Especificación Técnica

Servicio de Autenticación Multi-Tenant con JWT, Redis y Rate Limiting

---

1. Descripción general del sistema

Este servicio provee **autenticación centralizada** como microservicio reutilizable para múltiples proyectos (tenants). Su objetivo es que los backends clientes **no implementen lógica de autenticación**, sino que deleguen completamente esta responsabilidad.

Funcionalidades principales: - Autenticación basada en JWT - Soporte multi-tenant - Rate limiting con Redis (IP / usuario) - Bloqueo temporal de cuentas - Recuperación de contraseña vía email con código - Administración de tenants

---

2. Convenciones generales

2.1 Headers comunes

Header	Requerido	Descripción
Content-Type	Sí	application/json
X-Tenant-Id	Sí (excepto admin)	Identificador del tenant
X-Tenant-Api-Key	Sí (endpoints sensibles)	Credencial del tenant
Authorization	Opcional	Bearer JWT
X-Admin-Api-Key	Solo admin	Credencial global del sistema

---

3. Endpoints de administración (Admin)

3.1 Crear tenant

POST `/admin/tenants`

Headers

```
X-Admin-Api-Key: <ADMIN_API_KEY>
Content-Type: application/json
```

Body

```
{
  "name": "proyecto_bd_2025"
}
```

Descripción Crea un nuevo tenant (proyecto) que podrá usar el servicio de autenticación.

Funcionamiento esperado - Valida el ADMIN_API_KEY - Genera un TENANT_API_KEY - Guarda el tenant en base de datos - Devuelve el api_key una sola vez

Response

```
{
  "tenantId": "uuid",
  "tenantName": "proyecto_bd_2025",
  "apiKey": "tenant_api_key_generada"
}
```

---

4. Endpoints de autenticación

4.1 Registro de usuario

POST `/auth/register`

Headers

```
X-Tenant-Id: proyecto_bd_2025
X-Tenant-Api-Key: <TENANT_API_KEY>
Content-Type: application/json
```

Body

```
{
  "email": "user@email.com",
  "password": "password123"
}
```

Descripción Registra un usuario dentro del tenant especificado.

Funcionamiento esperado - Valida tenant y api_key - Verifica unicidad del email por tenant - Hashea contraseña con bcrypt - Guarda usuario con FK al tenant

---

4.2 Login

POST `/auth/login`

Headers

```
X-Tenant-Id: proyecto_bd_2025
Content-Type: application/json
```

Body

```
{
  "email": "user@email.com",
  "password": "password123"
}
```

Descripción Autentica al usuario y genera un JWT.

Funcionamiento esperado - Rate limit por IP - Valida credenciales - Controla intentos fallidos - Bloquea cuenta si aplica - Genera JWT con claims

Response

```
{
  "accessToken": "jwt_token",
  "expiresIn": 900
}
```

4.3 Refresh token (opcional)

POST `/auth/refresh`

Headers

```
Authorization: Bearer <REFRESH_TOKEN>
```

Descripción Genera un nuevo access token.

4.4 Logout

POST `/auth/logout`

Headers

```
Authorization: Bearer <JWT>
```

Funcionamiento esperado - Invalida el token en Redis (blacklist)

---

5. Recuperación de contraseña

5.1 Solicitar código de recuperación

POST `/auth/password/recover`

Headers

```
X-Tenant-Id: proyecto_bd_2025
Content-Type: application/json
```

Body

```
{
  "email": "user@email.com"
}
```

Funcionamiento esperado - Genera código temporal - Guarda código en Redis con TTL - Envía email

---

5.2 Validar código

POST `/auth/password/validate-code`

Body

```
{
  "email": "user@email.com",
  "code": "123456"
}
```

5.3 Resetear contraseña

POST `/auth/password/reset`

Body

```
{
  "email": "user@email.com",
  "code": "123456",
  "newPassword": "newPassword123"
}
```

---

6. Funcionamiento general del sistema

Flujo típico

1. Admin crea tenant
2. Proyecto guarda TENANT_ID y TENANT_API_KEY
3. Usuarios se registran
4. Usuarios hacen login y reciben JWT
5. Backend cliente valida JWT localmente
6. Auth service maneja seguridad y control

---

7. Rate limiting

- Endpoints públicos: por IP
- Endpoints autenticados: por userId + tenant
- Redis con TTL automático

---

8. Manejo de fallos

- Redis no es crítico
- Si Redis cae: fail-open
- Sistema sigue funcionando

---

9. Conclusión

Este documento define completamente el contrato, comportamiento y arquitectura del servicio de autenticación, permitiendo su reutilización real como microservicio académico y profesional.