Μη Λειτουργικές Απαιτήσεις

1. Ασφάλεια Web UI (Stateful Authentication)

Το Web UI της εφαρμογής υλοποιεί **stateful authentication**, βασισμένο σε **sessions και cookies**, αξιοποιώντας το Spring Security.

Χαρακτηριστικά:

- Η αυθεντικοποίηση χρηστών πραγματοποιείται μέσω φόρμας login (/login).
- Μετά την επιτυχή σύνδεση, δημιουργείται **HTTP session** και αποθηκεύεται cookie (JSESSIONID).
- Η αποσύνδεση (/logout) ακυρώνει το session και διαγράφει το cookie.
- Η πρόσβαση στις σελίδες του UI ελέγχεται με **role-based authorization**.

Εκτός από τους κανόνες πρόσβασης που ορίζονται στο SecurityFilterChain, το Web UI της εφαρμογής εφαρμόζει επιπλέον έλεγχο εξουσιοδότησης σε επίπεδο controller, μέσω της χρήσης του annotation @PreAuthorize.

Παραδείγματα:

- Πρόσβαση φοιτητών:
- @PreAuthorize("hasRole('STUDENT')")

  Χρησιμοποιείται σε λειτουργίες όπως:

  - Προβολή διαθεσιμότητας χώρων
  - Δημιουργία και διαχείριση κρατήσεων
  - Προβολή ιστορικού κρατήσεων
- Πρόσβαση προσωπικού:
- @PreAuthorize("hasRole('STAFF')")

  Χρησιμοποιείται σε λειτουργίες όπως:

  - Διαχείριση χώρων μελέτης
  - Προβολή στατιστικών πληρότητας
  - Ακύρωση κρατήσεων από προσωπικό

2. Ασφάλεια REST API (Stateless JWT Authentication)

Το REST API της εφαρμογής υλοποιεί **stateless authentication**, βασισμένο σε **JWT (JSON Web Tokens)**, ώστε να μπορεί να καταναλωθεί από εξωτερικά συστήματα.

Χαρακτηριστικά:

- Όλα τα endpoints κάτω από /api/v1/** είναι **stateless**.
- Δεν χρησιμοποιούνται sessions ή cookies.
- Η αυθεντικοποίηση γίνεται μέσω HTTP header: Authorization: Bearer <JWT>
- Τα JWT tokens εκδίδονται από το endpoint:
- POST /api/v1/auth/client-tokens
- Η επικύρωση των tokens γίνεται μέσω του JwtAuthenticationFilter.

Ρόλος API:

- **INTEGRATION**
  - Πρόσβαση σε analytics και δεδομένα (read-only)
  - Κατανάλωση REST endpoints από τρίτα συστήματα

Μηχανισμοί:

- Έλεγχος ρόλων με @PreAuthorize
- Custom handlers για:
  - Unauthorized (401)
  - Forbidden (403)
- Απόλυτος διαχωρισμός UI και API security chains

3. Έκδοση JWT για Integration Clients

Το REST API υποστηρίζει **machine-to-machine authentication** μέσω client credentials.

Διαδικασία:

1. Το εξωτερικό σύστημα καλεί:
2. POST /api/v1/auth/client-tokens
3. Παρέχει clientId και clientSecret.
4. Το σύστημα:
   - επαληθεύει τον client,
   - εκδίδει JWT με:subject,roles,expiration time.
5. Το token επιστρέφεται και χρησιμοποιείται σε επόμενες κλήσεις API.

Το JWT περιλαμβάνει:subject(client identity),issuer,audience,ρόλους(roles), χρόνο έκδοσης και λήξης.

Ο client αποστέλλει το token σε κάθε επόμενο request: Authorization: Bearer <token>.

JwtAuthenticationFilter

Κάθε API request περνά από JwtAuthenticationFilter, το οποίο:

- ελέγχει την ύπαρξη του Authorization header,
- επαληθεύει την εγκυρότητα του token,
- εξάγει τα roles,
- δημιουργεί Authentication αντικείμενο στο SecurityContext.

Αν το token είναι άκυρο ή ληγμένο:

- το request απορρίπτεται άμεσα με HTTP 401

3. Authorization & Role-Based Access Control (RBAC)

Το σύστημα εφαρμόζει αυστηρό έλεγχο ρόλων, τόσο στο UI όσο και στο API.

Ρόλος	Πρόσβαση
STUDENT	Κρατήσεις, ιστορικό, UI λειτουργίες
STAFF	Διαχείριση χώρων, ακυρώσεις, στατιστικά
INTEGRATION	REST API analytics & data

Η ανάθεση δικαιωμάτων γίνεται:

- Στο UI μέσω Spring Security routes και PreAuthorize
- Στο API μέσω JWT claims και @PreAuthorize

4. Τεκμηρίωση REST API (OpenAPI / Swagger)

Το REST API τεκμηριώνεται πλήρως μέσω **OpenAPI Specification** και προβάλλεται με **Swagger UI**.

Χαρακτηριστικά:

- Χρήση springdoc-openapi
- Swagger UI διαθέσιμο στο: localhost:8080/swagger-ui/index.html#/

- Περιλαμβάνονται μόνο τα REST endpoints (όχι UI controllers).
- Δηλωμένο security scheme:
    - BearerAuth (JWT)
- Ομαδοποίηση endpoints:
    - /api/v1/**

Η τεκμηρίωση επιτρέπει:

- Δοκιμή endpoints
- Παρουσίαση request/response σχημάτων
- Κατανόηση των απαιτούμενων ρόλων και tokens

5. Διαχείριση Σφαλμάτων

Το σύστημα χρησιμοποιεί **διαφορετικό μηχανισμό διαχείρισης σφαλμάτων** για UI και API.

REST API Error Handling (JSON-based)

Το REST API δεν επιστρέφει HTML σελίδες, αλλά δομημένα JSON errors.

GlobalErrorHandlerRestControllerAdvice

Ο κεντρικός error handler του API:

- μετατρέπει exceptions σε HTTP status codes,
- επιστρέφει αντικείμενο ApiError με:timestamp,status,error,message,path

AuthenticationEntryPoint (401 – Unauthorized)

Η κλάση RestApiAuthenticationEntryPoint ενεργοποιείται όταν:

- ο χρήστης **δεν είναι authenticated**
- προσπαθεί να προσπελάσει προστατευμένο API endpoint

Τότε:

- επιστρέφεται **HTTP 401**
- με JSON απόκριση ApiError
- χωρίς redirect ή HTML

AccessDeniedHandler (403 – Forbidden)

Η κλάση RestApiAccessDeniedHandler ενεργοποιείται όταν:

- ο χρήστης είναι authenticated
- δεν έχει τα απαιτούμενα δικαιώματα (roles)

Τότε:

- επιστρέφεται **HTTP 403**
- με JSON error response

Web UI

GlobalErrorHandlerControllerAdvice

- Χειρίζεται exceptions που προκύπτουν στο UI layer
- Επιστρέφει **HTML error pages** (π.χ. 404.html, error.html)
- Παρέχει φιλικά μηνύματα στον χρήστη

6.Input Validation και Ακεραιότητα Δεδομένων

Validation στο Domain Layer

Στο domain επίπεδο, οι οντότητες και τα DTOs έχουν validation annotations (π.χ. @NotNull, @NotBlank, @Size, @Positive, @Email) ώστε να αποτρέπεται η δημιουργία ή αποθήκευση μη έγκυρων αντικειμένων. Επιπλέον, εφαρμόζεται επιχειρησιακή επικύρωση στο service layer (π.χ. έλεγχος μοναδικότητας, έλεγχος ρόλων, χωρητικότητας και ποινών).

Validation σε Φόρμες (Web UI)

Στο Web UI, η επικύρωση γίνεται κατά το binding των φορμών μέσω DTOs με validation constraints. Σε περίπτωση σφαλμάτων, το αίτημα δεν προωθείται στο business logic και ο χρήστης ενημερώνεται άμεσα με κατάλληλα μηνύματα σφάλματος.

7. CORS και Ασφαλής Διαλειτουργικότητα

Για το REST API εφαρμόζεται CORS configuration:

- Επιτρέπεται πρόσβαση μόνο σε καθορισμένα origins.
- Υποστηρίζονται τα απαραίτητα HTTP methods.
- Δεν χρησιμοποιούνται credentials (cookies), καθώς το API είναι stateless.

Αυτό επιτρέπει την ασφαλή κατανάλωση του API από εξωτερικές εφαρμογές.