed !important">
3   <nav class="navbar-fixed-top">
4     <a href="#" class="logo">
5       <mat-icon style="margin-bottom: 6px">storefront</mat-icon> Fruver San Marco
6     </a>
7     <ul>
8       <li>
9         <!-- ubicación ícono del carrito de compras -->
10      </li>
11      <li>
12        <a style="cursor: pointer;" (click)="loginAction()">Login</a>
13      </li>
14    </ul>
15  </nav>
16 </div>
17 <div class="show">
18
19 </div>
20 <div class="footer" id="signup">
21   <h3>Todos los derechos reservados @fruverSanMarco.com</h3>
22 </div>
```

**layouts:** es la carpeta contenedora del componente **full**, donde dicho componente es el encargado de mostrar la barra lateral de navegación y el encabezado del dashboard cuando un usuario inicia sesión.

El archivo .ts se muestra a continuación.

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... ts full.component.ts x
src > app > layouts > Full > ts full.component.ts > FullComponent > _mobileQueryListener
  1   selector: '[app-full]',
  2   templateUrl: './full.component.html',
  3   styleUrls: ['./full.component.scss']
  4 }
  5 export class FullComponent implements OnDestroy {
  6   /**
  7    * Variable que sirve para aplicar estilos y diseños diferentes según las
  8    * características de un dispositivo o ventana de navegación
  9   */
 10  mobileQuery: MediaQueryList;
 11
 12  /**
 13   * Constructor de la clase
 14   * @param changeDetectorRef - Servicio para controlar el ciclo de detección de cambios (change detection) de los componentes
 15   * @param media - Servicio utilizado para detectar cambios en las consultas de medios CSS y realizar acciones en función
 16   */
 17  constructor(
 18    changeDetectorRef: ChangeDetectorRef,
 19    media: MediaMatcher
 20  ) {
 21    this.mobileQuery = media.matchMedia('(min-width: 768px)');
 22    this._mobileQueryListener = () => changeDetectorRef.detectChanges();
 23    this.mobileQuery.addListener(this._mobileQueryListener);
 24  }
 25
 26  /**
 27   * Método que permite realizar la limpieza y liberación de recursos antes de que el componente sea destruido y eliminado
 28   */
 29  ngOnDestroy(): void {
 30    this.mobileQuery.removeListener(this._mobileQueryListener);
 31  }
 32
 33
 34
 35
 36
 37
 38
 39
 40

```

y su respectivo html:

```

File Edit Selection View Go Run Terminal Help
EXPLORER ... ts full.component.ts < full.component.html x
src > app > layouts > Full > ts full.component.ts > full.component.html > div.main-container > mat-toolbar.topbar.relative > div.navbar-header > span
  1 <div class="main-container">
  2   <mat-toolbar color="primary" class="topbar relative">
  3     <div class="navbar-header">
  4       <button mat-icon-button (click)="snay.toggle()" value="sidebarclosed" style="vertical-align: middle;">
  5         <mat-icon style="margin-top: 3px;">menu</mat-icon>
  6       </button><b>
  7         <span style="vertical-align: middle;">
  8           <mat-icon style="margin-top: 3px;">storefront</mat-icon>
  9         </span>
 10       </b>
 11       <span fxShow="false" fxShow.gt-xs>
 12         Sistema de Gestión: Fruver
 13       </span>
 14     </div>
 15     <span fxFlex></span>
 16     <app-header></app-header>
 17   </mat-toolbar>
 18   <mat-sidenav-container class="example-sidenav-container" [style.marginTop.px]="mobileQuery.matches
 19     <mat-sidenav #snay id="snay" class="dark-sidebar pl-xs" [mode]="mobileQuery.matches ? 'side' :
 20       fixedTopGap="0" [opened]="mobileQuery.matches" [disableClose]="mobileQuery.matches">
 21       <app-sidebar></app-sidebar>
 22     </mat-sidenav>
 23     <mat-sidenav-content class="page-wrapper">
 24       <div class="page-content" style="margin-bottom: 30px;">
 25         <router-outlet></router-outlet>
 26       </div>
 27     </mat-sidenav-content>
 28   </mat-sidenav-container>
 29 </div>

```

Dentro del componente **full**, se encuentran otros dos componentes:

El componente **header**, encargado de mostrar el botón del usuario para desplegar un menú y poder cerrar sesión, como se muestra a continuación en la siguiente imagen.

The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "header.component.html - fruver-fe - Visual Studio Code". The code editor displays the following HTML content:

```
src > app > layouts > full > header > header.component.html > mat-menu.mymegamenu
      Go to component
1  <button [matMenuTriggerFor]="profile" mat-icon-button class="m-r-5">
2  |   <mat-icon>account_circle</mat-icon>
3  </button>
4  <mat-menu #profile="matMenu" class="mymegamenu">
5  |   <button mat-menu-item (click)="logout()">
6  |     <mat-icon>exit_to_app</mat-icon>Logout
7  </button>
8  </mat-menu>
```

y su respectivo archivo .ts

The screenshot shows the Visual Studio Code interface with the title bar "Activities Visual Studio Code" and "header.component.ts - fruver-fe - Visual Studio Code". The code editor displays the following TypeScript code:

```
src > app > layouts > full > header > header.component.ts > HeaderComponent > logout
1  import { Component } from '@angular/core';
2  import { MatDialog, MatDialogConfig } from '@angular/material/dialog';
3  import { Router } from '@angular/router';
4  import { ConfirmationComponent } from 'src/app/material-component/dialog/confirmation/confirmation.component';
5
6  @Component({
7    selector: 'app-header',
8    templateUrl: './header.component.html',
9    styleUrls: ['./header.component.scss']
10 })
11 export class HeaderComponent {
12
13 /**
14 * Constructor de la clase
15 * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación de URL
16 * @param dialog - Servicio utilizado para abrir cuadros de diálogo modales de Material Design
17 */
18 constructor(private router: Router,
19             private dialog: MatDialog) {
20 }
21
22 /**
23 * Método que sirve para cerrar sesión, donde se abre una ventana de diálogo
24 * para confirmar el cierre de sesión, y si es afirmativo, se limpia el almacenamiento
25 * local y se redirige a la raíz del aplicativo.
26 */
27 logout(){
28   const dialogConfig = new MatDialogConfig();
29   dialogConfig.data = {
30     message: "cerrar sesión?"
31   }
32   const dialogRef = this.dialog.open(ConfirmationComponent, dialogConfig);
33   const sub = dialogRef.componentInstance.onEmitStatusChange.subscribe((user:any) => {
34     dialogRef.close();
35     localStorage.clear();
36     this.router.navigate(['/']);
37   });
38 }
```

El otro componente es **sidebar**, encargado de mostrar las opciones de menú según los privilegios correspondientes. El archivo html se muestra a continuación.

Activities Visual Studio Code jul 3 11:15 sidebar.component.html - fruver-fe - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER sidebar.component.html
FRUVER-FE
  > best-seller
  > components
  > dashboard
  > home
    > home.component.html
    > home.component.scss
    TS home.component.ts
  > layouts / full
    > header
      > header.component.html
      > header.component.scss
      TS header.component.ts
    > sidebar
      > sidebar.component.html
      > sidebar.component.scss
      TS sidebar.component.ts

```

```

src > app > layouts > full > sidebar > sidebar.component.html > mat-nav-list
  Go to component
  1 mat-nav-list appAccordion>
  2   <mat-list-item style="background-color: #e53935; color: white; margin-top: 3px;" *ngFor="let menuItem of me
  3     <a appAccordionToggle [routerLink]="/fruver/", menuItem.state" *ngIf="(menuItem.role === '' || menuItem.role === tokenPa
  4       <mat-icon style="color: white; margin-right: 5px;">{{menuItem.icon}}</mat-icon>
  5       <span style="color: white">{{menuItem.name}}</span>
  6     </a>
  7   </mat-list-item>
  8 </mat-nav-list>
  9

```

y su respectivo archivo .ts

Activities Visual Studio Code jul 3 11:15 sidebar.component.ts - fruver-fe - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER sidebar.component.ts
FRUVER-FE
  > best-seller
  > components
  > dashboard
  > home
    > home.component.html
    > home.component.scss
    TS home.component.ts
  > layouts / full
    > header
      > header.component.html
      > header.component.scss
      TS header.component.ts
    > sidebar
      > sidebar.component.html
      > sidebar.component.scss
      TS sidebar.component.ts
  > full.component.html
  > full.component.scss
  TS full.component.ts
  > material-component
  > shared
  > shared
  TS app-routing.module.ts
  > app.component.html
  > app.component.scss
  TS app.component.ts
  > app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
  > assets
  > environments
  ★ favicon.ico
  > index.html
  TS main.ts
  > styles.css
  > editorconfig
  > angular.json
  > package-lock.json
  > .editorconfig
  > OUTLINE
  > TIMELINE
  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
  src > app > layouts > full > sidebar > sidebar.component.ts > SidebarComponent > tokenPayload
  & tokenPayload: string = localStorage.getItem('token');
  & styleUrls: ['./sidebar.component.scss'];
}

export class SidebarComponent implements OnDestroy {
  /**
   * Variable que sirve para aplicar estilos y diseños diferentes según las
   * características de un dispositivo o ventana de navegación
   */
  mobileQuery: MediaQueryList;

  /**
   * Variable que obtiene el token del usuario en sesión
   */
  token: any = localStorage.getItem('token');

  /**
   * Variable que obtiene el payload del token
   */
  tokenPayload: any;

  private _mobileQueryListener: () => void;

  /**
   * Constructor de la clase
   * @param changeDetectorRef Servicio para controlar el ciclo de detección de cambios (change detection) de los componentes
   * @param media Servicio utilizado para detectar cambios en las consultas de medios CSS y realizar acciones en función de las características del dispositivo o de la pantalla
   * @param menuItems Servicio utilizado para mostrar el menú de opciones de la barra lateral
   */
  constructor(
    changeDetectorRef: ChangeDetectorRef,
    media: MediaMatcher,
    public menuItems: MenuItems
  ) {
    this.tokenPayload = jwt_decode(this.token);
    this.mobileQuery = media.matchMedia(`(min-width: 768px)`);
    this._mobileQueryListener = () => changeDetectorRef.detectChanges();
    this.mobileQuery.addListener(this._mobileQueryListener);
  }

  /**
   * Método que permite realizar la limpieza y liberación de recursos antes de que el componente sea destruido y eliminado de la memoria.
   */
  ngOnDestroy(): void {
    this.mobileQuery.removeListener(this._mobileQueryListener);
  }
}

```

Ln 27, Col 20 Spaces:2 UTF-8 LF { } Ti

**material-component:** es una carpeta que contiene un módulo y un archivo de rutas para poder visualizar la gestión de los productos, y los pedidos como se muestra en la siguiente imagen. Cabe resaltar, que solo los usuarios con el rol de “admin” podrán acceder a dichas rutas.

```
material.routing.ts - fruver-fe - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... TS material.routing.ts
src > app > material-component > ts material.routing.ts > MaterialRoutes
1 import { Routes } from '@angular/router';
2 import { ManageProductComponent } from './components/products/manage-product/manage-product.component';
3 import { RouteGuardService } from './shared/services/route-guard.service';
4 import { ViewOrderComponent } from './components/orders/view-order/view-order.component';
5
6 /**
7 * Configuración de rutas para la gestión de productos y pedidos
8 */
9 export const MaterialRoutes: Routes = [
10   {
11     path: 'products',
12     component: ManageProductComponent,
13     canActivate: [RouteGuardService],
14     data: {
15       expectedRole: ['admin']
16     }
17   },
18   {
19     path: 'orders',
20     component: ViewOrderComponent,
21     canActivate: [RouteGuardService],
22     data: {
23       expectedRole: ['admin']
24     }
25   }
26 ];
Ln 18, Col 6 Spaces: 4 UTF-8 LF {} TypeScript
```

Dentro de la carpeta material-component, se encuentra una carpeta denominada dialog, la cual contiene al componente **confirmation**, encargado de mostrar un cuadro de diálogo con las opciones de “Si” y “No” respectivamente, emitiendo un evento si la opción seleccionada es “Si”. En la siguiente imagen se muestra el archivo html.

```
confirmation.component.html - fruver-fe - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... TS confirmation.component.ts confirmation.component.html ...
src > app > material-component > dialog > confirmation > confirmation.component.html > ...
Go to component
1 <mat-dialog-content class="mat-typography">
2   <h5>Está seguro de {{details.message}}?</h5>
3 </mat-dialog-content>
4 <mat-dialog-actions align="center">
5   <button mat-raised-button color="primary" (click)="handleChangeAction()">Si</button>
6   <button mat-raised-button mat-dialog-close>No</button>
7 </mat-dialog-actions>
8 |
```

y su respectivo archivo .ts

```

File Edit Selection View Go Run Terminal Help
File Explorer ... ts confirmation.components x confirmation.component.html
src > app > material-component > dialog > confirmation > ts confirmation.component.ts > ConfirmationComponent
1 import { Component, OnInit, EventEmitter, Inject } from '@angular/core';
2 import { MAT_DIALOG_DATA } from '@angular/material/dialog';
3
4 @Component({
5   selector: 'app-confirmation',
6   templateUrl: './confirmation.component.html',
7   styleUrls: ['./confirmation.component.scss']
8 })
9 export class ConfirmationComponent implements OnInit {
10
11   /**
12    * Variable que permite emitir eventos, para
13    * la comunicación entre componentes
14    */
15   onEmitStatusChange = new EventEmitter();
16
17   /**
18    * Variable para obtener el detalle del mensaje
19    */
20   details: any = {};
21
22   /**
23    * Constructor de la clase
24    * @param dialogRef - Servicio para injectar datos en un cuadro de diálogo (dialog) que se abre
25    * utilizando el MatDialog de Angular Material.
26    */
27   constructor(@Inject(MAT_DIALOG_DATA) public dialogRef: any) {}
28
29   /**
30    * Inicializador de la clase, donde se asigna el valor del dialogData
31    * a la variable details, si el dialogRef existe.
32    */
33   ngOnInit(): void {
34     if(this.dialogRef) {
35       this.details = this.dialogRef;
36     }
37   }
38
39   /**
40    * Método que permite disparar el evento, cuando se da click en el
41    * botón de "Sí"
42    */
43   handleChangeAction() {
44     this.onEmitStatusChange.emit(true);
45   }

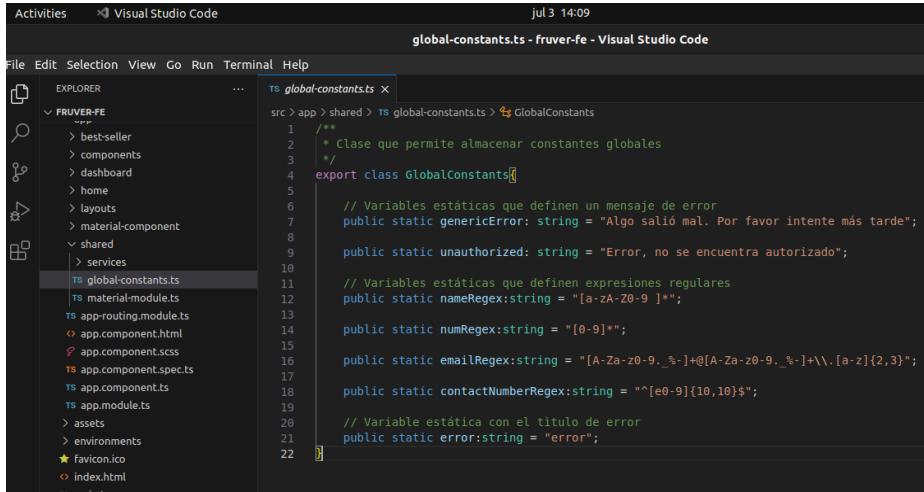
```

Finalmente se encuentra la carpeta **shared**, la cual contiene al archivo **material-module.ts**, donde se incluye todos los módulos de Angular Material que se requieren en la aplicación, como se muestra en la siguiente imagen.

```

File Edit Selection View Go Run Terminal Help
File Explorer ... ts material-module.ts
src > app > shared > ts material-module.ts > ...
9 import { MatDialogModule } from '@angular/material/dialog';
10 import { MatExpansionModule } from '@angular/material/expansion';
11 import { MatFormFieldModule } from '@angular/material/form-field';
12 import { MatGridListModule } from '@angular/material/grid-list';
13 import { MatIconModule } from '@angular/material/icon';
14 import { MatInputModule } from '@angular/material/input';
15 import { MatListModule } from '@angular/material/list';
16 import { MatMenuModule } from '@angular/material/menu';
17 import { MatPaginatorModule } from '@angular/material/paginator';
18 import { MatProgressBarModule } from '@angular/material/progress-bar';
19 import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
20 import { MatRadioModule } from '@angular/material/radio';
21 import { MatSelectModule } from '@angular/material/select';
22 import { MatSidenavModule } from '@angular/material/sidenav';
23 import { MatSliderModule } from '@angular/material/slider';
24 import { MatSlideToggleModule } from '@angular/material/slide-toggle';
25 import { MatSnackBarModule } from '@angular/material/snack-bar';
26 import { MatSortModule } from '@angular/material/sort';
27 import { MatTableModule } from '@angular/material/table';
28 import { MatTabsModule } from '@angular/material/tabs';
29 import { MatToolbarModule } from '@angular/material/toolbar';
30 import { MatTooltipModule } from '@angular/material/tooltip';
31 import { MatStepperModule } from '@angular/material/stepper';
32 import { MatBadgeModule } from '@angular/material/badge';
33 import { MatNativeDateModule, MatRippleModule } from '@angular/material/core';
34 import { MatBottomSheetModule } from '@angular/material/bottom-sheet';
35
36 import { CdkTableModule } from '@angular/cdk/table';
37 import { CdkAccordionModule } from '@angular/cdk/accordion';
38 import { AllyModule } from '@angular/cdk/ally';
39 import { BidiModule } from '@angular/cdk/bidi';
40 import { OverlayModule } from '@angular/cdk/overlay';
41 import { PlatformModule } from '@angular/cdk/platform';
42 import { ObserversModule } from '@angular/cdk/observers';
43 import { PortalModule } from '@angular/cdk/portal';
44
45 /**
46  * NgModule que incluye todos los módulos de Material que se requieren para la aplicación.
47  */
48 @NgModule({
49   exports: [
50     MatAutocompleteModule,
51     MatButtonModule,
52     MatCardModule,
53     MatChipsModule,
54     MatDatepickerModule,
55     MatDialogModule,
56     MatExpansionModule,
57     MatFormFieldModule,
58     MatGridListModule,
59     MatIconModule,
60     MatInputModule,
61     MatListModule,
62     MatMenuModule,
63     MatPaginatorModule,
64     MatProgressBarModule,
65     MatProgressSpinnerModule,
66     MatRadioModule,
67     MatSelectModule,
68     MatSidenavModule,
69     MatSliderModule,
70     MatSlideToggleModule,
71     MatSnackBarModule,
72     MatSortModule,
73     MatTableModule,
74     MatTabsModule,
75     MatToolbarModule,
76     MatTooltipModule,
77     MatStepperModule,
78     MatBadgeModule,
79     MatNativeDateModule,
80     MatRippleModule,
81     MatBottomSheetModule,
82     CdkTableModule,
83     CdkAccordionModule,
84     AllyModule,
85     BidiModule,
86     OverlayModule,
87     PlatformModule,
88     ObserversModule,
89     PortalModule
90   ]
91 })
92
93 
```

Dentro de **shared** también se encuentra el archivo **global-constants.ts**, el cual contiene constantes globales que se utilizarán en toda la aplicación, como se muestra en la siguiente imagen.



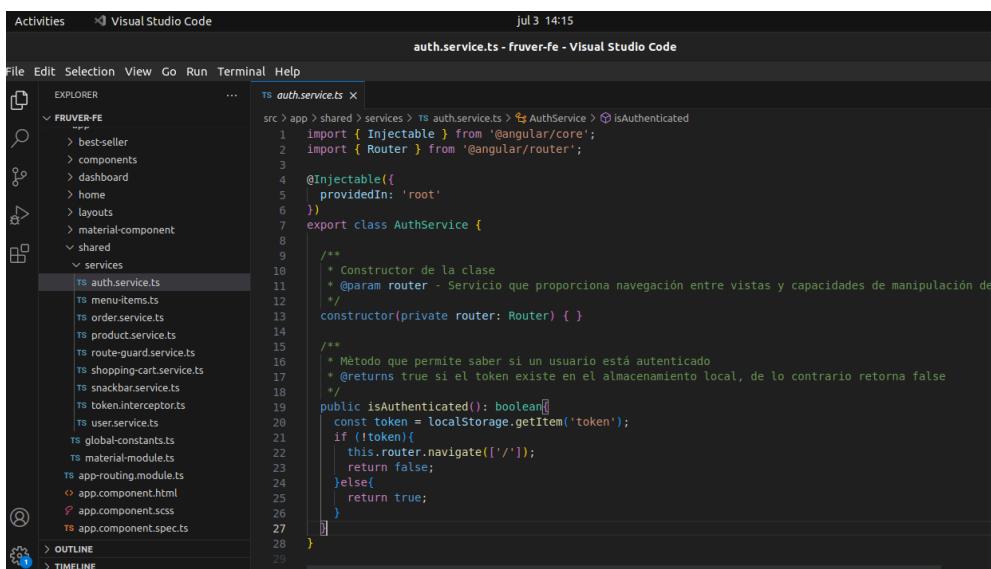
```

Activities > Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... ts global-constants.ts
src > app > shared > ts global-constants.ts > GlobalConstants
1  /**
2   * Clase que permite almacenar constantes globales
3   */
4  export class GlobalConstants{
5
6      // Variables estáticas que definen un mensaje de error
7      public static genericError: string = "Algo salió mal. Por favor intente más tarde";
8
9      public static unauthorized: string = "Error, no se encuentra autorizado";
10
11     // Variables estáticas que definen expresiones regulares
12     public static nameRegex:string = "[a-zA-Z0-9 ]*";
13
14     public static numRegex:string = "[0-9]*";
15
16     public static emailRegex:string = "[A-Za-z0-9._%-]+@[A-Za-z0-9._%-]+\.\[a-zA-Z\]{2,3}";
17
18     public static contactNumberRegex:string = "[e0-9]{10,10}$";
19
20
21     // Variable estática con el título de error
22     public static error:string = "error";
23
24 }

```

Finalmente, dentro de **shared** se encuentra la carpeta **services**, la cual se utiliza para almacenar todos los servicios. Los servicios son clases que encapsulan la lógica y la funcionalidad compartida en la aplicación, proporcionando una forma centralizada de administrar y compartir datos, realizar operaciones de red, interactuar con API externas, manipular datos, implementar lógica de negocio y más. En dicha carpeta se definen los siguientes servicios:

1. El servicio **AuthService**, el cual permite saber si un usuario se encuentra autenticado o no, como se muestra en la siguiente imagen.



```

Activities > Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER ... ts auth.service.ts
src > app > shared > services > ts auth.service.ts > AuthService > isAuthenticated
1  import { Injectable } from '@angular/core';
2  import { Router } from '@angular/router';
3
4  @Injectable({
5      providedIn: 'root'
6  })
7  export class AuthService {
8
9      /**
10      * Constructor de la clase
11      * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación de rutas
12      */
13      constructor(private router: Router) {}
14
15      /**
16      * Método que permite saber si un usuario está autenticado
17      * @returns true si el token existe en el almacenamiento local, de lo contrario retorna false
18      */
19      public isAuthenticated(): boolean{
20          const token = localStorage.getItem('token');
21          if (!token){
22              this.router.navigate(['/']);
23              return false;
24          }else{
25              return true;
26          }
27      }
28
29 }

```

2. El servicio **MenuItems**, el cual permite definir y obtener las opciones de menú que se mostrarán en el sidebar. La siguiente imagen muestra la implementación de dicho servicio.

```

Activities < Visual Studio Code
File Edit Selection View Go Run Terminal Help
menu-items.ts - fruver-fe - Visual Studio Code
src > app > shared > services > ts menu-items.ts > ↗ Menuitems
1  export interface Menu{
2    state: string;
3    name: string;
4    icon: string;
5    role: string;
6  }
7  /**
8   * Arreglo con las tres opciones que se visualizarán en el menú
9  */
10 const MENUITEMS = [
11   { state: 'dashboard', name: 'Dashboard', icon: 'dashboard', role: '' },
12   { state: 'products', name: 'Gestión de productos', icon: 'inventory_2', role: 'admin' },
13   { state: 'orders', name: 'Gestión de pedidos', icon: 'import_contacts', role: 'admin' }
14 ]
15 @Injectable({
16   providedIn: 'root'
17 })
18 export class MenuItems {
19   /**
20    * Método que sirve para obtener la lista de opciones
21    * @returns lista de opciones
22   */
23   getMenuItems(): Menu[]{
24     return MENUITEMS;
25   }
26 }
27
28
29
30
31
32
33
34

```

3. El servicio **OrderService**, el cual permite gestionar las solicitudes de compra realizadas por los clientes, consumiendo los respectivos endpoints generados en el backend. La siguiente imagen muestra la implementación de dicho servicio.

```

Activities < Visual Studio Code
File Edit Selection View Go Run Terminal Help
order.service.ts - fruver-fe - Visual Studio Code
src > app > shared > services > ts order.service.ts > ↗ OrderService
1  import { environment } from 'src/environments/environment';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class OrderService {
7
8    /**
9     * Variable que captura la url principal para consumir los respectivos endpoints
10    */
11    url = environment.apiUrl;
12
13    /**
14     * Constructor de la clase
15     * @param httpClient - Servicio injectado que permite realizar solicitudes HTTP
16     */
17    constructor(private httpClient: HttpClient) { }
18
19    /**
20     * Método que permite el registro de una solicitud de compra
21     * @param order - Pedido a registrar
22     * @returns Observable de la consulta
23     */
24    addOrder(order: any): Observable<any> {
25      return this.httpClient.post(`${this.url}/orders`, order);
26    }
27
28    /**
29     * Método que permite consolidar y notificar al cliente el envío de su pedido
30     * @param id - Id del pedido
31     * @returns Observable de la consulta
32     */
33    dispatchOrder(id: any): Observable<any> {
34      return this.httpClient.put(`${this.url}/orders`, { id });
35    }
36
37    /**
38     * Método que permite obtener todos los pedidos, que aún no han sido atendidos
39     * @returns Observable de la consulta
40     */
41    getOrders(): Observable<any> {
42      return this.httpClient.get(`${this.url}/orders`);
43    }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
167
168
169
169
170
171
172
173
174
175
176
177
177
178
179
179
180
181
182
183
184
185
186
187
187
188
189
189
190
191
192
193
194
195
196
197
197
198
199
199
200
201
202
203
204
205
206
207
207
208
209
209
210
211
212
213
214
215
215
216
217
217
218
219
219
220
221
222
223
224
225
225
226
227
227
228
229
229
230
231
232
233
234
234
235
236
236
237
238
238
239
239
240
241
241
242
243
243
244
245
245
246
246
247
247
248
248
249
249
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320

```

4. El servicio **ProductService**, el cual permite gestionar los productos en cuanto a registro, actualización, obtención y eliminación de productos, consumiendo los respectivos endpoints generados en el backend. La siguiente imagen muestra la implementación de dicho servicio.

```

product.service.ts - fruver-fe - Visual Studio Code
File Edit Selection View Go Run Terminal Help
File Explorer Search Services product.service.ts
src > app > shared > services > ts product.service.ts > ProductService > addProduct
... ts product.service.ts
  9  export class ProductService {
10    /**
11     * Variable que captura la url principal para consumir los respectivos endpoints
12     */
13    url = environment.apiUrl;
14
15    /**
16     * Constructor de la clase
17     * @param httpClient - Servicio injectado que permite realizar solicitudes HTTP
18     */
19    constructor(private httpClient: HttpClient) {}
20
21    /**
22     * Método que permite el registro de un producto
23     * @param product - Producto a registrar
24     * @returns Observable de la consulta
25     */
26    addProduct(product: any): Observable<any> {
27      return this.httpClient.post(`${this.url}/products`, product);
28    }
29
30    /**
31     * Método que permite la actualización de un producto
32     * @param product - Producto a actualizar
33     * @returns Observable de la consulta
34     */
35    updateProduct(product: any): Observable<any> {
36      return this.httpClient.put(`${this.url}/products`, product);
37    }
38
39    /**
40     * Método que permite obtener todos los productos registrados hasta el momento
41     * @returns Observable de la consulta
42     */
43    getProducts(): Observable<any> {
44      return this.httpClient.get(`${this.url}/products`);
45    }
46
47    /**
48     * Método que permite eliminar un producto
49     * @param id - Id del producto a eliminar
50     * @returns Observable de la consulta
51   }

```

5. El servicio **RouteGuardService**, el cual permite controlar si un determinado usuario tiene permiso para acceder a una ruta específica o componente. La siguiente imagen muestra la implementación de dicho servicio.

```

route-guard.service.ts - fruver-fe - Visual Studio Code
File Edit Selection View Go Run Terminal Help
File Explorer Search Services route-guard.service.ts
src > app > shared > services > ts route-guard.service.ts > RouteGuardService > canActivate
... ts route-guard.service.ts
  10  providedIn: 'root'
  11  export class RouteGuardService {
  12
  13    /**
  14     * Constructor de la clase
  15     * @param auth - Inyección del servicio de autenticación
  16     * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación de URL
  17     * @param snackBarService - Inyección del servicio de para mostrar mensajes
  18     */
  19    constructor(public auth: AuthService,
  20              public router: Router,
  21              private snackBarService: MatSnackBar) {}
  22
  23    /**
  24     * Método que se utiliza en el enrutamiento, para controlar si el usuario
  25     * tiene permiso para acceder a una ruta específica o componente
  26     * @param route - Servicio que contiene la información sobre una ruta asociada a un componente cargado
  27     * en un momento determinado
  28     * @returns un valor booleano, que determina si el usuario está autenticado y cuenta con permisos
  29     */
  30    canActivate(route: ActivatedRouteSnapshot): boolean {
  31      let expectedRoleArray: any = route.data['expectedRole'];
  32      let expectedRoleArray = expectedRoleArray['expectedRole'];
  33
  34      const token: any = localStorage.getItem('token');
  35
  36      let tokenPayload: any;
  37      try {
  38        tokenPayload = jwt_decode(token);
  39      } catch (err) {
  40        localStorage.clear();
  41        this.router.navigate(['']);
  42      }
  43
  44      let checkRole = false;
  45
  46      expectedRoleArray.forEach(item => {
  47        if (expectedRole == item['role']) {
  48          checkRole = true;
  49        }
  50      });
  51
  52      if (tokenPayload.role == 'user' || tokenPayload.role == 'admin') {
  53        if (this.auth.isAuthenticated() && checkRole) {
  54          return true;
  55        }
  56
  57        this.snackBarService.openSnackBar(GlobalConstants.unauthorized, GlobalConstants.error);
  58
  59        this.router.navigate(['/fruver/dashboard']);
  60
  61        localStorage.clear();
  
```

6. El servicio **ShoppingCartService**, el cual permite gestionar todo lo relacionado al carrito de compras, es decir, añadir un producto al carrito, actualizarlo, eliminarlo, encontrarlo por su id, obtener el valor total de la compra, reiniciar el carrito de compras y obtener dicho carrito. Las siguientes imágenes muestra la implementación de dicho servicio.

```

Activities < Visual Studio Code jul 3 14:32
shopping-cart.service.ts - fruver-fe - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ... ts route-guard.service.ts ts shopping-cart.service.ts
src > app > shared > services > ts shopping-cart.service.ts > ShoppingCartService
5   providedIn: 'root'
6 }
7 )
8 export class ShoppingCartService {
9 /**
10  * Variable que captura la url principal para consumir los respectivos endpoints
11  */
12 url = environment.apiUrl;
13
14 /**
15  * Constructor de la clase
16  */
17 constructor() { }
18
19 // Variable que se utiliza para almacenar los productos en el carrito de compras
20 private myList:any[]=[];
21
22 // Carrito observable (emitir y suscribirse)
23 private myCart = new BehaviorSubject<any>([]);
24 myCart$ = this.myCart.asObservable();
25
26 /**
27  * Método que permite añadir un producto al carrito de compras
28  * @param product - Producto a almacenar en el carrito de compras
29  */
30 addProduct(product:any){
31   if (this.myList.length === 0){
32     product.amount = 1;
33     this.myList.push(product);
34     this.myCart.next(this.myList);
35   }else{
36     const productExist = this.myList.find((element:any) => {
37       return element.id === product.id;
38     });
39     if (productExist){
40       product.amount += 1;
41       this.myCart.next(this.myList);
42     }else{
43       product.amount = 1;
44       this.myList.push(product);
45       this.myCart.next(this.myList);
46     }
47   }
48 }
49

```

```

Activities < Brave Web Browser jul 3 14:32
shopping-cart.service.ts - fruver-fe - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER ... ts route-guard.service.ts ts shopping-cart.service.ts
src > app > shared > services > ts shopping-cart.service.ts > ShoppingCartService
51 /**
52  * Método que permite eliminar un producto del carrito de compras
53  * @param id - Id del producto
54  */
55 deleteProduct(id:number){
56   this.myList = this.myList.filter( (product:any) => {
57     return product.id !== id;
58   });
59   this.myCart.next(this.myList);
60
61 /**
62  * Método que permite obtener un producto del carrito de compras
63  * @param id - Id del producto
64  */
65 findProductById(id: number){
66   return this.myList.find( (product:any) => {
67     return product.id === id;
68   });
69 }
70
71 /**
72  * Método que permite calcular el valor total de la compra
73  * @returns el valor total de la compra
74  */
75 totalCart():number {
76   const total = this.myList.reduce(function (acc, product) { return acc + (product.amount * product.price); }, 0)
77   return total
78 }
79
80 /**
81  * Método que permite reiniciar el carrito de compras
82  */
83 resetCart(){
84   this.myList = [];
85   this.myCart.next(this.myList);
86 }
87
88 /**
89  * Método que permite obtener el carrito de compras
90  * @returns carrito de compras
91  */
92 getList(): any[]{
93   return this.myList;
94 }

```

7. El servicio **SnackbarService**, el cual permite mostrar mensajes en cualquier parte de la aplicación. La siguiente imagen muestra la implementación de dicho servicio.

```

File Edit Selection View Go Run Terminal Help
File Explorer ... TS route-guard.service.ts TS shopping-cart.service.ts TS snackbar.service.ts
src > app > shared > services > snackbar.service.ts > SnackbarService > openSnackBar
1 import { Injectable } from '@angular/core';
2 import { MatSnackBar } from '@angular/material/snack-bar';
3
4 @Injectable({
5   providedIn: 'root'
6 })
7 export class SnackbarService {
8
9 /**
10 * Constructor de la clase
11 * @param snackBar - Servicio de envío de mensajes de snack bar Material Design
12 */
13 constructor(private snackbar: MatSnackBar) { }
14
15 /**
16 * Método que permite mostrar un mensaje en donde se requiera
17 * @param message - Mensaje a mostrar
18 * @param action - Acción a establecer
19 */
20 openSnackBar(message:string, action:string){
21   if (action == 'error'){
22     this.snackbar.open(message, '',{
23       horizontalPosition: 'center',
24       verticalPosition: 'top',
25       duration: 2500,
26       panelClass: ['black-snackbar']
27     });
28   }else{
29     this.snackbar.open(message, '',{
30       horizontalPosition: 'center',
31       verticalPosition: 'top',
32       duration: 2500,
33       panelClass: ['green-snackbar']
34     });
35   }
36 }

```

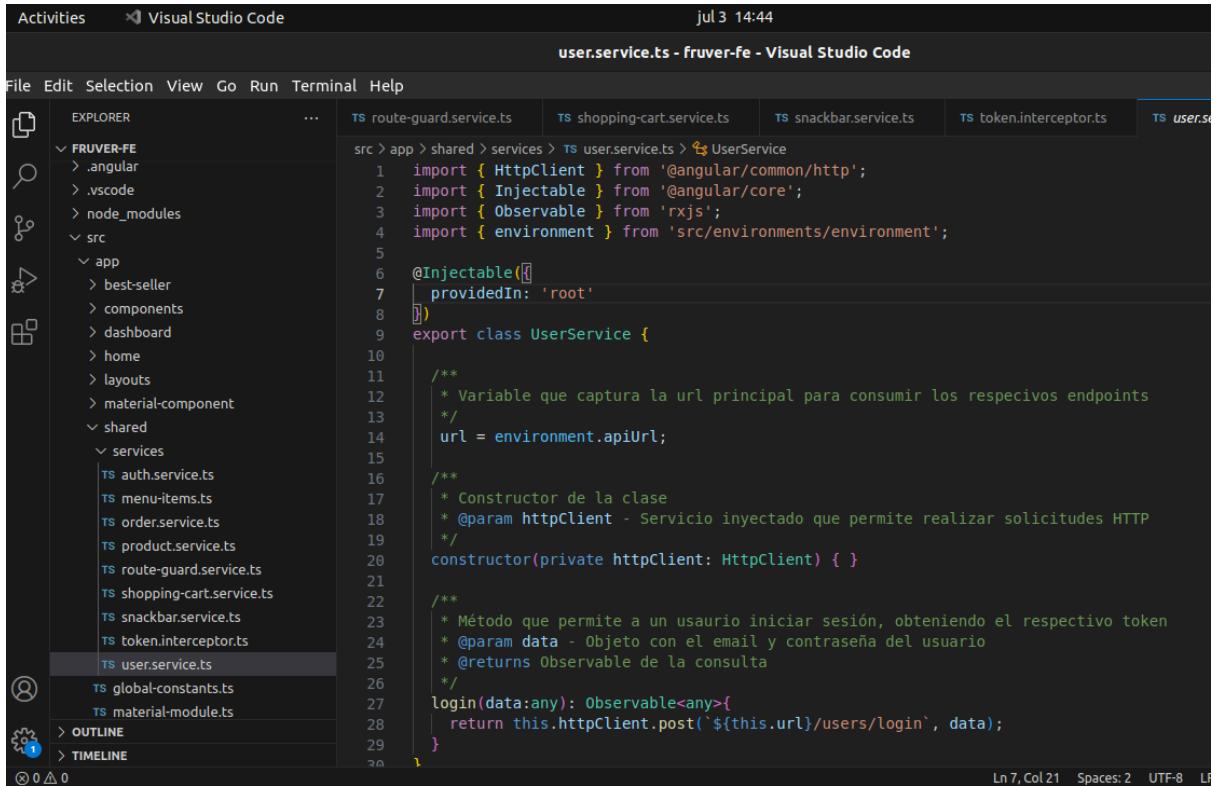
7. El servicio **TokenInterceptorService**, el cual permite interceptar y modificar las solicitudes HTTP salientes, y las respuestas entrantes antes de que lleguen a su destino final . La siguiente imagen muestra la implementación de dicho servicio.

```

File Edit Selection View Go Run Terminal Help
File Explorer ... TS route-guard.service.ts TS shopping-cart.service.ts TS snackbar.service.ts TS token.interceptor.ts
src > app > shared > services > token.interceptor.ts > TokenInterceptor > intercept
12 @Injectable()
13 export class TokenInterceptor implements HttpInterceptor {
14
15 /**
16 * Constructor de la clase
17 * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación
18 */
19 constructor(private router: Router) {}
20
21 /**
22 * Método que permite interceptar y modificar las solicitudes HTTP salientes,
23 * y las respuestas entrantes antes de que lleguen a su destino final.
24 * Cada vez que se consuma un endpoint, se enviará el token en el encabezado
25 * @param request - Solicitud entrante
26 * @param next - Manejador de solicitudes
27 * @returns Observable de la consulta
28 */
29 intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
30   const token = localStorage.getItem('token');
31   if (token){
32     // Se modifica la solicitud antes de enviarla
33     request = request.clone({
34       setHeaders: { authorization: `Bearer ${token}` }
35     });
36   }
37   // Enviar la solicitud modificada al siguiente manejador
38   return next.handle(request).pipe(
39     catchError((err:any)=>{
40       if (err instanceof HttpErrorResponse){
41         if (err.status === 401 || err.status === 403){
42           if(this.router.url !== '/') {
43             localStorage.clear();
44             this.router.navigate(['/']);
45           }
46         }
47       }
48     })
49   );
50 }

```

8. El servicio **UserService**, el cual permite iniciar sesión a un usuario, obteniendo el respectivo token desde el backend, como se muestra a continuación.



The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Activities, Visual Studio Code, jul 3 14:44, user.service.ts - fruver-fe - Visual Studio Code
- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Explorer View:** Shows the project structure under 'FRUVER-FE': .angular, .vscode, node\_modules, src (containing app, shared, services), and various service files like auth.service.ts, menu-items.ts, order.service.ts, product.service.ts, route-guard.service.ts, shopping-cart.service.ts, snackbar.service.ts, token.interceptor.ts, and user.service.ts (which is selected).
- Code Editor:** Displays the code for 'user.service.ts'. The code defines a UserService class with an injectable constructor taking an HttpClient. It includes methods for logging in users by posting to the '/users/login' endpoint.
- Status Bar:** Ln 7, Col 21, Spaces: 2, UTF-8, LF

## Carpeta components

Como se mencionó anteriormente, components es una carpeta contenedora de otras carpetas, donde se encuentran todos los componentes que se utilizarán para gestionar el backend, es decir, dicha carpeta contendrá las siguientes carpetas:

**user:** en dicha carpeta se encuentra el componente **login**, el cual muestra un formulario para que el usuario pueda ingresar su email y contraseña, y así poder iniciar sesión. Si el usuario logra autenticarse de manera correcta, la aplicación lo redirigirá al dashboard de gestión del aplicativo.

La siguiente imagen muestra el archivo html del componente, donde se utiliza un FormGroup para gestionar los controles del formulario.

Activities ➔ Visual Studio Code jul 4 06:45

login.component.html - fruver-fe - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER login.component.html ...
FRUVER-FE
> .angular
> .vscode
> node_modules
src
> app
> best-seller
> components
> cart
> orders
> products
> user/login
> dashboard
> home
> layouts
> material-component
> shared
TS login.component.ts
TS app-routing.module.ts
TS app.component.html
TS app.component.scss
TS app.component.spec.ts
TS app.component.ts
TS app.module.ts
> assets
> environments
★ favicon.ico
> index.html

```

```

<mat-toolbar color="primary" style="margin: 0px;">
  <mat-toolbar-row fxLayout="row">
    <span class="title-center">LOGIN</span>
  </mat-toolbar-row>
</mat-toolbar>
<mat-dialog-content>
  <form [formGroup]="loginForm">
    <div fxFlex fxLayout="column">
      <mat-form-field appearance="fill" fxFlex>
        <mat-label>Email</mat-label>
        <input matInput formControlName="email" required>
        <mat-error *ngIf="loginForm.controls.email.touched && loginForm.controls.email.invalid">
          <span *ngIf="loginForm.controls.email.errors.required">Este campo es obligatorio</span>
          <span *ngIf="loginForm.controls.email.errors.pattern">Este campo es inválido</span>
        </mat-error>
      </mat-form-field>
      <mat-form-field appearance="fill" fxFlex>
        <mat-label>Password</mat-label>
        <input type="password" matInput formControlName="password" required>
        <mat-error *ngIf="loginForm.controls.password.touched && loginForm.controls.password.invalid">
          <span *ngIf="loginForm.controls.password.errors.required">Este campo es obligatorio</span>
          <span *ngIf="loginForm.controls.password.errors.pattern">Este campo es inválido</span>
        </mat-error>
      </mat-form-field>
    </div>
  </form>
</mat-dialog-content>
<mat-dialog-actions align="center" style="margin: 14px;">
  <button mat-raised-button color="primary" type="submit" (click)="handleSubmit()" [disabled]!="!(loginForm.valid && loginForm.dirty)">Login</button>
  <button mat-raised-button color="primary" mat-dialog-close>Cerrar</button>
</mat-dialog-actions>

```

La siguiente imagen muestra el archivo .ts, el cuál se encuentra correctamente documentado.

Activities ➔ Visual Studio Code jul 4 06:46

login.component.ts - fruver-fe - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
EXPLORER login.component.html ts login.component.ts ...
FRUVER-FE
> .angular
> .vscode
> node_modules
src
> app
> best-seller
> components
> cart
> orders
> products
> user/login
TS login.component.html
TS login.component.scss
TS login.component.ts
> dashboard
> home
> layouts
> material-component
> shared
TS app-routing.module.ts
TS app.component.html
TS app.component.scss
TS app.component.spec.ts
TS app.component.ts
TS app.module.ts
> assets
> environments
★ favicon.ico
> index.html
TS main.ts
TS styles.scss
TS editorconfig
Angular.json
package-lock.json
package.json
README.md
> OUTLINE
> TIMELINE

```

```

loginForm: any = FormGroup;
responseMessage: any;

constructor(private formBuilder: FormBuilder,
  private router: Router,
  private userService: UserService,
  public dialogRef: MatDialogRef<LoginComponent>,
  private ngxService: NgxUiLoaderService,
  private snackbarservice: SnackBarService) {}

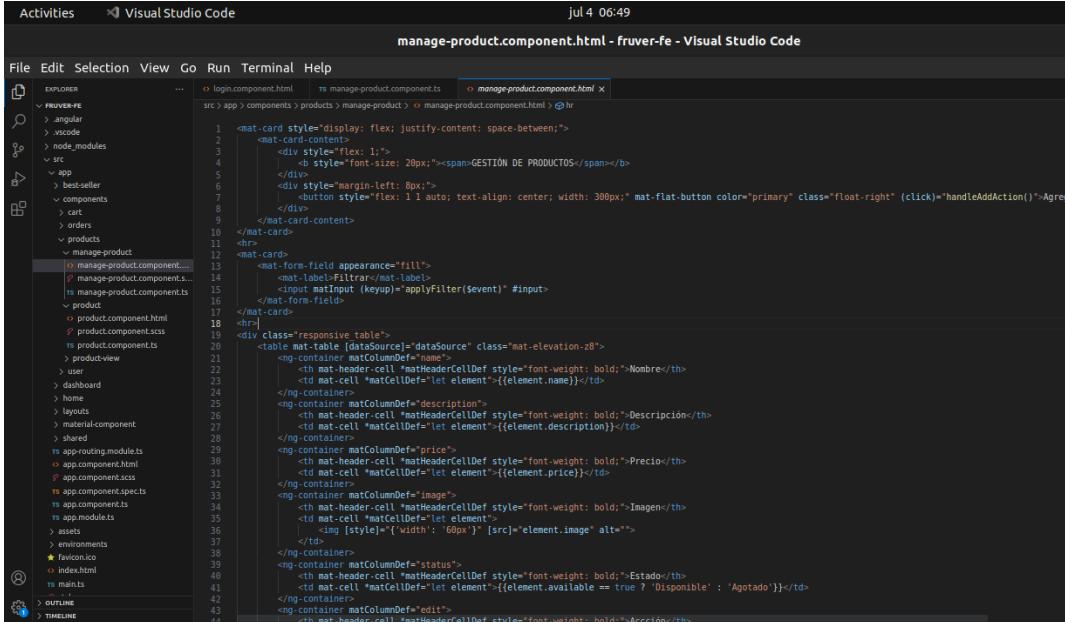
 ngOnInit(): void {
  this.loginForm = this.formBuilder.group({
    email: [null, [Validators.required, Validators.pattern(GlobalConstants.emailRegex)]],
    password: [null, [Validators.required]]
  });
}

 /**
 * Método que permite iniciar sesión a un usuario,
 * donde si el usuario y contraseña son correctos,
 * se redirige al dashboard de administración.
 */
 handleSubmit() {
  this.ngxService.start();
  let formData = this.loginForm.value;
  let data = {
    email: formData.email,
    password: formData.password
  };
  this.userService.login(data).subscribe(
    res => {
      if (res.success) {
        this.router.navigate(['/dashboard']);
        this.snackbarservice.showSuccess(res.message);
      } else {
        this.snackbarservice.showError(res.message);
      }
    },
    err => {
      this.snackbarservice.showError('Error al iniciar sesión');
    }
  );
}

```

**products:** en dicha carpeta se encuentran los siguientes componentes:

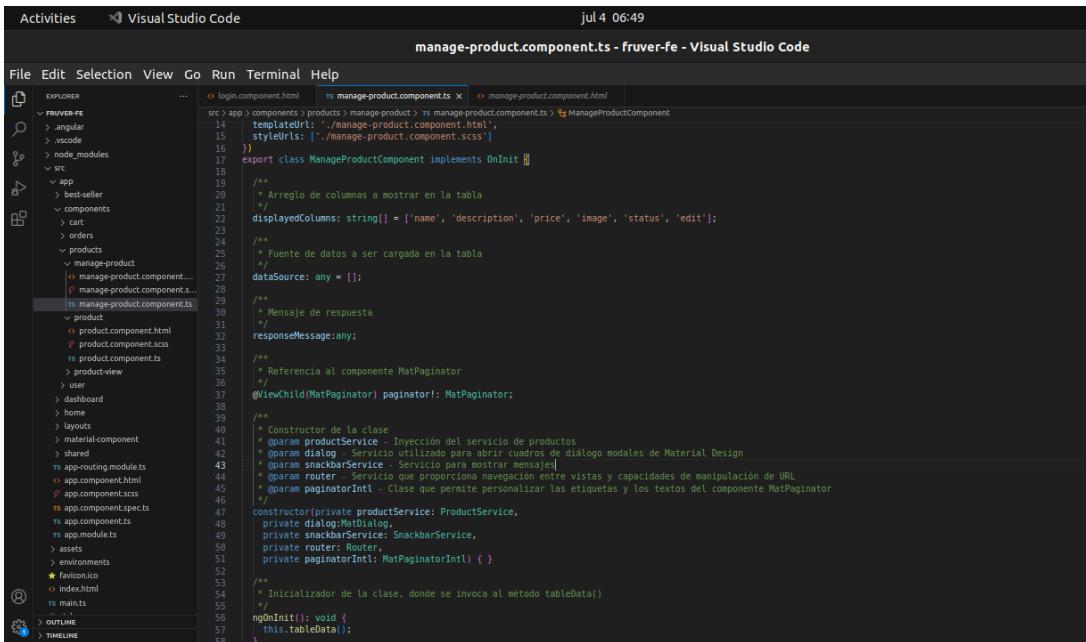
El componente **manage-product**, encargado de mostrar los productos en una tabla de Angular Material para poderlos gestionar correctamente, donde se cuenta con un botón para registrar un nuevo producto, y cada registro cuenta con los botones para editar un producto y eliminarlo respectivamente. La siguiente imagen muestra el archivo html del componente.



```
<mat-card style="display: flex; justify-content: space-between;>
  <mat-card-content>
    <div style="flex: 1;">
      <b style="font-size: 20px;"><span>GESTIÓN DE PRODUCTOS</span></b>
    </div>
    <div style="margin-left: 8px;">
      <button style="flex: 1 1 auto; text-align: center; width: 300px;" mat-flat-button color="primary" class="float-right" (click)="handleAddAction()">Agregar</button>
    </div>
  </mat-card-content>
</mat-card>

<mat-form-field appearance="fill">
  <mat-label>Filtrar</mat-label>
  <input matInput (keyup)="applyFilter($event)" type="text">
</mat-form-field>
<hr>
<div class="responsive-table">
  <table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
    <ng-container matColumnDef="Nombre">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Nombre</th>
      <td mat-cell *matCellDef="let element">{{element.name}}</td>
    </ng-container>
    <ng-container matColumnDef="Descripción">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Descripción</th>
      <td mat-cell *matCellDef="let element">{{element.description}}</td>
    </ng-container>
    <ng-container matColumnDef="Precio">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Precio</th>
      <td mat-cell *matCellDef="let element">{{element.price}}</td>
    </ng-container>
    <ng-container matColumnDef="Imagen">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Imagen</th>
      <td mat-cell *matCellDef="let element">
        
      </td>
    </ng-container>
    <ng-container matColumnDef="status">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Estado</th>
      <td mat-cell *matCellDef="let element">{{element.available == true ? 'Disponible' : 'Agotado'}}</td>
    </ng-container>
    <ng-container matColumnDef="edit">
      <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Acción</th>
      <td>
        <button mat-icon-button (click)="handleEdit(element)">Editar</button>
        <button mat-icon-button (click)="handleDelete(element)">Eliminar</button>
      </td>
    </ng-container>
  </table>
</div>
```

Y su respectivo archivo .ts, el cuál se encuentra correctamente documentado.



```
export class ManageProductComponent implements OnInit {
  /**
   * Arreglo de columnas a mostrar en la tabla
   */
  displayedColumns: string[] = ['name', 'description', 'price', 'image', 'status', 'edit'];
  /**
   * Fuente de datos a ser cargada en la tabla
   */
  dataSource: any = [];
  /**
   * Mensaje de respuesta
   */
  responseMessage: any;
  /**
   * Referencia al componente MatPaginator
   */
  @ViewChild(MatPaginator) paginator: MatPaginator;
  /**
   * Constructor de la clase
   */
  constructor(private productService: ProductService,
             private dialog: MatDialog,
             private snackBarService: MatSnackBarService,
             private router: Router,
             private paginatorIntl: MatPaginatorIntl) {}

  /**
   * Inicializador de la clase, donde se invoca al método loadDataTable()
   */
  ngOnInit(): void {
    this.loadDataTable();
  }
}

private loadDataTable() {
  this.productService.getProducts().subscribe(data => {
    this.dataSource = data;
  });
}
```

El componente **product**, encargado de mostrar un formulario para poder registrar o editar un respectivo producto, utilizando un **FormGroup** para gestionar los controles del formulario, como se muestra en la siguiente imagen.

```

<mat-toolbar color="primary">
  <span class="title-center">{{dialogData.action}}</span>
</mat-toolbar>

<mat-dialog-content class="mat-typography">
  <form [FormGroup]=>productForm>
    <div fxFlex fxLayout="column">
      <div fxFlex fxLayout="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Nombre</mat-label>
          <input matInput formControlName="name" required>
          <mat-error *ngIf="productForm.controls.name.touched && productForm.controls.name.invalid">
            <span *ngIf="productForm.controls.name.errors.required">Este campo es obligatorio</span>
            <span *ngIf="productForm.controls.name.errors.pattern">El campo es inválido</span>
          </mat-error>
        </mat-form-field>
      </div>
      <div fxFlex="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Precio</mat-label>
          <input matInput formControlName="price" required>
          <mat-error *ngIf="productForm.controls.price.touched && productForm.controls.price.invalid">
            <span *ngIf="productForm.controls.price.errors.required">Este campo es obligatorio</span>
            <span *ngIf="productForm.controls.price.errors.pattern">El campo es inválido</span>
          </mat-error>
        </mat-form-field>
      </div>
      <div fxFlex="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Descripción</mat-label>
          <input matInput formControlName="description">
        </mat-form-field>
      </div>
      <div fxFlex="row wrap">
        <input type="file" #fileInput hidden (change)="onImageSelected($event)">
        <button style="margin-left: 5px; margin-right: 15px;" mat-raised-button color="primary" (click)="fileInput.click()"->Cargar imagen</button>
        <mat-checkbox formControlName="available">El producto está disponible?</mat-checkbox>
      </div>
    </div>
  </form>
</mat-dialog-content>

```

Y su respectivo archivo **.ts**, el cuál se encuentra correctamente documentado.

```

selector: 'app-product',
templateUrl: './product.component.html',
styleUrls: ['./product.component.scss']
)
export class ProductComponent implements OnInit{
  /**
   * Variable que permite emitir un evento, cuando el componente
   * se encuentre en modo Añadir
   */
  onAddProduct = new EventEmitter();
  /**
   * Variable que permite emitir un evento, cuando el componente
   * se encuentre en modo Editar
   */
  onEditProduct = new EventEmitter();
  /**
   * Variable que representa un grupo de controles de formulario
   */
  productForm: any = FormGroup;
  /**
   * Variable que almacena la acción del diálogo (Añadir/Editar)
   */
  dialogAction: any = "Añadir";
  /**
   * Título a mostrar en formulario
   */
  action: any = "Añadir";
  /**
   * Mensaje de respuesta
   */
  responseMessage: any;
  /**
   * Imagen de tipo file
   */
  selectedImage: any;
  /**
   * -----
   */
}

```

El componente **product-view**, encargado de mostrar los productos en un DataView para que el usuario pueda añadirlos al carrito de compras. La siguiente imagen muestra el archivo html del componente.

```

<div class="card" style="margin-top: 50px;">
  <p-dataView #dv [value]="products" [paginator]="true" [rows]="9" filterBy="name,description,price">
    <ng-template pTemplate="header">
      <div class="flex flex-column md:flex-row md:justify-content-between">
        <p-dropdown [options]="sortOptions" [placeholder]="Ordenar Por">
          <ng-template let-option="option" [onSelect]="">
            <span>{{option.value}} - {{option.label}}</span>
          </ng-template>
        <span class="mb-2 md:mb-0" style="display: flex; justify-content: space-between;">
          <input type="search" placeholder="Buscar" style="min-width: 15em; font-size: 16px;" />
          <button>Buscar</button>
        </span>
      </div>
    </ng-template>
    <ng-template let-product pTemplate="gridItem">
      <div class="col-12 md:col-4">
        <div class="product-grid-item card">
          <div class="imgDiv">
            
            <div class="product-name">{{product.name}}{{product.description}}$ {{product.price}}

```

Y su respectivo archivo .ts, el cuál se encuentra correctamente documentado.

```

styleUrls: ['./product-view.component.scss']
)
export class ProductViewComponent implements OnInit {
  /**
   * Lista de los productos
   */
  products!: any[];

  /**
   * Lista de opciones de ordenamiento
   */
  sortOptions!: any[];

  /**
   * Variable para ordenar los datos por defecto
   */
  sortOrder!: number;

  /**
   * Llave de ordenamiento
   */
  sortKey!: string;

  /**
   * Nombre de la propiedad de los datos, que se usará
   * en la clasificación de forma predeterminada
   */
  sortField!: string;

  /**
   * Constructor de la clase
   * @param productService - Inyección del servicio de productos
   * @param cartService - Inyección del servicio del carrito de compras
   */
  constructor(private productService: ProductService,
    private cartService: ShoppingCartService) {}

  /**
   * Inicializador de la clase, donde se carga los productos disponibles, en el componente
   */
  ngOnInit(): void {
    this.productService.getproducts().subscribe((response:any)=>{
      ...
    })
  }
}

```

**orders**: en dicha carpeta se encuentran los siguientes componentes:

El componente **view-order**, encargado de mostrar los pedidos que aún no se han atendido en una tabla de Angular Material para poderlos gestionar correctamente, donde cada registro cuenta con un botón para ver el detalle de un pedido respectivo, y otro botón para consolidar y notificar al cliente el envío de ese pedido. La siguiente imagen muestra el archivo html del componente.

Activities Visual Studio Code jul 4 12:57

File Edit Selection View Go Run Terminal Help

EXPLORER ... view-order.component.html x

FRUVER-FE

> angular

> vscode

> node\_modules

src

> app

> best-seller

> components

> cart

> orders

> order-detail

> view-order

> view-order.component.html

> view-order.component.scss

ts view-order.component.ts

> products

> user

> dashboard

> home

> layouts

> material-component

> dialog

ts material.module.ts

ts material.routing.ts

> shared

ts app-routing.module.ts

app.component.html

app.component.scss

ts app.component.specs.ts

ts app.component.ts

...  
OUTLINE  
TIMELINE

src > app > components > orders > view-order > view-order.component.html > responsive\_table

Go to component:

```
1 <mat-card style="padding: 20px; font-size: 20px; text-align: center;">
2   <b><span>PEDIDOS POR ATENDER</span></b>
3 </mat-card>
4 <hr>
5 <mat-card>
6   <mat-form-field appearance="fill">
7     <mat-label>Filtrar</mat-label>
8     <input matInput (keyup)="applyFilter($event)" #input>
9   </mat-form-field>
10 </mat-card>
11 <hr>
12 <div class="responsive_table">
13   <table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
14     <ng-container matColumnDef="customerName">
15       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Cliente</th>
16       <td mat-cell *matCellDef="let order">{order.customer.name}</td>
17     </ng-container>
18     <ng-container matColumnDef="customerIdentificationNumber">
19       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Cédula</th>
20       <td mat-cell *matCellDef="let order">{order.customer.identificationNumber}</td>
21     </ng-container>
22     <ng-container matColumnDef="customerEmail">
23       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Email</th>
24       <td mat-cell *matCellDef="let order">{order.customer.email}</td>
25     </ng-container>
26     <ng-container matColumnDef="shoppingAddress">
27       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Dirección de envío</th>
28       <td mat-cell *matCellDef="let order">{order.shoppingAddress}</td>
29     </ng-container>
30     <ng-container matColumnDef="total">
31       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Total</th>
32       <td mat-cell *matCellDef="let order">${order.total}</td>
33     </ng-container>
34     <ng-container matColumnDef="view">
35       <th mat-header-cell *matHeaderCellDef style="font-weight: bold;">Acciones</th>
36       <td mat-cell *matCellDef="let order">[<a href="#">Ver</a> | <a href="#">Actualizar</a> | <a href="#">Borrar</a>]</td>
37     </ng-container>
38   </table>
39 </div>
```

Y su respectivo archivo .ts, el cuál se encuentra correctamente documentado.

Activities Visual Studio Code jul 4 12:57

view-order.component.ts - fruver-fe - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

FRUVER-FE

- > angular
- > vscode
- > node\_modules
- > src
  - > app
    - > best-seller
    - > components
      - > cart
      - > orders
        - > order-detail
        - > view-order
          - view-order.component.html
          - view-order.component.scss
          - view-order.component.ts
    - > products
    - > user
    - > dashboard
    - > home
    - > layouts
    - > material-component
    - > dialog
    - TS material.module.ts
    - TS material.routing.ts
    - > shared
    - TS app-routing.module.ts
      - > app.component.html
      - app.component.scss
      - app.component.specs.ts
      - app.components.ts
    - > OUTLINE
    - > TIMELINE

El componente **order-detail**, el cual se encarga de mostrar el detalle de un pedido, es decir, los datos del cliente, los productos asociados a ese pedido, y el total de la respectiva compra en una tabla de Angular Material. La siguiente imagen muestra el archivo html del componente.

Activities Visual Studio Code jul 4 12:56

order-detail.component.html - fruver-fe - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

FRUVER-FE

- > angular
- > vscode
- > node\_modules
- > src
  - > app
    - > best-seller
    - > components
      - > cart
      - > orders
        - > order-detail
          - order-detail.component.html
    - > view-order
    - > products
    - > user
    - > dashboard
    - > home
    - > layouts
    - > material-component
      - > dialog
      - material.module.ts
      - material.routing.ts
    - > shared
    - app-routing.module.ts
    - app.component.html
    - app.component.scss
    - app.component.spec.ts
    - app.component.ts

> OUTLINE

> TIMELINE

... order-detail.component.html x

src > app > components > orders > order-detail > order-detail.component.html > mat-dialog-content.mat-typography > table#customers > tr > td

```
    Go to component
  1   <mat-toolbar color="primary">
  2     <mat-toolbar-row fxLayout="row">
  3       <span class="title-center">DETALLE DEL PEDIDO</span>
  4       <span class="spacer"></span>
  5       <button class="mat-dialog-close" mat-icon-button mat-dialog-close>
  6         |   <mat-icon>cancel</mat-icon>
  7       </button>
  8     </mat-toolbar-row>
  9   </mat-toolbar>
10
11   <mat-dialog-content class="mat-typography">
12     <table id="customers" style="margin-bottom: 12px;">
13       <tr>
14         <td width="50%"><b>Cliente:</b>{{data.customer.name}}</td>
15         <td width="50%"><b>Cédula:</b>{{data.customer.identificationNumber}}</td>
16       </tr>
17       <tr>
18         <td width="50%"><b>Teléfono:</b>{{data.customer.phone}}</td>
19         <td width="50%"><b>Email:</b>{{data.customer.email}}</td>
20       </tr>
21     </table>
22     <div class="responsive_table" style="margin-bottom: 12px;">
23       <table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
24         <ng-container matColumnDef="name">
25           <th mat-header-cell "matHeaderCellDef">Producto</th>
26           <td mat-cell "matCellDef"="let productOder">{{productOder.product.name}}</td>
27         </ng-container>
28         <ng-container matColumnDef="price">
29           <th mat-header-cell "matHeaderCellDef">Precio</th>
30           <td mat-cell "matCellDef"="let productOder">$ {{productOder.product.price}}</td>
31         </ng-container>
32         <ng-container matColumnDef="amount">
33           <th mat-header-cell "matHeaderCellDef">Cantidad</th>
34           <td mat-cell "matCellDef"="let productOder">{{productOder.amount}}</td>
35         </ng-container>
36         <ng-container matColumnDef="cambiar" style="width: 10px;">
37           <th mat-header-cell "matHeaderCellDef">

Ln 14 Col 28 Space


```

Y su respectivo archivo .ts, el cuál se encuentra correctamente documentado.

Activities Visual Studio Code jul 4 12:56

order-detail.component.ts - fruver-fe - Visual Studio Code

File Edit Selection View Go Run Terminal Help

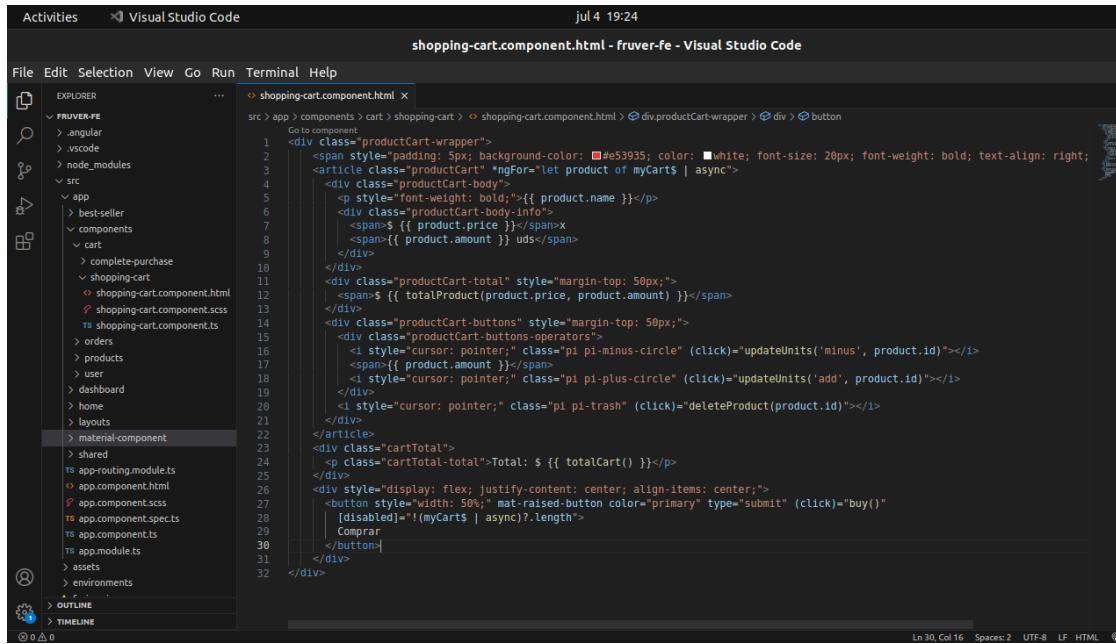
EXPLORER

FRUVER-FE

- > .angular
- > .vscode
- > node\_modules
- src
  - app
    - best-seller
    - components
      - cart
      - orders
        - order-detail
          - order-detail.component.html
          - order-detail.component.scss
          - order-detail.component.ts
    - view-order
    - products
    - user
    - dashboard
    - home
    - layouts
    - material-component
    - dialog
    - material.module.ts
    - material.routing.ts
    - shared
    - app-routing.module.ts
      - app.component.html
      - app.component.scss
      - app.component.spec.ts
      - app.component.ts
  - OUTLINE
  - TIMELINE

**cart**: en dicha carpeta se encuentran los siguientes componentes:

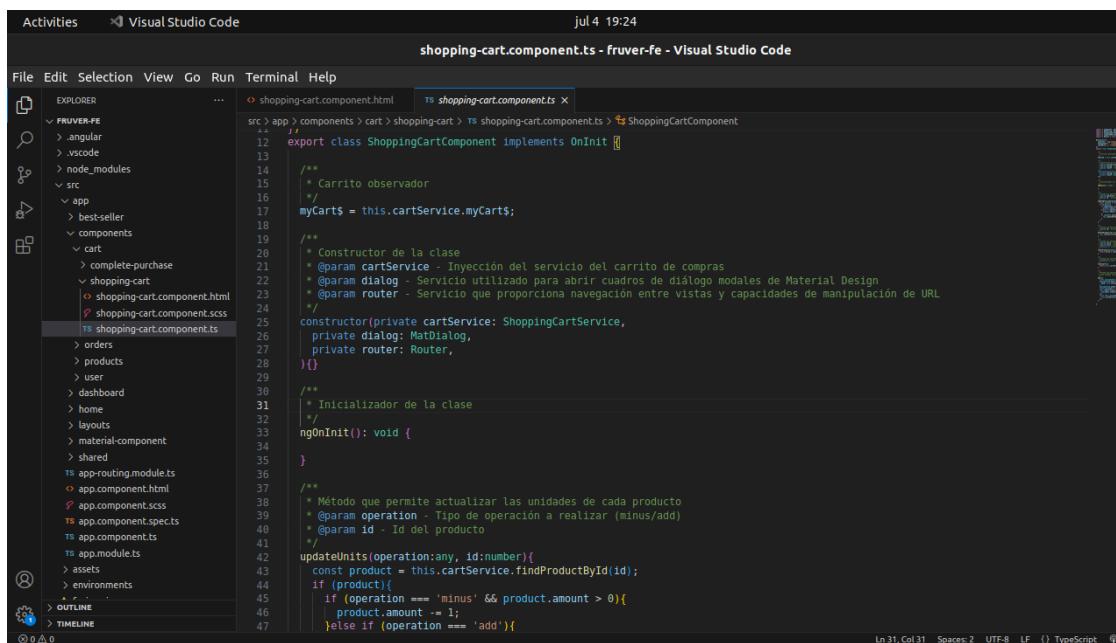
El componente **shopping-cart**, encargado de mostrar los productos que se van agregando al carrito de compras, donde se puede adicionar o restar cantidades de un producto en el carrito, obteniendo el subtotal de cada registro, y el total respectivo del carrito actual. La siguiente imagen muestra el archivo html del componente.



The screenshot shows the Visual Studio Code interface with the file `shopping-cart.component.html` open. The code is an Angular template for a shopping cart component. It includes a main container with padding, a product list, a total summary, and a button section. The code is well-documented with comments explaining the logic for updating units and deleting products.

```
<div class="productCart-wrapper">
  <div style="padding: 5px; background-color: #e53935; color: white; font-size: 20px; font-weight: bold; text-align: right; border-radius: 50%; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin-bottom: 10px; position: relative; z-index: 1;>
    <span>${ myCart.length }</span>
  </div>
  <article class="productCart">
    <p style="font-weight: bold;">${ product.name }</p>
    <div class="productCart-body">
      <span>${ product.price }</span>
      <span>${ product.amount }</span>
    </div>
    <div class="productCart-total" style="margin-top: 50px;">
      <span>${ totalProduct(product.price, product.amount) }</span>
    </div>
    <div class="productCart-buttons" style="margin-top: 50px;">
      <div class="productCart-buttons/operators">
        <i style="cursor: pointer; class="pi pi-minus-circle" (click)="updateUnits('minus', product.id)"></i>
        <span>${ product.amount }</span>
        <i style="cursor: pointer; class="pi pi-plus-circle" (click)="updateUnits('add', product.id)"></i>
      </div>
      <i style="cursor: pointer; class="pi pi-trash" (click)="deleteProduct(product.id)"></i>
    </div>
  </article>
  <div class="cartTotal">
    <p class="cartTotal-total">Total: ${ totalCart() }</p>
  </div>
  <div style="display: flex; justify-content: center; align-items: center;">
    <button style="width: 50%; mat-raised-button color="primary" type="submit" (click)="buy()" [disabled]="${ !myCart.length }">
      Comprar
    </button>
  </div>
</div>
```

Y su respectivo archivo `.ts`, el cuál se encuentra correctamente documentado.



The screenshot shows the Visual Studio Code interface with the file `shopping-cart.component.ts` open. The code defines a class `ShoppingCartComponent` with methods for initializing the cart, updating unit counts, and deleting products. The code uses TypeScript annotations like `OnInit` and `OnDestroy` and includes detailed JSDoc-style comments for each method.

```
export class ShoppingCartComponent implements OnInit {
  /**
   * Carrito observador
   */
  myCart$ = this.cartService.myCart$;

  /**
   * Constructor de la clase
   * @param cartService - Inyección del servicio del carrito de compras
   * @param dialog - Servicio utilizado para abrir cuadros de diálogo modales de Material Design
   * @param router - Servicio que proporciona navegación entre vistas y capacidades de manipulación de URL
   */
  constructor(private cartService: ShoppingCartService,
    private dialog: MatDialog,
    private router: Router,
  ){}

  /**
   * Inicializador de la clase
   */
  ngOnInit(): void {
  }

  /**
   * Método que permite actualizar las unidades de cada producto
   * @param operation - Tipo de operación a realizar (minus/add)
   * @param id - Id del producto
   */
  updateUnits(operation:any, id:number){
    const product = this.cartService.findProductById(id);
    if (product){
      if (operation === 'minus' && product.amount > 0){
        product.amount -= 1;
      }else if (operation === 'add'){
        product.amount += 1;
      }
    }
  }
}
```

El componente **complete-purchase**, encargado de mostrar un formulario para que el cliente complete el proceso de compra ingresando sus respectivos datos, utilizando un **FormGroup** para gestionar los controles del formulario, como se muestra en la siguiente imagen.

```

<mat-toolbar color="primary">
  <mat-toolbar-row fxLayout="row">
    <span class="title-center">COMPLETAR PROCESO DE COMPRA</span>
  </mat-toolbar-row>
</mat-toolbar>

<mat-dialog-content class="mat-typography">
  <form [formGroup]=>productForm>
    <div fxFlex fxLayout="column">
      <div fxFlex="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Nombre</mat-label>
          <input matInput formControlName="name" required>
          <mat-error *ngIf=>productForm.controls.name.touched && productForm.controls.name.invalid>
            <span *ngIf=>productForm.controls.name.errors.required>Este campo es obligatorio</span>
            <span *ngIf=>productForm.controls.name.errors.pattern>El campo es inválido</span>
          </mat-error>
        </mat-form-field>
      </div>
      <div fxFlex="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Número de identificación</mat-label>
          <input matInput formControlName="identificationNumber" required>
          <mat-error *ngIf=>productForm.controls.identificationNumber.touched && productForm.controls.identificationNumber.invalid>
            <span *ngIf=>productForm.controls.identificationNumber.errors.required>Este campo es obligatorio</span>
            <span *ngIf=>productForm.controls.identificationNumber.errors.pattern>El campo es inválido</span>
          </mat-error>
        </mat-form-field>
      </div>
      <div fxFlex="row wrap">
        <mat-form-field appearance="fill" fxFlex>
          <mat-label>Email</mat-label>
          <input matInput formControlName="email" required>
          <mat-error *ngIf=>productForm.controls.email.touched && productForm.controls.email.invalid>
            <span *ngIf=>productForm.controls.email.errors.required>Este campo es obligatorio</span>
            <span *ngIf=>productForm.controls.email.errors.pattern>El campo es inválido</span>
          </mat-error>
        </mat-form-field>
      </div>
    </div>
  </form>
</mat-dialog-content>

```

Y su respectivo archivo **.ts**, el cuál se encuentra correctamente documentado.

```

export class CompletePurchaseComponent implements OnInit {
  /**
   * Variable que permite emitir un evento, cuando la solicitud de compra es exitosa
   */
  onbuy = new EventEmitter();

  /**
   * Mensaje de respuesta
   */
  responseMessage:any;

  /**
   * Variable que representa un grupo de controles de formulario
   */
  productForm: any = FormGroup;

  /**
   * Productos que se almacenaron en el carrito de compras
   */
  private products: any[] = [];

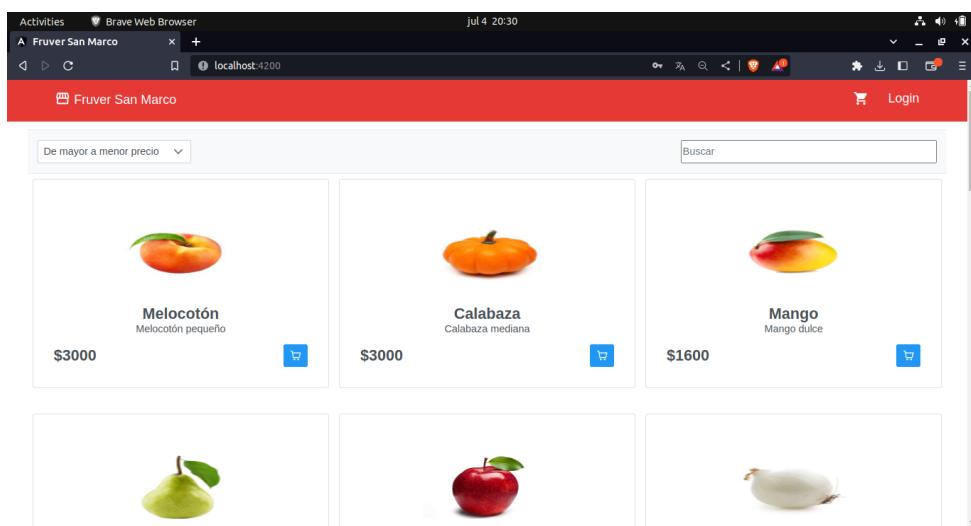
  /**
   * Constructor de la clase
   * @param dialogData - Servicio para inyectar datos en un cuadro de diálogo (dialog) que se abre
   * utilizando el MatDialog de Angular Material
   * @param formBuilder - Clase que proporciona métodos para generar instancias de formularios y controles reactivos
   * @param orderService - Inyección del servicio de pedidos
   * @param dialogRef - Servicio que permite controlar y manipular los cuadros de diálogo modales
   * @param snackBarService - Servicio para mostrar mensajes
   */
  constructor(@Inject(MAT_DIALOG_DATA) public dialogData: any,
    private formBuilder: FormBuilder,
    private orderService: OrderService,
    public dialogRef: MatDialogRef<CompletePurchaseComponent>,
    private snackBarService: MatSnackBar) {
  }
}

```

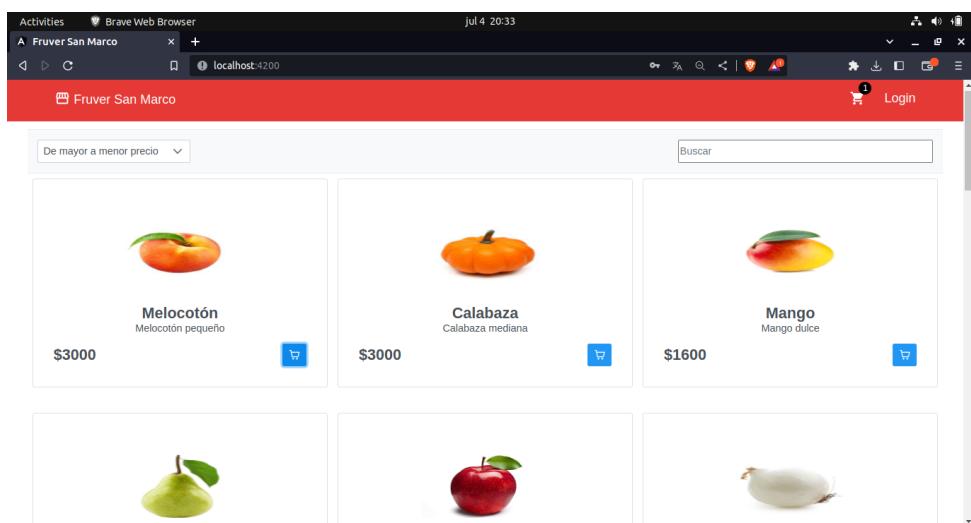
## Prueba del Aplicativo

Una vez finalizado el aplicativo, se realizaron las respectivas pruebas de funcionamiento. Para descargar las dependencias del proyecto se utiliza el comando “npm instal”l, y con el comando “ng serve” se inicia el servidor de desarrollo en angular.

Al ingresar a la página principal del aplicativo, se muestran todos los productos disponibles para que el usuario los pueda ir añadiendo al carrito de compras.



Al dar click en el botón (cart) azul de cualquier producto, se añade dicho producto al carrito de compras, y en la parte superior derecha, específicamente sobre el ícono del carrito de compras, se visualiza un contador de productos, como se muestra en la siguiente imagen.



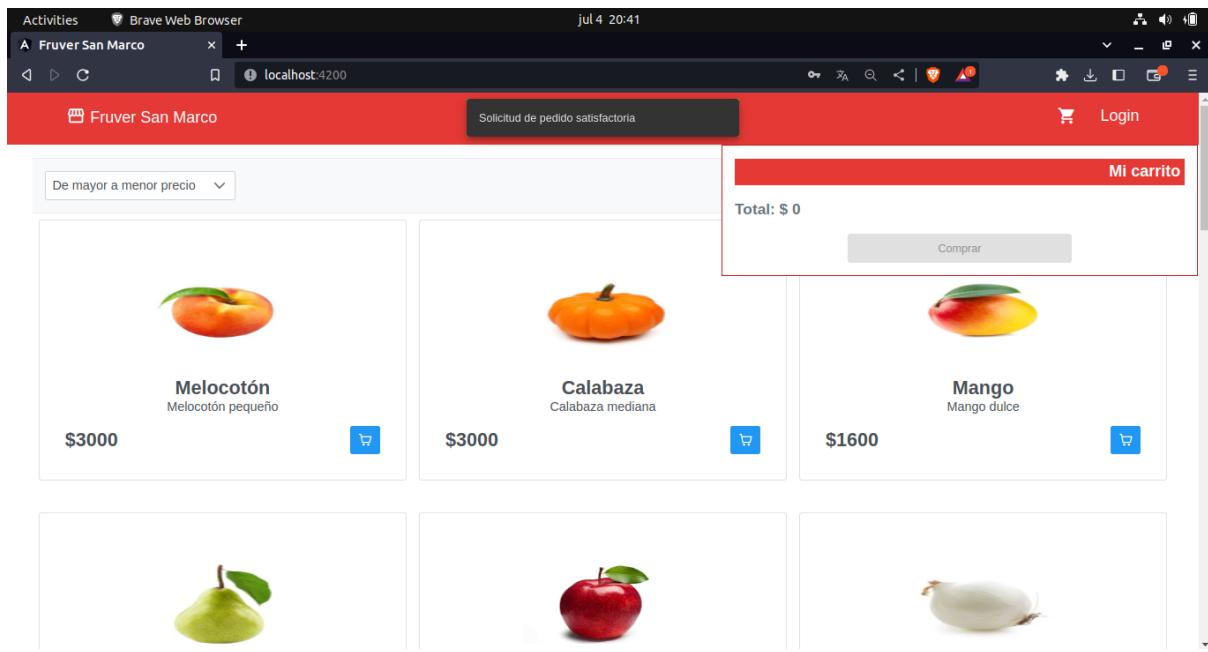
Una vez agregados los productos al carrito de compras, se da click sobre el ícono del carrito que se encuentra en la parte superior derecha, y se puede observar los productos añadidos al carrito, donde se puede modificar la cantidad de cada producto a comprar.

The screenshot shows a web browser window for 'Brave Web Browser' displaying the 'Fruver San Marco' website. The URL is 'localhost:4200'. The page features a red header with the site's name. On the left, there are two product cards: 'Melocotón' (peach) and 'Calabaza' (pumpkin), both priced at '\$3000'. On the right, a 'Mi carrito' (My Cart) sidebar is open, showing three items: 'Melocotón' (\$3000 x 1), 'Calabaza' (\$9000 x 3), and 'Pera' (\$3000 x 2). The total is 'Total: \$ 15000'. Below the sidebar is a red 'Comprar' (Buy) button. The background shows other fruit products like a pear and an apple.

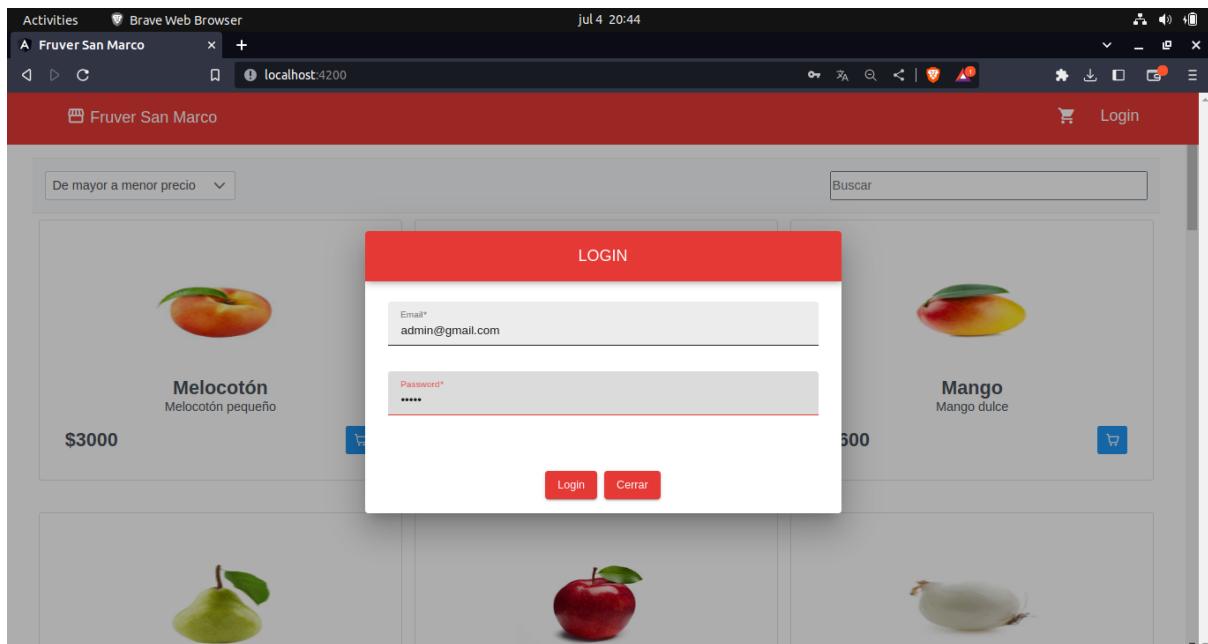
Una vez seguro de querer realizar la compra de dichos productos, se da click en el botón comprar, y se abre un modal para que el cliente pueda completar el proceso de compra, ingresando sus respectivos datos.

The screenshot shows the same web browser window with the purchase completion modal open. The modal title is 'COMPLETAR PROCESO DE COMPRA'. It contains fields for 'Nombre\*' (Name\*) with 'Jesús Ibarra Revelo' entered, 'Número de identificación\*' (Identification number\*) with '1145654565' entered, 'Email\*' (Email\*) with 'drevolo@gmail.com' entered, and 'Número de celular\*' (Cell phone number\*) with '3454565676' entered. Below these fields is a 'Dirección de envío\*' (Shipping address\*) field containing 'Calle 18D #8A E -57 Lorenzo'. At the bottom of the modal are two buttons: 'Solicitar pedido' (Request order) and 'Cerrar' (Close). The background of the main page shows the same fruit products and the shopping cart sidebar.

Una vez registrados los datos del cliente, se da click en el botón “solicitar pedido”, y el sistema muestra un mensaje diciendo que la solicitud del pedido fue satisfactoria, finalmente limpiando el carrito de compras.

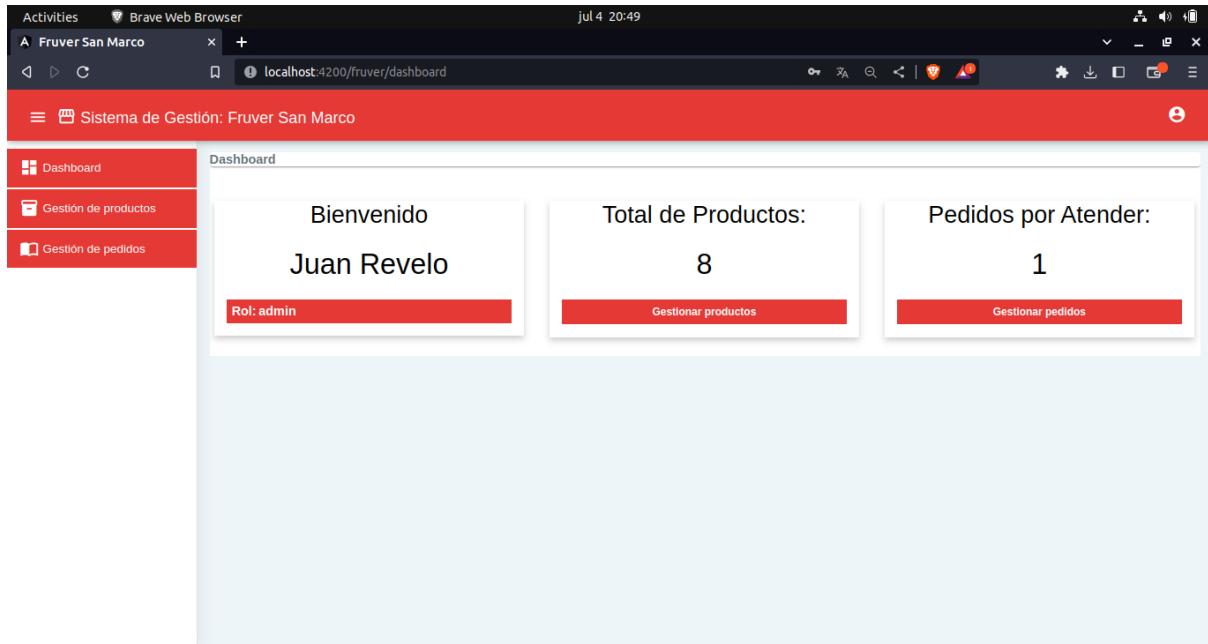


Una vez realizado lo anterior, se inicia sesión para poder atender dicha solicitud.

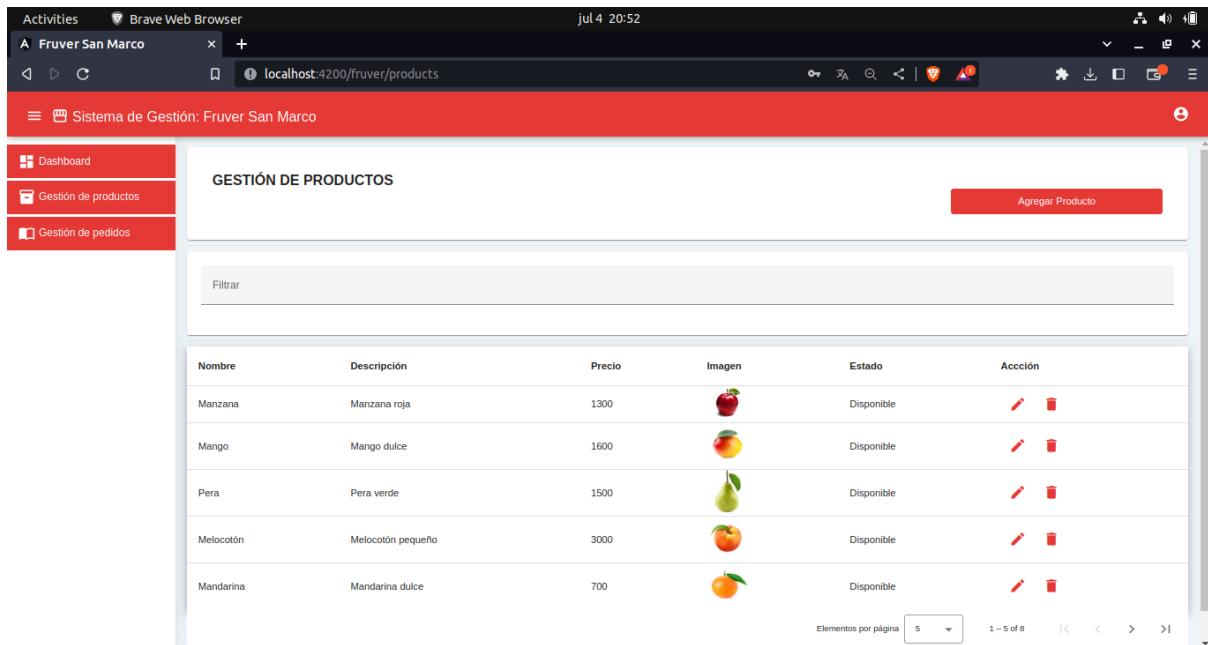


Una vez iniciado sesión, el sistema redirige al usuario administrador al dashboard de la aplicación. En dicha pantalla se visualizan tres cards, donde cada una de ellas

mostrará el nombre de usuario en sesión, el total de productos y el total de pedidos por atender respectivamente, como se muestra en la siguiente imagen.



Al ingresar a la opción de menú “Gestión de productos”, se visualiza la pantalla para administrar todos los productos registrados hasta el momento.



Cuando se da click al botón “Agregar Producto”, se muestra el respectivo formulario para agregar un producto.

Activities Brave Web Browser  
A Fruver San Marco x +  
localhost:4200/fruver/products

Sistema de Gestión: Fruver San Marco

### GESTIÓN DE PRODUCTOS

Añadir

Nombre*	Ciruela
Precio*	600
Descripción Ciruela roja	
<input type="button" value="Cargar imagen"/> <input checked="" type="checkbox"/> El producto está disponible?	
	<input type="button" value="Añadir"/> <input type="button" value="Cerrar"/>

Mandarina      Mandarina dulce      700           Disponible

Elementos por página: 5      1 - 5 of 8      < < > >

Una vez registrada la respectiva información del producto, se da click en el botón añadir, y el sistema registrará el producto, actualizando la tabla de gestión con el nuevo producto agregado.

Activities Brave Web Browser  
A Fruver San Marco x +  
localhost:4200/fruver/products

Sistema de Gestión: Fruver San Marco

Nombre	Descripción	Precio	Imagen	Estado	Acción
Manzana	Manzana roja	1300		Disponible	
Mango	Mango dulce	1600		Disponible	
Pera	Pera verde	1500		Disponible	
Melocotón	Melocotón pequeño	3000		Disponible	
Mandarina	Mandarina dulce	700		Disponible	
Cebolla	Cebolla Blanca	800		Disponible	
Calabaza	Calabaza mediana	3000		Disponible	
Durazno	Durazno amarillo	600		Disponible	
Ciruela	Ciruela roja	600		Disponible	

Elementos por página: 10      1 - 9 of 9      < < > >

También se puede editar un respectivo producto dando click al ícono de “lápiz”, el cuál abre el mismo formulario anterior, pero con la información del producto a editar.

A screenshot of a web browser window titled "Sistema de Gestión: Fruver San Marco". The URL is "localhost:4200/fruver/products". The left sidebar has options for "Dashboard", "Gestión de productos", and "Gestión de pedidos". The main content area shows a table of products with columns: Nombre, Descripción, Precio, Imagen, Estado, and Acción. One row is highlighted with a red background, indicating it is being edited. A modal window titled "Editar" is open over the table, containing fields for "Nombre\*" (Ciruela Mediana), "Precio\*" (600), and "Descripción" (Ciruela roja). It also has a "Cargar imagen" button, a checked checkbox for "El producto está disponible?", and two buttons at the bottom: "Actualizar" and "Cerrar".

Nombre	Descripción	Precio	Imagen	Estado	Acción
Manzana	Manzana roja	1300		Disponible	
Mango					
Pera	Ciruela Mediana				
Melocotón					
Mandarina					
Cebolla					
Calabaza					
Durazno					
Ciruela	Ciruela roja	600		Disponible	

Una vez registrada la respectiva información del producto, se da click en el botón actualizar, y el sistema actualizará el respectivo producto, mostrando dicha actualización en la tabla.

A screenshot of a web browser window titled "Sistema de Gestión: Fruver San Marco". The URL is "localhost:4200/fruver/products". The left sidebar has options for "Dashboard", "Gestión de productos", and "Gestión de pedidos". The main content area shows a table of products with columns: Nombre, Descripción, Precio, Imagen, Estado, and Acción. The table now includes a success message: "Producto actualizado satisfactoriamente". The product data has been updated: Manzana (Manzana roja, 1300, apple icon, Disponible), Mango (Mango dulce, 1600, mango icon, Disponible), Pera (Pera verde, 1500, pear icon, Disponible), Melocotón (Melocotón pequeño, 3000, peach icon, Disponible), Mandarina (Mandarina dulce, 700, orange icon, Disponible), Cebolla (Cebolla Blanca, 800, onion icon, Disponible), Calabaza (Calabaza mediana, 3000, pumpkin icon, Disponible), Durazno (Durazno amarillo, 600, peach icon, Disponible), and Ciruela Mediana (Ciruela roja, 600, plum icon, Disponible).

Nombre	Descripción	Precio	Imagen	Estado	Acción
Manzana	Manzana roja	1300		Disponible	
Mango	Mango dulce	1600		Disponible	
Pera	Pera verde	1500		Disponible	
Melocotón	Melocotón pequeño	3000		Disponible	
Mandarina	Mandarina dulce	700		Disponible	
Cebolla	Cebolla Blanca	800		Disponible	
Calabaza	Calabaza mediana	3000		Disponible	
Durazno	Durazno amarillo	600		Disponible	
Ciruela Mediana	Ciruela roja	600		Disponible	

Finalmente, se puede eliminar un producto, dando click sobre el ícono de “trash” que se muestra en cada registro.

Activities Brave Web Browser  
A Fruver San Marco x +  
localhost:4200/fruver/products  
Sistema de Gestión: Fruver San Marco

Nombre	Descripción	Precio	Imagen	Estado	Acción
Manzana	Manzana roja	1300		Disponible	
Mango	Mango dulce	1600		Disponible	
Pera	Pera verde	1500		Disponible	
Melocotón	Melocotón pequeño			Disponible	
Mandarina	Mandarina dulce			Disponible	
Cebolla	Cebolla Blanca	800		Disponible	
Calabaza	Calabaza mediana	3000		Disponible	
Durazno	Durazno amarillo	600		Disponible	
Ciruela Mediana	Ciruela roja	600		Disponible	

Elementos por página: 10 | 1 - 9 of 9 | < < > >|

Aparece un diálogo de confirmación, donde se da click en la opción “Si” para eliminar dicho producto. Una vez eliminado dicho producto, la tabla se actualiza correctamente.

Activities Brave Web Browser  
A Fruver San Marco x +  
localhost:4200/fruver/products  
Sistema de Gestión: Fruver San Marco

Nombre	Descripción	Precio	Imagen	Estado	Acción
Manzana	Manzana roja	1300		Disponible	
Mango	Mango dulce	1600		Disponible	
Pera	Pera verde	1500		Disponible	
Melocotón	Melocotón pequeño	3000		Disponible	
Mandarina	Mandarina dulce	700		Disponible	
Cebolla	Cebolla Blanca	800		Disponible	
Calabaza	Calabaza mediana	3000		Disponible	
Durazno	Durazno amarillo	600		Disponible	

Elementos por página: 10 | 1 - 8 of 8 | < < > >|

Por otro lado, para atender la solicitud de compra realizada anteriormente, se dirige a la opción de menú “Gestión de pedidos”. En dicha pantalla se muestran todos los pedidos que aún no han sido atendidos, es decir, los pedidos en estado “pendiente”.

The screenshot shows a web browser window titled "Brave Web Browser" with the URL "localhost:4200/fruver/orders". The title bar indicates the date and time as "jul 4 21:06". The main content area has a red header bar with the text "Sistema de Gestión: Fruver San Marco". On the left, there is a sidebar with three items: "Dashboard" (selected), "Gestión de productos", and "Gestión de pedidos". The main content area is titled "PEDIDOS POR ATENDER" and contains a table with one row of data. The columns are "Cliente", "Cédula", "Email", "Dirección de envío", "Total", and "Acciones". The data row shows "Jesus Ibarra Revelo", "1145654565", "drevelo@gmail.com", "Calle 18D #8A E -57 Lorenzo", "\$15000", and two icons: a magnifying glass and a checkmark. Below the table are buttons for "Elementos por página" (set to 5) and navigation arrows. The footer of the page shows "1 - 1 of 1".

Cada registro cuenta con dos acciones, una para poder ver el detalle de cada solicitud de pedido (ícono del ojo), y otra para consolidar y enviar el respectivo pedido (ícono del check).

Cuando se da click en el ícono del “ojito”, se abre un modal donde se puede visualizar todo el detalle de ese pedido, es decir, los datos del cliente, los productos asociados a ese pedido, y su respectivo valor total.

The screenshot shows a modal window titled "DETALLE DEL PEDIDO" over a blurred background of the previous screen. The modal has a red header bar with the text "PEDIDOS POR ATENDER". Inside the modal, there are two sections: "Cliente: Jesus Ibarra Revelo" and "Cédula: 1145654565" on the top left, and "Teléfono: 3454565676" and "Email: drevelo@gmail.com" on the top right. Below these are tables for "Productos" and "Detalle del Pedido". The "Productos" table lists "Melocotón", "Calabaza", and "Pera" with their respective prices (\$3000, \$3000, \$1500) and quantities (1, 3, 2). The "Detalle del Pedido" table shows the same items with their subtotals (\$3000, \$9000, \$3000) and a final "Total: \$15000".

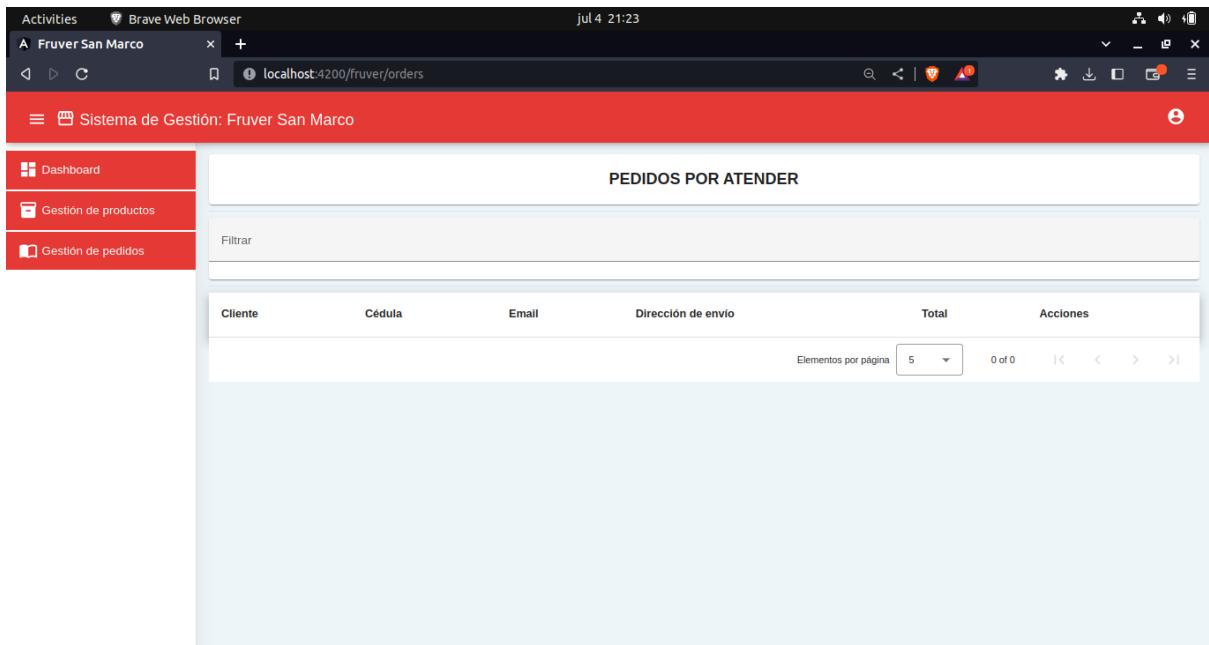
Finalmente, para consolidar y enviar un pedido se da click en el ícono del “check”, donde aparece un diálogo de confirmación, al cuál se le da click en la opción “Si”.

The screenshot shows a web browser window titled "Brave Web Browser" with the URL "localhost:4200/fruver/orders". The main content area displays a table titled "PEDIDOS POR ATENDER" (Pending Orders). A success message at the top right states: "Pedido consolidado y enviado exitosamente: Se notificó al cliente el envío de su pedido" (Order consolidated and sent successfully: The client was notified of the order's delivery). The table has columns: Cliente (Client), Cédula (Cedula), Email, Dirección de envío (Shipping Address), Total, and Acciones (Actions). The table body is currently empty, showing "Cargando..." (Loading...). On the left, a sidebar menu includes "Dashboard", "Gestión de productos", and "Gestión de pedidos".

Cuando dicha acción se resuelve con éxito, el sistema muestra un mensaje donde dice que el pedido fue consolidado y enviado exitosamente, notificando al cliente el envío de su pedido mediante un correo electrónico. Ingresando al correo del cliente, se puede observar que se envió el correo exitosamente por parte de la aplicación.

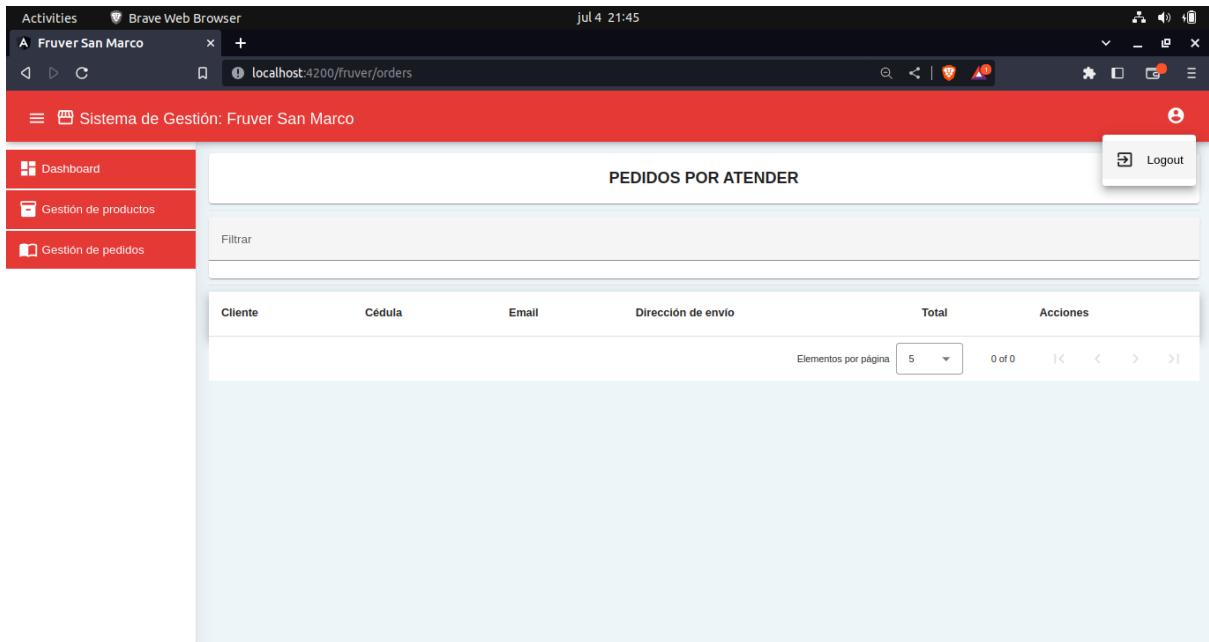
The screenshot shows a Gmail inbox with one unread email titled "Pedido Envíado - reveloju" from "fruversanmarco@gmail.com". The email details the order: Client: Jesús Ibarra Revelo, Identification Number: 1145654565, Cell Phone Number: 3454565676, Delivery Address: Calle 18D #8A E - 57 Lorenzo, Date: 2023-07-04. The total purchase amount is \$15000. The email ends with a thank you message: "Muchas Gracias por confiar en FRUVER SAN MARCO". At the bottom of the email view, there are "Reply" and "Forward" buttons.

Finalmente, el sistema actualiza el estado de la solicitud a “atendida”, y por tal razón, dicha solicitud ya no se visualiza en la tabla.



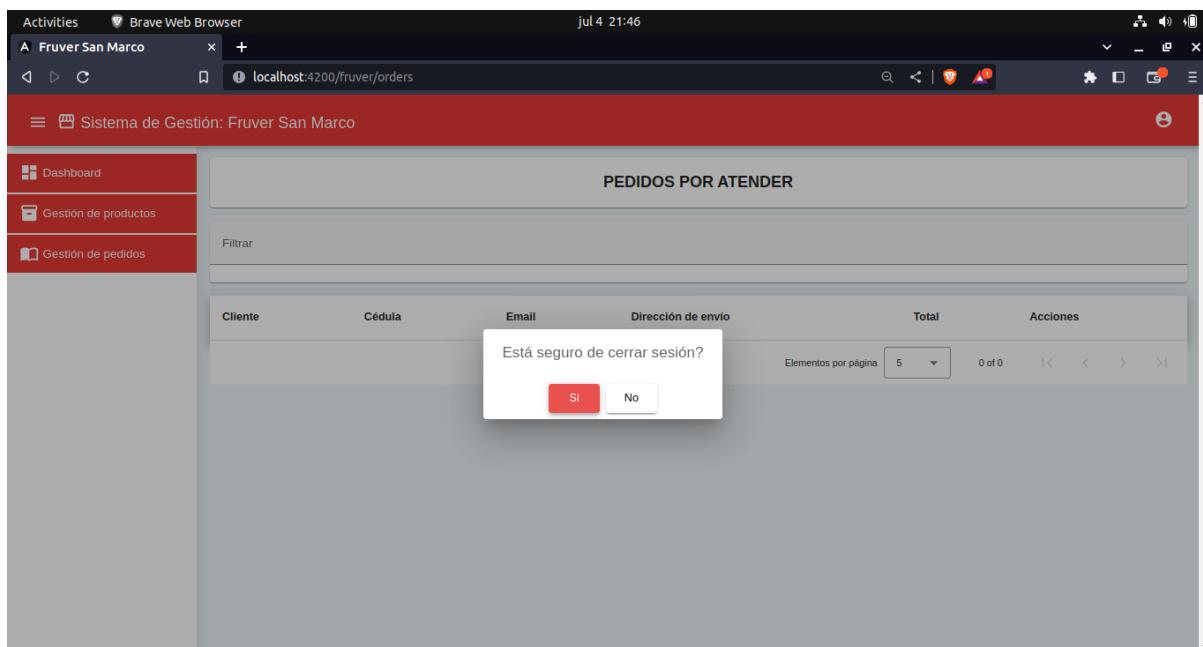
The screenshot shows a web browser window titled "Fruver San Marco" with the URL "localhost:4200/fruver/orders". The page has a red header bar with the title "Sistema de Gestión: Fruver San Marco". On the left, there is a sidebar with three items: "Dashboard", "Gestión de productos", and "Gestión de pedidos". The main content area is titled "PEDIDOS POR ATENDER" and contains a table with columns: Cliente, Cédula, Email, Dirección de envío, Total, and Acciones. A search bar labeled "Filtrar" is above the table. At the bottom right of the table, there are pagination controls: "Elementos por página" (set to 5), "0 of 0", and navigation arrows. The browser status bar at the top right shows the date "jul 4 21:23".

Por otro lado, para cerrar sesión se da clic en el icono “user“, donde aparece un menú item con la opción de cerrar sesión, como se muestra en la siguiente imagen.

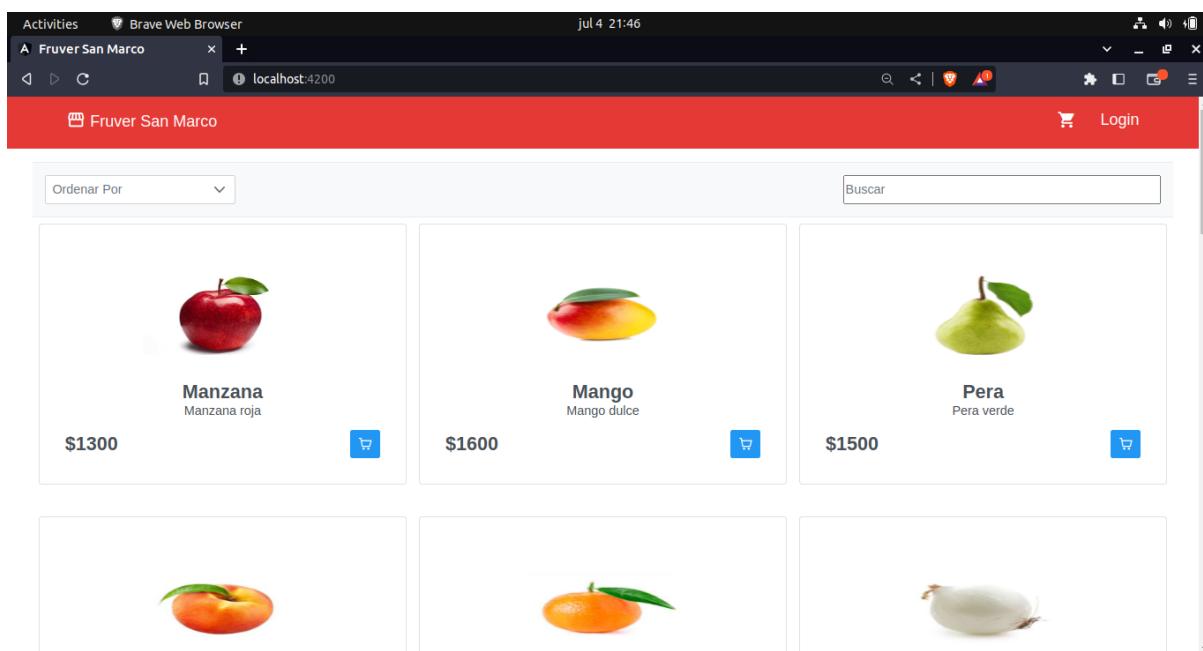


The screenshot shows the same web browser window as the previous one, but now the "user" icon in the top right corner of the header has a dropdown menu. The menu item "Logout" is highlighted with a red box. The rest of the interface is identical to the previous screenshot, showing an empty "PEDIDOS POR ATENDER" table.

Al dar click en la opción “Logout”, se muestra una ventana de confirmación para poder cerrar sesión.



Se da click en la opción “Sí”, y el sistema elimina la sesión del usuario y lo redirige a la vista principal del aplicativo.



La base de datos del proyecto también se subirá al repositorio de git con datos de ejemplo. El correo del usuario con el que se puede iniciar sesión es [admin@gmail.com](mailto:admin@gmail.com) y su contraseña es admin.