

APRENDIZAJE-EXPRESS — ISS-15

Feature Session: Inicio de Sesión, Registro, Perfil y Convivencia de las 3 Modalidades

Orquestación final del ciclo de vida de la sesión, emisión de tokens y cierre de la Fase II (Auth + RBAC)

Propósito del ISS-15 e Integración del Ciclo de Sesión

Punto de Entrada (Login)

- Recibe identifier (usuario o email) y password en clear.
- Valida la existencia del usuario activo en la BD.
- Verifica el hash de contraseña con bcrypt.
- Emite el par de tokens: Access (JWT 15m) y Refresh (Opaco 30d).

Renovación y Salida

- POST /api/sesion/refresh: rota el refresh token con reuse detection.
- Ventana deslizante que extiende la sesión del usuario activo.
- POST /api/sesion/logout: revoca la sesión activa inmediatamente.
- Integración transparente con el módulo RefreshTokens.

Perfil y Permisos Efectivos

- GET /api/sesion/perfil: retorna identidad del usuario (sin hash).
- GET /api/permisos: expone los (method, path) concedidos.
- Demuestra la convivencia de las 3 modalidades de acceso.
- Auditoría con deviceInfo (User-Agent recortado a 500 chars).

Arquitectura por Capas y Contratos DTO de Sesión

Contratos DTO (`features/auth/session/dto/`)

- LoginDto: identifier (minúsculas: username o email) + password.
- RefreshSessionDto: refresh_token viaja en el body (credencial de sesión).
- LogoutSessionDto: refresh_token a revocar explícitamente.
- SessionTokensDto: access_token, token_type, expires_in, refresh_token.
- ProfileDto: id, username, email, status (excluye estrictamente el hash).

Flujo de Responsabilidades

- SessionController: Extiende BaseController.run(), extrae req.body y req.auth, obtiene User-Agent recortado a 500 chars.
- SessionService: Orquesta la autenticación con UsersRepository, la emisión de refresh tokens con RefreshTokensService y la firma de JWT con jwt.ts.
- UsersRepository: Método findByIdentifierWithPassword() que recupera la entidad incluyendo la contraseña para verificación.
- SessionRoutes: Declaración explícita de las tres modalidades de acceso en el enrutado de Express.

Secuencia del Método Login & Emisión de Credenciales

Paso a Paso: POST /api/sesion/login

1. Búsqueda por Identificador: `UsersRepository.findByIdentifierWithPassword(identifier)` busca coincidencia por username O por email.
2. Validación de Estado: Si el usuario no existe o `status !== 'active'`, detiene la ejecución inmediatamente.
3. Verificación de Contraseña: `comparePassword(password, user.password_hash)` valida el hash bcrypt (12 rondas).
4. Emisión de Refresh Token Opaco: `RefreshTokensService.issue(user.id, deviceInfo)` genera un token opaco, guarda su SHA-256 en BD y retorna el valor en claro.
5. Firma de Access Token (JWT): `signAccessToken(user)` firma un JWT HS256 con claims estándar (sub, username, jti, iss, aud, exp=15m).
6. Respuesta Estructurada: Retorna 200 OK con `SessionTokensDto { access_token, token_type: 'Bearer', expires_in: 900, refresh_token }`.

Convivencia de las Tres Modalidades de Acceso

MODALIDAD 1: OPEN

Rutas sin identidad previa

- **POST /api/sesion/login**
- **POST /api/sesion/refresh**
- **POST /api/sesion/logout**

No exigen cabecera Authorization: Bearer.
Nota: OPEN no significa 'sin credencial'; refresh y logout exigen el refresh_token en el body.

MODALIDAD 2: JWT

Rutas con identidad verificada

- **GET /api/sesion/perfil**
- **GET /api/permisos**
- **GET /api/sesiones**

Exigen middleware authenticate. Verifican la firma del JWT y que el usuario siga activo. No requieren concesión granular en la matriz RBAC.

MODALIDAD 3: JWT + RBAC

Rutas protegidas por matriz

- **POST /api/clientes**
- **GET /api/productos**
- **POST /api/ventas**

Exigen authenticate + authorize.
Comprueban que el usuario tenga una concesión activa sobre la tupla (method, path) en resource_roles.

Seguridad Anti-Enumeración y Manejo de Errores Genericos

Principios de Seguridad en la Autenticación

- Mismo 401 'Invalid credentials': Tanto si el usuario no existe como si la contraseña es incorrecta, el servidor responde exactamente la misma excepción `AppError(401, 'Invalid credentials')`. Esto previene ataques de enumeración de usuarios.
- Doble Verificación de Usuario Activo: Si un usuario es desactivado (`status = 'inactive'`), el login falla con 401 genérico y cualquier intento de refresh posterior revoca automáticamente todas sus sesiones abiertas.
- Disyunción Explícita de Errores HTTP: 401 Unauthorized indica un problema de credencial/identidad (ausente, caducado, manipulado). 403 Forbidden indica problema de concesión (sé quién eres, pero no tienes permiso).
- Auditoría de Dispositivo: Cada inicio de sesión registra IP y User-Agent (hasta 500 caracteres) en la tabla `refresh_tokens` para monitoreo de seguridad.

Batería de Pruebas E2E y Smoke Test de las 3 Modalidades

Verificación Observable con `curl`

- 1. Login OPEN (admin / seller): POST /api/sesion/login -> 200 OK + access_token + refresh_token.
- 2. Lectura de Perfil JWT: GET /api/sesion/perfil con Bearer token -> 200 OK con datos de la identidad.
- 3. Consulta de Permisos: GET /api/permisos con token de seller -> 200 OK con 7 recursos (58 para admin).
- 4. Negocio Autorizado (admin): POST /api/clientes con token de admin -> 201 Created.
- 5. Deny by Default (seller): POST /api/clientes con token de seller -> 403 Forbidden.
- 6. Sin Cabecera Authorization: GET /api/clientes sin token -> 401 Unauthorized.
- 7. Renovación Rotativa: POST /api/sesion/refresh -> 200 OK + nuevos tokens; reusar el anterior da 401 y revoca la familia.

CRITERIOS DE CIERRE GATE — ISS-15 Y LABORATORIO FINAL

- [X] npx tsc --noEmit finaliza con cero errores de tipos en todo el proyecto.
- [X] Las 12 features (5 business + 7 auth) y las 11 tablas están completamente operativas.
- [X] Conviven las 3 modalidades de acceso por ruta (OPEN, JWT, JWT + RBAC).
- [X] Matriz RBAC determinista sembrada (2 roles, 58 recursos, 2 usuarios, 65 concesiones).
- [X] No existe la entidad Permission: el permiso es la fila resource_roles(role_id, resource_id).
- [X] Deny by default verificado: cualquier intento de acceso sin concesión activa responde 403.
- [X] Refresh token opaco con hash SHA-256, rotación en cada renovación y detección de reúso.
- [X] Swagger en /api/docs documenta bearerAuth, respuestas 401/403 y modalidad por operación.
- [X] Cierre de la Fase II: ¡Laboratorio Backend Completo en Verde!