**FutFlexShop**

Memoria del Proyecto del Ciclo Formativo de GS IFC303

**Desarrollo de Aplicaciones Web**

Curso: 2023/2024

**Marco Formación**

Autor: **Ibrahim El Assali**

**INDICE**

[1 Descripción. 1](#_Toc168997250)

[1.1 Descripción General del proyecto 1](#_Toc168997251)

[1.2 Contexto y Justificación del Proyecto 2](#_Toc168997252)

[2 Objetivos 3](#_Toc168997253)

[2.1 Objetivo principal 3](#_Toc168997254)

[2.2 Objetivos Específicos 3](#_Toc168997255)

[3 Recursos Empleados 4](#_Toc168997256)

[2.1. Next.js 5](#_Toc168997257)

[3.1.1 ¿Qué es Next.Js? 5](#_Toc168997258)

[3.1.2 Ventajas de Next.js frente a otros frameworks 5](#_Toc168997259)

[3.1.3 Comparación con Otros Frameworks 6](#_Toc168997260)

[3.2 MongoDB 7](#_Toc168997261)

[3.2.1 ¿Qué es MongoDb? 7](#_Toc168997262)

[3.2.2 Ventajas de MongoDB 7](#_Toc168997263)

[3.3 NextAuth.Js 9](#_Toc168997264)

[3.3.1 ¿Qué es NextAuthJs? 9](#_Toc168997265)

[3.3.2 Ventajas 9](#_Toc168997266)

[3.4 AWS y S3 11](#_Toc168997267)

[3.4.1 ¿Qué es AWS? 11](#_Toc168997268)

[3.4.2 ¿Para qué sirve AWS? 11](#_Toc168997269)

[3.4.3 ¿Qué es AWS S3? 11](#_Toc168997270)

[3.4.4 ¿Para qué sirve AWS S3? 12](#_Toc168997271)

[3.4.5 Ventajas de AWS S3 12](#_Toc168997272)

[3.5 Styled-Components 13](#_Toc168997273)

[3.5.1 ¿Qué es Styled-components? 13](#_Toc168997274)

[3.5.2 ¿Para qué sirve Styled-components? 13](#_Toc168997275)

[3.5.3 Ventajas de Styled-components: 13](#_Toc168997276)

[4 Desarrollo o Explicación 15](#_Toc168997277)

[4.1 Diseño UML de la aplicación web 15](#_Toc168997278)

[4.2 Desarrollo e implementación del proyecto 18](#_Toc168997279)

[4.2.1 Configuración inicial y creación del proyecto en NextJs 18](#_Toc168997280)

[4.3 Estructura del Proyecto 25](#_Toc168997281)

[5 Conclusión y Posibles Mejoras 30](#_Toc168997282)

[5.1 Resumen de Resultados: 30](#_Toc168997283)

[5.2 Posibles Mejoras: 30](#_Toc168997284)

[6 Bibliografía y Referencias 31](#_Toc168997285)

# Descripción.



## Descripción General del proyecto

Futflexshop es una e-commerce dedicada a la venta de camisetas y accesorios de fútbol. La plataforma está diseñada para ofrecer una experiencia de compra en línea intuitiva y eficiente tanto para los administradores como para los clientes que son aquellos usuarios que se registran o entran a la web incluso sin realizar ningún pedido. La aplicación se divide en dos partes principales:

Por un lado, se distingue la parte Administrativa:

1. Gestión de Productos: Se podrá añadir eliminar y editar productos en la interfaz gestionando precios, imágenes, categorías, descripciones.
2. Gestión de categorías: Se podrá editar añadir y eliminar categorías los cuales también tendrán ramas raíces y subcategorías con propiedades para cada categoría.
3. Gestión de Pedidos: En esta sección se podrá revisar datos y realizar un seguimiento de las ordenes

Por otro lado, tenemos la parte Cliente:

1. Catálogo de Productos: Los usuarios pueden navegar por una amplia gama de camisetas y accesorios de fútbol, con opción de filtrado y búsqueda de productos según las preferencias de cada cliente.
2. Carrito de Compras: Funcionalidad de carrito que permite a los usuarios agregar los productos o eliminarlos y proceder posteriormente al pago.

La aplicación se ha desarrollado con un enfoque integral para garantizar una experiencia de usuario fluida y eficiente. Hemos optado por Next.js por su capacidad de renderizado del lado del servidor y generación de sitios, lo que impulsa un rendimiento óptimo y una navegación sin interrupciones. Además, hemos integrado MongoDB para gestionar la base de datos, garantizando una gestión eficaz de los datos. Para el almacenamiento en la nube de imágenes, confiamos en AWS S3, aprovechando su robustez y escalabilidad.

En cuanto al diseño, hemos adoptado Tailwind CSS y Styled-components para garantizar una apariencia moderna y adaptable, brindando flexibilidad y facilidad de mantenimiento en el desarrollo de estilos. Esta combinación de tecnologías líderes nos permite ofrecer una aplicación poderosa, rápida y estéticamente atractiva para nuestros usuarios.

## Contexto y Justificación del Proyecto

Futflexshop se distingue no solo por su amplio catálogo de productos de alta calidad, sino también por su compromiso inquebrantable con la excelencia en el servicio al cliente. Nos esforzamos por superar las expectativas en cada interacción, brindando atención personalizada y asesoramiento experto a nuestros clientes. Además, nuestra plataforma está constantemente actualizada con las últimas tendencias y novedades en el mundo del fútbol, asegurando que los aficionados siempre tengan acceso a lo más nuevo y emocionante. En Futflexshop, no solo vendemos productos, construimos una comunidad apasionada de amantes del fútbol, ofreciendo una experiencia de compra incomparable que va más allá de lo ordinario.

# Objetivos

## Objetivo principal

El principal enfoque es la creación de un ecommerce intuitivo y fácil de usar tanto para clientes como para administradores. Mientras que los clientes disfrutan de una experiencia de compra fluida y personalizada, los administradores cuentan con un panel de control robusto y amigable que les permite gestionar eficientemente todos los aspectos del negocio en línea. Desde la edición y gestión de productos, pedidos y categorías, hasta el control del inventario, el sistema proporciona herramientas potentes pero simples de usar. Se busca hacer que la gestión del ecommerce sea una tarea eficiente y sin complicaciones, permitiendo a los administradores concentrarse en hacer crecer su negocio sin distracciones.

## 2.2 Objetivos Específicos

* Implementación de la Autenticación y Autorización
* Configurar NextAuth para gestionar la autenticación de usuarios utilizando Google como proveedor
* Gestión de Productos
* Crear una interfaz de administración que permita a los administradores añadir, editar y borrar productos.
* Implementar un sistema de categorías para organizar los productos
* Gestión de Categorías
* Crear una interfaz de administración que permita a los administradores añadir, editar y borrar Categorías.
* Almacenamiento de Imágenes de Productos
* Generar una interfaz de usuario y Experiencia de Usuario (UI/UX)
* Optimización del Rendimiento

# Recursos Empleados

A continuación, se va a hacer un recorrido por las tecnologías, webs y recursos utilizados en el proyecto.

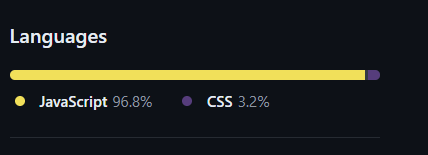
En la primera fase del proyecto, corresponde con el diseño de la aplicación se ha utilizado Diagrams.net, el cual es un software que permite crear diagramas mediante lenguaje UML, este lenguaje me permite representar … y fue usado en los siguientes casos:

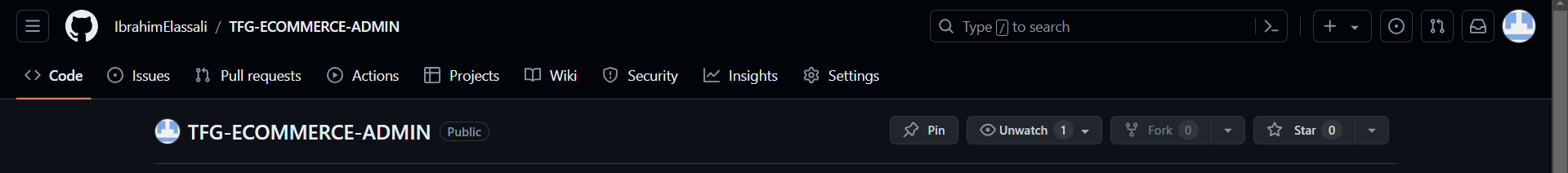
* Diagramas de casos de uso
* Diagramas de flujo
* Diagramas de

Una vez realizados todos los diseños y diagramas, se comienza con el despliegue de la aplicación usando Next.js. Primero, se asegura de tener el entorno de desarrollo configurado correctamente, instalando Node.js y el CLI de Next.js.

Después de la creación del proyecto, se navega al directorio de este e inicia el servidor de desarrollo (Yarn dev) y posteriormente, se configuran las variables de entorno necesarias para la aplicación, asegurándose de que todas las claves y configuraciones estén correctamente establecidas en un archivo .env

Como herramienta de gestión de base de datos se ha utilizado MongoDB además de AWS S3 para el almacenanimiento de las imágenes de los productos y como IDE se ha utilizado Visual Studio Code al estar hecha completamente en JavaScript como se puede ver en la siguiente captura:





Por otro lado, como bien se ve se usa GitHub donde se aloja el repositorio y Git como sistema de control de versiones

## Next.js

### ¿Qué es Next.Js?

Next.js es un framework de desarrollo web basado en React que facilita la creación de aplicaciones web tanto del lado del cliente como del servidor. Fue desarrollado por Vercel y se ha destacado por sus capacidades de renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG), lo que mejora el rendimiento y la optimización para motores de búsqueda (SEO).

### Ventajas de Next.js frente a otros frameworks

* Renderizado del Lado del Servidor (SSR) y Generación de Sitios Estáticos (SSG):

Next.js permite pre-renderizar páginas en el servidor o durante la compilación, lo que resulta en tiempos de carga más rápidos y mejor SEO en comparación con frameworks que solo soportan renderizado del lado del cliente, como React puro​

* Facilidad de Integración:

Next.js se integra fácilmente con diversas herramientas y bibliotecas populares, como React, TypeScript, Babel, y Webpack. Además, ofrece soporte nativo para CSS y optimización de imágenes, lo que simplifica considerablemente el flujo de trabajo de desarrollo​

* Enrutamiento Automático:

Utiliza un sistema de enrutamiento basado en archivos, donde las rutas se crean automáticamente a partir de la estructura de archivos del proyecto. Esto simplifica la gestión de rutas en comparación con otros frameworks que requieren configuraciones manuales

* Actualizaciones Rápidas y División de Código:

Next.js soporta "Fast Refresh" para una rápida retroalimentación de los cambios y facilita la división de código (code splitting), lo que permite cargar solo el código necesario para cada página, mejorando la eficiencia y rendimiento​

* Seguridad y Escalabilidad:

Las aplicaciones estáticas generadas por Next.js no tienen conexión directa con bases de datos, lo que aumenta la seguridad. Además, su arquitectura facilita la implementación en servicios sin servidor (serverless) como AWS Lambda​

* Comunidad y Soporte:

Aunque la comunidad de Next.js es más pequeña que la de React, está creciendo rápidamente y cuenta con un soporte robusto gracias a la activa participación de desarrolladores y al respaldo de Vercel, que ofrece funcionalidades adicionales como pruebas A/B y renderizado en el borde​

### Comparación con Otros Frameworks

* React por sí solo no ofrece SSR o SSG, lo que puede resultar en tiempos de carga más lentos y peor SEO en comparación con Next.js. Sin embargo, React es más fácil de aprender y tiene una comunidad más grande​
* Gatsby también ofrece generación de sitios estáticos como Next.js, pero está más enfocado en sitios web estáticos y puede no ser tan flexible para aplicaciones web dinámicas. Next.js, con su capacidad SSR, puede manejar una mayor variedad de casos de uso​.
* Angular y Vue ofrecen SSR a través de sus propias soluciones (Angular Universal y Nuxt.js, respectivamente), pero Next.js es generalmente preferido por su simplicidad de configuración y su enfoque en la experiencia del desarrollador​

En resumen, Next.js destaca por su capacidad de renderizado híbrido, facilidad de integración, y un robusto soporte de comunidad y herramientas, lo que lo convierte en una opción muy eficiente para el desarrollo de aplicaciones web modernas.

## MongoDB

### ¿Qué es MongoDb?

MongoDB es una base de datos NoSQL de tipo documental que almacena datos en formato JSON (BSON internamente). A diferencia de las bases de datos relacionales tradicionales, MongoDB no requiere un esquema fijo, lo que lo hace altamente flexible y escalable.

### Ventajas de MongoDB

* Flexibilidad del esquema:

MongoDB permite almacenar datos con estructuras variadas sin necesidad de definir esquemas estrictos de antemano. Esto facilita la adaptación a cambios en los requisitos de datos sin tiempo de inactividad significativo​

* Escalabilidad horizontal:

Gracias a su arquitectura de "sharding", MongoDB puede escalar horizontalmente distribuyendo datos a través de múltiples servidores. Esto permite manejar grandes volúmenes de datos y tráfico sin necesidad de aumentar significativamente el hardware​

Consultas y análisis poderosos:

MongoDB soporta consultas ad hoc que pueden ejecutarse sin necesidad de unir tablas, lo que simplifica el acceso a datos complejos y mejora el rendimiento de las consultas

* Desarrollo rápido y amigable para desarrolladores:

Su modelo de datos basado en documentos es intuitivo y se integra bien con lenguajes de programación modernos, permitiendo a los desarrolladores trabajar de manera más natural con los datos

* Alta disponibilidad y recuperación ante desastres:

MongoDB ofrece características robustas de replicación y recuperación ante fallos, lo que asegura la disponibilidad continua de los datos incluso en casos de fallos de hardware

* Versatilidad en casos de uso:

MongoDB es ideal para aplicaciones web, sistemas de gestión de contenidos, análisis en tiempo real y aplicaciones móviles debido a su capacidad para manejar datos no estructurados y de gran volumen de manera eficiente

* Soporte y comunidad:

Cuenta con una comunidad global activa y soporte empresarial, lo que facilita obtener ayuda y recursos para el desarrollo y mantenimiento de aplicaciones basadas en MongoDB

Estas características hacen de MongoDB una opción potente para proyectos que requieren flexibilidad, escalabilidad y velocidad, especialmente en aplicaciones modernas donde los datos cambian rápidamente y deben ser manejados eficientemente.

## NextAuth.jsNextAuth.Js

### ¿Qué es NextAuthJs?

NextAuth.js es una biblioteca de autenticación para aplicaciones Next.js que proporciona una forma sencilla y flexible de añadir autenticación y autorización a las aplicaciones web modernas.

### Ventajas

• Integración sencilla y rápida:

NextAuth.js se integra de manera fluida con Next.js, lo que permite a los desarrolladores añadir autenticación a sus aplicaciones con rapidez. Con solo unas pocas líneas de código, es posible configurar la autenticación utilizando proveedores comunes como Google, Facebook, GitHub y más.

• Compatibilidad con múltiples proveedores de autenticación:

NextAuth.js soporta una amplia gama de proveedores de autenticación de terceros, así como la autenticación basada en credenciales (correo electrónico y contraseña). Esto proporciona flexibilidad para utilizar distintos métodos de autenticación según las necesidades de la aplicación.

• Autenticación segura y gestión de sesiones:

La biblioteca gestiona de manera segura la autenticación y la gestión de sesiones. Utiliza cookies y tokens para mantener las sesiones de usuario seguras y proporcionar una experiencia de usuario consistente y protegida.

• Soporte para autenticación JWT:

NextAuth.js permite el uso de JSON Web Tokens (JWT) para la autenticación, lo que es ideal para aplicaciones que requieren autenticación sin estado. Los JWT facilitan la gestión de la autenticación en aplicaciones distribuidas y microservicios.

• Configuración flexible:

Ofrece una configuración altamente personalizable, permitiendo a los desarrolladores ajustar el comportamiento de la autenticación según las necesidades específicas de su aplicación. Esto incluye opciones avanzadas como la personalización de páginas de inicio de sesión y la gestión de permisos de acceso.

• Soporte para OAuth 2.0

La biblioteca soporta los estándares de autenticación más comunes, incluyendo OAuth 2.0. Esto asegura que la autenticación sea compatible con una amplia variedad de servicios y proveedores.

Estas características hacen de NextAuth.js una opción robusta y eficiente para añadir autenticación y autorización a aplicaciones Next.js, permitiendo a los desarrolladores centrarse en el desarrollo de las funcionalidades principales de sus aplicaciones sin preocuparse por la complejidad de la gestión de usuarios y sesiones.

## AWS y S3



### ¿Qué es AWS?

Amazon Web Services (AWS) es una plataforma de servicios en la nube ofrecida por Amazon la cual proporciona una infraestructura de TI escalable, confiable y de bajo costo en la nube, que incluye una amplia gama de servicios como almacenamiento, bases de datos, análisis, redes, inteligencia artificial, aprendizaje automático, IoT (Internet de las Cosas), seguridad, y más.

### ¿Para qué sirve AWS?

AWS sirve para permitir a las empresas y desarrolladores desplegar, gestionar y escalar aplicaciones de manera eficiente en la nube. Sus servicios están diseñados para ser flexibles y fáciles de usar algunas de las aplicaciones típicas de AWS son Alojamiento de sitios web y aplicaciones web , Almacenamiento y copias de seguridad, IA, computación sin servidor, aplicaciones webs y móviles …

### ¿Qué es AWS S3?

Amazon Simple Storage Service (S3) es un servicio de almacenamiento en la nube ofrecido por AWS. S3 proporciona almacenamiento escalable, seguro y de alta durabilidad para cualquier cantidad de datos. Los datos se almacenan en "buckets" y pueden ser organizados y gestionados de manera flexible.

### ¿Para qué sirve AWS S3?

Amazon S3 se utiliza para almacenar y proteger cualquier cantidad de datos, desde sitios web y aplicaciones hasta datos de análisis y copias de seguridad. Sus usos comunes incluyen almacenamiento de archivos y multimedia, copias de seguridad y recuperación y Big Data y análisis

### Ventajas de AWS S3

* Durabilidad y disponibilidad:

Amazon S3 ofrece una durabilidad de 99.999999999% y una alta disponibilidad, asegurando que los datos estén siempre accesibles y protegidos contra pérdidas.

* Escalabilidad:

S3 puede escalar automáticamente según las necesidades de almacenamiento, sin límites predefinidos, lo que facilita el crecimiento de las aplicaciones sin preocuparse por la capacidad de almacenamiento.

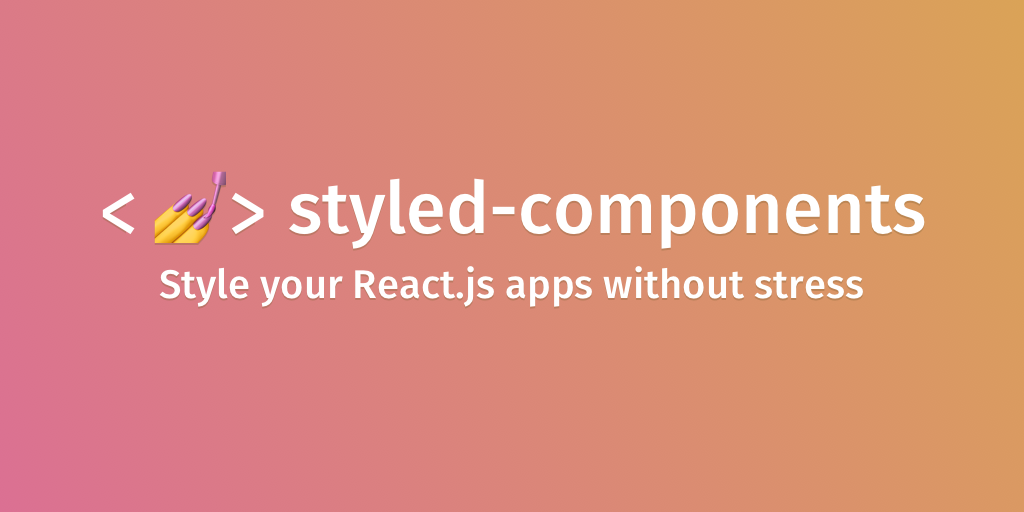
* Seguridad:

S3 proporciona múltiples opciones de seguridad, incluyendo cifrado en tránsito y en reposo, políticas de control de acceso detalladas y auditorías de acceso para asegurar los datos almacenados.

* Costo-efectividad:

Con opciones de almacenamiento por niveles y precios basados en el uso, S3 permite optimizar costos almacenando datos de acuerdo con su frecuencia de acceso.

## Styled-Components

­­­­­

### ¿Qué es Styled-components?

Styled-components es una biblioteca de JavaScript para estilizar componentes en aplicaciones web. Permite escribir estilos CSS dentro de los componentes de React utilizando una sintaxis similar a la de CSS.

### ¿Para qué sirve Styled-components?

Styled-components tiene como objetivo mejorar la organización y mantenibilidad del código CSS en aplicaciones React. Permite definir estilos de forma encapsulada junto con los componentes, lo que facilita la creación de interfaces de usuario coherentes y reutilizables.

### Ventajas de Styled-components:

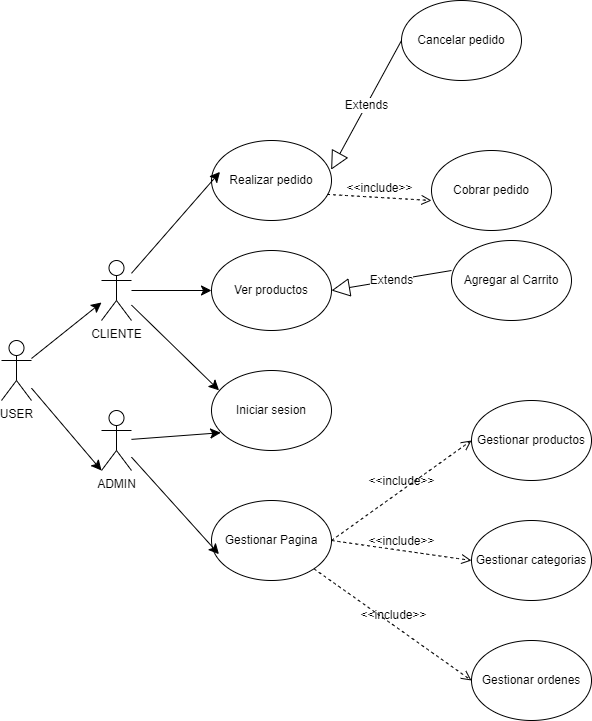
* Encapsulación de estilos: Los estilos definidos con Styled-components están asociados específicamente con los componentes en los que se utilizan, lo que evita los problemas de colisión de estilos y garantiza una mayor modularidad y reutilización del código.
* Facilidad de mantenimiento: Al tener los estilos y los componentes juntos en un mismo lugar, resulta más fácil entender, modificar y mantener el código. Esto es especialmente útil en proyectos grandes y complejos donde la gestión de estilos puede volverse complicada.
* Dinamismo: Styled-components permite la interpolación de propiedades de JavaScript en los estilos CSS, lo que facilita la creación de estilos dinámicos que pueden adaptarse según el estado o las propiedades de los componentes.
* Soporte para CSS-in-JS: Al proporcionar una solución integrada para escribir estilos en aplicaciones React, Styled-components simplifica el flujo de trabajo del desarrollo al eliminar la necesidad de herramientas externas y mejorar la coherencia en el estilo de la aplicación.
* Optimización del rendimiento: Styled-components está diseñado para optimizar el rendimiento de las aplicaciones React al reducir la cantidad de CSS redundante y al permitir la creación de estilos más específicos y eficientes.

­­­

# Desarrollo o Explicación

## Diseño UML de la aplicación web

El primer paso a la hora del diseño del proyecto ha sido elaborar diferentes diagramas UML con el objetivo de visualizar y contextualizar la implementación en lo máximo posible. Se incluye a continuación un diagrama de casos de usos que nos permite apreciar mas a fondo las acciones de los usuarios de la web





Para crear el diagrama de flujo usando Mermaid.js, se ha realizado el siguiente proceso:

* Inicio de Sesión: El diagrama comienza con el nodo "Iniciar Sesión".
* Decisión de Rol: Se evalúa si el usuario es administrador o cliente.
* Flujo de Administrador: Si el usuario es administrador, se dirige a la página de gestión donde puede:
* Gestionar productos: agregar, editar o eliminar.
* Gestionar órdenes: agregar, editar o eliminar.
* Gestionar categorías: agregar, editar o eliminar.
* Flujo de Cliente: Si el usuario es cliente, se le lleva a la página donde puede:
  + Buscar y seleccionar productos.
  + Verificar disponibilidad y agregar al carrito.
  + Verificar carrito, realizar pago y confirmar pedido.

Aquí muestro el código que se usó en Mermaid.Js para representar el diagrama de flujos

**flowchart** TD

    Inicio(Iniciar Sesión) **-->** |¿Es Admin?| DecisionAdmin{¿Es Admin?}

    DecisionAdmin **-->** |Sí| Gestion(Administrar Sitio)

    DecisionAdmin **-->** |No| ECommerce(Ecommerce)

    Gestion **-->** |Gestionar Productos| Productos

    Gestion **-->** |Gestionar Órdenes| Ordenes

    Gestion **-->** |Gestionar Categorías| Categorias

    Productos **-->** |Agregar Producto| AgregarProducto

    Productos **-->** |Editar Producto| EditarProducto

    Productos **-->** |Eliminar Producto| EliminarProducto

    Ordenes **-->** |Agregar Orden| AgregarOrden

    Ordenes **-->** |Editar Orden| EditarOrden

    Ordenes **-->** |Eliminar Orden| EliminarOrden

    Categorias **-->** |Agregar Categoría| AgregarCategoria

    Categorias **-->** |Editar Categoría| EditarCategoria

    Categorias **-->** |Eliminar Categoría| EliminarCategoria

    ECommerce **-->** BuscarProducto[Buscar Producto]

    BuscarProducto **-->** SeleccionarProducto[Seleccionar Producto]

    SeleccionarProducto **-->** DecisionDisponible{¿Producto Disponible?}

    DecisionDisponible **-->** |Sí| AgregarCarrito[Agregar al Carrito]

    DecisionDisponible **-->** |No| MensajeNo[Mostrar Mensaje No Disponible]

    MensajeNo **-->** BuscarProducto

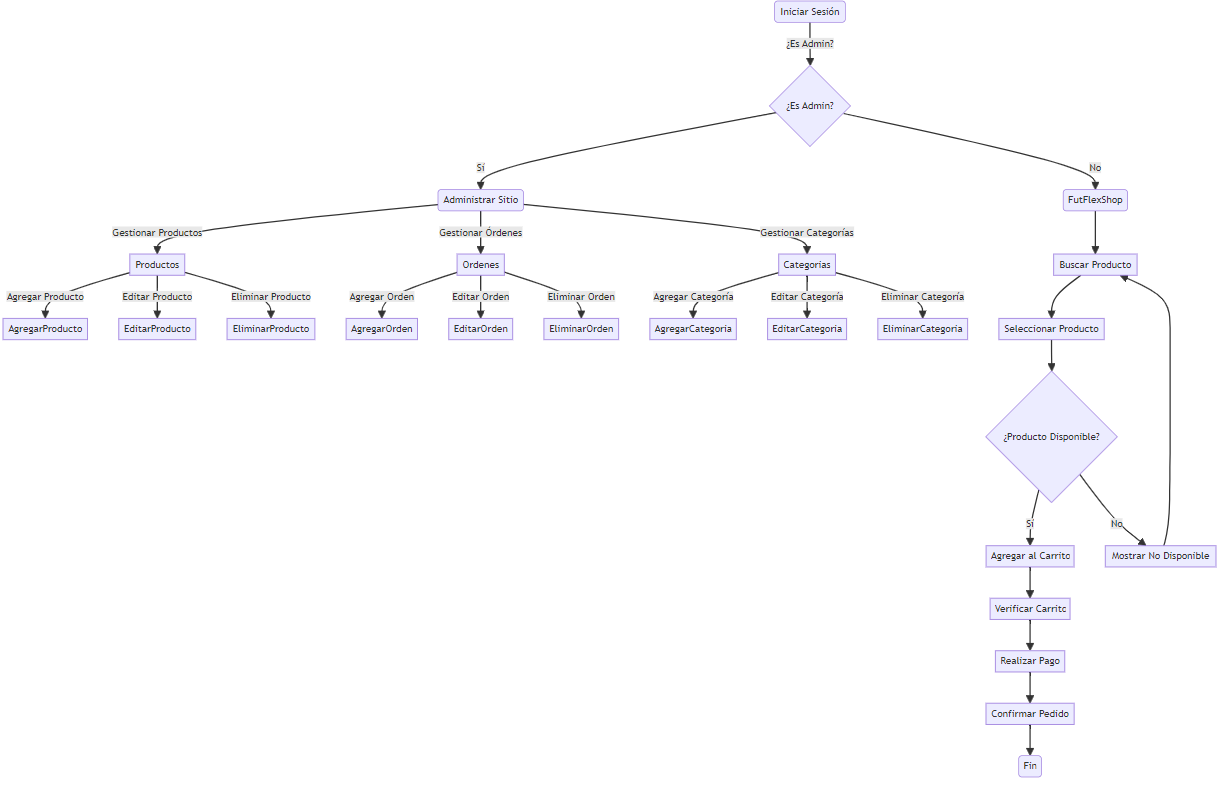
    AgregarCarrito **-->** VerificarCarrito[Verificar Carrito]

    VerificarCarrito **-->** RealizarPago[Realizar Pago]

    RealizarPago **-->** ConfirmarPedido[Confirmar Pedido]

    ConfirmarPedido **-->** Fin(Fin)

A continuación, muestro la imagen generada:



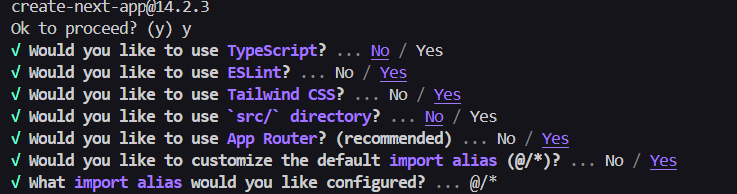
## Desarrollo e implementación del proyecto

### Configuración inicial y creación del proyecto en NextJs

Para iniciar el proyecto las dos cosas que se deben tener en cuenta es tener Node.Js y NPM instalados en el sistema. Abriremos la terminal (es preferible usar la terminal del sistema a la que incorporan algunos IDE como Visual Studio Code) y ejecutaremos los siguientes comandos:

npx create-next-app nombre-del-proyecto (futflexshop-admin y futflexshop)

cd nombre-del-proyecto

Al iniciar el primer comando nos ira mostrando en la terminal ciertas preguntas para la configuración del proyecto:  


* **Would you like to use TypeScript?** No  
  TypeScript es un lenguaje similar a JavaScript que añade tipos estáticos. En este proyecto, optamos por no usar TypeScript ya que no incorporaremos ninguna característica específica de este lenguaje.
* **Would you like to use ESLint?** Yes  
  ESLint es una herramienta de análisis estático que ayuda a identificar y corregir errores en el código. Optamos por incluirlo para mantener la calidad y coherencia del código.
* **Would you like to use Tailwind CSS?** Yes  
  Tailwind CSS es un framework de CSS que permite una manera rápida y eficiente de crear diseños atractivos y funcionales. Lo incluimos para facilitar el diseño y la maquetación de nuestra aplicación.
* **Would you like to use** src/ **directory?** Yes  
  Usar un directorio src/ para organizar el código fuente puede mejorar la estructura y claridad del proyecto.
* **Would you like to use App Router?** Yes  
  El App Router es la manera recomendada de manejar la navegación en una aplicación Next.js.
* **Would you like to customize the default import alias (**@/\***)?** Yes  
  Configurar alias de importación permite acortar y simplificar las rutas de los módulos importados, lo cual facilita el mantenimiento del código.

A continuación, creara el proyecto y tendremos esto:

/Ecommerce-admin

├──. next

├── node\_modules

├── public

├── pages

├── styles

├── .eslintrc.json

├── .gitignore

├── jsconfig.json

├── next.config.mjs

├── package-lock.json

├── package.json

├── postcss.config.js

├── postcss.config.mjs

├── tailwind.config.js

En este esquema:

/ecommerce-admin es la carpeta raíz de tu proyecto.

Next y node\_modules son carpetas que generalmente se generan automáticamente.

Public y styles son carpetas donde puedes guardar tus recursos públicos y estilos, respectivamente.

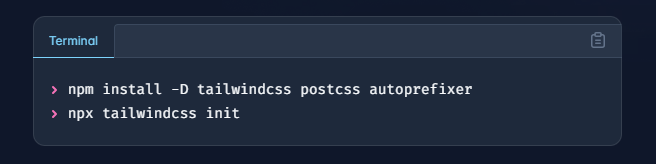
Pages es la parte fundamental de este proyecto donde almacenaremos rutas automáticas, api routes etc.

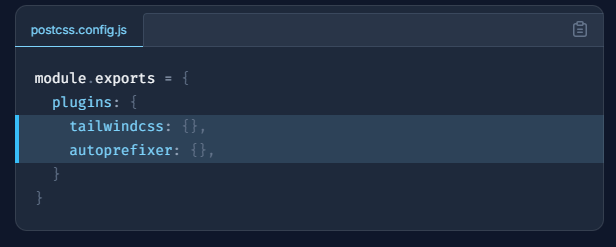
.eslintrc.json, .gitignore, jsconfig.json, next.config.mjs, package-lock.json, package.json, postcss.config.js, postcss.config.mjs, y tailwind.config.js son archivos de configuración y se ubican en la raíz del proyecto.

Después de la creación de la estructura inicial configuraremos Tailwind css siguiendo los siguientes pasos

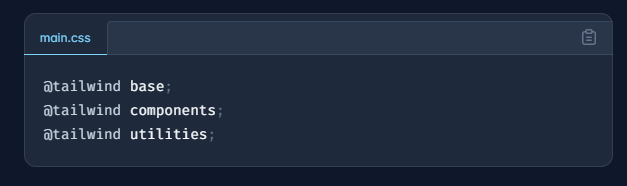
Ingresar en [tailwindcss.com](https://tailwindcss.com) y buscar la sección de instalación usando PostCSS.

Instalar Tailwind CSS mediante NPM con el siguiente comando:



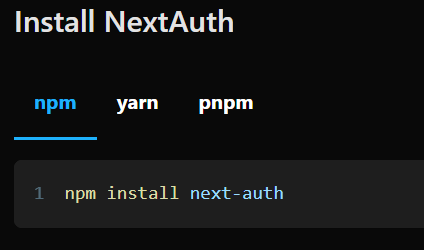
Añadir Tailwind a la configuración de PostCSS creando un archivo postcss.config.js en la raíz del proyecto con el siguiente contenido:  


En la carpeta styles, crear o editar el archivo global.css y añadir lo siguiente

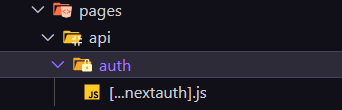


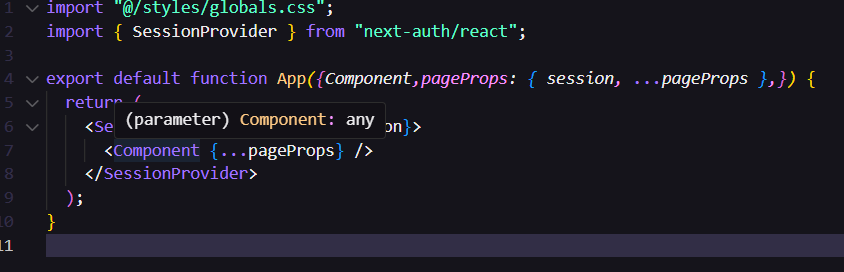
Después de realizar esto, se comenzará con la primera parte del proyecto, generando la página de inicio de sesión.

Se inicia instalando la dependencia de NextAuth.



Luego, se ingresa en la carpeta pages/api y se crea la carpeta auth con un archivo [...nextauth].js donde se pegará el código que permite las rutas dinámicas para NextAuth junto con su configuración. En el apartado de providers, se utilizará el proveedor Google para la identificación del administrador.



En el archivo \_app.js, se importará la siguiente línea de código: SessionProvider, que envuelve la aplicación, permitiendo que cualquier componente dentro de ella pueda acceder al estado de la sesión de autenticación usando el hook useSession.

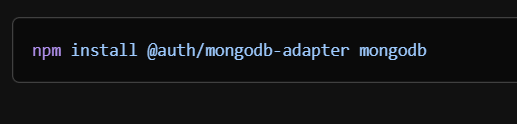
Ahora se ingresará a la carpeta index.js y se creará el siguiente código que se encargará de renderizar la página principal del proyecto, mostrando un mensaje de bienvenida al usuario autenticado. El componente Home, obtiene la sesión actual utilizando el hook useSession. El componente Layout envuelve el contenido principal de la página. En el contenido principal, se muestra un mensaje de bienvenida que incluye el nombre del usuario autenticado. También se muestra una imagen y el nombre del usuario dentro de un contenedor estilizado.

Ahora se realiza la conexión con MongoDB. Se ingresa en la página de MongoDB Atlas y se registra una cuenta. Una vez registrado, se crea un nuevo clúster en MongoDB Atlas y se obtiene la URL de conexión.

A continuación, se ingresa nuevamente en la página de NextAuth y se dirige al apartado de Adapter. Se busca MongoDB y se siguen las instrucciones proporcionadas. Aquí se describe cómo configurar el adaptador de MongoDB para NextAuth:

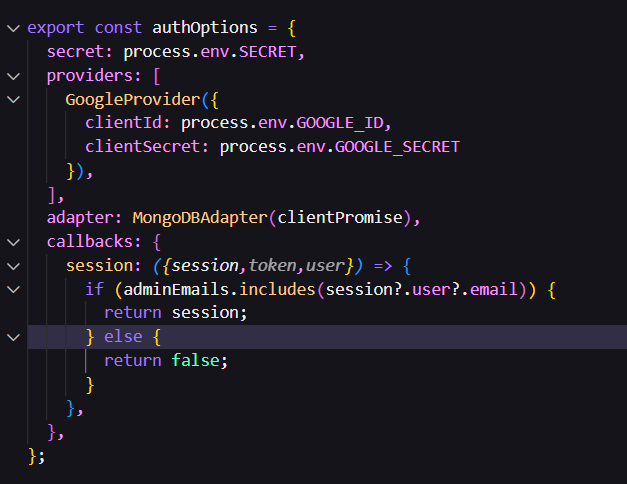
1. **Instalar dependencias**

Se instala las dependencias necesarias mediante :



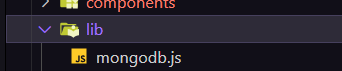
1. **Configurar el adaptador de MongoDB:**

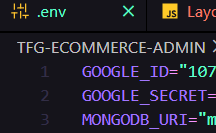
En el archivo [...nextauth].js, se configura NextAuth para utilizar el adaptador de MongoDB.



1. **Configurar la conexión a MongoDB**

Se crea un archivo mongodb.js en la carpeta lib para manejar la conexión a MongoDB. Este archivo utiliza mongodb para conectarse a la base de datos.



1. **Configurar variables de entorno**

Se crea un .env donde almacenaremos las credenciales tanto de Google como las de MongoDb

Al seguir estos pasos, se configura NextAuth para utilizar MongoDB como su adaptador, permitiendo gestionar sesiones de usuarios y otra información de autenticación directamente en una base de datos MongoDB

## Estructura del Proyecto

└─ 📁ECOMMERCE-ADMIN

   └── .env

   └── .eslintrc.json

   └── .gitignore

  └── 📁components

       └── IconBlocks.js

      └── Layout.js

       └── Logo.js

       └── Nav.js

      └── ProductForm.js

       └── Spinner.js

   └── jsconfig.json

  └── 📁lib

       └── mongodb.js

      └── mongoose.js

└── next.config.mjs

    └── package-lock.json

    └── package.json

    └── 📁pages

        └── 📁api

            └── 📁auth

                └── [...nextauth].js

            └── categories.js

            └── hello.js

            └── orders.js

            └── products.js

            └── upload.js

        └── categories.js

        └── index.js

        └── orders.js

        └── 📁products

            └── 📁delete

                └── [...id].js

            └── 📁edit

                └── [...id].js

            └── new.js

        └── products.js

        └── \_app.js

        └── \_document.js

**.env**: Archivo de configuración para variables de entorno donde se almacenan información sensible como claves de api credenciales y configuraciones de entorno.

**.eslintrc.json**: Configuración de ESLint para mantener la calidad del código definiendo reglas y estándares de codificación para garantizar un código limpio

**gitignore**: Lista de archivos y carpetas que Git debe ignorar.

**Carpeta components:**

**IconBlocks.js**: Componente que renderiza íconos de Productos, Categorías y Órdenes

**Layout.js**: Layout principal del sitio, gestiona la autenticación de usuario y la visualización del menú de navegación.

Si el usuario no esta autenticado muestra un botón de login con Google

**Logo.js**: Componente que muestra el logo y enlaza a la página principal

**Nav.js**: Es la barra de navegación lateral con enlaces a Inicio, Productos, Categorías y Órdenes, y un botón para cerrar sesión.

**ProductForm.js**: Formulario para crear y editar productos, con manejo de categorías, imágenes y propiedades específicas del producto. Utiliza useEffect para cargar las categorías disponibles desde la API y maneja el estado local del formulario con useState además que facilita la gestión y edición de productos de manera eficiente

**Spinner.js**: Componente que muestra un indicador de carga usando el spinner BarLoader de la librería React-Spinners

**-Carpeta lib**

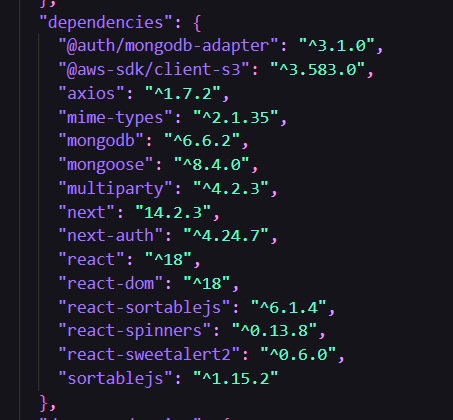
**mongodb.js**: Configura la conexión con la base de datos MongoDB utilizando el cliente de MongoDB. Proporciona una promesa desde el cliente de MongoDB que puede ser utilizada a lo largo de la aplicación para acceder a la base de datos

**mongoose.js**: Establece la configuración y los modelos de datos para Mongoose Facilita la interacción con la base de datos MongoDB mediante la definición de esquemas.

**jsconfig.json**: Configuración del proyecto para el soporte de JavaScript

**next.config.mjs**: Configuración específica para proyectos Next.js las cuales definen las configuraciones personalizadas para la compilación y el comportamiento de la aplicación Next.js.

**package-lock.json y package.json**: Archivos de configuración de npm que definen las dependencias del proyecto. Estas son las dependencias usadas en este proyecto



**Carpeta Pages**

**Carpeta Api:** La carpeta api en este proyecto se utiliza para definir rutas de api que permiten crear endpoints que pueden ser llamados desde el cliente o el servidor. Estos endpoints se utilizan para manejar operaciones backend como acceso a mongodb la autenticación del usuario la manipulación de los datos etc.

**[...nextauth].js**: Archivo que maneja la autenticación con NextAuth.js. Configura y gestiona los proveedores de autenticación, las sesiones

**categories.js**: Archivo que gestiona la creación, edición y eliminación de categorías en la aplicación. Utiliza axios (se utiliza para hacer solicitudes HTTP desde el navegador y Node.js. Es muy popular debido a su simplicidad y poder para gestionar peticiones asíncronas) para realizar llamadas a la API y react-sweetalert2 para confirmar acciones de eliminación.

UseState: Se utiliza para manejar el estado de las categorias nombres y propiedades

UseEffect : Se utiliza para cargar las categorias cuando se monta el componente

**Funciones principales**

* **fetchCategories:** Obtiene las categorías desde la API
* **saveCategory:** Guarda o actualiza una categoría en la base de datos
* **editCategory:** Rellena el formulario con los datos de una categoría existente para su edición
* **deleteCategory:** Elimina una categoría después de la confirmación del usuario

**products.js:** Archivo que gestiona la creación, edición y eliminación de productos en la aplicación

**Importaciones**:

* Layout
* Axios
* Link
* useEffect y useState

**Funciones principales**

**useEffect** : Llama a axios.get("/api/products") al llamar a este componente para obtener los productos del api y los almacena en el array products.

La tabla de productos contiene un enlace a /products/new que permite añadir nuevos productos .Por otro lado cada fila de producto contiene dos botones que enlazan con /products/edit/{product.\_id} y /products/delete/{product.\_id} que permite editar y borrar el producto

**Carpeta Delete**

Este archivo proporciona la interfaz que permite la confirmación para la eliminación de un producto en específico además de mostrar la información del producto a eliminar, y por otro lado ejecutar la llamada al api.

**Funciones principales**

**goBack:** Redirige a la página de productos

**deleteProduct:** Hace una solicitud DELETE a la API para eliminar el producto y luego llama a goBack para redirigir al usuario.

**Carpeta Edit**

Este archivo obtiene la información mediante el id del producto a editar y lo muestra en el formulario para su edición. Si el id esta presente se hace una solicitud GET al api y almacena el producto editado

└── 📁TFG-ECOMMERCE-Front

└── 📁components

└── Button.js

└── ButtonLink.js

└── CartContext.js

└── Center.js

└── Featured.js

└── Header.js

└── 📁icons

└── Bars.js

└── CartIcon.js

└── Input.js

└── NewProducts.js

└── ProductBox.js

└── ProductImages.js

└── ProductsGrid.js

└── Table.js

└── Title.js

└── WhiteBox.js

└── 📁lib

└── colors.js

└── mongoose.js

└── 📁models

└── Order.js

└── Product.js

└── 📁pages

└── 📁api

└── cart.js

└── checkout.js

└── hello.js

└── webhook.js

└── cart.js

└── index.js

└── 📁product

└── [id].js

└── products.js

└── \_app.js

└── \_document.js

Raíz del Proyecto

* 📁Ecommerce-Front: Carpeta principal del proyecto Frontend de tu e-commerce.
* 📁components
  + Button.js: Componente de botón reutilizable.
  + ButtonLink.js: Componente de botón que funciona como enlace.
  + CartContext.js: Contexto para manejar el estado del carrito de compras.
  + Center.js: Componente para centrar contenido.
  + Featured.js: Componente para mostrar productos destacados.
  + Header.js: Componente de cabecera/navegación del sitio.
* 📁icons: Carpeta que contiene iconos usados en el proyecto.
  + Bars.js: Icono de barras (probablemente para un menú).
  + CartIcon.js: Icono del carrito de compras.
  + Input.js: Componente reutilizable para campos de entrada.
  + NewProducts.js: Componente para mostrar nuevos productos.
  + ProductBox.js: Componente para mostrar información de un producto en forma de caja.
  + ProductImages.js: Componente para mostrar imágenes de productos.
  + ProductsGrid.js: Componente para mostrar una cuadrícula de productos.
  + Table.js: Componente de tabla reutilizable.
  + Title.js: Componente para mostrar títulos.
  + WhiteBox.js: Componente para contenedores con fondo blanco.
* 📁lib
  + colors.js: Archivo de configuración para definir colores usados en el proyecto.

mongoose.js: Archivo de configuración para conectar y configurar Mongoose (ORM para MongoDB).

* 📁models
  + Order.js: Modelo de datos para las órdenes.
  + Product.js: Modelo de datos para los productos.
* 📁pages
* 📁api: Carpeta que contiene endpoints API del backend.
  + cart.js: Endpoint para manejar el carrito de compras.
  + checkout.js: Endpoint para manejar el proceso de pago.
  + hello.js: Endpoint de prueba simple (probablemente para verificar que el API funciona).
  + webhook.js: Endpoint para manejar webhooks (posiblemente para integraciones como pagos).
  + cart.js: Página del carrito de compras.
  + index.js: Página principal del sitio.
* 📁product
  + [id].js: Página dinámica para mostrar los detalles de un producto específico basado en su ID.
  + products.js: Página que muestra todos los productos.
  + \_app.js: Componente principal que envuelve todas las páginas, utilizado para inicializar la aplicación.

# Conclusión y Posibles Mejoras

## Resumen de Resultados:

A lo largo de este proyecto, he comprendido mejor el funcionamiento de los ecommerce modernos. Ha sido una experiencia enriquecedora que me ha permitido aplicar conocimientos teóricos en un entorno real. El proyecto ha culminado exitosamente con la creación de una plataforma que ofrece a los clientes una experiencia completa. Por otro lado, los administradores tienen la capacidad de gestionar productos, órdenes y categorías a través de una interfaz intuitiva y amigable.

Además, este proyecto me ha aportado experiencia real y valiosa en el desarrollo de ecommerce. He tenido la oportunidad de recibir consejos de profesionales experimentados en este campo, lo cual ha sido invaluable. Estos consejos han abarcado desde mejores prácticas en la gestión de inventarios y optimización de la experiencia de usuario, hasta estrategias efectivas de marketing digital y gestión de datos. Trabajar en este proyecto me ha permitido enfrentar y resolver desafíos complejos, mejorando así mis habilidades técnicas y de resolución de problemas.

En resumen, este proyecto ha sido una plataforma para aplicar y expandir mis conocimientos, proporcionándome una comprensión profunda de los aspectos técnicos y estratégicos de los ecommerce modernos. Me siento más preparado para enfrentar futuros desafíos en este campo y contribuir a la creación de soluciones innovadoras y eficientes.

## Posibles Mejoras:

Para el desarrollo de este proyecto se propone la implementación de diversas mejoras y funcionalidades adicionales en la aplicación. Estas mejoras están orientadas a optimizar la experiencia del usuario y mejorar la eficiencia operativa de la aplicación. A continuación, se detallan las funcionalidades seleccionadas:

* Mejoras de Diseño
* Mejor Velocidad de Carga
* Apartado de Cuenta para Modificar Datos Personales
* Actualización de datos personales
* Cambio de contraseña
* Gestión de métodos de pago y direcciones de envío
* Página de Seguimiento de Pedidos
  + Actualizaciones en tiempo real
* Proceso para el Registro y el Inicio de Sesión

# Bibliografía y Referencias

Documentacion Componentes <https://nextui.org/>

Documentacion Componentes <https://ui.shadcn.com/>

Documentacion Componentes <https://mui.com/>

Documentacion oficial de NextJs <https://nextjs.org/>

Documentacion oficial de React <https://react.dev/>

Documentacion NextAuth <https://next-auth.js.org/>

Documentacion AWS <https://aws.amazon.com/>

Documentacion oficial de Mongo <https://www.mongodb.com/>

Documentacion Componentes <https://tailwindui.com/>

Documentacion StyledComponets <https://styled-components.com/>

Documentacion Tailwind Component <https://tailadmin.com/#components>