

UTN FACULTAD REGIONAL SAN FRANCISCO

Trabajo Práctico 3

Primera iteración

DISEÑO DE SISTEMAS DE INFORMACIÓN 2024

Alumna: Cleri, Martina
martinabelencleri@gmail.com

1. ENUNCIADO

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>Ladrillo Hueco 12x18x33cm 9 tubos Precio por unidad: \$390,00 Descripción: Ladrillo hueco cerámico 12x18x33 cm 9 tubos Ladrillo de cerramiento Uso: Especiales para tabiques divisorios y cerramientos (ambientes interiores y muros de cierre). Cantidad por pallet: 144 unidades</p>
	<p>Viga 4 mts Precio por unidad: \$ 10619 Descripción: Ladrillo hueco cerámico 12x18x33 cm 9 tubos Ladrillo de cerramiento Uso: 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>

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.

2. ARQUITECTURA EN CAPAS

Para la situación presentada, implementaremos una arquitectura en capas debido a su simplicidad, ya que nos permite entender de forma clara cuál es la responsabilidad de cada parte del sistema. Además, su estructura clara nos da una ventaja en su mantenimiento al ser más sencillo realizar cambios o mejorar partes específicas del sistema sin afectar a otras capas. El sistema contendrá las siguientes capas:

- **Capa Frontend – Sitio Web:**

Esta capa se encargaría de la interacción con el usuario, mostrando productos, permitiendo realizar cotizaciones y pedidos.

La interfaz de usuario sería implementada en React, que se comunica con el backend a través de peticiones HTTP.

- **Capa Backend – API RESTful:**

Implementada en Node.js con Express, esta capa contiene la lógica de negocio. Gestiona las cotizaciones, los pedidos y los descuentos, y se comunica con la base de datos a través de la capa de acceso a datos (DAL).

- **Capa de Persistencia (Data Access Layer):**

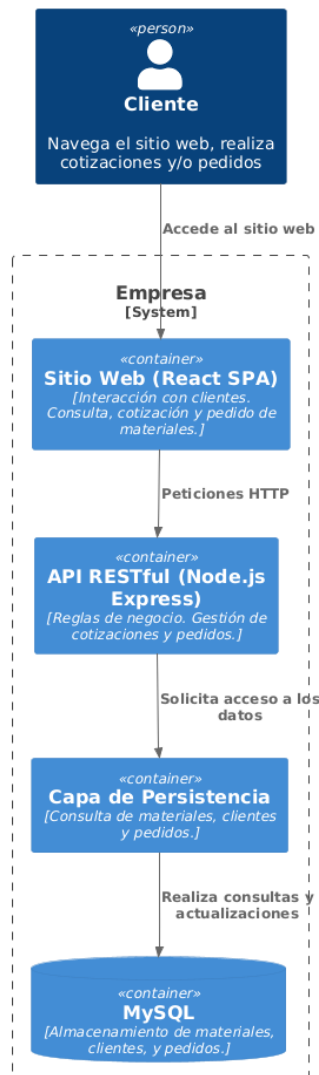
Esta capa gestionaría la comunicación con la base de datos como consultas SQL, leer y/o escribir datos.

- **Capa de Base de Datos:**

En esta capa se van a almacenar datos estructurados como productos, clientes, pedidos y cotizaciones en un sistema de base de datos relacional (MySQL).

DIAGRAMA DE ARQUITECTURA:

Sistema de Cotizaciones y Pedidos



3. REQUERIMIENTOS

3.1 REQUERIMIENTOS FUNCIONALES

RF1. El sistema debe garantizar que solo usuarios autenticados puedan acceder a la lista de materiales y las funcionalidades para generar cotizaciones y/o pedidos.

RF2. El sistema debe redirigir al usuario a una pantalla de registro/inicio de sesión en caso de no estar autenticado.

RF3. El sistema debe permitir al cliente ver la lista de materiales disponibles.

RF4. El sistema debe mostrar detalles del material como imagen, nombre, precio unitario y descripción.

RF5. El sistema debe permitir al cliente seleccionar un material de la lista para gestionar su cotización.

RF6. El sistema debe permitir al cliente ingresar las medidas de construcción en tres campos: largo, alto y ancho (opcional).

RF7. El sistema debe calcular el precio total y cantidad de unidades necesarias en función de las dimensiones ingresadas.

RF8. El sistema debe aplicar un descuento al precio total en caso de superarse X unidades del material seleccionado.

RF9. El sistema debe mostrar el resultado de la cotización al cliente incluyendo precio total, descuento (si corresponde) y cantidad de unidades necesarias del material seleccionado.

RF10. El sistema debe permitir al cliente confirmar la cotización como una orden de pedido.

RF11. El sistema debe mostrar al cliente un resumen de pedido que incluya los detalles del anterior.

RF12. El sistema debe permitir la gestión de pagos mediante tarjetas de crédito.

RF13. El sistema debe permitir al cliente completar un formulario con los datos de envío.

3.2 REQUERIMIENTOS NO FUNCIONALES

RNF1. El sistema debe ser alta disponibilidad (99.9%) para que los clientes puedan acceder al sistema en cualquier momento.

RNF2. El sistema debe realizar la cotización en menos de 3 segundos, y el tiempo de respuesta para las demás consultas no deben exceder 1 segundo.

RNF3. El sistema debe cumplir con las normativas locales de protección de datos y comercio electrónico.

RNF4. El sistema debe garantizar que los datos de los usuarios y las transacciones estén protegidos mediante el uso de HTTPS para la comunicación entre el frontend y backend.

RNF5. El sistema debe ser compatible con los navegadores web Chrome, Firefox, Safari y Edge.

RNF6. El sistema debe funcionar adecuadamente en diferentes dispositivos (móviles, tablets y PCs).

RNF7. El sistema debe tener un diseño responsive para el frontend de forma que la experiencia de usuario sea consistente en todos los tamaños de pantalla.

4. INTERFACES DE USUARIO

INTERFAZ REGÍSTRATE:



Regístrate

Nombre

Apellido

Correo electrónico

Contraseña

Confirmar contraseña

Registrarse

[¿Ya tienes cuenta? Iniciar sesión](#)

FLUJO AL HACER CLIC EN REGISTRARSE:

1. Se envía una solicitud HTTP POST al backend con los datos proporcionados por el usuario:

```
{
  "nombre_cliente": "Martina",
  "apellido_cliente": "Cleri",
  "email": "martina@gmail.com",
  "password": "123456"
}
```
2. El backend verifica si ya existe un usuario con el mismo correo electrónico en la base de datos. Si no existe, prosigue con la creación del cliente.
3. Se genera un nuevo registro en la tabla de clientes:

```
INSERT INTO clientes (nombre_cliente, apellido_cliente, email, password) VALUES ('Martina', 'Cleri', 'martina@gmail.com', '123456');
```
4. Si el registro fue exitoso, el backend devuelve una respuesta confirmando que el usuario ha sido creado:

```
{
  "mensaje": "Su cuenta ha sido registrada correctamente."
}
```

```
"id_cliente": 1
}
```

INTERFAZ INICIAR SESIÓN:

The screenshot shows a login form titled "Iniciar sesión". It contains two input fields: "Correo electrónico" with the placeholder "Ingrese su correo electrónico" and "Contraseña" with the placeholder "Ingrese su contraseña". Below these fields is a dark button labeled "Iniciar sesión". At the bottom, there is a link that says "¿No tienes cuenta? Registrarse".

FLUJO AL HACER CLIC EN INICIAR SESIÓN:

1. Se envía una solicitud HTTP POST al backend con los datos ingresados:


```
{
    "email": "martina@gmail.com",
    "password": "123456"
}
```
2. El backend realiza una consulta en la base de datos para verificar si el correo electrónico existe y si la contraseña ingresada coincide con la almacenada:



```
SELECT id_cliente, nombre_cliente, apellido_cliente, email, password
FROM clientes WHERE email = 'martina@gmail.com';
```
3. Si los datos son correctos, el backend responde con un mensaje de éxito y devuelve el ID del cliente para futuras referencias en la sesión:


```
{
    "mensaje": "¡Bienvenido/a nuevamente!",
    "id_cliente": 1
}
```
4. Si los datos no son correctos, el backend responde con un mensaje de error:


```
{
    "mensaje": "Correo electrónico o contraseña incorrectos. Intente nuevamente."
}
```


INTERFAZ LISTADO DE MATERIALES:

Listado de Materiales




Ladrillo Hueco
Ladrillo hueco cerámico
12x18x33 cm 9 tubos

\$390,00



Viga
Viga de hormigón pretensado
de 4 metros de largo

\$10619,00



Cemento Portland
Bolsa de cemento Portland de
50 kg

\$16500,00

FLUJO PARA BUSCAR/VER LA LISTA DE MATERIALES:

- Al cargar la página, se envía una solicitud HTTP GET al backend para obtener la lista de materiales disponibles:

GET /api/materiales

Lo mismo si hacemos clic en el botón **Buscar**:

GET /api/materiales?busqueda=ladrillo

- El backend realiza una consulta en la base de datos para recuperar todos los materiales:

*SELECT id_material, img_material, nombre_material, descripcion, precio
FROM materiales;*

O bien, para recuperar los materiales que coincidan con la búsqueda:

*SELECT id_material, img_material, nombre_material, descripcion, precio
FROM materiales WHERE nombre_material LIKE '%ladrillo%';*

- El backend devuelve la lista de materiales en formato JSON:

```
[
  {
    "id_material": 1,
    "img_material": "ladrillo.png",
    "nombre_material": "Ladrillo Hueco",
    "descripcion": "Ladrillo hueco cerámico 12x18x33 cm 9 tubos",
    "precio": 390.00
  },
  {
    "id_material": 2,
```



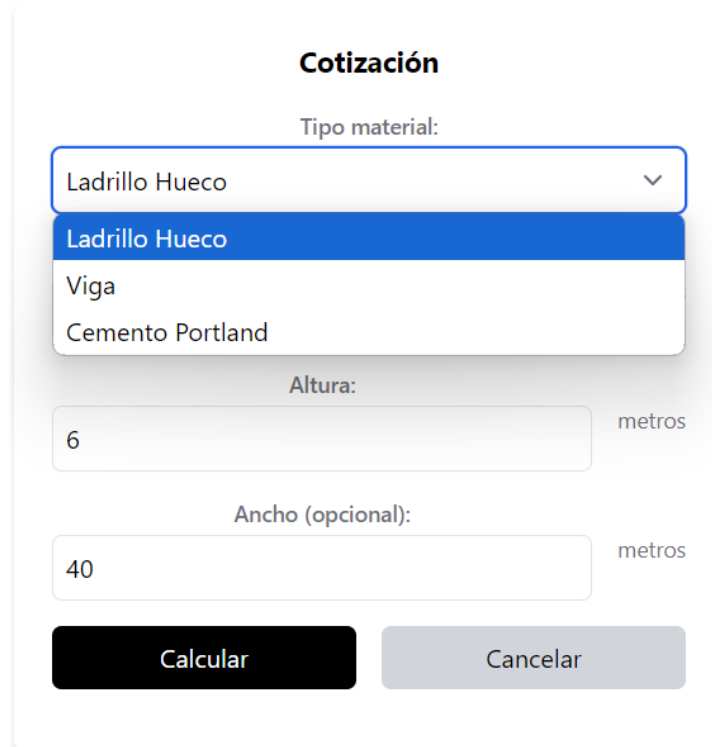
```

    "img_material": "viga.png",
    "nombre_material": "Viga",
    "descripcion": "Viga de hormigón pretensado de 4 metros de largo",
    "precio": 10619.00
  }
  {
    "id_material": 3,
    "img_material": "cemento.png",
    "nombre_material": "Cemento Portland",
    "descripcion": "Bolsa de cemento Portland de 50 kg",
    "precio": 16500.00
  }
]

```

FLUJO AL HACER CLIC EN COTIZAR:

1. El usuario es redirigido a una nueva pantalla de **Cotización**:



2. El frontend realiza una solicitud HTTP GET para obtener los detalles del material seleccionado y llenar el dropdown de Tipo material con las demás opciones disponibles:

GET /api/materiales

3. El backend consulta en la base de datos para obtener los detalles de todos los materiales disponibles para cotización:
SELECT id_material, nombre_material FROM materiales;
4. El backend devuelve una lista de los materiales en formato JSON, que se utilizarán para llenar el dropdown:

```
[
  { "id_material": 1, "nombre_material": "Ladrillo Hueco" },
  { "id_material": 2, "nombre_material": "Viga" },
  { "id_material": 3, "nombre_material": "Cemento Portland" }
]
```

INTERFAZ COTIZACIÓN:

Cotización

Tipo material:

Ladrillo Hueco
▼

Largo:

40

metros

Altura:

6

metros

Ancho (opcional):

40

metros

Calcular

Cancelar

FLUJO AL HACER CLIC EN CALCULAR:

1. Se envía una solicitud HTTP POST al backend con los detalles del material seleccionado y las dimensiones ingresadas:

```
{
  "id_material": 1,
  "largo": 40,
  "altura": 6,
  "ancho": 40
}
```

2. El backend utiliza el ID del material para obtener su precio y calcular el costo basado en las dimensiones proporcionadas:
SELECT precio FROM materiales WHERE id_material = 1;
3. El backend realiza los cálculos necesarios para determinar la cantidad total de material y el costo basado en las dimensiones y lo devuelve al frontend en una nueva pantalla de **Resultado**:

Resultado

Ladrillo Hueco

Cantidad: 16161 unidades / 112,23 pallet

Precio unitario: \$390,00

Subtotal: \$6.302.790

Descuento (5%): -\$315.139,50

Total: \$5.987.650,50

Confirmar como pedido

Volver a productos

```
{
  "nombre_material": "Ladrillo Hueco",
  "cantUnidades": 16161,
  "cantPallet": 112.23,
  "precioUnitario": 390.00,
  "subtotal": 6302790.00,
  "descuento": 315319.50,
  "precioTotal": 5987650.50
}
```

FLUJO AL HACER CLIC EN CONFIRMAR COMO PEDIDO:

1. Se envía una nueva solicitud HTTP POST al backend para crear un pedido en la base de datos:

```
{
  "id_cliente": 1,
  "id_material": 1,
  "nombre_material": "Ladrillo Hueco",
  "cantUnidades": 16161,
  "cantPallet": 112.23,
  "precioUnitario": 390.00,
  "subtotal": 6302790.00,
  "descuento": 315319.50,
  "precioTotal": 5987650.50
}
```

2. El backend crea un nuevo registro en la base de datos para almacenar el pedido:

```
INSERT INTO pedidos (id_cliente, id_material, nombre_material,
cantUnidades, cantPallet, precioUnitario, subtotal, descuento,
precioTotal, estado) VALUES (1, 1, 'Ladrillo Hueco', 16161, 112.23,
390.00, 6302790.00, 315319.50, 5987650.50, 'Pendiente');
```

- Después de crear el pedido, el backend responde con una confirmación de que el pedido ha sido registrado correctamente:

```
{
  "mensaje": "Su pedido ha sido registrado correctamente.",
  "id_pedido": 123
}
```

En caso de que el usuario haga clic en **Volver a productos**, se redirige al listado de materiales sin hacer ninguna solicitud al backend.

INTERFAZ RESUMEN DEL PEDIDO:

Resumen del pedido #123

Ladrillo Hueco

Cantidad: 16161 unidades / 112,23 pallet

Precio unitario: \$390,00

Subtotal: \$6.302.790

Descuento (5%): -\$315.139,50

Total a pagar: \$5.987.650,50

Continuar con el pago

Cancelar pedido

FLUJO PARA VER EL RESUMEN:

- El frontend realiza una solicitud HTTP GET al backend para obtener los detalles del pedido almacenado en la base de datos:

GET /api/pedido/123

- El backend consulta la base de datos para obtener los detalles del pedido:

SELECT nombre_material, cantUnidades, cantPallet, precioUnitario, subtotal, descuento, precioTotal, estado FROM pedidos WHERE id_pedido = 123;

- El backend devuelve los detalles del pedido en formato JSON para que el frontend los muestre en la pantalla:

```
{
  "id_pedido": 123,
  "nombre_material": "Ladrillo Hueco",
  "cantUnidades": 16161,
  "cantPallet": 112.23,
  "precioUnitario": 390.00,
  "subtotal": 6302790.00,
  "descuento": 315139.50,
  "precioTotal": 5987650.50
}
```

INTERFAZ PAGO:

Pago

Datos de la tarjeta:

Número de tarjeta

Titular de tarjeta

Vencimiento (MM/AA)

Código de seguridad

Confirmar pago

Cancelar

FLUJO PARA CONFIRMAR EL PAGO:

1. Se envía una solicitud HTTP POST al backend con los datos de la tarjeta:

```
{  "id_pedido": 123,  "id_cliente": 1,  "numeroTarjeta": "4123 4567 8901 2345",  "titularTarjeta": "Martina Cleri",  "fechaVencimiento": "11/26",  "codigoSeguridad": 456}
```
2. El backend procede a validar los datos de la tarjeta:
 - Si el número de la tarjeta es correcto.
 - Que la fecha de vencimiento no haya pasado.
 - Verificar el código de seguridad (CVV).Si la validación falla, el backend devuelve un error.
3. Después de la validación, el backend integra una pasarela de pago (como Mercado Pago) para procesar el pago:

```
const procesarPago = async (datosTarjeta) => {  // Lógica para conectar con la pasajera de pago  return true; // Simular que el pago fue exitoso};
```
4. Se registran los datos en la tabla "pagos":

```
INSERT INTO pagos (id_pedido, id_cliente, monto, estado) VALUES (123, 1, 5987650.50, 'Confirmado');
```

5. El backend también actualiza el estado del pedido en la base de datos:
UPDATE pedidos SET estado = 'Pagado' WHERE id_pedido = 123;
6. El backend responde confirmando que el pago fue procesado correctamente:

```
{
  "mensaje": "Su pago ha sido registrado correctamente.",
  "id_pedido": 123
}
```

INTERFAZ ENVÍO:

Envío

Datos de envío:

Confirmar envío

Cancelar

FLUJO PARA CONFIRMAR ENVÍO:

1. Se envía una solicitud HTTP POST al backend con los datos del envío:


```
{
    "id_pedido": 123,
    "id_cliente": 1,
    "nombreApellido": "Martina Cleri",
    "codigoPostal": 2400,
    "direccionEnvio": "Calle Falsa 1234",
    "telefono": "123456789"
  }
```
2. El backend valida los datos de envío:
 - Que el código postal sea válido.
 - Que el número de teléfono tenga el formato correcto.

Si falta algún dato o la validación falla, el backend devolverá un error.
3. El backend registra los datos en la tabla "envíos":
INSERT INTO envios (id_pedido, id_cliente, nombreApellido, codigoPostal, direccionEnvio, telefono, estado) VALUES (123, 1, 'Martina Cleri', 2400, 'Calle Falsa 1234', '123456789', 'Pendiente');

4. El backend devuelve una respuesta exitosa confirmando que el envío ha sido registrado correctamente:

```
{  
  "mensaje": "¡Muchas gracias por su compra! Pronto le  
    confirmaremos la fecha de envío.",  
  "id_envio": 789  
}
```

FLUJO PARA CANCELAR:

1. Se envía una solicitud HTTP PUT al backend para actualizar el estado del pedido:

```
{  
  "id_pedido": 123,  
  "estado": "Cancelado"  
}
```

2. El backend también actualiza el estado del pedido en la base de datos:

UPDATE pedidos SET estado = 'Cancelado' WHERE id_pedido = 123;

3. El backend confirma que el pedido ha sido cancelado:

```
{  
  "mensaje": "Su pedido ha sido cancelado.",  
  "id_pedido": 123  
}
```