Περιγραφή Λειτουργικότητας

1.Διαχείριση Προσωπικού (Person Service)

1. Person Entity — Η βασική οντότητα χρήστη

Η κλάση Person αναπαριστά έναν εγγεγραμμένο χρήστη του συστήματος (student ή staff).
Αντιστοιχεί άμεσα στον πίνακα person της βάσης και περιλαμβάνει όλα τα στοιχεία ταυτοποίησης και επικοινωνίας.

Κύρια χαρακτηριστικά

- Μοναδικότητες για studentId, email_address, mobile_phone_number ώστε να μην υπάρχουν διπλοεγγραφές.
- Έλεγχος εγκυρότητας μέσω annotations (@NotBlank, @Size, @Email).
- Αυτόματη δημιουργία του createdAt με @CreationTimestamp.
- Το mobilePhoneNumber αποθηκεύεται πάντα σε E.164 format (π.χ. +3069xxxxxxx) ύστερα από έλεγχο της εξωτερικής υπηρεσίας NOC.

2. PersonRepository — Πρόσβαση στη Βάση

Το PersonRepository επεκτείνει το JpaRepository και προσφέρει:

- CRUD λειτουργίες
- Εξειδικευμένους ελέγχους:
  - existsByStudentIdIgnoreCase()
  - existsByEmailAddressIgnoreCase()
  - existsByMobilePhoneNumber()
  - findByEmailAddressIgnoreCase()

Σκοπός: Έγκαιρη ανίχνευση διπλοεγγραφών πριν την αποθήκευση.

3. CreatePersonRequest — Τα δεδομένα της εγγραφής

4. PersonServiceImpl — Η επιχειρησιακή λογική της εγγραφής

Κύρια στάδια

1. Validation δεδομένων
   - Χρήση του Spring Validator.
   - Σε περίπτωση λάθους επιστρέφεται CreatePersonResult.fail().
2. Έλεγχος email
   - Γίνονται δεκτά μόνο ακαδημαϊκά email (*@hua.gr*).
3. Έλεγχος κινητού μέσω εξωτερικής υπηρεσίας NOC
   - Κλήση στο phoneNumberPort.validate().
   - Απορρίπτονται:

        - μη έγκυροι αριθμοί
        - σταθερά τηλέφωνα
    - Σε σωστό mobile, γίνεται αυτόματη μετατροπή σε E.164.
4. Έλεγχος διπλοεγγραφών
   Με χρήση existsBy…() από το repository.
5. Κρυπτογράφηση κωδικού
    - Χρήση passwordEncoder.encode().
    - Χρήση BCryptPasswordEncoder()
    - Ποτέ δεν αποθηκεύεται plaintext password.
6. Δημιουργία και αποθήκευση Person
    - Δημιουργείται νέο entity.
    - Αποθηκεύεται μέσω personRepository.save().
7. Αποστολή SMS μέσω NOC (αν notify=true)
    - smsNotificationPort.sendSms(…)
    - Αν υπάρξει αποτυχία, γίνεται log αλλά η εγγραφή συνεχίζεται.
8. Επιστροφή PersonView
    - Με χρήση του PersonMapper.

5. RegistrationController — Διαχείριση φόρμας εγγραφής

Υλοποιεί τη ροή εγγραφής χρήστη μέσω UI:

- GET /register
    - Προβολή φόρμας εγγραφής
    - Redirect αν ο χρήστης είναι ήδη συνδεδεμένος
- POST /register
    - Εκτελεί backend validation
    - Καλεί το personService.createPerson()
    - Επιτυχής εγγραφή ,κανείredirect στο login
    - Αποτυχία κανεί επιστροφή στη φόρμα με μήνυμα λάθους

6. AuthController & ProfileController — Ροή χρήστη

- Το AuthController χειρίζεται login/logout και redirect ανάλογα με τον ρόλο (student/staff).
- Το ProfileController εμφανίζει τα προσωπικά στοιχεία του συνδεδεμένου χρήστη.

7. PersonMapper — Μετατροπή σε View Object

Μετατρέπει την οντότητα Person σε PersonView, επιστρέφοντας μόνο τα αναγκαία στοιχεία:

2.Διαχείριση Δωματίων Μελέτης (Study Room Service)

Η λειτουργία Study Room Service επιτρέπει στο εξουσιοδοτημένο προσωπικό (STAFF) να:

- δημιουργεί νέα δωμάτια μελέτης,
- ενημερώνει υπάρχοντα δωμάτια,
- προβάλλει όλα τα διαθέσιμα δωμάτια.

1. StudyRoom Entity — Η οντότητα δωματίου

Η οντότητα StudyRoom αντιστοιχεί στο table *study_room* και περιλαμβάνει:

Validation που εφαρμόζεται:

- Υποχρεωτικά όλα τα πεδία (@NotNull, @NotBlank)
- Έλεγχος capacity (@Positive)
- Μοναδικότητα ονόματος μέσω repository
- Οι ώρες λειτουργίας πρέπει να είναι έγκυρες (opening < closing)

Το entity διασφαλίζει ότι κάθε δωμάτιο είναι σωστά ορισμένο πριν αποθηκευτεί.

2. StudyRoomRepository — Το layer πρόσβασης στη βάση

Το repository κληρονομεί από JpaRepository<StudyRoom, Long> και παρέχει:

- CRUD λειτουργίες
- custom μεθόδους:
    - existsByNameIgnoreCase(String name)
    - findById(Long id)

Σκοπός Repository:

- Έλεγχος μοναδικότητας ονόματος
- Εύρεση δωματίων για επεξεργασία

3. DTOs — Τα αντικείμενα εισόδου/εξόδου

Χρησιμοποιούνται ειδικά DTOs για διαχωρισμό entity/UI:

CreateStudyRoomRequest

- name, capacity, openingHour, closingHour
- περιλαμβάνει validation

UpdateStudyRoomRequest

- id, name, capacity, openingHour, closingHour

StudyRoomView

- ασφαλής μορφή για το UI (δεν εκθέτει εσωτερικά πεδία όπως active, createdAt)

CreateStudyRoomResult / UpdateStudyRoomResult

- success flag
- reason message
- επιστρεφόμενο StudyRoomView

4. StudyRoomMapper — Μετατροπή Entity σε View

Ο mapper δημιουργεί ένα StudyRoomView από το entity, ώστε:

- το UI να λαμβάνει μόνο τα απαραίτητα στοιχεία,
- να διατηρείται καθαρός διαχωρισμός domain και παρουσίασης.

5. StudyRoomServiceImpl — Η επιχειρησιακή λογική

Το service είναι υπεύθυνο για όλη τη ροή δημιουργίας και επεξεργασίας δωματίων.

5.1 Δημιουργία δωματίου (createStudyRoom)

Βήματα:

1. **Έλεγχος ρόλου χρήστη**
   Μόνο STAFF επιτρέπεται:
2. if (currentUser.type() != PersonType.STAFF)
3. **Μετατροπή strings σε LocalTime**
   Έλεγχος σωστής μορφής.
4. **Validation κανόνων**
   - **Όνομα μη κενό**
   - **Capacity > 0**
   - **Opening < Closing**
   - **Μοναδικό όνομα μέσω:**
   - existsByNameIgnoreCase(name)
5. **Δημιουργία entity & αποθήκευση**
6. **Επιστροφή αποτέλεσμα σε StudyRoomView**

5.2 Ενημέρωση δωματίου (updateStudyRoom)

- Ανάκτηση δωματίου μέσω repository.
- Έλεγχος capacity και ωρών λειτουργίας.
- Έλεγχος μοναδικότητας ονόματος (αν αλλάζει).
- Αποθήκευση αλλαγών.
- Επιστροφή UpdateStudyRoomResult.

6. UI Controllers — Αλληλεπίδραση STAFF με το σύστημα

Endpoints

- /room/create (GET/POST) → δημιουργία δωματίου
- /room/list → προβολή όλων των δωματίων
- /room/edit/{id} (GET/POST) → επεξεργασία δωματίου

7. CurrentUserProvider — Έλεγχος πρόσβασης

Το service χρησιμοποιεί:

currentUserProvider.requireCurrentUser()

Έτσι γνωρίζει:

- ποιος χρήστης κάνει την ενέργεια,
- τον ρόλο του (STUDENT/STAFF).

Χωρίς STAFF δεν επιτρέπεται καμία τροποποίηση δωματίων.

3. Διαθεσιμότητα Δωματίων Μελέτης (Room Availability Service)

Η λειτουργία Room Availability Service υπολογίζει πόσες θέσεις είναι διαθέσιμες σε κάθε δωμάτιο μελέτης για μια συγκεκριμένη ημερομηνία.
Επιτρέπει στους φοιτητές να βλέπουν σε πραγματικό χρόνο τη διαθεσιμότητα και να προχωρούν σε κράτηση.

Η υλοποίηση βασίζεται σε τρία επίπεδα:

- REST API για επικοινωνία με το frontend (FullCalendar)
- Service Layer για τους κανόνες υπολογισμού
- UI Layer για την παρουσίαση (Thymeleaf + FullCalendar)

1. RoomsAvailabilityRestController — Το REST API

Ο controller εκθέτει το endpoint:

GET /api/rooms/availability?date=YYYY-MM-DD

Λειτουργία:

- λαμβάνει την ημερομηνία ως string,
- τη μετατρέπει σε LocalDate,
- Επιστρεφει κενή λίστα σε ημερομηνίες πριν την σημερινή
- καλεί το AvailabilityService,
- επιστρέφει JSON με τη διαθεσιμότητα όλων των δωματίων.
- Αυτό διαβάζεται από το FullCalendar για εμφάνιση στο ημερολόγιο.

2. AvailabilityServiceImpl — Υπολογισμός διαθέσιμων θέσεων

Το service υλοποιεί όλη τη λογική υπολογισμού:

1. Ανάκτηση όλων των δωματίων
   studyRoomRepository.findAll()
2. Αποκλεισμός δωματίων με capacity ≤ 0
3. Ανάκτηση ενεργών κρατήσεων για την επιλεγμένη ημέρα
   findByRoomIdAndDateAndStatus(..., ACTIVE)
4. Υπολογισμός δεσμευμένων θέσεων
5. seatsBooked = sum(seatsReserved)
6. Υπολογισμός διαθέσιμων
7. available = capacity - seatsBooked
8. Μετατροπή σε DTO μέσω mapper

Το αποτέλεσμα είναι λίστα RoomAvailabilityView, ένα για κάθε δωμάτιο.

3. AvailabilityMapper — Μετατροπή σε DTO

mapper εξασφαλίζει:

- ότι δεν εκτίθεται ολόκληρο το StudyRoom entity,
- ότι οι ημερομηνίες συνδυάζονται με τις ώρες λειτουργίας,
- καθαρή μορφή δεδομένων για το frontend.

4. RoomAvailabilityView — DTO για το UI

Το DTO περιέχει μόνο τα στοιχεία που χρειάζεται το ημερολόγιο:

- ID και όνομα δωματίου
- συνολική χωρητικότητα
- διαθέσιμες θέσεις
- start/end ώρα για την επιλεγμένη ημερομηνία

Αποτελεί ασφαλή μορφή δεδομένων που δεν εξαρτάται από το entity.

5. StudyRoomAvailabilityController — Το UI για φοιτητές

Το endpoint:

```
GET /room/calendar
```

- απαιτεί ρόλο STUDENT,
- φορτώνει την τρέχουσα ημερομηνία,
- καλεί την υπηρεσία διαθεσιμότητας,
- εμφανίζει το FullCalendar.

Οι φοιτητές βλέπουν δυναμικά τη διαθεσιμότητα και επιλέγουν ημέρα/δωμάτιο για κράτηση.

6. Frontend — Calendar με FullCalendar

Το frontend:

- φορτώνει το FullCalendar JS library
- όταν ο φοιτητής πατήσει μια ημέρα, εκτελεί:

```
fetch("/api/rooms/availability?date=YYYY-MM-DD")
```

- δημιουργεί events ανά δωμάτιο,
- τα χρωματίζει:
    - πράσινο για διαθέσιμες θέσεις,
    - κόκκινο για πλήρη δωμάτια.

Με αυτόν τον τρόπο ο φοιτητής ενημερώνεται άμεσα για τη διαθεσιμότητα.

4. Διαχείριση Κρατήσεων Φοιτητή (Create / Cancel / History) –

Η λειτουργία διαχείρισης κρατήσεων επιτρέπει στον φοιτητή να δημιουργεί κράτηση, να ακυρώνει μια υπάρχουσα και να προβάλλει το πλήρες ιστορικό του.

1. Reservation Entity

Το Reservation αποτελεί το domain model μιας κράτησης.

Είναι η βασική οντότητα για όλες τις λειτουργίες κρατήσεων.

2. ReservationRepository

Παρέχει εξειδικευμένα queries:

- Ιστορικό φοιτητή
- Ενεργές μελλοντικές κρατήσεις:
- Έλεγχος ορίου 4 κρατήσεων/ημέρα:

- Έλεγχος διαθεσιμότητας θέσεων:

3. ReservationServiceImpl

A. Δημιουργία κράτησης (bookReservation)

Έλεγχοι πριν τη δημιουργία:

1. Ο χρήστης πρέπει να είναι φοιτητής
2. Το δωμάτιο πρέπει να υπάρχει
3. Η κρατηση θα πρέπει να είναι τωρινή ή αργότερη ημερομηνία
4. Όριο 4 κρατήσεων την ίδια ημέρα
5. Έλεγχος διαθέσιμων θέσεων
   Συνολικές δεσμευμένες θέσεις < χωρητικότητα δωματίου
6. Δημιουργία και αποθήκευση της κράτησης

Επιστρέφεται ReservationView.

B. Ακύρωση κράτησης (cancelReservation)

Έλεγχοι:

- Η κράτηση ανήκει στον φοιτητή
- Η κράτηση είναι ακόμη ACTIVE
- Η ημερομηνία της κράτησης πρέπει να είναι μετά τη σημερινή.
- Ενημερώνεται το status CANCELLED και καταχωρείται cancelledAt.

C. Ιστορικό κρατήσεων

Με βάση το query:

findByStudentIdOrderByDateDesc

Περιλαμβάνει ενεργές, ακυρωμένες και ακυρωμένες από προσωπικό.

4. ReservationView & Mapper

Το ReservationView είναι το DTO που βλέπει το UI .

5. UI Controllers

1. Προβολή ενεργών κρατήσεων

GET /rooms/reservations/my
Εμφανίζει μόνο τις ACTIVE κρατήσεις από σήμερα και μετά.

2. Ακύρωση κράτησης

POST /rooms/reservations/cancel
Δέχεται reservationId , καλεί το service , δείχνει σελίδα επιτυχίας.

3. Ιστορικό

GET /rooms/reservations/history
Πλήρης λίστα όλων των κρατήσεων του φοιτητή.

6. UI Views (Thymeleaf)

- my_reservations.html — ενεργές κρατήσεις + δυνατότητα ακύρωσης
- reservation_success.html — μήνυμα επιτυχούς κράτησης
- reservation_cancel_success.html — μήνυμα επιτυχούς ακύρωσης
- history.html — πλήρες ιστορικό με χρωματική ένδειξη κατάστασης

5.Ακύρωση Κρατήσεων από το Προσωπικό

Η λειτουργία αυτή επιτρέπει στο προσωπικό (STAFF) να ακυρώνει ενεργές κρατήσεις φοιτητών. Χρησιμοποιείται συνήθως σε περιπτώσεις παραβάσεων, κλεισίματος χώρων, ή ανάγκης διαχείρισης των διαθέσιμων θέσεων.

1. ReservationStatus – Κατάσταση Κράτησης

Η επιπλέον κατάσταση:

- CANCELLED_BY_STAFF
  δηλώνει ότι η ακύρωση έγινε από εξουσιοδοτημένο μέλος του προσωπικού και όχι από τον φοιτητή.

2. Επιχειρησιακή Λογική (ReservationServiceImpl)

Η μέθοδος cancelReservationByStaff() εφαρμόζει όλα τα απαραίτητα validation βήματα:

Βήματα ακύρωσης:

1. Έλεγχος ρόλου χρήστη
   Μόνο STAFF έχει δικαίωμα ακύρωσης:

2. Έλεγχος ύπαρξης κράτησης
   Ανάκτηση μέσω repository:

3. Ακύρωση μόνο ACTIVE κρατήσεων
   Δεν επιτρέπεται ακύρωση ήδη ακυρωμένης κράτησης.
4. Ενημέρωση κατάστασης

```
reservation.setStatus(CANCELLED_BY_STAFF);
reservation.setCancelledAt(Instant.now());
```

5. Επιστροφή ReservationView
   μέσω του ReservationMapper.

3. StaffBookingController – UI για το Προσωπικό

Ο controller παρέχει δύο λειτουργίες:

1. Προβολή ενεργών κρατήσεων προς ακύρωση

GET /bookings/manage
Φορτώνει όλες τις ACTIVE κρατήσεις.

2. Εκτέλεση ακύρωσης

POST /bookings/cancel
Στέλνει reservationId ,καλεί το service , εμφανίζει σελίδα επιτυχίας.

Και τα δύο endpoints είναι προστατευμένα με:

```
@PreAuthorize("hasRole('STAFF')")
```

4. UI Views (Thymeleaf)

- manage_bookings.html
  Πίνακας με όλες τις ενεργές κρατήσεις και κουμπί *Ακύρωση*.
- reservation_cancelled_by_staff.html
  Σελίδα επιβεβαίωσης της ακύρωσης από το προσωπικό.