Παραδοχές Συστήματος

Στο πλαίσιο της ανάπτυξης της εφαρμογής Study Rooms, κάναμε τις παρακάτω παραδοχές σχετικά με το σύστημα, τους χρήστες, τις τεχνολογίες και το περιβάλλον ανάπτυξης και εκτέλεσης. Οι παραδοχές αυτές καθορίζουν τα όρια, τις τεχνικές επιλογές και το γενικό πλαίσιο μέσα στο οποίο σχεδιάστηκε και υλοποιήθηκε το σύστημα.

1. Παραδοχές σχετικά με το Σύστημα

- Το σύστημα υλοποιείται ως μία ενιαία εφαρμογή Spring Boot, η οποία περιλαμβάνει τόσο το Web UI όσο και το REST API.
- Ακολουθείται layered αρχιτεκτονική (Controller – Service – Repository), με σαφή διαχωρισμό αρμοδιοτήτων:
    - Controllers για διαχείριση αιτημάτων (UI και REST),
    - Services για την επιχειρησιακή λογική,
    - Repositories για πρόσβαση στη βάση δεδομένων.
    - H2 database
- Για την ενσωμάτωση εξωτερικών υπηρεσιών χρησιμοποιούνται έννοιες Hexagonal Architecture, μέσω Ports & Adapters, ώστε:
    - ο πυρήνας της εφαρμογής να μην εξαρτάται από συγκεκριμένες υλοποιήσεις τρίτων υπηρεσιών,
    - οι εξωτερικές υπηρεσίες να αντιμετωπίζονται ως black boxes.

2. Παραδοχές σχετικά με τους Χρήστες

- Το σύστημα υποστηρίζει αποκλειστικά τους παρακάτω ρόλους:
    - **STUDENT** (φοιτητής),
    - **STAFF** (προσωπικό),
    - **INTEGRATION CLIENT** (εξωτερικά συστήματα).
- Κάθε χρήστης διαθέτει ένα μοναδικό account και δεν υποστηρίζεται κοινή χρήση λογαριασμών.
- Οι φοιτητές θεωρείται ότι:
    - ενεργούν αποκλειστικά για τις δικές τους κρατήσεις,
    - δεν έχουν πρόσβαση σε δεδομένα άλλων φοιτητών.
    - Εχούν ακαδημαϊκό email.
- Το προσωπικό (STAFF):
    - θεωρείται έμπιστο,
    - δεν απαιτείται επιπλέον μηχανισμός έγκρισης (π.χ. double approval) για ενέργειες όπως ακύρωση κρατήσεων ή διαχείριση χώρων.
    - Εχούν ακαδημαϊκό email.
- Οι INTEGRATION CLIENTS εκπροσωπούν μηχανές/συστήματα και όχι φυσικά πρόσωπα.

3. Παραδοχές σχετικά με τις Τεχνολογίες

- Το back-end υλοποιείται με Spring Boot και Spring Framework modules.
- Η αποθήκευση δεδομένων γίνεται σε σχεσιακή βάση H2**.**
- Η πρόσβαση στη βάση δεδομένων υλοποιείται μέσω JPA/Hibernate.
- Γίνεται χρήση Bean Validation τόσο σε επίπεδο DTOs όσο και σε επίπεδο domain, ώστε να διασφαλίζεται η εγκυρότητα των δεδομένων.
- Το Web UI υλοποιείται με:
    - Spring MVC,
    - Thymeleaf,
    - HTML/CSS και JavaScript για δυναμικές λειτουργίες.
- Το REST API:
    - επιστρέφει δεδομένα σε μορφή JSON,
    - είναι σχεδιασμένο ώστε να μπορεί να καταναλωθεί από τρίτα συστήματα (π.χ. άλλα services, SPA ή mobile εφαρμογές).
- Η ασφάλεια υλοποιείται ως εξής:
    - **UI**: stateful authentication με cookies και Spring Security,
    - **REST API**: stateless authentication με JWT tokens,
    - διαχωρισμός ρόλων μέσω Spring Security annotations.
- Η τεκμηρίωση του REST API γίνεται με OpenAPI Specification και παρέχεται μέσω Swagger UI**.**

4.Παραδοχές σχετικά με τις Εξωτερικές Υπηρεσίες

- Το σύστημα ενσωματώνει πολλαπλές εξωτερικές υπηρεσίες μέσω REST calls, οι οποίες χρησιμοποιούνται για:
    - επαλήθευση αριθμών κινητού τηλεφώνου,
    - αποστολή SMS ειδοποιήσεων,
    - έλεγχο εθνικών αργιών.
- Οι εξωτερικές υπηρεσίες αντιμετωπίζονται ως black boxes:
    - το σύστημα γνωρίζει μόνο τα διαθέσιμα endpoints,
    - τα σχήματα των request/response,
- Η ενσωμάτωση των εξωτερικών υπηρεσιών γίνεται μέσω Ports & Adapters, σύμφωνα με τις αρχές της Hexagonal Architecture:
    - το domain και η επιχειρησιακή λογική δεν εξαρτώνται από συγκεκριμένες υλοποιήσεις τρίτων υπηρεσιών,
    - κάθε εξωτερική υπηρεσία υλοποιείται σε ξεχωριστό adapter (PhoneNumberPortImpl, SmsNotificationPortImpl, HolidayPortImpl).

5.Παραδοχή

Επιλέξαμε να δημιουργήσουμε ένα endpoint για εσωτερική κατανάλωση από το web UI της εφαρμογής, το οποίο χρησιμοποιείται αποκλειστικά μέσω AJAX κλήσεων (JavaScript fetch) για την εμφάνιση της διαθεσιμότητας των study rooms στο ημερολόγιο.

Το συγκεκριμένο endpoint δεν αποτελεί μέρος του REST API της εφαρμογής, καθώς δεν προορίζεται για κατανάλωση από εξωτερικούς πελάτες ή τρίτα συστήματα. Για τον λόγο αυτό:

δεν προστατεύεται με JWT stateless authentication,

δεν περιλαμβάνεται στην τεκμηρίωση OpenAPI / Swagger,

και αξιοποιεί την υφιστάμενη stateful, session-based ασφάλεια του UI.

6. Παραδοχή

• Σε περίπτωση όπου μία κράτηση χώρου μελέτης βρίσκεται σε κατάσταση ACTIVE και παρέλθει η ημερομηνία της κράτησης χωρίς να έχει πραγματοποιηθεί ακύρωση από τον φοιτητή ή από το προσωπικό, η κράτηση θεωρείται No-Show.

• Η ανίχνευση του No-Show γίνεται αυτόματα βάσει της ημερομηνίας της κράτησης και της κατάστασής της, χωρίς να απαιτείται χειροκίνητη παρέμβαση από το προσωπικό.

• Σε περίπτωση No-Show, το σύστημα εφαρμόζει ποινή (penalty) στον αντίστοιχο φοιτητή, η οποία συνίσταται σε απαγόρευση δημιουργίας νέων κρατήσεων για χρονικό διάστημα τριών (3) ημερών.

• Η ποινή αποθηκεύεται σε επίπεδο χρήστη (φοιτητή) ως χρονικό όριο (timestamp), μέχρι το οποίο δεν επιτρέπεται η δημιουργία νέων κρατήσεων.

• Κατά τη δημιουργία νέας κράτησης, το σύστημα ελέγχει αν ο φοιτητής βρίσκεται υπό ενεργή ποινή και, σε θετική περίπτωση, απορρίπτει το αίτημα κράτησης.

• Η υλοποίηση της παραπάνω λογικής θεωρείται απλοποιημένη και επαρκής για τους σκοπούς της εργασίας, χωρίς να περιλαμβάνει επιπλέον μηχανισμούς (π.χ. ειδοποιήσεις, σταδιακή κλιμάκωση ποινών).

7. Παραδοχές σχετικά με το Deployment

- Η εφαρμογή εκτελείται:
  - τοπικά (localhost),
  - σε ένα JVM περιβάλλον.
- Δεν γίνεται χρήση:
  - containerization (Docker),
  - orchestration (Kubernetes, Docker Swarm),
  - CI/CD pipeline.