

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD CUAJIMALPA

LICENCIATURA EN TECNOLOGÍAS Y SISTEMAS DE
INFORMACIÓN.

PROYECTO FINAL

Sistema de Inventario Full Stack

UNIDAD DE ENSEÑANZA
PROGRAMACIÓN WEB DINÁMICO

Dr. Guillermo Monroy

Marin Sanchez Jose Abraham
2233030514

INTRODUCCIÓN

En este reporte documentare todo el proceso de la realización del proyecto final de la UEA Programación Web Dinámico en donde elegí de entre varias opciones el Sistema de Gestión de Inventario Básico , este proyecto es un sistema fullstack y el objetivo principal es tener nuestra página web con secciones de administración de productos, registros, categorías y un dashboard de métricas.

REQUERIMIENTOS

En las especificaciones dadas por el profesor se trabajara en cumplir con los siguientes requisitos funcionales.

- Creación de CRUD de productos y Categorías.
- Registros de movimientos de stock de entradas y salidas.
- La tabla de productos deberá contar con la implementación de filtros, paginación y búsqueda.
- Creación de un dashboard que muestre métricas simples como pueden ser productos con bajo stock y el total de categorías.
- Documentación con Swagger
- Para la parte del frontend además se hará uso de:
 - Routing.
 - Servicios de comunicación HTTP.
 - Formularios reactivos.
 - Componentes organizados por módulos.

TECNOLOGÍAS

Para realizar este proyecto se usaron las siguientes tecnologías:

FRONTEND : Angular y Angular Material.

BACKEND : Spring Boot 3.5.3, Spring Web , Spring Data JPA.

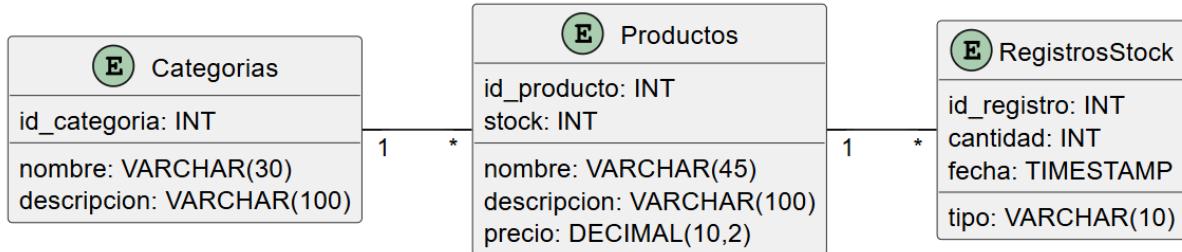
LENGUAJES: Java, Typescript, Javascript, HTML, CSS.

BASE DE DATOS : MySQL.

OTROS: Swagger, Github(https://github.com/MarinoX/proyecto_web_dinamico)

ARQUITECTURA

Estructura de la Base de Datos:

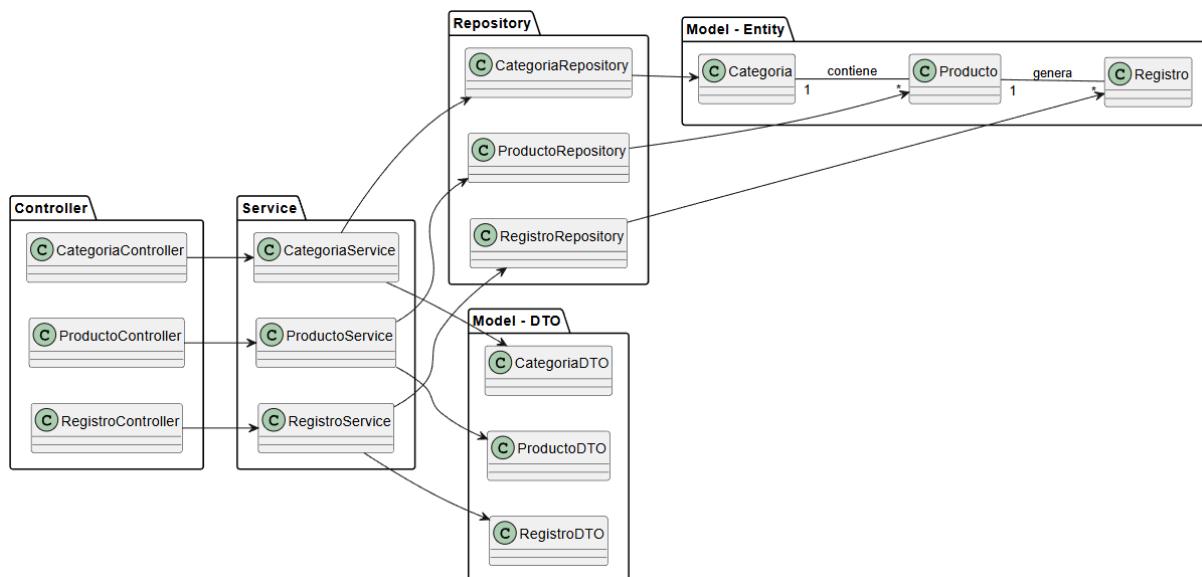


Mi base de datos hecha en SQL está organizada alrededor de la tabla **Categorías** ya que esta funciona como el punto de clasificación principal. Cada categoría tiene un nombre y una descripción, y sirve para agrupar productos de forma ordenada.

Luego tengo la tabla de **Productos** que es pues la parte principal del inventario. Aquí se almacena todo lo importante: nombre, descripción, precio y el stock actual. Cada producto está ligado a una categoría mediante una relación uno a muchos, lo que ayuda a evitar el duplicar información y mantiene todo bien normalizado.

Finalmente tengo la tabla **RegistrosStock** que guarda el historial de entradas y salidas del inventario. Al hacer un nuevo registro indica si fue una entrada o una salida, la cantidad de stock y la fecha exacta.

Estructura del Backend:

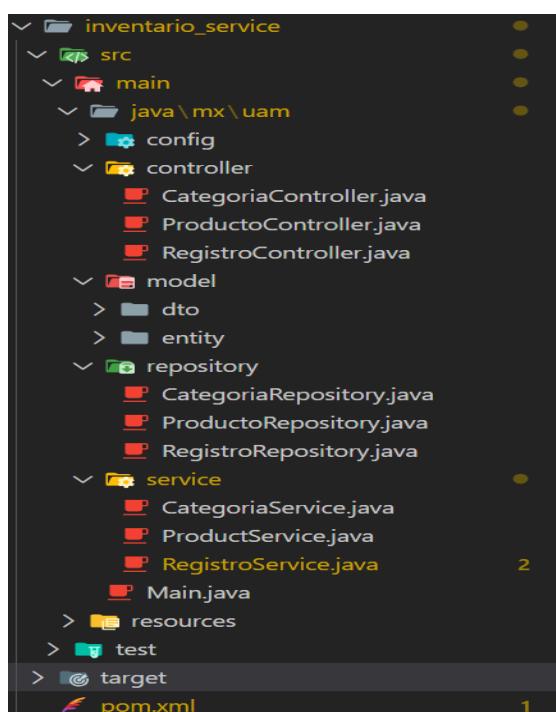


El backend está estructurado siguiendo el patrón MVC, dividiendo la arquitectura en varias capas: Controllers, Services, Repositories y Models (Entity y DTO).

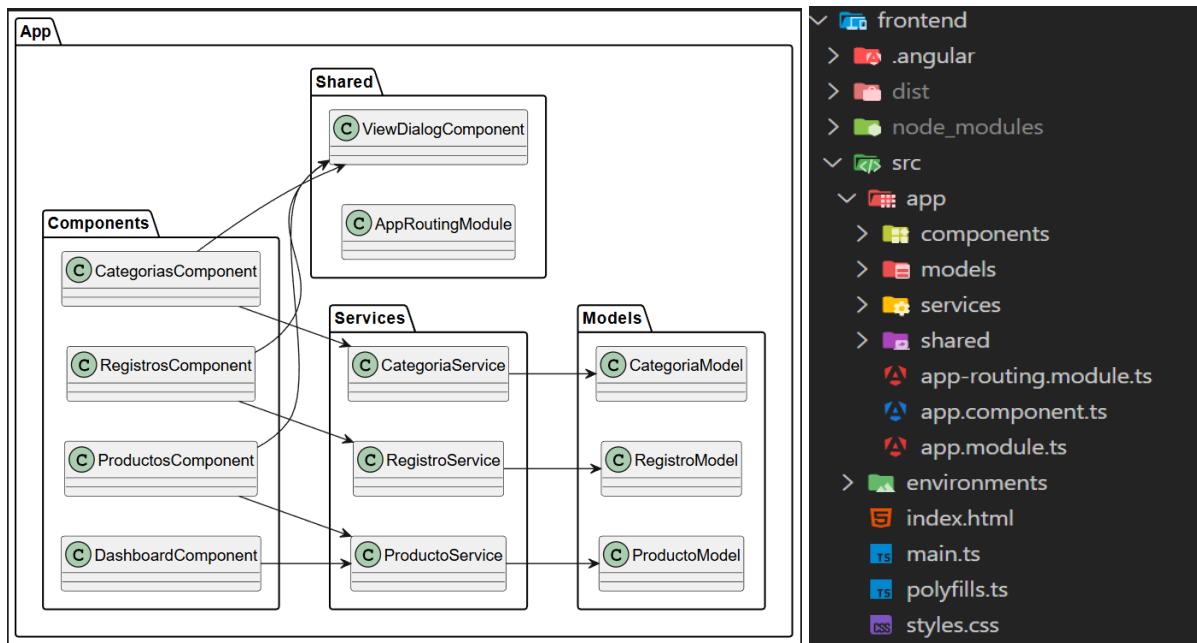
Los Controllers (CategoriaController, ProductoController, RegistroController) actúan como puntos de entrada que reciben las peticiones HTTP y las dirigen a la capa correspondiente donde cada controller maneja las rutas REST específicas para sus recursos.

Los servicios contienen toda la lógica de negocio de la aplicación ya que aquí es donde ocurren las validaciones, cálculos y transformaciones de datos. Por ejemplo el RegistroService no solo crea registros de stock sino que verifica que haya stock suficiente antes de permitir una salida y si si se puede realizar el movimiento automáticamente actualiza el stock del producto.

Los repositoriy contienen las queries y operaciones CRUD necesarias y manejan el acceso directo a la base de datos mediante Spring Data JPA. Los modelos están divididos en Entities (Categoria, Producto, Registro) que mapean directamente las tablas de la base de datos y los DTO que son los objetos que se envían y reciben en las peticiones HTTP.



Estructura del Frontend:

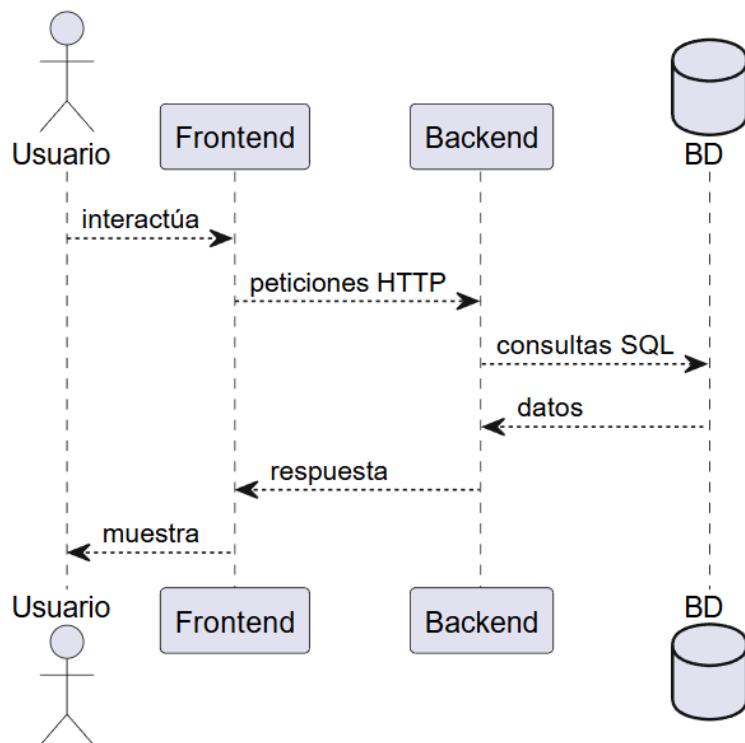


El frontend está construido con Angular y sigue una arquitectura modular basada en componentes. La carpeta App actúa como contenedor principal que contiene toda la aplicación, dentro de esta se encuentran varios componentes para cada apartado específico como el DashboardComponent que muestra métricas del inventario, CategoriasComponent que gestiona las categorías, ProductosComponent para la administración de los productos y RegistrosComponent para visualizar y crear registros de entrada y salida de stock, aquí cada componente es responsable de su propia ui y maneja la interacción con el usuario.

Los Servicios son la conexión entre los componentes y el backend. Los servicios realizan las llamadas HTTP al servidor y transforman los datos recibidos en los Models que son las interfaces que definen la estructura de datos de la aplicación. Haciendo que los componentes no interactúen directamente con el backend sino que solicitan datos a los servicios.

La carpeta Shared contiene componentes y módulos reutilizables en toda la aplicación, como el ViewDialogComponent y el AppRoutingModule que gestiona la navegación entre los diferentes componentes.

Finalmente la interacción de mi sistema full stack sería esta:



PRUEBAS DE BACKEND MEDIANTE SWAGGER

Para la ejecución del proyecto es esencial la lectura del README del repositorio de GITHUB.

Para la documentación de mi backend utilicé Swagger a través de la dependencia el cual genera de forma automática toda la documentación basada en las anotaciones de mis controllers.

```
@GetMapping  
@Operation(summary = "Obtener todas las categorías", description = "Recupera una lista de todas las categorías"  
public List<CategoriaDTO> findAll() {  
    return categoriaService.findAll();  
}
```

A la documentación se puede acceder desde el navegador y nos permite visualizar los endpoints, sus parámetros necesarios y nos deja probar las peticiones directamente sin necesidad de usar Postman, CURL o el mismo frontend.

En mi proyecto la documentación se encuentra disponible en la ruta:

<http://localhost:8080/swagger-ui/index.html>

The screenshot shows the Swagger UI interface for a REST API. At the top, there's a header with the title "Swagger UI" and a sub-header "localhost:8080/swagger-ui/index.html". Below the header, the title "OpenAPI definition" is displayed along with "v0 OAS 3.0". A "Servers" dropdown menu is set to "http://localhost:8080 - Generated server url". The main content area is organized into three sections: "Categoría", "Producto", and "Registro".

- Categoría**: Operaciones CRUD relacionadas con las categorías de productos.
 - GET /categorias/{id} Obtener categoría por ID
 - PUT /categorias/{id} Actualizar categoría
 - DELETE /categorias/{id} Eliminar categoría
 - GET /categorias Obtener todas las categorías
 - POST /categorias Crear categoría
- Producto**: Operaciones CRUD relacionadas con los productos.
 - GET /productos/{id} Obtener producto por ID
 - PUT /productos/{id} Actualizar producto
 - DELETE /productos/{id} Eliminar producto
 - GET /productos Obtener todos los productos
 - POST /productos Crear producto
- Registro**: Operaciones CRUD relacionadas con los registros de stock de productos.
 - GET /registros/{id} Obtener registro por ID

Desde aquí se pueden ver todos los endpoints de Productos, Categorías y Registros de Stock. Cada uno muestra información como: método HTTP, parámetros, respuestas esperadas, códigos de error y ejemplos de uso ,lo que facilita el desarrollo porque se pueden revisar que las rutas funcionan de manera correcta.

This screenshot provides a detailed view of the Swagger UI for the GET /categorias/{id} endpoint. At the top, the method "GET" and the endpoint "/categorias/{id}" are shown, with a description "Obtener categoría por ID". Below this, the "Parameters" section lists a required parameter "id" of type "integer(\$int32)" with a description "ID de la categoría (path)". A text input field contains the value "1". The "Responses" section shows a single response for status code 200 with the description "OK". The "Media type" dropdown is set to "application/json", and the "Example Value" is a JSON object: { "id": 1, "nombre": "string", "descripcion": "string" }.

Algunas pruebas para demostrar el uso de la documentación swagger son las siguientes:

1.- Crear Categoría.

The screenshot shows the Swagger UI interface for a POST request to the endpoint `/categorias`. The title is "Crear categoría".
Description: "Crea una nueva categoría".
Parameters: "No parameters".
Request body (required): "application/json".
Content:

```
{ "id": 0, "nombre": "Limpieza", "descripcion": "Herramientas y elementos de limpieza" }
```


Request URL: "http://localhost:8080/categorias".
Server response: 200 OK. Response body:

```
{ "id": 3, "nombre": "Limpieza", "descripcion": "Herramientas y elementos de limpieza" }
```

Buttons: "Cancel", "Reset", "Copy", "Download".

2.- Obtener todas las categorías.

The screenshot shows the Swagger UI interface for a GET request to the endpoint `/categorias`. The title is "Obtener todas las categorías".
Description: "Recupera una lista de todas las categorías de productos disponibles en el inventario".
Parameters: "No parameters".
Buttons: "Execute", "Clear".
Responses:
Curl:

```
curl -X 'GET' \
'http://localhost:8080/categorias' \
-H 'accept: */*'
```


Request URL: "http://localhost:8080/categorias".
Server response: 200 OK. Response body:

```
[ { "id": 1, "nombre": "Botanas", "descripcion": "Papas y golosinas varias" }, { "id": 2, "nombre": "Panadería", "descripcion": "doras y panes" }, { "id": 3, "nombre": "Limpieza", "descripcion": "Herramientas y elementos de limpieza" } ]
```

Buttons: "Cancel", "Copy", "Download".

3.- Obtener categoría por id.

GET /categorias/{id} Obtener categoría por ID

Reforma una categoría por su ID

Parameters

Name	Description
id * required	integer(\$int32) ID de la categoría (path)

1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/categorias/1' \
-H 'accept: */*'
```

Request URL

```
http://localhost:8080/categorias/1
```

Server response

Code	Details
200	Response body

```
[{"id": 1, "nombre": "Botanas", "descripcion": "Sección de Frituras y dulces varios"}]
```

Download

4.- Actualizar Categoría.

PUT /categorias/{id} Actualizar categoría

Actualiza una categoría existente

Parameters

Name	Description
id * required	integer(\$int32) ID de la categoría (path)

4

Request body required

```
{"descripcion": "Art\u00edculos y comida para perros"}
```

application/json

Code Details

Code	Details
200	Response body

```
[{"id": 4, "nombre": "Mascotas", "descripcion": "Art\u00edculos y comida para perros"}]
```

Download

5.- Eliminar Categoría

DELETE /categorias/{id} Eliminar categoría

Elimina una categoría por su ID

Parameters

Name	Description
id * required	integer(\$int32) ID de la categoría (path)

4

Code **Details**

204 *Undocumented*

Response headers

```
connection: keep-alive
date: Mon, 22 Oct 2023 03:47:24 GMT
keep-alive: timeout=60
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
```

Revisión en obtener todas las categorías para ver que ya no este la categoria.

200

Response body

```
[{"id": 1, "nombre": "Botanas", "descripcion": "Sección de Frituras y dulces varios"}, {"id": 2, "nombre": "Panadería", "descripcion": "dónas y panes"}, {"id": 3, "nombre": "Health Care", "descripcion": "Artículos de cuidado personal"}]
```

6.- Obtener producto por id.

GET /productos/{id} Obtener producto por ID

Retorna un producto por su ID

Parameters

Name	Description
id * required	integer(\$int32) ID del producto (path)

5

Code **Details**

200

Response body

```
{"id": 5, "nombre": "Chorros", "descripcion": "rollos de chocolate", "precio": 20, "stock": 3, "categoria": {"id": 2, "nombre": "Panadería", "descripcion": "dónas y panes"}}
```

7.- Actualizar producto.

Antes de actualizar:

```
{  
    "id": 5,  
    "nombre": "Chocorroles",  
    "descripcion": "roles de chocolate",  
    "precio": 24,  
    "stock": 3,  
    "categoria": {  
        "id": 2,  
        "nombre": "Panaderia",  
        "descripcion": "doras y panes"  
    }  
}
```

Usando el endpoint de actualizar producto por id

The screenshot shows a REST API tool interface. At the top, it says "PUT /productos/{id} Actualizar producto". Below that, it says "Actualiza un producto existente". Under "Parameters", there is a table with one row. The "Name" column has "id * required" and the "Description" column has "integer(\$int32) ID del producto (path)". The value "5" is entered in the input field. Under "Request body" (required), the content type is set to "application/json". The JSON body is: { "descripcion": "Pan de chocolate en forma de rollo" }. In the response section, the status code is 200, and the "Response body" is identical to the original product object with the updated description.

8.- Obtener todos los registros

The screenshot shows a REST API tool interface. At the top, it says "GET /registros Obtener todos los registros". Below that, it says "Recupera una lista de todos los registros de stock". Under "Server response", there is a table with two columns: "Code" and "Details". The "Code" column has "200" and the "Details" column has "Response body". The response body is a list of four objects, each representing a record with fields: id, productoId, cantidad, tipo, and fecha. The first record has id 3, productoId 1, cantidad 120, tipo "entrada", and fecha "2025-12-06T18:14:13.076075". The second record has id 4, productoId 1, cantidad 101, tipo "salida", and fecha "2025-12-06T18:14:53.052514". The third record has id 5, productoId 5, cantidad 20, tipo "SALIDA", and fecha "2025-12-10T20:48:48.888397". The fourth record has id 6, productoId 5, cantidad 10, tipo "SALIDA", and fecha "2025-12-10T20:48:48.888397".

9.- Obtener registro por producto

GET /registros/producto/{productId} Obtener registros por producto

Retorna todos los registros de un producto específico

Parameters

Name	Description
productId * required	integer(\$int32) (path)

Value: 1

Cancel

200 Response body

```
[{"id": 2, "productId": 1, "cantidad": 10, "tipo": "entrada", "fecha": "2025-12-06T18:14:13.076075"}, {"id": 4, "productId": 1, "cantidad": 10, "tipo": "salida", "fecha": "2025-12-06T18:14:53.052514"}, {"id": 11, "productId": 1, "cantidad": 10, "tipo": "SALIDA", "fecha": "2025-12-10T20:53:39.979004"}, {"id": 35, "productId": 1, "cantidad": 10, "tipo": "Entrada", "fecha": "2025-12-10T20:53:39.979004"}]
```

10.- Crear nuevo registro.

Se hará un registro de ingreso y uno de salida, el producto actualmente es el siguiente:

200 Response body

```
{"id": 2, "nombre": "Cheetos", "descripcion": "Cheetos torcidos", "precio": 12, "stock": 24, "categoria": [{"id": 1, "nombre": "Botanas", "descripcion": "Sección de Frituras y dulces varios"}]}
```

Download

Registro de entrada.

POST /registros Crear registro

Crea un nuevo registro de stock

Parameters

No parameters

Request body * required

application/json

```
{ "id": 0, "productId": 2, "cantidad": 11, "tipo": "ENTRADA", "fecha": "2025-12-12T04:12:23.638Z"}
```

200 Response body

```
{"id": 21, "productId": 2, "cantidad": 11, "tipo": "ENTRADA", "fecha": "2025-12-11T22:14:48.3372149"}
```

Download

Revisando el producto con el endpoint de obtener producto ha obtenido 11 mas de stock.

```
{  
    "id": 2,  
    "nombre": "Cheetos",  
    "descripcion": "Cheetos torcidos",  
    "precio": 21,  
    "stock": 35,  
    "categoria": {  
        "id": 1,  
        "nombre": "Botanas",  
        "descripcion": "Sección de Frituras y dulces varios"  
    }  
}
```



Download

Registro de Salida.

POST /registros Crear registro

Crea un nuevo registro de stock

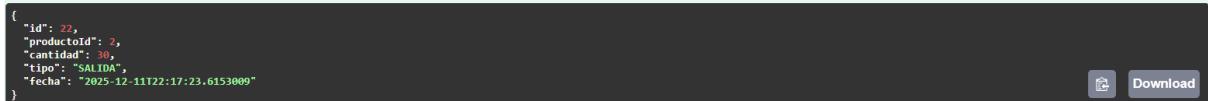
Parameters

No parameters

Request body required

application/json

```
{  
    "id": 0,  
    "productId": 2,  
    "cantidad": 30,  
    "tipo": "SALIDA",  
    "fecha": "2025-12-12T04:17:06.507Z"  
}
```



Cancel Reset

Download

Stock actualizado después de la salida de 30 de stock., ahora solo le quedan 5.

```
{  
    "id": 2,  
    "nombre": "Cheetos",  
    "descripcion": "Cheetos torcidos",  
    "precio": 21,  
    "stock": 5,  
    "categoria": {  
        "id": 1,  
        "nombre": "Botanas",  
        "descripcion": "Sección de Frituras y dulces varios"  
    }  
}
```



Download

FRONTEND

Al frontend que es la interfaz gráfica donde el usuario podrá utilizar las operaciones CRUD de manera visual se podrá acceder desde <http://localhost:8080/>, donde se desplegará la página web.

En esta página se encuentran varios apartados como Dashboard que muestra métricas básicas de nuestro inventario, Categorías donde se muestran las categorías que tenemos y nos deja agregar una nueva o modificarla o eliminarla.

Productos que nos muestra la tabla de productos con varios filtros disponibles y también las operaciones CRUD de manera visual usando formularios, y el registro que nos muestra las entradas y salidas de stock con filtros y paginación.

The screenshot shows the 'Dashboard' section of the application. At the top, there are five colored boxes displaying summary data: 'Total Productos' (5), 'Total Categorías' (3), 'Total Registros' (15), 'Ingresos Hoy' (0), and 'Salidas Hoy' (30). Below these, a yellow box highlights 'Productos con Stock Bajo' (3). A table titled 'Productos con Stock Bajo' lists three items: Cheetos (Stock 5, Precio \$21.00), Sabritas (Stock 8, Precio \$25.00), and Chocorroles (Stock 3, Precio \$24.00).

En el dashboard podemos ver una tabla con productos con bajo stock y métricas como el total de productos, categorías, registros , la cantidad de productos con bajo stock y la cantidad de ingresos y salidas de stock del día.

Tenemos el apartado de categorías donde se muestran las categorías que tenemos y nos permite agregar nuevas categorías, editarlas y también eliminarlas (siempre que no tenga productos).

The screenshot shows the 'CATEGORÍAS' (Categories) page. It displays a table with three rows: 'Botanas' (Descripción: Sección de Frituras y dulces varios), 'Panadería' (Descripción: donas y panes), and 'Health Care' (Descripción: Artículos de cuidado personal). Each row has edit and delete icons in the 'Acciones' column. At the bottom, there are pagination controls for 'Items per page' (10) and '1 – 3 of 3'.

The screenshot shows the 'CATEGORÍAS' page with a modal window titled 'Nuevo categoría' (New Category). The form fields are 'Nombre*' (Name*) with 'Farmacia' entered and 'Descripción' (Description) with 'Medicinas y curaciones' (Medicines and treatments) entered. At the bottom of the modal are 'Cancelar' (Cancel) and 'Guardar' (Save) buttons. The background table remains the same as in the previous screenshot.

CATEGORÍAS				
Nueva categoría				
ID	Nombre	Descripción	Acciones	
1	Botanas	Sección de Frituras y dulces varios		
2	Panadería	dones y panes		
6	Health Care	Artículos de cuidado personal		
7	Farmacia	Medicinas y curaciones		

Items per page: 10 | < < > >| 1 – 4 of 4

INVENTARIO

[Dashboard](#) [Productos](#) [Categorías](#) [Registros](#)

CATEGORÍAS

[Nueva categoría](#)

ID	Nombre	Descripción	Acciones	
1	Botanas	Sección de Frituras y dulces varios		
2	Panadería	dones y panes		
6	Health Care	Artículos de cuidado personal		
7	Farmacia	Medicinas y curaciones		

Editar categoría

Nombre*
Panadería

Descripción
Variedad de panes y galletas

Cancelar **Guardar**

Items per page: 10 | < < > >| 1 – 4 of 4

2 Panadería Variedad de panes y galletas

localhost:8080/categorias

INVENTARIO

[Dashboard](#) [Productos](#) [Categorías](#) [Registros](#)

CATEGORÍAS

[Nueva categoría](#)

ID	Nombre	Descripción	Acciones	
1	Botanas	Sección de Frituras y dulces varios		
2	Panadería	Variedad de panes y galletas		

localhost:8080 dice

Eliminar categoría?

Aceptar **Cancelar**

En el apartado de productos podemos encontrar una tabla de todos nuestros productos donde vemos id, nombre, descripción, precio, stock, categoría y un apartado de las acciones disponibles donde se puede agregar o quitar stock (creando a la vez el registro), editar y eliminar el producto.

Así también tenemos un botón para agregar nuevos productos, y del lado izquierdo tenemos un apartado de filtros donde podemos buscar por nombre, precio mínimo o máximo así como por cantidad de stock , además de un botón para limpiar filtros , además de la paginación de la tabla.

Agregar nuevo producto:

NUEVO PRODUCTO

ID	Nombre	Descripción	Precio	Stock	Categoría	Acciones
1	Doritos	Botana doritos nacho	\$20.00	26	Botanas	
2	Cheetos	Cheetos torcidos	\$21.00	5	Botanas	
3	Sabritas	sabritas de sal	\$25.00	8	Botanas	
5	Chocorrolles	Pan de chocolate en forma de rollo	\$24.00	3	Panadería	
6	Chips Ahoy	galletas con chispas de chocolate	\$18.00	10	Panadería	

Items per page: 10 | 1 - 5 of 5

FILTROS

Limpiar

INVENTARIO **Dashboard** **Productos** **Categorías** **Registros**

PRODUCTOS

Editar producto:

EDITAR PRODUCTO

ID	Nombre	Descripción	Precio	Stock	Categoría	Acciones
1	Doritos	Botana doritos nacho	\$20.00	26	Botanas	
2	Cheetos	Fritura de maíz con queso y chile	\$21.00	5	Botanas	
3	Sabritas	sabritas de sal	\$25.00	8	Botanas	
5	Chocorrolles	Pan de chocolate en forma de rollo	\$24.00	3	Panadería	
6	Chips Ahoy	galletas con chispas de chocolate	\$18.00	10	Panadería	
9	Paracetamol	Medicamento utilizado para tratar dolores leves	\$19.00	33	Farmacia	

Items per page: 10 | 1 - 6 of 6

FILTROS

Limpiar

INVENTARIO **Dashboard** **Productos** **Categorías** **Registros**

PRODUCTOS

2 Cheetos Fritura de maíz con queso y chile \$21.00 5 Botanas

Eliminar Producto:

A screenshot of a web browser displaying a product management interface. The title bar shows 'localhost:8080/productos'. A modal dialog box is centered, asking 'localhost:8080 dice' (localhost:8080 says) 'Eliminar producto?' (Delete product?). There are 'Aceptar' (Accept) and 'Cancelar' (Cancel) buttons. The main content area shows a table titled 'Nuevo producto' (New product) with columns: ID, Nombre, Descripción, Precio, Stock, Categoría, and Acciones. The table contains three rows of data: Doritos, Cheetos, and Sabritas. Each row has edit and delete icons in the Acciones column.

Filtrado de productos:

Three screenshots illustrating product filtering:

- Screenshot 1:** Shows a search filter set to 'CH'. The results table shows items: Cheetos, Chororroles, and Chips Ahoy.
- Screenshot 2:** Shows a category filter set to 'Botanas'. The results table shows items: Doritos, Cheetos, and Sabritas.
- Screenshot 3:** Shows a more complex filter with 'Precio min' set to 11 and 'Stock min' set to 11. The results table shows items: Doritos and Paracetamol.
- Screenshot 4:** Shows a filter with 'Precio max' set to 20. The results table shows items: Doritos, Chips Ahoy, and Paracetamol.

The interface includes a 'Nuevo producto' (New product) button at the top of each table. Each table has columns: ID, Nombre, Descripción, Precio, Stock, Categoría, and Acciones. The tables show data corresponding to the filters applied.

El apartado de registros nos muestra una tabla de los registros donde se ve el tipo de movimiento y la fecha del mismo, se cuentan con las acciones CRUD así como con un filtrado por fecha y por tipo de movimiento.

The screenshots illustrate the 'REGISTROS' (Registers) page of an inventory management system, showing various filter configurations and data entries.

Screenshot 1: Shows a general view of the registers. The filters are set to 'Entrada/Ingreso' with 'Desde' (From) and 'Hasta' (To) dates. The table lists 8 entries, all of which are ENTRADA (Entry) type. The last entry is highlighted.

ID	Producto ID	Cantidad	Tipo	Fecha	Acciones
3	1	120	entrada	12/6/25, 6:14 PM	
9	6	8	ENTRADA	12/10/25, 8:50 PM	
13	2	9	ENTRADA	12/10/25, 9:45 PM	
14	2	5	ENTRADA	12/10/25, 9:58 PM	
16	1	12	ENTRADA	12/10/25, 10:27 PM	

Screenshot 2: Shows a new entry being created. The 'Nuevo registro' (New registration) dialog is open, prompting for 'Producto*' (Product*), 'Cantidad*' (Quantity*), and 'Tipo*' (Type*). The product selected is Doritos, quantity is 20, and type is Entrada (Entry).

ID	Producto ID	Cantidad	Tipo	Fecha	Acciones
3	1			12/6/25, 6:14 PM	
9	6			12/10/25, 8:50 PM	
13	2			12/10/25, 9:45 PM	
14	2			12/10/25, 9:58 PM	
16	1			12/10/25, 10:27 PM	

Screenshot 3: Shows a view of the registers with a 'Salida' (Output) filter applied. The table lists 5 entries, all of which are SALIDA (Output) type. The last entry is highlighted.

ID	Producto ID	Cantidad	Tipo	Fecha	Acciones
4	1	101	salida	12/6/25, 6:14 PM	
8	5	20	SALIDA	12/10/25, 8:48 PM	
10	6	10	SALIDA	12/10/25, 8:51 PM	
11	1	10	SALIDA	12/10/25, 8:53 PM	
12	2	20	SALIDA	12/10/25, 9:45 PM	

Screenshot 4: Shows a view of the registers with a 'Todos' (All) filter applied, spanning from 12/9/2025 to 12/11/2025. The table lists 5 entries, alternating between ENTRADA (Entry) and SALIDA (Output) types. The last entry is highlighted.

ID	Producto ID	Cantidad	Tipo	Fecha	Acciones
8	5	20	SALIDA	12/10/25, 8:48 PM	
9	6	8	ENTRADA	12/10/25, 8:50 PM	
10	6	10	SALIDA	12/10/25, 8:51 PM	
11	1	10	SALIDA	12/10/25, 8:53 PM	
12	2	20	SALIDA	12/10/25, 9:45 PM	

CONCLUSIÓN

Este proyecto final de sistema de inventario me sirvió para integrar y aplicar conocimientos de varias UEA's que he cursado a lo largo de la carrera como sistemas distribuidos, bases de datos, web estático y ahora web dinámico.

Además este proyecto me sirvió para explorar más a fondo un perfil full-stack ya que estuve combinando tanto la parte visual e interactiva del frontend como la lógica, reglas de negocio del backend. Trabajar con ambos lados al mismo tiempo me ayudó a entender mejor cómo se comunican, cómo se diseñan flujos limpios entre capas y qué responsabilidades tiene cada sección del sistema. En conclusión fue una experiencia que reforzó mis habilidades y me dio una visión más completa de lo que implica desarrollar software profesional de principio a fin.

URL de proyecto en GITHUB: https://github.com/MarinoX/proyecto_web_dinamico