

**TRABAJO PRÁCTICO N°3 - 2da iteración:**



**Docentes:** Juan Pablo Ferreyra, Pablo Pioli

**Alumnos:** Cena Elian.

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Diseño de Sistemas de Información

Trabajo Práctico N°3

**Contenido**

Enunciado del problema .....	3
Estilo arquitectónico .....	5
Requerimientos de software.....	6
Interfaces con sus flujos de datos .....	7

## Enunciado del problema

Una empresa dedicada a la fabricación de materiales para la construcción se encuentra distribuida en diferentes 3 plantas productivas, una oficina comercial y vendedores que atienden a clientes mayoristas en diferentes zonas.

La sucursal A extrae materia prima que se utiliza como insumo en la planta C. La planta B elabora productos semi-terminados en base a alambres de acero que se utilizan para producir en la planta C. Por su parte, la planta C utiliza elabora ladrillos, vigas de cemento y bloques pre-armados de diferentes medidas. Desde la planta C se realiza el envío de los pedidos directamente al cliente.

Cada planta productiva realiza ingresos de stock de materias primas, consulta de stock, generación de órdenes de producción de los diferentes productos y envío de productos a las diferentes plantas.

Por decisión de la gerencia se necesita reducir los tiempos de atención a clientes minoristas, para ello se pretende ofrecer la posibilidad de cotizar y generar pedidos directamente en el sitio web de la empresa, para ello, una vez identificados los clientes podrán consultar los productos, ejemplo:

	<p><b>Ladrillo Hueco 12x18x33cm 9 tubos</b> <b>Precio por unidad:</b> \$390,00 <b>Descripción:</b> Ladrillo hueco cerámico 12x18x33 cm 9 tubos Ladrillo de cerramiento <b>Uso:</b> Especiales para tabiques divisorios y cerramientos (ambientes interiores y muros de cierre). <b>Cantidad por pallet:</b> 144 unidades</p>
	<p><b>Viga 4 mts</b> <b>Precio por unidad:</b> \$ 10619 <b>Descripción:</b> Ladrillo hueco cerámico 12x18x33 cm 9 tubos Ladrillo de cerramiento <b>Uso:</b> Especiales para tabiques divisorios y cerramientos (ambientes interiores y muros de cierre). <b>Uso:</b> Son utilizadas para techar en la construcción. Se colocan sobre las paredes y van acompañadas entre viga y viga por ladrillos para techo y malla sima.</p>

## **INGENIERÍA EN SISTEMAS DE INFORMACIÓN**

Diseño de Sistemas de Información

Trabajo Práctico N°3

Podrán cotizar, ingresando cantidad de metros cuadrados a construir y tipos de materiales, en base a dicha información se debería poder determinar la cantidad de materiales necesarios, por ejemplo: Para construir un galpón de 40m x 40m, de 6m de altura, con ladrillo de tipo bloques de 18cm x 33cm se necesitaría cubrir una superficie de 960 metros cuadrados, con lo cual la cantidad de ladrillos, considerando una separación de 40 cm entre vigas, se necesitaría:

- 16161 ladrillos, equivalentes a 112,23 pallets
- Importe \$ 6.302.790.

Se debería poder gestionar los descuentos por cantidad, por ejemplo, a partir de los 10mil ladrillos ofrecer un 5% de descuento sobre el valor del producto.

A partir de dicha cotización el cliente podrá realizar un pedido, debiendo completar información de domicilio de envío. La empresa cuenta con servicio de envío.

Una vez aprobado el pedido, se acuerda una forma de pago. Una vez que el cliente realiza el pago se envía el pedido.

Debido al éxito del sitio web desarrollado, la Gerencia está entusiasmada y desea desarrollar una app mobile que brinde las funciones actuales del sitio web, pero que además permita a los clientes:

- Pagar los pedidos realizados mediante diferentes plataformas de pago, modo, mercadopago, etc.
- Sumar puntos por cada compra realizada, los puntos se definen como una cierta cantidad  $n$  de puntos cada  $x$  cantidad de pesos gastados. Dicha regla:  $n$  (puntos) =  $x$  (pesos) podrá ser modificada por un usuario administrador.
- Implementar una API existente (de otra empresa) que a partir de un plano Autocad devuelve los metros cuadrados de pared y techo necesarios. Esto le permitirá al cliente facilitar la determinación de metros cuadrados de pared y techo, esta información se deberá tomar de insumo en la cotización.

## Estilo arquitectónico

Para este problema específico utilizaremos un estilo arquitectónico en capas, ya que nos permite tener varias capas separadas independientemente, cada una con una responsabilidad distinta. Este enfoque nos permite mejorar la modularidad y la mantenibilidad de nuestro software.

El sistema tendrá las siguientes capas:

- Capa de frontend: Permite a los clientes minoristas interactuar directamente con la interfaz web. Esto incluye poder ingresar al sistema, registrarse, visualizar los productos, un formulario para cotización, una interfaz para visualizar el pedido y otra para realizar el pago.
- Capa de backend: Una API que sirve como intermediario entre el frontend y la base de datos. Luego se utilizarán 2 API más, una para MercadoPago y otra para AutoCAD.
- Capa de base de datos: Encargada de mantener los datos a lo largo del tiempo de los usuarios, la información de los productos, las cotizaciones y pedidos.

## Requerimientos de software

### Requerimientos funcionales:

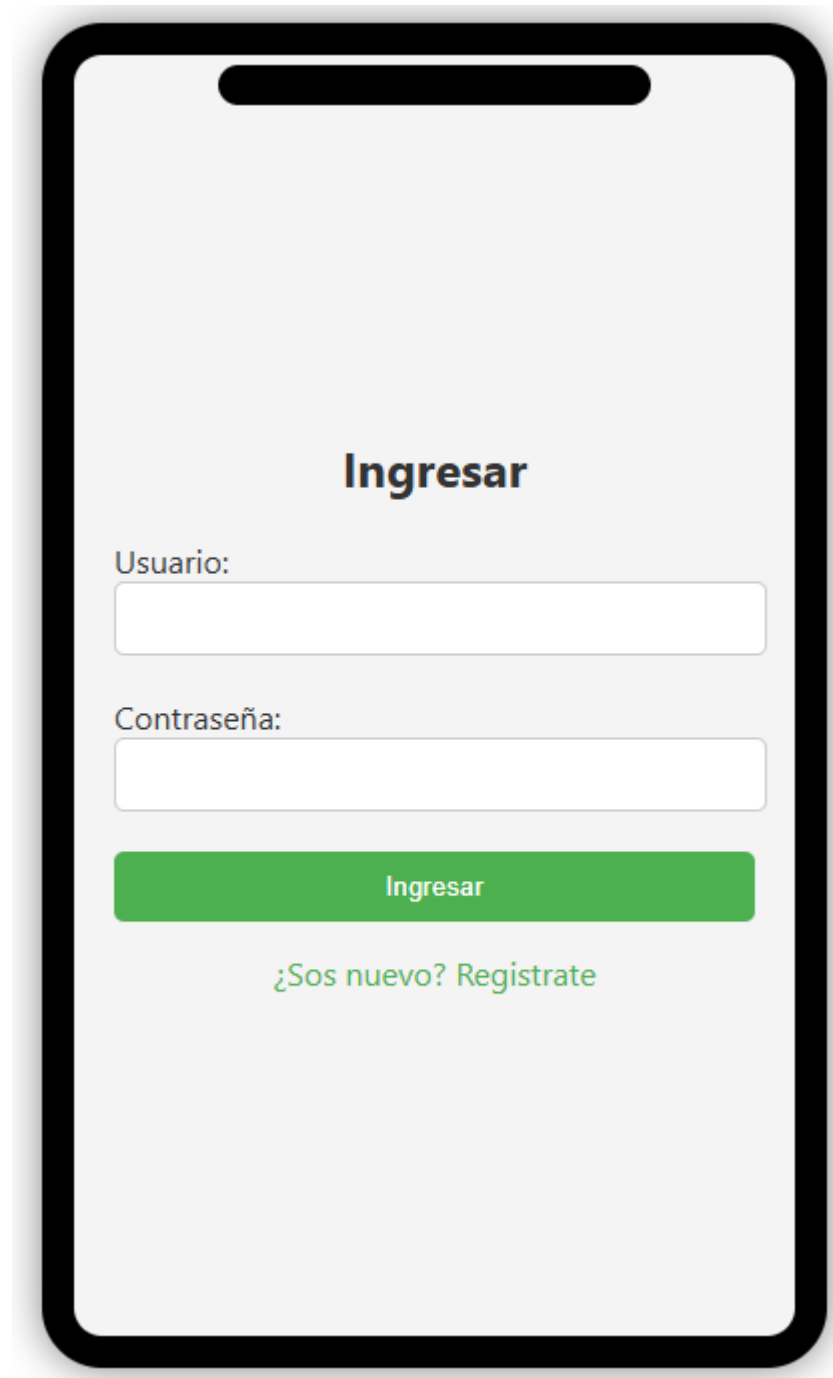
- El sistema debe permitir registrar un usuario
- El sistema debe permitir que un usuario inicie sesión.
- El sistema debe permitir a los clientes generar cotizaciones.
- El sistema debe permitir a los clientes ingresar un archivo AutoCAD que calcule los metros cuadrados necesarios automáticamente.
- El sistema debe gestionar descuentos por comprar grandes cantidades.
- El sistema debe permitir a los clientes aceptar la cotización.
- El sistema debe permitir a los clientes generar un pedido basado en la cotización previa.
- El sistema debe permitir a los clientes aceptar el pedido.
- El sistema debe permitir a los clientes ingresar una dirección para el envío.
- El sistema debe permitir a los clientes múltiples opciones de pago incluyendo MercadoPago.
- El sistema debe permitir a los clientes buscar los materiales por categoría.
- El sistema debe permitir sumar puntos por cada compra realizada.

### Requerimientos no funcionales:

- El sistema debe ser capaz de manejar un gran volumen de usuarios.
- El sistema debe soportar una gran carga de cotizaciones al mismo tiempo.
- El sistema debe soportar una gran carga de pedidos al mismo tiempo.
- El tiempo de generar una cotización debe ser menor a 5 segundos.
- El sistema debe ser compatible con diferentes navegadores.

## Interfaces con sus flujos de datos

### Interfaz para ingresar un usuario ya registrado



The image shows a mobile application interface for logging in. It features a light gray background with a black border. At the top, there is a black notch. The title "Ingresar" is centered in a bold, black font. Below the title, there are two input fields: "Usuario:" and "Contraseña:". Each field has a white background and a light gray border. Below the "Contraseña:" field, there is a green button with the text "Ingresar" in white. At the bottom, there is a link that says "¿Sos nuevo? Registrate" in a green font.

**Ingresar**

Usuario:

Contraseña:

Ingresar

¿Sos nuevo? Registrate

**Flujo de Datos cuando el usuario hace click en el botón "Ingresar"**

**Frontend**

Se hace un metodo HTTP POST a la API, Los datos que se envían son usuario y contraseña. Los datos se envían en formato JSON.

```
{  
  "usuario": "Juan",  
  "contraseña": "123"  
}
```

**API**

La API recibe la solicitud POST el usuario y la contraseña.

La API envía una consulta SELECT a la base de datos que busca el usuario y la contraseña en la tabla correspondiente.

```
SELECT usuario, contraseña  
WHERE usuario = 'Juan' and contraseña = '123';
```

Si la consulta no devuelve resultados, significa que el usuario o contraseña no existe en la base de datos. La API responde con un error al frontend.

Si el usuario y contraseña son válidos la API genera una respuesta de éxito al frontend.

**Frontend**

Si el inicio de sesión es exitoso, el frontend redirige al usuario a la lista de materiales.

Si el inicio de sesión falla, el frontend muestra un mensaje de error indicando que la contraseña o el usuario son incorrectas.



**Interfaz para registrar un nuevo usuario**

**Registro**

Nombre:

Apellido:

Email:

Usuario:

Contraseña:

Repetir contraseña:

☐ Acepto términos y condiciones

☐ Recibir ofertas y promociones

**Registrar**

**Flujo de Datos cuando el usuario hace click en el botón "Registrar"**

**Frontend:**

Se hace un método HTTP POST a la API.

El sistema realiza validaciones inmediatas en el frontend antes de enviar los datos, como verificar que las contraseñas coinciden, se ha aceptado el checkbox de términos y condiciones, no haya campos vacíos o incorrectos.

Los datos se envían en un JSON con el siguiente formato.

```
{  
  "nombre": "nombre1",  
  "apellido": "apellido1",  
  "email": "email1",  
  "usuario": "usuario1",  
  "contraseña": "contraseña1"  
}
```

**API**

La API recibe esta solicitud y se comunica con la base de datos para guardar el nuevo usuario a través de una consulta SQL INSERT INTO para insertar los datos del nuevo usuario:

```
INSERT INTO (nombre, apellido, email, usuario, contraseña)  
VALUES ('nombre1', 'apellido1', 'email1', 'usuario1', 'contraseña1');
```

La API responde al frontend, indicando si el registro fue exitoso o si hubo algún error (por ejemplo, si el email ya está registrado).

**Base de datos**

La base de datos confirma que recibió un nuevo registro y devuelve un mensaje de éxito o error a la API.

**API**

La API recibe la respuesta de la base de datos y devuelve el mensaje al frontend:

Si el registro fue exitoso se devuelve una respuesta de éxito.

Si hubo un error la API devuelve un mensaje de error.

**Frontend**

Dependiendo de la respuesta de la API, el frontend puede mostrar un mensaje de registro exitoso o de error. Si es exitoso, el sistema redirige al usuario a la página para ingresar.

**Interfaz para ver la lista de productos**

## Catálogo

Puntos acumulados: 0

Filtrar por:

Mostrando 1 página de 6 ->



**Ladrillos huecos 0,1 x 0,1 x 0,1**

A partir de 1000 unidades descuento del 5%.

Cantidad por pallet: 144

Precio por unidad: \$10



**Chapa ondulada 1,5 x 2**

A partir de 100 unidades descuento del 3%.

Cantidad por pallet: 30

Precio por unidad: \$50

## INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Diseño de Sistemas de Información

Trabajo Práctico N°3

### **Flujo de datos al abrir la página principal luego de ingresar con un usuario**

El usuario ve el listado de productos disponibles, puede seleccionar cualquiera para cotizar su precio o incluso puede filtrar por categoría.

#### **Frontend**

Envía una solicitud HTTP GET a la API, solicitando la lista de productos.

En este caso, no es necesario enviar datos en el cuerpo de la solicitud GET solo se envía un token de autenticación.

#### **API**

La API recibe la solicitud del frontend y realiza una consulta SQL a la base de datos para recuperar la lista de productos.

SELECT id\_material, nombre\_producto, puntos, categoria, cantidad\_pallet, precio\_unidad, imagen\_url, descuento, ancho, alto, largo.

#### **Base de datos:**

La base de datos recibe la consulta SQL, la procesa, y devuelve los datos solicitados.

id_material	nombre_producto	puntos	categoría	cantidad_pallet	precio_unidad	imagen_url	descuento	ancho	alto	largo
1	Ladrillo hueco	0	Ladrillos	144	10	www.imagen ladrillo.com	5	0.1	0.1	0.1

**API**

Una vez que la API ha recibido los datos de la base de datos, los procesa y los prepara para ser enviados al frontend en formato JSON.

```
[
  {
    "id_material": 1,
    "nombre_producto": "Ladrillo hueco",
    "puntos": 0,
    "categoria": "Ladrillos",
    "cantidad_pallet": 144,
    "precio_unidad": 10,
    "imagen_url": "www.imagenladrillo.com",
    "descuento": 5,
    "ancho": 0.1,
    "alto": 0.1,
    "largo": 0.1
  }
]
```

**Frontend**

El frontend recibe la respuesta JSON de la API. A partir de aquí, los datos se utilizan para actualizar la interfaz de usuario, mostrando la lista de productos en pantalla.

## Flujo de Datos cuando el usuario hace presiona en el botón

### “Buscar”

#### Frontend:

Se hace un método HTTP GET a la API.

El sistema realiza validaciones inmediatas en el frontend antes de enviar los datos, como verificar que no este vacío la casilla de categoría.

GET /api/catalogo?categoria=Ladrillos

#### API

La API recibe la solicitud del frontend y realiza una consulta SQL a la base de datos para recuperar la lista de productos.

SELECT id\_material, nombre\_producto, puntos, categoria, cantidad\_pallet, precio\_unidad, imagen\_url, descuento, ancho, alto, largo.

WHERE “categoría” = Ladrillos.

#### Base de datos:

La base de datos recibe la consulta SQL, la procesa, y devuelve los datos solicitados.

id_material	nombre_producto	puntos	categoría	cantidad_pallet	precio_unidad	imagen_url	descuento	ancho	alto	largo
1	Ladrillo hueco	0	Ladrillos	144	10	www.imagen ladrillo.com	5	0.1	0.1	0.1

#### API

Una vez que la API ha recibe los datos de la base de datos, los procesa y los prepara para ser enviados al frontend en formato JSON.

```
[  
  {  
    "id_material": 1,
```

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Diseño de Sistemas de Información

Trabajo Práctico N°3

```
"nombre_producto": "Ladrillo hueco",  
"puntos": 0,  
"categoria": "Ladrillos",  
"cantidad_pallet": 144,  
"precio_unidad": 10,  
"imagen_url": "www.imagenladrillo.com",  
"descuento": 5,  
"ancho": 0.1,  
"alto": 0.1,  
"largo": 0.1  
}  
]
```

**Frontend**

El frontend recibe la respuesta JSON de la API. A partir de aquí, los datos se utilizan para actualizar la interfaz de usuario, mostrando la lista de productos en pantalla.



**Flujo de Datos cuando el usuario hace click en el botón “cotizar”**

**Frontend**

Se envía una solicitud GET a la API ya que estás solicitando cotizar un producto que se va a ver en otra pantalla. Se envía solo el id\_material del producto porque el backend ya tiene todos los datos relacionados con ese producto. Los datos se envían como parámetros en la URL de la solicitud.

GET /api/cotizacion?idMaterial=1

**API**

La API recibe la solicitud del frontend y realiza una consulta SQL a la base de datos para recuperar los datos necesarios para mostrar la pantalla de cotizar.

```
select nombre_producto, cantidad_pallet, precio_unidad, imagen_url, descuento, ancho, alto, largo where id_material = 1
```

Esos datos son necesarios para mostrar información y para realizar los cálculos correspondientes.

**Base de datos:**

La **base de datos** recibe la consulta SQL, la procesa, y devuelve los datos solicitados.

id_material	nombre_producto	descuento	cantidad_pallet	precio_unidad	imagen_url	ancho	alto	largo
1	Ladrillo hueco	5	144	10	www.imagen ladrillo.com	0.1	0.1	0.1

**API**

Una vez que la API ha recibido los datos de la base de datos, los procesa y los prepara para ser enviados al **frontend** en formato **JSON**.

```
[  
  {  
    "id_material": 1,  
    "nombre_producto": "Ladrillo hueco",  
    "descuento": 5,  
    "cantidad_pallet": 144,  
    "precio_unidad": 10,  
    "imagen_url": "www.imagenladrillo.com",  
    "ancho": 0.1,  
    "alto": 0.1,  
    "largo": 0.1  
  }  
]
```

### **Frontend**

El frontend recibe la respuesta JSON de la API y la procesa. A partir de aquí, los datos se utilizan para actualizar la interfaz de usuario, mostrando la pantalla para cotizar el producto seleccionado.

**Interfaz para ver una cotización**

### Ladrillos huecos



Sube tu archivo de plano (AutoCAD):  
 Sin archivos seleccionados

Metros cuadrados de pared: 960 m<sup>2</sup>  
Metros cuadrados de techo: 1600 m<sup>2</sup>  
Cantidad ladrillos: 16161  
Cantidad pallets: 112.23  
Descuentos por cantidad: \$315,140  
Precio final: \$6,302,790

## **Flujo de Datos al Presionar el Botón "Calcular"**

El usuario ingresa un archivo de un plano AutoCAD

### **Frontend**

Al apretar el botón "Calcular" se envía una solicitud post (los métodos post son útiles para enviar archivos) a la API junto con el archivo.

```
{  
  "IdMaterial": "1",  
  "archivo": "archivo.dwg",  
}
```

### **API AUTOCAD**

Recibe el archivo.dwg, lo analiza y obtiene los metros cuadrados de pared y techo.

Una vez que tiene los datos, los envía al frontend.

### **Frontend**

Usando los datos recibidos por la API de AutoCAD, el frontend realiza los cálculos necesarios para determinar la cantidad de ladrillos requeridos, la cantidad de pallets necesarios, los descuentos si hubiera y el precio final.

Este cálculo se realiza utilizando una función de algún lenguaje de programación que recibe los datos ingresados y las características del producto como dimensiones de los ladrillos y el precio unitario.

Una vez completados los cálculos, los resultados se muestran en la misma pantalla de cotización.

## **Flujo de Datos al Presionar el Botón "Aceptar cotización"**

### **Frontend**

Se hace un método POST a la API con los datos de la cotización ya que se envía información del producto. Los datos con formato JSON son id\_material, cantidad\_material, cantidad\_pallet, precio\_final.

```
{
  "IdMaterial": "1",
  "cantidad_material": 16161,
  "cantidad_pallet": 113,
  "precio_final": 6.302.790
}
```

### **API**

El backend realiza una consulta SQL para registrar el pedido en la base de datos.

```
INSERT INTO pedidos (id_material, cantidad_material, cantidad_pallets,
precio_final)
VALUES (1, 16161, 113, 6.302.790);
RETURNING id_pedido;
```

En este caso, la base de datos devuelve el ID del pedido recién creado, que el backend puede utilizar para referencia.

Una vez que el pedido ha sido registrado correctamente en la base de datos, el backend devuelve una respuesta al frontend. Esta respuesta podría ser un mensaje de éxito o error si hubo un fallo en la base de datos y junto a eso un id del pedido.

```
{
  "mensaje": "Pedido registrado con éxito",
  "id_pedido": 6789
}
```

**Frontend**

Una vez que el frontend recibe la respuesta, si todo ha salido bien, se redirige al usuario a la siguiente pantalla donde se muestra el resumen de la cotización y se solicita la información de envío.

**Flujo de Datos al Presionar el Botón "Rechazar cotización"**

Se procede con toda la carga del listado de productos como se dijo anteriormente.

**Interfaz para ver un pedido**

## Pedido

### Productos cotizados



Ladrillos huecos

Cantidad ladrillos: 16161

Cantidad pallet: 113

Precio final: \$6.302.790

[Eliminar](#) [Agregar nuevo producto](#)

### Datos de envío

Nombre  Provincia

Apellido  Ciudad

DNI  Código postal

Teléfono  Dirección

CancelarAceptar

## **Flujo de Datos al Presionar el Botón "Aceptar"**

### **Frontend**

Se hace un método HTTP POST a la API, los datos que se envían para ser registrados junto con el pedido son:

```
{
  "nombre": "nombre1",
  "apellido": "apellido1",
  "dni": 1,
  "telefono": 1,
  "provincia": "provincia1",
  "ciudad": "ciudad1",
  "código_postal": 1,
  "direccion": "direccion1"
}
```

### **API**

El backend recibe los datos de envío y los asocia al pedido que está siendo procesado.

Se hace una consulta SQL para guardar los datos del cliente y su dirección de envío en la base de datos, junto con los detalles del pedido.

```
INSERT INTO pedidos (nombre, apellido, dni, teléfono, provincia, ciudad,
código_postal, dirección)
VALUES ('nombre1', 'apellido1', 1, 1, 'provincia1', 'ciudad1', 1, 'direccion1');
WHERE id_pedido = 6789;
```

El backend devuelve una **respuesta al frontend**. Esta respuesta podría ser un mensaje de éxito o error si hubo un fallo en la base de datos

### **Frontend**

El frontend redirige al usuario a la pantalla de pago en caso de éxito, en caso de error aparece un mensaje en la pantalla.



### **Flujo de Datos al Presionar el Botón "Cancelar"**

Se procede con la interfaz para ver la cotización de un producto.

### **Flujo de Datos al Presionar el Botón "Agregar nuevo producto"**

Se procede a la carga de la interfaz para ver la lista de productos con sus respectivos botones para cotizar, una vez aceptada la cotización se llega nuevamente a la interfaz del pedido con todos los productos.

### **Flujo de Datos al Presionar el Botón "Eliminar"**

#### **Frontend**

Se hace un método HTTP POST a la API para eliminar ese producto de la lista de productos cotizados en el pedido.

```
{  
  "idMaterial": 1,  
  "idPedido": 6789  
}
```

#### **API**

El backend recibe los datos.

Se hace una consulta SQL para eliminar el producto del pedido.

```
DELETE FROM pedido WHERE idMaterial = 1 AND idPedido = 6789;
```

El backend devuelve una **respuesta al frontend**. Esta respuesta podría ser un mensaje de éxito o error si hubo un fallo en la base de datos


#### **Frontend**


El frontend actualiza el pedido, eliminando el producto seleccionado, y si no queda ninguno vuelve a la pantalla principal para ver la lista de productos.


**Interfaz para realizar el pago con tarjetas**

## Pago

### Forma de pago

☒ Tarjeta de crédito 

☐ Tarjeta de débito 

☐ MercadoPago 

### Datos del pago

Nombre

Número de tarjeta

Código de seguridad

MM  AA

## **Flujo de Datos al Presionar el Botón "Aceptar"**

### **Frontend**

Se hace un metodo HTTP POST a la API, Los datos que se envían incluyen la información del pedido y los detalles de pago:

```
{  
  "nombre": "nombre1",  
  "numero_tarjeta": 5042 5013 5012 5007,  
  "código_seguridad": 245,  
  "fecha_caducidad": 07/24  
}
```

### **API**

Si la forma de pago es con tarjeta de crédito o débito, la API se comunica con un servicio de pagos externo (como visa u otro) para validar y procesar el pago. Estos datos se envían en formato JSON:

```
{  
  "moneda": "ARS",  
  "precio": 6.302.790,  
  "descripcion": "Pago de pedido",  
  "nombre": "nombre1",  
  "numero_tarjeta": 5042 5013 5012 5007,  
  "código_seguridad": 245,  
  "fecha_caducidad": 07/24  
}
```

Una vez que el pago ha sido procesado correctamente por el servicio de pago, la API calcula los puntos basándose en el precio del pedido. (puntos = [n puntos=x pesos] en este caso se decidió que sea 1 punto=1 peso) y luego procede a interactuar con la base de datos

```
INSERT INTO pedidos (id_pedido, descripción_pago)  
VALUES (1, 'Pago aceptado');
```

## INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Diseño de Sistemas de Información

Trabajo Práctico N°3

Se actualizan los puntos del usuario:

UPDATE usuario

SET puntos = puntos + 6302790

La API luego responde al frontend, confirmando que el pedido ha sido registrado, que el pago ha sido procesado exitosamente y los puntos han sido sumados.

### **Frontend**

El frontend recibe esta respuesta y finaliza el proceso de compra, retornando a la pantalla principal que muestra la lista de productos.

### **Flujo de Datos al Presionar el Botón "Cancelar"**

Se procede con la interfaz para ver el pedido de un producto.

**Interfaz para realizar el pago con MercadoPago**



## **Flujo de Datos cuando se muestra el qr y cuando se escanea**

### **Frontend**

El usuario selecciona la forma de pago con tarjeta o MercadoPago.

El frontend genera un código QR con los datos de pago:

```
{
  "nombre": "nombre1",
  "numero_tarjeta": "5042 5013 5012 5007",
  "codigo_seguridad": "245",
  "fecha_caducidad": "07/24",
  "precio": 6.302.790,
  "moneda": "ARS",
  "descripcion": "Pago de pedido"
}
```

El código QR generado es enviado al usuario en la interfaz, donde puede ser escaneado.

Una vez que el QR es escaneado, la aplicación o terminal de pago extrae la información contenida en el código QR y la utiliza para realizar la transacción.

### **API MercadoPago**

Cuando recibe los datos del QR, realiza una solicitud HTTP POST a la API con los mismos datos del pago extraídos del código QR:

```
{
  "nombre": "nombre1",
  "numero_tarjeta": "5042 5013 5012 5007",
  "codigo_seguridad": "245",
  "fecha_caducidad": "07/24",
  "precio": 6.302.790,
  "moneda": "ARS",
  "descripcion": "Pago de pedido"
}
```

## INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Diseño de Sistemas de Información

Trabajo Práctico N°3

Si el pago es aprobado, la API interactúa con la base de datos para registrar el pedido:

```
INSERT INTO pedidos (id_pedido, descripcion_pago)
VALUES (1, "Pago aceptado");
```

Y actualiza los puntos del usuario:

```
UPDATE usuario
SET puntos = puntos + 6302790
```

En este caso, 1 punto = 1 peso, por lo que el usuario obtiene 6,302,790 puntos.

### **Frontend**

Si el pago es exitoso, el frontend puede mostrar un mensaje que indique que el pago fue procesado correctamente, que los puntos fueron añadidos al usuario y redirige al usuario a la pantalla principal, si el pago falla se envía un mensaje de alerta.

### **Flujo de Datos al Presionar el Botón "Cancelar"**

Se procede con la interfaz para ver el pedido de un producto.