**📚 Sistema SSO - Documentación Completa**

**🎯 ¿Qué es Sistema SSO?**

**Single Sign-On (SSO) Centralizado** construido con Node.js, PostgreSQL, Redis y Docker que permite a los usuarios **autenticarse una sola vez** y acceder a **múltiples microservicios** sin necesidad de login adicional.

**🏆 Características Principales:**

* ✅ **Un solo login** para todo el ecosistema
* ✅ **JWT tokens** seguros con verificación centralizada
* ✅ **Permisos granulares** por usuario y rol
* ✅ **Arquitectura de microservicios** escalable
* ✅ **Frontend unificado** con dashboard administrativo
* ✅ **API Gateway** inteligente
* ✅ **Docker** para despliegue fácil

**🏗️ Arquitectura del Sistema**

┌─────────────────────────────────────────────────────────────────┐

│ 🌐 CAPA DE PRESENTACIÓN │

├─────────────────────────────────────────────────────────────────┤

│ Frontend/ Frontend/ Frontend/ │

│ index.html dashboard.html demo-complete.html │

│ (Login) (Admin Panel) (Demo Completo) │

└─────────────────────┬───────────────────────────────────────────┘

│ 🔑 JWT Token (localStorage)

▼

┌─────────────────────────────────────────────────────────────────┐

│ 🚪 CAPA DE GATEWAY │

├─────────────────────────────────────────────────────────────────┤

│ API Gateway (Puerto 3000) │

│ • Punto de entrada único │

│ • Enrutamiento inteligente │

│ • Rate limiting │

│ • CORS handling │

└─┬─────────┬─────────┬─────────┬─────────┬─────────┬─────────────┘

│ │ │ │ │ │

▼ ▼ ▼ ▼ ▼ ▼

┌─────────────────────────────────────────────────────────────────┐

│ 🔧 CAPA DE MICROSERVICIOS │

├─────────────────────────────────────────────────────────────────┤

│ Auth User Hello [Future [Future [...] │

│ Service Service Service Projects] Finance] │

│ :3001 :3002 :3010 :3020 :3021 │

│ │

│ • Autenticación • Gestión • Demo • Proyectos │

│ • Usuarios • Básica • Pruebas • Contratos │

│ • Permisos • Health • SSO • Finanzas │

│ • JWT • Check • Ejemplos • Reportes │

└─────────────────────┬───────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────────────────────┐

│ 💾 CAPA DE DATOS │

├─────────────────────────────────────────────────────────────────┤

│ PostgreSQL Redis │

│ (Base de Datos Principal) (Cache y Sesiones) │

│ │

│ • Usuarios y roles • Blacklist de tokens │

│ • Permisos granulares • Cache de verificaciones │

│ • Auditoría completa • Sesiones temporales │

│ • Microservicios registrados • Rate limiting │

└─────────────────────────────────────────────────────────────────┘

**🔐 Flujo de Autenticación SSO**

**1. 🔑 Proceso de Login**

sequenceDiagram

participant U as Usuario

participant F as Frontend

participant G as API Gateway

participant A as Auth Service

participant D as Database

U->>F: 1. Abre index.html

U->>F: 2. Ingresa credenciales

F->>G: 3. POST /auth/login

G->>A: 4. Proxy a Auth Service

A->>D: 5. Verificar credenciales

D->>A: 6. Usuario válido

A->>A: 7. Generar JWT Token

A->>G: 8. Respuesta con token

G->>F: 9. Token + user info

F->>F: 10. Guardar en localStorage

F->>U: 11. Redirigir a dashboard

**2. 🛡️ Verificación en Microservicios**

sequenceDiagram

participant F as Frontend

participant G as API Gateway

participant M as Microservicio

participant A as Auth Service

F->>G: 1. Request + Authorization: Bearer TOKEN

G->>M: 2. Proxy request con headers

M->>A: 3. Verificar token

A->>M: 4. Usuario válido + permisos

M->>M: 5. Procesar lógica de negocio

M->>G: 6. Respuesta con datos

G->>F: 7. Datos al frontend

**🗂️ Estructura del Proyecto**

microservices-auth/

├── 📁 api-gateway/ # Gateway principal

│ ├── src/

│ │ └── index.js # Servidor gateway

│ ├── package.json

│ └── Dockerfile

│

├── 📁 auth-service/ # Servicio de autenticación

│ ├── src/

│ │ ├── index.js # Servidor principal

│ │ ├── middleware/

│ │ │ └── auth.js # Middleware de auth

│ │ └── config/

│ │ ├── database.js # Conexión PostgreSQL

│ │ └── redis.js # Conexión Redis

│ ├── package.json

│ └── Dockerfile

│

├── 📁 user-service/ # Servicio de usuarios

│ ├── src/

│ │ └── index.js # Servidor básico

│ ├── package.json

│ └── Dockerfile

│

├── 📁 hello-service/ # Servicio de ejemplo

│ ├── src/

│ │ └── index.js # Demostración SSO

│ ├── package.json

│ └── Dockerfile

│

├── 📁 frontend/ # Interfaz de usuario

│ ├── index.html # Pantalla de login

│ ├── dashboard.html # Panel administrativo

│ ├── demo-complete.html # Demo completo

│ └── profile.html # Gestión de perfil

│

├── 📁 database/ # Scripts de BD

│ ├── init.sql # Inicialización básica

│ └── enhanced\_init.sql # Schema completo

│

├── 📁 shared/ # Librerías compartidas

│ └── middleware/

│ └── microservice-auth.js # Auth para microservicios

│

├── docker-compose.yml # Orquestación

└── README.md # Documentación

**🔧 Componentes del Sistema**

**🚪 API Gateway (Puerto 3000)**

**Responsabilidades:**

* ✅ **Punto de entrada único** para todas las requests
* ✅ **Enrutamiento inteligente** a microservicios
* ✅ **Manejo de CORS** para frontend
* ✅ **Rate limiting** y seguridad básica
* ✅ **Proxy transparente** con headers

**Rutas principales:**

// Autenticación

POST /auth/login → Auth Service

GET /auth/verify → Auth Service

GET /auth/profile → Auth Service

PUT /auth/password → Auth Service

// Gestión de usuarios

GET /users → Auth Service

POST /users → Auth Service

GET /users/stats/overview → Auth Service

GET /users/export/csv → Auth Service

// Hello Service (ejemplo)

GET /hello → Hello Service

GET /hello/private → Hello Service

GET /hello/admin → Hello Service

GET /whoami → Hello Service

// Verificación de permisos

POST /auth/check-permission → Auth Service

GET /auth/microservice-health → Auth Service

**🔐 Auth Service (Puerto 3001)**

**Responsabilidades:**

* ✅ **Autenticación de usuarios** con JWT
* ✅ **Gestión de usuarios** CRUD completa
* ✅ **Sistema de permisos** granulares
* ✅ **Verificación de tokens** para otros servicios
* ✅ **Estadísticas** y reportes de usuarios

**Funcionalidades clave:**

// Autenticación

- Login con email/password

- Generación de JWT tokens (24h validez)

- Verificación de tokens

- Logout (blacklist opcional)

// Gestión de usuarios

- Crear, leer, actualizar, eliminar usuarios

- Cambio de contraseñas (propia y admin)

- Activar/desactivar usuarios

- Búsqueda y filtrado avanzado

// Sistema de permisos

- Verificación de permisos por usuario

- Roles predefinidos (admin, moderator, user)

- Permisos granulares por servicio/acción

- Asignación dinámica de permisos

// Estadísticas

- Resumen de usuarios activos/inactivos

- Distribución por roles

- Actividad reciente

- Exportación de datos

**👋 Hello Service (Puerto 3010)**

**Propósito:** Demostración de integración SSO

**Rutas de ejemplo:**

GET /health // Público

GET /hello // Público

GET /hello/private // Requiere login

GET /hello/admin // Solo admins

GET /whoami // Info del usuario actual

**📱 Frontend Unificado**

**Componentes:**

1. **index.html** - Pantalla de login elegante
   * Formulario de autenticación
   * Validación de credenciales
   * Redirección automática
   * Manejo de errores
2. **dashboard.html** - Panel administrativo completo
   * Gestión de usuarios
   * Estadísticas en tiempo real
   * Configuración de perfiles
   * Navegación entre módulos
3. **demo-complete.html** - Demo interactivo
   * Pruebas de todos los servicios
   * Logs en tiempo real
   * Verificación de permisos
   * Ejemplos de integración

**🛡️ Sistema de Seguridad**

**🔑 Autenticación JWT**

// Estructura del Token

{

"header": {

"alg": "HS256",

"typ": "JWT"

},

"payload": {

"id": "user-uuid",

"email": "admin@admin.com",

"role": "admin",

"firstName": "Admin",

"lastName": "User",

"iat": 1642123456,

"exp": 1642209856

},

"signature": "hash\_secreto"

}

**🛡️ Middleware de Verificación**

// En cada microservicio

const authMiddleware = async (req, res, next) => {

// 1. Extraer token del header Authorization

// 2. Verificar con Auth Service

// 3. Agregar req.user con info del usuario

// 4. Continuar o rechazar request

};

**🔐 Permisos Granulares**

-- Estructura de permisos

permissions:

- projects.create

- projects.read

- projects.update

- projects.delete

- finance.read

- finance.manage

- users.create

- users.manage

**👥 Usuarios del Sistema**

**Usuarios Predeterminados:**

| **Usuario** | **Email** | **Password** | **Rol** | **Permisos** |
| --- | --- | --- | --- | --- |
| **Admin** | admin@admin.com | admin123 | admin | ✅ Todos los permisos |
| **Usuario** | testuser@test.com | Test123 | user | ❌ Solo lectura básica |

**Roles Disponibles:**

const rolePermissions = {

'admin': [

'projects.create', 'projects.read', 'projects.update', 'projects.delete',

'contracts.create', 'contracts.read', 'contracts.update',

'finance.read', 'finance.manage',

'users.create', 'users.read', 'users.update', 'users.delete'

],

'moderator': [

'projects.read', 'projects.update',

'contracts.read',

'finance.read',

'users.read'

],

'user': [

'projects.read',

'contracts.read'

]

};

**🚀 Cómo Usar el Sistema**

**1. 🏁 Inicialización**

# Clonar y navegar al proyecto

cd microservices-auth

# Levantar todos los servicios

docker-compose up --build

# Verificar que estén corriendo

docker-compose ps

**2. 🔐 Acceso al Sistema**

1. 🌐 Abrir: file:///path/to/frontend/index.html

2. 📝 Credenciales: admin@admin.com / admin123

3. 🔑 Click "Login Rápido"

4. 🎯 Automáticamente redirige a dashboard

**3. 🧪 Pruebas del Sistema**

# Health checks

curl http://localhost:3000/health

curl http://localhost:3000/auth/microservice-health

curl http://localhost:3000/hello/health

# Login por API

curl -X POST http://localhost:3000/auth/login \

-H "Content-Type: application/json" \

-d '{"email":"admin@admin.com","password":"admin123"}'

# Usar token (reemplazar TOKEN)

curl http://localhost:3000/hello/private \

-H "Authorization: Bearer TOKEN"

**📊 Monitoreo y Logs**

**📈 Dashboard de Estadísticas**

* Total de usuarios activos/inactivos
* Distribución por roles
* Usuarios registrados por mes
* Actividad reciente de logins

**📋 Logs del Sistema**

🔐 Auth Service logs:

- Login attempts (exitosos/fallidos)

- Token verifications

- Permission checks

- User management actions

👋 Hello Service logs:

- Request authentication

- Permission verifications

- User access patterns

🚪 API Gateway logs:

- Request routing

- Proxy operations

- Error handling

**🛠️ Configuración**

**🌍 Variables de Entorno**

# API Gateway

PORT=3000

AUTH\_SERVICE\_URL=http://auth-service:3001

USER\_SERVICE\_URL=http://user-service:3002

# Auth Service

PORT=3001

JWT\_SECRET=mi\_super\_secreto\_jwt\_2024

DATABASE\_URL=postgresql://postgres:password@postgres:5432/auth\_db

REDIS\_URL=redis://redis:6379

# Base de Datos

POSTGRES\_DB=auth\_db

POSTGRES\_USER=postgres

POSTGRES\_PASSWORD=password

**🐳 Docker Services**

Services activos:

- api-gateway:3000 # Gateway principal

- auth-service:3001 # Autenticación

- user-service:3002 # Gestión usuarios

- hello-service:3010 # Demo SSO

- postgres:5432 # Base de datos

- redis:6379 # Cache

- pgadmin:5050 # Admin BD

**🔮 Escalabilidad**

**➕ Agregar Nuevo Microservicio**

// 1. Crear nuevo servicio

mi-nuevo-servicio/

├── src/index.js // Usar middleware auth

├── package.json

└── Dockerfile

// 2. Agregar a docker-compose.yml

mi-nuevo-servicio:

build: ./mi-nuevo-servicio

ports: ["3021:3021"]

// 3. Agregar rutas al API Gateway

app.get('/nuevo-servicio/\*', proxyToService);

// 4. ¡Listo! SSO automático

**🎯 Beneficios del Sistema**

✅ **Para Desarrolladores:**

* Autenticación pre-construida
* Middleware reutilizable
* Permisos granulares listos
* Integración transparente

✅ **Para Usuarios:**

* Single Sign-On real
* Experiencia unificada
* Seguridad robusta
* Interface moderna

✅ **Para DevOps:**

* Containerizado completo
* Escalabilidad horizontal
* Monitoring integrado
* Despliegue automático

**🆘 Solución de Problemas**

**❌ Errores Comunes**

**"Token inválido o expirado"**

# Verificar que Auth Service esté corriendo

docker-compose logs auth-service

# Renovar token haciendo login de nuevo

**"Servicio no disponible"**

# Verificar todos los servicios

docker-compose ps

# Reiniciar servicios

docker-compose down && docker-compose up

**"Permisos insuficientes"**

-- Verificar permisos del usuario

SELECT \* FROM user\_permissions\_view WHERE email = 'usuario@email.com';

**🎯 Roadmap y Futuro**

**🚧 Próximas Mejoras**

* [ ] Refresh tokens automáticos
* [ ] Two-Factor Authentication (2FA)
* [ ] Single Logout (SLO)
* [ ] OAuth2/OIDC integration
* [ ] Microservicio de notifications
* [ ] Microservicio de file storage
* [ ] Dashboard analytics avanzado
* [ ] Mobile app integration

**🎪 Microservicios Futuros**

* [ ] **Projects Service** - Gestión de proyectos
* [ ] **Finance Service** - Contabilidad y finanzas
* [ ] **Documents Service** - Gestión documental
* [ ] **Reports Service** - Reportes y analytics
* [ ] **Notifications Service** - Notificaciones push
* [ ] **Calendar Service** - Gestión de calendarios

**📞 Soporte y Contacto**

**📚 Documentación Adicional**

* [Guía del Desarrollador](https://claude.ai/chat/developer-guide.md)
* [API Reference](https://claude.ai/chat/api-docs.md)
* [Docker Setup](https://claude.ai/chat/docker-guide.md)

**🛠️ Herramientas de Debug**

* pgAdmin: http://localhost:5050
* API Gateway Health: http://localhost:3000/health
* Auth Service Health: http://localhost:3001/health
* Hello Service Health: http://localhost:3010/health

**✨ Conclusión**

Este sistema SSO proporciona una **base sólida y escalable** para construir aplicaciones empresariales con múltiples microservicios. La arquitectura permite **desarrollo ágil**, **seguridad robusta** y **experiencia de usuario unificada**.