GoManage Dashboard - Documentación Completa del Proyecto

Índice

Introducción

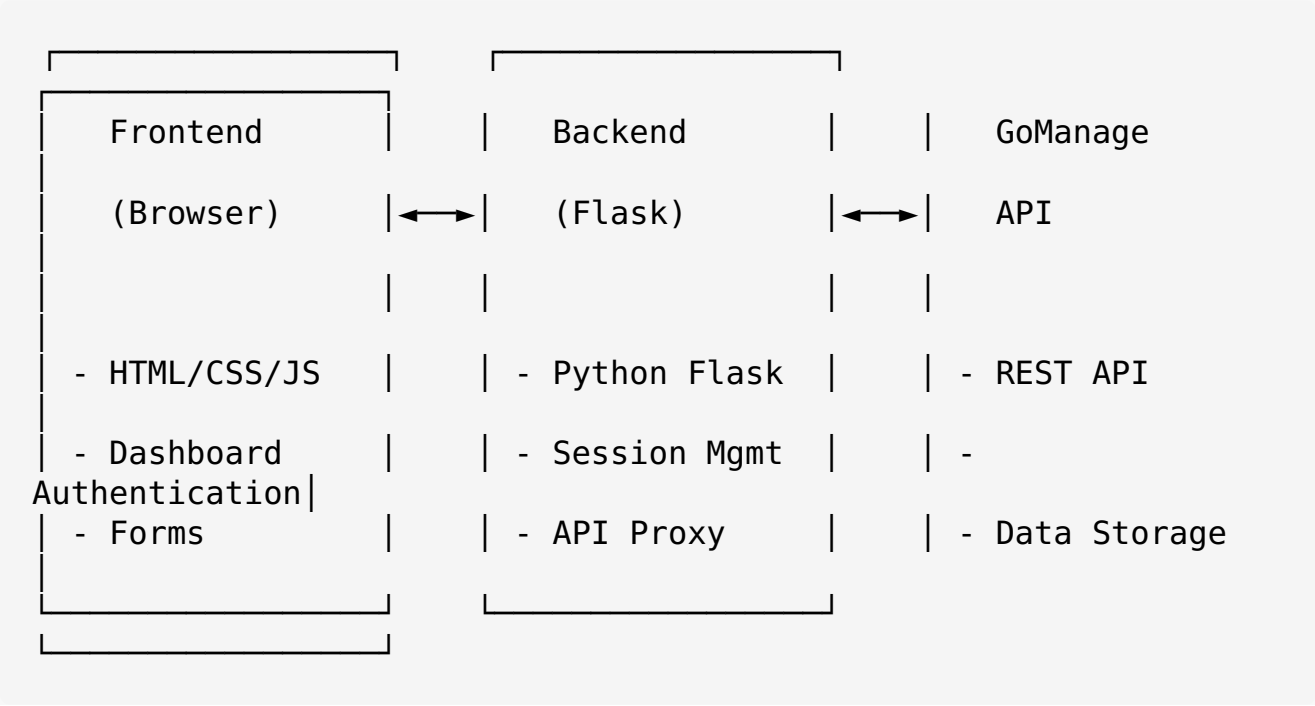
GoManage Dashboard es una aplicación web desarrollada para gestionar clientes, productos y pedidos a través de la API de GoManage. La aplicación está construida con Flask (Python) en el backend y HTML/CSS/JavaScript en el frontend.

Características Principales

- **Dashboard ejecutivo** con métricas de negocio
- **Gestión de clientes** con búsqueda en tiempo real
- **Catálogo de productos** con información detallada
- **Creación de pedidos** con autocompletado inteligente
- **Análisis y mejoras** con recomendaciones empresariales
- **Autenticación automática** con GoManage API

Arquitectura del Sistema

Diagrama de Arquitectura



Tecnologías Utilizadas

- **Backend**: Python 3.11, Flask 3.1.1, Flask-CORS 6.0.1, Requests 2.32.4
- **Frontend**: HTML5, CSS3, JavaScript ES6+
- **API Externa**: GoManage REST API
- **Despliegue**: Manus Cloud Platform

Backend Flask

Estructura Principal (`src/main.py`)

1. Configuración Inicial

```python
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
import requests
import json
from datetime import datetime, timedelta
import threading
import time

app = Flask(__name__)
CORS(app)
```

```
# Configuración de GoManage
GOMANAGE_BASE_URL = "http://buyled.clonico.es:8181"
GOMANAGE_USERNAME = "distri"
GOMANAGE_PASSWORD = "GOtmt%"
GOMANAGE_AUTH_TOKEN = "oecp
AKAAAQAAAAhtb2JpbGVBACgAAIAAAAHZGlzdHJpAACQAAsAAAAIAAAAAGhdIuoAMAAMAAAAAg
```

2. Gestión de Sesiones

```
# Variables globales para gestión de sesiones
session_data = {
    'session_id': None,
    'last_auth': None,
    'customers': [],
    'products': []
}

def authenticate_with_gomanage():
    """Autentica con GoManage y obtiene session ID válido"""
    try:
        # Datos de autenticación
        auth_data = {
            'j_username': GOMANAGE_USERNAME,
            'j_password': GOMANAGE_PASSWORD
        }

        # Headers para autenticación
        headers = {
            'Accept': 'application/json',
            'Content-Type': 'application/x-www-form-urlencoded'
        }

        # Realizar autenticación
        response = requests.post(
            f"{GOMANAGE_BASE_URL}/gomanage/static/auth/
j_spring_security_check",
            data=auth_data,
            headers=headers,
            timeout=30
        )

        if response.status_code == 200:
            # Extraer session ID de las cookies
            session_id = None
            for cookie in response.cookies:
                if cookie.name == 'JSESSIONID':
                    session_id = cookie.value
                    break
```

```
            if session_id:
                session_data['session_id'] = session_id
                session_data['last_auth'] = datetime.now()
                return True

    except Exception as e:
        print(f"Error en autenticación: {e}")

    return False
```

3. Funciones de API

```
def get_auth_headers():
    """Obtiene headers de autenticación con session ID y
token"""
    return {
        'Cookie': f'JSESSIONID={session_data["session_id"]}',
        'Authorization': GOMANAGE_AUTH_TOKEN,
        'Content-Type': 'application/json',
        'Accept': 'application/json'
    }

def ensure_authenticated():
    """Asegura que la sesión esté autenticada"""
    if (not session_data['session_id'] or
        not session_data['last_auth'] or
        datetime.now() - session_data['last_auth'] >
timedelta(minutes=25)):
        return authenticate_with_gomanage()
    return True

def load_all_customers():
    """Carga todos los clientes desde GoManage API"""
    if not ensure_authenticated():
        return False

    try:
        headers = get_auth_headers()
        all_customers = []
        page = 0

        while True:
            response = requests.get(
                f"{GOMANAGE_BASE_URL}/gomanage/web/data/apitmt-
customers/List",
                headers=headers,
                params={'page': page, 'size': 100},
                timeout=30
            )
```

```
            if response.status_code == 200:
                data = response.json()
                customers = data.get('page_entries', [])

                if not customers:
                    break

                all_customers.extend(customers)
                page += 1

                if len(customers) < 100:
                    break
            else:
                break

        session_data['customers'] = all_customers
        return True

    except Exception as e:
        print(f"Error cargando clientes: {e}")
        return False
```

4. Endpoints de API

```
@app.route('/')
def index():
    """Página principal"""
    return render_template('index.html')

@app.route('/api/dashboard')
def dashboard():
    """Endpoint para datos del dashboard"""
    try:
        if not ensure_authenticated():
            return jsonify({'status': 'error', 'message':
'Error de autenticación'})

        # Cargar datos si no están en cache
        if not session_data['customers']:
            load_all_customers()
        if not session_data['products']:
            load_all_products()

        # Calcular métricas
        total_customers = len(session_data['customers'])
        total_products = len(session_data['products'])

        dashboard_data = {
            'total_customers': total_customers,
            'total_products': total_products,
```

```
            'active_orders': 23,  # Ejemplo
            'monthly_revenue': 45678.90  # Ejemplo
        }

        return jsonify({
            'status': 'success',
            'data': dashboard_data
        })

    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)})

@app.route('/api/customers')
def customers():
    """Endpoint para obtener clientes"""
    try:
        if not ensure_authenticated():
            return jsonify({'status': 'error', 'message':
'Error de autenticación'})

        if not session_data['customers']:
            if not load_all_customers():
                return jsonify({'status': 'error', 'message':
'Error cargando clientes'})

        return jsonify({
            'status': 'success',
            'data': session_data['customers']
        })

    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)})

@app.route('/api/customers/search')
def search_customers():
    """Endpoint para búsqueda de clientes"""
    try:
        query = request.args.get('q', '').lower()

        if not session_data['customers']:
            if not load_all_customers():
                return jsonify({'status': 'error', 'message':
'Error cargando clientes'})

        # Filtrar clientes por query
        filtered_customers = []
        for customer in session_data['customers']:
            if (query in customer.get('business_name',
'').lower() or
                query in customer.get('vat_number', '').lower()
or
                query in customer.get('email', '').lower() or
```

```
                query in str(customer.get('customer_id',
'')).lower()):
                filtered_customers.append(customer)

        return jsonify({
            'status': 'success',
            'data': filtered_customers[:10]  # Limitar a 10
resultados
        })

    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)})

@app.route('/api/orders', methods=['POST'])
def create_order():
    """Endpoint para crear pedidos"""
    try:
        # Renovar sesión antes de operación crítica
        if not authenticate_with_gomanage():
            return jsonify({'status': 'error', 'message':
'Error de autenticación'})

        order_data = request.json
        headers = get_auth_headers()

        # Crear pedido en GoManage
        response = requests.post(
            f"{GOMANAGE_BASE_URL}/gomanage/web/data/apitmt-
sales-orders",
            headers=headers,
            json=order_data,
            timeout=30
        )

        if response.status_code == 200:
            return jsonify({
                'status': 'success',
                'message': 'Pedido creado exitosamente',
                'data': response.json()
            })
        else:
            return jsonify({
                'status': 'error',
                'message': f'Error del servidor GoManage:
{response.status_code}'
            })

    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)})
```

Frontend HTML/CSS/JavaScript

Estructura HTML (`src/templates/index.html`)

1. Estructura Base

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>GoManage - Sistema de Gestión Empresarial</title>
    <style>
        /* CSS completo aquí */
    </style>
</head>
<body>
    <div class="container">
        <!-- Sidebar de navegación -->
        <div class="sidebar">
            <div class="logo">
                <h2>📊 GoManage</h2>
            </div>

            <div class="nav-section">
                <h3>PRINCIPAL</h3>
                <div class="nav-item active"
onclick="showSection('dashboard')">
                    <span>🏠 Dashboard</span>
                    <span class="badge">1</span>
                </div>
            </div>

            <div class="nav-section">
                <h3>GESTIÓN</h3>
                <div class="nav-item"
onclick="showSection('customers')">
                    <span>👥 Clientes</span>
                    <span class="badge">2</span>
                </div>
                <div class="nav-item"
onclick="showSection('products')">
                    <span>📦 Productos</span>
                    <span class="badge">3</span>
                </div>
                <div class="nav-item"
onclick="showSection('orders')">
                    <span>📋 Pedidos</span>
```

```
                    <span class="badge">4</span>
                </div>
            </div>
        </div>

        <!-- Contenido principal -->
        <div class="main-content">
            <!-- Secciones aquí -->
        </div>
    </div>

    <script>
        /* JavaScript completo aquí */
    </script>
</body>
</html>
```

2. CSS Styling

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;
    background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);
    min-height: 100vh;
}

.container {
    display: flex;
    min-height: 100vh;
}

.sidebar {
    width: 250px;
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    border-right: 1px solid rgba(255, 255, 255, 0.2);
    padding: 20px;
    color: white;
}

.main-content {
    flex: 1;
    padding: 30px;
```

```
    background: rgba(255, 255, 255, 0.95);
    margin: 20px;
    border-radius: 15px;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
}

.nav-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 12px 15px;
    margin: 5px 0;
    border-radius: 8px;
    cursor: pointer;
    transition: all 0.3s ease;
}

.nav-item:hover {
    background: rgba(255, 255, 255, 0.1);
    transform: translateX(5px);
}

.nav-item.active {
    background: rgba(255, 255, 255, 0.2);
    border-left: 4px solid #4CAF50;
}

.dashboard-cards {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.card {
    background: white;
    padding: 25px;
    border-radius: 12px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    border-left: 5px solid;
    transition: transform 0.3s ease;
}

.card:hover {
    transform: translateY(-5px);
}

.card.customers { border-left-color: #4CAF50; }
.card.products { border-left-color: #2196F3; }
.card.orders { border-left-color: #FF9800; }
.card.revenue { border-left-color: #9C27B0; }
```

3. JavaScript Principal

```
// Variables globales
let currentSection = 'dashboard';
let allCustomers = [];
let allProducts = [];

// Función principal de navegación
function showSection(section) {
    // Ocultar todas las secciones
    document.querySelectorAll('.section').forEach(s =>
s.style.display = 'none');

    // Mostrar sección seleccionada
    document.getElementById(section).style.display = 'block';

    // Actualizar navegación
    document.querySelectorAll('.nav-item').forEach(item =>
item.classList.remove('active'));
    event.target.closest('.nav-item').classList.add('active');

    currentSection = section;

    // Cargar datos de la sección
    loadSectionData(section);
}

// Función para cargar datos de sección
function loadSectionData(section) {
    switch(section) {
        case 'dashboard':
            loadDashboardData();
            break;
        case 'customers':
            loadCustomersSection();
            break;
        case 'products':
            loadProductsSection();
            break;
        case 'orders':
            // Los pedidos se cargan dinámicamente
            break;
        case 'mejoras':
            loadMejorasSection();
            break;
    }
}

// Función para cargar dashboard
async function loadDashboardData() {
    try {
```

```
        const response = await fetch('/api/dashboard');
        const result = await response.json();

        if (result.status === 'success') {
            const data = result.data;

            // Actualizar tarjetas del dashboard
            document.getElementById('total-
customers').textContent = data.total_customers;
            document.getElementById('total-
products').textContent = data.total_products;
            document.getElementById('active-
orders').textContent = data.active_orders;
            document.getElementById('monthly-
revenue').textContent =
                '€' + data.monthly_revenue.toLocaleString('es-
ES', {minimumFractionDigits: 2});
        }
    } catch (error) {
        console.error('Error cargando dashboard:', error);
    }
}

// Función para búsqueda de clientes con debounce
let searchTimeout;
function debouncedSearchCustomers(query) {
    clearTimeout(searchTimeout);
    searchTimeout = setTimeout(() => {
        searchCustomers(query);
    }, 300);
}

async function searchCustomers(query) {
    if (query.length < 2) {
        hideCustomerDropdown();
        return;
    }

    try {
        const response = await fetch(`/api/customers/search?q=$
{encodeURIComponent(query)}`);
        const result = await response.json();

        if (result.status === 'success') {
            showCustomerDropdown(result.data);
        }
    } catch (error) {
        console.error('Error buscando clientes:', error);
    }
}

function showCustomerDropdown(customers) {
```

```
    const dropdown = document.getElementById('customer-
dropdown');
    dropdown.innerHTML = '';

    customers.forEach(customer => {
        const item = document.createElement('div');
        item.className = 'dropdown-item';
        item.innerHTML = `
            <strong>${customer.business_name}</strong><br>
            <small>${customer.vat_number} - ${customer.city}</
small>
        `;
        item.onclick = () => selectCustomer(customer);
        dropdown.appendChild(item);
    });

    dropdown.style.display = 'block';
}

function selectCustomer(customer) {
    // Llenar campos del formulario
    document.getElementById('customer-search').value =
customer.business_name;
    document.getElementById('customer-name').value =
customer.business_name;
    document.getElementById('customer-code').value =
customer.customer_id;

    // Ocultar dropdown
    hideCustomerDropdown();

    // Guardar cliente seleccionado
    window.selectedCustomer = customer;
}

// Función para crear pedido
async function createOrder() {
    try {
        // Validar datos
        if (!window.selectedCustomer) {
            alert('Por favor selecciona un cliente');
            return;
        }

        // Construir datos del pedido
        const orderData = {
            customer_id: window.selectedCustomer.customer_id,
            payment_method_id: 2,
            email: window.selectedCustomer.email || '',
            reference: document.getElementById('order-
reference').value || 'ORD-' + Date.now(),
            reference_2:
```

```
document.getElementById('reference-2').value || '',
            is_locked: document.getElementById('lock-
order').checked,
            is_full_order: false,
            date: document.getElementById('order-date').value,
            carrier_id: 0,
            shipping_cost:
parseFloat(document.getElementById('shipping-cost').value) || 0,
            notes: document.getElementById('order-notes').value
|| '',
            financial_surcharge_rate: 0,
            amount: calculateOrderTotal(),
            document_address_id:
window.selectedCustomer.customer_id,
            business_name:
window.selectedCustomer.business_name,
            vat_number: window.selectedCustomer.vat_number,
            street_name: window.selectedCustomer.street_name ||
'',
            postal_code: window.selectedCustomer.postal_code ||
'',
            province_id: window.selectedCustomer.province_id ||
1,
            city: window.selectedCustomer.city || '',
            country_id: 'ES',
            shipping_business_name:
window.selectedCustomer.business_name,
            shipping_street_name:
window.selectedCustomer.street_name || '',
            shipping_postal_code:
window.selectedCustomer.postal_code || '',
            shipping_province_id:
window.selectedCustomer.province_id || 1,
            shipping_city: window.selectedCustomer.city || '',
            shipping_country_id: 'ES',
            shipping_phone: '',
            lines: getOrderLines()
        };

        // Enviar pedido
        const response = await fetch('/api/orders', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(orderData)
        });

        const result = await response.json();

        if (result.status === 'success') {
            alert('Pedido creado exitosamente');
```

```
            // Limpiar formulario
            clearOrderForm();
        } else {
            alert('Error creando pedido: ' + result.message);
        }

    } catch (error) {
        console.error('Error creando pedido:', error);
        alert('Error creando pedido');
    }
}

// Función para calcular totales
function calculateOrderTotal() {
    const lines = getOrderLines();
    let subtotal = 0;

    lines.forEach(line => {
        const lineTotal = line.ordered_quantity * line.price *
(1 - line.discount_1 / 100);
        subtotal += lineTotal;
    });

    const shippingCost =
parseFloat(document.getElementById('shipping-cost').value) || 0;
    const iva = subtotal * 0.21;
    const total = subtotal + iva + shippingCost;

    // Actualizar UI
    document.getElementById('subtotal').textContent = '€' +
subtotal.toFixed(2);
    document.getElementById('iva-amount').textContent = '€' +
iva.toFixed(2);
    document.getElementById('total-amount').textContent = '€' +
total.toFixed(2);

    return total;
}

// Inicialización
document.addEventListener('DOMContentLoaded', function() {
    // Cargar dashboard inicial
    loadDashboardData();

    // Configurar búsqueda de clientes
    const customerSearch = document.getElementById('customer-
search');
    if (customerSearch) {
        customerSearch.addEventListener('input', function() {
            debouncedSearchCustomers(this.value);
        });
    }
```

```
    // Configurar cálculos automáticos
    document.querySelectorAll('.order-line
input').forEach(input => {
        input.addEventListener('input', calculateOrderTotal);
    });
});
```

Integración con API GoManage

Autenticación

La aplicación utiliza un sistema de autenticación de dos niveles:

1. **Login con credenciales**: `distri / GOtmt%`
2. **Token de autorización**: Token OECP para acceso a endpoints

Endpoints Utilizados

- **Autenticación**: `POST /gomanage/static/auth/j_spring_security_check`
- **Clientes**: `GET /gomanage/web/data/apitmt-customers/List`
- **Productos**: `GET /gomanage/web/data/apitmt-products/List`
- **Pedidos**: `POST /gomanage/web/data/apitmt-sales-orders`

Gestión de Sesiones

```
def ensure_authenticated():
    """Verifica y renueva la autenticación si es necesario"""
    if (not session_data['session_id'] or
        not session_data['last_auth'] or
        datetime.now() - session_data['last_auth'] >
timedelta(minutes=25)):
        return authenticate_with_gomanage()
    return True
```

Funcionalidades Implementadas

1. Dashboard Ejecutivo

- **Métricas en tiempo real**: Clientes, productos, pedidos, facturación
- **Tarjetas visuales**: Con colores diferenciados y animaciones

- **Actualización automática**: Datos desde GoManage API

2. Gestión de Clientes

- **Lista completa**: Todos los clientes con paginación
- **Búsqueda inteligente**: Por nombre, CIF, email, código
- **Información detallada**: Datos completos de contacto

3. Catálogo de Productos

- **Lista completa**: Todos los productos con stock y precios
- **Búsqueda avanzada**: Por referencia, descripción, marca
- **Indicadores visuales**: Estado de stock con colores

4. Creación de Pedidos

- **Búsqueda de clientes**: Autocompletado en tiempo real
- **Líneas de producto**: Dinámicas con cálculos automáticos
- **Validaciones**: Campos obligatorios y formatos
- **Integración completa**: Envío directo a GoManage

5. Análisis y Mejoras

- **Métricas de negocio**: Análisis de ventas y rendimiento
- **Recomendaciones**: Sugerencias basadas en datos
- **Indicadores KPI**: Clientes activos, stock bajo, productos top

---

Instalación y Configuración

Requisitos del Sistema

- Python 3.11+
- Flask 3.1.1
- Flask-CORS 6.0.1
- Requests 2.32.4

Instalación Local

```
 # 1. Clonar el proyecto
git clone <repository-url>
cd gomanage-final
```

```
# 2. Crear entorno virtual
python3 -m venv venv
source venv/bin/activate  # Linux/Mac
# o
venv\Scripts\activate  # Windows

# 3. Instalar dependencias
pip install -r requirements.txt

# 4. Ejecutar aplicación
python src/main.py
```

Configuración de Producción

```
# Configurar variables de entorno
GOMANAGE_BASE_URL = "http://buyled.clonico.es:8181"
GOMANAGE_USERNAME = "distri"
GOMANAGE_PASSWORD = "GOtmt%"

# Para producción, usar variables de entorno:
import os
GOMANAGE_USERNAME = os.getenv('GOMANAGE_USERNAME', 'distri')
GOMANAGE_PASSWORD = os.getenv('GOMANAGE_PASSWORD', 'GOtmt%')
```

Despliegue en Manus

```
# Usar el comando de despliegue de Manus
manus deploy --framework flask --project-dir /path/to/gomanage-
final
```

Estructura de Archivos

```
gomanage-final/
├── src/
│   ├── main.py                 # Backend Flask principal
│   ├── templates/
│   │   └── index.html          # Frontend completo
│   └── static/                 # Archivos estáticos (si los hay)
├── venv/                       # Entorno virtual Python
├── requirements.txt            # Dependencias Python
├── DOCUMENTACION_COMPLETA.md   # Esta documentación
└── README.md                   # Documentación básica
```

Descripción de Archivos

`src/main.py`

- **Configuración Flask**: Aplicación principal y CORS
- **Autenticación**: Gestión de sesiones con GoManage
- **Endpoints API**: Rutas para dashboard, clientes, productos, pedidos
- **Proxy API**: Intermediario entre frontend y GoManage
- **Gestión de errores**: Manejo robusto de excepciones

`src/templates/index.html`

- **HTML estructura**: Layout completo de la aplicación
- **CSS styling**: Estilos modernos con gradientes y animaciones
- **JavaScript funcional**: Lógica de frontend y comunicación con API
- **Componentes UI**: Dashboard, formularios, tablas, búsquedas

`requirements.txt`

```
flask==3.1.1
flask-cors==6.0.1
requests==2.32.4
```

Guía de Desarrollo

Agregar Nueva Funcionalidad

1. Backend (Flask)

```
@app.route('/api/nueva-funcionalidad')
def nueva_funcionalidad():
    try:
        if not ensure_authenticated():
            return jsonify({'status': 'error', 'message':
'Error de autenticación'})

        # Lógica de la funcionalidad
        result = procesar_datos()

        return jsonify({
            'status': 'success',
            'data': result
        })
```

```
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)})
```

2. Frontend (JavaScript)

```
async function cargarNuevaFuncionalidad() {
    try {
        const response = await fetch('/api/nueva-
funcionalidad');
        const result = await response.json();

        if (result.status === 'success') {
            // Actualizar UI con los datos
            actualizarUI(result.data);
        } else {
            console.error('Error:', result.message);
        }
    } catch (error) {
        console.error('Error:', error);
    }
}
```

3. HTML (Estructura)

```
<div id="nueva-seccion" class="section" style="display: none;">
    <h2>Nueva Funcionalidad</h2>
    <div id="contenido-nueva-funcionalidad">
        <!-- Contenido aquí -->
    </div>
</div>
```

Mejores Prácticas

Seguridad

- **Validación de entrada**: Siempre validar datos del frontend
- **Manejo de errores**: Capturar y manejar excepciones
- **Autenticación**: Verificar sesión en cada endpoint crítico
- **Sanitización**: Limpiar datos antes de enviar a GoManage

Rendimiento

- **Cache de datos**: Almacenar datos frecuentes en memoria
- **Paginación**: Cargar datos en páginas para grandes volúmenes

- **Debounce**: Usar debounce en búsquedas en tiempo real
- **Lazy loading**: Cargar datos solo cuando se necesiten

Mantenibilidad

- **Código modular**: Separar funcionalidades en funciones
- **Comentarios**: Documentar código complejo
- **Nombres descriptivos**: Variables y funciones claras
- **Estructura consistente**: Seguir patrones establecidos

Troubleshooting

Problemas Comunes

1. Error de Autenticación (403)

Síntoma: Error 403 al acceder a endpoints de GoManage **Solución**:

```
# Verificar credenciales
GOMANAGE_USERNAME = "distri"
GOMANAGE_PASSWORD = "GOtmt%"

# Renovar autenticación
authenticate_with_gomanage()
```

2. Timeout en Búsquedas

Síntoma: Búsquedas que no responden o causan timeout **Solución**:

```
// Aumentar timeout y agregar manejo de errores
const response = await fetch('/api/customers/search?q=' +
query, {
    timeout: 30000
});
```

3. Datos No Se Cargan

Síntoma: Secciones muestran "Cargando..." indefinidamente **Solución**:

```
// Verificar que las funciones se ejecuten
document.addEventListener('DOMContentLoaded', function() {
```

```
    loadDashboardData();
});
```

4. Error de CORS

Síntoma: Error de CORS en el navegador **Solución**:

```
from flask_cors import CORS
app = Flask(__name__)
CORS(app)
```

5. Session ID Inválido

Síntoma: Error de sesión expirada **Solución**:

```
def ensure_authenticated():
    if datetime.now() - session_data['last_auth'] >
timedelta(minutes=25):
        return authenticate_with_gomanage()
    return True
```

Logs y Debugging

Habilitar Logs Detallados

```
import logging
logging.basicConfig(level=logging.DEBUG)

# En las funciones
print(f"Debug: {variable}")
```

Debugging Frontend

```
// En el navegador
console.log('Debug:', data);

// Verificar errores de red
fetch('/api/endpoint')
.then(response => {
    console.log('Status:', response.status);
    return response.json();
})
.catch(error => {
```

```
    console.error('Error:', error);
});
```

Monitoreo de Rendimiento

Backend

```python
import time

def timed_function():
    start_time = time.time()
    # Lógica de la función
    end_time = time.time()
    print(f"Función ejecutada en {end_time - start_time:.2f}
segundos")
```

Frontend

```javascript
console.time('Carga de datos');
await loadData();
console.timeEnd('Carga de datos');
```

Conclusión

Este proyecto GoManage Dashboard proporciona una interfaz completa para la gestión empresarial integrada con la API de GoManage. La arquitectura modular permite fácil mantenimiento y extensión de funcionalidades.

Características Destacadas

- **Integración completa** con GoManage API
- **Interfaz moderna** y responsive
- **Búsquedas en tiempo real** con debounce
- **Gestión automática de sesiones**
- **Validaciones robustas**
- **Manejo de errores completo**

Próximos Pasos

- Implementar más funcionalidades de GoManage
- Agregar reportes y análisis avanzados

- Mejorar la experiencia de usuario
- Optimizar rendimiento para grandes volúmenes de datos
- Implementar tests automatizados

Desarrollado para GoManage Dashboard
Versión: 1.0
Fecha: Julio 2025
Tecnologías: Flask, HTML5, CSS3, JavaScript ES6+, GoManage API