**🚀 Sistema de Autenticación Centralizado - Documentación Completa**

**🎯 Descripción General**

Tu sistema es una **solución completa de Single Sign-On (SSO)** que permite a múltiples microservicios compartir un sistema de autenticación centralizado. Funciona como el "guardián" de todo tu ecosistema de aplicaciones.

**🏗️ Arquitectura del Sistema**

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

│ NGINX (Gateway) │

│ Puerto 80 - Punto de entrada │

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

│ │

┌──────────▼──────────┐ ┌▼──────────────────┐

│ Frontend (React) │ │ Backend (Fastify) │

│ Puerto 3001 │ │ Puerto 3000 │

│ Panel Admin │ │ API + JWT │

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

│

┌────────▼─────────┐

│ PostgreSQL │

│ Puerto 5432 │

│ Base de Datos │

└──────────────────┘

**📦 Componentes del Sistema**

**1. Frontend - Panel de Administración**

**Tecnología:** React + TypeScript + TailwindCSS

**Funcionalidades:**

* ✅ **Dashboard con métricas** del sistema
* ✅ **Gestión de usuarios** (crear, editar, activar/desactivar)
* ✅ **Gestión de roles y permisos** granulares
* ✅ **Monitoreo de microservicios** con health checks
* ✅ **Login/logout** con manejo de sesiones
* ✅ **Interfaz responsiva** y moderna

**Páginas disponibles:**

* /login - Página de inicio de sesión
* /dashboard - Métricas y estado del sistema
* /users - CRUD completo de usuarios
* /roles - Gestión de roles y permisos
* /microservices - Registro y monitoreo de servicios

**2. Backend - API de Autenticación**

**Tecnología:** Fastify + TypeScript + Prisma ORM

**APIs disponibles:**

**🔐 Autenticación (/api/auth)**

* POST /login - Inicio de sesión
* POST /refresh - Renovar tokens
* POST /logout - Cerrar sesión específica
* POST /logout-all - Cerrar todas las sesiones
* GET /me - Información del usuario actual
* GET /sessions - Sesiones activas del usuario
* GET /validate - **Endpoint especial para validación de tokens** 🚨

**👥 Usuarios (/api/users)**

* GET /users - Listar usuarios (con paginación y filtros)
* GET /users/:id - Obtener usuario específico
* POST /users - Crear usuario
* PUT /users/:id - Actualizar usuario
* DELETE /users/:id - Eliminar usuario (soft delete)
* PUT /users/:id/activate - Activar usuario
* PUT /users/:id/deactivate - Desactivar usuario
* PUT /users/:id/password - Cambiar contraseña
* GET /users/:id/roles - Roles del usuario
* PUT /users/:id/roles - Asignar roles

**🛡️ Roles (/api/roles)**

* GET /roles - Listar roles
* GET /roles/permissions - **Permisos disponibles del sistema**
* GET /roles/:id - Obtener rol específico
* POST /roles - Crear rol
* PUT /roles/:id - Actualizar rol
* DELETE /roles/:id - Eliminar rol
* PUT /roles/:id/activate - Activar rol
* PUT /roles/:id/deactivate - Desactivar rol
* GET /roles/:id/users - Usuarios con este rol

**🔧 Microservicios (/api/microservices)**

* GET /microservices - Listar servicios registrados
* GET /microservices/:id - Obtener servicio específico
* POST /microservices - **Registrar nuevo microservicio**
* PUT /microservices/:id - Actualizar configuración
* DELETE /microservices/:id - Eliminar registro
* GET /microservices/:id/health - Health check individual
* POST /microservices/:id/health-check - Ejecutar health check
* POST /microservices/health-check-all - **Health check masivo**
* PUT /microservices/:id/activate - Activar servicio
* PUT /microservices/:id/deactivate - Desactivar servicio

**📊 Dashboard (/api/dashboard)**

* GET /dashboard - **Métricas completas del sistema**

**3. Base de Datos - PostgreSQL**

**Esquema completo:**

**Tablas principales:**

* **users** - Información de usuarios del sistema
* **roles** - Roles con permisos en formato JSON
* **user\_roles** - Relación many-to-many usuarios-roles
* **microservices** - Registro de microservicios del ecosistema
* **sessions** - Sesiones JWT activas/inactivas
* **audit\_logs** - Logs de auditoría de acciones
* **system\_config** - Configuración del sistema

**4. NGINX - Gateway y Proxy**

**Funcionalidades clave:**

* ✅ **Proxy reverso** para frontend y backend
* ✅ **Validación automática** de tokens para microservicios
* ✅ **Load balancing** (preparado para múltiples instancias)
* ✅ **SSL/HTTPS** (configuración preparada)

**🔗 Integración con Microservicios Externos**

**Método 1: Middleware Automático (Node.js)**

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

const token = req.headers.authorization;

if (!token) {

return res.redirect('http://localhost/login');

}

try {

const response = await axios.get('http://localhost/api/auth/validate', {

headers: { Authorization: token }

});

req.user = response.data.user;

req.permissions = response.data.permissions;

next();

} catch (error) {

return res.redirect('http://localhost/login');

}

};

// Usar en rutas protegidas

app.use('/protected', authMiddleware);

**Método 2: Integración via NGINX (Automática)**

El sistema ya incluye configuración para proteger microservicios automáticamente:

# Ejemplo en nginx.conf

location /mi-microservicio/ {

# Validar automáticamente con tu sistema

auth\_request /auth/validate;

# Headers automáticos con info del usuario

auth\_request\_set $user\_id $upstream\_http\_x\_user\_id;

auth\_request\_set $user\_permissions $upstream\_http\_x\_user\_permissions;

proxy\_set\_header X-User-ID $user\_id;

proxy\_set\_header X-User-Permissions $user\_permissions;

# Redirect automático si no autenticado

error\_page 401 = @redirect\_login;

proxy\_pass http://mi-microservicio:3000/;

}

**Método 3: Validación Directa en Microservicio**

# Ejemplo en Python/FastAPI

import requests

def validate\_token(token: str):

response = requests.get(

"http://localhost/api/auth/validate",

headers={"Authorization": f"Bearer {token}"}

)

if response.status\_code == 200:

return response.json()["user"]

return None

@app.middleware("http")

async def auth\_middleware(request: Request, call\_next):

token = request.headers.get("authorization")

if token:

user = validate\_token(token)

request.state.user = user

response = await call\_next(request)

return response

**🎛️ Sistema de Permisos Granular**

**Permisos Disponibles:**

{

"users": ["users.create", "users.read", "users.update", "users.delete"],

"roles": ["roles.create", "roles.read", "roles.update", "roles.delete"],

"microservices": ["microservices.create", "microservices.read", "microservices.update", "microservices.delete"],

"system": ["system.config", "system.logs", "system.health"],

"dashboard": ["dashboard.view", "dashboard.analytics"],

"profile": ["profile.read", "profile.update"],

"special": ["\*"] // Super admin - todos los permisos

}

**Roles por Defecto:**

* **super\_admin** - Acceso completo (\*)
* **admin** - Gestión de usuarios y microservicios
* **user** - Solo perfil propio y dashboard básico

**🔒 Seguridad Implementada**

**JWT Tokens:**

* ✅ **Access Token**: 15 minutos (configurable)
* ✅ **Refresh Token**: 7 días (configurable)
* ✅ **Renovación automática** desde el frontend
* ✅ **Revocación** de sesiones específicas o masiva

**Contraseñas:**

* ✅ **Hash bcrypt** con 12 rounds
* ✅ **Políticas de contraseñas** (8+ caracteres, mayús, minus, números)
* ✅ **Bloqueo temporal** por intentos fallidos (5 intentos = 15 min)

**Auditoría:**

* ✅ **Logs de todas las acciones** (login, cambios, etc.)
* ✅ **Tracking de IP y User-Agent**
* ✅ **Registro de cambios** (valores anteriores y nuevos)

**📊 Monitoreo y Health Checks**

**Health Checks Automáticos:**

* ✅ **Verificación cada 5 minutos** de todos los microservicios
* ✅ **Estados**: Healthy, Unhealthy, Unreachable, Timeout
* ✅ **Métricas de respuesta** (tiempo, status codes)
* ✅ **Alertas** para servicios con problemas

**Dashboard Métricas:**

* 📈 **Usuarios**: Total, activos, inactivos
* 📈 **Roles**: Distribución y uso
* 📈 **Microservicios**: Estados y salud
* 📈 **Sistema**: Uptime, BD, rendimiento

**🚀 Configuración y Despliegue**

**Variables de Entorno Importantes:**

# Base de datos

DATABASE\_URL=postgresql://admin:password123@postgres:5432/auth\_system

# JWT

JWT\_SECRET=tu-clave-super-secreta

JWT\_EXPIRES\_IN=15m

JWT\_REFRESH\_EXPIRES\_IN=7d

# Usuario admin

DEFAULT\_ADMIN\_USERNAME=admin

DEFAULT\_ADMIN\_PASSWORD=admin123

# Seguridad

MAX\_LOGIN\_ATTEMPTS=5

LOCKOUT\_TIME=15m

**Comandos Docker:**

# Levantar todo el sistema

docker-compose up -d

# Ver logs en tiempo real

docker-compose logs -f

# Reiniciar servicios

docker-compose restart backend frontend

# Acceder a la BD

docker-compose exec postgres psql -U admin -d auth\_system

**🌟 Casos de Uso Principales**

**1. Empresa con Múltiples Aplicaciones**

* Portal de empleados
* Sistema de nóminas
* CRM interno
* Dashboard de ventas
* **Todos comparten el mismo login** 🎯

**2. Plataforma de Microservicios**

* API Gateway centralizado
* Autenticación unificada
* Permisos granulares por servicio
* Monitoreo centralizado

**3. SaaS Multi-tenant**

* Diferentes niveles de usuario
* Roles personalizados por cliente
* Integración con servicios externos

**🔧 Personalización y Extensión**

**Agregar Nuevos Permisos:**

// En roles.service.ts

private readonly AVAILABLE\_PERMISSIONS = [

// ... existentes

'reports.create',

'reports.view',

'analytics.advanced'

];

**Registrar Nuevo Microservicio:**

curl -X POST http://localhost/api/microservices \

-H "Authorization: Bearer <token>" \

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

-d '{

"name": "Mi Nuevo Servicio",

"url": "http://mi-servicio:8080",

"healthCheckUrl": "http://mi-servicio:8080/health",

"requiresAuth": true,

"allowedRoles": ["admin", "user"]

}'

**🎯 Ventajas del Sistema**

✅ **Single Sign-On**: Un login para todos los servicios  
✅ **Escalable**: Soporta múltiples microservicios  
✅ **Seguro**: JWT + bcrypt + auditoría  
✅ **Monitoreable**: Health checks automáticos  
✅ **Fácil integración**: APIs simples y middleware  
✅ **Panel completo**: Gestión visual de todo  
✅ **Docker ready**: Despliegue con un comando

**🚨 Endpoints Críticos para Integración**

**🔑 Endpoint Principal de Validación**

GET /api/auth/validate

Authorization: Bearer <access-token>

Response:

{

"valid": true,

"user": {

"id": 1,

"username": "admin",

"permissions": ["\*"]

}

}

**📝 Registrar Microservicio**

POST /api/microservices

{

"name": "Mi Servicio",

"url": "http://mi-servicio:3000",

"requiresAuth": true

}

**🎉 Resumen**

Tu sistema es una **solución enterprise completa** que proporciona:

1. **Autenticación centralizada** para todo tu ecosistema
2. **Panel de administración** completo y profesional
3. **APIs robustas** para integración con cualquier tecnología
4. **Seguridad avanzada** con JWT, bcrypt y auditoría
5. **Monitoreo automático** de todos los servicios
6. **Fácil despliegue** con Docker Compose

Es perfecto para empresas que manejan múltiples aplicaciones y necesitan un sistema de autenticación unificado, seguro y escalable. 🚀