# Uppgift: Lägg till säkerhet med Spring Security

## Beskrivning

Ni ska lägga till säkerhetsfunktioner i ert Spring Boot-projekt med hjälp av Spring Security.

## Grundkrav för Godkänt (G)

### 1. Inloggning med databas

- Skapa inloggning med användarnamn och lösenord

- Spara användare och lösenord i databasen

- Användare kan logga in och ut

- Använd sessions (som Spring Security gör automatiskt)

### 2. Roller och åtkomstkontroll

- Skapa minst 2 roller (USER och ADMIN)

- Olika sidor kräver olika roller (vissa bara för inloggade, andra bara för admins)

- Spara roller i databasen och tilldela till användare

- Skydda minst 3 sidor baserat på roller/inloggning

- Visa olika innehåll beroende på användarens roll och inloggningsstatus

- Omdirigera till inloggningssidan vid obehörig åtkomst

### 3. Säkra lösenord och registrering

- Kryptera lösenord med BCrypt

- Spara aldrig lösenord i klartext

- Implementera lösenordspolicy (minst 8 tecken, kombinera bokstäver och siffror)

- Skapa registreringsfunktion för nya användare med validering

- Nya användare får automatiskt USER-roll

- Kontrollera att användarnamn inte redan finns

### 4. Formulärskydd och input-validering

- Aktivera CSRF-skydd på alla formulär

- Demonstrera att CSRF-skydd fungerar

- Validera all användarinput på server-sidan

- Förhindra XSS-attacker genom korrekt escaping i templates

- Implementera längd- och formatvalidering på formulärfält

### 5. Säker felhantering och logging

- Visa aldrig känslig information i felmeddelanden till användare

- Logga säkerhetsrelaterade händelser (inloggningsförsök, misslyckade försök)

- Skapa anpassade felsidor (404, 403, 500) som inte avslöjar systemdetaljer

- Implementera strukturerad loggning för säkerhetshändelser

## Krav för Väl Godkänt (VG)

\*\*Alla G-krav + följande:\*\*

### 1. JWT tokens istället för sessions

- Ersätt sessions med JWT tokens

- Generera JWT token vid lyckad inloggning

- Validera JWT tokens för alla skyddade endpoints

- Inkludera användarroller i JWT payload

- Konfigurera Spring Security att inte använda sessions (stateless)

- Implementera automatisk token-förnyelse (refresh tokens)

### 2. Metodbaserad säkerhet

- Använd @PreAuthorize-annotationer för att skydda service-metoder

- Implementera rollbaserad kontroll på affärslogik-nivå

- Skydda REST API-endpoints med detaljerad auktorisering

- Demonstrera både roll- och objektbaserad åtkomskontroll

### 3. Avancerat attack-skydd

- Implementera rate limiting för inloggningsförsök (max 5 försök per minut)

- Account lockout efter upprepade misslyckade inloggningar

- Konfigurera säkra HTTP-headers (HSTS, X-Frame-Options, Content Security Policy)

- Implementera session/token timeout med automatisk utloggning

- Skydd mot brute force-attacker

### 4. Databas- och API-säkerhet

- Säkerställ att JPA/Hibernate förhindrar SQL injection

- Konfigurera databas med begränsad användarrättigheter

- Kryptera känslig data utöver lösenord (t.ex. personnummer)

- CORS-konfiguration för säkra cross-origin requests

- Input sanitization för JSON/XML data i REST API

- API rate limiting per endpoint

### 5. Säkerhetsarkitektur och dokumentation

- Förklara JWT vs sessions: fördelar, nackdelar och när man använder vad

- Dokumentera er threat model och vilka hot ni skyddar mot

- Beskriv säkerhetsarkitekturen och designbeslut

- Analysera kvarvarande säkerhetsrisker och begränsningar

- Reflektera över säkerhet kontra användarupplevelse

## Bonuskrav för intresserade

### Avancerad autentisering

- Implementera två-faktor autentisering (2FA) med TOTP

- "Remember me" funktionalitet med säkra persistent tokens

- Lösenordsåterställning via e-post med säkra tokens

### Säkerhetstestning och övervakning

- Genomför penetration testing av er egen applikation

- Använd säkerhetsverktyg som OWASP ZAP

- Implementera audit logging av alla kritiska säkerhetshändelser

- Automatisk varning vid misstänkta säkerhetsincidenter

## Inlämning

### Kod och dokumentation

- Komplett källkod via GitHub-länk

## Bedömningskriterier

### Godkänt (G)

- Komplett autentisering och auktorisering med databas

- Fungerande rollsystem med minst USER och ADMIN

- Säker lösenordshantering och användarregistrering

- CSRF-skydd och input-validering implementerat

- Säker felhantering utan informationsläckage

- Tydlig kod med kommentarer kring säkerhetsbeslut

### Väl Godkänt (VG)

- Alla G-krav plus avancerad JWT-implementation

- Metodbaserad säkerhet med @PreAuthorize

- Omfattande skydd mot vanliga webbattacker

- Databas- och API-säkerhet på produktionsnivå

- Djup förståelse av säkerhetsarkitektur och trade-offs

- Professionell dokumentation och reflektion

### Bonus

- Avancerade autentiseringsmetoder (2FA, etc.)

- Säkerhetstestning av egen applikation

- Produktionsklar säkerhetskonfiguration

- Innovation inom säkerhetsområdet

## Deadline

Fredag 26 september kl. 23:59

## Tips och vägledning

- \*\*Börja med G-kraven\*\* - bygg en solid grund innan ni går vidare

- \*\*Testa säkerheten kontinuerligt\*\* - försök komma åt skyddade resurser utan behörighet

- \*\*Läs Spring Security-dokumentationen\*\* - den är er bästa vän

- \*\*Använd OWASP Top 10\*\* som guide för vilka hot ni ska skydda mot

- \*\*För VG: Fokusera på arkitektur\*\* - förstå varför, inte bara hur

- \*\*Tänk på användarupplevelsen\*\* - säkerhet ska inte göra systemet oanvändbart

## Koppling till hemtentan

Denna praktiska uppgift är nära kopplad till hemtentans teoretiska analys:

- Era implementationsbeslut här blir underlag för hemtentans hotanalys

- Erfarenheterna från utveckling hjälper er att ge djupare svar på hemtentans säkerhetsfrågor

- Problem ni stöter på här kan bli viktiga reflektioner i hemtentan