UNIVERSIDAD
NACIONAL
DE COLOMBIA
SEDE BOGOTÁ
FACULTAD DE INGENIERÍA

Diego Felipe Solorz…
Juan David Loaiza …
Daniel Aguilar Castro
Juan David Chacon …

UNIVERSIDAD NACIONAL DE
COLOMBIA SEDE BOGOTÁ
FACULTAD DE
INGENIERÍA
DEPARTAMENTO DE SISTEMAS E INDUSTRIAL

INGENIERÍA DE SOFTWARE 1
BOGOTÁ D.C
2024-2

PUNTO 1

CONVERSACIÓN ESCOGIDA: GP 2

En esta conversación la cliente ofrece un millón de pesos colombianos

PUNTO 2

Levantamiento de Requerimientos

Contexto del Negocio

El cliente tiene un pequeño negocio en línea enfocado en la venta de productos, principalmente artesanías, accesorios y artículos de decoración para el hogar. Además, tiene interés en explorar asesorías de decoración como servicio adicional. Actualmente, enfrenta problemas de organización en la gestión de inventarios, pedidos, ventas y promoción, utilizando herramientas básicas como WhatsApp y Excel.

El objetivo principal es desarrollar una solución digital para mejorar la organización, la eficiencia en la gestión de procesos y la experiencia del cliente.

Descripción General del Software

Se propone una aplicación web accesible desde dispositivos móviles y computadoras, con una interfaz sencilla y amigable. Esta herramienta permitirá gestionar inventarios, pedidos, ventas, y ofrecer un catálogo en línea para los clientes.

El cliente también expresa interés en funciones avanzadas como notificaciones automáticas, pasarelas de pago y posibilidades de expansión para incluir servicios de asesoría más adelante.

Lista de Funcionalidades

Gestión de Pedidos

- Registro y seguimiento de pedidos: Permitir registrar pedidos con información como cliente, productos solicitados, estado (pendiente, pagado, enviado, completado) y fecha límite de entrega.
- Actualización automática del inventario: Reducir automáticamente las unidades disponibles en el inventario al registrar un pedido.
- Historial de pedidos: Mostrar un historial de todos los pedidos realizados para seguimiento y análisis.

Gestión de Inventarios

- Control del stock: Visualizar los productos disponibles con cantidad en tiempo real.
- Alertas de bajo stock: Notificar al administrador cuando un producto esté por agotarse, enviando mensajes a través de WhatsApp y correos electrónicos semanales.
- Gestión de productos: Permitir añadir, editar o eliminar productos, incluyendo nombre, precio, descripción, fotos y cantidad en stock.

Ventas y Finanzas

- Gestión de pagos: Registrar información de pagos, como el monto, el medio de pago (transferencia, efectivo) y si el cliente ha pagado.
- Resumen financiero: Mostrar ingresos, gastos y ganancias en reportes claros y visuales.

Catálogo en Línea

- Visualización de productos: Crear un catálogo accesible para los clientes, organizado por categorías (artesanías, accesorios, decoración).
- Pedidos desde el catálogo: Permitir que los clientes seleccionen productos directamente desde el catálogo para generar un pedido.

Notificaciones

- Para el administrador: Alertas de bajo stock y resumen semanal del estado del inventario.
- Para los clientes: Confirmación automática de pedidos vía WhatsApp, incluyendo un link al estado del pedido.

Usabilidad y Personalización

- Interfaz intuitiva: Diseño sencillo que facilite el uso tanto para el administrador como para los clientes.
- Accesibilidad: Funcionalidad en línea, compatible con dispositivos móviles y computadoras.
- Crecimiento modular: Posibilidad de agregar funcionalidades, como un calendario para asesorías o herramientas de promoción, sin rehacer el sistema.

Promoción y Opiniones

- Sección de historias: Mostrar cómo otros clientes usan los productos con fotos y comentarios.
- Calculadora de envío: Ofrecer un estimado del costo de envío dependiendo de la ubicación del cliente.

PUNTO 3

Categoría	Funcionalidad	Prioridad (MoSCoW)	Tiempo Estimado (en días)	Complejidad
Gestión de Pedidos	Registro de pedidos con datos como cliente, productos, estado y fecha de entrega.	Must Have	2	Baja
	Actualización automática del inventario al registrar un pedido.	Must Have	2	Media
	Historial de pedidos para seguimiento y análisis.	Should Have	3	Media
Gestión de Inventarios	Visualización del stock de productos en tiempo real.	Must Have	2	Baja
	Alertas de bajo stock (WhatsApp y correo semanal).	Should Have	3	Media
	Gestión de productos (añadir, editar, eliminar).	Must Have	3	Baja
Ventas y Finanzas	Registro de pagos con monto, método y estado.	Must Have	2	Media
	Resumen financiero (ingresos, gastos, ganancias).	Should Have	3	Alta
Catálogo en Línea	Visualización de productos por categorías.	Must Have	3	Media

	Pedidos directos desde el catálogo.	Should Have	4	Alta
Notificaciones	Confirmación automática de pedidos a clientes vía WhatsApp.	Could Have	3	Alta
	Resumen semanal del inventario vía correo.	Could Have	2	Media
Usabilidad y Personalización	Diseño responsivo (móvil y computadora).	Must Have	5	Alta
	Interfaz intuitiva con tablas o listas.	Must Have	5	Media
Promoción y Opiniones	Calculadora de envío.	Could Have	3	Alta

Recursos Propuestos

- Presupuesto disponible: $1,000,000 COP.
- Desarrolladores involucrados: 1 (Full-stack developer con experiencia en proyectos similares).
- Tecnologías seleccionadas:
  - Frontend: HTML, CSS (con Bootstrap), y JavaScript (con Vue.js para simplicidad).
  - Backend: Node.js con Express.js.
  - Base de datos: MongoDB para almacenamiento en la nube.
  - Hosting: Plataforma gratuita o de bajo costo (Heroku o Render).
  - Integraciones: Twilio (notificaciones WhatsApp) y Nodemailer (correos).

Distribución de Trabajo

- **Semana 1 (Días 1 - 7)**
  - Configuración del entorno de desarrollo y base de datos.
  - Desarrollo del módulo de gestión de pedidos:
    - Registro de pedidos.
    - Actualización automática del inventario.
    - Gestión básica de productos.

- **Semana 2 (Días 8 - 14)**
  - Diseño responsivo de la interfaz para móvil y computadora.
  - Desarrollo del catálogo en línea con visualización de productos por categorías.
  - Implementación del módulo de pagos con registro de métodos y estados.

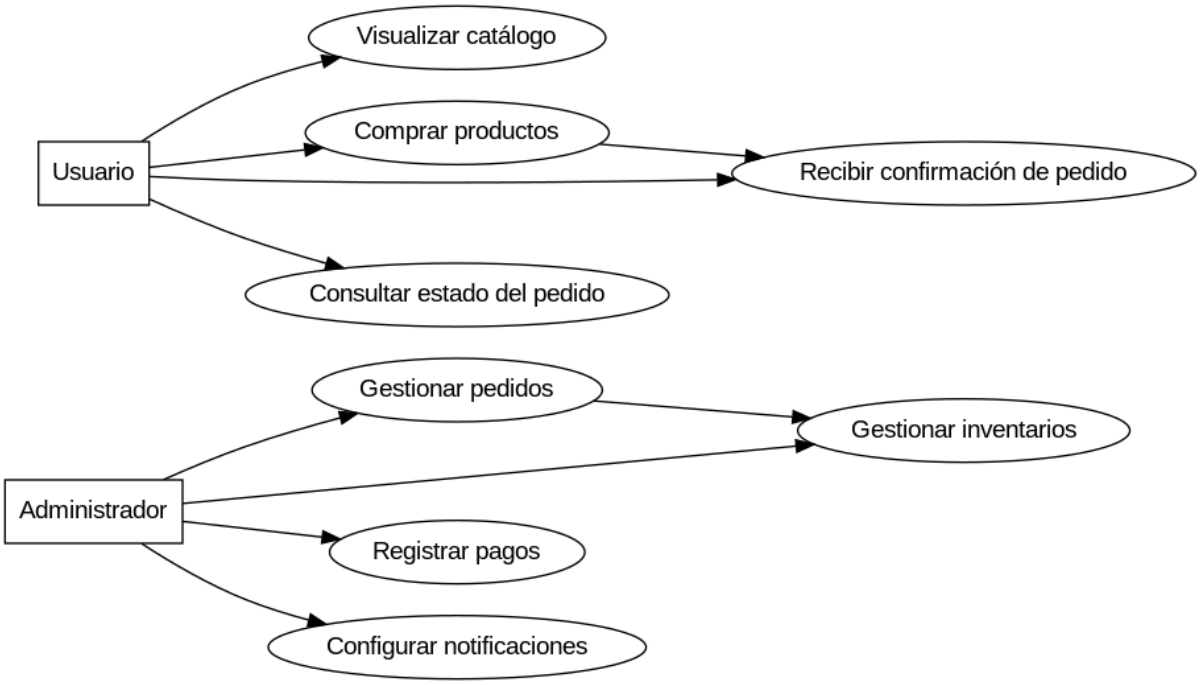
- **Semana 3 (Días 15 - 21)**
  - Creación del historial de pedidos.
  - Desarrollo de notificaciones automáticas:
    - Confirmación de pedidos vía WhatsApp.
    - Alertas de bajo stock.
  - Implementación de interfaz intuitiva con tablas para pedidos y stock.

- **Semana 4 (Días 22 - 30): Pruebas y despliegue**
  - Integración de todos los módulos.
  - Pruebas de usuario para asegurar que el sistema sea intuitivo y funcional.
  - Despliegue en una plataforma gratuita o de bajo costo.
  - Documentación básica para el cliente.

Impacto en la Triada de Desarrollo

1. Tiempo: 30 días ajustados al presupuesto.
2. Costo: $1,000,000 COP, empleando un desarrollador full-stack.
3. Alcance:
   - Todas las funcionalidades 'Must Have' serán implementadas.
   - Algunas 'Should Have' y 'Could Have' quedarán como opciones para una segunda etapa dependiendo de los recursos disponibles.

PUNTO 4


Visualizar catálogo
Comprar productos
Usuario
Recibir confirmación de pedido
Consultar estado del pedido
Gestionar pedidos
Gestionar inventarios
Administrador
Registrar pagos
Configurar notificaciones

PUNTO 5 HU1.
Anexo de Documentos Relacionados:

- *http status codes - Is it correct to return 404 when a REST resource is not found? - Stack Overflow*

Descripción conceptual

Módulo	*Gestión de Pedidos*
Descripción de la(s) funcionalidad(es) requerida(s):	*Historial de pedidos para seguimiento y análisis.*

Descripción técnica

En este apartado generamos dos endpoints de tipo GET uno con los datos del pedido y otro con una lista con los id de pedido.

Backend

URL	**Método**	**Código html**
https://user.domain.heroku/pedidos/:idUsuario	GET	200 404
Descripción Al poner la ID del pedido retornara 200 si el query es exitoso y 404 si no se encuentra ningún pedido		
Datos de entrada $idUsuario: 001	**Datos de salida** **200:** **{** “status”: “Succes”, “statusCode”: 200, “data”: { “pedidos”: [001,002,003] **}** **404:** **{** “status”: “Failed”, “statusCode”: 404, “data”: { } **}**	

<table>
<tr><th>URL
https://user.domain.heroku/pedidos/:idPedido</th><th>Método

GET</th><th>Código html

200
404</th></tr>
<tr><td colspan="3">Descripción
Al poner la ID del pedido retornara 200 si el query es exitoso y 404 si no se encuentra el código del pedido</td></tr>
<tr><td>Datos de entrada
$idPedido: 001</td><td colspan="2">Datos de salida
200:
{
“status”: “Succes”,
“statusCode”: 200,
“data”: {
“idPedido’: “001”,
“estadoPedido”: “Entregado”
“fecha”: 01/01/2025,
“datosCliente”:{
“nombre”: “Jhon Doe”,
“id”: 0001,
“correo”: “test@test.com”,
“ubicacion”: “091N-14W”
},
“idProductos”: [001,002,003]
}
}

404:
{
“status”: “Failed”,
“statusCode”: 404,
“data”: { }
}</td></tr>
</table>

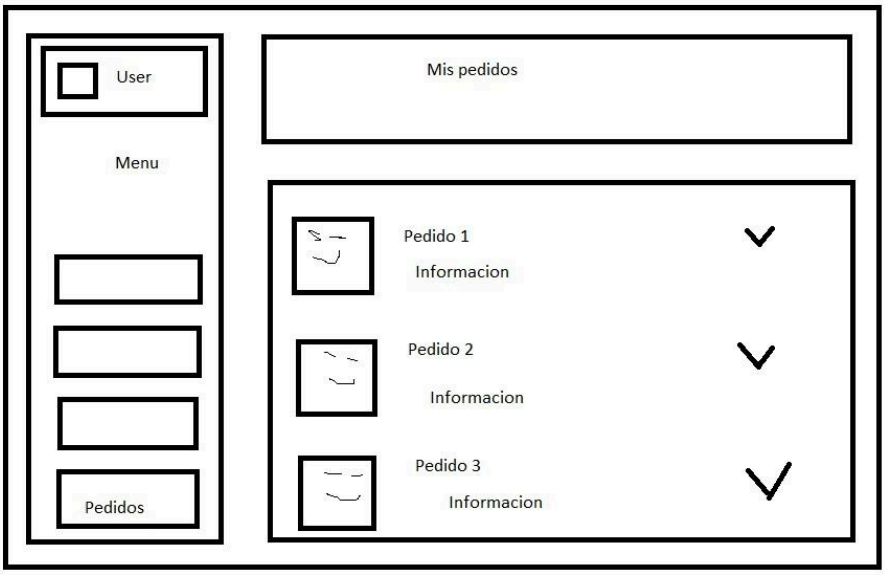
Frontend

Este apartado debe consistir en una pantalla la cual nos va a listar todos los pedidos que haya hecho el cliente y al darle click se expande con la información.

Interacción esperada:

> *El usuario accede a través del menú principal a su historial de pedidos, donde verá un listado con todos los pedidos que ha hecho. De no tener pedidos estará vacío.*

Mockups/Prototipos:



Pantalla historial pedidos, elaboración propia.

Flujo visual y eventos:

Le puede dar click a las flechitas hacia abajo, que le desplegara la información de los pedidos que ha hecho, también puede hacer click en el nombre de los pedidos y será redirigido a ellos.

PUNTO 5 HU2.

Anexo de Documentos Relacionados:

- HTTP status code for update and delete? - Stack Overflow

Descripción conceptual

Módulo	*Gestión de Pedidos*
Descripción de la(s) funcionalidad(es) requerida(s):	*Registro de pedidos con datos como cliente, productos, estado y fecha de entrega..*

Descripción técnica

En este apartado solamente existe el backend para el cliente, ya que es un paso entremedias tras la compra por parte del cliente.

Para la actualización el usuario va a tener un pequeño menú de actualización.

Backend

Vamos a tener un endpoint de tipo POST para crear los datos del pedido y un método patch para actualizar los datos de estado y fecha..

URL	**Método**	**Código html**
https://user.domain.heroku/pedidos/:pedidoId	PATCH	201 404
Descripción Daremos de respuesta el codigo 201		

Datos de entrada
$pedidoId: 001
{
“fecha”:”01/01/2025”,
“estado”:”Entregado”
}
	Datos de salida
	201:
	{
	“status”: “Updated”,
	“statusCode”: 201,
	“pedidoId”: 001
	}
	
	404:
	{
	“status”: “Not found”,
	“statusCode”: 404,
	“data”: { }
	}	

<table>
<tr><td>URL
https://user.domain.heroku/pedidos/create</td><td>Método
POST</td><td>Código html
201
400</td></tr>
<tr><td colspan="3">Descripción
Daremos de respuesta el codigo 201</td></tr>
<tr><td>Datos de entrada
{
“usuarioId”: 001,
“productosId”:[001,002,003],
“estado”: “En proceso”
“fechaPedido”:”31/12/2024”
}</td><td colspan="2">Datos de salida
201:
{
“status”: “Succes”,
“statusCode”: 201,
“pedidoId”: 001
}

400:
{
“status”: “Failed”,
“statusCode”: 400,
“data”: { }
}</td></tr>
</table>

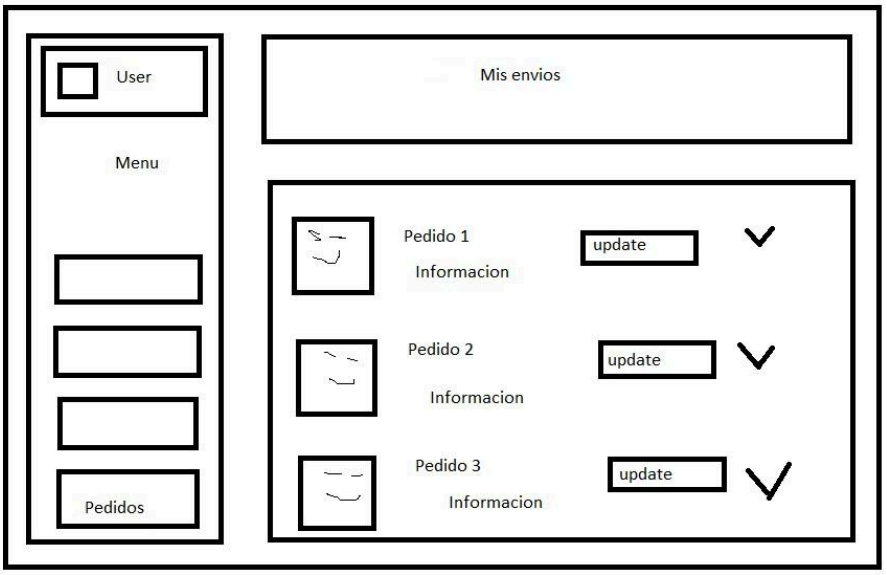
Frontend

El usuario que hace el envío va a contar con un pequeño espacio para actualizar los envíos por código.

Interacción esperada:

> *Al darle click al boton actualizar envios en el menú principal, verá todos los envíos y puede seleccionar cual actualizar*

Mockups/Prototipos:



Flujo visual y eventos:

Al darle click en la flecha, también se cargará la información para el usuario que ya generamos en la HU1. La diferencia reside en la capacidad de actualizar el estado con el botón de update, que abre el menú donde puede elegir la fecha y el estado que se guardaran automáticamente al cambiarlos.

PUNTO 5 HU3.

Descripción conceptual

Módulo	*Gestión de Pedidos*
Descripción de la(s) funcionalidad(es) requerida(s):	*Actualización automática del inventario al registrar un pedido.*

Descripción técnica

En este apartado solamente contaremos con backend, ya que es una funcionalidad intermedia que se encarga de actualizar el inventario de forma automática al momento de la compra/inserción del pedido, el usuario no interactúa directamente con la funcionalidad

Backend

Aquí se debe explicar de manera técnica lo esperado,

<table>
<tr><td>URL
https://user.domain.heroku/inventory</td><td>Método
PUT</td><td>Código html
201
404</td></tr>
<tr><td colspan="3">Descripción
Retorna 201 si se actualiza el inventario, 404 si no encuentra el objeto en el inventario a actualizar.</td></tr>
<tr><td>Datos de entrada
{
“idProducto”: 001
“cantidadComprada’: 1
}</td><td colspan="2">Datos de salida
201:
{
“status”: “Succes”,
“statusCode”: 201,
“data’: {
“idProducto”: 001
“cantidad”: 99
}
}

400:
{
“status”: “Not found”,
“statusCode”: 404,
“data”: { }
}</td></tr>
</table>

Notas Adicionales:

No se adiciona información sobre el frontend ya que como se explico en la sección de backend, esta interacción solamente ocurre como un proceso entremedias de la creación de

pedido/compra y solo sera reflejado cuando el usuario acceda a la base de datos o la información de esta, es decir la interacción del usuario con esta funcionalidad es indirecta.

PUNTO 5 HU4.

Anexo de Documentos Relacionados:

- *En este apartado se agregan toda la documentación que sea relevante para entender o sustentar la historia de usuario*

Descripción conceptual

Módulo	*Gestión de inventarios*
Descripción de la(s) funcionalidad(es) requerida(s):	*Gestión de productos (añadir, editar, eliminar).*

Descripción técnica

En este apartado existen dos partes, la del backend y la del frontend

Backend

Se esperan 2 endpoint, el primero será un POST para la creación de productos, mientras que el otro sera dinamico con respecto al product ID y soportara los métodos PUT y DELETE,

URL	**Método**	**Código html**
https://user.domain.heroku/inventory/create	POST	201 400
Descripción Retorna 201 si se crea correctamente el producto y 400 si ocurre algún error en el proceso		

Datos de entrada
{
“productName”: “Awesome product”,
“description’: “Awesome description”,
“quantity”: 100,
“price?”: 1000
}
	Datos de salida
	201:{
	“status”: “Succesful”,
	“statusCode”: 201,
	“data”:
	{
	“idProduct”: 001
	}
	}
	400: {
	“status”: “Error”,
	“statusCode”:400,
	“data”: {}
	}	

URL	**Método**	**Código html**
https://user.domain.heroku/inventory/:idproducto	PUT	201 404
Descripción Retorna 201 si el producto fue actualizado correctamente, si no se encuentra el producto retorna 404		

Datos de entrada
$idProducto
{
“productName?”: “Awesome product”,
“description?”: “Awesome description”,
“quantity?”: 100,
“price?”: 1000
}
	Datos de salida
	201:{
	“status”: “Succesful”,
	“statusCode”: 201,
	“data”:
	{
	“idProduct”: 001,
	“productName?”: “Awesome product”,
	“description?”: “Awesome description”,
	“quantity?”: 100,
	“price?”: 1000
	}
	}
	404: {
	“status”: “Not found”,
	“statusCode”:404,
	“data”: {}
	}	

URL	**Método**	**Código html**
https://user.domain.heroku/inventory/:idproducto	DELETE	201 404
Descripción Retorna 201 si el producto fue eliminado correctamente, si no se encuentra el producto retorna 404		

Datos de entrada
$idProducto
	Datos de salida
	Datos de salida
	201:{
	“status”: “Succesful”,
	“statusCode”: 201,
	“data”:
	{ }
	}
	404: {
	“status”: “Not found”,
	“statusCode”:404,
	“data”: {}
	}	

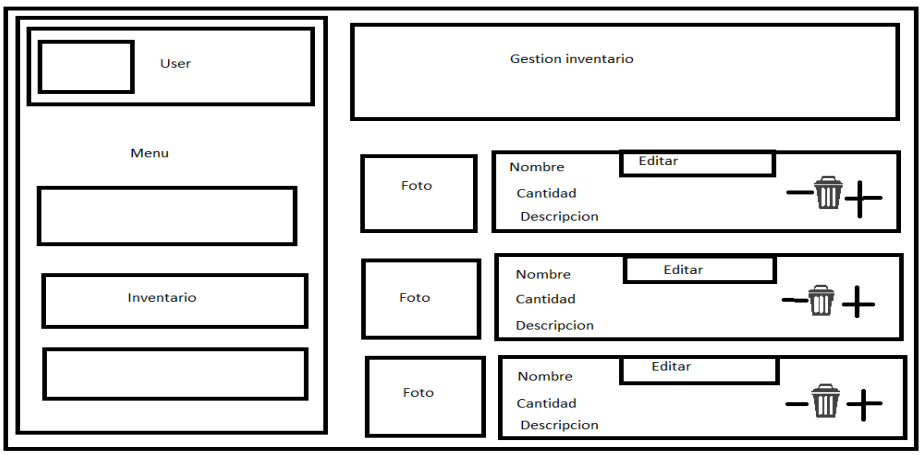
Frontend

A continuación vamos a ver un flujo sencillo y amigable donde el usuario va a poder manipular a gusto todos los detalles de su inventario y sus productos.

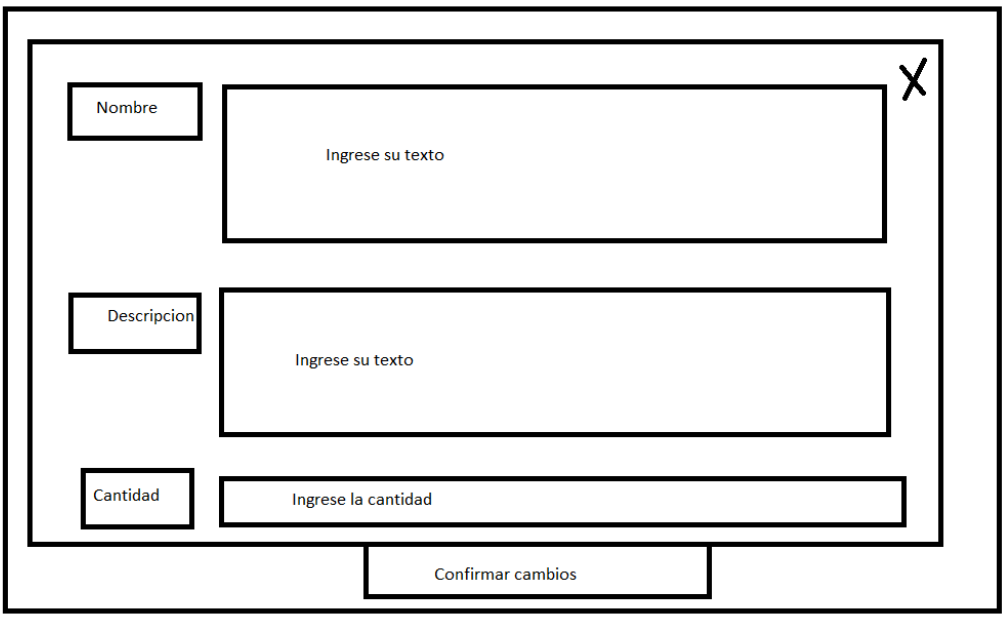
Interacción esperada:

> *Se espera que el usuario al darle click en el menú de gestión de inventario le cargue la ventana de gestión de inventario, donde se va a encontrar con varias opciones. El botón de editar en cada uno de los productos del inventario le va a llevar a la ventana de edición producto, donde va a poder editar el nombre, cantidad y descripción del producto. En caso de darle click al símbolo de más o menos va a aumentar o decrementar la cantidad de productos de forma rápida y sencilla. Finalmente si da click en el bote de basura le va a cargar la ventana emergente de eliminar producto, allí es donde el cliente decide si eliminarlo o si le clicko por accidente omitir.*

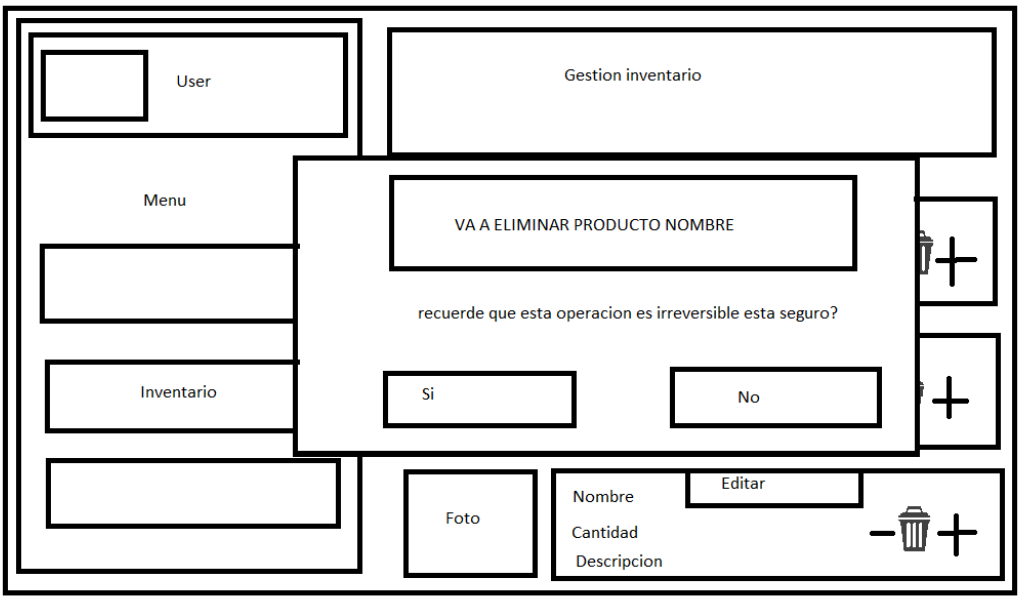
Mockups/Prototipos:



Pantalla gestion inventario



Pantalla edicion producto



Pantalla gestion inventario, ventana emergente de eliminar producto

Flujo visual y eventos:

Se espera que carguen dos pantallas en dos eventos precisos, a la hora de eliminar el pop up de confirmación y para editar una pestaña más grande donde pueda editar el usuario los datos del producto. De no tener productos cargará en blanco esta pantalla, es decir sin productos.

Notas Adicionales:

(Espacio para comentarios, dependencias o aclaraciones técnicas relevantes)

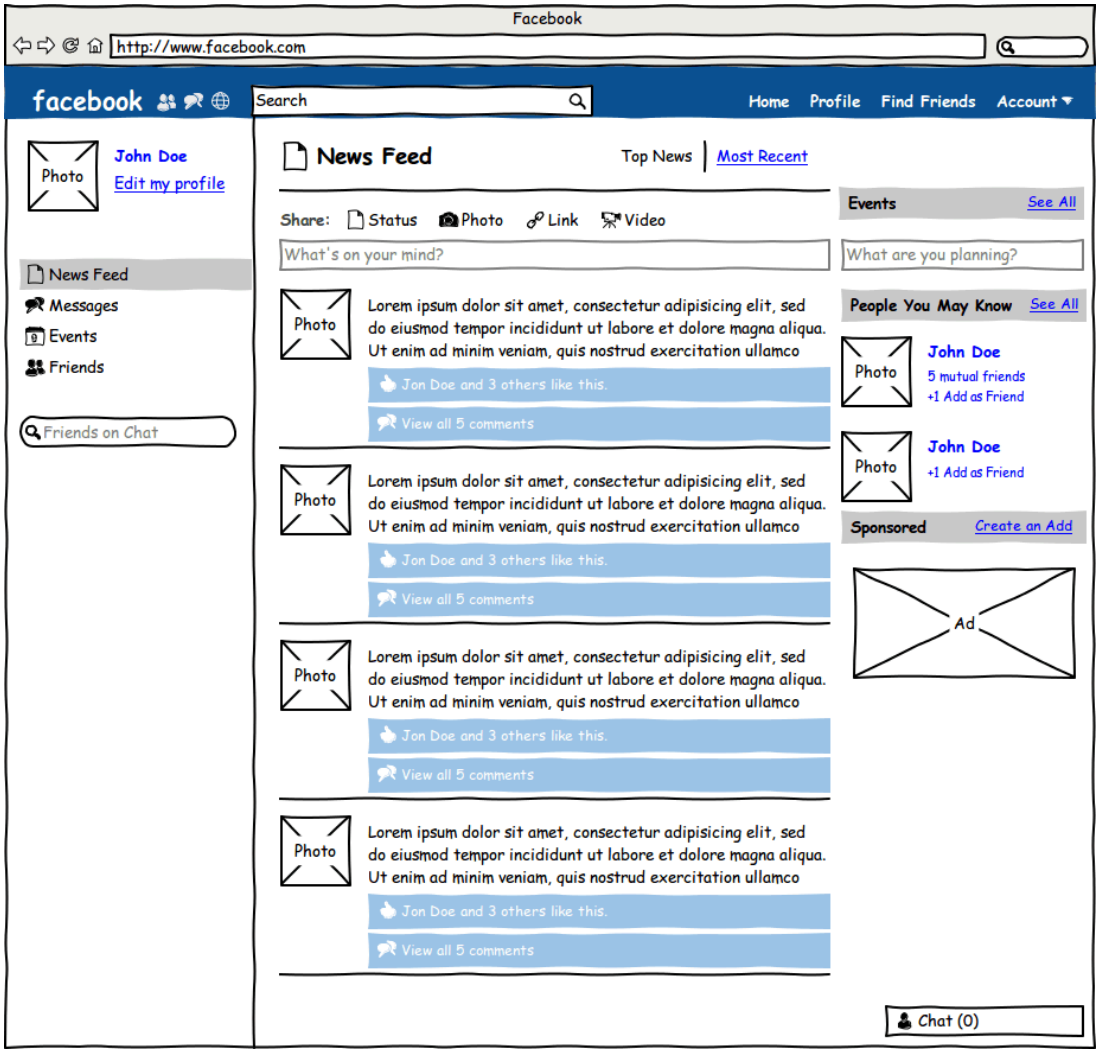


Imagen tomada de: https://wireframesketcher.com/sample-mockups.html