CES
Juan Pablo II
MADRID

MÓDULO PROYECTO

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Buyproducts

Aitor Tercero Hoya

Rubén Juárez Cádiz

ÍNDICE.

1. CONTENIDO.

Resumen

Español:

Esta aplicación está diseñada para que los usuarios puedan comprar unos productos disponibles mediante los enlaces que proporcionamos en nuestra página web.

El funcionamiento es muy básico para el usuario, ya que si no sabe cómo funciona la aplicación tiene una página de ayuda que indica los pasos a seguir. La primera pantalla muestra la página de inicio junto a un navbar con las opciones de iniciar sesión, o de crear una cuenta para registrarse. Al crear un nuevo usuario, este recibe un correo electrónico de confirmación. Una vez el usuario inicie sesión, volverá a la página de inicio, pero en el navbar ahora se mostrarán las categorías de nuestros productos, en las que se muestra una imagen, sus características básicas, el precio y un botón que ofrece la posibilidad de comprar.

Este botón te llevará a la página donde está disponible el producto que se quiere llegar a adquirir y nos ofrece la posibilidad de poder conocer las características del artículo de una manera más detallada.

Para salir de esta página web, solo hay que pulsar en el botón que se encuentra a la derecha del navbar, y le trasladará a la primera pantalla cerrando la sesión.

Inglés:

This application is designed so that users can purchase products available through the links we provide on our website.

The operation is very basic for the user, because if you don't know how the application works, it has a help page that indicates the steps to follow. The first screen shows the home page along with a navbar with the options to log in, or to create an account to register. When creating a new user, they receive a confirmation email. Once the user logs in, they will return to the home page, but the categories of our products will now be displayed in the navbar, in which an image, their basic characteristics, the price and a button that offers the possibility of buying are shown.

This button will take you to the page where the product you want to acquire is available and offers us the possibility of being able to know the characteristics of the item in a more detailed way.
To exit this web page, just click on the button to the right of the navbar, and it will take you to the first screen, closing the session.

Justificación.

Está aplicación ha sido creada para ayudar al público a encontrar determinados productos que se encuentran englobados en diferentes categorías, dentro de una misma página, sin tener que ir de una página web a otra. Por ejemplo en Nike, solo se puede llegar a adquirir artículos de deporte, en cambio en esta página podemos llegar a seleccionar todo en este mismo sitio web, ya que contamos con diferentes categorías.

Objetivos.

- Página web para crear un nuevo usuario e inicio de sesión.
- Conexión con base de datos (Supabase).
- Correcta gestión de los datos con la base de datos.
- Creación de una página de ayuda para poder facilitar el uso de nuestra aplicación.
- Creación de un navbar.
- Creación de los productos de cada categoría mediante la lectura de un archivo JSON.
- Creación del botón cerrar sesión.
- Creación de un Readme en Github para el usuario.

2. DESARROLLO.

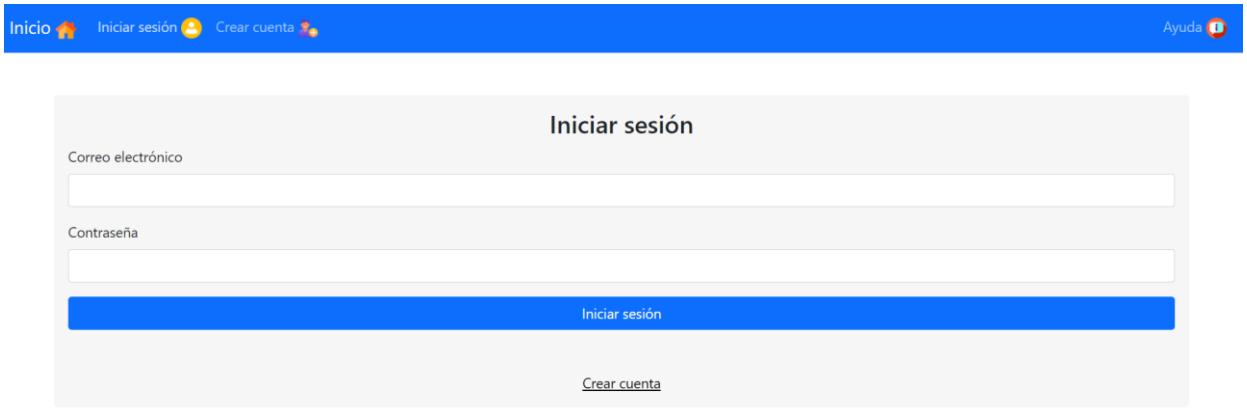
Implementación o Estudio.

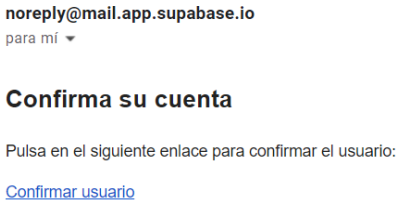
Todo el proyecto se ha realizado usando Angular y como framework para las clases en lugar de estar usando css, he usado Bootstrap. He ido subiendo todos los cambios que iba haciendo al repositorio de Github, para tenerlo actualizado en todo momento.

Lo primero en lo que me centré principalmente fue en crear el login, haciéndolo lo más estético posible para el usuario. La primera vez que se accede a la web hay que crear una cuenta, que se dará de alta en la base de datos, y desde ese momento cuando se quiera acceder a la web solo habrá que iniciar sesión con el correo electrónico y la contraseña que se había creado, porque el usuario ya se habría registrado previamente.

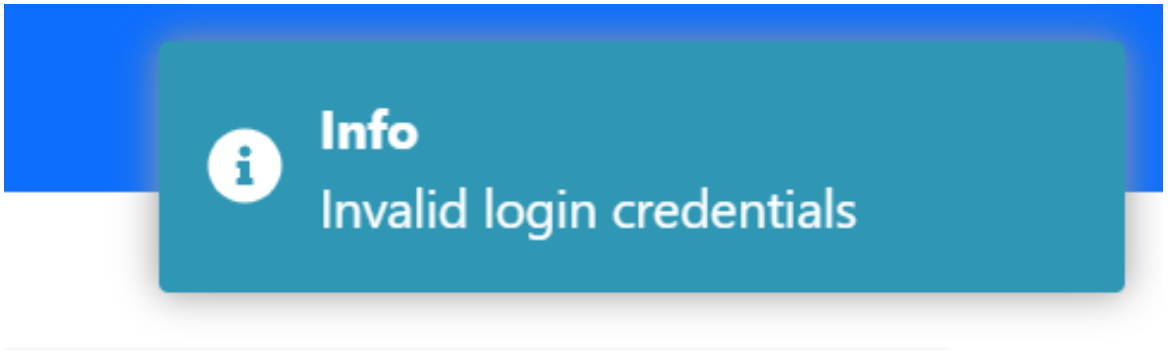
El login está conectado a una base de datos creada en Supabase, que antes de ver cómo hacer el login vi un tutorial de cómo se usaba Supabase, ya que primero tenía que tener un conocimiento previo de la base de datos que iba a usar.

Para poder conectar Supabase con el proyecto, es necesario instalar el paquete que gestiona Supabase.
Una vez que ya este todo preparado, probamos con la creación de un usuario, para asegurarnos de que se realiza correctamente la conexión con la base de datos.
Al crear una nueva cuenta, se pedirá al usuario que rellene un formulario para darse de alta en la base datos. En este caso el usuario tendrá que introducir un correo electrónico y una contraseña, y una vez creada la cuenta, habrá que confirmar la misma, yendo al correo introducido, para aceptar el email que le ha llegado.



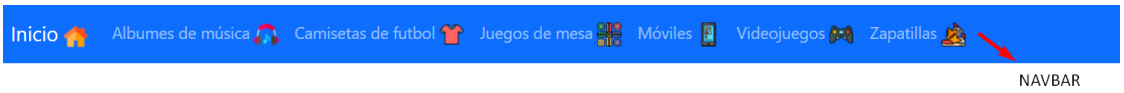


Cuando intentas hacer el login y el correo o la contraseña introducido son erróneos, aparecerá un mensaje de animación en el lado derecho de la pantalla, diciéndonos que los credenciales usados son erróneos. Este mensaje conseguimos mostrarlo gracias a la librería ngx-toastr.

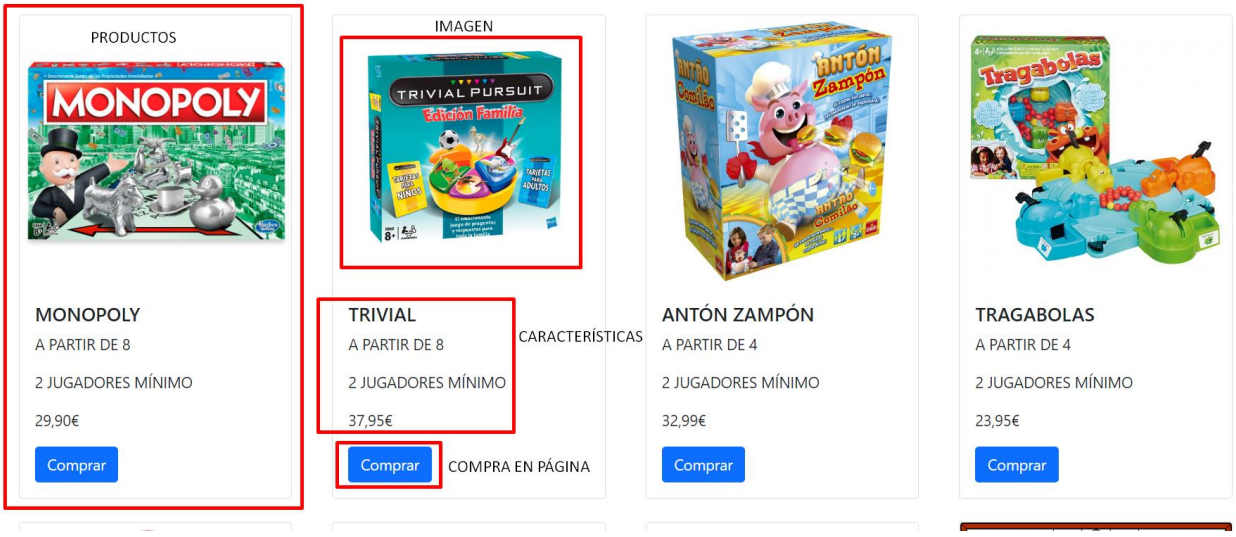


También hay creada una página de ayuda en el caso de que el usuario no conozca el funcionamiento de esta página, en la que se explica detalladamente los pasos a seguir, mediante el uso de imágenes poder disfrutar todo lo posible de esta página web.

Una vez se haya creado el usuario, y este ha iniciado sesión a través de sus credenciales, le aparecerá el navbar creado con las categorías de los productos que se ofertan.



Cada una de las categorías cuentan con una gran cantidad de productos de diferentes marcas, las cuales están hechas mediante las lecturas de JSON. En ellas se muestra una imagen del producto, características determinadas de cada artículo en función de su categoría, el precio, y un botón con opción de compra, que al pulsarlo nos llevará a una página web en la que ese producto seleccionado se encuentra disponible, además de contar con características más detalladas del mismo en esa página.



Cuando el usuario desea abandonar la página, se dirigirá a la derecha del navbar, donde se puede encontrar la opción de cerrar sesión y salir de la página.

También contamos con un Readme en el repositorio de Github tanto en español como en inglés con los pasos a seguir. Para poder ejecutar la aplicación en su ordenador, es obligatorio instalar las dependencias necesarias y cambiar las variables de la base de datos del usuario para que así pueda conectar con su base de datos, ya que si no seguirá estando conectado a la mía.

Resultados.

Los resultados obtenidos han sido los esperados, ya que desde un principio quería que la página web tuviera un login y un registro, que se guardara en una base de datos, en este caso Supabase.
He intentado el uso del menor código posible de CSS, gracias al uso de Bootstrap.
La lectura de archivos JSON también ha sido la esperada, ya que con su uso me ha sido más fácil y cómodo poder llegar a mostrar todos los productos en menos tiempo.

3. CONCLUSIONES.

Personalmente espero en un futuro, poder subir está página web a algún servidor como son Heroku, GitHub Pages o Netifly y así ya no tener la página en localhost.
Otra de las cosas a mejorar sería la creación de nuevas categorías para que a su vez existan más productos disponibles para la venta al público. Pero lo más importante sería poder llegar a algún acuerdo con las empresas de las páginas webs para que por cada usuario que compre uno de nuestros productos anunciados, el creador de la página web se lleve un pequeño beneficio por esa venta.
No estaría mal pensado que en un futuro la página tenga funcionabilidad para poder ser traducida en otros idiomas, y así poder expandir la web por otros países, ya que de momento solo opera en España.

4. REVISIÓN BIBLIOGRÁFICA.

- YouTube creador: Domini Code: “Alternativa para Firebase - Crear API Rest con Supabase.”
  Link: https://www.youtube.com/watch?v=vp3amSFh3aM.

- YouTube creador: Domini Code: “Login con Angular 13, Supabase y Bootstrap 5. Login con Angular 13.”
  Link: https://www.youtube.com/watch?v=qUhkSk52fV8

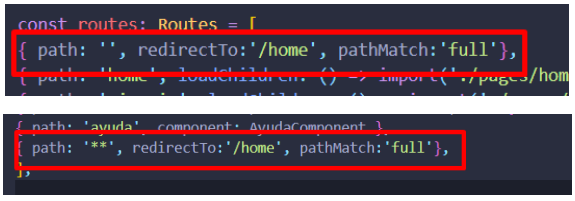
5. ANEXOS.

Manual técnico.

He usado Supabase como base de datos para guardar las cuentas de los usuarios que se registran para usar la aplicación. Para usar el paquete de supabase, lo tendremos que instalar en la consola de comandos usando: **npm i @supabase/supabase-js**.
También he usado bootstrap para evitar tener mucho código de CSS, y así tenerlo de una manera más ordenada. Para usar bootstrap he instalado el paquete en la terminal de comandos usando: **npm i bootstrap**.
Para crear la página web he usado Angular.

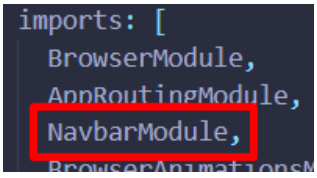
En la página de Supabase, una vez se crea la base de datos, mostrará la página de inicio de la base de datos creada en la que se mostrarán dos campos importantes como son la publickey y la url, las cuales nos permitirán el enlace entre el proyecto y la base de datos, mediante el archivo environments.ts, creando una variable a la que le pasaremos esos dos campos.
Además en la base de datos de Supabase, he activado una opción para que cuando el usuario cree una cuenta, reciba un correo de confirmación. Y habiendo modificado el mensaje del mismo.

En el app-routing.module.ts hace que por defecto se muestre la página de inicio cada vez que abrimos la aplicación, en la que explica el contenido de la web.
En lugar de tener una página error 404, cuando el usuario introduzca una ruta que no pertenece al proyecto, haremos que vuelva a la página de inicio.

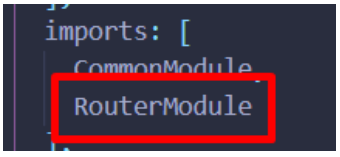


El comando para crear módulos que he empleado ha sido: ng g m nombreModulo. Donde la g significa generate y la m viene de module. Y he utilizado componentes creados mediante el comando: ng g c nombreComponente. Donde la g significa generate y la c de component.

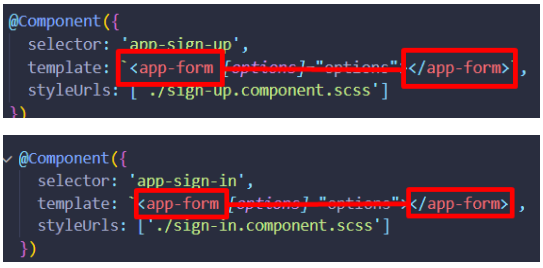
Una vez creado el componente navbar, exportaremos el navbarComponent, para que este se muestre en toda la aplicación, ya que lo importaremos en el app.module.ts.



Al importar RouterModule en el navbar.module.ts, permitir el manejo de rutas ya que es la funcionabilidad de este módulo.



Crearemos el componente form, que es el encargado de tener el html tanto de iniciar sesión, como de registarse, para ello iremos al components.ts de iniciar sesión y de registarse y usaremos el app-form en templates e importaremos formModule.



Realizaremos la importación del ReactiveFormsModule en form.module.ts, que nos permite crear formularios reactivos.

Creamos el método initForm() en form.component.ts, al que le pasamos los campos de nuestro formulario, en este caso un email y una contraseña.

```
private initForm():void {
  this.authForm = this.fb.group({
    email: ['', Validators.required],
    password: ['', Validators.required]
  })
}
```

“Llamaremos” al campo email y contraseña en nuestro archivo html en el que se encuentra nuestro formulario, que hemos creado anteriormente en el método initForm().

Creamos un interfaz que tenga los campos id, y label en el form.component.ts, e importando el formModule tanto en iniciar sesión como en registarse. A continuación, realizamos una constante, que nos facilitará realizar el cambio del formulario.

```
export const ACTIONS = {
  signIn: 'Iniciar sesión',
  signUp: 'Crear cuenta',
}
```

Y ahora esta constante la “llamaremos” en nuestro component.ts de iniciar sesión y de registarse.

```
options: OptionsForm = {
  id: ACTIONS.signIn,
  label: ACTIONS.signIn
}
```

```
options: OptionsForm = {
  id: ACTIONS.signUp,
  label: ACTIONS.signUp,
}
```

Ahora en el form.component.html solo tendremos que llamar a nuestra variable options.label para que nos muestre el título de la opción correcta si el usuario se está registrando o iniciando sesión. El texto del botón también se cambiará mediante el uso de la variable options.label, y mostrará la opción de cambiar de inicio de sesión a registrarse mediante un texto debajo del bóton, mediante el uso de la directiva *ngIf, y la variable options.id.

```
<h3 class="h3">
  {{options.label}}
</h3>
```

```
<div class="d-grid">
  <button type="submit" class="btn btn-dark">{{options.label}}</button>
</div>
<div class="text-center mt-5">
  <ng-container *ngIf="options.id === signIn; else showSignTemplate">
    <a [routerLink]="['/sign-up']" class="text-black">
    Crear cuenta
    </a>
  </ng-container>
</div>
</form>
<ng-template #showSignTemplate>
  <a [routerLink]="['/sign-in']" class="text-black">
  Ya tengo cuenta
  </a>
```

Crear la variable SupabaseClient en el archivo auth.service.ts, y en el constructor se le pasará la url y la publickey, que nos viene de la variable supabase creada anteriormente en el archivo environment.ts.

Crearemos la función iniciar sesión y registarse, a las cuales se les pasan las credenciales, y en caso de que falle nos devuelve un error y en caso de que funcione correctamente, nos devuelve un usuario.

```
async signIn(credentials: UserCredentials): Promise<supabaseResponse>{
  try {
    const {user, error, ...rest} = await this.supabaseClient.auth.signIn(credentials);
    this.setUser();
    return error ? error : user;

  } catch (error) {
    console.log(error);
    return error as ApiError;
  }
}

async signUp(credentials: UserCredentials): Promise<supabaseResponse>{
  try {
    const {user, error, ...rest} = await this.supabaseClient.auth.signUp(credentials);
    this.setUser();
    return error ? error : user;

  } catch (error) {
    console.log(error);
    return error as ApiError;
  }
}
```

Creamos el método setUser(), que almacena una constante de inicio de sesión temporal, la cual habremos creado en nuestro archivo de constantes.

```
private setUser():void{
  const session = localStorage.getItem(USER_STORAGE_KEY) as unknown as User;
  this.userSubject.next(session);
}
```

```
export const USER_STORAGE_KEY = 'supabase.auth.token';
```

Creamos la función onSubmit() para el formulario, que mediante la variable actionToCall hará la ejecución del botón, y en caso de que funcione devuelve al usuario a la página de inicio.

```
async onSubmit(): Promise<void>{
  const credentials: UserCredentials = this.authForm.value;
  let actionToCall;

  if(this.options.id === ACTIONS.signIn){
    actionToCall = this.authSvc.signIn(credentials);
  } else {
    actionToCall = this.authSvc.signUp(credentials);
  }

  try {
    const result = await actionToCall as UserResponse;
    if(result.email){
      this.redirectUser();
    } else{
      this.toastSvc.info(result.message, 'Info');
    }
  } catch (error) {
    console.log(error);
  }
}
```

Crear la función onLogout() en el navbar.components.ts para implementarlo en el html, en la opción de cerrar sesión, en la que una vez se pulsa llevará al usuario a la página de inicio mostrando las opciones de iniciar sesión y registrarse.

```
user$ = this.authSvc.user$;
constructor(private readonly authSvc: AuthService) { }

async onLogout():Promise<void>{
  try {
    await this.authSvc.signOut();
  } catch (error) {
    console.log(error);

  }
}
```

Mediante el uso de la directiva *ngIf permite que muestre las opciones del navbar de iniciar sesión y de registrase, en caso de no haber ningún usuario con la sesión abierta, y en caso de que exista esa sesión mostrará en el navbar las categorías de los productos disponibles y la opción de cerrar sesión.

```
<div class="collapse navbar-collapse" id="navbarNavAltMarkup">
  <ng-container *ngIf="(user$ | async) === null; else logOutTemplate">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
      <li class="nav.item">
```

```
</ng-container>
<ng-template #logOutTemplate>
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item">
      <a class="nav-link" [routerLink]="['/albumesmusi
    </li>
    <li class="nav-item">
      <a class="nav-link" [routerLink]="['/camisetasfu
    </li>
    <li class="nav-item">
      <a class="nav-link" [routerLink]="['/juegosdemes
```

También el uso de guards, los cuales nos ayudarán a que si se encuentra una sesión abierta por parte de un usuario, y este intenta volver a las páginas de iniciar sesión o de registrarse, lo devolverá a la página de inicio, ya que está registrado. En el app-routing.module.ts declaramos las rutas en las que queremos pasar los guards creados anteriormente.

```
export class AuthGuard implements CanActivate{
  constructor(private readonly router: Router){}
  canActivate(): boolean{
    if(localStorage.getItem(USER_STORAGE_KEY)){
      this.router.navigate(['/home']);
      return false;
    }
    return true;
  }
}
```

```
dule').then(m => m.HomeModule) },
-in/sign-in.module').then(m => m.SignInModule), canActivate: [AuthGuard]},
-up/sign-up.module').then(m => m.SignUpModule), canActivate: [AuthGuard]},
```

En caso de que el usuario intente iniciar sesión con unas credenciales incorrectas, le aparecerá un mensaje de error, mediante el uso de la librería ngx-toastr, la cual tendremos que haber instalado mediante el uso del comando: **npm i ngx-toastr** e importar los módulos necesarios en el app.module.ts.

```
imports: [
  BrowserModule,
  AppRoutingModule,
  NavbarModule,
  BrowserAnimationsModule,
  HttpClientModule,
  ToastrModule.forRoot({
    preventDuplicates: true,

  })
```

Para que nos muestre los productos de cada una de las categorías, creamos el archivo json, en el cual estarán definidos unos campos, y en el component.ts elegimos los elementos del objeto que queremos que aparezacan en el html.

```
export class AlbumesmusicaComponent implements OnInit {

  albumes : {imagen: string, nombre: string, cantante: string, numerocanciones: string, precio: string, url: string}[] = albumesmusica;

  constructor() { }

  ngOnInit(): void {
  }
}
```

Y se mostrarán los productos creados mediante las cartas de bootstrap de una manera ordenada, mostrando elemento a elemento los artículos mediante el uso de la directiva *ngFor.

Manual de usuario.

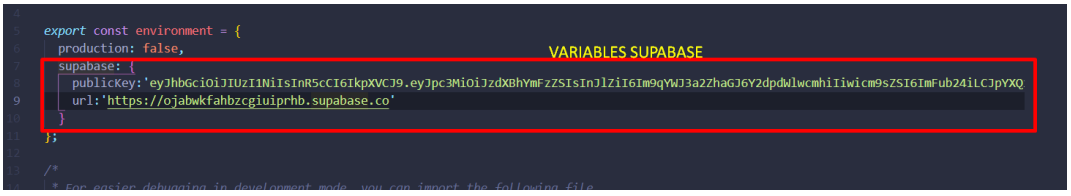
Lo primero que debes hacer es descargar de Github el proyecto en formato zip, para después descomprimirlo. El repositorio donde se encuentra el proyecto es https://github.com/Aitor-tercero/TFG.

Lo siguiente que vamos a hacer es abrir la carpeta TFG descomprimida en nuestro Visual Studio Code, pero antes de poder arrancar la aplicación debemos instalar las dependencias que se encuentran en el archivo package.json, ya que si no la aplicación no funcionará.

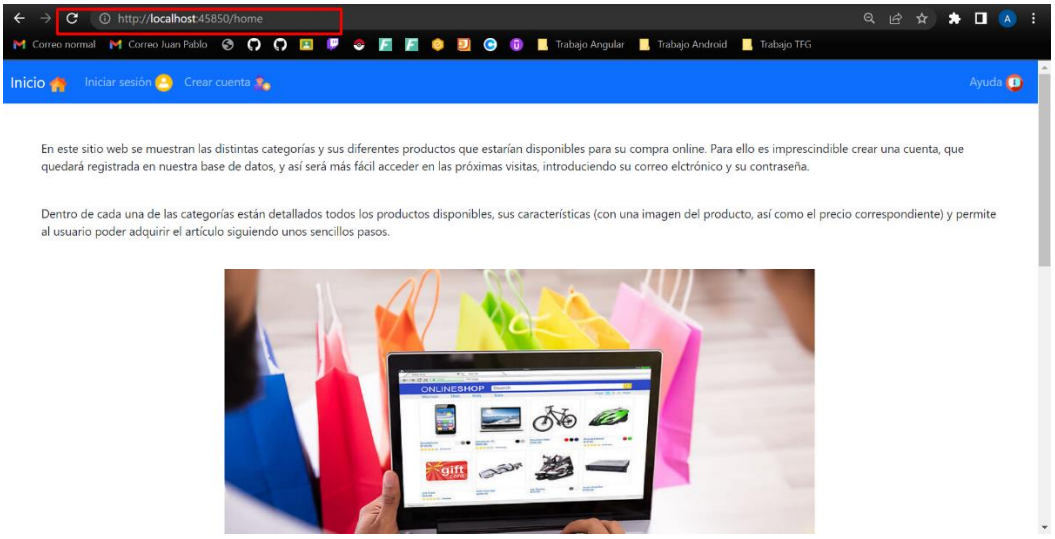
Para instalar estas dependencias, tendremos que abrir la consola de comandos de Visual Studio Code, y poner los siguientes comandos para descargar las dependencias:

- npm i bootstrap @supabase/supabase-js → Este comando es para instalar tanto Bootstrap como el paquete de Supabase.
- npm i ngx-toastr → Este comando es para instalar el paquere de animaciones, para que cuando el usuario no entre con sus credenciales correctas le salte un mensaje.

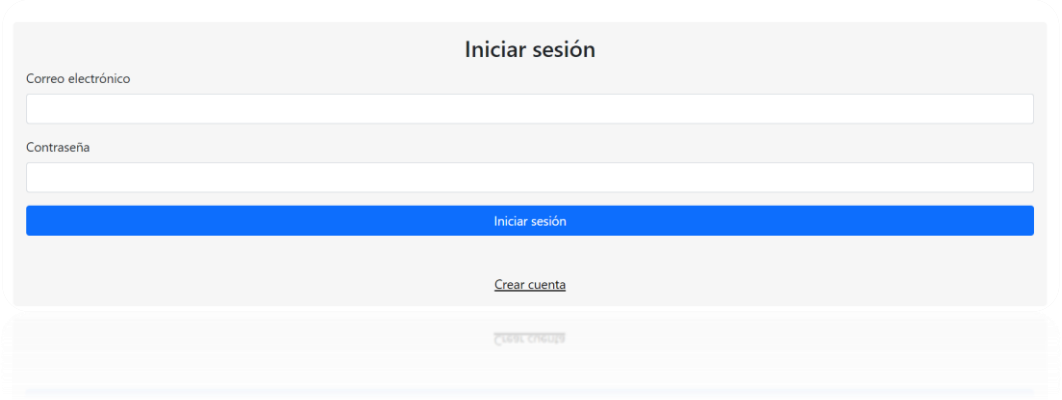
Una vez realizado este paso debemos ir al archivo environments.ts, que se encuentra en la carpeta environments, donde el usuario debe introducir la publickey y la url de su base de datos en Supabase, para que deje de estar conectado a mi base de datos.



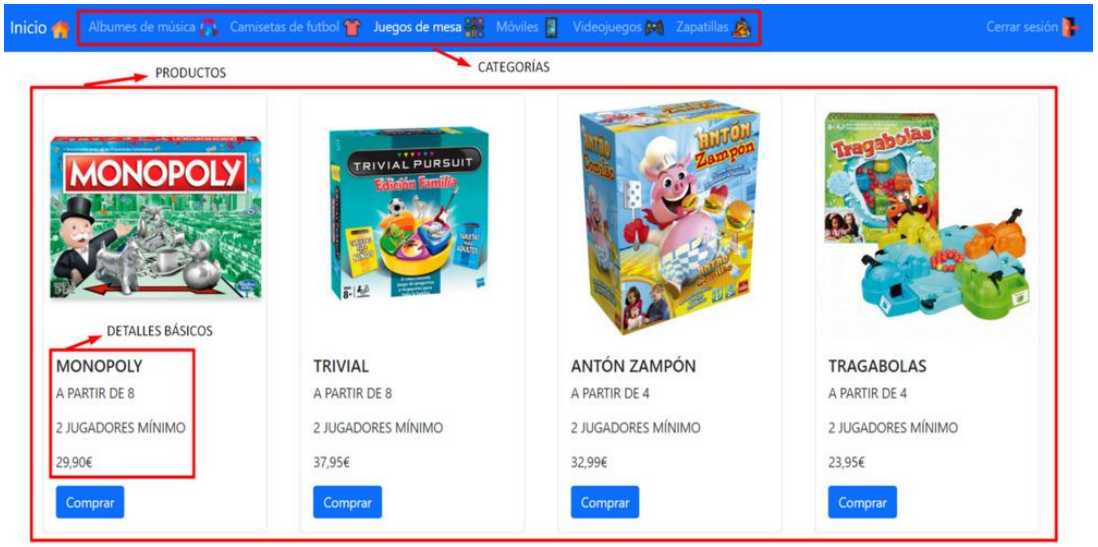
Con las variables cambiadas y las dependencias instaladas solo tenemos que hacer ng serve para arrancar el servidor y se abrirá en nuestro navegador en la ruta http://localhost:45850/home.



Lo primero que debes hacer para disfrutar de la aplicación es crear una cuenta siempre y cuando no tengas una cuenta existente, y si estás dado de alta, solo tienes que dirigirte donde pone iniciar sesión y rellenar el formulario introduciendo el correo electrónico y la contraseña usados anteriormente.



Una vez el usuario haya iniciado sesión, le aparecerán en el navbar las categorías de los productos de los que disponemos. En cada categoría, los artículos tienen una imagen, unas características básicas, el precio del producto y un botón de comprar, que trasladará al usuario a la página de compra donde está disponible. La mayor parte de los artículos de la web de compra cuentan con una explicación más detallada de sus características.



Cuando el cliente quiera abandonar el sitio web, solo tiene que clicar en el botón de la parte derecha del navbar (cerrar sesión), y directamente volverá a la página de inicio, habiendo desaparecido las categorías y volviendo a aparecer en el navbar las opciones del login (iniciar sesión o crear cuenta).