PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

SISTEMA DE CONTROL DE STOCK ATLAS GYM

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autores: Rachel Maria Arias Paneque & Alvaro Isaac Mullo Guerrero

Tutor: Patricio Alvear

Quito, Ecuador

2024

Dedicatoria

Dedico esta tesis a mi familia, personas las cuales amo mucho, son mi soporte y que siempre han estado ahí para mí en este camino, a mi madre Janth, por apoyarme y brindarme mentoría, a mi hermano Miguel, por creer en mí y darme la oportunidad de estudiar y aprender, a mi hermana Katherine, por siempre creer en mí y apoyarme, a mi pareja Rachel, por recorrer este largo camino conmigo desde el principio, lleno de dificultades y altibajos, pero siempre siendo el apoyo el uno del otro, logrando sobresalir y conseguir esta meta juntos, y a mi Sol, Cleopatra y difunto General, mis compañeros gatunos que también estuvieron ahí conmigo en las noches de desvelo estudiando y/o programando.

Esta tesis es dedicada a mi madre Yanet, por ser mi apoyo y una persona muy fuerte, mi abuela Paulina, por su amor incondicional y su apoyo estando a kilómetros de distancia, mi padre Jorge, por sus conejos y su ayuda, ya que sin él no hubiera podido permitirme los estudios, mi mejor amiga Sara y mi pareja y gran compañero de vida Isaac, por estar desde un inicio conmigo apoyándome y alentándome a seguir y a mejorar. Y dedicárselas también a mi compañera canina Kira, mi motivación también y parte de mi vida. Aunque solo sean unas simples dedicatorias de un documento, para mi es muy especial, porque indirecta o directamente se han involucrado en mi progreso, me han dado ese apoyo y fuerzas para seguir motivada día a día. Gracias.

Tabla de contenidos

Lista de tablas

Lista de figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, **ALVARO ISAAC MULLO GUERRERO** con C.I. 1755290150 autor(a) del trabajo de titulación intitulado: **"SISTEMA DE CONTROL DE STOCK ATLAS GYM"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual l de Universidad.

Quito, 31 de Julio 2024

Alvaro Isaac Mullo Guerrero

C.I. 1755290150

DECLARACIÓN y AUTORIZACIÓN

Yo, **Rachel Maria Arias Panque** con C.I. 1756382477 autor(a) del trabajo de titulación intitulado: **"SISTEMA DE CONTROL DE STOCK ATLAS GYM"**, previa a la obtención del título de **Tecnólogo Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC autor(a) del trabajo de titulación intitulado: **"SISTEMA DE CONTROL DE STOCK ATLAS GYM"**, previa a la obtención del título de **Tecnóloga Superior en Desarrollo de Software** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 31 de Julio 2024

Rachel Maria Arias Panque

C.I. 1756382477

Agradecimientos

Agradezco profundamente a mi asesor de tesis, el Ing. Patricio Alvear, por su paciencia, guía, enseñanzas y valiosos comentarios que me han ayudado al desarrollo de mi proyecto y documento de tesis.

Quiero expresar mi gratitud a la Pontificia Universidad Católica del Ecuador por proporcionar los recursos tanto para el desarrollo del proyecto como para el documento de tesis.

También quiero agradecer a mi familia por estar siempre ahí, apoyándome y ayudándome con sus enseñanzas y experiencias, así como por su amor en estos 2 años de estudio.

Agradezco a la universidad y a sus valiosos recursos que me han proporcionado estos 2 años académicos. A todos los profesores a lo largo de esta trayectoria, por sus enseñanzas que van más allá del aula y nos han dejado lecciones de vida muy valiosas. Agradecerle especialmente a la Ing. Diana Molina por preocuparse y ayudarnos de todas las maneras posibles. Agradecer a nuestro tutor de tesis, el Ing. Patricio Alvear, por su guía constante, su paciencia y su compromiso en cada etapa de este proceso. Y sobre todo a mi familia que siempre ha estado ahí para apoyarme y darme fuerzas.

Introducción

Esta tesis presenta una solución a una problemática que ha sido hallada en la administración de un gimnasio llamado Atlas Gym. El proyecto surge en base a la necesidad del dueño del gimnasio, ya que necesitaba de un sistema de control de Stock para gestionar el inventario en su local.

Los principales requerimientos que el dueño solicitaba que el sistema cumpliera, era el permitir registrar productos con su respectivo stock, registrar las ventas diarias y que sus empleados también pudieran acceder al sistema, pero con cierta regulación; es decir reducirles el acceso solo al apartado de ventas.

Para ello ha sido necesario dividir el proyecto en dos partes complementarias, el Frontend y el Backend. Para el desarrollo del Frontend se empleó React con JavaScript y como principal framework para los elementos visuales "MaterialUi" Por otro lado para el desarrollo del Backend se utilizó como principal framework Django con Python, de la mano con la metodología MVC (Modelo, Controlador, Vista). Este sistema automatizará y asegurará la gestión del inventario de Atlas-Gym, reemplazando las hojas de cálculo, mejorando la administración y ahorrando tiempo valioso para las operaciones diarias.

Capítulo I

Levantamiento de Requisitos y Diseño del Sistema Frontend

Funcionalidades

Funcionalidad N°1: Registro productos

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F1	Registro producto	29/04/2024	13/05/2024	En esta pantalla se registrará el stock de los productos, también contara con un botón de "agregar producto" y se tendrán los detalles de: -Producto, Precio, Stock y Total. También cada producto contara con 2 botones, uno para eliminar y otro para editar.	Alta
	Entrada	Proceso		Salida	
	Botón de Agregar producto	Se abre un modal para ingresar Producto, Precio y Stock		El producto se agrega a la lista de productos	
	Botón de eliminar producto	Se abre un mensaje de confirmación si es que desea eliminar el producto		El producto se elimina a la lista de productos	
	Botón de editar producto	Se abre un modal para editar Precio y Stock		Los detalles del producto se editan	

Tabla 1: Registro producto frontend

Funcionalidad N°2: Venta producto

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F2	Venta producto	13/05/2024	27/05/2024	En esta pantalla se realizarán las ventas de los productos, habrá un botón de "realizar venta" se desplegarán todos los productos registrados y cada uno tendrá 2 campos: -La cantidad del producto vendido.	Alta

				-El responsable de la venta.	
	Entrada	Proceso		Salida	
	Botón de realizar venta	Se abre un modal para confirmar si se quiere realizar la venta		Se realiza la venta, reduciendo la cantidad de stock de cada producto y quedando registrada.	

Tabla 2: Venta producto frontend

Funcionalidad N°3: Historial de Ventas

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F3	Historial de ventas	27/05/2024	10/06/2024	En esta pantalla se observará el historial de las ventas, con los detalles: -Fecha -Producto -Cantidad -Responsable	Alta
	Entrada	Proceso		Salida	
	Datos de la base de datos	Se cargarán los datos n la tabla de la página		No aplica.	

Tabla 3: Historial de ventas frontend

Funcionalidad N°4: Login

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F4	Login	10/06/2024	24/06/2024	Aquí se realizará la identificación para ingresar al sistema, contará con 2 campos: -Ingresar usuario -Ingresar contraseña	Alta
	Entrada	Proceso		Salida	
	Botón de Login	Se realizará la verificación de las credenciales		Se iniciará sesión en caso de que las credenciales estén correctas, existirán 2 usuarios, admin y guest.	

Tabla 4: Login frontend

Mockups

Para el diseño (Mockup) del sistema utilizamos una herramienta llamada Figma.

Distribuidos de la siguiente manera:

Pantalla N°1: Control Stock

Se observarán los productos registrados, se podrán agregar, eliminar y editar.

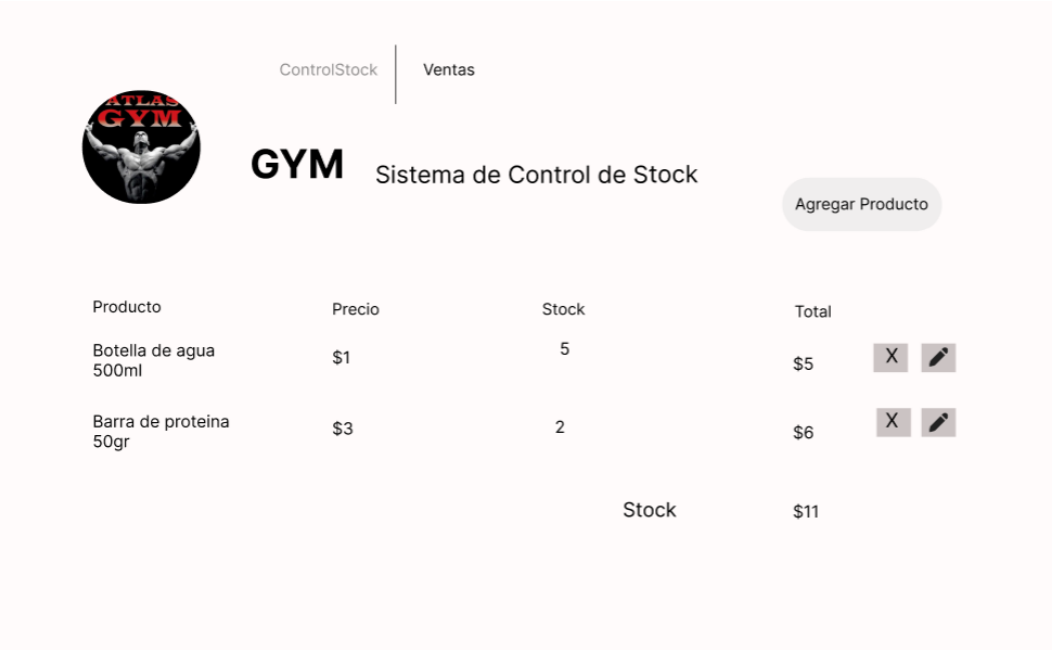

Ilustración 1: Control stock

Fuente: Mullo, A.

- Modal N°1: Registrar productos.

  Se agregarán los detalles del producto.

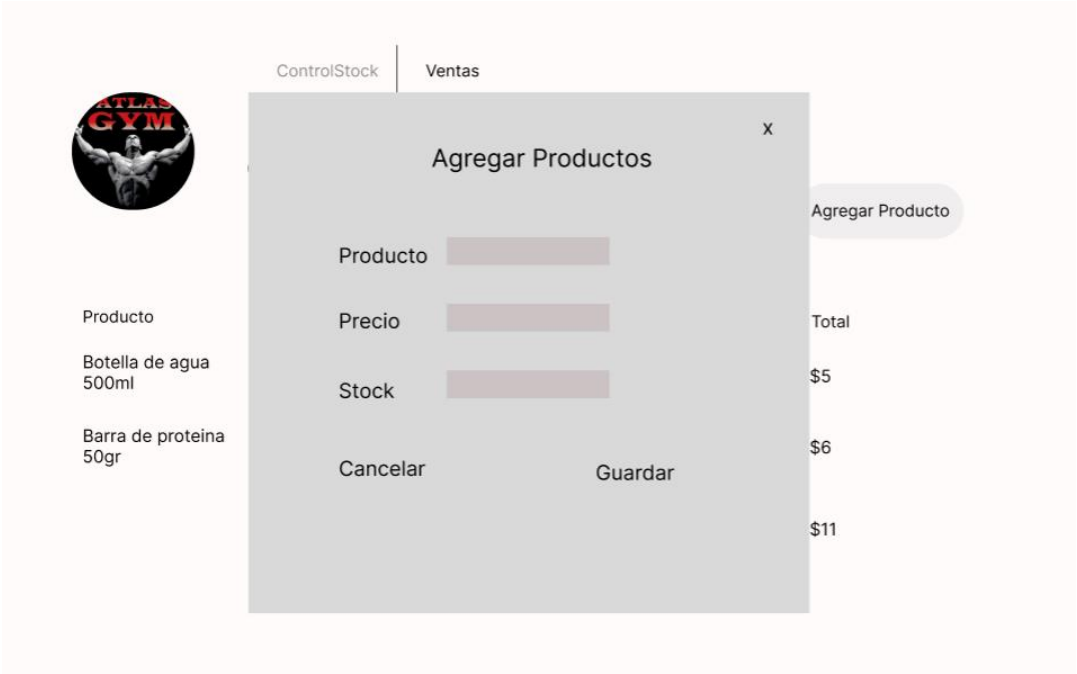

Ilustración 2: Agregar producto

Fuente: Mullo, A.

- Modal N°2: Editar productos.

  Se editarán los detalles del producto

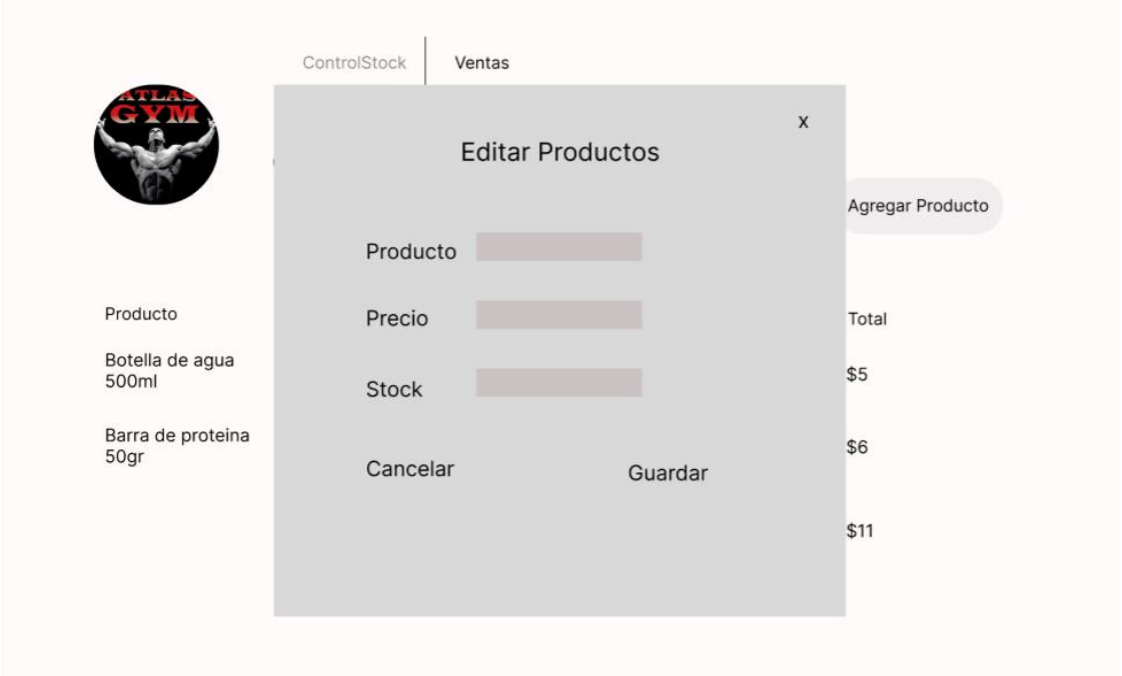

Ilustración 3: Editar producto

F Fuente: Mullo, A.

- Modal N°3: Confirmación para eliminar producto.

  Confirmar o cancelar la eliminación de un producto.

Ilustración 4: Confirmación para eliminar producto

Fuente: Mullo, A.

Pantalla N°2: Realizar Ventas

Realizar la venta del producto, con su cantidad y responsable

Ilustración 5: Realizar ventas

Fuente: Mullo, A.

Pantalla N°3: Historial de Ventas

Visualizar las ventas realizadas con fecha, producto, cantidad y responsable

Ilustración 6: Historial de ventas

Fuente: Mullo, A.

Pantalla N°4: Login

Se realizará el inicio de sesión.

Ilustración 7: Login

Fuente: Mullo, A.

Casos de uso

Los casos de uso son los siguientes:

N°1 Caso de uso realizar venta

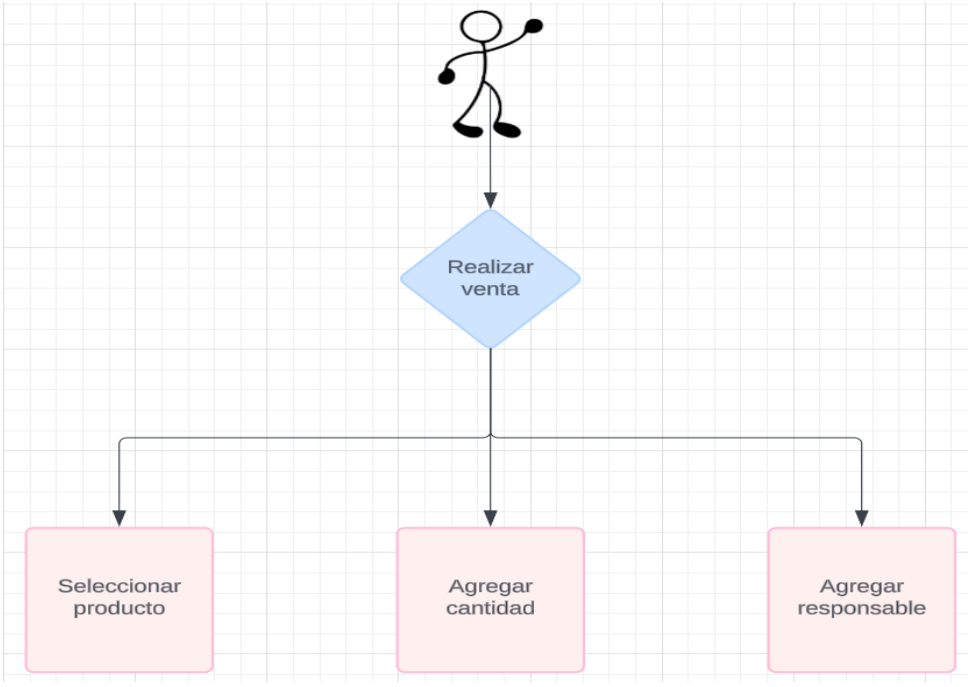

Ilustración 8: Caso de uso 1

Fuente: Mullo, A.

N°2 Caso de uso realizar venta

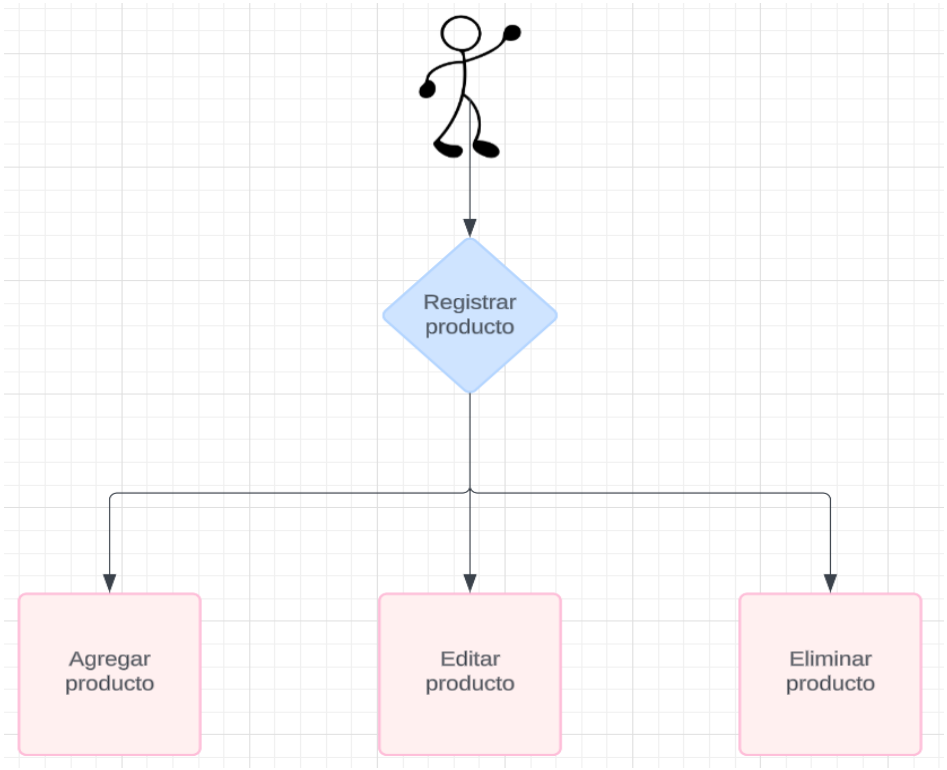

Ilustración 9: Caso de uso 2

Fuente: Mullo, A.

N°3 Caso de uso login

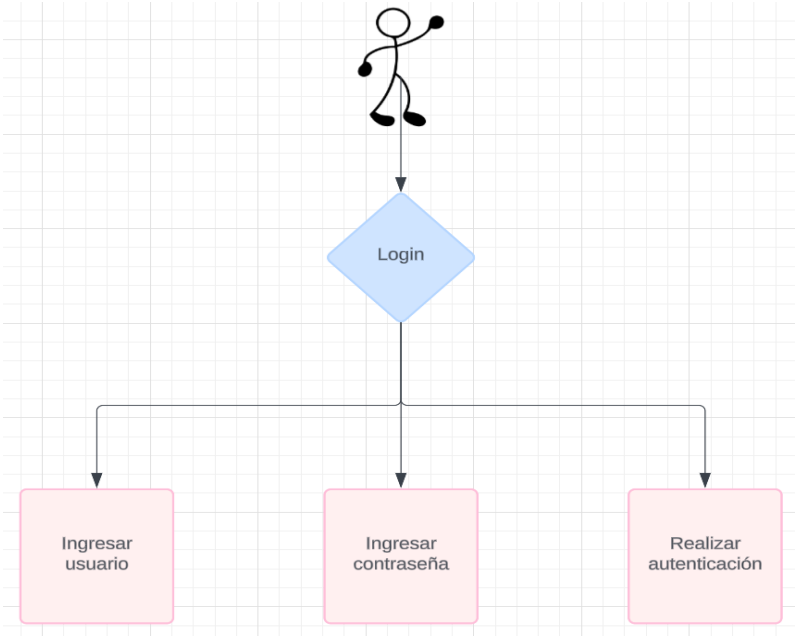

Ilustración 10: Caso de uso 3

Fuente: Mullo, A.

Levantamiento de Requisitos y Diseño del Sistema Backend

Funcionalidades

Funcionalidad N°1: Registrar productos

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F1	Registrar producto	29/04/2024	13/05/2024	La funcionalidad "Registrar producto" permite la captura de datos de productos desde el frontend, los valida y almacena en la base de datos, y finalmente devuelve un mensaje de confirmación o error al usuario.	Alta
	Entrada	Proceso		Salida	
	Datos del producto desde el Frontend.	Validación y almacenamiento en la base de datos.		Mensaje de confirmación o error hacia el frontend.	

Tabla 5: Registrar producto

Funcionalidad 2: Registrar ventas

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F2	Venta producto	13/05/2024	27/05/2024	Se gestiona los datos de las ventas y sus detalles desde el frontend, actualiza el stock y registra la transacción en la base de datos, devolviendo un mensaje de confirmación de la venta y el estado actualizado del stock al frontend.	Alta
	Entrada	Proceso		Salida	
	Se recogen los datos de la venta y sus respectivos detalles	Actualización del Stock y registro en la base de datos.		Confirmación de la venta y actualización del estado del stock, mensaje hacia el frontend.	

Tabla 6: Venta producto

Funcionalidad 3: Historial de Ventas

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F3	Historial de venta	27/05/2024	10/06/2024	Esto permite a los usuarios del frontend solicitar y consultar el historial de ventas, recuperando los datos desde la base de datos y enviándolos al frontend para su visualización y análisis detallado.	Alta
	Entrada	Proceso		Salida	
	Solicitud desde el frontend para consultar el historial de ventas.	Búsqueda y recuperación de datos de ventas almacenados en la base de datos.		Envío de los datos de ventas al frontend para su visualización y análisis.	

Tabla 7: Historial de ventas

Funcionalidad 4: Log in

No	Nombre de actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F4	Login	10/06/2024	24/06/2024	Se valida las credenciales en la base de datos ingresadas desde el frontend, verificando los datos, y permitiendo el acceso a las funcionalidades respectivas del sistema según el tipo de usuario.	Alta
	Entrada	Proceso		Salida	
	Credenciales del usuario proporcionadas por el Frontend.	Autenticación de las credenciales		Respuesta al frontend para acceder a las funcionalidades respectivas en base al tipo de usuario.	

Tabla 8: Login

Para una mejor comprensión de la arquitectura del sistema se utiliza las siguientes ilustraciones de modelos entidad-relación como herramienta visual para representar la estructura y relaciones de los datos de la Base de Datos.

Modelo Entidad-Relación

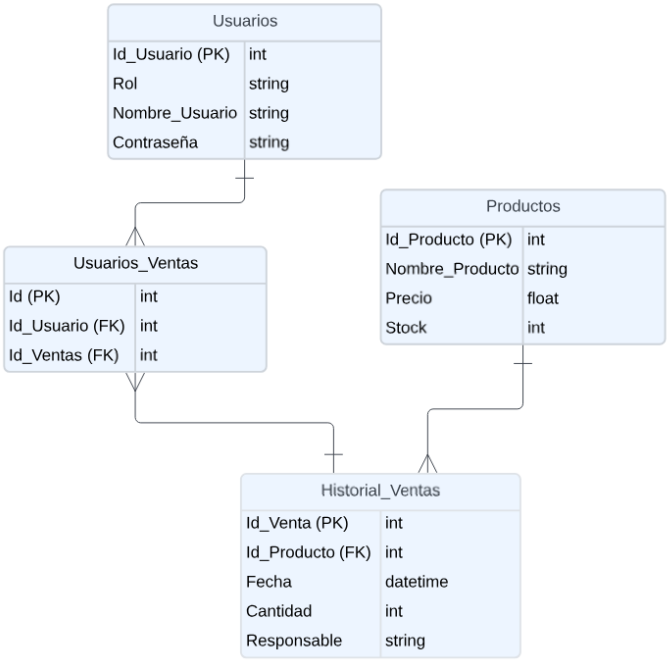

Ilustración 11: Modelo Entidad-Relación

Fuente: Arias, R.

Diagrama de Clases UML

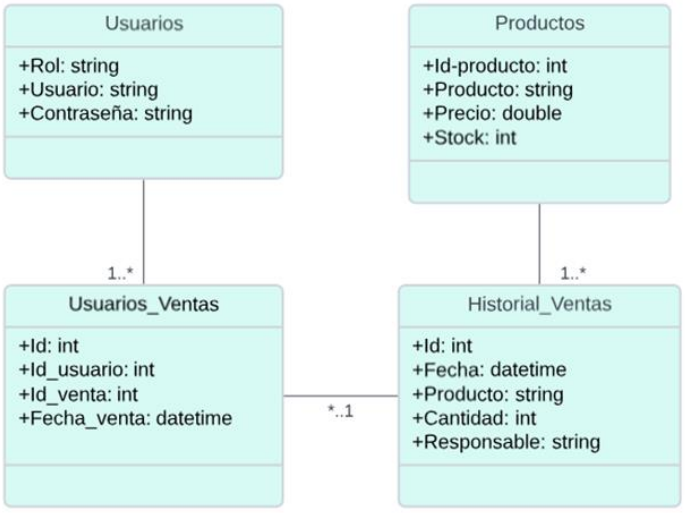

Ilustración 12: Diagrama de Clases UML

Fuente: Arias, R.

Capítulo II

Construcción del Sistema Frontend

Estándares de construcción

Para la construcción se utilizó la metodología de desarrollo Atomic Design, ya que esta nos permite realizar aplicaciones complejas juntando pequeños componentes. (Barajas, Sf, Sp).

Para todo el proyecto todas las funciones, variables y clases de CSS serán escritas en inglés.

En el caso de las funciones estarán escritas en “UpperCamellCase”.

En el caso de las variables y clases de CSS, estarán escrita en “loweCamellCase”

Definición de Paginas

Como se mencionó anteriormente, se está utilizando Atomic Design para el desarrollo, por lo tanto, esta aplicación constara de 4 principales páginas las cuales son:

- Login
- Stock Manage (Control de Stock)
- Sales (Ventas)
- Sales History (Historial de Ventas)

Codificación de las páginas

Login

Esta página se encuentra estructurada y codificada de la siguiente manera:

Login

Este es el componente principal, sirve como elemento visual, pero también es el encargado de confirmar que las credenciales ingresadas son válidas y actuar en base a ello, esto mediante los datos que son enviados por el “LoginForm”.

LoginForm

Es el encargado de recibir los datos ingresados por el usuario y enviarlos hacia el componente “Login”.

Stock Manage

Esta página se encuentra estructurada y codificada de la siguiente manera:

StockManage:

El módulo Principal llamado “StockManage”, el cual contiene el módulo “StockManageTable”.

En este se visualiza el componente junto a sus estilos.

StockManageTable:

Aquí se manejan los datos y las funciones de la tabla mediante la utilización de las APIS proporcionadas por el Backend.

Las funciones que se encuentran aquí son:

- Listar los productos y seleccionar la cantidad de registros que se desean ver.
- Agregar Productos
- Editar Productos
- Eliminar Productos

Contiene los módulos “EditModal”, “DeleteModal”, “StockManageHeader”. El cual maneja la recolección de los datos, editarlos y eliminarlos en tiempo real a “StockManageTable”.

StockManageHeader:

Este último contiene el módulo “AddModal”. El cual maneja la recolección de los datos y agregarlos en tiempo real a “StockManageTable”.

Sales

Esta pantalla se encuentra estructurada y codificada de la siguiente manera:

Sales

Este siendo el módulo principal está compuesto de “SalesTable” y “SalesHeader”.

SalesTable

Aquí se muestran las ventas realizadas mediante la API proporcionada por el Backend con los datos de: Fecha, Productos, Cantidad, Responsable.

Se pueden ampliar la cantidad de registros a observar.

SalesHeader

Es la cabecera de la página, la cual contiene un botón con el texto “Realizar Venta”, y es el encargado de abrir el Modal para realizar la venta.

SaleModal

Estos compuestos de 3 campos necesarios para realizar una venta, aquí se selecciona el producto a vender, se agrega la cantidad y el responsable, y al momento de darle a “Confirmar”, se realizará la venta quedando registrado y reduciendo la cantidad de stock.

Sales History

Esta pantalla se encuentra estructurada y codificada de la siguiente manera:

SalesHistory

Este siendo el módulo principal está compuesto de “SalesHistoryTable” y “SalesHistoryHeader”.

SalesHistoryTable

Aquí se ven las ventas realizadas vistas a mayor detalle, cuenta con Nro. De venta, Fecha, Producto, Cantidad, Responsable y el total de la venta basado en el precio del producto y la cantidad vendida.

SalesHistoryHeader

Este componente solo sirve para mostrar elementos visuales como el logo, y el nombre del gimnasio.

Componentes adicionales

useNavigation

Es un hook personalizado reutilizable basado en el hook “useNavigate” de React Router. Simplifica la navegación entre páginas al proporcionar una función que se puede reutilizar, aceptando solo la ruta del componente al que queremos acceder como parámetro. A diferencia de “useNavigate”, “useNavigation” no requiere definir una nueva función para cada ruta.

Los Hooks personalizados en React permiten extraer lógica de componentes en funciones reutilizables. Un Hook personalizado es una función que empieza con "use" y puede llamar a otros Hooks. Por ejemplo, UseNavigation simplifica la navegación al proporcionar una función reutilizable que acepta solo la ruta del componente al que queremos acceder, mejorando la reutilización del código.

AuthContext

Este contexto es una herramienta para manejar el estado global de autenticación a través de la aplicación. Este contexto permite compartir y acceder a datos relacionados con la autenticación sin necesidad de propagar “props” en cada componente.

En este componente está definida la lógica para realizar la autenticación enviando los datos hacia el backend y recibiendo una respuesta en base a los datos proporcionados. También gracias a este contexto se puede asignar que datos, elementos y componentes deben ser vistos por los usuarios “admin” y “employee”.

Los contextos en React proporcionan una forma de pasar datos a través de componentes sin tener que pasar “props” manualmente en cada nivel. Es especialmente útil para datos que deben ser accesibles por muchos componentes en diferentes niveles, como un tema de usuario, datos de autenticación, etc.

ProtectedRoute

Este componente nos sirve para evitar que los usuarios que no tienen permisos puedan acceder a los demás componentes mediante la URL, dándole así libre acceso al usuario “admin”, pero un acceso restrigido y limitado al usuario “employee”.

Construcción del Sistema Backend

Estándares de Construcción

Metodología MVC (Modelo, Vista, Controlador).

Para la construcción del sistema se hizo uso de la metodología de desarrollo MVC.

La metodología MVC, separa la lógica (Modelo), la interfaz de usuario conocido como Vista y la interacción del usuario (Controlador). Esta estructura ayuda al mantenimiento y escalabilidad del sistema.

- Estándares de Implementación de los Modelos

Se hace uso de CamelCase para los nombres de los modelos: ‘Usuario’, ‘Producto’, ‘Historial Ventas’ y ‘Usuario Venta’.

- Estándares de Implementación de Paquetes

Los paquetes están nombrados completamente en minúsculas, sin espacios ni guiones.

- Estándares de Implementación de Atributos

Para el caso de los atributos, estos están escritos en snake_case.

- Estándares de Implementación de Funciones

En el caso de las funciones y métodos están escritos en snake_case.

Definición de Modelos

En el sistema se implementan modelos como ‘Usuario’, ‘Producto’, ‘Historial Ventas’ y ‘Usuario Venta’

Modelo Usuario

Este modelo consta de los siguientes atributos, identificador del usuario como clave primaria, rol, nombre del usuario y la clave del usuario.

Modelo Producto

El modelo Producto contiene todos los atributos de un producto, aquí consta del identificador del producto como clave primaria, nombre del producto, precio y su stock.

Modelo Historial Ventas

El modelo Historial Ventas consta de los siguientes atributos, el identificador de la venta como clave primaria, el identificador de los productos, el nombre del producto, la fecha, la cantidad, el responsable de la venta y el precio del producto vendido.

Modelo Usuario Venta

Por último, el modelo Usuario Venta consta de 3 identificadores como atributos, su identificador principal cómo clave primaria, el identificador de usuario como clave foránea y el identificador venta como clave foránea.

Codificación de Modelos

Se tiene 4 modelos para crear las entidades y atributos en la base de datos.

El modelo 'Usuario'

Contiene los siguientes atributos

- Atributo 'id_usuario': clave primaria del modelo 'Usuario'.
- Atributo 'rol': Este campo almacena el rol del usuario, su capacidad es de 32 caracteres.
- Atributo 'nombre_usuario': Posee un máximo de caracteres de 32.
- Atributo 'clave': Campo de caracteres para el nombre del usuario, con una longitud máxima de 25 caracteres.

Este modelo posee un método '__str__' el cual retorna el nombre de usuario como representación en String del objeto.

El modelo ‘Producto’

Para la creación del modelo ‘Producto’ se tomó en cuenta sus 4 atributos, ‘id_prodcuto’, ‘nombre_producto’, ‘precio’ y ‘stock’.

A nivel de código el ‘id_producto’ es la clave primaria del modelo ‘Producto’, el nombre del producto tiene un máximo de 100 caracteres, el campo de precio admite un ‘Float’ es decir un decimal y el campo de stock un ‘Integer’ es decir un entero.

De la misma manera posee un método __str__ el cual retorna el nombre del producto.

El modelo ‘Historial Ventas’

El modelo ‘Historial Ventas’ tiene 7 atributos los cuales se configuran de la siguiente manera:

- Atributo ‘id_venta’: Este campo se autoincrementa e indica que es clave primaria del modelo ‘Historial Ventas’.
- Atributo ‘id_producto’: Es una clave foránea del modelo ‘Producto’, adicional a ello ‘on_delete=models.CASCADE’ indica que al eliminar un producto todas las ventas asociadas a ese producto también serán eliminados automáticamente.
- Atributo ‘nombre_producto: Este campo permite hasta 100 caracteres como máximo.
- Atributo ‘fecha’: Este campo de fecha y hora registra automáticamente la fecha de creación de la venta
- Atributo ‘cantidad’: Campo de tipo entero el cual recoge una cierta cantidad de productos.

- Atributo ‘responsable’: Campo de texto que almacena el nombre del responsable al realizar una venta.
- Atributo ‘precio’: Campo flotante (decimal) que indica el precio de un producto.

Adicional a ello el método ‘Historial Ventas’ hace uso de 2 métodos:

- Método ‘save’: Personaliza el comportamiento de guardado del modelo para actualizar automáticamente el campo nombre_producto con el nombre actual del producto vinculado cada vez que se guarda una venta.
- Método ‘__str__’ : Método especial que devuelve una representación en cadena del objeto, mostrando el identificador de la venta y el nombre del producto asociado.

Clase ‘Meta’

Subclase que especifica cómo deben ordenarse las instancias del modelo ‘Historial Ventas’ por defecto, en este caso, por ‘id_venta’ en orden descendente.

El modelo ‘Usuario Venta’

- Contiene los siguientes atributos:
- Atributo ‘id’: Campo auto incremental el cual es clave primaria.
- Atributo ‘id_usuario’: ‘ForeignKey’ (clave foránea) que vincula este modelo con el modelo ‘Usuario’, estableciendo una relación de muchos a uno. Si un usuario es eliminado, todas sus ventas asociadas también se eliminarán. (on_delete=models.CASCADE).
- Atributo ‘id_ventas’: ‘ForeignKey’ que enlaza con el modelo ‘Historial Ventas’, también con eliminación en cascada, ya que, si una venta

específica es eliminada, cualquier registro en ‘Usuario Venta’ que se refiera a esa venta también se eliminará.

Método ‘__str__’: Proporciona una representación en cadena del objeto, que muestra el nombre del usuario y el identificador de la venta asociada, facilitando la identificación de registros en interfaces de usuario o logs.

Definición de Controladores

Los controladores proporcionan la lógica para interactuar con los modelos, estos permiten operaciones CRUD para cada modelo.

El controlador ‘LoginView’ maneja la autentificación verificando las credenciales de los usuarios. Su función es recoger el nombre de usuario y clave del Request realizado

En ello poseo una función o método post el cual maneja las solicitudes POST HTTP.

‘self’ hace referencia a la instancia de la clase y ‘request’ es el objeto que contiene los datos de la solicitud. Luego de ello se extraen o se recogen los datos: nombre de usuario y clave.

Los controladores (vistas) en Django REST Framework se implementan como “ViewSets” y “APIView”.

ViewSets proporciona una interfaz completa para operaciones CRUD (create, read, update and delete) estándar sin la necesidad de definir métodos explícitos para cada operación.

APIView se lo utiliza para una operación más específica como lo es el Login.

Codificación de Controladores

Las vistas están directamente conectadas con los Serializadores, donde cada "ViewSet" utiliza un serializador específico para convertir los modelos de datos en JSON y viceversa, facilitando asi la conexión con el Front.

Definición de Serializadores

Cada modelo tiene un serializador. Estos ayudan a manejar los campos de una entidad.

'UsuarioSerializer'

Serializa el modelo 'Usuario' incluyendo todos sus campos permitiendo que la información sea accesible y manipulable a través de la API.

'ProductoSerializer'

Serializa el modelo 'Producto' incluyendo todos sus atributos. Esto permite listar y gestionar productos a través de la API.

'HistorialVentasSerializer'

Serializa el modelo 'Historial Ventas'. Esto captura todas las transacciones de ventas, incluyendo la relación con el producto vendido y el responsable de la venta.

'UsuarioVentaSerializer'

Serializa el modelo 'Usuario Venta' que relaciona los modelos 'Usuario' e 'Historial Ventas' lo cual permite gestionar el uso de la API.

Codificación de Serializadores

Cada serializador hereda 'ModelSerializer' y utiliza una clase interna llamada 'Meta' la cual especifica el modelo correspondiente y los campos a incluir en la

serialización, usando ‘fields = ‘__all__’’ para incluir todos los campos de ese modelo en específico.

Definición de URLS

En Django, las URLS mapean las solicitudes de los usuarios a las vistas específicas que manejan estas solicitudes. Aquí se maneja las Urls del proyecto por tal se emplea el uso de las APIs.

Codificación de URLS

Este código configura las rutas para una API REST utilizando Django REST Framework junto al propio Django.

- Importaciones

‘path’ e ‘include’ son utilizadas para definir patrones de URL e incluir rutas de otras aplicaciones.

- Configuración del Router

‘router’ crea una instancia de ‘DefaultRouter’ para manejar el registro de ‘viewsets’ de los controladores.

‘router.register’ registra ‘viewsets’ con rutas especificas configurando así las URLs para operaciones CRUD en cada modelo.

- Definición de ‘urlpatterns’

```
path('admin/', admin.site.urls),
```

Integra el sitio administrativo de Django, permitiendo la gestión de modelos en una interfaz gráfica.

```
path('api/', include(router.urls)),
```

Incluye todas las URLs generadas por el router bajo el prefijo ‘api/’, organizando las rutas de la API en un solo punto de acceso.

```
path('api/login/', LoginView.as_view(), name='login'),
```

Define una ruta específica para el proceso de inicio de sesión, manejado por ‘LoginView’.

Conexión con Base de Datos

Para este proyecto se utilizó el motor de base de datos PostgreSQL.

La conexión a la base de datos se gestiona automáticamente por Django ORM, ya que este framework Backend proporciona su propio ORM. Esto quiere decir que la creación de los modelos usando Django se ven directamente reflejados en la base de datos.

En el archivo de ‘settings.py’ hay un apartado llamado ‘DATABASES’ aquí van todos los datos de la base de datos, como adaptador para la base de datos postgres, el nombre de la base de datos, el usuario que emplea dicha base, la contraseña, el host y el puerto.

Para que cada modelo se vea reflejado en la base de datos basta con los comandos ‘makemigrations’ y ‘migrate’ para crear migraciones a la base de datos y migrar.

Anexos

https://github.com/IsaacMullo/AtlasGym-Frontend

https://github.com/Reichzz/AtlasGym-Backend

Capítulo III

Pruebas y Despliegue Frontend

Pruebas funcionales

N°1 Caso de prueba de login con rol admin

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad del login.	
Precondiciones	• Existir usuarios registrados en la base de datos. • Ingresar datos correctos en los campos de usuario y contraseña • Usuario con rol administrador
Pasos para seguir	Ingresar el usuario y contraseña correctos, posteriormente dar click a botón ingresar, este validara los datos e ingresara al sistema a la página "sales (ventas)", al ser administrador contara con 3 botones para navegar a través de las 3 páginas existentes.
Datos de entrada	• Usuario • Contraseña
Resultados esperados	Autenticación e ingreso al sistema
Criterios de Aceptación/Rechazo	**Aceptación:** La autenticación de datos fue correcta y se ingresó al sistema. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 9: Prueba funcional 1

N°2 Caso de prueba de login con rol employee

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad del login.	
Precondiciones	• Existir usuarios registrados en la base de datos. • Ingresar datos correctos en los campos de usuario y contraseña • Usuario con rol employee
Pasos para seguir	Ingresar el usuario y contraseña correctos, posteriormente dar click a botón ingresar, este validara los datos e ingresara al sistema a la página "sales (ventas)", al ser employee solo contara con acceso a la misma.
Datos de entrada	• Usuario • Contraseña
Resultados esperados	Autenticación e ingreso al sistema
Criterios de Aceptación/Rechazo	**Aceptación:** La autenticación de datos fue correcta y se ingresó al sistema. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 10: Prueba funcional 2

N°3 Caso de prueba de login fallido

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad del login.	
Precondiciones	• Ingresar datos incorrectos en los campos de usuario y contraseña
Pasos para seguir	Ingresar el usuario y contraseña incorrectos, posteriormente dar click a botón ingresar y saldrá un mensaje que dirá “Credenciales no válidas”.
Datos de entrada	• Usuario • Contraseña
Resultados esperados	Aparición de mensaje de error.
Criterios de Aceptación/Rechazo	**Aceptación:** La autenticación de datos fue incorrecta, no ingresó al sistema y apareció un mensaje de error. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 11: Prueba funcional 3

N°4 Caso de prueba agregar stock

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de agregar stock.	
Precondiciones	• Dar click al botón de agregar producto • Ingresar nombre de producto, stock y precio.
Pasos para seguir	Ingresar nombre de productos, stock y precio, y dar click al botón guardar.
Datos de entrada	• Nombre • Stock • Precio
Resultados esperados	Cierre de modal y agregación del producto en la tabla y base de datos.
Criterios de Aceptación/Rechazo	**Aceptación:** El producto fue correctamente agregado. **Rechazo:** El producto no se agregó por falta de datos.
Desarrollador Asignado	Isaac Mullo

Tabla 12: Prueba funcional 4

N°5 Caso de prueba validación de campos al agregar stock

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de agregar stock.	
Precondiciones	• Dar click al botón de agregar producto • Falta de datos en alguno de los campos.

Pasos para seguir	Al dar click al botón de guardar no permitirá el registro de estos datos mientras los campos no sean llenados.
Datos de entrada	• Nombre • Stock • Precio
Resultados esperados	Cierre de modal sin cambios en la tabla de registros y en la base datos.
Criterios de Aceptación/Rechazo	**Aceptación:** El producto no fue agregado. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 13: Prueba funcional 5

N°6 Caso de prueba editar Stock

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de editar stock.	
Precondiciones	• Dar click al icono/botón de editar producto ubicado en la tabla. • Datos nombre de producto, stock, precio.
Pasos para seguir	Se mostrará modal con todos los datos del registro y al dar click al botón de guardar, se actualizará el registro con los nuevos datos ingresados.
Datos de entrada	• Nombre • Stock • Precio
Resultados esperados	Cierre de modal y edición del producto en la tabla y base de datos.
Criterios de Aceptación/Rechazo	**Aceptación:** El producto fue editado agregado. **Rechazo:** El producto no se editó por falta de datos.
Desarrollador Asignado	Isaac Mullo

Tabla 14: Prueba funcional 6

N°7 Caso de prueba validación de campos al editar stock

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de editar stock.	
Precondiciones	• Dar click al icono/botón de editar producto ubicado en la tabla. • Falta de datos en alguno de los campos.
Pasos para seguir	Al dar click al botón de guardar no permitirá la edición del registro de estos datos mientras los campos no sean llenados.
Datos de entrada	• Nombre • Stock • Precio

Resultados esperados	Cierre de modal sin cambios en la tabla de registros y en la base datos.
Criterios de Aceptación/Rechazo	**Aceptación:** El producto no fue editado. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 15: Prueba funcional 7

N°8 Caso de prueba eliminar Stock

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de eliminar stock.	
Precondiciones	• Dar click al botón de eliminar producto ubicado en la tabla.
Pasos para seguir	Al dar click al botón de confirmar, se eliminará el producto de la tabla de registros y de la base de datos.
Datos de entrada	Ninguno
Resultados esperados	Cierre de modal con el registro del producto eliminado.
Criterios de Aceptación/Rechazo	**Aceptación:** El producto fue eliminado. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 16: Prueba funcional 8

N°9 Caso de prueba realizar venta

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de realizar venta.	
Precondiciones	• Dar click al botón de realizar venta. • Ingresar producto y cantidad.
Pasos para seguir	Seleccionar producto y cantidad, al momento de dar al botón confirmar.
Datos de entrada	• Nombre • Stock • Precio
Resultados esperados	Cierre de modal y realización de la venta, registro en la tabla de registros y base de datos, y reducción del stock del producto seleccionado.
Criterios de Aceptación/Rechazo	**Aceptación:** La venta se realizó y registro con éxito. **Rechazo:** La venta no se realizó por falta de datos.
Desarrollador Asignado	Isaac Mullo

Tabla 17: Prueba funcional 9

N°10 Caso de prueba validación de campos al realizar venta

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de realizar venta.

Precondiciones	• Dar click al botón de realizar venta. • Falta de datos en alguno de los campos.
Pasos para seguir	Al dar click al botón de confirmar no permitirá la realización de la venta.
Datos de entrada	• Nombre • Stock • Precio
Resultados esperados	Se mantendrá el modal abierto y no se realizará ningún cambio en la tabla o base de datos hasta que se llenen los datos de estos campos.
Criterios de Aceptación/Rechazo	**Aceptación:** La venta se realizó y registro con éxito. **Rechazo:** La venta no se realizó por falta de datos.
Desarrollador Asignado	Isaac Mullo

Tabla 18: Prueba funcional 10

N°11 Caso de prueba validación de acceso a módulos en base al rol

DESCRIPCIÓN DEL CASO DE PRUEBA Se probará la funcionalidad de ruta protegida.	
Precondiciones	• Iniciar sesión con una cuenta employee.
Pasos para seguir	La url saldrá por defecto como "/sales" y tendrá que ser cambiada por "/stock" o "/history".
Datos de entrada	• Endpoint de la URL
Resultados esperados	Al realizar el cambio deberá regresar al usuario a la página de login.
Criterios de Aceptación/Rechazo	**Aceptación:** Retorno a la página de login. **Rechazo:** No aplica.
Desarrollador Asignado	Isaac Mullo

Tabla 19: Prueba funcional 11

Resultados de las pruebas:

N°1 Prueba de login con rol admin

- El usuario administrador ingresa el nombre de usuario y contraseña correctos, y si los datos se validan exitosamente, el sistema es dirigido a la página de "sales(ventas)", donde se muestran tres botones para navegar a las tres páginas existentes, siendo así un resultado exitoso.

N°2 Prueba de login con rol employee

- El usuario empleado ingresa el nombre de usuario y contraseña correctos, y si el sistema valida los datos exitosamente, entonces el sistema ingresa a la página de “sales(ventas)”, restringida solo a esta página, siendo así un resultado exitoso.

N°3 Prueba de login fallido

- El usuario ingresa un nombre de usuario y contraseña incorrectos. Después de eso, el sistema muestra un mensaje que dice "Credenciales no válidas", siendo así un resultado exitoso.

N°4 Prueba de agregar stock

- El usuario ingresa el nombre del producto, stock y precio, y al dar click en el botón guardar, el sistema cierra el modal y agrega el producto en la tabla y en la base de datos, siendo así un resultado exitoso.

N°5 Prueba de validación de campos al agregar stock

- El usuario intenta guardar un producto con datos incompletos, y el sistema no permite el registro de estos datos mientras los campos no estén llenos, cerrando el modal sin cambios en la tabla de registros y en la base de datos, siendo así un resultado exitoso.

N°6 Prueba de Editar Stock

- El usuario edita los datos del producto (nombre, stock, precio) y al dar click en el botón de guardar, el sistema actualiza el registro con los nuevos datos ingresados y cierra el modal, siendo así un resultado exitoso.

N°7 Prueba de Validación de campos al editar stock

- El usuario intenta editar un producto con datos incompletos, y no se le permite la edición del registro mientras los campos no estén llenos, cerrando el modal sin cambios en la tabla de registros y en la base de datos, siendo así un resultado exitoso.

N°8 Prueba Eliminar Stock

- El usuario da click en el botón de eliminar producto y confirma la acción, el sistema elimina el producto de la tabla de registros y de la base de datos, cerrando el modal, siendo así un resultado exitoso.

N°9 Prueba Realizar venta

- El usuario selecciona el producto y la cantidad, y al confirmar la venta, el sistema registra la venta, reduce el stock del producto seleccionado y cierra el modal, actualizando la tabla de registros y la base de datos, siendo así un resultado exitoso.

N°10 Prueba Validación de campos al realizar venta

- El usuario intenta realizar una venta con datos incompletos, y el sistema no permite la realización de la venta, manteniendo el modal abierto y sin realizar ningún cambio en la tabla o base de datos hasta que se llenen los datos requeridos, siendo así un resultado exitoso.

N°11 Prueba Validación de acceso a módulos en base al rol

- El usuario con rol de employee intenta acceder a módulos restringidos cambiando la URL, y el sistema lo redirige a la página de login, siendo así un resultado exitoso.

Instalación de Software FrontEnd

Pre-requisitos:

- Tener instalado Node.js.

- Contar con un IDE.

Pasos por seguir para instalar el software de manera local:

1. Clonar el proyecto del repositorio de GitHub Publico.

2.Abrir la carpeta del proyecto con el IDE (Integrated Development Environment) de preferencia.

3.Abrir la terminal y ejecutar el comando “npm install” seguido de “npm run dev”, con esto nuestro proyecto se estará ejecutando localmente en la siguiente url y puerto: “http://localhost:5173”.

Despliegue FrontEnd:

El despliegue del Frontend se realizó en la plataforma gratuita Vercel, siguiendo los siguientes pasos.

1.Tener el código cargado en Github y crear una cuenta en Vercel.

2.Importar el repositorio a la cuenta de Vercel

3.Corregir los errores que arroja la página

4.Una vez realizado el paso anterior, realizar el despliegue.

Pruebas y Despliegue Backend

Pruebas Unitarias

N°1 Caso de Prueba API Login/ Método Get Usuarios

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API retorna la información del rol, usuario y clave.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Realizar una petición GET a la API de Login.
Datos de Entrada	• Id del usuario
Resultados Esperados	Que el método GET devuelva un listado de todos los usuarios y adicional a ello en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** Que devuelva una respuesta HTTP 200 devolviendo un formato Json con el listado de los usuarios. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 20: Prueba unitaria 1

N°2 Caso de Prueba API Login/ Método Post Usuarios

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API permite guardar la información del usuario.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Se realiza una petición de tipo POST para la API de Login y se envía como parámetros el rol, usuario y clave.
Datos de Entrada	• Rol • Usuario • Clave
Resultados Esperados	Que el método POST permita realizar registros para los usuarios.
Criterios de Aceptación / Rechazo	**Aceptación:** El usuario se añadió **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 21: Prueba unitaria 2

N°3 Caso de Prueba API Login /Método Put Usuarios

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API permite actualizar la información de los usuarios.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id del usuario.
Pasos para seguir	Se realiza una petición de tipo Put y se envía como parámetro el id del usuario.
Datos de Entrada	• Id del usuario
Resultados Esperados	Que el método PUT permita actualizar a los usuarios en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** Se modificó el usuario. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 22: Prueba unitaria 3

N°4 Caso de Prueba API Login/Método Delete Usuarios

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API permite eliminar por el id los usuarios existentes.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id del usuario.
Pasos para seguir	Una petición de tipo Delete, enviando como parámetro el id del usuario al que se desea eliminar.
Datos de Entrada	• Id del usuario
Resultados Esperados	Eliminar registros en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** Se eliminó el usuario. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 23: Prueba unitaria 4

N°5 Caso de Prueba API Productos/Método Get Productos

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API es retornar la información de los productos.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Realizar una petición GET a la API de Productos.
Datos de Entrada	• Id del producto
Resultados Esperados	Devolver un listado de los productos existentes.

Criterios de Aceptación / Rechazo	**Aceptación:** Que devuelva una respuesta HTTP 200 devolviendo un formato Json con el listado de los productos registrados. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 24: Prueba unitaria 5

N°6 Caso de Prueba API Productos/ Método Post Productos

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API permite guardar productos.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Realizar una petición Post a la API de Productos.
Datos de Entrada	• Nombre del Producto • Precio • Stock
Resultados Esperados	Agregar y guardar productos.
Criterios de Aceptación / Rechazo	**Aceptación:** Guarda registros para los productos. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 25: Prueba unitaria 6

N°7 Caso de Prueba API Productos/Método Put Productos

DESCRIPCIÓN DEL CASO DE PRUEBA El método de esta API permite actualizar los productos mediante su id.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id del producto.
Pasos para seguir	Para la petición de tipo Put, se envía como parámetro el id del producto.
Datos de Entrada	• Id del producto • Nombre del Producto • Precio • Stock
Resultados Esperados	Permitir que se actualicen los productos en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** Productos actualizados. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 26: Prueba unitaria 7

N°8 Caso de Prueba API Productos/Método Delete Productos

DESCRIPCIÓN DEL CASO DE PRUEBA El método de esta API permite eliminar productos en base al id.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id del producto
Pasos para seguir	Para la petición Delete, se envía como parámetro el id del producto.
Datos de Entrada	• Id del Producto
Resultados Esperados	Que permita eliminar productos en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** Se eliminó el producto en base al id. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 27: Prueba unitaria 8

N°9 Caso de Prueba API Historial Ventas/ Método Get

DESCRIPCIÓN DEL CASO DE PRUEBA El método de esta API devuelve la información de las ventas.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Realizar una petición Get a la API de Historial Venta.
Datos de Entrada	• Id de la Venta
Resultados Esperados	Devolver toda la información acerca de las ventas realizadas.
Criterios de Aceptación / Rechazo	**Aceptación:** Devuelve los datos registrados en la base de datos, adicional a ello su código de respuesta HTTP 200. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 28: Prueba unitaria 9

N°10 Caso de Prueba API Historial Ventas/ Método Post

DESCRIPCIÓN DEL CASO DE PRUEBA El método para esta API permite registrar las ventas en Historial Ventas.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Realizar una petición Post a la API de Historial Ventas.
Datos de Entrada	• Nombre del producto

	• Cantidad • Responsable • Precio
Resultados Esperados	El permitir crear y guardar registros en Historial Ventas.
Criterios de Aceptación / Rechazo	**Aceptación:** Se crearon nuevos registros. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 29: Prueba unitaria 10

N°11 Caso de Prueba API Historial Ventas/Método PUT

DESCRIPCIÓN DEL CASO DE PRUEBA El método de esta API permite actualizar los registros del Historial de las Ventas.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id de la venta.
Pasos para seguir	Se realiza la petición de tipo Put enviando como parámetro el id de la venta.
Datos de Entrada	• Id venta • Nombre del producto • Cantidad • Responsable • Precio
Resultados Esperados	Que permita actualizar los registros en base al id.
Criterios de Aceptación / Rechazo	**Aceptación:** La API permitió el actualizar los datos. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 30: Prueba unitaria 11

N°12 Caso de Prueba API Historial Ventas/ Método DELETE

DESCRIPCIÓN DEL CASO DE PRUEBA Esta API permite eliminar los registros en Historial Ventas.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id de la venta.
Pasos para seguir	Se envía como parámetro el id de la venta a eliminar.
Datos de Entrada	• El id de la venta.
Resultados Esperados	Eliminar los registros en Historial Ventas en base al id.

Criterios de Aceptación / Rechazo	**Aceptación:** Se eliminó correctamente los datos en base al id. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 31: Prueba unitaria 12

N°13 Caso de Prueba API Usuario Ventas – Método GET

DESCRIPCIÓN DEL CASO DE PRUEBA Esta API permite obtener el id_usuario y el id_venta en Usuario Venta.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Una petición del método Get.
Datos de Entrada	• Id del Usuario • Id de las Ventas
Resultados Esperados	Devuelve un Json con la información del Usuario Ventas.
Criterios de Aceptación / Rechazo	**Aceptación:** Estado del método HTTP devuelve 200. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 32: Prueba unitaria 13

N°14 Caso de Prueba API Usuario Ventas /Método POST

DESCRIPCIÓN DEL CASO DE PRUEBA Esta API permite crear una nueva relación entre un usuario y una venta específica, registrando la información en la tabla UsuarioVenta.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución.
Pasos para seguir	Asegurarse de que los IDs de Usuario e HistorialVentas existen en la base de datos.
Datos de Entrada	• Id del Usuario • Id de las Ventas
Resultados Esperados	Devuelve un JSON con la información del UsuarioVenta recién creado, incluyendo los IDs relacionados.
Criterios de Aceptación / Rechazo	**Aceptación:** Permite guardar registros con sus respectivas relaciones. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 33: Prueba unitaria 14

N°15 Caso de Prueba API Usuario Ventas/Método Put

DESCRIPCIÓN DEL CASO DE PRUEBA Esta API permite actualizar el historial de ventas de un usuario existente en la tabla UsuarioVenta.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El id de la venta debe existir en la base de datos.
Pasos para seguir	• Asegurarse de que los Ids de Usuario e HistorialVentas existan en la base de datos. • Ingresar los parámetros a editar.
Datos de Entrada	• Id del Usuario • Id de las Ventas
Resultados Esperados	Devuelve un JSON con la información del UsuarioVenta actualizado, incluyendo los nuevos datos relacionados.
Criterios de Aceptación / Rechazo	**Aceptación:** Estado del método HTTP devuelve 200 y el cuerpo de la respuesta contiene los datos actualizados. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 34: Prueba unitaria 15

N°16 Caso de Prueba API Usuario Ventas/Método Delete

DESCRIPCIÓN DEL CASO DE PRUEBA Esta API permite eliminar una relación existente entre un usuario y una venta específica en la tabla UsuarioVenta.	
Precondiciones	• La base de datos debe estar disponible y accesible. • El servidor de la API debe estar en ejecución. • El ID de la relación UsuarioVenta que se desea eliminar debe existir en la base de datos.
Pasos para seguir	Asegurarse de que el ID de la relación UsuarioVenta que se desea eliminar existe en la base de datos.
Datos de Entrada	El id de la relación “UsuarioVenta”
Resultados Esperados	Eliminar los registros de UsuarioVenta.
Criterios de Aceptación / Rechazo	**Aceptación:** El estado del método HTTP devuelve 204 “No Content”, y la relación UsuarioVenta es eliminada. **Rechazo:** No aplica
Desarrollador Asignado	Rachel Arias

Tabla 35: Prueba unitaria 16

Pruebas de Rendimiento

Para la Pruebas de Rendimiento se utilizó Jmeter, una herramienta para realizar pruebas de Carga.

Pruebas de Carga API Productos

Con 100 registros, el tiempo de respuesta promedio fue de 3 milisegundos.

Latencia promedio: 3 milisegundos.

Con 300 registros, el tiempo de respuesta promedio fue de 9 milisegundos.

Latencia promedio: 9 milisegundos.

Pruebas de Carga API Ventas

Con 500 registros, está dado por hecho, que tantas peticiones bajan el sistema.

Resultados de las Pruebas

N°1 Prueba API Login

- El método Get, devolvió los datos del Login y el código de respuesta HTTP obtenido fue 200.
- El método Post, permitió insertar datos para el Login y su código de respuesta HTTP obtenido fue 201.
- El método Put, permitió el actualizar registros del Login en base al id. Se obtuvo un código de respuesta HTTP de 200.
- El método Delete, permitió el eliminar registros del Login en base al id, obteniendo un código de respuesta HTTP 204.

N°2 Prueba API Productos

- El método Get, devolvió un listado de los productos.

Código de Respuesta HTTP: 200

- El método Post, permitió crear y guardar productos.

Código de Respuesta HTTP: 201

- El método Put, permitió el actualizar los productos existentes en base al id.

Código de Respuesta HTTP: 201

- El método Delete, permitió el eliminar productos en base al id.

Código de Respuesta HTTP: 204

N°3 Prueba API Historial-Ventas

- Con el método Get se obtuvo un código de respuesta HTTP de 200 y devolvió un listado de las ventas.
- El método Post, permitió el crear nuevas ventas, devolviendo a su vez una respuesta HTTP de 201.
- El método Put, actualizó las ventas existentes, en base al id. Su código de respuesta HTTP, de la misma manera fue de 201.
- Y con el método Delete, se eliminó ventas en base al id. Devolviendo este el código de 204.

N°4 Prueba API Usuario-Ventas

- El método Get, devolvió una respuesta HTTP 200 con un listado de las ventas relacionado al usuario.
- El método Post, permitió el crear un nuevo registro devolviendo así un código de respuesta de 201.

- El método Put, actualizó los registros en base al id, devolviendo el mismo código al igual que el método Post.
- El método “Delete” permitió eliminar los registros de Usuario-Ventas, en base al id, devolviendo un código de tipo HTTP 204.

Instalación del Software BackEnd

Backend:

Pre-Requisitos:

- Tener Python 3.0, para esto puedes verificar la versión de Pyhton que tengas con “python --version”
- Contar con un IDE de preferencia Visual Studio Code.

Instalar dependencias una vez clonado el repositorio:

- pip install djangorestframework
- pip install django-cors-headers

Antes de correr el proyecto se debe migrar el esquema de la base de datos, el nombre de la base de datos debe coincidir con los valores predefinidos en “settings.py” “Databases”. Para esto necesitas “python manage.py migrate”

Con todo listo se debería correr en su localhost mediante el siguiente comando “python manage.py runserver”

Si te clonas el repositorio no puedes correrlo de manera local puesto que se eliminaron las dependencias para correrlo en Windows, necesario para subirlo a la plataforma Railway.

Despliegue Backend

Para el despliegue del Backend se hizo uso de una plataforma llamada Railway, seguido de los siguientes pasos:

- El código del backend fue alojado en un repositorio, de preferencia GitHub.
- Se registró con la cuenta propia de GitHub a Railway y se añadió el repositorio del backend.
- Se corrigen errores en el código, debido a importaciones, para finalmente la plataforma despliegue el backend.

Para el despliegue de la base de datos, se utilizó un servicio llamado RDS en la nube de AWS (Amazon Web Services). Este servicio permite administrar una base de datos común en la nube.

Conclusiones

- Se concluyó que el uso de materialUI, fue la mejor opción para elementos visuales, ya que este nos ayudó a la creación de elementos visuales reutilizables, creación de tablas en la cual se muestran los datos y los elementos de login.

- La utilización de la librería de JavaScript, Axios, es la mejor opción, debido a su manejo sencillo de realización consumo de APIS para realizar solicitudes HTTP y HTTPS.

- Se concluyó que el utilizar el framework de Backend, Django, con su librería RestFramework, es una manera efectiva, rápida y sencilla de crear una o varias RESTAPI, para su consumo en el Frontend.

- Al utilizar AWS con el servicio RDS, este con su capa gratuita ofrece una disponibilidad de 24 horas durante 12 meses, teniendo una alta disponibilidad de la aplicación, esta solo se verá afectada en caso de que la plataforma sufra de inconvenientes.

Recomendaciones

- Se recomienda el uso de React junto con Vite para el desarrollo de proyectos web, dado que esta combinación optimiza significativamente los tiempos de desarrollo y mejora la experiencia del desarrollador.

- Se recomienda Utilizar railway para el despliegue del Backend es la mejor opción, debido a que este simplifica la manera en la que se realiza esta acción, solamente siendo necesario el cargar y dar acceso a un repositorio, en este caso mediante la plataforma de GitHub.

- Se recomienda utilizar Amazon Web Services, con su servicio RDS, para el despliegue de una base de datos relacional debido a que el despliegue, manejo y acceso de esta es sencillo y con una curva de aprendizaje baja.

Referencias bibliográficas

React. (s.f.). *Reutilizar la lógica con hooks personalizados.* https://es.react.dev/learn/reusing-logic-with-custom-hooks

Platzi. (s.f.). *¿Qué es Atomic Design? Conoce los beneficios.* https://platzi.com/clases/2484-react-webpack-sass/42217-que-es-atomic-design/

React. (s.f.). *createContext.* https://react.dev/reference/react/createContext

React. (s.f.). *useContext.* https://react.dev/reference/react/useContext

Vite. (s.f.). *Why vite.* https://vitejs.dev/guide/why.html

Django. (2024). *Django Rest Framework.* https://www.django-rest-framework.org

Django. (2024). *Documentation.* https://docs.djangoproject.com/en/5.0/faq/

Django. (2024). *Serializers.* https://www.django-rest-framework.org/api-guide/serializers/

Django. (2024). *django.urls functions for use in URLconfs.* https://docs.djangoproject.com/en/5.0/ref/urls/