T D10 Solutions - Arquitectura Completa

III ESTADO ACTUAL DEL PROYECTO

BACKEND COMPLETADO (Puerto 5001)

• Framework: Flask + SQLAlchemy

• Base de datos: SQLite (site.db)

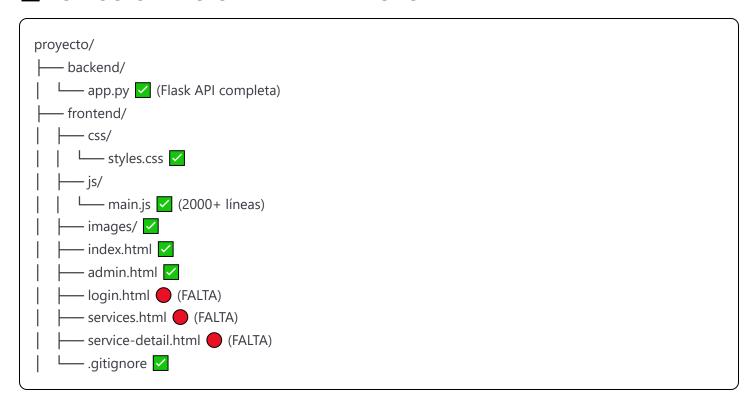
• CORS: Configurado para frontend

Endpoints Backend Funcionando:

- Auth: (/auth/login), (/auth/verify)
- Servicios: (/services) (GET, POST, PUT, DELETE)
- Usuarios: /users (GET, POST, PUT, DELETE)
- Health: (/health)

6 FRONTEND ACTUAL - ANÁLISIS DE DEPENDENCIAS

ESTRUCTURA ACTUAL Y DEPENDENCIAS



MAPA DE DEPENDENCIAS

Styles.css (Base de todo)

Depende de:

- Fuentes: Google Fonts (Inter)
- Reset CSS nativo

Es requerido por:

- **v** index.html
- **v** admin.html
- login.html
- services.html
- service-detail.html

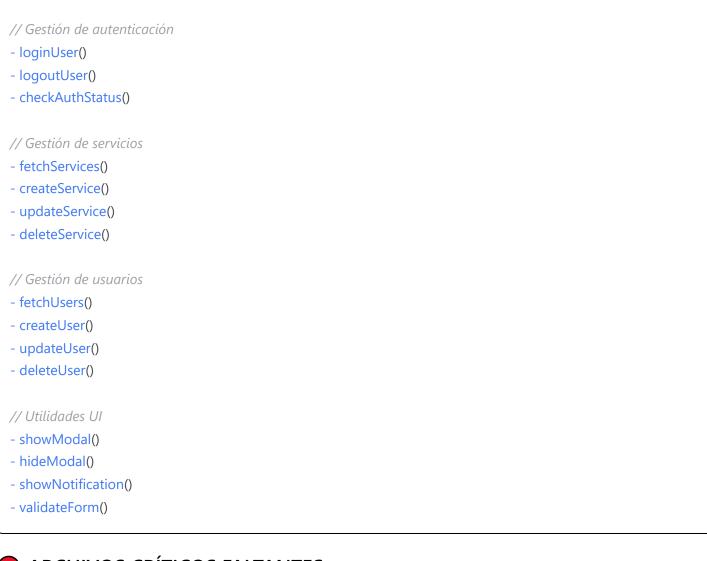
main.js (Motor principal)

Depende de:

- Backend Flask API (puerto 5001)
- DOM completamente cargado
- Elementos HTML específicos

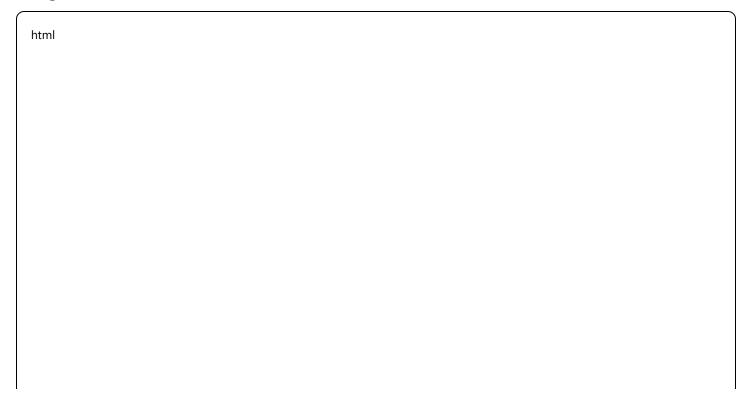
Funcionalidades que debe incluir:

javascript		



ARCHIVOS CRÍTICOS FALTANTES

1. login.html (CRÍTICO)



```
<!-- Estructura necesaria -->
<!DOCTYPE html>
<html lang="es">
<head>
          <meta charset="UTF-8">
          <title>Iniciar Sesión | D10 Solutions</title>
          k href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="style="base" rel=
          k rel="stylesheet" href="css/styles.css">
</head>
<body>
          <div class="login-container">
                    <div class="login-form">
                              <form id="loginForm">
                                         <input type="email" id="email" required>
                                         <input type="password" id="password" required>
                                         <button type="submit">Iniciar Sesión</button>
                              </form>
                    </div>
          </div>
          <script src="js/main.js"> </script>
</body>
</html>
```

Depende de:

- styles.css (estilos de formulario)
- main.js (función loginUser)
- Backend (/auth/login)

Es requerido por:

- index.html (botón "Iniciar Sesión")
- admin.html (redirección si no autenticado)

2. services.html (IMPORTANTE)

```
html

<!-- Lista completa de servicios -->

<div class="services-grid" id="servicesGrid">

<!-- Contenido dinámico desde main.js -->

</div>
```

Depende de:

- styles.css (grid y cards)
- main.js (fetchServices, renderServices)
- Backend (/services)

Es requerido por:

- index.html (enlace "Servicios")
- service-detail.html (navegación)

3. service-detail.html (IMPORTANTE)

```
html
<!-- Detalle individual de servicio -->
<div class="service-detail" id="serviceDetail">
<!-- Contenido dinámico -->
</div>
```

Depende de:

- styles.css (layout detalle)
- main.js (fetchService, renderServiceDetail)
- Backend (/services/{id})
- URL parameters

FUNCIONALIDADES REQUERIDAS EN main.js



javascript

```
// En login.html
document.getElementById('loginForm').addEventListener('submit', async (e) => {
  e.preventDefault();
  const email = document.getElementById('email').value;
  const password = document.getElementById('password').value;
  await loginUser(email, password);
});
async function loginUser(email, password) {
  try {
     const response = await fetch('http://127.0.0.1:5001/auth/login', {
       method: 'POST',
       headers: { 'Content-Type': 'application/json' },
       body: JSON.stringify({ email, password })
     });
     if (response.ok) {
       const data = await response.json();
       localStorage.setItem('token', data.token);
       window.location.href = 'admin.html';
     }
  } catch (error) {
     showNotification('Error al iniciar sesión', 'error');
  }
}
```

III Gestión de Servicios

javascript

```
// En admin.html
async function fetchServices() {
  try {
     const response = await fetch('http://127.0.0.1:5001/services');
     const data = await response.json();
     renderServicesTable(data.services);
  } catch (error) {
     console.error('Error:', error);
// En services.html
async function renderServicesGrid() {
  const services = await fetchServices();
  const grid = document.getElementById('servicesGrid');
  grid.innerHTML = services.map(service => `
     <div class="service-card">
        <h3>${service.name}</h3>
        ${service.description}
        <div class="service-price">$${service.price}/mes</div>
        <a href="service-detail.html?id=${service.id}">Ver más</a>
     </div>
  `).join('');
```

Z PLAN DE IMPLEMENTACIÓN

FASE 1: Completar Páginas Críticas (INMEDIATO)

- 1. **login.html** → Formulario de autenticación
- 2. **services.html** → Lista de servicios
- 3. service-detail.html → Detalle individual

FASE 2: Integración JavaScript (CRÍTICO)

- 4. Actualizar main.js con funciones faltantes:
 - Autenticación completa
 - Gestión de servicios frontend
 - Navegación entre páginas
 - Validaciones de formulario

FASE 3: Estilos y UX (IMPORTANTE)

- 5. Actualizar **styles.css** con:
 - Estilos para login.html
 - Grid responsivo para services.html
 - Layout para service-detail.html
 - Estados de loading y errores

FASE 4: Funcionalidades Avanzadas (OPCIONAL)

- 6. Sistema de notificaciones
- 7. Validación de tokens
- 8. Gestión de estados de carga
- 9. Manejo de errores mejorado

\triangle

DEPENDENCIAS CRÍTICAS IDENTIFICADAS

Flujo de Navegación

```
index.html → login.html → admin.html

↓ ↓

services.html → service-detail.html
```

Dependencias de Archivos

main.js DEPENDE DE:
styles.css DEPENDE DE: —— Google Fonts (Inter) —— Variables CSS consistentes
Cada HTML DEPENDE DE:

O PRÓXIMOS PASOS RECOMENDADOS

- 1. CREAR login.html con formulario funcional
- 2. CREAR services.html con grid dinámico
- 3. CREAR service-detail.html con vista detalle
- 4. ACTUALIZAR main.js con funciones de integración
- 5. PROBAR flujo completo de navegación

MEJORES PRÁCTICAS APLICADAS

JavaScript

- Funciones async/await para APIs
- Manejo de errores con try/catch
- Z Event listeners específicos
- Z Separación de responsabilidades

CSS

- Variables CSS para consistencia
- Grid y Flexbox para layouts
- Responsive design
- **Section** Estados hover y focus

HTML

- Semántica correcta
- Accesibilidad (labels, alt text)
- Meta tags apropiados
- Z Estructura consistente

Seguridad

- Validación en frontend y backend
- Tokens de autenticación
- Headers CORS configurados
- Sanitización de inputs

¿Quieres que implemente alguno de estos archivos faltantes como siguiente paso?