SISTEMA DE AUTENTICACIÓN SEGURO

Diseño e Implementación de Mejores Prácticas en Criptografía

Trabajo Práctico Final - Noviembre 2025

Criptografía y Ciberseguridad

RESUMEN EJECUTIVO

El presente informe detalla el diseño de un sistema de autenticación seguro. Se utilizan algoritmos de hashing modernos (Argon2id), tokens JWT con rotación automática, refresh token stateful, y múltiples controles preventivos. El sistema cumple con OWASP Top 10, NIST SP 800-63B y principios de privacidad por diseño.

1. REQUISITOS FUNCIONALES Y NO FUNCIONALES

1.1. Requisitos Funcionales (RF)

Los RF especifican qué funcionalidades debe proporcionar el sistema:

RF1 - Registro El sistema permite crear una cuenta con email y contraseña únicos.

RF2 - Verificación de Email Tras el registro, el usuario verifica su email mediante enlace único y de corta duración.

RF3 - Login El usuario autenticado con email y contraseña correctos recibe tokens de sesión.

RF4 - Logout El usuario cierra su sesión de forma segura revocando todos los tokens.

RF5 - Cambio de Contraseña El usuario autenticado cambia su contraseña (requiriendo la antigua).

RF6 - Recuperación de Contraseña Usuario que olvidó su contraseña recibe enlace de reseteo por email.

RF7 - Gestión de Sesión (Refresh) El sistema renueva sesiones expiradas sin pedir credenciales, mediante refresh token.

1.2. Requisitos No Funcionales (RNF)

Los RNF especifican cómo el sistema implementa seguridad y calidad:

RNF1 - Confidencialidad Las contraseñas se almacenan hasheadas (Argon2id), nunca en texto plano.

RNF2 - Resistencia a Fuerza Bruta Rate limiting en endpoints críticos (login, recuperación).

RNF3 - Transporte Seguro Comunicación cifrada con HTTPS/TLS 1.2+.

RNF4 - Gestión Segura de Sesión Sesiones gestionadas por JWT (acceso) + cookies seguras (refresh).

RNF5 - Prevención XSS Tokens de sesión en cookies HttpOnly, no accesibles por JavaScript.

RNF6 - Prevención CSRF Cookies con flag SameSite=Strict.

RNF7 - Mínimo Privilegio Tokens contienen solo user_id y roles, sin información sensible (PII).

2. MODELO DE DATOS

El modelo prioriza seguridad utilizando UUIDs como claves primarias para mitigar enumeración de recursos.

2.1. Tabla: Usuarios

Columna	Tipo	Descripción
id	UUID PK	Identificador único (UUID v4)
email	VARCHAR UNIQUE	Email como identificador de login
password_hash	VARCHAR	Hash Argon2id de la contraseña
email_verified	BOOLEAN	Flag de verificación de email
created_at	TIMESTAMP	Fecha de creación
updated_at	TIMESTAMP	Fecha de última modificación

2.2. Tabla: Sesiones

Tabla fundamental para gestión stateful de refresh tokens, permitiendo revocación y rotación.

Columna	Tipo	Descripción
id	UUID PK	Identificador de sesión
user_id	UUID FK	Usuario propietario
refresh_token_hash	VARCHAR UNIQUE	Hash SHA-256 del refresh token
ip_address	VARCHAR	IP de origen (auditoría)
user_agent	TEXT	Metadata del cliente
expires_at	TIMESTAMP	Expiración del refresh token
created_at	TIMESTAMP	Creación de sesión

Nota Crítica: El hash del refresh token se almacena, no el token en texto plano. Si la BD fuese comprometida, los atacantes no podrían usar los tokens directamente.

3. FLUJOS DE AUTENTICACIÓN

3.1. Flujo de Login y Emisión de Tokens

1. **Petición**: Cliente envía email y password a `/api/login`.
2. **Rate Limiting**: Servidor aplica límites para mitigar fuerza bruta.
3. **Verificación**: Se busca el usuario por email y verifica contraseña con `argon2.verify()`. Error genérico 401 para evitar enumeración.
4. **Generación de Tokens** (exitosa):
   - **Access Token (JWT)**: Payload mínimo (sub: user.id), expiración 15 min, algoritmo RS256/ES256.
   - **Refresh Token (Opaque)**: String aleatorio 32 bytes, expiración 7 días.
5. **Persistencia**: Hash SHA-256 del refresh token se almacena en tabla Sesiones.
6. **Respuesta**: Ambos tokens enviados en cookies HttpOnly+Secure+SameSite=Strict.

3.2. Flujo de Refresh y Rotación de Tokens

1. **Contexto**: Access token expira, cliente detecta 401 en petición protegida.
2. **Petición**: Cliente envía refresh token a `/api/refresh`.
3. **Validación**:
   - Servidor hashea el refresh token recibido.
   - Busca hash en tabla Sesiones.
   - Si no existe o expiró: 401 Unauthorized, re-login requerido.
4. **Rotación** (implementación clave):
   - Token válido se **elimina inmediatamente** de BD (invalidando su uso futuro).
   - Se genera nuevo access token (15 min).
   - Se genera nuevo refresh token (7 días).
   - Hash del nuevo refresh token se almacena en Sesiones.
5. **Justificación**: Si el token es comprometido, el usuario legítimo será desautenticado en su próximo refresh (token ya invalidado por atacante). Detecta y contiene brechas.
6. **Respuesta**: Nuevos tokens en cookies seguras.

3.3. Flujo de Logout (Revocación)

1. **Petición**: Cliente envía petición a `/api/logout` (con refresh token).
2. **Revocación**: Servidor hashea token, busca y **elimina** registro en Sesiones.
3. **Respuesta**: 204 No Content.
4. **Limpieza Cliente**: Aplicación borra tokens de su almacenamiento.

Este mecanismo stateful permite revocación permanente e inmediata del lado del servidor, a diferencia de JWTs de acceso (stateless).

4. DIAGRAMAS DE SECUENCIA

Los siguientes diagramas ilustran los flujos críticos del sistema de autenticación.

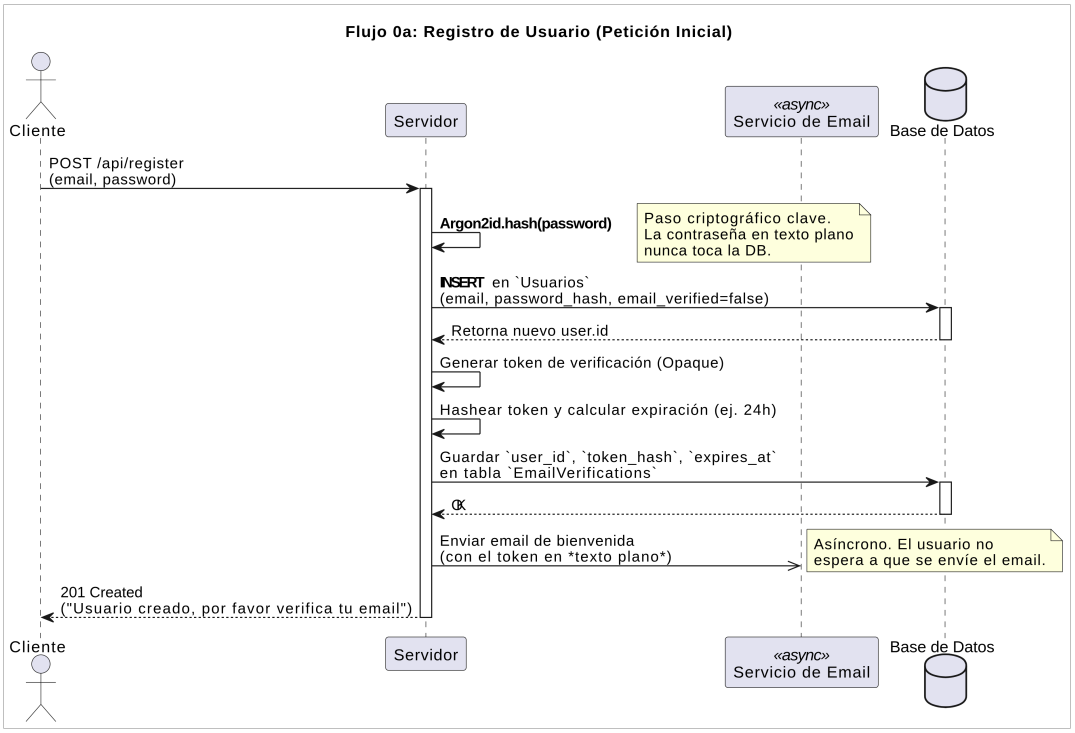

Figura 1: Flujo de Registro: El cliente envía email y contraseña. El servidor valida, hashea la contraseña con Argon2id, y almacena. Se genera token de verificación de email y se envía asincronamente.

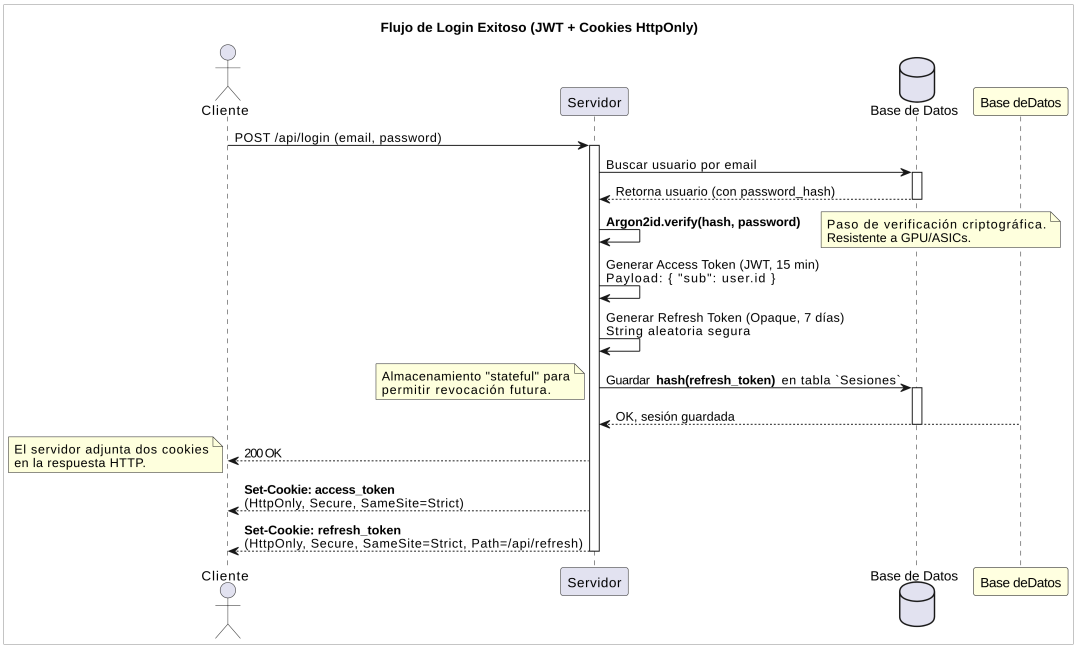

Figura 2: Flujo de Login: Después de validaciones y rate limiting, se generan access token (JWT, 15 min) y refresh token (opaco, 7 días). Ambos se envían en cookies seguras.

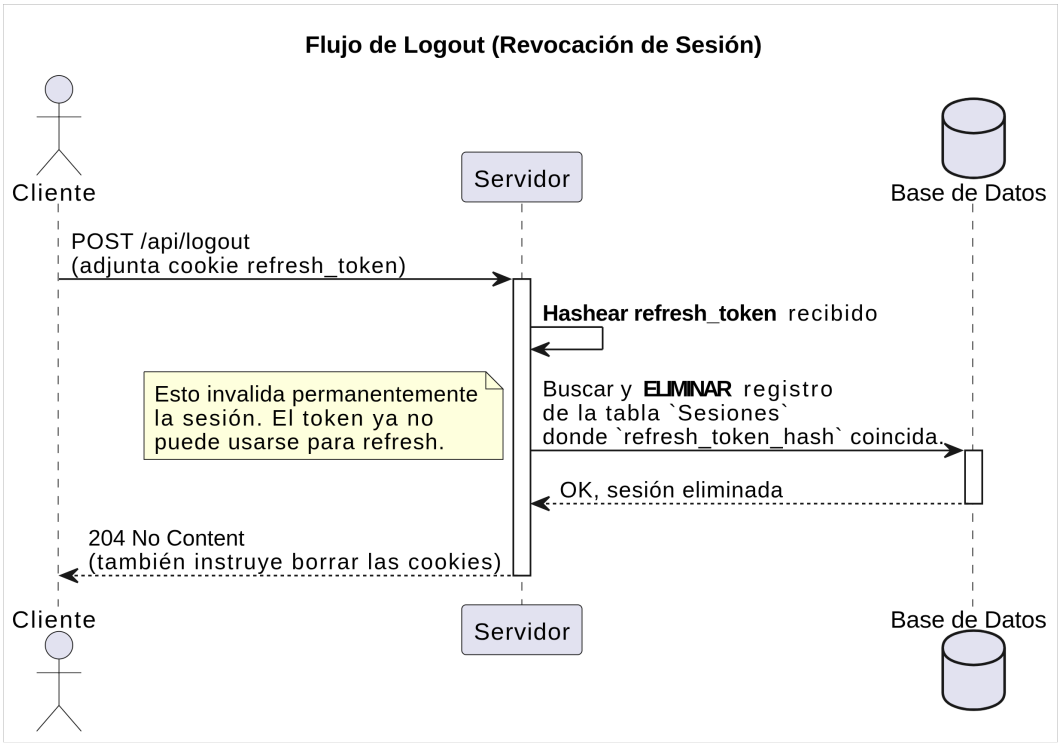

Figura 3: Flujo de Logout: El servidor revoca el refresh token eliminando su hash de la tabla Sesiones. El acceso token se añade a lista negra. Cliente limpia almacenamiento.

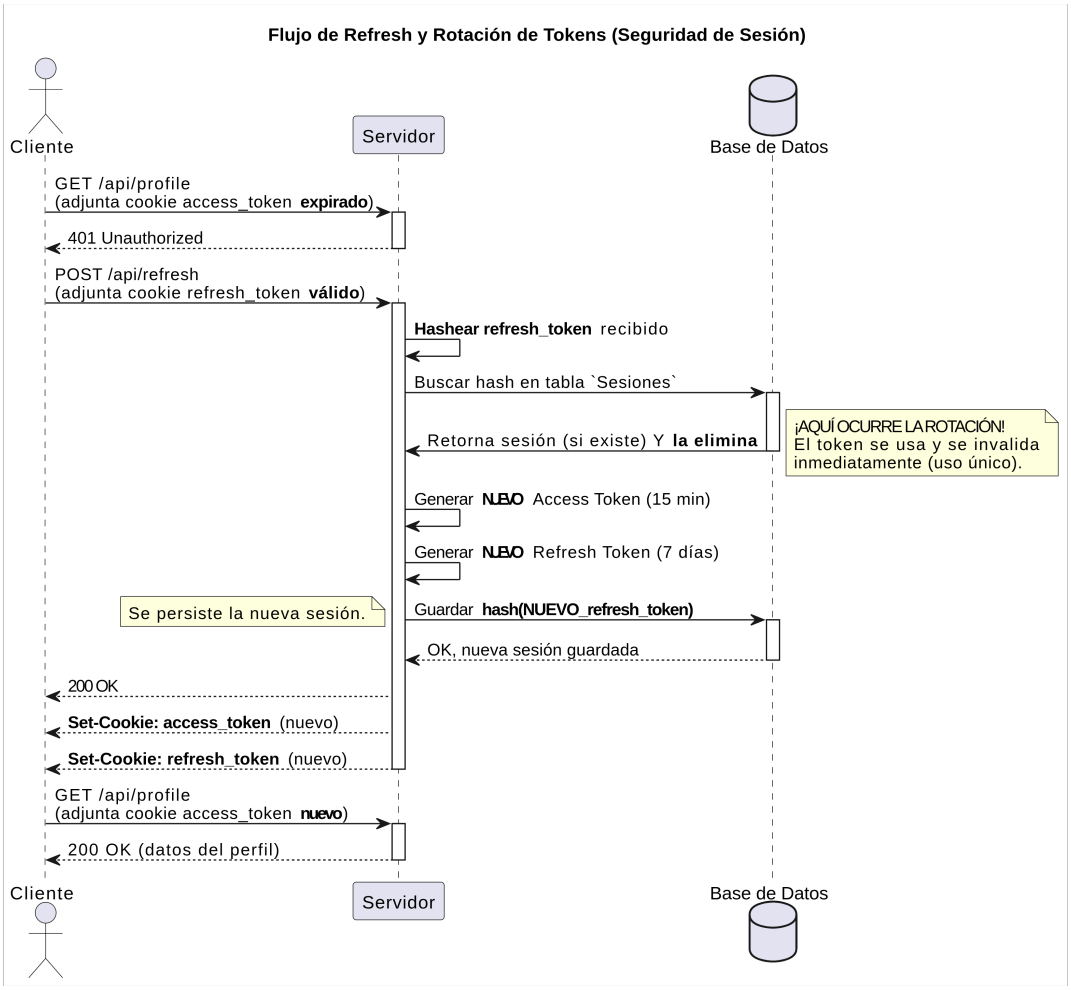

Figura 4: Flujo de Refresh: Rotación de tokens - token antiguo se elimina inmediatamente. Si se detecta reuso, se revocan todos los tokens de la familia (indicador de compromiso).

5. DECISIONES CRIPTOGRÁFICAS

5.1. Hashing de Contraseñas: Argon2id

Algoritmo: Argon2id es la recomendación actual de OWASP y ganador de Password Hashing Competition (2015).

Ventajas:

- **Memory-Hard**: Incrementa costo de fuerza bruta tanto en tiempo como en memoria.
- **GPU/ASIC Resistance**: Resistencia superior al hardware especializado vs. bcrypt/PBKDF2.
- **Moderna**: Diseñada considerando tendencias de hardware actual.

Parámetros OWASP: memoryCost=19456, timeCost=2, parallelism=1.

5.2. Firmas de Tokens: RS256 o ES256

Selección: Algoritmos asimétricos (clave pública/privada) en lugar de HS256.
Ventajas:

- Servicio de autenticación firma con clave privada.
- Microservicios verifican con clave pública.
- Evita compartir secreto simétrico entre servicios.
- Escalabilidad y seguridad mejoradas en arquitecturas distribuidas.

6. GESTIÓN DE SESIONES Y ALMACENAMIENTO DE TOKENS

6.1. Método: Cookies con Seguridad Reforzada

Se descarta LocalStorage/SessionStorage por vulnerabilidad inherente a XSS. Los tokens se almacenan en cookies con atributos de seguridad:

- **HttpOnly=True**: Previene acceso desde JavaScript, mitiga XSS.
- **Secure=True**: Solo transmitidas por HTTPS, previene man-in-the-middle.
- **SameSite=Strict**: Restringe envío a peticiones del mismo origen, previene CSRF.
- **Path Específico**: Refresh token con Path=/api/refresh limita scope.

6.2. Inferencia de Estado en Frontend

Como tokens no son accesibles por JavaScript, el cliente infiere estado de autenticación realizando petición a endpoint protegido (ej. `/api/me`):

- **200 OK**: Sesión válida, datos del usuario retornados.
- **401 Unauthorized**: Usuario no autenticado.

7. PRIVACIDAD Y MINIMIZACIÓN DE DATOS

7.1. Minimización de Datos

Registro: Solo información estrictamente necesaria para autenticación (email, contraseña).

JWT Payload: Access tokens contienen solo user_id (UUID) y roles, excluyendo PII. Si endpoint requiere datos del usuario, consulta BD usando user_id del token.

7.2. Políticas de Retención

- **Sesiones**: Cron job purga registros con expires_at vencido.
- **Logs**: Anonimizados o eliminados tras 90 días de retención.

7.3. Derecho al Olvido

Sistema provee endpoint de eliminación de cuenta (previa re-autenticación):

- Hard delete del registro en tabla Usuarios.
- ON DELETE CASCADE elimina en cascada registros en Sesiones.

8. CONCLUSIONES

El sistema de autenticación diseñado implementa múltiples capas de seguridad: hashing moderno (Argon2id), tokens de corta duración con rotación automática, refresh tokens stateful para revocación, cookies seguras contra XSS/CSRF, y privacidad por diseño. Cumple con estándares OWASP, NIST SP 800-63B y principios de minimización de datos.

Las mejoras futuras incluyen autenticación multifactor (TOTP, WebAuthn), análisis de riesgo adaptativo, y arquitectura zero trust. La seguridad es un proceso continuo que requiere revisiones de código, pentesting, y actualizaciones de dependencias.

REFERENCIAS

1. OWASP. (2021). OWASP Top Ten 2021. https://owasp.org/Top10/
2. OWASP. (2023). Authentication Cheat Sheet. https://cheatsheetseries.owasp.org/
3. NIST. (2020). SP 800-63B: Digital Identity Guidelines - Authentication.
4. RFC 7519: JSON Web Token (JWT). https://tools.ietf.org/html/rfc7519
5. Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: Memory-hard function for password hashing.
6. OWASP. Password Storage Cheat Sheet. https://cheatsheetseries.owasp.org/
7. Mozilla. Web Security Guidelines. https://infosec.mozilla.org/guidelines/web_security
8. EU GDPR (2016/679). Reglamento General de Protección de Datos.