TRABAJO PRÁCTICO 3

SEGUNDA ITERACIÓN

DISEÑO DE SISTEMAS DE INFORMACIÓN

PROFESORES:
Ing. Juan Pablo Ferreyra
Ing. Pablo Pioli

ALUMNO:
Tobías Gasparotto Vietto

Índice

Propuesta de solución

Expansión del Proyecto: App Móvil
Dado el éxito del sitio web, se desarrollará una aplicación móvil que replicará las funcionalidades del sistema web y añadirá nuevas características. Esta app móvil estará disponible para iOS y Android y ofrecerá las siguientes capacidades:

- Acceso al catálogo de productos, cotización, pedidos y pagos.
- Gestión de pedidos en curso y consulta de historial.
- Acumulación de puntos de fidelización.
- Integración con múltiples plataformas de pago.

Implementación

1) Frontend Móvil:
   La aplicación móvil se desarrollará en React Native o Flutter para asegurar una experiencia uniforme en dispositivos iOS y Android.
   El diseño de la app mantendrá la estética y funcionalidad del sitio web, optimizando la usabilidad en pantallas móviles.
   Vistas principales:
   - Catálogo de productos: Presentación de productos con detalles y opciones de cotización.
   - Cotización: Capacidad para revisar los pedidos y calcular los materiales según los planos o la estimación del usuario.
   - Historial de puntos: Visualización de los puntos acumulados y reglas de obtención.
   - Múltiples métodos de pago: Selección del método de pago en el flujo de checkout.
   - Pantallas para administración: Altas/bajas/modificaciones de productos, modificación de las reglas de puntuación y descuentos..

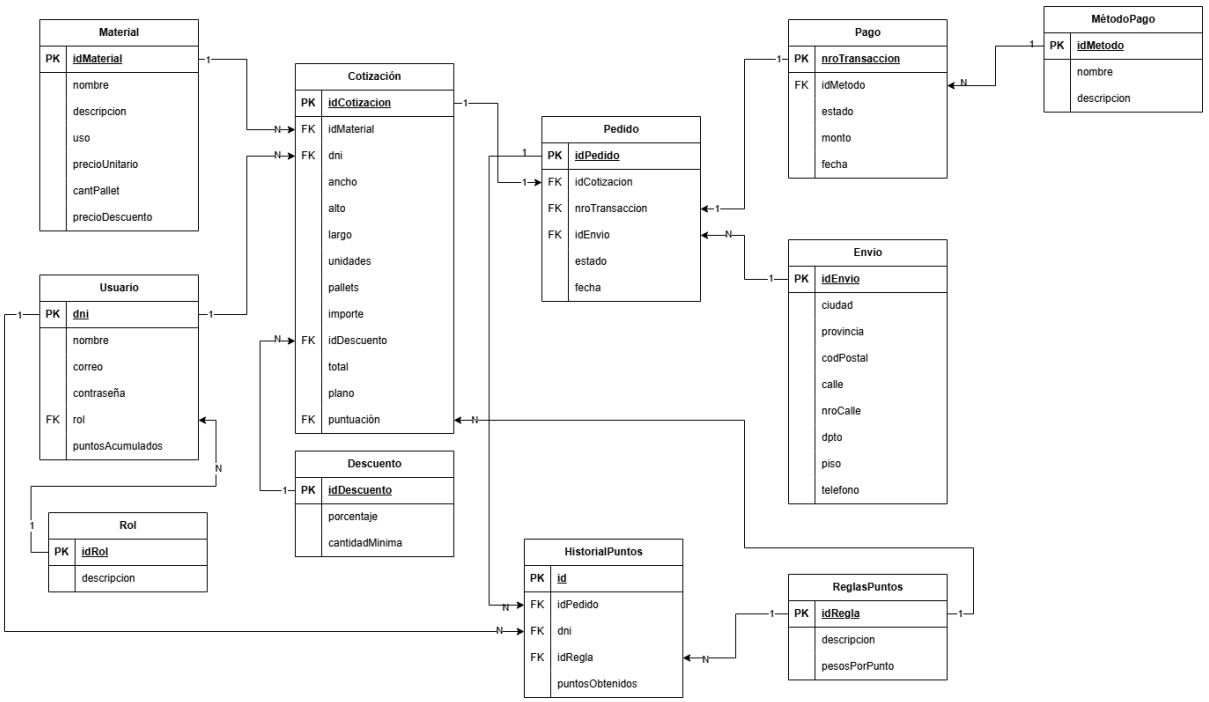
2) Backend (Node.js con Express)
   Gestión de múltiples plataformas de pago: Adaptación de la integración con Mercado Pago para soportar opciones adicionales como MODO.
   Acumulación de puntos de fidelización: Lógica en el backend para registrar los puntos acumulados por cada compra, siguiendo la regla definida en la configuración.
   API Autocad: Configuración de un endpoint para recibir el plano en Autocad, invocar la API externa y almacenar el metraje calculado, facilitando el proceso de cotización.
   Endpoints adicionales:
   - Gestión de puntos: Endpoints para consultar puntos acumulados y modificar la regla de puntos (solo para administradores).
   - Integración de métodos de pago adicionales y gestión de transacciones de cada método.

3) Base de Datos (PostgreSQL)

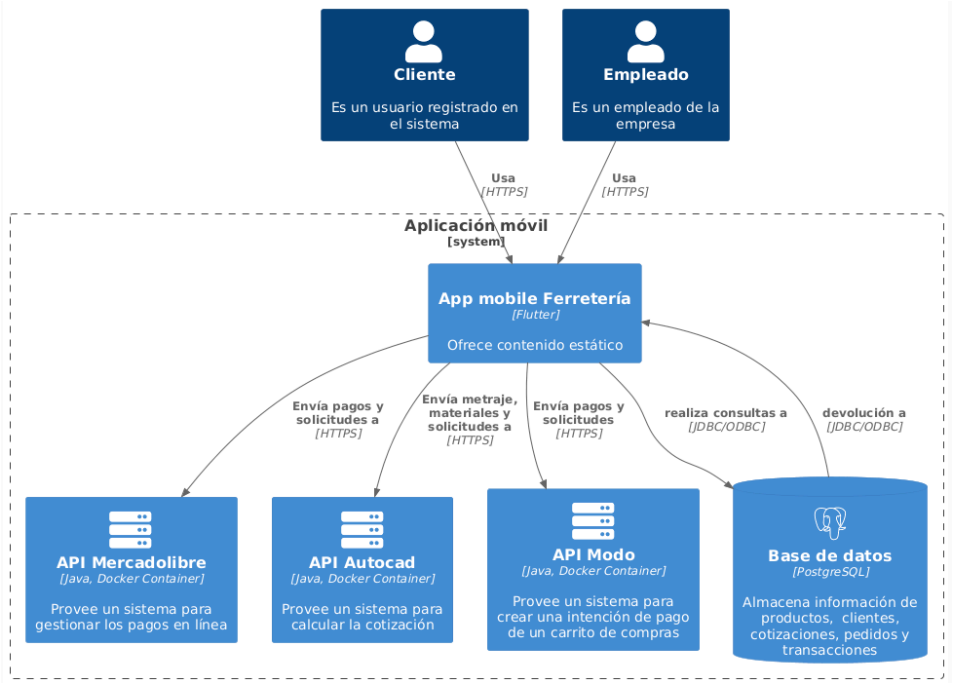
Tablas de fidelización: Tablas para almacenar la regla de puntos y el historial de puntos de los usuarios.

Almacenamiento de planos: Campos adicionales para almacenar los datos de los planos subidos por los usuarios y los resultados de la API de Autocad.

Estructuras para múltiples métodos de pago: Modificación de las tablas de transacciones para diferenciar las plataformas de pago.



Diseño de arquitectura



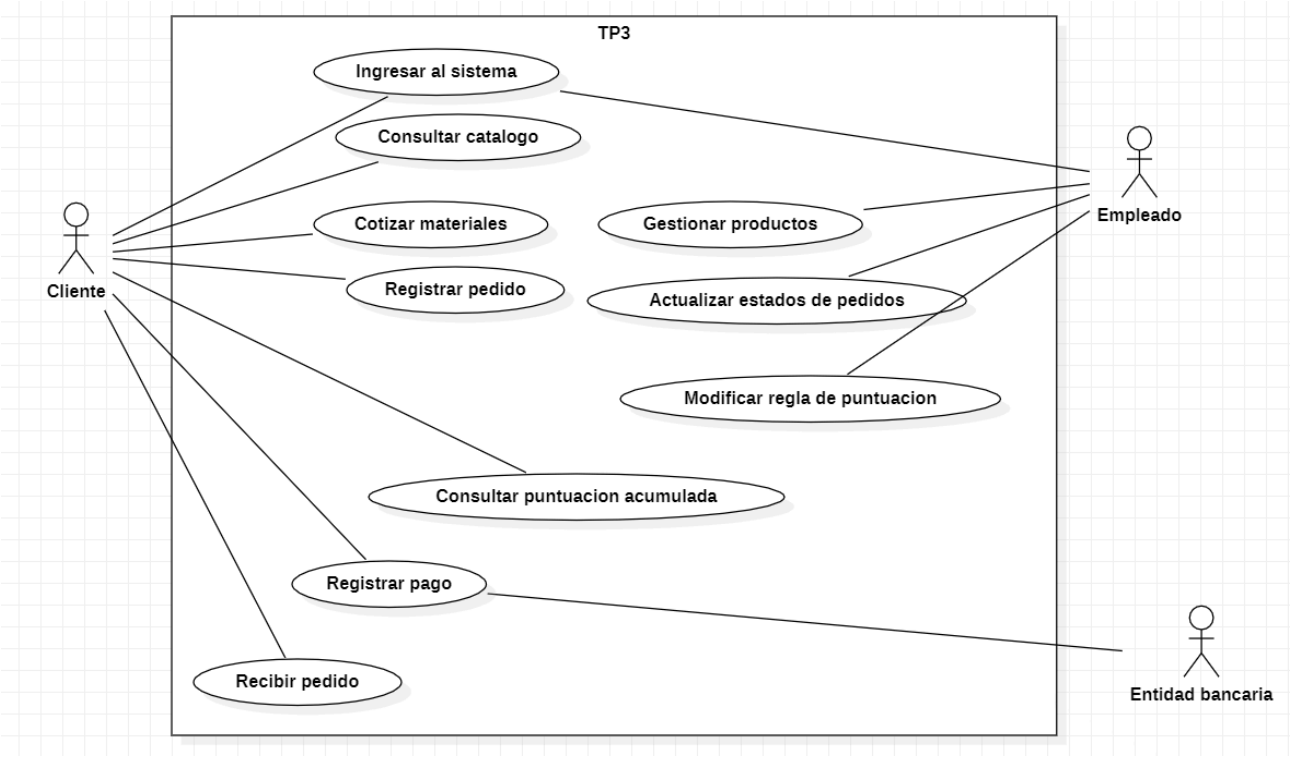
Vista interna del proceso de negocios

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. También puede modificar la regla de puntuación que se obtiene al realizar una compra.
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 cargar un plano a Autocad para obtener los metros cuadrados de pared y techo necesarios.
- 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 múltiples medios de pago: tarjeta de crédito/débito, Mercado Pago y Modo.
- El sistema debe permitir a los usuarios administradores crear, editar y eliminar materiales.
- El sistema debe permitir a los usuarios administradores editar la regla de puntuación que obtendrán los clientes al realizar una compra.
- El sistema debe permitir a los clientes visualizar su puntuación acumulada.
- El sistema debe permitir avisar al cliente los puntos que obtuvo al realizar su compra.

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 reciben

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

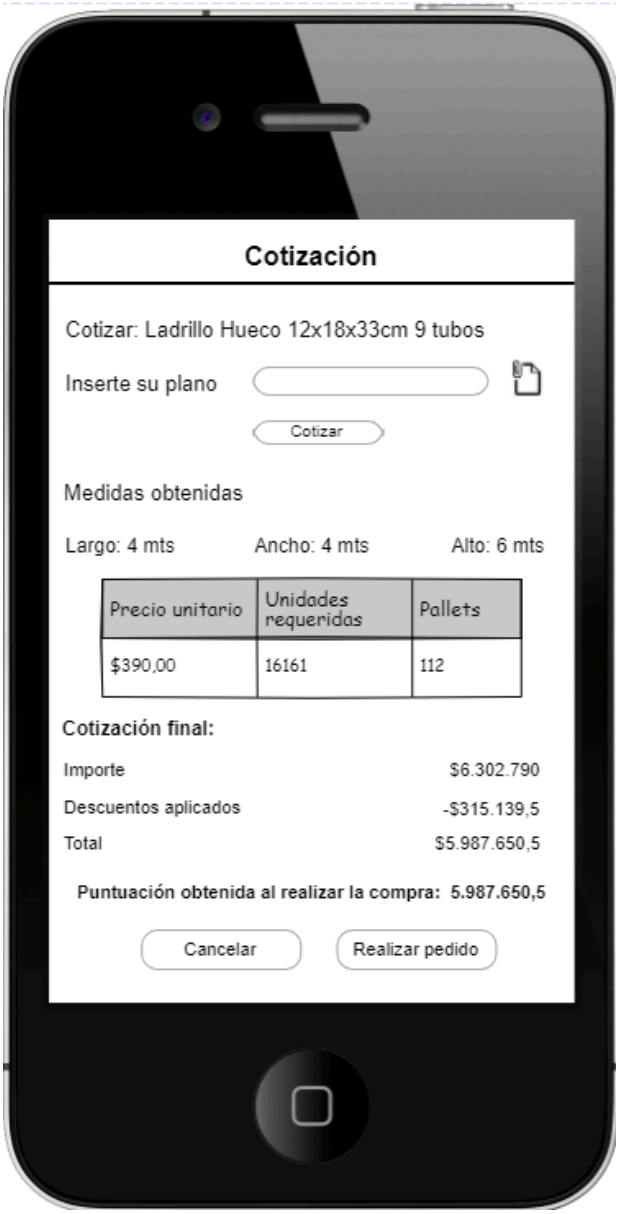
El frontend envía una solicitud al backend solicitando los datos del perfil de ese usuario.

El backend recibe la solicitud y procesa la petición.

El backend consulta la base de datos para buscar los datos del usuario. Esta consulta extrae la información correspondiente a los campos necesarios: nombre, correo, DNI y puntuación acumulada.

La base de datos devuelve los datos al backend. El backend, a su vez, estructura estos datos en un formato adecuado (JSON) y los envía como respuesta al frontend.

El frontend recibe la respuesta con los datos del usuario y los muestra en los campos de la pantalla de perfil.



Datos que se envían	Datos que se reciben

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

	cantPallets:int; puntuación: double; }] }

El usuario carga un plano, el archivo del plano y los detalles del material se envían al backend.

El backend envía el archivo del plano a la API de AutoCAD, que devuelve las dimensiones del proyecto (alto, ancho, y largo).

Con las dimensiones obtenidas, el backend calcula la cantidad de material, precio total, descuentos, cantidad de pallets, y puntuación de compra.

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 = 4 (en metros)

largo = 4 (en metros)

altura = 6 (en metros)

Especificaciones del ladrillo:

tamaño_ladrillo_m2 = 0.18 * 0.33

precioUnitario = 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 = cantidad_ladrillos * precio_por_ladrillo

Especificaciones de las vigas:

separacion_vigas = 0.40 (separación entre vigas en metros)

precioUnitario = 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 = 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.

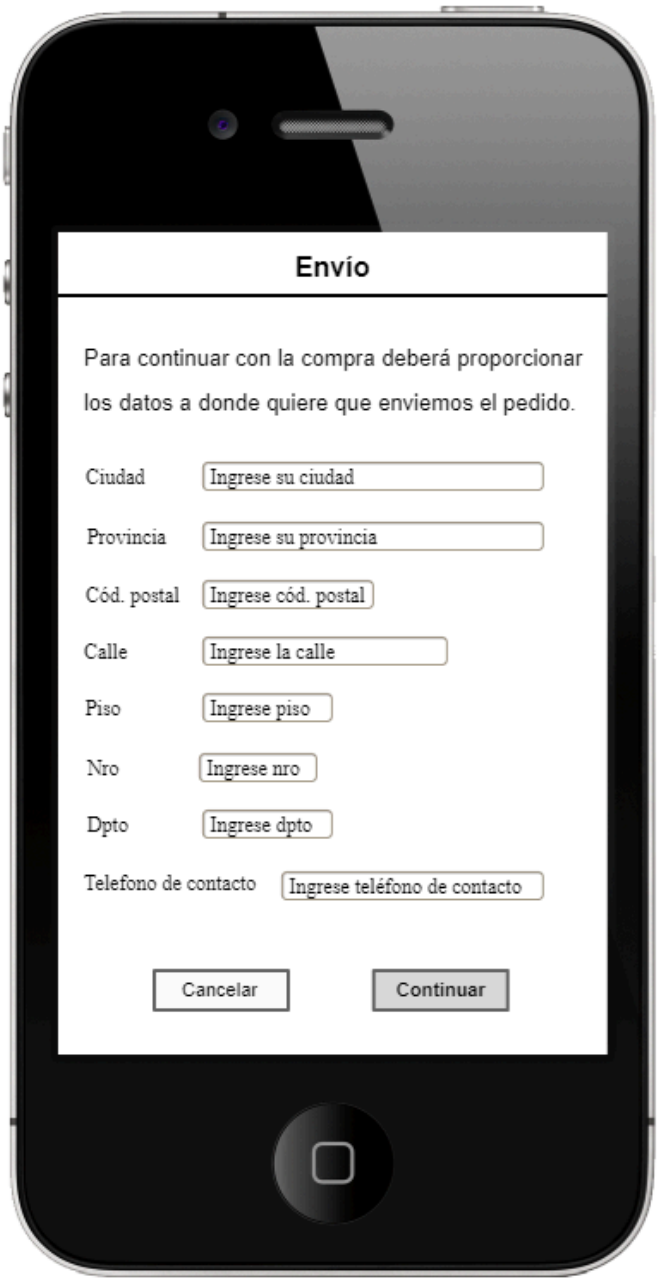
Ejemplo:

```
if cantidad_ladrillos >= 10000:
   importe_total_ladrillos *= 0.95  (Descuento del 5%)
```

También el backend deberá calcular la puntuación que el cliente va a obtener si confirma el pedido:

Actualmente la regla de puntuación es: puntuación = importe

Una vez que el backend ha realizado los cálculos, devuelve un JSON con la cotización completa. Cuando el frontend recibe los datos los muestra en la pantalla de cotización para que el usuario pueda confirmar (pasamos a la pantalla de envío) o cancelar el pedido.



Datos que se envían
{
envio: [{
ciudad: string;
provincia: string;
cod.postal: int;
calle: string;
nro.calle: int;
piso: int;
dpto: int;
telefono: string;
}]
}

El frontend luego de que el usuario decida realizar el pedido, va a solicitar que se proporcionen los datos de envío para el pedido.

Backend recibirá los datos para almacenarlos en la base de datos, más precisamente en la tabla envío que va a estar relacionada con el pedido.

Datos que se envían	Datos que se reciben

{
pago: [{
pedidoId: int;
idMetodo: int;
}]
}
	{
	pago: [{
	nroTransaccion: string;
	estado: string;
	}]

Al hacer clic en una de las opciones, el sistema toma la decisión correspondiente:

- Tarjeta de Crédito/Débito: Abre un formulario de pago con la API de Mercado Pago para que el usuario ingrese los datos de su tarjeta.

- Mercado Pago: Redirige al usuario a la plataforma de Mercado Pago para que complete el pago utilizando el saldo de su cuenta de Mercado Pago.
- MODO: Redirige al usuario a la plataforma de MODO para que realice el pago con el saldo de su cuenta de MODO.

El backend recibe la solicitud de pago desde el frontend. Basado en el método de pago seleccionado, toma las siguientes acciones:

- Tarjeta de Crédito/Débito (API de Mercado Pago):
    - El backend genera una preferencia de pago en la API de Mercado Pago, proporcionando detalles como el monto, la descripción del pedido, y los datos del cliente.
    - Recibe un enlace o token que se envía al frontend para abrir el formulario de pago de Mercado Pago.
- Mercado Pago (Redirección a Cuenta):
    - El backend crea una transacción en la API de Mercado Pago, con la información necesaria para que el usuario pague directamente con el saldo de su cuenta.
    - Retorna un enlace de pago que redirige al usuario a la plataforma de Mercado Pago.
- MODO (Redirección a Cuenta):
    - El backend crea una transacción con la API de MODO y recibe un enlace de pago para redirigir al usuario a la plataforma de MODO.

Cada API proporciona un enlace o token de pago que permite al frontend mostrar el formulario de pago o redirigir al usuario.

Después de completar la transacción, la API de pago (Mercado Pago o MODO) envía una notificación al backend con el estado del pago (aprobado, rechazado, pendiente).

El backend guarda esta información en la base de datos para registrar el estado de la transacción y el método de pago utilizado.

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. El botón descuento te lleva a otra página para editar los descuentos asociados al producto.

Datos que se envían	Datos que se reciben
{	{
materiales: [	materiales: [
{	{
id.material:int;	id.material:int;
nombre: string;	nombre: string;
descripción: string;	descripción: string;
foto: string;	foto: string;
precio: double;	precio: double;
cantPorPallet:int	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	Datos que se reciben
{	{
materiales: [	materiales: [
{	{
id.material:int;	id.material:int;
nombre: string;	nombre: string;
idDescuento : [{	idDescuento : [{
porcentaje: int;	porcentaje: int;
cantidadMinima: int;	cantidadMinima: int;
}]	}]
}]	}]
}	}

Al querer editar el descuento de un material, el backend recupera y nos suministrará los datos previamente cargados del descuento. Una vez aplicados los cambios y al presionar el botón "Aplicar descuento" el backend actualiza el registro de dicho material. Si presionamos el botón “Eliminar descuento” el backend elimina los datos que recuperó previamente.

Datos que se envían
{ materiales: [{ id.material:int; nombre: string; precio: double; descripción: string; uso: string; cantPorPallet:int foto: string; ancho: double; alto: double; largo: double; }] }

Cuando se presiona el botón de añadir, el formulario envía los datos ingresados en el formato JSON al backend. Se verifica si existe algún producto con ese nombre, en caso de que no se encuentre ninguno se crea un nuevo registro con los datos proporcionados.

Datos que se envían	Datos que se reciben

{
ReglasPuntos : [{
pesosPorPuntos: string;
}]
}
	{
	ReglasPuntos : [{
	pesosPorPuntos: string;
	}]
	}

El frontend solicita la regla de puntuación actual al backend, el cual consulta la base de datos y devuelve la regla para que el frontend la muestre en la pantalla.

Cuando el usuario ingresa una nueva regla y presiona "Guardar cambios", el frontend envía la nueva configuración al backend. El backend valida la regla, actualiza el valor en la base de datos y envía una confirmación al frontend.