Documentación Técnica

Fecha creación

11 Diciembre 2025

Responsable de desarrollo

Jefferson Pérez Bedoya

Responsable de documentación

Jefferson Pérez Bedoya

Sistema de Consulta de Clientes

Rios del Desierto SAS

Descripción General

Sistema web para consultar información de clientes y generar reportes de fidelización basado en el monto de compras del último mes.

Tecnologías Utilizadas:

- **Backend:** Python 3.14, Flask, SQLAlchemy
- **Base de Datos:** SQLite
- **Frontend:** HTML5, CSS3, JavaScript (Vanilla)
- **Librerías:** pandas, openpyxl, Flask-CORS

Arquitectura del Sistema

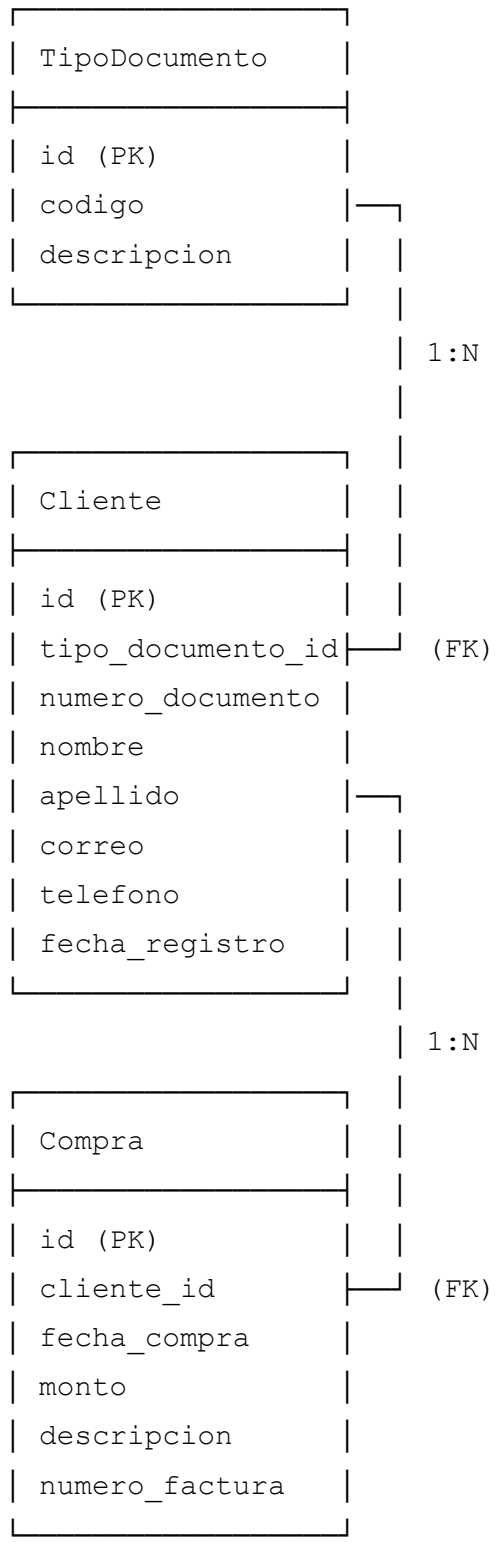
Patrón de Diseño

- **Arquitectura REST API**: Backend expone endpoints REST que el frontend consume
- **Separación de responsabilidades**: Backend (lógica) y Frontend (presentación) independientes
- **ORM Pattern**: SQLAlchemy como capa de abstracción de base de datos

Componentes Principales

Modelo de Datos

Diagrama Entidad-Relación



Tablas

1. TipoDocumento

Campo	Tipo	Descripción
id	Integer	Primary Key
codigo	String(10)	CC, NIT, PA
descripcion	String(50)	Descripción del tipo

Valores predefinidos:

- CC: Cédula de Ciudadanía
- NIT: NIT
- PA: Pasaporte

2. Cliente

Campo	Tipo	Descripción
id	Integer	Primary Key
tipo_documento_id	Integer	Foreign Key → TipoDocumento
numero_documento	String(20)	Número único (Index)
nombre	String(100)	Nombre del cliente
apellido	String(100)	Apellido del cliente
correo	String(150)	Email
telefono	String(20)	Teléfono de contacto
fecha_registro	DateTime	Fecha de registro

3. Compra

Campo	Tipo	Descripción
id	Integer	Primary Key
cliente_id	Integer	Foreign Key → Cliente
fecha_compra	DateTime	Fecha de la compra
monto	Float	Monto en COP
descripcion	String(200)	Descripción del producto
numero_factura	String(50)	Número de factura único

API Endpoints

Base URL

`http://127.0.0.1:5000/api`

1. Buscar Cliente

Endpoint: POST /api/buscar-cliente

Request Body:

```
{   "tipo_documento": "CC",   "numero_documento": "1234567890" }
```

Response (200 OK):

```
{   "cliente": {     "id": 1,     "tipo_documento": "Cédula de Ciudadanía",     "numero_documento": "1234567890",     "nombre": "Juan",     "apellido": "Pérez García",     "correo": "juan.perez@email.com",     "telefono": "3101234567",     "fecha_registro": "2024-06-15"   },   "compras": [     {       "id": 1,       "fecha_compra": "2024-12-05 10:30:00",       "monto": 1200000.0,       "descripcion": "Laptop Dell",       "numero_factura": "FC-2024-1001"     }   ],   "total_compras": 8343483.46,   "numero_compras": 8 }
```

Response (404 Not Found):

```
{   "error": "Cliente no encontrado" }
```

2. Exportar Cliente

Endpoint: POST /api/exportar-cliente

Request Body:

```
{   "numero_documento": "1234567890",   "formato": "csv"  // o "excel" }
```

Response:

- Archivo CSV o Excel para descarga

3. Reporte de Fidelización

Endpoint: GET /api/reporte-fidelizacion

Query Parameters: Ninguno

Response:

- Archivo Excel con clientes que superan 5,000,000 COP en el último mes

Response (404 Not Found):

```
{   "mensaje": "No hay clientes que superen los 5,000,000 COP en el último mes" }
```

4. Listar Clientes

Endpoint: GET /api/listar-clientes

Response (200 OK):


```
{   "total": 7,   "clientes": [     {       "tipo_documento":
"Cédula de Ciudadanía",       "codigo_tipo": "CC",
"numero_documento": "1234567890",       "nombre_completo": "Juan
Pérez García",       "correo": "juan.perez@email.com",
"telefono": "3101234567",       "total_ultimo_mes": 8343483.46,
"califica_fidelizacion": true     }   ] }
```


5. Tipos de Documento

Endpoint: GET /api/tipos-documento

Response (200 OK):


```
{   "tipos_documento": [     {"codigo": "CC", "descripcion":
"Cédula de Ciudadanía"},     {"codigo": "NIT", "descripcion":
"NIT"},     {"codigo": "PA", "descripcion": "Pasaporte"}   ] }
```


Seguridad

CORS (Cross-Origin Resource Sharing)

- Configurado para permitir peticiones desde cualquier origen en desarrollo
- En producción, especificar dominios permitidos en config.py

Validación de Datos

- Validación de campos requeridos en el backend
- Validación HTML5 en el frontend
- Sanitización de inputs para prevenir SQL Injection (SQLAlchemy ORM)

Manejo de Errores

- Try-catch en todos los endpoints
- Mensajes de error descriptivos
- Status codes HTTP apropiados

Módulos del Backend

app.py

Aplicación Flask principal. Contiene:

- Configuración de Flask y extensiones
- Definición de endpoints
- Lógica de negocio
- Inicialización de base de datos

Funciones principales:

- buscar_cliente(): Busca cliente y sus compras
- exportar_cliente(): Genera exportación CSV/Excel
- reporte_fidelizacion(): Genera reporte de clientes >5M COP
- listar_clientes(): Lista todos los clientes
- init_database(): Crea tablas y datos iniciales

models.py

Modelos de SQLAlchemy. Define:

- Clase TipoDocumento
- Clase Cliente
- Clase Compra
- Relaciones entre modelos
- Métodos to_dict() para serialización

config.py

Configuración de la aplicación:

- URI de base de datos
- Secret key
- Configuración CORS
- Carpeta de exportaciones

Módulos del Frontend

index.html

Estructura de la página:

- Sección de clientes disponibles
- Formulario de búsqueda
- Área de resultados
- Tablas de información
- Botones de exportación

style.css

Estilos visuales:

- Diseño responsive
- Grid layout para tarjetas
- Estilos de formularios
- Animaciones y transiciones
- Temas de color

script.js

Lógica del frontend:

- Comunicación con API (fetch)
- Manejo de eventos
- Renderizado dinámico
- Formateo de datos (moneda, fechas)
- Descarga de archivos

Funciones principales:

- cargarListaClientes(): Carga tarjetas de clientes
- buscarCliente(): Busca un cliente específico
- exportarCliente(): Descarga CSV/Excel
- generarReporteFidelizacion(): Descarga reporte
- formatearMoneda(), formatearFecha(): Utilidades

Flujo de Datos

1. Búsqueda de Cliente

```
Usuario ingresa documento     ↓ Frontend valida campos     ↓ POST
/api/buscar-cliente     ↓ Backend busca en BD     ↓ Retorna JSON
con datos     ↓ Frontend renderiza resultados
```

2. Exportación

```
Usuario click "Exportar"     ↓ POST /api/exportar-cliente     ↓
Backend genera archivo con pandas     ↓ Retorna archivo binario
↓ Frontend descarga automáticamente
```

3. Reporte de Fidelización

```
Usuario click "Reporte"     ↓ GET /api/reporte-fidelizacion     ↓
Backend filtra clientes >5M último mes     ↓ Genera Excel con
pandas + openpyxl     ↓ Retorna archivo Excel     ↓ Frontend
descarga automáticamente
```

Lógica de Negocio

Reglas de Fidelización

- **Período evaluado:** Últimos 30 días desde la fecha actual
- **Umbral:** $5,000,000 COP
- **Cálculo:** Suma de montos de todas las compras del cliente en el período
- **Resultado:** Solo clientes que superan el umbral aparecen en el reporte

Implementación:

```
fecha_limite = datetime.now() - timedelta(days=30)
compras_recientes = Compra.query.filter(     Compra.fecha_compra
>= fecha_limite ).all()
```

Testing

Casos de Prueba Sugeridos

1. **Buscar cliente existente** → Debe mostrar información completa
2. **Buscar cliente inexistente** → Debe mostrar error apropiado
3. **Exportar a CSV** → Debe descargar archivo válido
4. **Exportar a Excel** → Debe descargar archivo válido
5. **Reporte de fidelización** → Debe incluir solo clientes >5M
6. **Tipos de documento** → Debe listar CC, NIT, PA

7. **Lista de clientes** → Debe mostrar todos los clientes

Optimizaciones Implementadas

1. **Índices de Base de Datos:** Campo numero_documento indexado para búsquedas rápidas
2. **Lazy Loading:** Relaciones SQLAlchemy con lazy=True
3. **Serialización eficiente:** Métodos to_dict() optimizados
4. **Formateo en cliente:** JavaScript maneja formateo de datos
5. **Pandas para reportes:** Procesamiento eficiente de datos