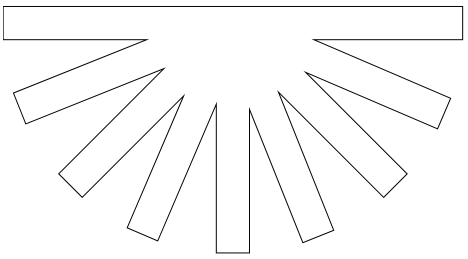
DISEÑO DE SISTEMAS DE INFORMACIÓN

TRABAJO PRÁCTICO 3



PROFESORES:

Ing. Juan Pablo Ferreyra
Ing. Pablo Pioli

REALIZADO POR:

Talía Zurbriggen

Índice

Propuesta de solución

El proyecto se centrará en el desarrollo de un sitio web que permita a los clientes cotizar y realizar pedidos de los materiales de construcción. Se integrará la API de Mercado Pago para gestionar los pagos online, y la plataforma será utilizada tanto por la empresa (para gestionar productos, pedidos y envíos) como por los clientes (para cotizar, hacer pedidos y pagar).

Implementación

1) Frontend

   La aplicación será desarrollada en React. Esto permite una navegación fluida y una mejor experiencia de usuario.

   Vistas principales:

   Catálogo de productos: Los clientes pueden consultar productos disponibles, ver detalles, precios, y características.

   Cotización: Los clientes pueden calcular la cantidad de materiales necesarios y generar cotizaciones en función de sus proyectos.

   Pagos: Integración con la API de Mercado Pago para gestionar los pagos en línea.

2) Backend

   El backend gestionará la lógica de negocio y la comunicación con las APIs externas (como Mercado Pago) y la base de datos. Se puede desarrollar usando Node.js con un framework como Express, permitiendo la escalabilidad y una estructura modular.

   API REST:

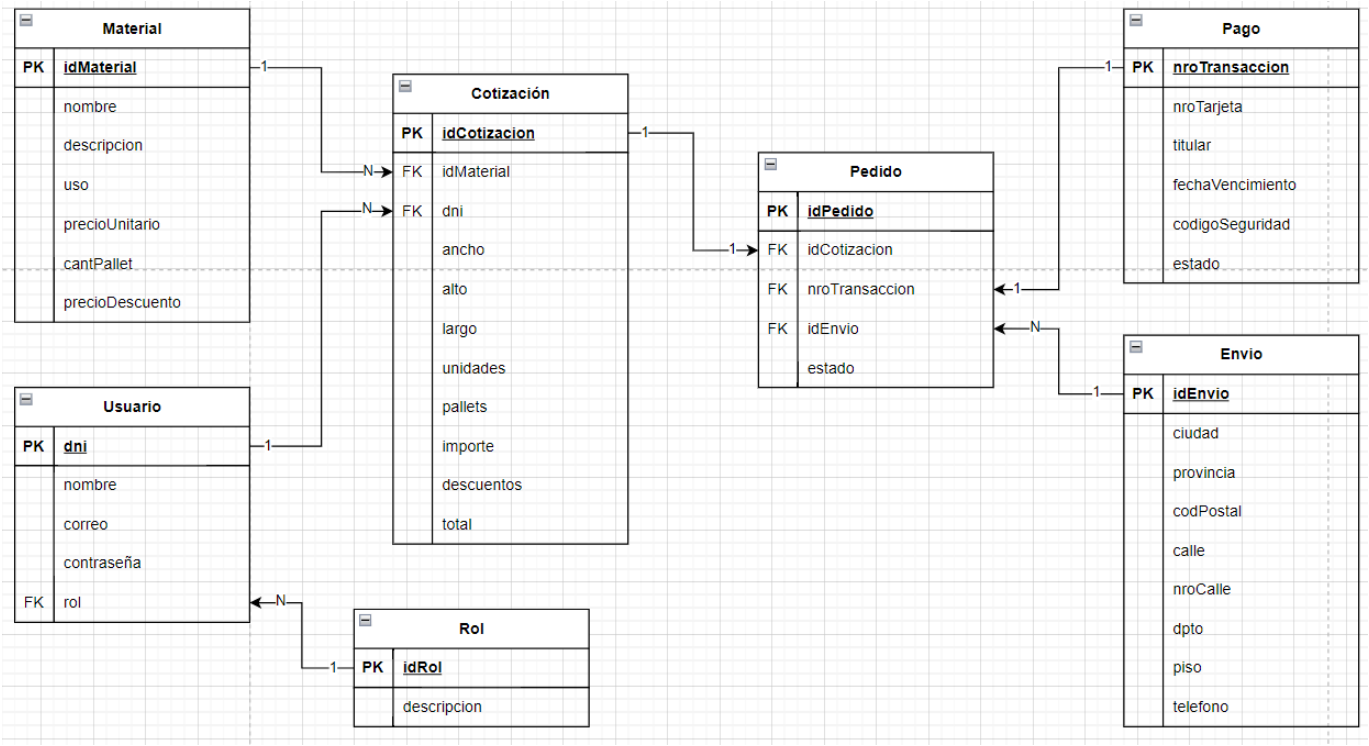
   Endpoints para gestionar productos, cotizaciones, pedidos y pagos.

   Integración con Mercado Pago para el proceso de pago y validación del estado de transacciones.

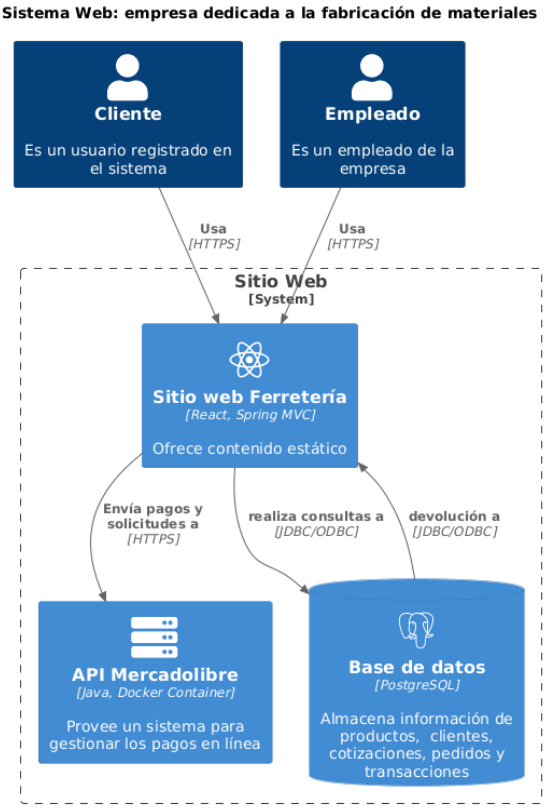
   Autenticación y gestión de usuarios mediante tokens (JWT).

3) Base de Datos

   Una base de datos SQL como PostgreSQL puede almacenar los datos de los productos, clientes, cotizaciones, pedidos y transacciones. También es posible utilizar NoSQL (como MongoDB) si se requiere flexibilidad en los modelos de datos.



Diseño de arquitectura

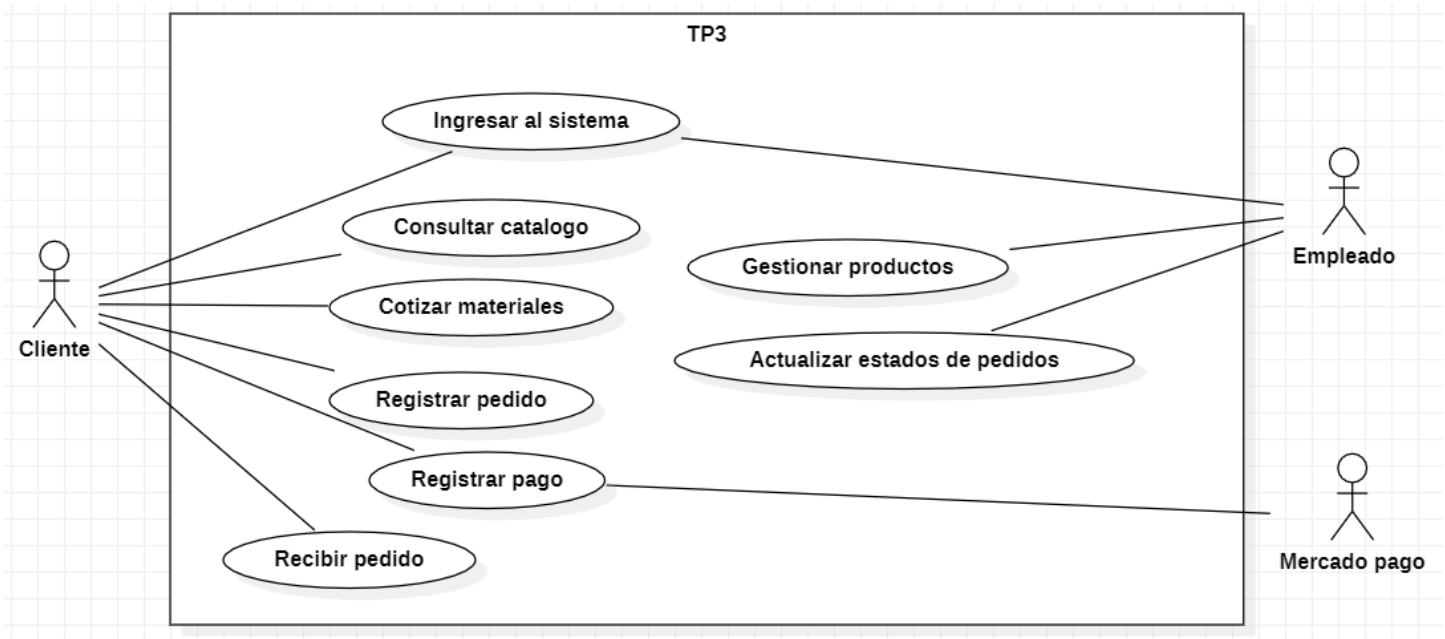


Vista interna del proceso de negocios

PROCESOS CLAVE	
Gestión de ventas	Los clientes podrán consultar los productos, realizar una cotización en base a las dimensiones requeridas, realizar un pedido, gestionar su pago y coordinar el envío.
Administración	La empresa gestiona sus productos disponibles, realiza modificaciones en ellos y también elimina productos que ya no se encontrarán en el catálogo
Envío	La empresa tendrá a su disposición los pedidos que deben ser enviados junto con los datos del mismo. Se harán reportes de entregas cuando el pedido fue entregado. Coordinación de viajes.

Diagrama BPMN de los procesos gestión de ventas y envío:
https://modeler.camunda.io/share/66450ba4-2468-4764-910c-4910762837e9

Diagrama de casos de uso



Requerimientos

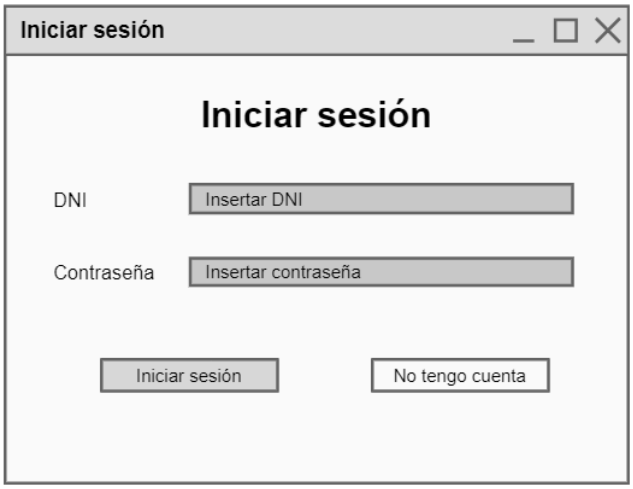
Requerimientos funcionales:

- El sistema debe permitir al cliente identificarse mediante un registro e inicio de sesión.
- El sistema debe mostrar un catálogo de productos disponibles con sus detalles, como precio, dimensiones y descripción.
- El cliente debe poder ingresar los datos de su proyecto y el sistema debe calcular automáticamente la cantidad de materiales necesarios y el importe total.
- El sistema debe ofrecer descuentos automáticos sobre los productos según la cantidad solicitada.
- El sistema debe poder convertir una cotización en un pedido, teniendo en cuenta detalles como la dirección de envío.
- El sistema debe permitir a los clientes ingresar la dirección de envío y calcular los costos o tiempos de entrega según la ubicación.
- El sistema debe permitir a los clientes elegir una forma de pago y completar la transacción para aprobar el pedido.
- El sistema debe permitir a los usuarios administradores crear, editar y eliminar materiales.

Requerimientos no funcionales (ISO/IEC 25000):

- El sistema debe procesar las consultas de cotización y pedidos en menos de 2 segundos, garantizando una respuesta rápida incluso con altos volúmenes de usuarios simultáneos.
- El sistema debe ser capaz de escalar automáticamente para soportar un mayor número de usuarios concurrentes, especialmente durante eventos de alta demanda.
- El sitio web debe tener una disponibilidad mínima del 99.9%, asegurando que esté accesible para los clientes en todo momento.
- El sistema debe proteger los datos personales y financieros de los clientes mediante cifrado en todas las transacciones y conexiones seguras (HTTPS).
- Se debe realizar una prueba de usabilidad con clientes para mejorar la experiencia.
- El sistema debe estar diseñado para permitir actualizaciones y mejoras sin interrumpir el servicio, con módulos bien estructurados para facilitar el mantenimiento.
- El sistema debe poder recuperarse automáticamente de errores comunes sin pérdida de datos ni interrupción prolongada del servicio.

Prototipos de interfaz de usuarios



Datos que se envían:

```
{
 usuario: [
 {
   dni: int;
   contraseña: string;
 }]
}
```

El frontend envía una solicitud con los datos del formulario (dni y contraseña) al backend.

El backend toma el dni enviado y busca en la base de datos si existe un usuario registrado con esos datos. Si encuentra el usuario, el backend toma la contraseña almacenada en la base de datos y compara si coincide con la contraseña que el usuario ingresó.

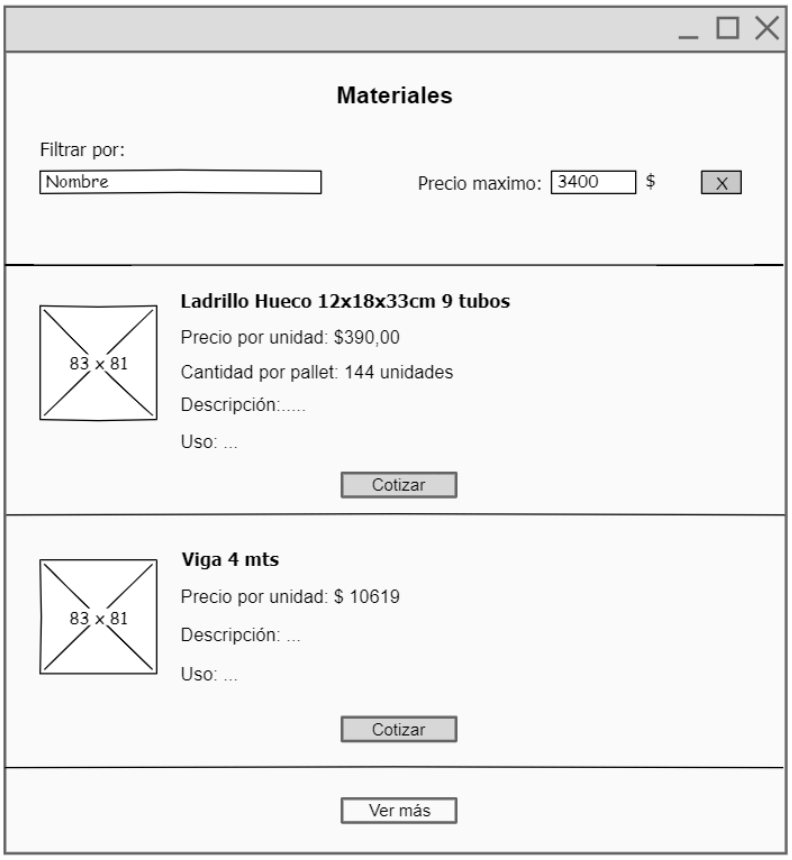
Si la contraseña es correcta, el backend genera un token de autenticación utilizando JWT (JSON Web Token). Este token contiene información del usuario (como dni, nombre, rol), en un formato seguro y encriptado. Una vez autenticado el usuario, el backend puede verificar el rol del usuario (cliente o administrador) y proporcionar acceso a las funcionalidades específicas que correspondan a ese rol.

Datos que se envían:

```
{
  usuario: [
  {
    dni:int;
    nombre: string;
    correo: string;
    contraseña: string;
  }
 ]
}
```

Cuando se presiona el botón "Crear cuenta", el formulario envía los datos ingresados en el formato JSON al backend. Éste último se va encargar de procesar la solicitud verificando que no exista algún usuario con el mismo dni. Si no existe se crea un registro con la información recibida, en el caso contrario, se va a devolver un mensaje de error indicando que ese usuario ya está registrado.

Usuario cliente

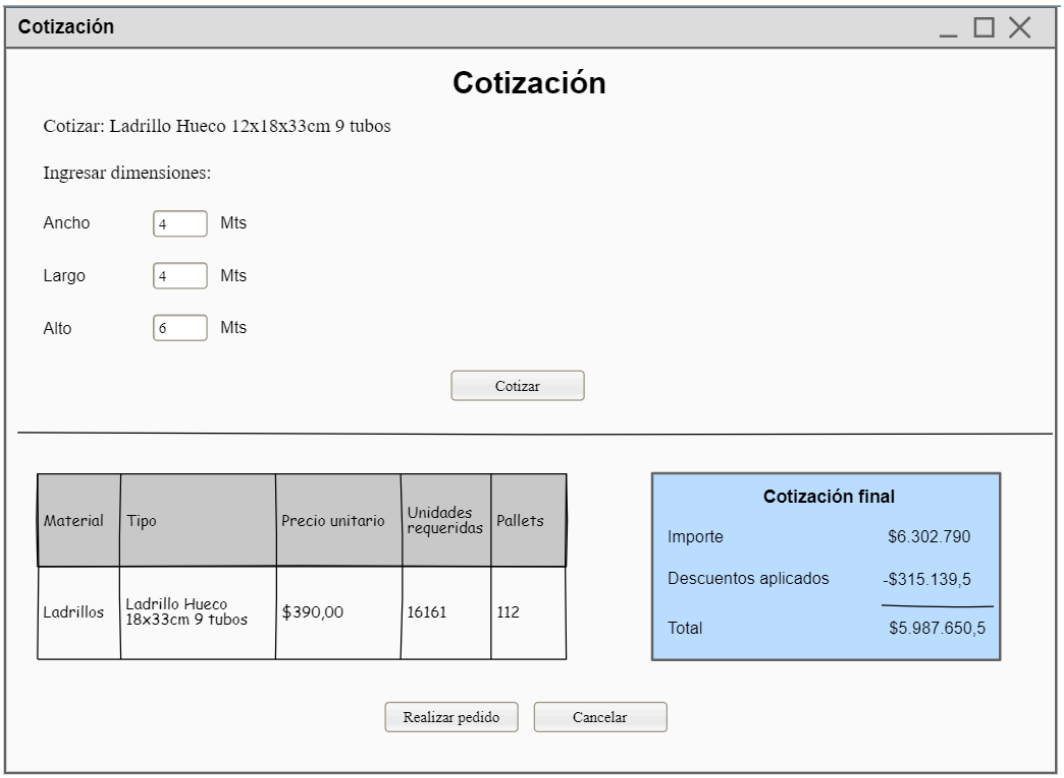


Datos que se reciben:

```
{
 materiales: [
 {
  id.material:int;
  nombre: string;
  descripción: string;
  foto: string;
  precio: double;
  cantPorPallet:int
 }
 ]
}
```

El backend hace una consulta a la base de datos para recuperar los datos de los materiales.

Después de obtener los datos de la base de datos, el backend responde al frontend con los datos solicitados en formato JSON. Estos datos incluyen toda la información de los materiales.



Datos que se envían	Datos que se reciben

{
cotizar: [{
id.material:int;
largo: double;
ancho: double;
alto: double;
}]
}
	{
	cotización: [{
	cant.Ladrillos:int;
	precioTota:doublel;
	precioConDescuento:double;
	cantPallets:int;
	}]
	}

El backend recibe un JSON con la información del material que el usuario desea cotizar, incluyendo:

- id del material (para saber qué material es).
- Largo, Ancho y Alto (para calcular cuánto material se necesitará).

Con el id del material, el backend puede hacer una consulta a la base de datos para obtener el precio por unidad y cualquier otra información adicional necesaria.

Usando las dimensiones recibidas (largo, ancho y alto), el backend calcularía cuántas unidades de material se necesitan. Este cálculo se basará en las dimensiones del producto y cómo encaja en el área especificada.

Ejemplo de cálculo de unidades e importe utilizando vigas y ladrillos:

Datos de entrada:

- ancho = 40 (en metros)
- largo = 40 (en metros)
- altura = 6 (en metros)

Especificaciones del ladrillo:

- tamaño_ladrillo_m2 = 0.18 * 0.33
- precio_por_ladrillo = 390
- ladrillos_por_pallet = 144

  Cálculo de la superficie total a cubrir con ladrillos:

- superficie_paredes = 2 * altura * (ancho + largo)
- cantidad_ladrillos = superficie_paredes / tamaño_ladrillo_m2
- cantidad_pallets = cantidad_ladrillos / ladrillos_por_pallet
- importe_total_ladrillos = cantidad_ladrillos * precio_por_ladrillo

Especificaciones de las vigas:

- separacion_vigas = 0.40 (separación entre vigas en metros)
- precio_por_viga = 10619

  Cálculo de la cantidad de vigas:

- longitud_pared = largo (Usamos el largo para colocar las vigas a lo largo del techo)
- cantidad_vigas_por_pared = longitud_pared / separacion_vigas
- cantidad_vigas_totales = cantidad_vigas_por_pared * 2
- importe_total_vigas = cantidad_vigas_totales * precio_por_viga

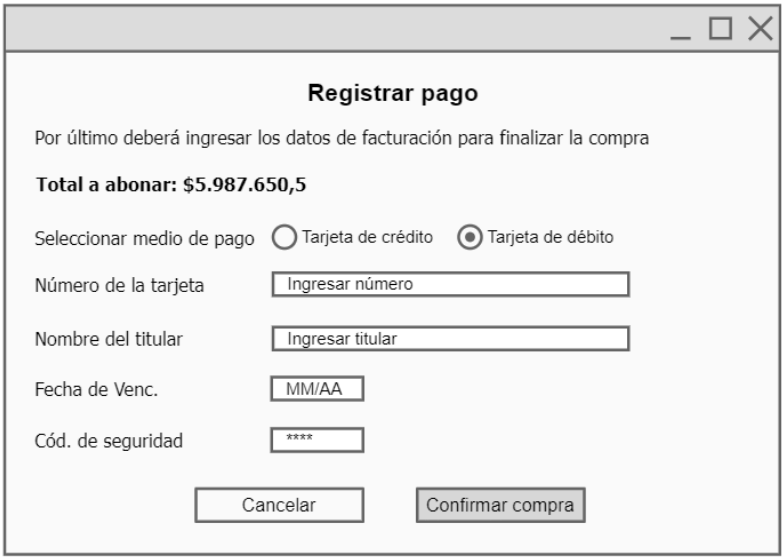
Si el número de unidades excede un límite definido para un descuento, el backend puede aplicar el descuento automáticamente.

Ejemplo:

```
if cantidad_ladrillos >= 10000:

   importe_total_ladrillos *= 0.95  (Descuento del 5%)
```

Una vez que el backend ha realizado los cálculos, devuelve un JSON con la cotización completa.

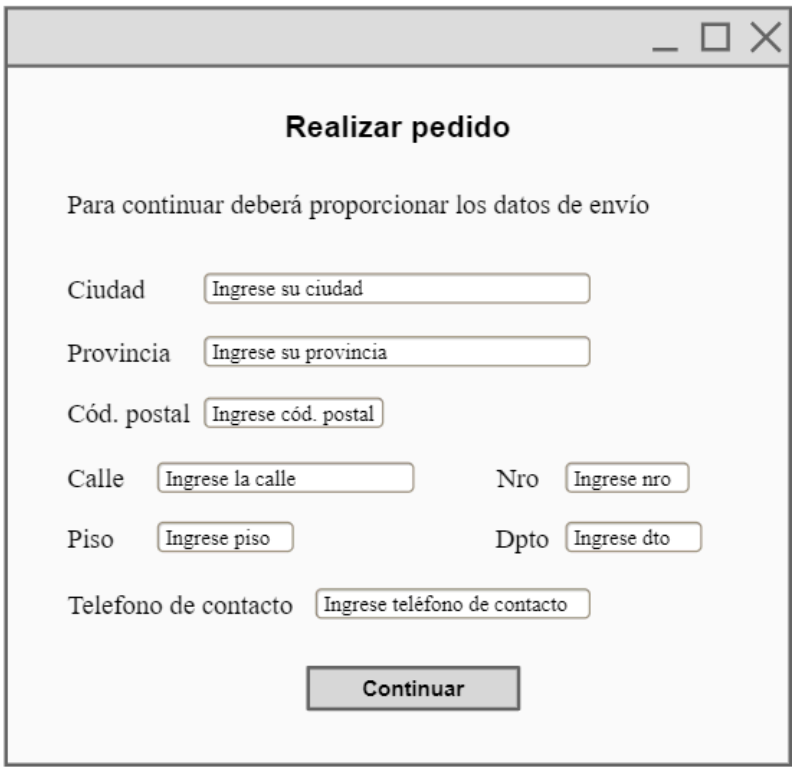


Datos que se envían	Datos que se reciben

{
pago: [{
id.transaccion: int,
total: double,
metodoPago: string,
tarjeta: {
numero: string,
nombreTitular: string,
fechaVencimiento: string,
codigoSeguridad: string
}]
}
	{
	pago: [{
	estado: string
	}]
	}

El frontend solicita los datos del pago al usuario y envía la información al backend.

Backend procesa el pago utilizando la API de Mercado Pago y devuelve el resultado al frontend para informar al usuario del estado del pago.



Datos que se envían:

```
{
 envio: [{
   ciudad: string;
   provincia: string;
   cod.postal: int;
   calle: string;
   nro.calle: int;
   piso: int;
   dpto: int;
   telefono: string;
 }]
}
```

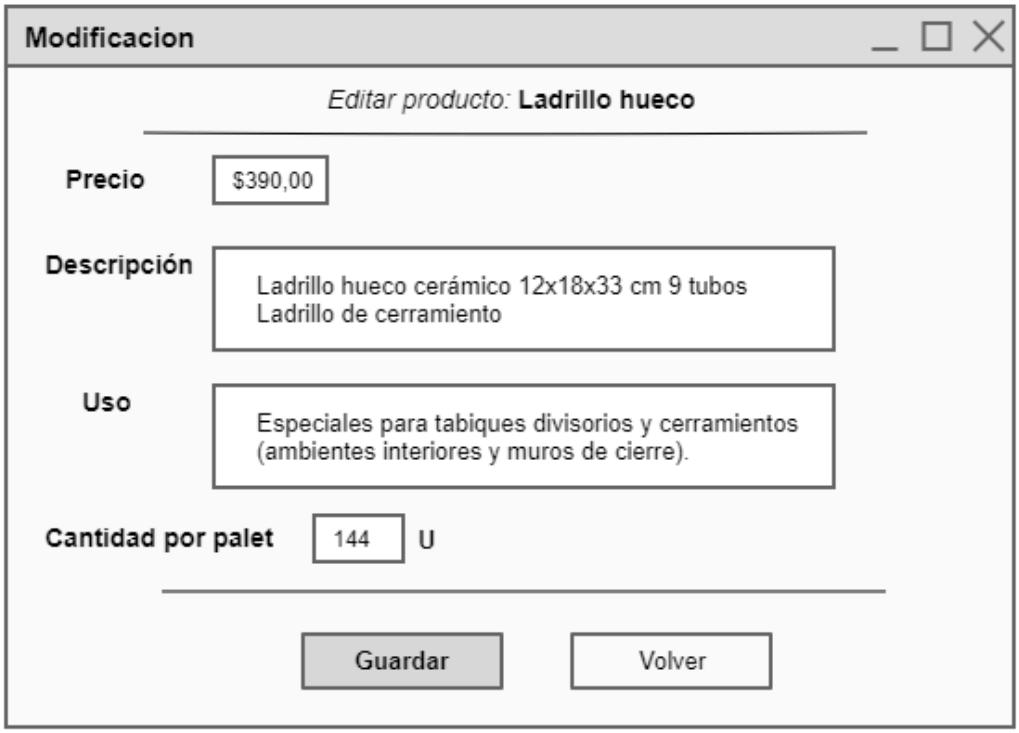
El backend recibiría los datos del formulario y los validará. Una vez se realice la validación, se procede a almacenar esta información en la base de datos, asociándola a un pedido.

Usuario administrador:

Datos que se reciben:

```
{
  materiales: [
   {
     Id.material:int;
     nombre: string;
   }
  ]
}
```

El backend va a recuperar la lista de materiales con sus nombres, teniendo la opción de añadir, editar(próxima interfaz) o eliminar un material. Al presionar "Guardar", el backend recibe un array con los materiales, que incluye el ID y el nombre de cada uno, y actualiza la base de datos en consecuencia. El botón "Cancelar" descarta cualquier cambio realizado.



Datos que se envían	Datos que se reciben

{
materiales: [
{
id.material:int;
nombre: string;
descripción: string;
foto: string;
precio: double;
cantPorPallet:int
}
]
}
	{
	materiales: [
	{
	id.material:int;
	nombre: string;
	descripción: string;
	foto: string;
	precio: double;
	cantPorPallet:int
	}
	]
	}

En primera instancia, al querer editar un material, el backend recupera y nos suministrará los datos previamente cargados del material. Una vez aplicados los cambios y al presionar el botón "Guardar" el backend actualiza el registro de dicho material.

Datos que se envían:

```
{
  agregar: [{
   nombre: string;
   precio: double;
   descripción: string;
   uso: string;
   cantidadPalet: int;
  }]
}
```

El frontend solicita los datos del nuevo material a través de un formulario donde el usuario administrador ingresa toda la información del material. Luego envía esos datos al backend.

El backend recibe los datos y los valida para asegurarse de que sean correctos). Luego inserta los datos del material en la base de datos, almacenándolos de manera permanente.

El backend devuelve una respuesta al frontend, indicando si el material fue agregado exitosamente o si hubo algún error (por ejemplo, datos incorrectos o problemas al conectarse con la base de datos).