

DISEÑO DE SISTEMAS

TP3

Requerimientos funcionales

1. Registro de clientes:

- El sistema debe permitir a los clientes registrarse e identificarse para cotizar, generar pedidos y acumular puntos por compra.

2. Consulta de productos:

- Los clientes deben poder consultar los productos disponibles, junto con sus descripciones, precios por unidad, cantidades por pallet, y el valor de puntos acumulables en cada compra.
- Filtros de búsqueda:
 1. Nombre: Los clientes pueden buscar productos ingresando una palabra clave o el nombre completo del producto.
 2. Hasta precio: Los clientes pueden establecer un valor máximo, y el sistema debe mostrar los productos cuyo precio sea igual o menor a dicho valor.

3. Cotización de productos:

- • El sistema debe permitir que los clientes ingresen los metros cuadrados a construir y los tipos de materiales necesarios. También debe contar con la opción de cargar un archivo en formato Autocad, el cual se procesará a través de una API para calcular automáticamente los metros cuadrados de pared y techo.
- • Cálculos:
 1. Calcular la cantidad de materiales necesarios en función de los metros cuadrados ingresados o los calculados por la API de Autocad.
 2. Determinar el importe total de la cotización, incluyendo el detalle de la cantidad de pallets necesarios y el costo total.

4. Descuentos por cantidad:

- El sistema debe gestionar descuentos según la cantidad de productos solicitados.

5. Sistema de puntos:

- Los clientes podrán acumular puntos por cada compra.
- Configuración: La fórmula de puntos ($n \text{ puntos} = x \text{ pesos}$) podrá ser modificada por un usuario administrador.

6. Pago a través de múltiples plataformas:

- El sistema debe permitir a los clientes realizar pagos a través de diferentes plataformas de pago (por ejemplo, MercadoPago, transferencia bancaria, etc.), sin necesidad de gestionar directamente los datos de tarjeta de crédito.

Requerimientos no funcionales

1. Rendimiento:

- El sistema debe procesar cotizaciones y generación de pedidos en menos de 3 segundos en condiciones normales de operación.
- Respuesta de carga de productos: La consulta de productos en la interfaz debe cargar en un tiempo máximo de 2 segundos por cada 100 productos.
- Procesamiento de puntos: El cálculo de puntos por compra debe ejecutarse en un tiempo máximo de 1 segundo después de que el pedido sea confirmado.

2. Escalabilidad:

- El sistema debe soportar hasta 5,000 usuarios concurrentes sin degradar su rendimiento, con una capacidad de crecimiento a 10,000 usuarios mediante ajuste de infraestructura en picos de demanda.
- API de Autocad: La integración con la API externa debe procesar y devolver resultados en un tiempo máximo de 10 segundos para planos de hasta 50 MB.

3. Disponibilidad:

- El sistema debe estar disponible 99.9% del tiempo al año, permitiendo consultas y pedidos 24/7, con un tiempo máximo de inactividad permitido de 8 horas al año

4. Seguridad:

- Cifrado de datos: Todos los datos de clientes y transacciones deben estar cifrados con un algoritmo mínimo de AES-256.
- Tiempo de expiración de sesión: Las sesiones de usuario deben expirar automáticamente después de 15 minutos de inactividad para reducir riesgos de acceso no autorizado.
- Pagos: La integración con plataformas de pago como MercadoPago debe cumplir con la certificación PCI DSS nivel 1 para garantizar la seguridad de las transacciones.

5. Mantenibilidad:

- Las actualizaciones del sistema deben realizarse sin interrumpir el servicio al usuario. Cada nueva versión debe estar instalada y funcional en un tiempo máximo de 2 horas durante horarios de baja demanda.
- Documentación: Cada componente del sistema debe contar con al menos un 90% de cobertura en la documentación, y el código crítico debe tener al menos un 85% de cobertura de pruebas automáticas.

6. Usabilidad:

- Tiempo de aprendizaje: Un usuario promedio sin conocimientos técnicos debe poder realizar una cotización y pedido en menos de 5 minutos.
- Navegación intuitiva: Las operaciones principales (registro, consulta de productos, cotización, pedido) deben ser accesibles en no más de 5 clics desde la página o menú principales.

7. Compatibilidad con diferentes dispositivos:

- El sistema debe ser responsive y funcionar de forma óptima en dispositivos con resoluciones entre 320x568 píxeles (smartphones) y 1920x1080 píxeles (pantallas de escritorio).
- Compatibilidad de versiones: La aplicación debe ser compatible con versiones de Android a partir de la 7.0 y iOS a partir de la 13.0.

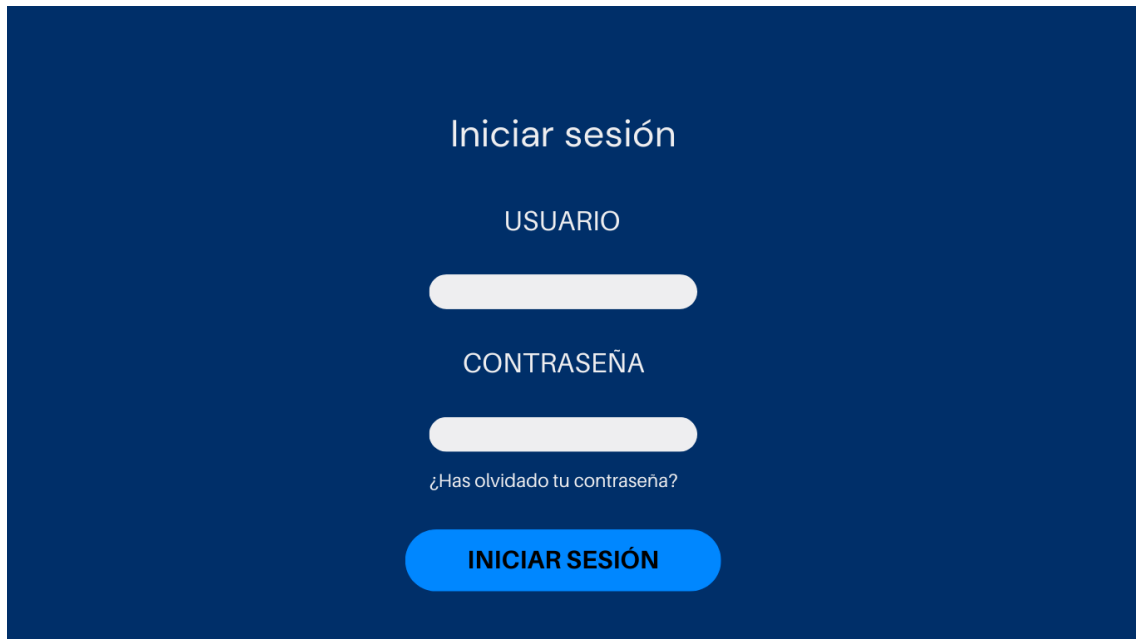
Iniciar sesión:

Campo	Front-end	Back-end	Base de Datos
usuario	string	string	varchar
contraseña	string	string	varchar

Descripción de los campos:

usuario: Nombre de usuario para autenticación.

contraseña: Contraseña asociada al usuario para completar la autenticación.



Front-end (JavaScript):

```
let credenciales = {  
  usuario: "umagario",      // string  
  password: "dsi.2024"     // string  
};
```

Back-end (Node.js):

```
const { usuario, password } = req.body; // string, string
```

Base de datos (MySQL):

```
CREATE TABLE clientes (  
  id_cliente INT AUTO_INCREMENT PRIMARY KEY, -- int  
  usuario VARCHAR(50),                       -- string  
  password VARCHAR(255)                      -- string  
);
```

Productos disponibles:


Campo	Front-end	Back-end	Base de Datos
material	string	string	varchar
precio_unitario	number	number	decimal
descripcion	string	string	text
cant_pallet	number	number	int
puntos_acumulables	number	number	int
stock_disponible	number	number	Int
imagen_producto	string	string	varchar

Descripción de los campos:

material: Nombre o tipo de material.
precio_unitario: Precio por unidad.
descripcion: Descripcion del producto.
cant_pallet: Cantidad de unidades por pallet.
puntos_acumulables: Puntos que el cliente puede acumular por cada unidad comprada.
stock_disponible: Cantidad de unidades disponibles en stock.
imagen_producto: URL o referencia a la imagen del producto.

PRODUCTOS DISPONIBLES


FILTRAR



100 Puntos cada

\$10.000

Ladrillo hueco12x18x33 cm 9 tubos
Precio por unidad: \$390,00
Descripción: Ladrillo hueco cerámico
12x18x33 cm 9 tubos
Ladrillo de cerramiento
Cantidad por pallet: 144 unidades
Stock: 2.000 unidades disponibles



200 Puntos cada

\$10.000

Viga 4 mts
Precio por unidad: \$10619
Descripción: Ladrillo hueco cerámico
12x18x33 cm 9 tubos
Ladrillo de cerramiento
Stock: 500 unidades disponibles

COTIZAR

Front-end (JavaScript):

```
let producto = {  
  material: "Cemento",           // string  
  precio_unitario: 300,          // number  
  descripcion: "Cemento de alta calidad", // string  
  cant_pallet: 10,              // number  
  puntos_acumulables: 100,      // number  
  stock_disponible: 50,         // number  
  imagen_producto: "url-de-la-imagen" // string (URL de la imagen)  
};
```

Back-end (Node.js):

```
const { material,  
  precio_unitario,  
  descripcion,  
  cant_pallet,  
  puntos_acumulables,  
  stock_disponible,  
  imagen_producto } = req.body;
```

Base de datos (MySQL):

```
CREATE TABLE productos (  
  id_producto INT AUTO_INCREMENT PRIMARY KEY,  
  material VARCHAR(100),  
  precio_unitario DECIMAL(10, 2),  
  descripcion TEXT,  
  cantidad_pallet INT,  
  puntos_acumulables INT,  
  stock_disponible INT,  
  imagen_producto VARCHAR(255)  
);
```

Filtrar:

Campo	Front-end	Back-end	Base de Datos
nombre	string	string	varchar
hasta_precio	number	number	decimal

Descripción de los campos:

nombre: Palabra clave o nombre del producto para búsqueda.

hasta_precio: Precio máximo para filtrar productos dentro de ese rango.

Front-end (JavaScript):

```
let filtros = {  
  nombre: "Cemento", // string  
  hastaPrecio: 500 // number  
};
```

Back-end (Node.js):

```
const { nombre, hastaPrecio } = req.query; // string, number
```

Base de datos (MySQL):

```
SELECT * FROM productos  
WHERE material LIKE '%Cemento%'  
AND precio_unitario <= 500;
```


Cotizar:

Campo	Front-end	Back-end	Base de Datos
nombre	string	string	varchar
largo	number	number	decimal
ancho	number	number	decimal
alto	number	number	decimal
metros_cuadrados	number	number	decimal
cant_pallets	number	number	int
detalle	string	string	text
descuento	number	number	decimal
puntos_acumulables	number	number	int
importe	number	number	decimal
archivo_autocad	file	file	varchar
metros_pared	number	number	decimal
metros_techo	number	number	decimal

Descripción de los campos:

nombre: Nombre o identificador de la cotización.

largo: Longitud en metros de la construcción.

ancho: Ancho en metros de la construcción.

alto: Altura en metros de la construcción.

metros_cuadrados: Area total calculada en metros cuadrados.

cant_pallet: Número de pallets necesarios.

detalle: Descripción adicional de la cotización.

descuento: Descuento aplicado a la cotización.

puntos_acumulables: Puntos que el cliente acumula por la cotización.

importe: Total de la cotización después de descuentos.

archivo_autocad: Archivo Autocad cargado por el cliente para calcular los metros cuadrados de la construcción.

metros_pared: Metros cuadrados de pared calculados a partir del archivo Autocad.

metros_techo: Metros cuadrados de techo calculados a partir del archivo Autocad.

COTIZACIÓN

NOMBRE: xxxxxxxxxxxxxxxxxxxx

LARGO:

xxxx

ANCHO:

xxxx

ALTO:

xxxx

ARCHIVO AUTOCAD

PARED:

xxxx

TECHO:

xxxx

ÁREA TOTAL:

xxxxxxx

CANT. PALLET:

xxxxxxx

DESCUENTO:

\$\$\$\$\$\$\$\$\$

IMPORTE:

\$\$\$\$\$\$\$\$\$

PUNTOS A ACUMULAR:

xxxx

Detalle: xxxxxxxxxxxx

COMPRAR

Front-end (JavaScript):

```
let cotizacion = {  
  nombre: "Cemento", // string  
  largo: 10, // number  
  ancho: 5, // number  
  alto: 3, // number  
  metros_cuadrados: 50, // number (ejemplo de cálculo: largo * ancho)  
  cant_pallet: 10, // number  
  detalle: "Cotización de cemento para proyecto", // string  
  descuento: 10.00, // number  
  puntos_acumulables: 100, // number  
  importe: 2000.00, // number  
  archivo_autocad: "url-del-archivo-autocad", // string (URL o nombre del archivo)  
  metros_pared: 40, // number (ejemplo)  
  metros_techo: 20 // number (ejemplo)  
};
```

Back-end (Node.js):

```
const {  
  nombre,  
  largo,  
  ancho,  
  alto,  
  metros_cuadrados,  
  cant_pallet,  
  detalle,  
  descuento,  
  puntos_acumulables,  
  importe,  
  archivo_autocad,  
  metros_pared,  
  metros_techo  
} = req.body;
```

Base de datos (MySQL):

```
CREATE TABLE cotizaciones (  
  id_cotizacion INT AUTO_INCREMENT PRIMARY KEY,  
  nombre VARCHAR(100),  
  largo DECIMAL(10, 2),  
  ancho DECIMAL(10, 2),  
  alto DECIMAL(10, 2),  
  metros_cuadrados DECIMAL(10, 2),  
  cant_pallet INT,  
  detalle TEXT,  
  descuento DECIMAL(5, 2),  
  puntos_acumulables INT,  
  importe DECIMAL(10, 2),  
  archivo_autocad VARCHAR(255),  
  metros_pared DECIMAL(10, 2),  
  metros_techo DECIMAL(10, 2)  
);
```

Pago:

Campo	Front-end	Back-end	Base de Datos
monto_total	number	Number	decimal
metodo_pago	string	string	varchar
estado_pago	string	string	varchar
transaccion_id	string	string	varchar
fecha_pago	string	string	varchar

Descripción de los campos:

monto_total: Monto total de la compra.
metodo_pago: Método de pago utilizado (Ej: “MercadoPago”, “Modo”,etc).
estado_pago: Estado del pago (Ej: “Pendiente”, “Aprobado”, “Fallido”).
transaccion_id: Identificador único de la transacción generado por la plataforma de pago.
fecha_pago: Fecha en la que se realizó el pago.

PAGO

MONTO A PAGAR: \$XXXXXXXXXXXXX

MÉTODO DE PAGO:

PAGAR

PAGO

ID TRANSACCIÓN: XXXX

ESTADO DE PAGO: APROBADO

FECHA DE PAGO: xx/xx/xxxx

Front-end (JavaScript):

```
let pago = {  
  monto_total: 1500.00,           // number  
  metodo_pago: "Modo",           // string  
  estado_pago: "Completado",     // string  
  transaccion_id: "TX123456789", // string  
  fecha_pago: "2024-11-12"      // string  
};
```

Back-end (Node.js):

```
const {  
  monto_total,  
  metodo_pago,  
  estado_pago,  
  transaccion_id,  
  fecha_pago  
} = req.body;
```

Base de datos (MySQL):

```
CREATE TABLE pagos (  
  id_pago INT AUTO_INCREMENT PRIMARY KEY,  
  monto_total DECIMAL(10, 2),  
  metodo_pago VARCHAR(50),  
  estado_pago VARCHAR(50),  
  transaccion_id VARCHAR(100),  
  fecha_pago VARCHAR(20)  
);
```