

Asignatura: Programación III	Día y hora: Martes y Jueves de 6:30 a 8:10 p.m.	Grupo: 01
Docente: Ing. Rodrigo Ortiz, MIR	E-mail: d.ortiz@ufg.edu.sv	Aula: B25-26
Estudiante:		

Apellidos

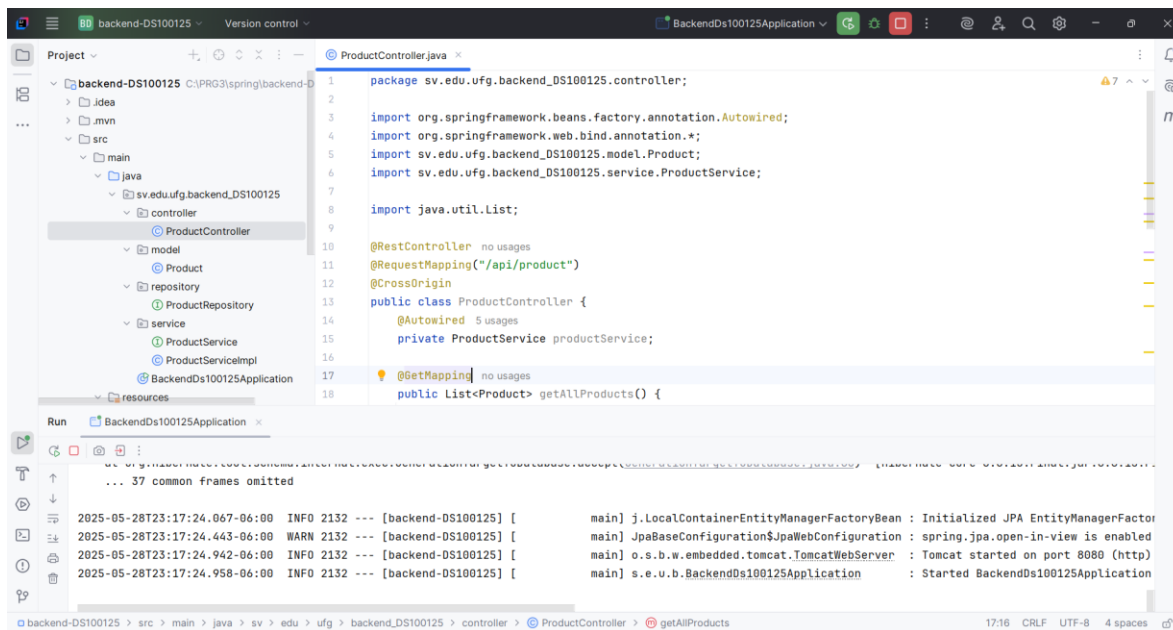
Nombres

Carrera:	Carné:	Fecha:	Calificación:
-----------------	---------------	---------------	----------------------

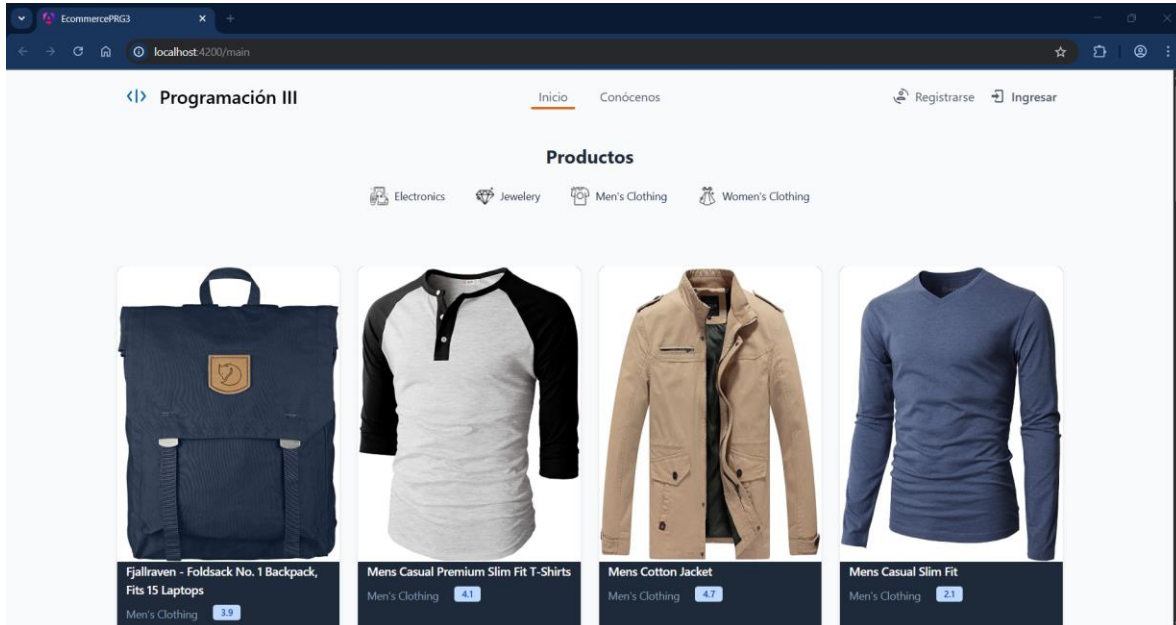
Realiza los pasos que ya conoces para restaurar la base de datos y para poner en funcionamiento tu API Rest (IntelliJ – Spring Boot) y tu Aplicación Ecommerce (Angular)

Tu laboratorio 4 consta de los siguientes requerimientos:

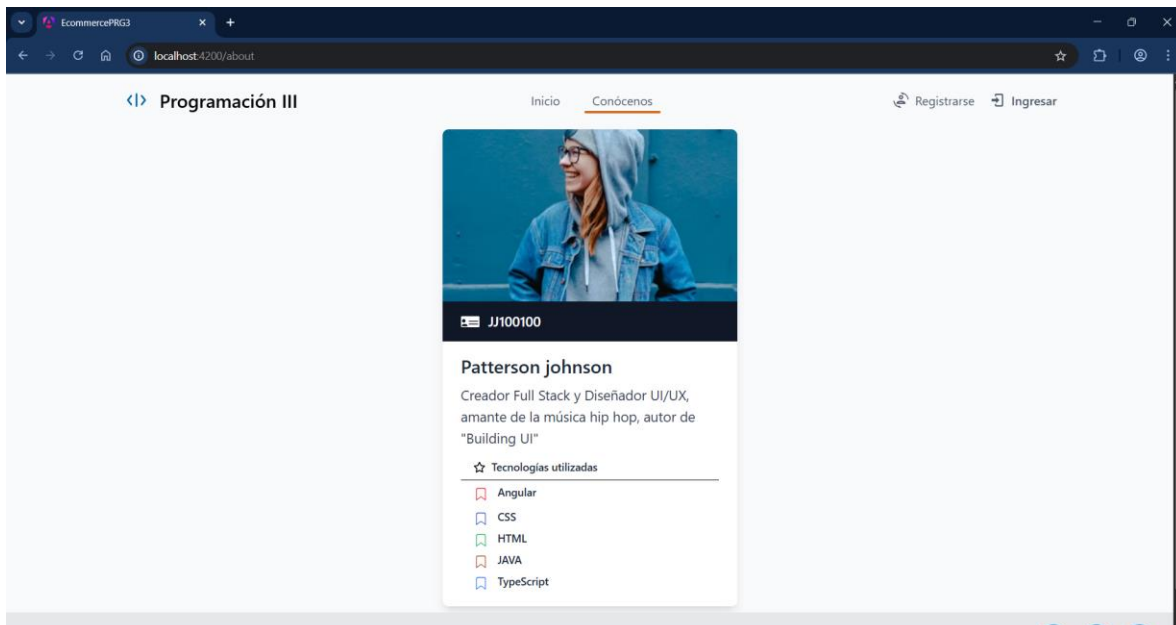
API Rest con Spring Boot CRUD (Funcional) (20%)



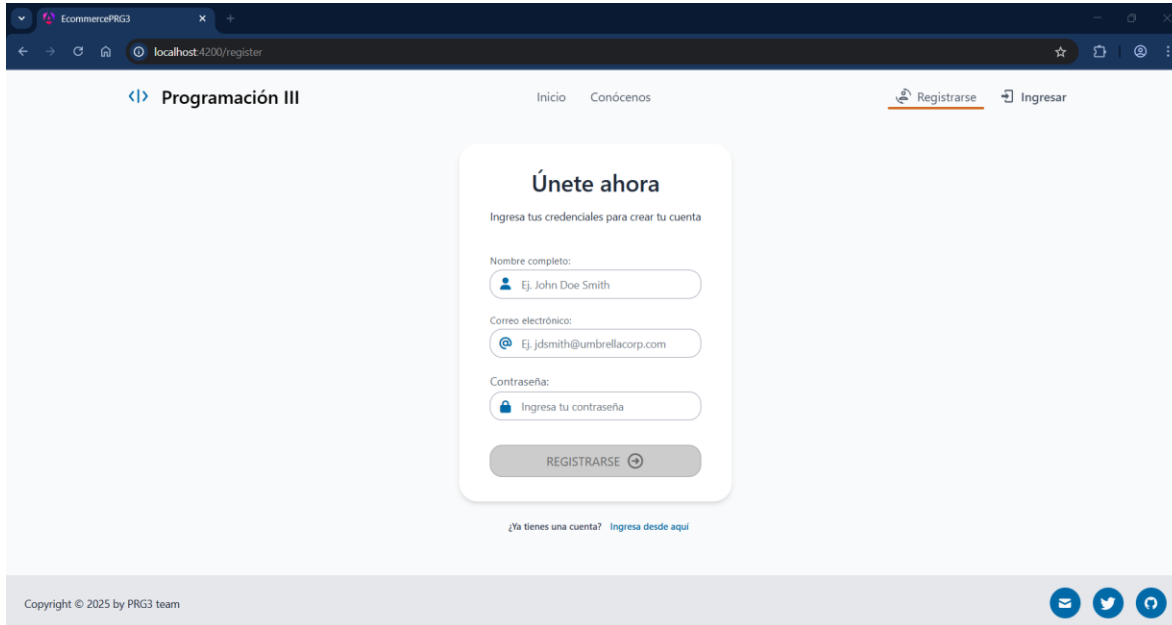
Listado de productos cargados desde el API (5%)



Pantalla del autor de la aplicación (5%) (Con tu información completa)

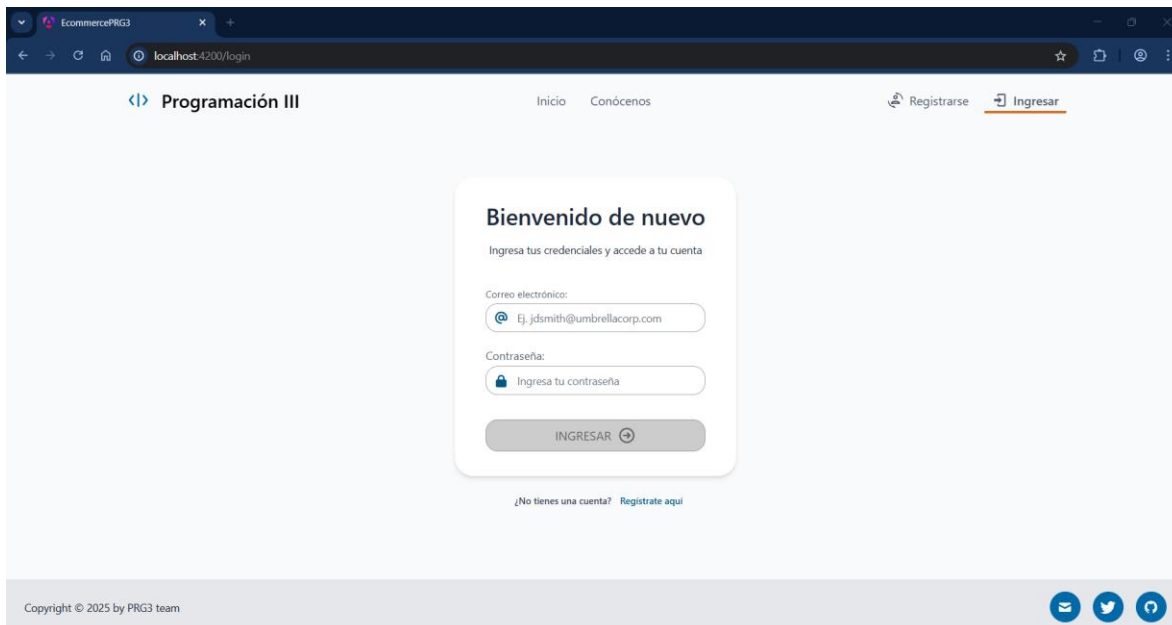


Diseño del Formulario de Registro (Prototipo) (5%)



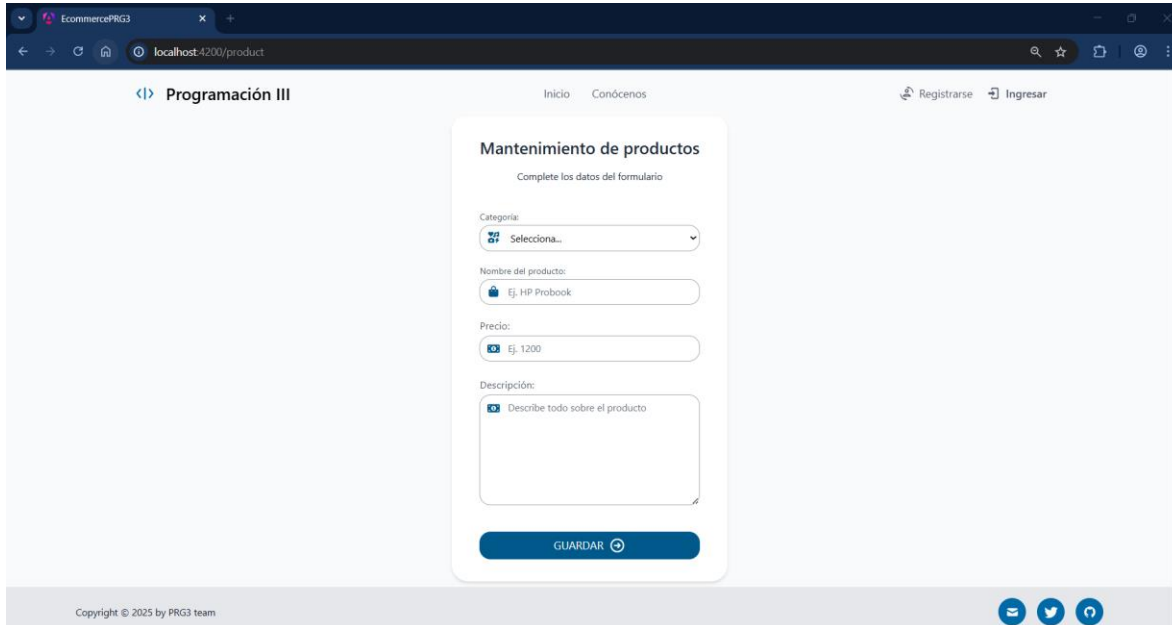
The screenshot shows a web browser window with the URL `localhost:4200/register`. The page title is "Programación III". The navigation bar includes links for "Inicio", "Conócenos", "Registrarse" (highlighted), and "Ingresar". The main content area features a registration form titled "Únete ahora" with the subtitle "Ingresa tus credenciales para crear tu cuenta". The form includes three input fields: "Nombre completo:" with the example "Ej. John Doe Smith", "Correo electrónico:" with the example "Ej. jdsmith@umbrellacorp.com", and "Contraseña:" with the placeholder "Ingresa tu contraseña". A "REGISTRARSE" button is at the bottom of the form. Below the form, there is a link: "¿Ya tienes una cuenta? [Ingresa desde aquí](#)". The footer contains the copyright notice "Copyright © 2025 by PRG3 team" and three social media icons (email, Twitter, and a circular arrow).

Diseño del Formulario de Login (Prototipo) (5%)



The screenshot shows a web browser window with the URL `localhost:4200/login`. The page title is "Programación III". The navigation bar includes links for "Inicio", "Conócenos", "Registrarse", and "Ingresar" (highlighted). The main content area features a login form titled "Bienvenido de nuevo" with the subtitle "Ingresa tus credenciales y accede a tu cuenta". The form includes two input fields: "Correo electrónico:" with the example "Ej. jdsmith@umbrellacorp.com" and "Contraseña:" with the placeholder "Ingresa tu contraseña". An "INGRESAR" button is at the bottom of the form. Below the form, there is a link: "¿No tienes una cuenta? [Regístrate aquí](#)". The footer contains the copyright notice "Copyright © 2025 by PRG3 team" and three social media icons (email, Twitter, and a circular arrow).

Mantenimiento de productos (Funcional) (20%)



Autenticación con JWT (Login funcional) (20%)

En tu proyecto de Angular crea lo siguiente utilizando el generador de componentes:

- Un **servicio** llamado **auth** colocado en una carpeta llamada **services**
- Un **guard** de nombre **auth** colocado en una carpeta llamada **guards**

```
1 ng generate guard guards/auth
```

Cuando pregunte el tipo de guard a crear selecciona CanActivate

```
? Which type of guard would you like to create? (Press <space> to select, <a> to toggle all,
<i> to invert selection, and <enter> to proceed)
>(*) CanActivate
( ) CanActivateChild
( ) CanDeactivate
( ) CanMatch
```

Configuración de las interfaces

Abre tu archivo **interfaces.ts** y agrega las interfaces para el token de autenticación y para las credenciales del usuario

```
1  export interface Product {
2    id: number | null;
3    title: string;
4    price: number;
5    description: string;
6    category: string;
7    image: string | null;
8    rating_rate: number | null;
9    rating_count: number | null;
10 }
11
12 export interface AuthToken {
13   access_token: string;
14   refresh_token: string;
15 }
16
17 export interface User {
18   email: string;
19   password: string;
20 }
```

Configuración de variable de entorno

Abre tu archivo **environment.ts** y agrega la propiedad **authURL** como se muestra a continuación:

```
1  export const environment = {
2    production: false,
3    debugMode: true,
4    baseUrl: "http://localhost:8080/api/product",
5    fakeUrl: "https://fakestoreapi.com/products/categories",
6    authURL: "https://api.escuelajs.co/api/v1/auth/login"
7  };
```

Configuración del servicio de autenticación

Abre tu archivo **auth.service.ts** y agrégale el siguiente código:

```
1 import { HttpClient } from '@angular/common/http';
2 import { EventEmitter, inject, Injectable, signal } from '@angular/core';
3 import { Observable, tap } from 'rxjs';
4 import { AuthToken } from '../interfaces/interfaces';
5 import { environment } from '../../environments/environment';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class AuthService {
11   private http = inject(HttpClient)
12   isLoggedIn = signal(this.hasToken());
13
14   constructor() { }
15
16   login(credentials: {email: string, password: string}) : Observable<AuthToken>{
17     return this.http.post<AuthToken>(`${environment.authURL}`, credentials).pipe(
18       tap(response => {
19         this.saveToken(response.access_token)
20         this.isLoggedIn.set(true)
21       })
22     )
23   }
24
25   saveToken(token: string): void{
26     localStorage.setItem('token', token)
27   }
28
29   getToken(): string | null{
30     return localStorage.getItem('token')
31   }
32
33   clearToken(): void{
34     localStorage.removeItem('token')
35   }
36
37   logout(): void{
38     this.clearToken()
39     this.isLoggedIn.set(false)
40   }
41
42   private hasToken(): boolean {
43     const token = this.getToken();
44     return !!token && token.length > 0;
45   }
46 }
```

Configuración del guard para proteger las rutas (10%)

Abre tu archivo **auth.guard.ts** y agrégale el siguiente código:

```
1  import { inject } from '@angular/core';
2  import { CanActivateFn, Router } from '@angular/router';
3  import { AuthService } from '../services/auth.service';
4
5  export const authGuard: CanActivateFn = (route, state) => {
6      const authSrv = inject(AuthService)
7      const router = inject(Router)
8      if(authSrv.isLoggedIn())
9      {
10         return true;
11     }
12     else{
13         router.navigate(['/login'])
14         return false
15     }
16 };
```

Configuración de las rutas agregando CanActivate

Abre tu archivo **app.routes.ts** y agrégale el siguiente código:

```
1 import { Routes } from '@angular/router';
2 import { authGuard } from '../guards/auth.guard';
3
4 export const routes: Routes = [
5   {
6     path: '',
7     redirectTo: '/main',
8     pathMatch: 'full'
9   },
10  {
11    path: 'about',
12    loadChildren: () => import('../pages/about/about.component'),
13  },
14  {
15    path: 'login',
16    loadChildren: () => import('../pages/login/login.component'),
17  },
18  {
19    path: 'main',
20    loadChildren: () => import('../pages/main/main.component'),
21  },
22  {
23    path: 'register',
24    loadChildren: () => import('../pages/register/register.component'),
25  },
26  {
27    path: 'product',
28    canActivate: [authGuard],
29    loadChildren: () => import('../pages/product/product.component'),
30  },
31  {
32    path: 'product/:id',
33    canActivate: [authGuard],
34    loadChildren: () => import('../pages/product/product.component'),
35  },
36  {
37    path: '**',
38    redirectTo: '/main',
39  },
40 ];
```


Configuración de la lógica para la página de login

Abre tu archivo **login.component.ts** y agrégale el siguiente código:

```
1  import { Component, inject } from '@angular/core';
2  import { FormBuilder, ReactiveFormsModule, Validators } from '@angular/forms';
3  import { Router, RouterLink } from '@angular/router';
4  import { AuthService } from '../../services/auth.service';
5  import { User } from '../../interfaces/interfaces';
6
7  @Component({
8    selector: 'app-login',
9    imports: [RouterLink, ReactiveFormsModule],
10   templateUrl: './login.component.html',
11   styleUrls: ['./login.component.css']
12 })
13 export default class LoginComponent {
14   private frmBuilder = inject(FormBuilder);
15   private authSrv = inject(AuthService)
16   private router = inject(Router)
17
18   loginForm = this.frmBuilder.group({
19     email: ['', [Validators.email, Validators.required]],
20     password: ['', Validators.required],
21   });
22
23   onLogin(){
24     console.table(this.loginForm.value);
25
26     if(this.loginForm.valid){
27       this.authSrv.login(this.loginForm.value as User).subscribe(
28         {
29           next: () => {
30             this.loginForm.reset()
31             // inicio de sesión exitoso, puede redirigir a otra página
32             this.router.navigate(['/'])
33           },
34           error: (err) => {
35             console.log(`Error en el inicio de sesión ${err}`);
36           }
37         }
38       )
39     }
40   }
41 }
```

Configuración del componente principal

Abre tu archivo **app.component.ts** y agrégale el siguiente código:

```

1  import { Component, inject, signal } from '@angular/core';
2  import { Router, RouterOutlet } from '@angular/router';
3  import { HeaderComponent } from '../components/header/header.component';
4  import { FooterComponent } from '../components/footer/footer.component';
5  import { AuthService } from '../services/auth.service';
6
7  @Component({
8    selector: 'app-root',
9    imports: [RouterOutlet, HeaderComponent, FooterComponent],
10   templateUrl: './app.component.html',
11   styleUrls: ['./app.component.css']
12 })
13 export class AppComponent {
14   private authSrv = inject(AuthService)
15   private router = inject(Router)
16
17   title = 'ecommerce-PRG3'
18   isLoggedIn = this.authSrv.isLoggedIn;
19
20   logout(){
21     this.authSrv.logout()
22     this.router.navigate(['/'])
23   }
24 }

```

Ejecuta tu proyecto y comprueba su funcionamiento en el navegador.

Criterio	Ponderación	Obtenido
API Rest con Spring Boot CRUD (Funcional)	20%	
Listado de productos cargados desde el API	5%	
Pantalla del autor de la aplicación	5%	
Formulario de Registro (Prototipo)	5%	
Formulario de Login (Prototipo)	5%	
Mantenimiento de productos (Funcional)	20%	
Autenticación con JWT (Login funcional)	20%	
Guards en rutas de formulario de productos	10%	
Archivo de WORD en formato .doc o .docx	10%	

Retos comodines:

La realización de uno de estos retos adicionales puede otorgar **1 punto** en la calificación, si y solo si es necesario:

- Agrega alertas de tipo toast a los eventos:

- Crear un nuevo producto
- Actualizar un producto
- Eliminar un producto



- Agrega un botón para poder cerrar la sesión en la barra de navegación, este botón solo debe mostrarse si el usuario está logueado.



- Para la entrega debes comprimir **todos** los archivos y carpetas, **exceptuando** la carpeta **node_modules** y tampoco la carpeta **.angular**
- Recuerda agregar este archivo en formato **.docx** o **.doc** cualquier otro formato será penalizado con - 1 punto.
- Todos los porcentajes de cada elemento de este laboratorio están sujetos a la **funcionalidad** del proyecto.