

Informe Frontend Fruver la 17

Presentado por:

Luis Fernando Oviedo Dominguez

Presentado a:

Mg. Euler Vicente Aux Revelo

Universidad de Nariño

2023

Con el fin de consumir la API realizada desde ExpressJS se decide realizar el Front_End en Angular donde se crearon los siguientes modelos, además cabe resaltar que se estilizó la página web en base a Bootstrap 5.2 y CSS personalizado:

CDN para trabajar con las bibliotecas de Bootstrap

```
<!-- Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kgtqWx7BEP05v2W4pG7LbmBvPOMQzVt61r17FJ0q459Wjd/ckQkNc6fr" crossorigin="anonymous">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Cabin&display=swap" rel="stylesheet">
</head>
<body>
  <app-root></app-root>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ktrA3oP18ky27zy6oqb41ZV28BFD0oUt8sqd1E6+1tDRm905E7ytd0g94LYE=0" crossorigin="anonymous"></script>
</body>
</html>
```

Producto.model.ts

En este modelo se definen los atributos del producto con su respectivo tipo con el fin de capturar la información procedente del backend

```
export class ProductoModel {
  constructor(
    public idProducto: string, public nombre: string, public detalle:string,
    public cantidad: string, public categoria: string, public precio: string,
    public imagen: string
  ) {
  }
}
```

Pedido.model.ts

En este modelo se definen los atributos del pedido para interactuar con el backend

```
export class PedidoModel {
  constructor(
    public idPedido: string, public detalle: string,
    public cantidad: string, public fecha: string,
  ) {
  }
}
```

Usuario.model.ts

En este modelo se estructura los atributos definidos en el backend con el fin de interactuar con esos recursos

```
export class UsuarioModel {  
  constructor(  
    public idUsuario: string, public nombre: string, public direccion: string,  
    public correo: string, public contrasena: string, public tipo: string  
  ) {}  
}
```

Posteriormente se crean los servicios de cada modelo de datos con el fin de realizar las peticiones a métodos específicos definidos en el backend, estos se detallan a continuación:

producto.service.ts

Se crean las mismas rutas definidas en el manejador de rutas del backend para los productos así como el llamado de su respectiva función

```
obtenerProductos() {  
  return this.http.get<ProductoModel[]>(`${this.BASE_URL}/productos`);  
}  
  
obtenerProducto(idProducto: string) {  
  return this.http.get<ProductoModel[]>(`${this.BASE_URL}/productos/${idProducto}`);  
}  
  
obtenerProductoBuscar(nombre: string) {  
  return this.http.get<ProductoModel[]>(`${this.BASE_URL}/productos/busqueda/${nombre}`);  
}  
  
agregarProducto(producto: ProductoModel) {  
  return this.http.post<string>(`${this.BASE_URL}/productos`, producto);  
}  
  
actualizarProducto(producto: ProductoModel) {  
  return this.http.put<string>(`${this.BASE_URL}/productos/${producto.idProducto}`, producto);  
}  
  
borrarProducto(idProducto: string) {  
  return this.http.delete<string>(`${this.BASE_URL}/productos/${idProducto}`);  
}
```

pedido.service.ts

Se generan las rutas idénticas que se encuentran definidas en el controlador de rutas del servidor para los pedidos, junto con la invocación de su función correspondiente.

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

  obtenerPedidos() {
    return this.http.get<PedidoModel[]>(`${this.BASE_URL}/pedidos`);
  }
  obtenerPedido(idPedido: string) {
    return this.http.get<PedidoModel[]>(`${this.BASE_URL}/pedidos/${idPedido}`);
  }

  agregarPedido(pedido: PedidoModel) {
    return this.http.post<string>(`${this.BASE_URL}/pedidos`, pedido);
  }
  actualizarPedido(pedido: PedidoModel) {
    return this.http.put<string>(`${this.BASE_URL}/pedidos/${pedido.idPedido}`, pedido);
  }
  borrarPedido(idPedido: string) {
    return this.http.delete<string>(`${this.BASE_URL}/pedidos/${idPedido}`);
  }
}
```

usuario.service.ts

Se generan las rutas exactas que se han definido previamente en el manejador de rutas del servidor para los usuarios, junto con la invocación de su función correspondiente y funciones adicionales para el cierre de sesión del usuario.

```
export class UsuarioService {

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

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

  obtenerUsuario(idUsuario: string) {
    return this.http.get<UsuarioModel[]>(`${this.BASE_URL}/registro/usuarios/${idUsuario}`);
  }

  agregarUsuario(usuario: UsuarioModel) {
    return this.http.post<string>(`${this.BASE_URL}/registro/usuarios`, usuario);
  }
  autenticarUsuario(correo: string, contrasena: string) {
    const url = `${this.BASE_URL}/acceder`;
    return this.http.post<string>(url, { correo, contrasena });
  }

  logout() {
    localStorage.removeItem('currentUser');
  }
}
```

filtro-productos.service.ts

Se crea este servicio con el fin de compartir datos con otros componentes y con esto reducir la redundancia de código para el filtrado de productos

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class FiltroProductosService {
  categoriaSeleccionada: BehaviorSubject<string | null> = new BehaviorSubject<string | null>(null);

  constructor() { }
}
```

Paso a seguir se crean los componentes necesarios para ejecutar cada una de las operaciones para la frutería, luego en **app-routing.module.ts** se definen cada una de las rutas de los componentes creados como se muestra a continuación:

```
const routes: Routes = [
  {path:'productos',component: ListadoProductosComponent},
  { path: 'productos/editar/:idProducto', component: EditarProductosComponent},
  { path: 'productos/agregar', component: EditarProductosComponent },
  { path: 'productos/detalles/:idProducto', component: DetalleProductosComponent },
  { path: 'registro/usuarios', component: RegistroComponent},
  { path: 'acceder', component: LoginComponent},
  { path: 'acceder/administracion', component: AdministracionComponent},
  { path: 'acceder/administracion/productos', component: AdmProductosComponent},
  { path: 'acceder/administracion/pedidos', component: AdmPedidosComponent},
  { path: 'pedidos/editar/:idPedido', component: EditarPedidosComponent},
  { path: 'pedidos/agregar', component: DetalleProductosComponent },
  {path:'**',redirectTo:'/productos',pathMatch:'full'}
];
```

A continuación, se detalla el funcionamiento de cada componente creado en el proyecto Angular

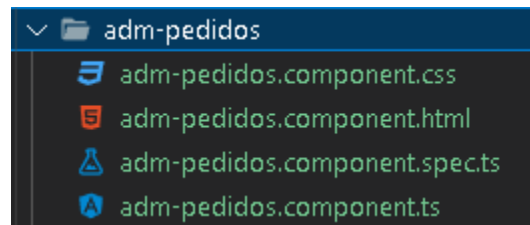
Cabe resaltar que cada componente creado debe ser declarado en **app.module.ts**

```
@NgModule({  
  declarations: [  
    AppComponent,  
    EncabezadoComponent,  
    FooterComponent,  
    ListadoProductosComponent,  
    FiltroComponent,  
    LoginComponent,  
    RegistroComponent,  
    EditarProductosComponent,  
    DetalleProductosComponent,  
    AdministracionComponent,  
    AdmProductosComponent,  
    AdmPedidosComponent,  
    EditarPedidosComponent  
  ],  
  imports: [  
    BrowserModule,  
    AppRoutingModule,  
    HttpClientModule,  
    FormsModule  
  ],  
  providers: [  
    ProductoService,  
    UsuarioService,  
    PedidoService,  
    FiltroProductosService  
  ],  
  bootstrap: [AppComponent]  
})
```

Además, se importan los servicios y bibliotecas que se disponen a utilizar en el proyecto

```
imports: [  
  BrowserModule,  
  AppRoutingModule,  
  HttpClientModule,  
  FormsModule  
],  
providers: [  
  ProductoService,  
  UsuarioService,  
  PedidoService,  
  FiltroProductosService  
],  
bootstrap: [AppComponent]
```

adm-pedidos



Permite visualizar todos los pedidos registrados en una tabla, asimismo permite acceder a otras funcionalidades como la edición y eliminación de pedidos

```
<tbody>
  <tr *ngFor="let pedido of pedidos | async">
    <td>{{ pedido.idPedido }}</td>
    <td>{{ pedido.detalle }}</td>
    <td>{{ pedido.cantidad }}</td>
    <td>{{ pedido.fecha }}</td>
    <td>
      <fieldset class="form-group col-sm-1">
        <a class="btn btn-info" [routerLink]="['/pedidos/editar/', pedido.idPedido]">Editar</a>
      </fieldset>
    </td>
    <td>
      <fieldset class="form-group col-sm-1">
        <a class="btn btn-danger" (click)="borrarPedido(pedido.idPedido)">Borrar</a>
      </fieldset>
    </td>
  </tr>
</tbody>
```

Además, se hizo el llamado al método **obtenerPedidos** para hacer el listado y a la función **borrarPedido** para eliminar el pedido seleccionado

```
pedidos: Observable<PedidoModel[]> | undefined;

constructor(private pedidoService: PedidoService, private location: Location) { }

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

atras() {
  this.location.back();
}
}
```

Vista

Pedidos					
ID	Detalle	Cantidad	Fecha	Editar	Eliminar
12	baya de la enredadera	2	2023-07-03T23:45:50.000Z	Editar	Borrar
13	fruta pomácea y sabor muy dulce	3	2023-07-03T23:45:42.000Z	Editar	Borrar
14	cereza fresca de color rojo	1	2023-07-03T23:49:59.000Z	Editar	Borrar
20	baya de la enredadera	1	2023-07-04T15:31:07.000Z	Editar	Borrar
21	piña tropical	1	2023-07-04T15:31:12.000Z	Editar	Borrar

Atrás

adm-productos

Igualmente, proporciona la capacidad de ver una lista completa de todos los productos registrados en un formato tabular. Además, ofrece la opción de acceder a otras características, como la posibilidad de editar o eliminar los productos.

```
<tr *ngFor="let producto of productos | async">
  <td>{{ producto.idProducto }}</td>
  <td>{{ producto.nombre }}</td>
  <td>{{ producto.detalle }}</td>
  <td>{{ producto.cantidad }}</td>
  <td>{{ producto.categoria }}</td>
  <td>{{ producto.precio }}$</td>
  <td>
    <fieldset class="form-group col-sm-1">
      <a class="btn btn-info" [routerLink]="['/productos/editar/', producto.idProducto]">Editar</a>
    </fieldset>
  </td>
  <td>
    <fieldset class="form-group col-sm-1">
      <a class="btn btn-danger" (click)="borrarProducto(producto.idProducto)">Borrar</a>
    </fieldset>
  </td>
</tr>
```

Adicionalmente, se efectuó la invocación al método obtenerProductos con el fin de generar la lista, así como se utilizó la función borrarProducto para eliminar el pedido que fue seleccionado.


```

export class AdmProductosComponent implements OnInit{
  productos: Observable<ProductoModel[]> | undefined;

  constructor(private productoService: ProductoService,private location: Location) { }

  ngOnInit(){
    this.productos = this.productoService.obtenerProductos();
  }

  borrarProducto(idProducto: string) {
    this.productoService.borrarProducto(idProducto).subscribe(data => {
      console.log("Registro Eliminado");
      this.ngOnInit();
    });
  }
  atras() {
    this.location.back();
  }
}

```

Vista

<div>Insertar producto</div> <h3>Productos</h3>							
ID	Nombre	Detalle	Cantidad	Categoría	Precio	Editar	Eliminar
1	uva	Uva cubana	39	temporada	5000\$	Editar	Borrar
2	mango	Mango "Golden Sunrise"	20	pepita	5000\$	Editar	Borrar
3	kiwi	baya de la enredadera	20	pepita	2000\$	Editar	Borrar
4	piña	piña tropical	13	tropical	4600\$	Editar	Borrar
6	cereza	cereza fresca de color rojo	17	baya	6500\$	Editar	Borrar
7	coco	Fruto seco tropical	58	tropical	5500\$	Editar	Borrar
8	fresa	Fruto de color rojo brillante y succulento	150	baya	3000\$	Editar	Borrar
9	manzana	fruta pomácea y sabor muy dulce	43	pepita	2500\$	Editar	Borrar

administración

Este componente es un panel de control el cual permite acceder a las operaciones con los productos como para los pedidos

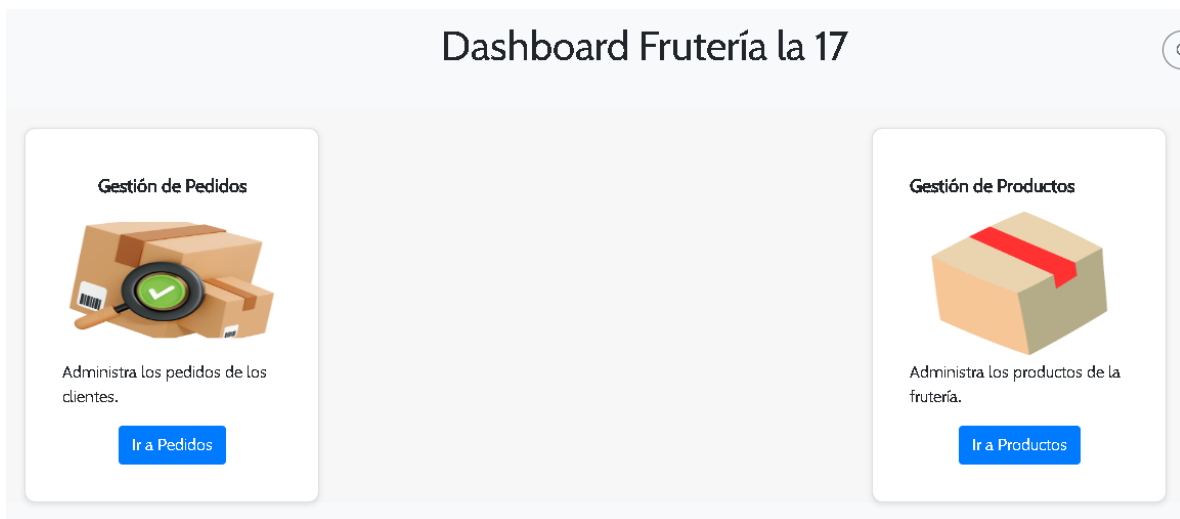
```
<div class="dashboard">
  <div class="card" style="width: 18rem;">
    <div class="card-body">
      <h5 class="card-title text-center">Gestión de Pedidos</h5>
      <img [src]="Apedido" class="card-img-top" width="200" height="150" alt="..." />
      <p class="card-text">Administra los pedidos de los clientes.</p>
      <div class="text-center">
        <a href="#" class="btn btn-primary" [routerLink]="['/acceder/administracion/pedidos']">Ir a Pedidos
      </div>
    </div>
  </div>
</div>
```

Accede a las operaciones de los pedidos

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Gestión de Productos</h5>
    <img [src]="Aproducto" class="card-img-top" width="200" height="150" alt="..." />
    <p class="card-text">Administra los productos de la frutería.</p>
    <div class="text-center">
      <a href="#" class="btn btn-primary" [routerLink]="['/acceder/administracion/productos']">Ir a
    </div>
  </div>
</div>
```

Accede a las operaciones de los productos

Vista



detalle-productos

Este componente contiene la información de un producto seleccionado desde la pagina inicial y es donde se realiza el pedido del producto especificando la cantidad a comprar

```
<div class="product-image">
  <img [src]="producto.imagen" alt="Imagen del producto">
</div>
<div class="product-description">
  <h2 class="product-title">{{producto.nombre}}</h2>
  <p class="product-price">Precio: {{producto.precio}}$</p>
  <div class="product-details">
    <p><strong>Cantidad: </strong>{{producto.cantidad}}</p>
    <div>
      <input type="number" [(ngModel)]="pedido.cantidad" name="cantidad" class="quantity-input" placeholder="Cantidad a comprar" />
    </div>
  </div>
  <p class="product-description">{{producto.detalle}}</p>
</div>
<fieldset class="form-group">
  <div class="buy-button">
    <button type="submit" class="btn btn-primary" [disabled]="!usuarioForm.valid">Comprar</button>
  </div>
</fieldset>
```

Visualizan los datos del producto seleccionado

```
idProducto = '';
cant:number = 0;
producto: ProductoModel | undefined;
pedido = new PedidoModel("", "", "", new Date().toISOString());

constructor(
  private productoService: ProductoService,
  private route: ActivatedRoute,
  private router: Router,
  private pedidoService: PedidoService,
  private location: Location
) { }

ngOnInit(){
  this.idProducto = this.route.snapshot.params['idProducto'];
  this.productoService.obtenerProducto(this.idProducto).subscribe(data => {
    this.producto = data[0],
    this.pedido.detalle = this.producto.detalle;
    this.pedido.cantidad = '1';
  }, error => {
    console.log(error);
  });
}
```

Por parte de la lógica se obtiene los datos del producto, así como se crea un objeto de tipo pedido con el fin de pasar la información del producto, además se define por defecto a la cantidad del pedido en 1

Vista



The screenshot shows a product card for 'mango'. On the left is an image of a ripe mango with two green leaves. To the right of the image, the text 'mango' is displayed in a bold font. Below it, 'Precio: 5000\$' is shown. Underneath that, 'Cantidad: 20' is displayed. A small input box contains the number '1'. Below the input box, the text 'Mango "Golden Sunrise"' is visible. To the right of the input box is a blue button labeled 'Comprar'. At the bottom center of the card is a grey button labeled 'Atrás'.

editar-pedidos

Este componente es capaz de editar un pedido existente en la base de datos, además muestra la información actual del pedido en los campos de envío

```
<fieldset class="form-group">
  <div class="mb-3 text-center">
    <label for="fecha" class="form-label">Fecha</label>
    <input
      type="date"
      class="form-control"
      id="fecha"
      name="fecha"
      required
      [(ngModel)]="pedido.fecha"
    />
  </div>
</fieldset>

<fieldset class="form-group">
  <div class="d-grid">
    <button type="submit" class="btn btn-primary" [disabled]="!pedidoForm.form.valid">Actual
  </div>
</fieldset>
```

Por parte de la lógica se crean objetos de tipo pedido para obtener los datos a partir de un idPedido existente y posteriormente actualizarlo usando el método **actualizarPedido**

```

export class EditarPedidosComponent implements OnInit {
  idPedido = '';
  pedido = new PedidoModel("", "", "", "");

  constructor(private pedidoService: PedidoService, private route: ActivatedRoute, private router: Router) { }

  ngOnInit() {
    this.idPedido = this.route.snapshot.params['idPedido'];

    this.pedidoService.obtenerPedido(this.idPedido).subscribe(data => {
      this.pedido = data[0];
    }, error => {
      console.log(error);
    });
  }
}

```

Realiza la operación de actualización y se redirige a la lista de pedidos

```

onSubmit() {
  console.log("Submit realizado");

  this.pedidoService.actualizarPedido(this.pedido).subscribe(
    data => {
      console.log(data);
      this.router.navigate(['acceder/administracion/pedidos/']);
    }
  );
}

```

Vista

Edicion de datos


Detalle

cereza fresca de color rojo

Cantidad

1

Fecha

dd/mm/aaaa 

Actualizar

editar-productos

Igualmente, como el anterior componente este permite la edición de productos, así como también la inserción de nuevos productos, es así como se define un modelo de producto y con ayuda del **idProducto** se verifica si se desea editar o registrar una nueva fruta

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

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

ngOnInit() {
  this.idProducto = this.route.snapshot.params['idProducto'];

  if (this.idProducto) {
    //Editar
    this.productoService.obtenerProducto(this.idProducto).subscribe(data => {
      this.producto = data[0];
    }, error => {
      console.log(error);
    });
  }
  else {
    console.log('Nuevo Producto');
  }
}
```

Cuando se utiliza la función **onSubmit**, se lleva a cabo la acción de modificar o registrar, dependiendo de si el **idProducto** ya existe o no

```
onSubmit() {
  if (this.producto.idProducto) {
    this.productoService.actualizarProducto(this.producto).subscribe(
      data => {
        console.log(data);
        this.router.navigate(['/acceder/administracion/productos']);
      }
    );
  }
  else {
    console.log('Nuevo Producto');
    this.productoService.agregarProducto(this.producto).subscribe(data => {
      this.router.navigate(['/acceder/administracion/productos']);
    });
  }
}
```

Vista

Edición de datos

Nombre

uva

Detalle

Uva cubana

Cantidad

39

Categoría

de temporada

Precio

5000

Imagen

Seleccionar archivo

Ninguno archivo selec.

Actualizar

encabezado

Incluye la sección superior de la página, que consta del logotipo de inicio, una barra de búsqueda y también botones para iniciar sesión, registrarse y cerrar sesión, los cuales varían según la ubicación actual en la página.

```
<div class="container-fluid">
  <a class="navbar-brand" [routerLink]="['/productos']">
    <img [src]="logo" alt="" width="65" height="53">
  </a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#n
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent" *ngIf="!esVisible()">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0 gap-5" *ngIf="!esVisibleReg()">
      <li class="nav-item">
        <a class="nav-link active" aria-current="page" [routerLink]="['/productos']">Inicio</a>
      </li>
    </ul>
  </div>
</div>
```

Por parte de la lógica se crearon funciones para hacer invisibles o invisibles algunos componentes según la ruta actual

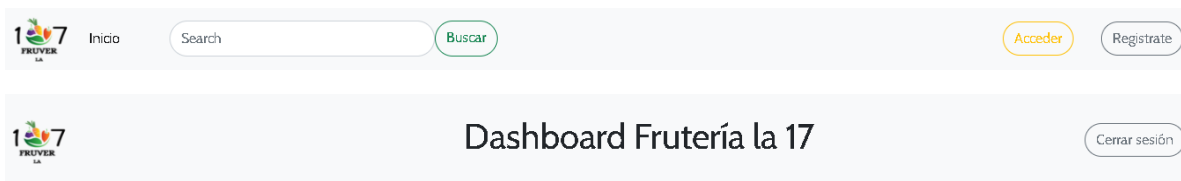
```

esVisible(): boolean {
  const currentUrl = this.router.url;
  return (
    currentUrl === '/acceder/administracion' ||
    currentUrl === '/acceder/administracion/productos' ||
    currentUrl === '/acceder/administracion/pedidos'
  );
}

esVisibleReg(): boolean {
  const currentUrl = this.router.url;
  return (
    currentUrl === '/registro/usuarios' ||
    currentUrl === '/acceder'
  );
}

```

Vista



filtro

Puede mostrar los productos de acuerdo con la categoría a la que pertenecen.

```

<div class="space" *ngIf="!esVisible()">
  <div class="d-flex gap-3">
    <div class="titulo">
      <legend><h6>Filtro</h6></legend>
    </div>
    <div class="row align-items-center">
      <div class="col-2">
        <select class="form-control" id="categoria" name="categoria" required [(ngModel)]="categoriaSeleccionada" (ngModelChange)
          <option value="">todos</option>
          <option value="citrico">citrico</option>
          <option value="pepita">de pepita</option>
          <option value="tropical">tropicales</option>
          <option value="baya">bayas</option>
          <option value="temporada">de temporada</option>
        </select>
      </div>
    </div>
  </div>
</div>

```

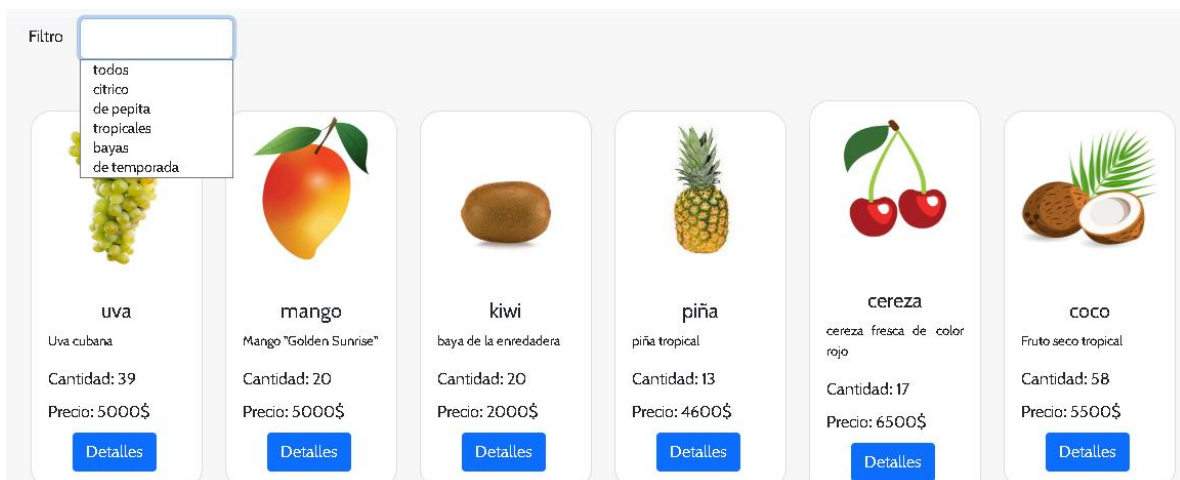
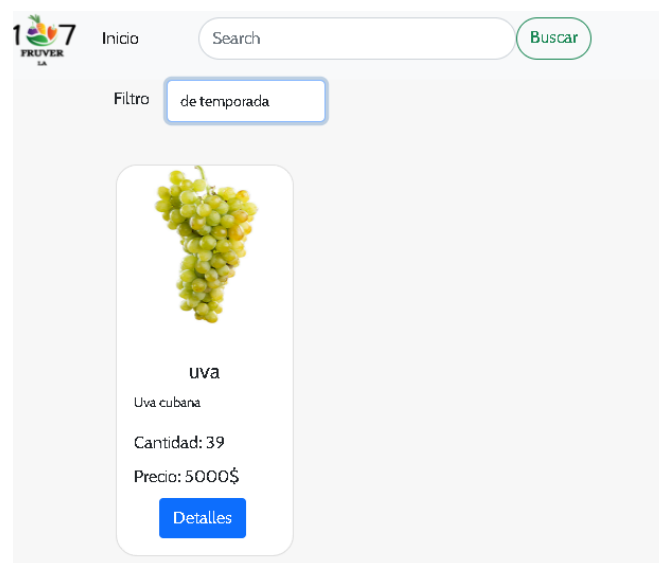
Permite mostrar los productos según las categorías establecidas en el componente select


```
export class FiltroComponent implements OnInit {
  constructor(public router: Router, private filtroProductosService: FiltroProductosService) {}
  categoriaSeleccionada: string | null = null;

  ngOnInit() {
    this.filtroProductosService.categoriaSeleccionada.next(this.categoriaSeleccionada);
  }
  onCategoriaSeleccionadaChange() {
    if (this.categoriaSeleccionada === '') {
      this.categoriaSeleccionada = null;
    }
    this.filtroProductosService.categoriaSeleccionada.next(this.categoriaSeleccionada);
  }
}
```

Por parte de la lógica se inicializa un string para luego en la función **onCategoriaSeleccionada** reconozca el valor seleccionado y luego compararlo con las categorías existentes

Vista



footer

Muestra el pie de pagina en todas las paginas web y componentes creados

```
<footer class="footer">
  <div class="container">
    <p>© 2023 Fruver la 17 Frutería. Todos los derechos reservados.</p>
    <p>Dirección: Calle #123, Pasto, Nariño</p>
    <p>Teléfono: 3214657876</p>
  </div>
</footer>
```

Vista

© 2023 Fruver la 17 Frutería. Todos los derechos reservados.

Dirección: Calle #123, Pasto, Nariño

Teléfono: 3214657876

listado-productos

visualiza los productos registrados en la base de datos o solamente muestran los productos filtrados por una categoría en especial

```
div class="container d-grid gap-5" >
  <div class="row align-items-center" >
    <div *ngFor="let producto of productos | async" class="col-2">
      <div class="" *ngIf="categoriaSeleccionada === null || producto.categoria === categoriaSeleccionada">
        <div class="card gap-3">
          
          <div class="card-body">
            <h5 class="card-title text-center">{{ producto.nombre }}</h5>
            <p class="card-text">{{ producto.detalle }}</p>
            <p class="card-title">Cantidad: {{ producto.cantidad }}</p>
            <p class="card-title">Precio: {{ producto.precio }}$</p>
            <a class="btn btn-primary mx-4" [routerLink]="['/productos/detalles/', producto.idProducto]">Detalle
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
/div>
```

Por parte de la lógica se inicializa la variable **categoriaSeleccionada** y se hace el llamado al servicio **filtroProductosService** con el fin de hacer el filtrado de productos y en la inicialización se obtiene los productos por medio del método **obtenerProductos**

```

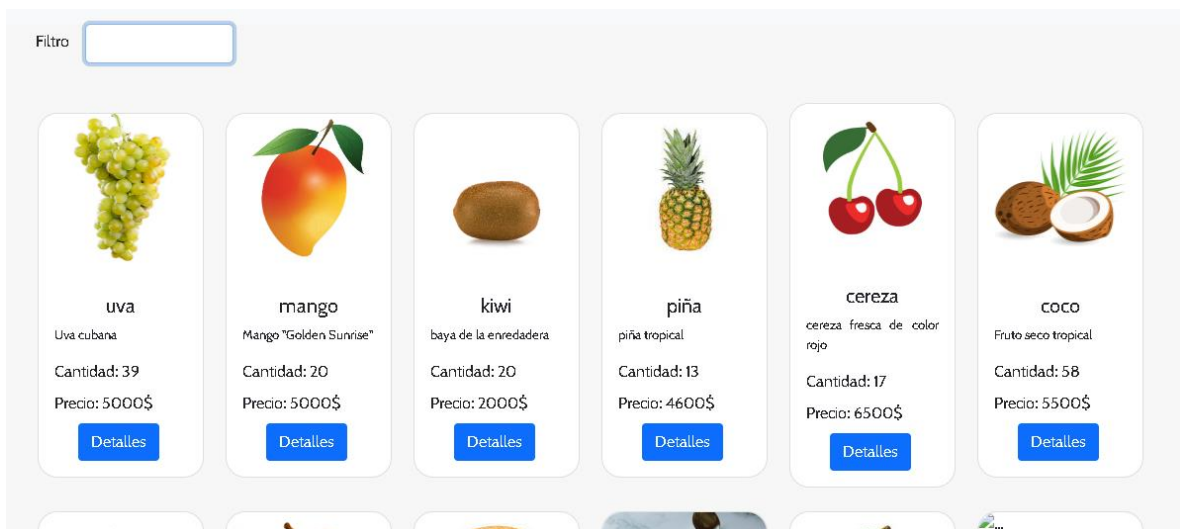
productos: Observable<ProductoModel[]> | undefined;
categoriaSeleccionada: string | null = null;

constructor(
  private productoService: ProductoService,
  private filtroProductosService: FiltroProductosService
) {}

ngOnInit(){
  this.productos = this.productoService.obtenerProductos();
  this.filtroProductosService.categoriaSeleccionada.subscribe(categoria => {
    this.categoriaSeleccionada = categoria;
  });
}

```

Vista



login

permite ingresar las credenciales del usuario con el fin de acceder al panel de administración

```
onents > login > login.component.html > div.container
<div class="card-header">
  <h3 class="card-title text-center">Iniciar sesión</h3>
</div>
<div class="card-body">
  <form (ngSubmit)="onSubmit()">
    <div class="mb-3 text-center">
      <label for="email" class="form-label">Correo electrónico</label>
      <input type="text" class="form-control" id="correo" name="correo" placeholder="correo@example.com">
    </div>
    <div class="mb-3 text-center">
      <label for="password" class="form-label">Contraseña</label>
      <input type="password" class="form-control" id="contrasena" name="contrasena" placeholder="Contraseña">
    </div>
    <div class="d-grid">
      <button type="submit" class="btn btn-primary">Iniciar sesión</button>
    </div>
  </form>
</div>
</div>
<br>
<div class="text-center">
  <button class="btn btn-warning" [routerLink]="['/productos']">Inicio</button>
</div>
```

Por parte de la lógica se valida que el usuario exista para acceder al panel de administración

```
onSubmit() {
  this.usuarioService.autenticarUsuario(this.correo,this.contrasena).subscribe(
    (response) => {
      console.log('Inicio de sesión exitoso:', response);
      this.router.navigate(['/acceder/administracion']);
    },
    (error) => {
      console.error('Error en el inicio de sesión:', error);
      this.router.navigate(['/acceder/']);
    }
  );
}
```

Vista

The image shows a login form titled "Iniciar sesión". It has a green header bar with the title. Below the header, there are two input fields: "Correo electrónico" (Email) and "Contraseña" (Password). The email field contains the text "correo@example.com". The password field contains the text "Contraseña". Below the password field is a green button labeled "Iniciar sesión". At the bottom of the form, there is a yellow button labeled "Inicio".

registro

Permite realizar el registro de usuarios al sistema y validando que se ingresen todos los campos para realizar esta operación

```
</div>
</fieldset>
<fieldset class="form-group">
  <div class="mb-3 text-center">
    <label for="password" class="form-label">Contraseña</label>
    <input
      type="password"
      class="form-control"
      id="contrasena"
      name="contrasena"
      placeholder="Contraseña"
      required
      [(ngModel)]="usuario.contrasena"
    />
  </div>
</fieldset>

<fieldset class="form-group">
  <div class="d-grid">
    <button type="submit" class="btn btn-primary" [disabled]="!usuarioForm.form.valid">Registrar
  </div>
</fieldset>
</form>
```

Por parte de la lógica se crea un nuevo objeto inicializado de el usuario con el fin de recoger esa información ingresada y enviarla al método **agregarUsuario**

```

usuario = new UsuarioModel("", "", "", "", "", "2");

constructor(private usuarioService: UsuarioService, private route: ActivatedRoute, private router: Router)

ngOnInit() {

}

onSubmit() {
  console.log('Nuevo Usuario');
  this.usuarioService.agregarUsuario(this.usuario).subscribe(data => {
    this.router.navigate(['/pedidos']);
  });
}
}

```

Vista

Registro

Nombre

Dirección

Correo electrónico

Contraseña

Registrarse

Inicio