



UNIVERSIDAD  
**NACIONAL**  
DE COLOMBIA

TALLER 01:  
REQUERIMIENTOS

Ingeniería de Software I

Wulffredo Javier Barco Godoy - wbarco@unal.edu.co  
Brahian Camilo Gómez Carvajal - bgomezca@unal.edu.co  
Juan David Rivera Buitrago - jriverabu@unal.edu.co  
William Darío Vanegas Marín - wvanegas@unal.edu.co

Profesor  
Oscar Eduardo Alvarez Rodríguez

Universidad Nacional de Colombia  
Facultad de Ingeniería  
Ingeniería de Sistemas y Computación  
Bogotá, Colombia  
20 de Diciembre de 2024

La conversación elegida por el grupo es la [conversación GP2](#).

## **Levantamiento de Requerimientos**

La emprendedora, propietaria de un pequeño negocio dedicado a la venta de artesanías y productos decorativos, enfrenta varios desafíos operativos debido a la falta de un sistema organizado. Actualmente, gestiona su inventario en hojas de cálculo que rara vez actualiza, coordina pedidos a través de WhatsApp e Instagram, y lleva un registro manual de pagos y entregas en una libreta. Este método le genera confusión al organizar los pedidos, controlar el inventario y mantenerse al día con las consultas de los clientes, lo que afecta directamente sus ventas. Además, carece de una plataforma centralizada donde los clientes puedan explorar su catálogo, lo que incrementa la carga de trabajo al tener que enviar fotos manualmente.

Con el objetivo de resolver estos problemas, se propone un sistema basado en una página web responsiva que permita a la emprendedora gestionar pedidos, actualizar el inventario automáticamente y brindar a los clientes un catálogo en línea. Entre las funcionalidades prioritarias se destacan: el registro de pedidos como tareas con estados claros (pendiente, pagado, completado), la automatización de las notificaciones de stock bajo y confirmaciones de pedidos mediante WhatsApp, y un diseño intuitivo adaptado a dispositivos móviles. A futuro, el sistema será escalable, incorporando herramientas de gestión financiera y promoción. La emprendedora, con un presupuesto inicial limitado y urgencia para implementar, prioriza soluciones prácticas que optimicen sus operaciones sin complicaciones técnicas innecesarias.

Del planteamiento anterior, se detalla lo siguiente:

### **1. Objetivo:**

Desarrollar una aplicación web para gestionar ventas, inventario y catálogo de productos para una pequeña empresa de artesanías y decoración.

### **2. Funcionalidades principales:**

#### ***2.1 Gestión de Productos***

- Registro de productos con los siguientes atributos:
  - Nombre del producto.
  - Precio.
  - Cantidad en stock.
  - Descripción breve.
  - Imágenes del producto.
- Capacidad de añadir, editar y eliminar productos.
- Seguimiento automático de inventario.
- Notificaciones de stock bajo (vía WhatsApp).

#### ***2.2 Gestión de Pedidos***

- Sistema de gestión de pedidos como tareas
- Atributos de pedidos:
  - Número de pedido único.
  - Fecha del pedido.

- Productos solicitados.
- Cantidades.
- Estado del pedido (Pendiente, Pagado, Enviado, Completado, Cancelado).
- Fecha límite de entrega.
- Monto total.
- Medio de pago.

### **2.3 Gestión de Clientes**

- Registro de información de clientes:
  - Nombre completo.
  - Número de contacto.
  - Dirección de envío.
  - Historial de pedidos.
- Capacidad de modificar información de clientes.

### **2.4 Interfaz de Usuario**

- Diseño responsivo (móvil y computadora).
- Acceso web sin necesidad de descargar aplicación.
- Panel de administración para gestión de productos, pedidos e inventario.
- Vista de catálogo para clientes.
- Notificaciones de pedidos por WhatsApp para clientes.

### **2.5 Características Adicionales**

- Mini calculadora de envío.
- Sección de comentarios y opiniones de clientes.
- Opción de personalización de productos.
- Resumen de ventas e inventario.

## **3. Requerimientos técnicos:**

### **3.1 Requisitos de Diseño**

- Paleta de colores:
  - Color principal: Beige claro/crema.
  - Color secundario: Terracota o salmón suave.
  - Colores de fondo: Gris suave o blanco.
  - Detalles: Verde oliva o musgo.
- Tipografía: Montserrat o Poppins.
- Diseño limpio, minimalista y fácil de usar.

### **3.2 Requisitos de Integración**

- Integración con WhatsApp para notificaciones.
- Posible integración con pasarelas de pago locales (Nequi, Daviplata).
- Compatibilidad con dispositivos móviles y computadoras.

## **4. Requerimientos no funcionales:**

- Rendimiento: Carga rápida, especialmente en móviles.
- Seguridad: Acceso restringido, sólo visible para administrador.
- Escalabilidad: Capacidad de agregar funciones en el futuro.
- Usabilidad: Interfaz intuitiva, máximo 2 clics para realizar acciones.

#### **5. Consideraciones de presupuesto:**

- Presupuesto inicial: Aproximadamente 1,000,000 COP.
- Desarrollo en etapas.
- Priorización de funcionalidades esenciales.

#### **6. Cronograma:**

- Implementación inicial: 1 mes.
- Desarrollo incremental de funcionalidades adicionales.

Las prioridades iniciales serían:

- Gestión de pedidos y actualización de inventario.
- Catálogo en línea vinculado al sistema de inventario.
- Notificaciones a clientes y administrador.



## **Análisis de Requerimientos con el método Moscow**

### ***Must have (Obligatorio)***

#### **1. Gestión de Inventario**

- **Tiempo estimado:** 2 semanas
- **Complejidad:** Alta
- **Recursos:**
  - 1 desarrollador backend
  - 1 desarrollador frontend
- **Tecnologías:**
  - Backend: Node.js con Express
  - Base de datos: MongoDB
  - ORM: Mongoose

#### **2. Catálogo de Productos**

- **Tiempo estimado:** 1.5 semanas
- **Complejidad:** Media
- **Recursos:**
  - 1 desarrollador frontend
- **Tecnologías:**
  - React.js
  - Tailwind CSS

### ***Should Have (Importante)***

#### **1. Gestión de Pedidos**

- **Tiempo estimado:** 2 semanas
- **Complejidad:** Alta
- **Recursos:**
  - 1 desarrollador backend
  - 1 desarrollador frontend
- **Tecnologías:**
  - GraphQL
  - Apollo Server

#### **2. Notificaciones por WhatsApp**

- **Tiempo estimado:** 1 semana
- **Complejidad:** Media
- **Recursos:**
  - 1 desarrollador especialista en integraciones
- **Tecnologías:**
  - Twilio API
  - WhatsApp Business API

### ***Could Have (Deseable)***

#### **1. Mini Calculadora de Envío**

- **Tiempo estimado:** 1 semana
- **Complejidad:** Baja

- **Recursos:**
  - 1 desarrollador frontend
- **Tecnologías:**
  - JavaScript
  - Geolocalización API

## **2. Sistema de Comentarios de Clientes**

- **Tiempo estimado:** 1 semana
- **Complejidad:** Baja
- **Recursos:**
  - 1 desarrollador frontend
- **Tecnologías:**
  - Firebase Authentication
  - Firebase Realtime Database

### ***Won't Have (No incluido en este momento)***

1. Integración de Pasarela de Pagos.
2. Sistema de Asesorías y Agenda.
3. Análisis Financiero Avanzado.

## **Estimación General del Proyecto**

### ***Recursos Humanos***

- **Total desarrolladores:** 4-5
- **Perfiles:**
  - 2 desarrolladores backend.
  - 2 desarrolladores frontend.
  - 1 desarrollador especialista en integraciones.

### ***Tiempo de Desarrollo***

- **Duración total:** 8-10 semanas.
- **Desglose:**
  - Planificación: 1 semana.
  - Desarrollo core (Must Have): 4-5 semanas.
  - Desarrollo complementario (Should Have): 2-3 semanas.
  - Pruebas y ajustes: 1-2 semanas.

### ***Costos Estimados***

- **Desarrollo:**
  - Costo por desarrollador junior/semi-senior: \$3,000,000 - \$5,000,000 COP/mes.
  - Total mensual equipo: \$12,000,000 - \$20,000,000 COP.
- **Infraestructura:**
  - Servicios en la nube: \$500,000 - \$1,000,000 COP/mes.
  - Licencias y herramientas: \$300,000 - \$600,000 COP/mes.

### ***Presupuesto Total Estimado***

- **Rango:** \$15,000,000 - \$25,000,000 COP.
- **Comparativo con presupuesto inicial:** Supera el millón de pesos original.
  - Necesidad de buscar financiamiento adicional.
  - Considerar desarrollo por etapas.

Otras consideraciones adicionales están representados por los siguientes aspectos:

### ***Riesgos***

- Complejidad de integración con WhatsApp
- Cambios en requisitos durante desarrollo
- Limitaciones presupuestarias

### ***Mitigación de Riesgos***

- Metodología ágil (Scrum)
- Sprints cortos
- Revisiones frecuentes con cliente
- Prototipado rápido

### ***Stack Tecnológico Propuesto***

- **Frontend:** React.js
- **Backend:** Node.js con GraphQL
- **Base de Datos:** MongoDB
- **Infraestructura:** AWS o DigitalOcean
- **Comunicaciones:** Twilio
- **Diseño:** Tailwind CSS





## Diagrama y especificación de casos de uso

### Diagrama de casos de uso - Administrador

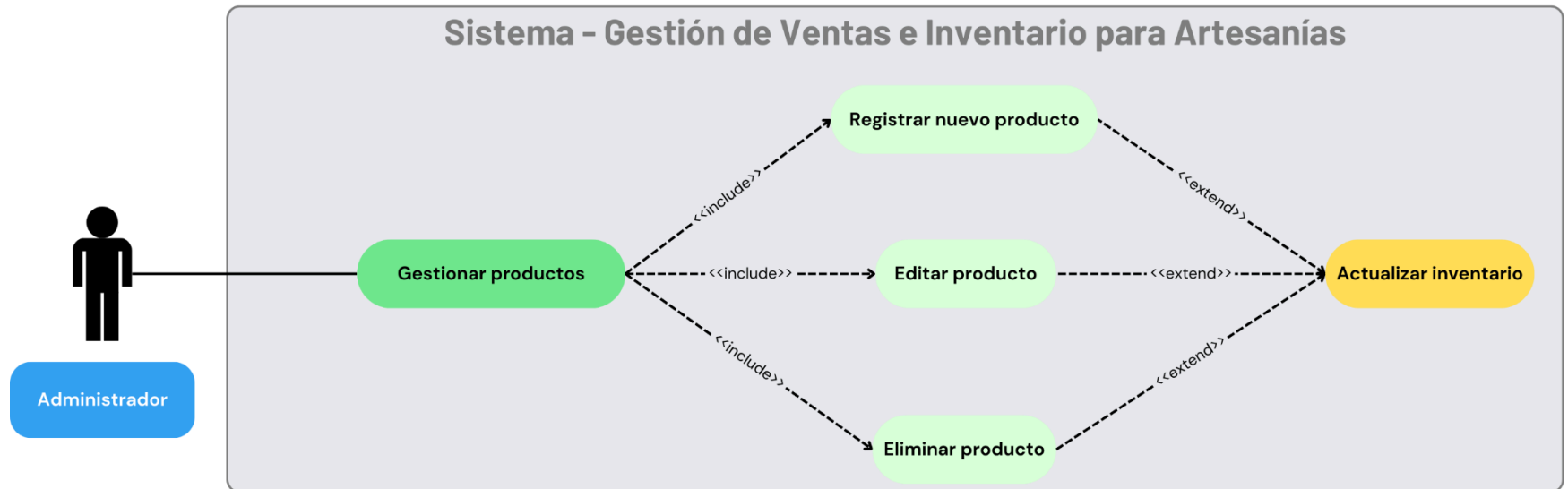


Imagen 1. Diagrama de casos de uso del administrador con base en el caso de uso "Gestionar productos".

## Diagrama de casos de uso - Usuario

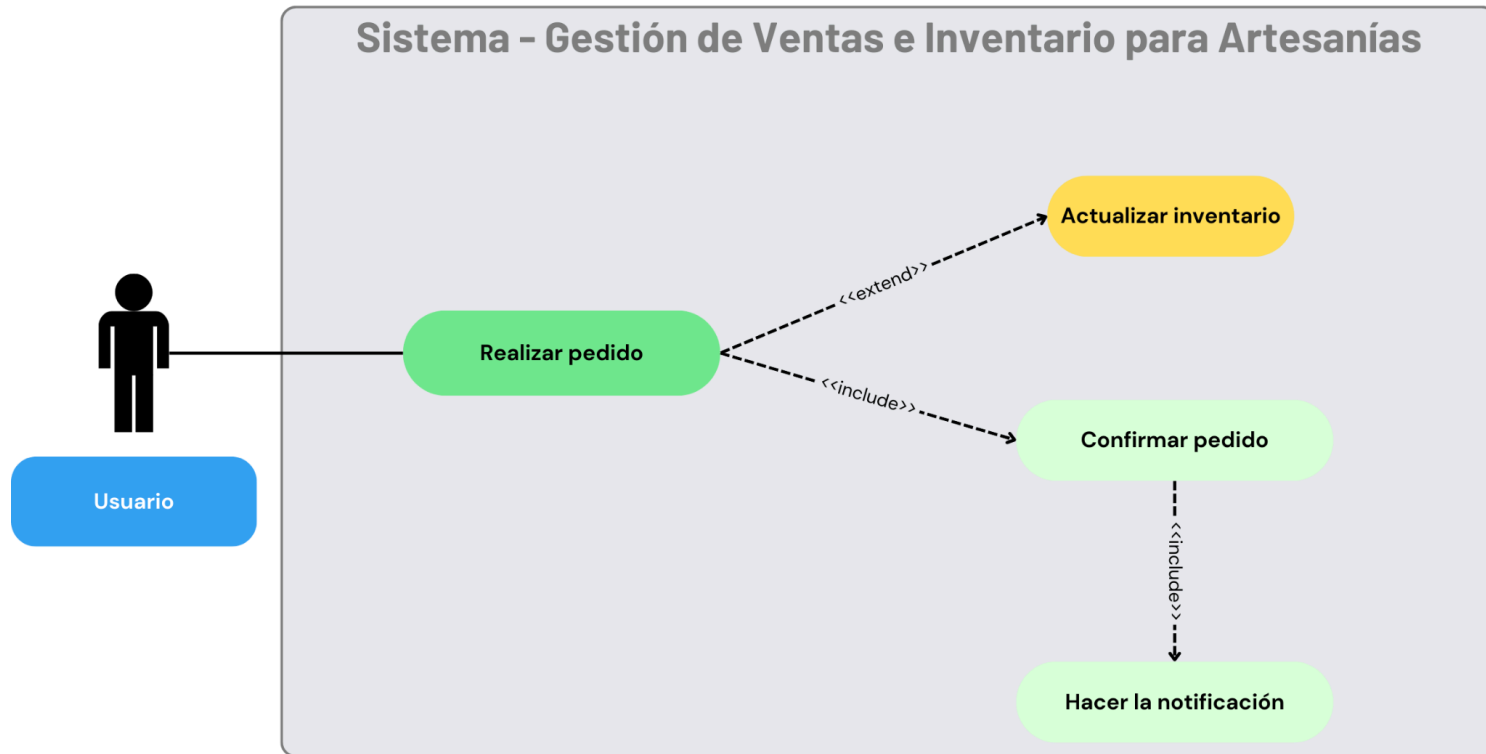


Imagen 2. Diagrama de casos de uso del usuario con base en el caso de uso "Realizar pedido".

## Diagrama de casos de uso



Imagen 3. Diagrama de casos de uso integrado del administrador y el usuario con base en “Realizar pedido” y “Gestionar productos”.

<b>Gestionar productos del inventario</b>
<b>Actor</b> Administrador
<b>Descripción</b> Permite al administrador registrar, editar o eliminar productos en el sistema.
<b>Precondiciones</b> <ul style="list-style-type: none"> <li>• El administrador debe estar autenticado en el sistema.</li> <li>• Debe existir conexión con la base de datos.</li> </ul>
<b>Flujo de eventos</b> <ol style="list-style-type: none"> <li>1. El administrador accede al panel de gestión de productos.</li> <li>2. El administrador selecciona alguna de las siguientes opciones: <ol style="list-style-type: none"> <li>a) Registrar nuevo producto. <ol style="list-style-type: none"> <li>I. El administrador ingresa la información necesaria de los detalles de un producto: <ul style="list-style-type: none"> <li>• Nombre.</li> <li>• Descripción.</li> <li>• Precio.</li> <li>• Cantidad disponible.</li> <li>• Imágenes del producto.</li> </ul> </li> <li>II. El sistema confirma la acción de ingreso de la información.</li> <li>III. El sistema guarda la información en la base de datos.</li> </ol> </li> <li>b) Editar información de producto existente. <ol style="list-style-type: none"> <li>I. El administrador selecciona uno de los productos de la lista extraída.</li> <li>II. El cliente selecciona y/o edita la información dependiendo de los cambios que desee realizar según: <ul style="list-style-type: none"> <li>• Nombre.</li> <li>• Descripción.</li> <li>• Precio.</li> <li>• Cantidad disponible.</li> <li>• Imágenes del producto.</li> </ul> </li> <li>III. El sistema confirma la acción de ingreso de la información.</li> <li>IV. El sistema guarda la información en la base de datos.</li> </ol> </li> <li>c) Eliminar producto. <ol style="list-style-type: none"> <li>I. El administrador selecciona uno de los productos de la lista extraída.</li> <li>II. El administrador confirma la eliminación de la información del producto seleccionado.</li> <li>III. El sistema elimina la información de la base de datos.</li> </ol> </li> </ol> </li> </ol>
<b>Postcondiciones</b> <ul style="list-style-type: none"> <li>• El inventario se actualiza correctamente.</li> <li>• Los cambios son visibles en el catálogo de productos.</li> </ul>

<b>Realizar pedido</b>
<b>Actor</b> Usuario
<b>Descripción</b> Permite seleccionar productos del catálogo, especificar las cantidades requeridas de cada producto y confirmar el pedido.
<b>Precondiciones</b> <ul style="list-style-type: none"> <li>• El usuario debe estar registrado.</li> <li>• El usuario ha iniciado sesión.</li> <li>• Debe haber productos disponibles en el inventario.</li> </ul>
<b>Flujo de eventos</b> <ol style="list-style-type: none"> <li>1. El usuario accede al inventario de productos por medio del catálogo mostrado.</li> <li>2. Selecciona los productos que se requieran y especifica las cantidades.             <ol style="list-style-type: none"> <li>a) Si no hay cantidad suficiente de productos disponibles:                 <ol style="list-style-type: none"> <li>I. El sistema notifica al usuario que no hay suficiente disponibilidad del o de los productos requeridos.</li> <li>II. El usuario ajusta la cantidad requerida o puede eliminar productos del pedido.</li> </ol> </li> </ol> </li> <li>3. El usuario revisa el resumen del pedido.             <ol style="list-style-type: none"> <li>a) Si el usuario lo requiere, puede ajustar el pedido (Volver al paso 2).</li> </ol> </li> <li>4. El usuario confirma el pedido realizado.</li> <li>5. El sistema almacena el pedido realizado.</li> <li>6. El sistema genera un código de pedido único.</li> <li>7. El sistema envía una confirmación al correo electrónico y al número de WhatsApp del usuario.</li> </ol>
<b>Postcondiciones</b> <ul style="list-style-type: none"> <li>• El pedido queda registrado en el sistema.</li> <li>• El inventario se actualiza de manera automática una vez se realiza el pedido.</li> <li>• Se envía una notificación al usuario y al administrador sobre el nuevo pedido.</li> </ul>



## Historias de Usuario

### Historia de Usuario #1: Gestión de Productos

#### Anexo de Documentos Relacionados:

- Sección 2.1 de este documento: Gestión de Productos.

#### Descripción conceptual

<b>Módulo</b>	<i>Módulo de Gestión de Productos</i>
<b>Descripción de la(s) funcionalidad(es) requerida(s):</b>	<i>Permite al administrador ingresar, editar o eliminar productos en el sistema de tal manera que el inventario se mantenga actualizado y que los productos del catálogo sean visibles para los usuarios.</i>

#### Descripción técnica

##### Backend

URL	Método	Código html
localhost:8080/products	POST	200 422
<b>Caso de uso técnico</b> Al ingresar el nuevo producto, debe retornar código 200 si se realiza de forma exitosa. Se retornará el código 422 si hay datos faltantes o si hay datos ingresados de manera equivocada.		
<b>Datos de entrada</b> <b>200:</b> { "nombre": "Cuadro Decorativo", "precio": 30000, "cantidad": 10, "descripcion": "Cuadro pintado a mano", "imagenes": ["cuadro1.jpg", "cuadro2.jpg"] }	<b>Datos de salida</b> <b>200:</b> { "status": "success", "data": { "producto_id": 123, "nombre": "Cuadro Decorativo" } }	
<b>422:</b> { "nombre": "Cuadro Decorativo", "precio": 30000,	<b>422:</b> { "status": "unprocessed", "message": "La cantidad debe ser un número	



<pre>"cantidad": x, "descripcion": "Cuadro pintado a mano", "imagenes": ["cuadro1.jpg", "cuadro2.jpg"] }</pre>	<pre>positivo" }</pre>
--	------------------------

URL	Método	Código html
localhost:8080/products	GET	200 422

**Caso de uso técnico**  
Al consultar, debe retornar código 200 y devolverá el listado con los datos de los productos disponibles.

Datos de entrada	Datos de salida
	<b>200:</b> <pre>{   "status": "success",   "data": [     {       "producto_id": 123,       "nombre": "Cuadro Decorativo",       "precio": 30000,       "cantidad": 10,       "descripcion": "Cuadro pintado a mano",       "imagenes": ["cuadro1.jpg", "cuadro2.jpg"]     },     {       "producto_id": 124,       "nombre": "Jarrón",       "precio": 20000,       "cantidad": 5,       "descripcion": "Jarrón artesanal",       "imagenes": ["jarron1.jpg"]     }   ] }</pre>
	<b>422:</b> <pre>{   "status": "unprocessed",   "message": "No fue posible obtener la lista de productos" }</pre>

	}
--	---

URL	Método	Código html
localhost:8080/products	PATCH	200 422
<b>Caso de uso técnico</b> Al actualizar el precio y la cantidad de un producto, debe retornar código 200 y devolver los datos del producto donde se evidencie el cambio en precio y cantidad con los datos ingresados anteriormente. Se retornará el código 422 si hay datos faltantes o si hay datos ingresados de manera equivocada.		
<b>Datos de entrada</b> <b>200:</b> <pre>{   "precio": 35000,   "cantidad": 8 }</pre>		<b>Datos de salida</b> <b>200:</b> <pre>{   "status": "success",   "data": {     "producto_id": 123,     "nombre": "Cuadro Decorativo",     "precio": 35000,     "cantidad": 8   } }</pre>
<b>422:</b> <pre>{   "precio": 35000,   "cantidad": x }</pre>		<b>422:</b> <pre>{   "status": "unprocessed",   "message": "La cantidad debe ser un número positivo" }</pre>

URL	Método	Código html
localhost:8080/products	DELETE	200 422
<b>Caso de uso técnico</b> Al eliminar un producto, debe retornar código 200 y un mensaje con la confirmación de que la eliminación fue realizada con éxito. Se retornará el código 422 si la eliminación no pudo efectuarse.		
<b>Datos de entrada</b>		<b>Datos de salida</b>

<b>200:</b> <pre>{   "nombre": "Cuadro Decorativo" }</pre>	<b>200:</b> <pre>{   "status": "success",   "message": "Producto eliminado correctamente" }</pre>
	<b>422:</b> <pre>{   "status": "error",   "message": "No fue posible eliminar el producto. Producto no fue encontrado." }</pre>

### Frontend

#### Interacción esperada:

- El administrador ingresa a la ventana donde están los productos mediante un botón del menú principal.
- El administrador puede registrar un producto nuevo completando el formulario con los datos requeridos.
- Los productos se enumeran en una tabla con opciones para editar o eliminar.

#### Mockups/Prototipos:

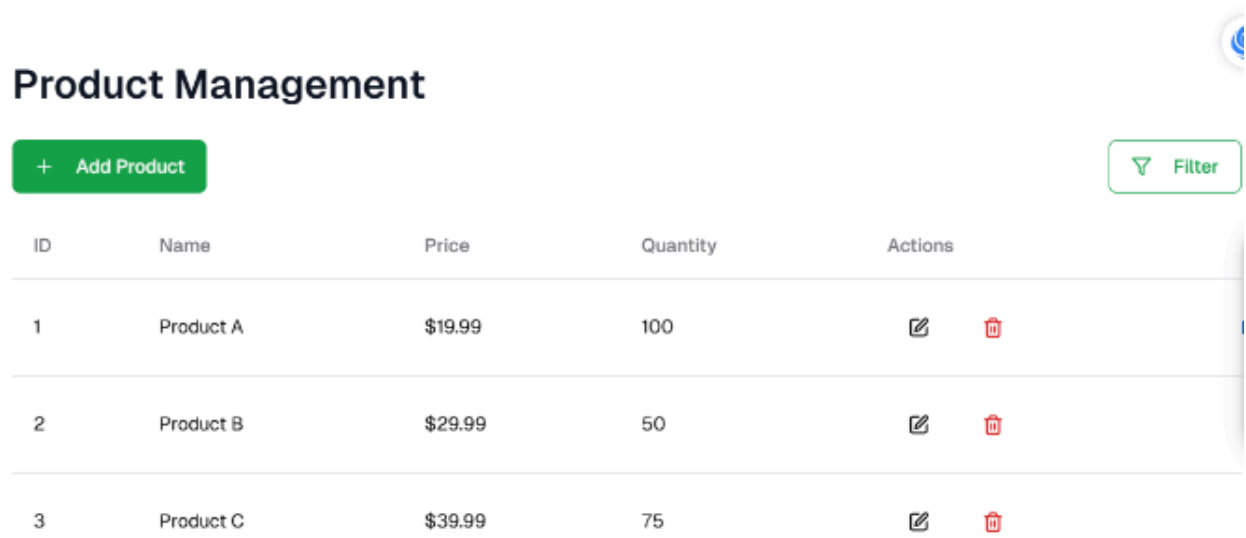
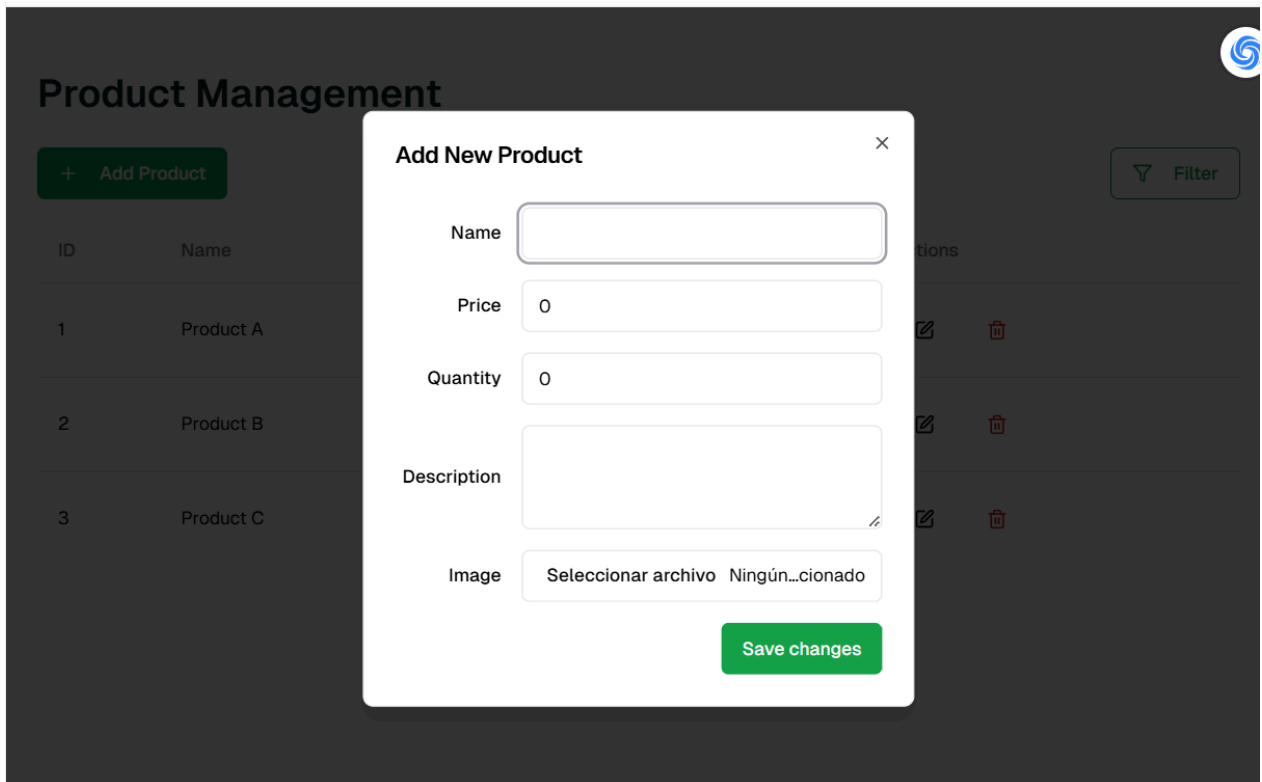


Imagen 4. Prototipo de ventana donde aparece el listado de productos.



The image shows a 'Product Management' interface. On the left, there is a table with columns 'ID' and 'Name'. It contains three rows: '1 Product A', '2 Product B', and '3 Product C'. Above the table is a green button with a plus icon and the text 'Add Product'. To the right of the table is a 'Filter' button. In the center, a modal window titled 'Add New Product' is open. It has a close button (X) in the top right corner. The modal contains the following fields: 'Name' (text input), 'Price' (text input with '0'), 'Quantity' (text input with '0'), 'Description' (text area), and 'Image' (file selection button with text 'Seleccionar archivo' and 'Ningún...cionado'). A green 'Save changes' button is at the bottom right of the modal. A small blue circular logo is in the top right corner of the main interface.

ID	Name
1	Product A
2	Product B
3	Product C

**Add New Product** X

Name

Price

Quantity

Description

Image

Seleccionar archivo Ningún...cionado

Save changes

Imagen 5. Prototipo de formulario emergente donde se agregan nuevos productos ingresando los campos requeridos.

### Flujo visual y eventos:

- Se realiza la selección de opción entre agregar, editar o eliminar productos.
- Aparece un formulario de manera emergente según la acción.
- Se realiza la actualización de la tabla de productos tras la acción.



## Historia de Usuario #2: Gestión de Pedidos

### Anexo de Documentos Relacionados:

- Sección 2.2 de este documento: Gestión de Pedidos.

### Descripción conceptual

<b>Módulo</b>	<i>Módulo de Gestión de Pedidos</i>
<b>Descripción de la(s) funcionalidad(es) requerida(s):</b>	<i>Sistema que permite registrar y gestionar pedidos con estados claros y notificaciones automáticas, bien sea para crear, actualizar y seguir los pedidos por medio de su estado</i>

### Descripción técnica

#### Backend

URL	Método	Código html
localhost:8080/orders	POST	200 422
<b>Caso de uso técnico</b> Verifica la disponibilidad de productos, registra el pedido con un estado inicial "Pendiente" y genera un código único.		
<b>Datos de entrada</b> <b>200:</b> { "cliente_id": 123, "productos": [{"producto_id": 123, "cantidad": 2}], "usuario_id": 456, "total": 60000 }	<b>Datos de salida</b> <b>200:</b> { "status": "success", "data": { "pedido_id": 789, "estado": "Pendiente" } }	
<b>422:</b> { "cliente_id": 123, "productos": [{"producto_id": 123, "cantidad": 2}], "usuario_id": 456, "total": 60000 }	<b>422:</b> { "status": "error", "message": "No fue posible realizar el pedido. La cantidad no fue definida." }	

--	--

URL	Método	Código html
localhost:8080/orders	GET	200
<b>Caso de uso técnico</b> Devuelve todos los pedidos registrados, incluyendo sus estados y productos asociados.		
<b>200:</b> <b>Datos de entrada</b>	<b>Datos de salida</b> <b>200:</b> <pre>{   "status": "success",   "data": [     { "pedido_id": 789, "productos": [{ "producto_id": 123, "cantidad": 2}], "estado": "Pendiente" }   ] }</pre>	

URL	Método	Código html
localhost:8080/orders	PATCH	200 422
<b>Caso de uso técnico</b> Actualiza el estado del pedido según las instrucciones recibidas.		
<b>200:</b> <b>Datos de entrada</b> <pre>{ "estado": "Pagado" }</pre>	<b>Datos de salida</b> <b>200:</b> <pre>{   "status": "success",   "data": { "pedido_id": 789, "estado": "Pagado" } }</pre>	
<b>422:</b> <b>Datos de entrada</b> <pre>{ "estado": "Pagadx" }</pre>	<b>Datos de salida</b> <b>422:</b> <pre>{   "status": "error",   "message": "No fue posible actualizar el estado del pedido." }</pre>	

## Frontend

### Interacción esperada:

- Se esperaría que haya un formulario de creación de pedidos desde el catálogo.
- Al realizar el pedido, debería aparecer un resumen de pedido antes de confirmar.

### Mockups/Prototipos:

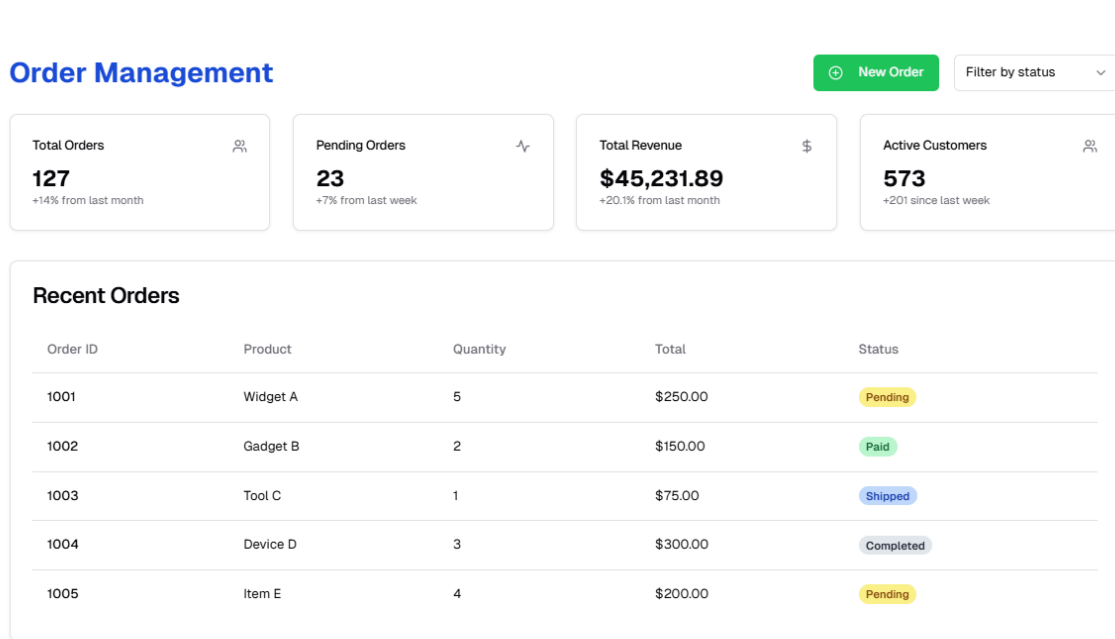


Imagen 6. Prototipo de formulario de creación de pedidos.

### Flujo visual y eventos:

- El usuario selecciona los productos del catálogo y la cantidad de cada uno.
- El usuario revisa el resumen de pedido en una pantalla posterior.
- El usuario confirma el pedido con un botón.
- Aparece una notificación de confirmación y se envía a su número de WhatsApp.





## Historia de Usuario #3: Gestión de Clientes

### Anexo de Documentos Relacionados:

- Sección 2.3 de este documento: Gestión de Clientes

### Descripción conceptual

<b>Módulo</b>	<i>Módulo de Gestión de Clientes</i>
<b>Descripción de la(s) funcionalidad(es) requerida(s):</b>	<i>Permite registrar y modificar información de clientes, incluyendo su historial de pedidos</i>

### Descripción técnica

#### Backend

URL	Método	Código html
localhost:8080/clientes	POST	200 422
<b>Caso de uso técnico</b> Se realiza el registro y la validación de los datos del cliente y guarda la información en la base de datos en caso exitoso. Se retornará el código 422 si hay datos faltantes o ingresados de manera equivocada.		
<b>Datos de entrada</b> <b>200:</b> { "nombre": "Carlos Pérez", "contacto": "3001234567", "direccion": "Calle 123" }	<b>Datos de salida</b> <b>200:</b> { "status": "success", "data": { "cliente_id": 101, "nombre": "Carlos Pérez" } }	
<b>422:</b> { "nombre": "Carlos Pérez", "contacto": "xxxxxxx", "direccion": "Calle 123" }	<b>422:</b> { "status": "unprocessed", "message": "El contacto debe ser de tipo numérico." }	

URL	Método	Código html
-----	--------	-------------

localhost:8080/clients	GET	200
<b>Caso de uso técnico</b> Recupera la lista de clientes registrados en la base de datos		
<b>200:</b> <b>Datos de entrada</b>	<b>200:</b> <b>Datos de salida</b> <pre>{   "status": "success",   "data": [     { "cliente_id": 101, "nombre": "Carlos Pérez",       "contacto": "1234567890", "direccion": "Calle 123" }   ] }</pre>	
<b>422:</b>	<b>422:</b> <pre>{   "status": "error",   "message": "No se pudieron obtener los productos" }</pre>	

URL	Método	Código html
localhost:8080/clients	PATCH	200
<b>Caso de uso técnico</b> Actualiza los datos del cliente según los parámetros proporcionados		
<b>200:</b> <b>Datos de entrada</b> <pre>{ "direccion": "Nueva Calle 456" }</pre>	<b>200:</b> <b>Datos de salida</b> <pre>{   "status": "success",   "data": { "cliente_id": 101, "direccion": "Nueva Calle 456" } }</pre>	
<b>422:</b> <pre>{ "direccion": " " }</pre>	<b>422:</b> <pre>{   "status": "unprocessed",   "message": "No hay dirección ingresada de forma correcta." }</pre>	

## Frontend

### Interacción esperada:

- Para este caso, se esperaría que haya un formulario para registrar los datos del cliente.
- Debería aparecer una tabla con la lista de clientes y su historial de pedidos.

### Mockups/Prototipos:

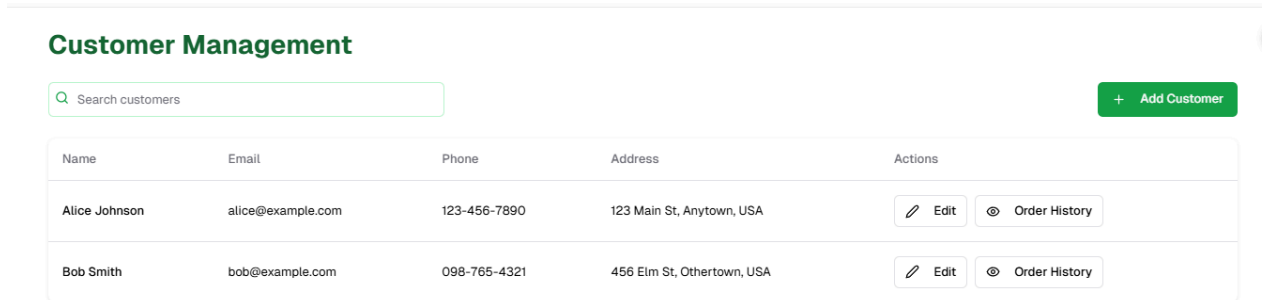


Imagen 7. Prototipo de ventana de gestión de clientes, donde aparece una tabla con el listado completo de clientes.

### Flujo visual y eventos:

- Se realiza la selección de cliente en una tabla.
- Posteriormente, se hace la edición de datos mediante un formulario de registro de clientes.
- Se hace la confirmación del registro por medio de los cambios de datos reflejados en la tabla.



## Historia de Usuario #4: Interfaz de Usuario

### Anexo de Documentos Relacionados:

- Sección 2.4 de este documento: Interfaz de Usuario

### Descripción conceptual

<b>Módulo</b>	<i>Módulo de Interfaz de Usuario</i>
<b>Descripción de la(s) funcionalidad(es) requerida(s):</b>	<i>Proporciona una plataforma responsiva para la administración de productos, pedidos y clientes</i>

### Descripción técnica

#### Backend

URL	Método	Código html
localhost:8080/ui	GET	200
<b>Caso de uso técnico</b> Proporciona los recursos necesarios para la interfaz de usuario.		
<b>200:</b> <b>Datos de entrada</b>	<b>Datos de salida</b> <b>200:</b> { "status": "success", "data": "Recursos cargados correctamente" }	
<b>422:</b>	<b>422:</b> { "status": "error", "message": "No se cargaron los recursos correctamente." }	

#### Frontend

### Interacción esperada:

- Se esperaría una ventana con un panel de administración que contenga acceso a módulos.
- Debería haber un menú lateral para navegar entre módulos.

### Mockups/Prototipos:

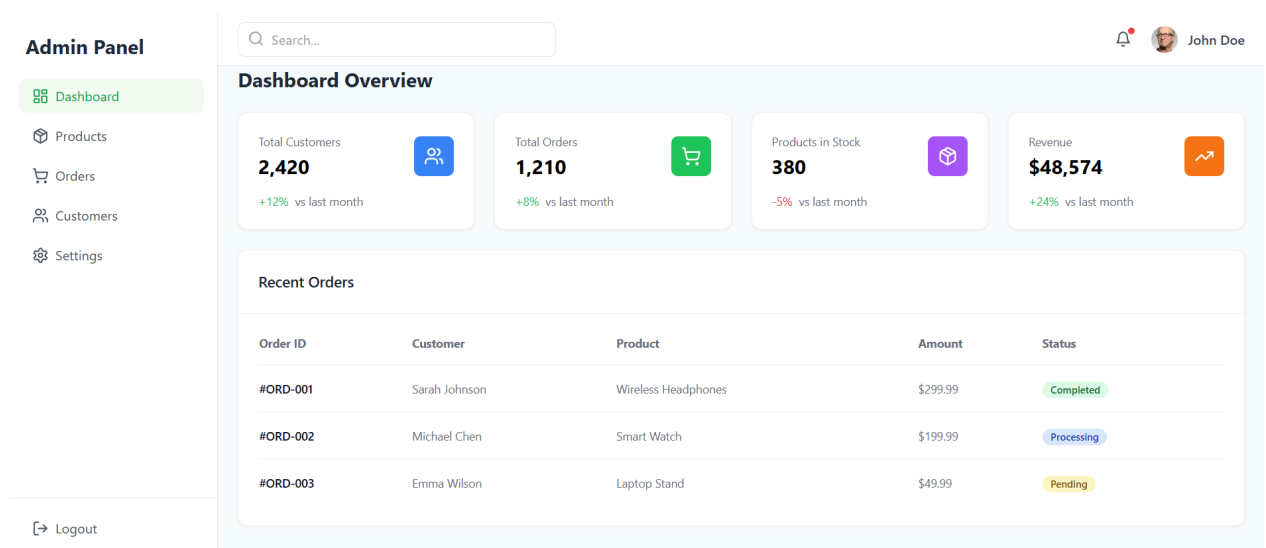


Imagen 8. Prototipo de ventana de administración.

### Flujo visual y eventos:

- Al ingresar, debería aparecer una pantalla inicial con accesos rápidos.
- Se esperaría que haya una navegación fluida entre productos, pedidos y clientes.