ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

Έξυπνο Σύστημα Ανακοίνωσης Συνεδρίων / CFPs

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Γεώργιου Ζεϊμπέκη

Επιβλέπων: Κυριάκος Κρητικός
Μέλη εξεταστικής επιτροπής: Κρητικός Κυριάκος, Συμεωνίδης Παναγιώτης, Κωστούλας Θεόδωρος

Πίνακας Περιεχομένων

Περίληψη **4**
Abstract **5**
Κατάλογος Εικόνων **6**
Κατάλογος Πινάκων **9**
Κατάλογος Παραθεμάτων Κώδικα **10**
1. Εισαγωγή **11**
1.1 Ανάγκη για ένα Έξυπνο Σύστημα Διαχείρισης CFP 11
1.2 Συνεισφορά Διπλωματικής 11
1.3 Οργάνωση Αναφοράς 12
2. Υπόβαθρο **13**
3. Σχετικές Εργασίες **16**
3.1 Silverfish 16
3.2 CIERS 16
3.3 ConfSys 16
3.4 OpenResearch 17
4. Ανάπτυξη Συστήματος **19**
4.1 Ανάλυση Απαιτήσεων **19**
4.1.1 Λειτουργικές Απαιτήσεις 19
4.1.1.1 Λειτουργίες Διαχείρισης Χρηστών 20
4.1.1.2 Λειτουργίες Διαχείρισης Συνεδρίων 21
4.1.2 Λειτουργίες Διαχείρισης Θεμάτων 24
4.1.3 Λειτουργίες Κοινωνικής Δικτύωσης 25
4.1.4 Μη Λειτουργικές Απαιτήσεις 26
4.2 Σχεδίαση **26**
4.2.1 Διάγραμμα Περιβάλλοντος (Context Diagram) 27
4.2.2 Διάγραμμα Συστατικών (Component Diagram) 28
4.2.3 Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram) 31
Διάγραμμα Διαχείρισης Συνεδρίων 33
Διάγραμμα Διαχείρισης Λογαριασμού Χρήστη 34
Διάγραμμα Κοινωνικής Δικτύωσης Χρηστών 35
Διάγραμμα Διαχείρισης Θεμάτων 36
4.2.4 Διάγραμμα Ακολουθίας (Sequence Diagram) 36
4.2.4.1 Συνέδρια 36
4.2.4.2 Χρήστες 38
4.2.4.3 Θέματα 40
4.2.4.4 Social Media 42
4.2.5 Διάγραμμα Δραστηριοτήτων (Activity Diagram) 44
4.2.5.1 Συνέδριο 44
4.2.5.2 Χρήστες 46
4.2.5.3 Θέματα 48
4.2.5.4 Social Media 50
4.2.6 Διάγραμμα Κλάσεων (Class Diagram) 52

4.3 Υλοποίηση..... 58
4.3.1 Γλώσσα Προγραμματισμού και Τεχνολογίες..... 58
4.3.1.1 Frontend (Περιβάλλον Χρήστη)..... 58
4.3.1.2 Backend (Λογική Εφαρμογής)..... 59
4.3.1.3 Database Layer (Βάση Δεδομένων)..... 60
4.3.1.4 Εξωτερικές Υπηρεσίες & APIs..... 60
4.3.2 Δομή Κώδικα..... 61
4.3.2.1 Δομή UI (Frontend)..... 61
4.3.2.2 Δομή Backend..... 62
4.3.2.3 Κλάσεις/Μοντέλα Τομέα (Domain Models)..... 63
4.2.2.4 Κλάσεις Backend (Business Logic & API Controllers)..... 63
4.4 Δοκιμή του Συστήματος..... 64
4.4.1 Εισαγωγή..... 64
4.4.2 Δοκιμές Μονάδων..... 66
4.4.3 Integration Testing (Δοκιμές Ενοποίησης)..... 75
4.4.4 UI Testing (Δοκιμές Frontend)..... 77
4.5 Βαθμός Ικανοποίησης Απαιτήσεων..... 77
5. Εγκατάσταση & Επίδειξη Συστήματος..... 82
5.1 Εγκατάσταση..... 82
5.1.1 Προαπαιτούμενα..... 82
5.1.2 Οδηγίες Εγκατάστασης και Εκτέλεσης..... 83
5.2 Επίδειξη..... 83
6. Συμπεράσματα & Μελλοντική Εργασία..... 93
Βιβλιογραφία..... 94

Περίληψη

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη ενός ολοκληρωμένου backend συστήματος ιστού (web system) για τη διαχείριση και προώθηση επιστημονικών συνεδρίων, με κύριο άξονα τις προσκλήσεις υποβολής εργασιών (Call for Papers - CFP). Το σύστημα παρέχει λειτουργίες, όπως η δημιουργία, ενημέρωση και αναζήτηση συνεδρίων και θεματικών περιοχών, η διαχείριση αιτημάτων φιλίας και κοινωνικής δικτύωσης μεταξύ των χρηστών, καθώς και η αξιολόγηση συνεδρίων μέσω συγκεκριμένων κριτηρίων. Βασίζεται σε τεχνολογίες Java και Spring Boot, εξασφαλίζοντας καθαρή αρχιτεκτονική, ασφάλεια και επεκτασιμότητα. Η υλοποίηση καλύπτει πλήρως το backend, ενώ το frontend της οραματισμένης εφαρμογής ιστού (web application) (δηλ. του συστήματος) παραμένει σε αρχικό στάδιο λόγω της πολυπλοκότητας του όλου έργου λογισμικού που δεν είναι δυνατό να ολοκληρωθεί πλήρως στο πλαίσιο μιας και μόνο διπλωματικής εργασίας. Η εργασία τεκμηριώνει την ανάλυση απαιτήσεων, το σχεδιασμό, την ανάπτυξη και τον έλεγχο της εφαρμογής, θέτοντας μια στιβαρή βάση για μελλοντική επέκταση και βελτίωση. Η συμβολή της εργασίας έγκειται στην παροχή ενός ευέλικτου, ασφαλούς και επεκτάσιμου backend που υποστηρίζει την αποτελεσματική διαχείριση επιστημονικών εκδηλώσεων.

Λέξεις-κλειδιά: Διαχείριση επιστημονικών συνεδρίων, Πρόσκληση για υποβολή άρθρων (Call for Papers - CFP), Ανάπτυξη Backend, Spring Boot, Κοινωνική δικτύωση, Ασφάλεια εφαρμογών, Αξιολόγηση συνεδρίων, RESTful API, Τεκμηρίωση λογισμικού

Abstract

This thesis focuses on the development of a comprehensive web-based backend system for managing and promoting scientific conferences, with a primary emphasis on Call for Papers (CFP). The system offers features, such as creating, updating, and searching conferences and thematic topics, managing friendship requests and social networking among users, and evaluating conferences based on specific criteria. Built with Java and Spring Boot technologies, it ensures a clean architecture, security, and scalability. The implementation fully covers the backend, while the frontend of the envisioned web application (i.e., of the web-based system) remains at an early development stage due to the overall software project's complexity that makes it impossible to fully implement this project in the context of a single thesis. This work documents the requirements analysis, design, development, and testing of the application, providing a solid foundation for future extension and improvement. The contribution lies in delivering a flexible, secure, and scalable backend supporting effective scientific event management.

Keywords: Scientific conference management, Call for Papers (CFP), Backend development, Spring Boot, Social networking, Application security, Conference evaluation, RESTful API, Software documentation

Κατάλογος Εικόνων

Αρ.	Τίτλος Εικόνας	Σελίδα
1	Αρχιτεκτονική Τυπικού Πληροφοριακού Συστήματος	16
2	Διάγραμμα Περιβάλλοντος του Συστήματος	29
3	Διάγραμμα Συστατικών του Συστήματος	31
4	Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Συνεδρίων	34
5	Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Λογαριασμού Χρήστη	35
6	Διάγραμμα Περίπτωσης Χρήσης Κοινωνικής Δικτύωσης Χρηστών	36
7	Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Θεμάτων	37
8	Διάγραμμα Ακολουθίας Δημιουργίας Νέου Συνεδρίου	38
9	Διάγραμμα Ακολουθίας Εγγραφής Ερευνητή σε Θέμα	39
10	Διάγραμμα Ακολουθίας Εισόδου Χρήστη και Αυθεντικοποίησης	40
11	Διάγραμμα Ακολουθίας Εγγραφής Χρήστη	41
12	Διάγραμμα Ακολουθίας Εισαγωγής Νέου Θέματος	42
13	Διάγραμμα Ακολουθίας Διαγραφής Θέματος	43
14	Διάγραμμα Ακολουθίας Βαθμολόγησης Συνεδρίου	44
15	Διάγραμμα Ακολουθίας Αποστολής Αιτήματος Φιλίας	45
16	Διάγραμμα Δραστηριοτήτων Δημοσίευσης Συνεδρίου από Επιμελητή	46
17	Διάγραμμα Δραστηριοτήτων Αυτόματης Ενημέρωσης Συνεδρίων μέσω Crawling	47
18	Διάγραμμα Δραστηριοτήτων Διαγραφής Λογαριασμού Χρήστη	48
19	Διάγραμμα Δραστηριοτήτων Επαναφοράς Κωδικού Χρήστη	49
20	Διάγραμμα Δραστηριοτήτων Συσχέτισης Θέματος με Συνέδριο από Επιμελητή	50
21	Διάγραμμα Δραστηριοτήτων Θέασης Θέματος	51
22	Διάγραμμα Δραστηριοτήτων Αποδοχής ή Απόρριψης Αιτήματος Φιλίας	52
23	Διάγραμμα Δραστηριοτήτων Δημιουργίας Νέου Thread σε Συνέδριο ή Θέμα	53

24	Διάγραμμα Κλάσεων Συστήματος	58
25	Ιεραρχική δομή React σελίδων UI	63
26	Δομή φακέλων backend εφαρμογής Περιλαμβάνονται οι φάκελοι config, controllers, services, repositories, models, dto, utils, καθώς και ο φάκελος resources με τα αρχεία ρυθμίσεων και στατικών πόρων	64
27	Αίτημα εγγραφής χρήστη μέσω του endpoint /api/users με βασικά στοιχεία εγγραφής	85
28	Επιτυχής είσοδος χρήστη μέσω του endpoint /api/auth/login και παραλαβή JWT token για ασφαλή πρόσβαση	86
29	Χρήση του JWT token στο κουμπί Authorize του Swagger UI για αυθεντικοποιημένη πρόσβαση στα endpoints	86
30	Εμφάνιση των στοιχείων χρήστη με βάση το username, μέσω του endpoint /api/users/{username}. Το σύστημα επιστρέφει το προφίλ χωρίς τον κωδικό πρόσβασης	87
31	Ενημέρωση των στοιχείων χρήστη μέσω του endpoint /api/users/{id} με PUT αίτηση	87
32	Αυτοδιαγραφή λογαριασμού ερευνητή μέσω του endpoint /api/users/me με DELETE αίτηση	88
33	Αίτημα δημιουργίας νέου συνεδρίου μέσω του endpoint /api/conferences. Ο χρήστης στέλνει όλα τα απαραίτητα στοιχεία του συνεδρίου (τίτλος, περιγραφή, τύπος κλπ)	89
34	Ενημέρωση στοιχείων συνεδρίου μέσω του endpoint /api/conferences/{id} με PUT αίτηση	89
35	Ενημέρωση ενός στοιχείου συνεδρίου μέσω του endpoint /api/conferences/{id} με PATCH αίτηση	90
36	Παράδειγμα POST αίτησης για τη δημιουργία νέου θέματος (topic) μέσω του endpoint /api/topics	91
37	Παράδειγμα αποστολής POST αιτήματος στο endpoint /api/topics/{topicId}/conferences/{conferenceId} για τη σύνδεση ενός υπάρχοντος θέματος με ένα συνέδριο	91
38	Παράδειγμα αποστολής POST αιτήματος για την εγγραφή (subscribe) ενός χρήστη σε συγκεκριμένο θέμα (topic)	92

39 Παράδειγμα GET αιτήματος στο endpoint /api/users/search για την αναζήτηση χρηστών με βάση το ονοματεπώνυμο ή το username 92

40 Παράδειγμα PUT αιτήματος στο endpoint /api/users/{id} όπου ένας επιμελητής μπορεί να ενημερώσει τα στοιχεία ενός χρήστη 93

41 Παράδειγμα DELETE αιτήματος στο endpoint /api/users/{id} όπου ένας επιμελητής μπορεί να διαγράψει έναν ερευνητή 93

Κατάλογος Πινάκων

Αρ.	Τίτλος Πίνακα	Σελίδα
1	Σύγκριση Σχετικών Συστημάτων	19
2	Βαθμός Ικανοποίησης Λειτουργικών Απαιτήσεων	79
3	Βαθμός Ικανοποίησης Μη Λειτουργικών Απαιτήσεων	81

Κατάλογος Παραθεμάτων Κώδικα

Αρ.	Τίτλος Παραθέματος	Σελίδα
1	Μονάδα ελέγχου του repository `UserRepository` με χρήση JUnit και `@DataJpaTest`, η οποία επαληθεύει τη σωστή λειτουργία της μεθόδου `findByEmail` βάσει καταχωρημένου χρήστη.	67
2	Μονάδα ελέγχου της υπηρεσίας `UserServiceImpl` με χρήση Mockito, η οποία ελέγχει ότι η μέθοδος `createUser` αποθηκεύει και επιστρέφει σωστά έναν νέο χρήστη μέσω του `UserRepository`.	69
3	Έλεγχος του REST endpoint `/api/users` του `UserController` με χρήση `MockMvc`, ο οποίος επαληθεύει την επιτυχή δημιουργία χρήστη και την ορθότητα των επιστρεφόμενων JSON πεδίων.	71
4	Έλεγχος των βασικών λειτουργιών δημιουργίας και επαλήθευσης JWT tokens με χρήση της βιβλιοθήκης `jjwt`. Περιλαμβάνει δοκιμές για την εγκυρότητα του `subject`, των χρονικών σφραγίδων και την ανίχνευση ληγμένων tokens.	73
5	Έλεγχος της σωστής φόρτωσης του bean `SecurityConfig` και της ύπαρξης του `SecurityFilterChain` στο Spring Security context. Περιλαμβάνει και έλεγχο για απενεργοποίηση του μηχανισμού CSRF.	75
6	Δοκιμή ενοποίησης με χρήση MockMvc για την επαλήθευση του endpoint δημιουργίας χρήστη μέσω POST.	77

1. Εισαγωγή

1.1 Ανάγκη για ένα Έξυπνο Σύστημα Διαχείρισης CFP

Η ταχεία ανάπτυξη των τεχνολογιών πληροφορικής και επικοινωνιών έχει αλλάξει ριζικά τον τρόπο με τον οποίο οι επιστημονικές κοινότητες οργανώνουν, προωθούν και συμμετέχουν σε συνέδρια. Τα επιστημονικά συνέδρια αποτελούν σημαντικό χώρο ανταλλαγής γνώσεων, δικτύωσης και παρουσίασης νέων ερευνητικών ευρημάτων. Παράλληλα, οι προσκλήσεις για υποβολή άρθρων (*Call for Papers* - CFP) λειτουργούν ως οι κύριοι μηχανισμοί διάδοσης των επιστημονικών εκδηλώσεων, παρέχοντας πληροφορίες σχετικά με θεματολογίες, προθεσμίες και διαδικασίες υποβολής. Ωστόσο, με την εκθετική αύξηση του αριθμού των συνεδρίων και των προσκλήσεων, έχει καταστεί εξαιρετικά δύσκολο για τους ερευνητές και ακαδημαϊκούς να παρακολουθούν όλες τις σχετικές ευκαιρίες συμμετοχής και να επιλέγουν αυτές που ταιριάζουν καλύτερα στα ενδιαφέροντά τους.

Η ανάγκη για ένα έξυπνο και αυτοματοποιημένο σύστημα διαχείρισης και προώθησης επιστημονικών συνεδρίων είναι πλέον επιτακτική. Τα υπάρχοντα συστήματα είτε βασίζονται σε απλές βάσεις δεδομένων με χειροκίνητη καταχώρηση πληροφοριών είτε αξιοποιούν περιορισμένες δυνατότητες αυτόματης εξαγωγής και επεξεργασίας σχετικών δεδομένων. Ουσιαστικά, η αποτελεσματική διαχείριση των CFP απαιτεί όχι μόνο τη αυτοματοποιημένη συλλογή και ανάλυση των δεδομένων, αλλά και την έξυπνη αντιστοίχιση των συνεδρίων με τα ενδιαφέροντα των χρηστών, την προσωποποιημένη προώθηση των σχετικών εκδηλώσεων, καθώς και τη δημιουργία ενός διαδραστικού περιβάλλοντος επικοινωνίας και ανταλλαγής απόψεων μεταξύ ερευνητών και διοργανωτών.

1.2 Συνεισφορά Διπλωματικής

Η προτεινόμενη εργασία συμβάλλει στην **αποτελεσματικότερη διαχείριση και αναζήτηση επιστημονικών συνεδρίων**, αντιμετωπίζοντας την **κατακερματισμένη πληροφόρηση, την μη αποτελεσματική αναζήτηση και την έλλειψη εξατομικευμένης ενημέρωσης**. Μέσω **προσωποποιημένων προτάσεων και στοχευμένων ειδοποιήσεων**, το προτεινόμενο σύστημα επιτρέπει στους ερευνητές να εντοπίζουν γρήγορα συνέδρια που τους ενδιαφέρουν, εξαλείφοντας την ανάγκη για χειροκίνητη αναζήτηση σε πολλαπλές πηγές.

Για την επίτευξη αυτών των στόχων, η προτεινόμενη λύση παρέχει δυνατότητες κοινωνικής δικτύωσης, όπως σχολιασμός και αξιολόγηση συνεδρίων, καθώς και βασική υποστήριξη για τη διαχείριση χρηστών, συνεδρίων και θεμάτων μέσω RESTful API. Επιπλέον, η εφαρμογή υλοποιήθηκε με αρθρωτή (modular) αρχιτεκτονική και διαχωρισμό μεταξύ frontend και backend, επιτρέποντας ευκολία στην ανάπτυξη και επεκτασιμότητα.

Πλεονεκτήματα της Εφαρμογής:
Το σύστημα παρέχει πολλαπλά οφέλη τόσο για τους διοργανωτές όσο και για τους συμμετέχοντες:

1. **Προσωποποιημένες Προτάσεις:** Με βάση τα θέματα ενδιαφέροντος που έχει δηλώσει ο κάθε εγγεγραμμένος χρήστης του συστήματος.

2. **Δυνατότητες Κοινωνικής Δικτύωσης:** Σχολιασμός, αιτήματα φιλίας, και αξιολόγηση συνεδρίων.
3. **Ευκολία Κλιμάκωσης και Ανάπτυξης:** Λόγω της διαχωρισμένης αρχιτεκτονικής, της παροχής REST API και της δοχειοποίησής της (containerisation), η εφαρμογή μπορεί να επεκταθεί και κλιμακωθεί εύκολα σε σύγχρονα περιβάλλοντα υπολογιστικού νέφους (cloud computing).
4. **Ασφαλής Διαχείριση:** Χρήση JWT Authentication και Role-Based Access Control (RBAC) για προστασία των βασικών πληροφοριακών οντοτήτων που χειρίζεται το σύστημα.

Η υλοποίηση ενός τέτοιου συστήματος θα προσφέρει μια καινοτόμο λύση για τη διαχείριση επιστημονικών συνεδρίων, βελτιώνοντας τη διαδικασία ανακάλυψης και προώθησης CFP και ενισχύοντας τη διαδραστικότητα μεταξύ διοργανωτών και συμμετεχόντων.

1.3 Οργάνωση Αναφοράς

Η εργασία χωρίζεται στα εξής επόμενα κεφάλαια:

- **Υπόβαθρο:** Παρουσιάζεται η έννοια ενός έξυπνου συστήματος ανακοίνωσης συνεδρίων, περιγράφοντας τις βασικές αρχές λειτουργίας του, καθώς και τις κύριες δυνατότητες που προσφέρει, όπως αυτοματοποιημένη συλλογή δεδομένων, προσωποποιημένες ειδοποιήσεις και έξυπνη αναζήτηση συνεδρίων.
- **Σχετικές Εργασίες:** Ανάλυση υπαρχόντων συστημάτων και σύγκριση με το προτεινόμενο σύστημα.
- **Ανάπτυξη Συστήματος:** Προσδιορίζεται η διαδικασία ανάπτυξης του συστήματος και τα κύρια αποτελέσματά της.
- **Εγκατάσταση & Επίδειξη Συστήματος:** παρέχονται οδηγίες εγκατάστασης καθώς και γίνεται επίδειξη του συστήματος με βάση ορισμένα σενάρια χρήσης.
- **Συμπεράσματα και Μελλοντική Εργασία:** Ανακεφαλαίωση βασικής συνεισφοράς διπλωματικής εργασίας καθώς και προτάσεις για περαιτέρω βελτίωση ή επέκταση του ανεπτυγμένου συστήματος.

2. Υπόβαθρο

Ένα έξυπνο σύστημα ανακοίνωσης συνεδρίων είναι μια προηγμένη διαδικτυακή εφαρμογή που στοχεύει στη συγκέντρωση, διαχείριση και παρουσίαση πληροφοριών σχετικά με επιστημονικά συνέδρια και προσκλήσεις για υποβολή άρθρων (Call for Papers - CFP). Η ανάγκη για τέτοια συστήματα έχει αναδειχθεί λόγω του μεγάλου όγκου συνεδρίων που διεξάγονται παγκοσμίως, καθιστώντας δύσκολη την παρακολούθηση των σχετικών γεγονότων από ερευνητές, ακαδημαϊκούς και διοργανωτές. Ερευνητές και διοργανωτές συνεδρίων έρχονται αντιμέτωποι με προβλήματα, όπως η πληθώρα ανακοινώσεων σε διαφορετικές πλατφόρμες, η αναποτελεσματική κατηγοριοποίηση των CFP και η αδυναμία έγκαιρης ενημέρωσης για σημαντικές προθεσμίες και αλλαγές προγραμμάτων, γεγονός που οδηγεί σε απώλεια σημαντικών ευκαιριών. Η ύπαρξη ενός μηχανισμού αυτοματοποιημένων ειδοποιήσεων και προσωποποιημένων ενημερώσεων μπορεί να συμβάλλει καθοριστικά στη διευκόλυνση των χρηστών, παρέχοντας στοχευμένες πληροφορίες με βάση τα ερευνητικά τους ενδιαφέροντα και τις προτιμήσεις τους. [3] [7].

Η **"έξυπνη" φύση** των προαναφερόμενων συστημάτων βασίζεται στην αυτοματοποιημένη επεξεργασία δεδομένων και στην προσφορά προσωποποιημένων υπηρεσιών που εξοικονομούν χρόνο και βελτιώνουν τη συνολική εμπειρία των χρηστών. Αυτά τα συστήματα αξιοποιούν τεχνολογίες **εξόρυξης δεδομένων (Data Mining)** και **επεξεργασίας φυσικής γλώσσας (NLP)** για την εξαγωγή και κατηγοριοποίηση περιεχομένου από μη δομημένα δεδομένα που βρίσκονται σε ιστοσελίδες (πχ. συνεδρίων) και πλατφόρμες (όπως κλασικά συστήματα ανακοίνωσης συνεδρίων). Παράλληλα, μέσω τεχνικών μηχανικής μάθησης (*Machine Learning*), τα συστήματα αυτά μπορούν να αναλύσουν τη συμπεριφορά των χρηστών και να προσφέρουν εξατομικευμένες προτάσεις βασισμένες στο προφίλ, ιστορικό και ενδιαφέροντά τους.

Τα συστήματα αυτά προσφέρουν πλήθος λειτουργικοτήτων, όπως:

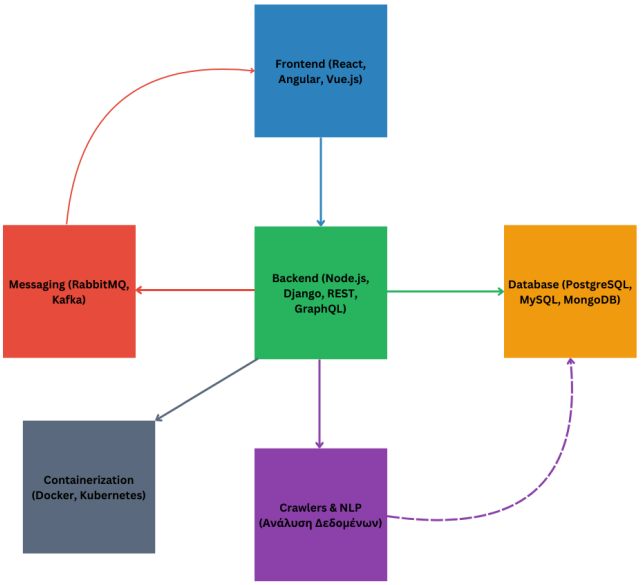
- **Έξυπνη αναζήτηση συνεδρίων**, επιτρέποντας στους χρήστες να φιλτράρουν αποτελέσματα βάσει θεματολογίας, τοποθεσίας, ημερομηνιών και προτιμήσεων.
- **Αυτόματη άντληση και ανανέωση δεδομένων (crawling & updating)** από επίσημες ιστοσελίδες και πλατφόρμες CFP, εξασφαλίζοντας επικαιροποιημένες πληροφορίες για νέα και τροποποιημένα συνέδρια.
- **Προσωποποιημένες προτάσεις συνεδρίων**, αξιοποιώντας τεχνολογίες ανάλυσης προφίλ χρηστών και μηχανισμούς αντιστοίχισης ερευνητικών ενδιαφερόντων.
- **Ανάλυση τάσεων και θεμάτων:** Μέσω τεχνικών ανάλυσης δεδομένων, τα συστήματα μπορούν να εντοπίζουν αναδυόμενες ερευνητικές τάσεις από τα θέματα που εμφανίζονται συχνά σε CFP [7].
- **Κοινωνική Δικτύωση:** Υποστήριξη λειτουργιών, όπως η αξιολόγηση, ο σχολιασμός και η ανταλλαγή απόψεων μεταξύ χρηστών, καθώς και η δυνατότητα δημιουργίας θεματικών φόρουμ για συνεργασία και δικτύωση [3].
- **Ειδοποιήσεις σε πραγματικό χρόνο:** Χρήση μηχανισμών *Publish/Subscribe* για ενημερώσεις σε πραγματικό χρόνο σχετικά με νέες ανακοινώσεις, αλλαγές ή κρίσιμες ημερομηνίες.

Παράλληλα, οι **διοργανωτές συνεδρίων** έχουν στη διάθεσή τους έξτρα εργαλεία (διαχείρισης συνεδρίων) για:

- **Διαχείριση καταχωρήσεων CFP.**
- **Συλλογή και ανάλυση στατιστικών στοιχείων** σχετικά με τη δημοτικότητα και τη συμμετοχή στα συνέδρια [3].
- **Αυτοματοποιημένη αντιστοίχιση κριτών με υποβληθέντα άρθρα,** χρησιμοποιώντας τεχνολογίες NLP και μηχανικής μάθησης.

Μια κοινή **αρχιτεκτονική** για τέτοια συστήματα περιλαμβάνει:

1. **Επίπεδο Παρουσίασης (Frontend):** Χρήση σύγχρονων τεχνολογιών όπως **React**, **Angular** και **Vue.js** για τη δημιουργία ευέλικτων, λειτουργικών και αποκριτικών (responsive) διεπαφών χρήστη (user interfaces).
2. **Επίπεδο Λειτουργικής Λογικής (Backend):** APIs (**REST** ή **GraphQL**) και και πλαίσια ανάπτυξης (frameworks), όπως **Node.js** ή **Django**, προσφέρονται και χρησιμοποιούνται, αντίστοιχα, για τη διαχείριση αιτημάτων και την εκτέλεση της κύριας λογικής της εφαρμογής / συστήματος.
3. **Επίπεδο Αποθήκευσης Δεδομένων:** Χρήση σχεσιακών βάσεων δεδομένων (**PostgreSQL**, **MySQL**) ή NoSQL βάσεων, όπως **MongoDB**, για την αποθήκευση δομημένων και μη δομημένων δεδομένων, αντίστοιχα.
4. Οι μηχανισμοί **Αυτοματοποίησης & Ανάλυσης Δεδομένων:** Crawlers και εργαλεία ανάλυσης NLP εξάγουν και κατηγοριοποιούν δεδομένα από διαφορετικές πηγές [7].
5. **Ειδοποιήσεις και Ανταλλαγή Μηνυμάτων:** Χρήση *Message Brokers*, όπως **RabbitMQ** ή **Kafka**, για τη διαχείριση ειδοποιήσεων και μηνυμάτων.
6. **Εξισορρόπηση Φορτίου και Απόδοση Συστήματος:** Για τη διασφάλιση της **υψηλής διαθεσιμότητας και απόδοσης**, το σύστημα χρησιμοποιεί **μηχανισμούς εξισορρόπησης φορτίου (Load Balancing)**, οι οποίοι κατανέμουν δυναμικά τα εισερχόμενα αιτήματα μεταξύ πολλαπλών διακομιστών backend. Αυτό επιτυγχάνεται με τη χρήση **αντίστροφου μεσολαβητή (reverse proxy) και load balancers**, όπως **NGINX, HAProxy ή cloud-based λύσεις (AWS ELB, GCP Load Balancer)**.



Εικόνα 1: Αρχιτεκτονική Τυπικού Πληροφοριακού Συστήματος

3. Σχετικές Εργασίες

Η παρούσα ενότητα εξετάζει τις σχετικές ερευνητικές εργασίες και συστήματα που συνδέονται με τη διαχείριση επιστημονικών συνεδρίων και προσκλήσεων για υποβολή άρθρων (CFP), με ανάλυση της βασικής συνεισφοράς τους, των θετικών και αρνητικών στοιχείων τους, και μια σύγκριση με την προτεινόμενη διπλωματική εργασία.

3.1 Silverfish

Το **Silverfish**[1] αποτελεί ένα **σύστημα εξόρυξης γνώσης και συσχέτισης**, το οποίο συλλέγει δεδομένα μέσω **μεταφόρτωσης από χρήστες και web crawling**. Η βασική του συνεισφορά έγκειται στην αξιοποίηση ενός **γράφου συνύπαρξης (co-occurrence graph)** για την **ανάλυση θεματικών περιοχών** και τη **βελτίωση της διάδοσης πληροφοριών**, με ιδιαίτερη έμφαση στην εξυπηρέτηση **ακαδημαϊκών χρηστών**.

Ωστόσο, η λειτουργία του περιορίζεται σε δεδομένα που παρέχονται από χρήστες και δεν έχει επεκτασιμότητα σε διαφορετικές θεματολογίες. Συγκεκριμένα, το σύστημα βασίζεται στη γνώση που καταχωρούν οι ίδιοι οι χρήστες και δεν διαθέτει μηχανισμούς γενίκευσης για την ανακάλυψη νέων, μη καταγεγραμμένων θεματικών περιοχών, γεγονός που μειώνει την ικανότητά του να αναγνωρίζει και να συσχετίζει πεδία που δεν έχουν ήδη προστεθεί στη βάση δεδομένων του [1].

3.2 CIERS

Το **CIERS (Combined Information Extraction and Retrieval System)** συνδυάζει τεχνικές **ανάκτησης πληροφοριών (Information Retrieval - IR)** και **εξαγωγής πληροφοριών (Information Extraction - IE)** με στόχο τη βελτίωση της ακρίβειας στην ανάκτηση προσκλήσεων για την υποβολή άρθρων (CFP). Η βασική συνεισφορά του συστήματος είναι η **αυτοματοποιημένη εξαγωγή και αναγνώριση χαρακτηριστικών** όπως **ημερομηνίες και τοποθεσίες**, επιτρέποντας πιο **ακριβή και στοχευμένη αναζήτηση** [2].

Παρόλο που το **CIERS** επιτυγχάνει υψηλή ακρίβεια στις ανακτήσεις, η προσαρμογή του σε νέες θεματικές περιοχές **απαιτεί σημαντικές τροποποιήσεις στους αλγόριθμούς του**, καθώς βασίζεται σε **κανόνες και προκαθορισμένα πρότυπα (rule-based approaches) ή εκπαιδευμένα μοντέλα που εξαρτώνται από συγκεκριμένα χαρακτηριστικά κειμένου**.

Αυτό σημαίνει ότι η μεταφορά του σε διαφορετικούς επιστημονικούς τομείς απαιτεί **επαναπροσαρμογή των χαρακτηριστικών που εξάγει (feature engineering) ή εκ νέου εκπαίδευση των μοντέλων**. Ως αποτέλεσμα, **η εφαρμογή του δεν είναι άμεσα επεκτάσιμη** και απαιτεί εκτεταμένη παραμετροποίηση και προσαρμογή για να λειτουργήσει σε νέα πεδία.

3.3 ConfSys

[1] Srinivasa, S., & Rachakonda, A. R. (2008). *Silverfish: A Contextual Knowledge Extraction System*. International Conference on Management of Data.

Το **ConfSys**[2] είναι ένα σύστημα διαχείρισης επιστημονικών συνεδρίων που αυτοματοποιεί τη διαδικασία υποβολής άρθρων, αξιοποιώντας μεθόδους **επεξεργασίας φυσικής γλώσσας (NLP)** και **μηχανικής μάθησης** για την εξαγωγή μεταδεδομένων. Βασική του συνεισφορά είναι η ενσωμάτωση έξυπνων μηχανισμών κατανομής εργασιών στους αξιολογητές, καθώς και η αυτοματοποιημένη διαχείριση των διαδικασιών συνεδρίων [3].

Ενώ το ConfSys μειώνει την ανάγκη για χειροκίνητη παρέμβαση, η λειτουργικότητά του περιορίζεται από στατικά προκαθορισμένα μοντέλα και έχει περιορισμένη προσαρμογή σε συνέδρια που δεν ακολουθούν τυποποιημένες διαδικασίες.

3.4 OpenResearch

Το **OpenResearch**[3] παρέχει μια πλατφόρμα crowdsourcing για τη συλλογή και οργάνωση μεταδεδομένων επιστημονικών εκδηλώσεων, αξιοποιώντας τεχνολογίες **Semantic Web** και **Linked Data**. Η βασική συνεισφορά του είναι η δυνατότητα αναζήτησης επιστημονικών εκδηλώσεων βάσει πολλαπλών κριτηρίων, όπως το ιστορικό αποδοχής άρθρων, η βιωσιμότητα της εκδήλωσης ως σειράς συνεδρίων και η φήμη των διοργανωτών [4] [5].

Ωστόσο, ένα κρίσιμο ζήτημα είναι η αποτίμηση της ποιότητας μιας εκδήλωσης πριν αυτή διεξαχθεί. Στο OpenResearch, η ποιότητα αποτιμάται κυρίως μέσω ιστορικών δεδομένων, όπως τα acceptance rates, η συνέπεια στην οργάνωση και η φήμη των συμμετεχόντων. Αυτό σημαίνει ότι για νέες εκδηλώσεις ή για συνέδρια με χαμηλή ιστορική τεκμηρίωση, η ακρίβεια αυτής της αποτίμησης είναι αμφισβητήσιμη.

Η προτεινόμενη διπλωματική εργασία **υπερβαίνει τις δυνατότητες** των υπαρχόντων συστημάτων, όχι μόνο μέσω της ενσωμάτωσης επιμέρους λειτουργιών, όπως οι **εξατομικευμένες προτάσεις**, αλλά κυρίως μέσω του **συνδυασμού τους σε ένα ενιαίο και αποδοτικό σύστημα**. Σε αντίθεση με τα υπάρχοντα συστήματα, τα οποία είτε εστιάζουν σε συγκεκριμένες λειτουργίες είτε παρουσιάζουν περιορισμούς ως προς την κλιμάκωση τους, η προτεινόμενη λύση ενσωματώνει **λειτουργίες κοινωνικής δικτύωσης** (όπως αξιολογήσεις και σχόλια συνεδρίων), παρέχοντας μια **ολιστική προσέγγιση** στη διαχείριση προσκλήσεων υποβολής άρθρων (CFP) και επιστημονικών συνεδρίων.

Επιπλέον, η χρήση **τεχνολογιών δοχειοποίησης (containerization)** διασφαλίζει **εύκολη φιλοξενία και κλιμάκωση σε περιβάλλοντα νέφους**, καθιστώντας το σύστημα πιο **ευέλικτο και προσαρμόσιμο** σε διαφορετικά επιστημονικά πεδία και ανάγκες. Η **ενοποιημένη λειτουργικότητα** του συστήματος επιτρέπει τη **βελτίωση της αποτελεσματικότητας** στη διάχυση επιστημονικής πληροφορίας, προσφέροντας στους χρήστες **βελτιωμένη εμπειρία αλληλεπίδρασης** και **πιο στοχευμένες προτάσεις** συγκριτικά με τα προαναφερόμενα συστήματα.

Παρακάτω παρατίθεται ένας συγκριτικός πίνακας που δείχνει **ποιες λειτουργίες υποστηρίζει κάθε σύστημα** και πώς το προτεινόμενο σύστημα **υπερέχει** συνδυάζοντας όλες αυτές τις λειτουργίες.

---

[2] Yadav, Y. O., & Desai, B. C. (2023)**.** *ConfSys - An Intelligent Conference Management System*. International Database Engineered Applications Symposium (IDEAS)

[3] Behrend, A., Vahdati, S., Lange, C., & Engels, C. (2017). *Towards a Cloud-Based Service for Maintaining and Analyzing Data About Scientific Events*

Πίνακας 1: Σύγκριση Σχετικών Συστημάτων

#	Silverfish	CIERS	ConfSys	OpenResearch	Προτεινόμενο Σύστημα
Εξατομικευμένες προτάσεις	✔	✖	✖	✖	✔
Διαχείριση θεματικών περιοχών	✔	✔	✖	✔	✔
Σύστημα κοινωνικής δικτύωσης	✖	✖	✖	✔	✔
Υποστήριξη κλιμάκωσης σε νέφος	✖	✖	✖	✖	✔
Ενοποιημένη λειτουργικότητα	✖	✖	✖	✖	✔

4. Ανάπτυξη Συστήματος

Η ανάπτυξη του προτεινόμενου συστήματος ακολούθησε το **μοντέλο του καταρράκτη (Waterfall Model)**, το οποίο χαρακτηρίζεται από **διακριτές και διαδοχικές φάσεις ανάπτυξης**. Κάθε φάση ολοκληρώνεται πριν από την έναρξη της επόμενης, διασφαλίζοντας **σαφή σχεδιασμό, οργανωμένη υλοποίηση και τεκμηριωμένη αξιολόγηση** του συστήματος.

Σκοπός της παρούσας ενότητας είναι να παρουσιάσει **τα αποτελέσματα της ανάπτυξης** σε κάθε δραστηριότητα/φάση που επιτελέστηκε, ακολουθώντας τη **δομή του μοντέλου καταρράκτη**. Συγκεκριμένα, περιγράφονται τα αποτελέσματα για τις δραστηριότητες/φάσεις **ανάλυσης απαιτήσεων, σχεδίασης, υλοποίησης, δοκιμών και αξιολόγησης**, αναδεικνύοντας με αυτό τον τρόπο πώς το κάθε στάδιο συνέβαλε στη συνολική ολοκλήρωση της εφαρμογής.

Στην παρούσα εργασία, η ανάπτυξη του συστήματος ακολουθεί το **μοντέλο καταρράκτη (Waterfall Model)**, καθώς προσφέρει μια σαφή δομή με διακριτά στάδια, επιτρέποντας τη λεπτομερή τεκμηρίωση και ανάλυση πριν από την υλοποίηση. Η επιλογή του συγκεκριμένου μοντέλου βασίζεται στην ανάγκη για **σαφή καθορισμό των απαιτήσεων εκ των προτέρων**, δεδομένου ότι το σύστημα διαχείρισης συνεδρίων απαιτεί **σταθερή αρχιτεκτονική και σαφή ροή δεδομένων**. Στα επόμενα τμήματα, παρουσιάζονται τα αποτελέσματα εφαρμογής του **μοντέλου καταρράκτη**, περιγράφοντας πώς κάθε φάση της ανάπτυξης καλύπτει συγκεκριμένες δραστηριότητες, ξεκινώντας από την ανάλυση απαιτήσεων και συνεχίζοντας με τη σχεδίαση, την υλοποίηση, τη δοκιμή και την τελική αξιολόγηση του συστήματος.

4.1 Ανάλυση Απαιτήσεων

Η **ανάλυση απαιτήσεων** αποτελεί ένα από τα πιο κρίσιμα βήματα στην ανάπτυξη ενός συστήματος και άρα του έξυπνου συστήματος διαχείρισης και ανακοίνωσης επιστημονικών συνεδρίων. Στόχος της είναι η σαφής κατανόηση των αναγκών και των προσδοκιών των χρηστών, καθώς και η διαμόρφωση ενός λεπτομερούς συνόλου απαιτήσεων που θα καθοδηγήσει τα επόμενα στάδια ανάπτυξης. Οι απαιτήσεις ενός συστήματος διακρίνονται σε **λειτουργικές** και **μη λειτουργικές**, ανεξάρτητα από το μοντέλο ανάπτυξης που ακολουθείται, καθώς κάθε προσέγγιση ανάπτυξης λογισμικού (π.χ. Agile, V-Model, Spiral) πάντοτε περιλαμβάνει την ανάλυση απαιτήσεων και άρα και αυτούς τους δύο βασικούς τύπους απαιτήσεων.

4.1.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις προσδιορίζουν τις κύριες λειτουργίες που πρέπει να υποστηρίζει το σύστημα. Οι απαιτήσεις χωρίζονται σε τέσσερις (4) κατηγορίες, με βάση τις οντότητες προς διαχείριση:

Θα πρέπει να υποστηριχθούν τρία είδη χρηστών:

- *επισκέπτης*, ο οποίος μπορεί να αναζητά και να βλέπει βασικές πληροφορίες συνεδρίων
- *ερευνητής*: επιπρόσθετα του επισκέπτη, μπορεί να κάνει subscribe σε θέματα (topics) συνεδρίων για να λαμβάνει την αντίστοιχη πληροφορία, όταν αυτή δημοσιεύεται. Ακόμη θα μπορεί να δημιουργεί νέα θέματα. Επίσης, θα μπορεί να εισάγει/διαχειρίζεται πληροφορίες για συνέδρια που θα διοργανώνει ο ίδιος. Τέλος, θα μπορεί να χρησιμοποιεί τις δυνατότητες κοινωνικής δικτύωσης του συστήματος (βαθμολόγηση συνεδρίων, σχολιασμός τους, δημιουργία αιτημάτων φιλίας, κα.)
- *επιμελητής*: υπερ-χρήστης του συστήματος, ο οποίος θα έχει ως βασικό σκοπό την επιμέλεια των πληροφοριών που υποβάλλονται για συνέδρια (συμπ. του σχολιασμού τους και των θεμάτων των συνεδρίων). Επίσης, είναι υπεύθυνος για τον έλεγχο και την διαχείριση χρηστών.

Οι βασικές λειτουργίες του συστήματος διαχωρίζονται με βάση την οντότητα συστήματος που αφορούν ως εξής.

4.1.1.1 Λειτουργίες Διαχείρισης Χρηστών

Το σύστημα θα πρέπει να είναι σε θέση να διαχειρίζεται τους χρήστες του. Οι λειτουργίες διαχείρισης περιλαμβάνουν τις εξής:

- **FR1:** *Εγγραφή χρήστη:* θα πρέπει να επιτρέπεται σε έναν επισκέπτη να αιτείται την εγγραφή του ως χρήστης στο σύστημα. Κατά την αίτηση, θα πρέπει να παρέχονται οπωσδήποτε το όνομα χρήστη (username), το ονοματεπώνυμο, το email του καθώς και ο κωδικός του (password). Προαιρετικά, μπορεί να παρέχεται και ο τίτλος εργασίας του, ο οργανισμός στον οποίο εργάζεται και η ηλικία του. Θα πρέπει να ελέγχεται αν υπάρχει χρήστης με το ίδιο όνομα ή email. Σε αυτή την περίπτωση, η εγγραφή θα αποτυγχάνει και αντίστοιχο μήνυμα λάθους θα πρέπει να εμφανίζεται. Ο κωδικός του χρήστη θα πρέπει να κατακερματίζεται
    - *Προαιρετικό*: θα μπορούσε να υποστηρίζεται και εγγραφή χρήστη μέσω κάποιας άλλης υπηρεσίας, όπως Google Accounts.
- **FR2:** *Διαγραφή χρήστη* ο χρήστης θα πρέπει να μπορεί να αιτείται την διαγραφή του. Αυτό θα έχει ως επίπτωση την διαγραφή όλων των στοιχείων που αφορούν τον χρήστη, εκτός από τα συνέδρια και τα θέματα που έχει δημιουργήσει, η πληροφορία των οποίων θα παγώνει (δεν θα μπορεί να αλλάξει). Αν όμως η διαγραφή γίνεται από τον επιμελητή, αυτός θα έχει την ευχέρεια να διαγράψει τα πάντα όσον αφορά τον χρήστη προς διαγραφή, εφόσον το κρίνει απαραίτητο (πχ. ο χρήστης είναι κακόβουλος και παρέχει εν γένει ψευδείς πληροφορίες στο σύστημα).
- **FR3:** *Ενημέρωση στοιχείων χρήστη:* ο χρήστης θα πρέπει να μπορεί να ανανεώνει τις πληροφορίες για το άτομό του, όπως το ονοματεπώνυμό του, την ηλικία του, και το email του. Για λόγους επαναφοράς κωδικού, καλό είναι να αποθηκεύεται και η απάντηση σε μια από τις ερωτήσεις που εμφανίζει το σύστημα στον χρήστη (και ο χρήστης επιλέγει προς απάντηση).
- **FR4:** *Επαναφορά/ενημέρωση κωδικού*: εφόσον ο χρήστης επιθυμεί την επαναφορά κωδικού, το σύστημα θα του εμφανίζει μια σειρά από ερωτήσεις κι αυτός θα πρέπει να δώσει την απάντηση που έχει δώσει προηγουμένως σε μια από αυτές κατά την ενημέρωση των στοιχείων του. Εφόσον η απάντηση είναι σωστή, τότε θα δημιουργείται ένας προσωρινός ασφαλής κωδικός για τον χρήστη, ο οποίος θα αποστέλλεται στο email του. Έπειτα, ο χρήστης θα μπορεί να αιτηθεί την ενημέρωση του κωδικού του ώστε να τον αλλάξει. Σημειώνεται πως αν ο χρήστης δεν αλλάξει τον

κωδικό μέσα σε συγκεκριμένο χρονικό διάστημα, τότε θα απενεργοποιείται ο λογαριασμός του. Για την ενημέρωση, ο χρήστης θα πρέπει να δώσει τόσο τον προηγούμενο όσο και τον νέο κωδικό (τον τελευταίο εις διπλούν). Εν γένει, οι κωδικοί θα εναπόκεινται σε ορισμένους περιορισμούς (πχ. τουλάχιστον 10 χαρακτήρων που να περιλαμβάνουν γράμματα, ψηφία και ειδικούς χαρακτήρες) ώστε να θεωρούνται έγκυροι και ασφαλείς. Για να επιτύχει η ενημέρωση, ο προηγούμενος κωδικός θα πρέπει να είναι σωστός και ο νέος κωδικός να είναι τόσο έγκυρος όσο και ασφαλής καθώς και να ταιριάζουν απόλυτα οι 2 τιμές του που παρέχονται από τον χρήστη.

- **FR5:** *Ταυτοποίηση*: ο χρήστης θα πρέπει να μπορεί να ταυτοποιείται στο σύστημα παρέχοντας το όνομα χρήστη & κωδικό του. Προαιρετικά, μπορεί να υποστηρίζεται και εξωτερική υπηρεσία ταυτοποίησης (πχ. Google Accounts). Έπειτα, θα παράγεται ένα μοναδικό token με συγκεκριμένο χρονικό όριο εγκυρότητας, το οποίο θα μπορεί να χρησιμοποιείται κατά την αίτηση εκτέλεσης των διαφόρων λειτουργιών που παρέχονται από το σύστημα προς τους χρήστες του.
- **FR6:** *Εμφάνιση χρήστη*: θα μπορεί να εμφανίζεται όλη η πληροφορία για έναν χρήστη με βάση το όνομα χρήστη του. Αυτό θα ισχύει είτε στην περίπτωση επιμελητή είτε του ίδιου του χρήστη που ζητάει την εμφάνιση των πληροφοριών του.
- **FR7:** *Αναζήτηση χρήστη*: θα παρέχεται η δυνατότητα αναζήτησης του χρήστη με βάση το ονοματεπώνυμό του ή το όνομα χρήστη του. Θα επιστρέφεται πίσω λίστα από χρήστες που ταιριάζει στους όρους αναζήτηση του χρήστη (ουσιαστικά ενός επιμελητή)

4.1.1.2 Λειτουργίες Διαχείρισης Συνεδρίων

Το σύστημα θα πρέπει να είναι σε θέση να διαχειρίζεται τις πληροφορίες για τα συνέδρια με βάση τις ακόλουθες λειτουργίες:

- **FR8:** *Εισαγωγή συνεδρίου*: η εισαγωγή ενός συνεδρίου μπορεί να πραγματοποιείται με δύο εναλλακτικούς τρόπους:
    - χειρωνακτικά από έναν ερευνητή για το συνέδριο που διοργωνώνει
    - αυτόματα από το σύστημα καθώς κάνει crawl CFP & conference cites

  Η πληροφορία για ένα συνέδριο μπορεί να περιλαμβάνει τα εξής:
    - *όνομα συνεδρίου*
    - *ακρώνυμο συνεδρίου (πχ. ESOCC 2025)*
    - *τύπος συνεδρίου* (κανονικό ή workshop)
    - *συνέδριο φιλοξενίας*: συσχέτιση με ένα συνέδριο που φιλοξενεί το τρέχων συνέδριο τύπου workshop
    - *συνφιλοξενούμενα συνέδρια*: συσχέτιση με συνέδρια που διοργανώνονται μαζί στον ίδιο τόπο με το τρέχων συνέδριο
    - *τόπος διεξαγωγής*
    - *διοργανωτές συνεδρίου*
    - *επιτροπή προγράμματος*
    - *σύνδεσμοι σε παλαιότερες εκδόσεις του συνεδρίου*
    - *σύνδεσμος site του συνεδρίου*
    - *διαθεσιμότητα συνεδρίου*: ο ερευνητής μπορεί να επιλέξει αρχικά το συνέδριο να μην είναι ορατό στους άλλους χρήστες και στους επισκέπτες του συστήματος. Κι αυτό διότι μπορεί όλη η απαραίτητη πληροφορία για ένα συνέδριο να μην είναι ακόμη ολοκληρωμένη. Όταν λοιπόν ένα συνέδριο είναι

μη διαθέσιμο, όχι μόνο δεν είναι ορατό, αλλά και οι χρήστες που έχουν εγγραφεί σε θέματα σχετικά με το συνέδριο δεν θα μπορούν να ενημερώνονται από το σύστημα για αλλαγές που αφορούν το συνέδριο.

- *θέματα συνεδρίου*: το συνέδριο είναι καλό να σχετίζεται με ένα ή παραπάνω θέματα ώστε να ενημερώνονται οι subscribers των θεμάτων αυτών για το συνέδριο. Τα θέματα είναι ουσιαστικά ερευνητικοί τομείς που πραγματεύεται το συνέδριο - τα θέματα είναι βασικές οντότητες του συστήματος που διαχειρίζονται από τους χρήστες του.
- *κύριο μέρος / CFP*: αυτό μπορούμε να θεωρήσουμε πως είναι μια υπο-οντότητα ενός συνεδρίου με τις εξής πληροφορίες προς κάλυψη. Σημειώνουμε πως όλες οι υπο-οντότητες ενός συνεδρίου θα πρέπει να είναι πλήρως διαχειρίσιμες (CRUD):
  - *περιγραφή*: κείμενο περιγραφής για το CFP
  - *διορία υποβολής abstract*
  - *διορία υποβολής άρθρων*
  - *διορία τελικής απόφασης*: για το αν θα γίνουν δεκτά ή όχι τα άρθρα
  - *διορία τελικής υποβολής δεκτών άρθρων*
  - *τύποι άρθρων προς υποβολή* (πχ. κανονικό, σύντομο)
  - *δυνατότητα indexing αποδεκτών άρθρων*
  - *publisher των άρθρων* (πχ. Springer)
- *call for workshops*: άλλη μια υπο-οντότητα που αφορά το κάλεσμα για υποβολή προτάσεων για διεξαγωγή workshop. Βασικές πληροφορίες προς κάλυψη:
  - *περιγραφή*: κείμενο περιγραφής για το κάλεσμα
  - *διορία υποβολής προτάσεων για workshop*
  - *διορία τελικής απόφασης*: για το ποιες προτάσεις workshop θα γίνουν δεκτές
- *call for demonstrations*: παρόμοιο με CFP αλλά αφορά την υποβολή σύντομων άρθρων με δυνατότητα επίδειξής τους (με έναν ή παραπάνω τρόπους - εξαρτάται από τις ανάγκες του συνεδρίου) κατά τη διάρκεια του συνεδρίου. Θα υπάρχουν τα ίδια πεδία με CFP αλλά δεν έχει νόημα το *τύπος άρθρων προς υποβολή*. Θα μπορούσε να περιλαμβάνει ως πεδίο τον τρόπο/τρόπους επίδειξης (αν και μπορεί να καλύπτονται από την κειμενική περιγραφή).
- *call for PhD contributions*: παρόμοιο με CFP αλλά αφορά την υποβολή άρθρων από PhD candidates. Θα υπάρχουν τα ίδια πεδία με CFP αλλά δεν έχει νόημα το *τύπος άρθρων προς υποβολή*.
- *call for posters*: παρόμοι με CFP αλλά αφορά την υποβολή πολύ σύντομων άρθρων με συνήθως δισέλιδη μορφή που συνοδεύονται, εφόσον γίνουν δεκτά, με posters που θα αναρτηθούν και θα παρουσιαστούν στους χώρους του συνεδρίου. Παρόμοια υπο-οντότητα με CFP αλλά δεν έχει το πεδίο *τύπος άρθρων προς υποβολή*.
- *call for participation:* κάλεσμα για συμμετοχή στο συνέδριο. Πεδία που μπορούν να καλυφθούν:
  - *περιγραφή*: κείμενο περιγραφής του καλέσματος
  - *σύνδεσμος προς την ιστοσελίδα της εγγραφής* (στο συνέδριο)

        - *σύνδεσμος προς την ιστοσελίδα που εμφανίζει τα άρθρα που έγιναν δεκτά στο συνέδριο*
- **FR9:** *Ενημέρωση συνεδρίου*: η ενημέρωση της πληροφορίας για ένα συνέδριο θα πραγματοποιείται με δύο διαφορετικούς τρόπους:
    - *αντιδραστικός (reactive)*: ο κάτοχος του συνεδρίου μπορεί να αλλάζει όλες τις πληροφορίες για το συνέδριό του, εκτός από το αναγνωριστικό του συνεδρίου, το οποίο παράγεται από το ίδιο το σύστημα και δεν αλλάζει
    - *προληπτικός (proactive)*: το σύστημα περιοδικά (το διάστημα περιοδικότητας μπορεί να είναι μια παράμετρος διαμόρφωσης του συστήματος) μπορεί να κάνει crawl CFP sites & conference cites ώστε να αντλεί είτε νέα πληροφορία, εφόσον πλέον είναι διαθέσιμη, είτε ανανεώσεις υπάρχουσας πληροφορίας (πχ. αλλαγές σε διορίες υποβολών άρθρων) για να ενημερώνει τα συνέδρια που έχουν καταχωρηθεί στο σύστημα.

  Κατά την ενημέρωση συνεδρίου, εφόσον γίνονται αλλαγές σε ημερομηνίες/διορίες ή προσθήκες υπο-οντοτήτων, τότε θα πρέπει να ενημερώνονται αυτόματα από το σύστημα οι subscribers σε θέματα που σχετίζονται με το συνέδριο.
- **FR10:** *Διαγραφή συνεδρίου*: θα μπορεί να διαγράφεται το συνέδριο με βάση το αναγνωριστικό του. Αυτό σημαίνει πως διαγράφεται τόσο το συνέδριο όσο και οι υπο-οντότητές του καθώς και οποιοδήποτε thread σχετικό με το συνέδριο. Αλλά δεν θα πρέπει να διαγράφονται τα θέματα που είναι σχετικά με το συνέδριο.
- **FR11:** *Αναζήτηση συνεδρίου*: Θα πρέπει να υποστηρίζονται δύο βασικά είδη αναζήτησης από οποιοδήποτε χρήστη του συστήματος:
    - *απλή αναζήτηση*: με βάση λέξεις-κλειδιά. Εδώ ο χρήστης παρέχει την είσοδό του μέσω απλής φόρμας και του επιστρέφονται πίσω τα σχετικά αποτελέσματα. Το ταίριασμα μπορεί να γίνει σε οποιαδήποτε πληροφορία του συνεδρίου.
    - *εξεζητημένη αναζήτηση*: Παρουσιάζεται στον χρήστη μια σύνθετη φόρμα όπου μπορεί να παρέχει τιμές σε πεδία που σχετίζονται με ιδιότητες ή υπο-οντότητες ενός συνεδρίου. Προαιρετικά, ο χρήστης μπορεί να συνδυάζει τις τιμές των πεδίων με λογικούς τελεστές για μια πιο σύνθετη αναζήτηση. Και σε αυτή την περίπτωση (αναζήτησης), επιστρέφεται πίσω μια λίστα από συνέδρια που ικανοποιούν τους περιορισμούς αναζήτησης του χρήστη.
- **FR12:** *Δημοσίευση συνεδρίου*: Ο κάτοχος του συνεδρίου, όποτε το θεωρήσει πρέπον, θα μπορεί να δημοσιεύσει (χειρωνακτικά μέσω του συστήματος) σε subscribers (θεμάτων του συνεδρίου) το συνέδριο ή υπο-οντότητες αυτού. Σημειώνεται εδώ πως μιλάμε για χειρωνακτική on-demand δημοσίευση. Οπότε, είναι επιπρόσθετη σε σχέση με αυτές που πραγματοποιούνται αυτόματα από το σύστημα.
- **FR13:** *Εμφάνιση συνεδρίου*: Με βάση τον παρεχόμενο κωδικό του συνεδρίου, το σύστημα θα εμφανίζει όλες τις διαθέσιμες πληροφορίες για ένα συνέδριο.

Σημειώνεται πως η λειτουργία του crawling εκτελείται περιοδικά και με αυτόματο τρόπο από το σύστημα. Δεν χρειάζεται η χειρωνακτική του εκτέλεση με βάση κάποια μέθοδο RESTful API ή μέσω του UI του συστήματος. Το crawling θα πρέπει να γίνεται έξυπνα ώστε να καλύπτονται όσο γίνεται περισσότερο όλα τα δυνατά πεδία περιγραφής συνεδρίου και των υπο-οντοτήτων του. Η εξυπνάδα έγκειται, όπως προαναφέρθηκε, όχι μόνο ως προς την εισαγωγή πληροφοριών αλλά και την ανανέωσή τους. Θα μπορούσε να υλοποιηθεί μέσω μεθόδων μηχανικής μάθησης ή LLMs.

4.1.2 Λειτουργίες Διαχείρισης Θεμάτων

Επειδή ένα θέμα μπορεί να είναι κοινό μεταξύ διαφορετικών συνεδρίων και είναι ένας βασικός μηχανισμός ειδοποίησης χρηστών για συνέδρια, είναι σημαντικό να γίνεται σωστά και πλήρως η διαχείρισή του. Για αυτό τον λόγο, το σύστημα θα πρέπει να υποστηρίζει τις εξής λειτουργίες διαχείρισης θεμάτων:

- **FR14:** *Εισαγωγή θέματος*: ένας επιμελητής ή ερευνητής θα μπορεί να εισάγει ένα θέμα συνεδρίων, το οποίο θα πρέπει να είναι μοναδικό. Η πληροφορία για ένα θέμα που θα πρέπει να δοθεί περιλαμβάνει τα εξής:
    - *όνομα θέματος*
    - *συντομογραφία θέματος* (προαιρετικό)
    - *κωδικός* (παράγεται αυτόματα από το σύστημα - αυτό ισχύει για όλες τις οντότητες που διαχειρίζεται το σύστημα)
    - *περιγραφή θέματος*: σύντομη περιγραφή του θέματος
    - *πατρικό θέμα*: αν υπάρχει θέμα που θεωρείται πως είναι πατρικό του τρέχοντος, καλό είναι να προσδιορίζεται
    - *θέματα παιδιά*: παρομοίως, αν υπάρχουν θέματα που θεωρούνται πως είναι παιδιά του τρέχοντος θέματος, θα πρέπει κι αυτά να αναφερθούν (προσοχή, δεν ορίζονται εδώ αλλά αναφέρονται διότι υπάρχουν ήδη)
- **FR15:** *Ενημέρωση θέματος*: θα πρέπει να επιτρέπεται η τροποποίηση όλων των πεδίων για ένα θέμα εκτός από τον κωδικό που είναι σταθερός και δεν μπορεί να αλλάξει από οποιοδήποτε χρήστη. Αν γίνεται αλλαγή στην αναφορά σε πατρικό θέμα ή θέματα παιδιά, τότε η αλλαγή αυτή θα πρέπει να αντικατοπτριστεί και στο πατρικό θέμα / θέματα παιδιά (με την λογική πως αν δεν έχουμε πλέον ένα πατρικό θέμα, το πατρικό θέμα δεν θα μας έχει παιδί).
- **FR16:** *Διαγραφή θέματος*: θα πρέπει να επιτρέπεται η διαγραφή ενός θέματος με βάση τον κωδικό του. Όταν διαγράφεται ένα θέμα, αυτόματα αυτό σημαίνει πως η αναφορά του αφαιρείται από συνέδρια, πατρικά θέματα και θέματα παιδιά. Επιπλέον, η εγγραφές χρηστών για ειδοποιήσεις σχετικές με το θέμα θα πρέπει να διαγραφούν κι αυτές.
- **FR17:** *Εγγραφή σε θέμα*: Ένας ερευνητής μπορεί να εγγράφεται (subscribe) σε ένα θέμα. Αυτό σημαίνει πως όταν προκύπτει κάποια ανανέωση σχετική με το θέμα (για ένα συνέδριο που σχετίζεται με το θέμα ή τις υπο-οντότητές του), τότε ο ερευνητής θα πρέπει να ενημερώνεται από το σύστημα. Η ενημέρωση θα πρέπει να γίνεται εντός του UI του συστήματος. Προαιρετικά, θα μπορούσε να πραγματοποιείται πχ. μέσω ηλεκτρονικού μηνύματος στο email του χρήστη (αυτό θα μπορούσε να είναι έξτρα επιλογή που παρέχει ο χρήστης κατά την εκτέλεση της λειτουργίας αυτής).
- **FR18:** *Διαγραφή από θέμα*: αντίστοιχα, ένας ερευνητής μπορεί να διαγραφεί (unsubscribe) από ένα θέμα ώστε να μην λαμβάνει πλέον αντίστοιχες ειδοποιήσεις σχετικές με αυτό
- **FR19:** *Αναζήτηση θέματος*: θα πρέπει να υποστηρίζεται η απλή αναζήτηση θεμάτων με βάση λέξεις-κλειδιά που να ταιριάζουν είτε με το όνομα είτε με την περιγραφή του θέματος.
- **FR20:** *Εμφάνιση θέματος*: θα πρέπει να υποστηρίζεται η εμφάνιση θέματος με βάση τον κωδικό του. Η εμφάνιση θα πρέπει να απεικονίζει όλα τα πεδία ενός θέματος και

ειδικότερα να εμφανίζει συνδέσμους για την εμφάνιση των πατρικών θεμάτων και θεμάτων παιδιών του θέματος.

4.1.3 Λειτουργίες Κοινωνικής Δικτύωσης

Το σύστημα θα πρέπει να παρέχει ορισμένες βασικές λειτουργίες κοινωνικής δικτύωσης, οι οποίες περιλαμβάνουν:

- **FR21:** *Βαθμολόγηση συνεδρίων*: ένας ερευνητής μπορεί να βαθμολογεί ένα συνέδριο με βάση κάποια κλίμακα βαθμολόγησης (πχ. 1 έως 5 αστεράκια). Επιπρόσθετα, εκτός από την βαθμολογία του συνεδρίου, μπορεί να παρέχει και κείμενο που να δικαιολογεί την βαθμολογία αυτήν. Με βάση την βαθμολογία που παρέχεται από τους χρήστες του συστήματος, το σύστημα θα πρέπει να υπολογίζει τη συνολική βαθμολογία του συνεδρίου (πχ. με βάση τον μέσο όρο ή μια πιο εξειδικευμένη τεχνική ώστε να αφαιρεθούν outliers & κακόβουλες κριτικές) και να την εμφανίζει μαζί με τις άλλες πληροφορίες του συνεδρίου (πχ. όταν ζητείται η εμφάνιση του συνεδρίου). Η βαθμολογία αυτή μπορεί να αποτελεί άλλο ένα κριτήριο αναζήτησης για συνέδρια ενώ μπορεί να εμφανίζεται στην προεπισκόπηση των αποτελεσμάτων από μια αναζήτηση. Πατώντας πάνω σε μια συνολική βαθμολογία συνεδρίου, ο χρήστης θα μπορεί να δει όλες τις επιμέρους βαθμολογίες. Κάθε επιμέρους βαθμολογία θα συνοδεύεται από το αντίστοιχο κείμενο δικαιολόγησης και από το όνομα του ερευνητή που την υπέβαλε.
- ***FR22:*** *Threads/συζητήσεις*: το σύστημα θα πρέπει να επιτρέπει σε εγγεγραμμένους χρήστες να συμμετέχουν είτε σε threads που είναι ειδικά ως προς ένα συνέδριο ή σε πιο γενικά threads. Επίσης, θα μπορούν οι ίδιοι να δημιουργούν threads. Η συμμετοχή σε threads προσδιορίζει την ικανότητα υποβολής κάποιου κειμένου σχετικού με το thread και προαιρετικά την επισήμανση εκείνου του κειμένου (thread) στο οποίο γίνεται απάντηση ή σχολιασμός (αν δεν γίνεται επισήμανση, τότε το νέο κείμενο μπαίνει στο τέλος του thread). Τα threads ειδικά ως προς ένα συνέδριο μπορούν να εμφανίζονται κατά την εμφάνιση ενός συνεδρίου (πχ. μπορούν να εμφανίζονται τα 2 threads με την μεγαλύτερη συμμετοχή και ένας σύνδεσμος προς όλα τα σχετικά threads του συνεδρίου). Τα γενικά threads θα εμφανίζονται στην αρχική σελίδα του εγγραγραμμένου χρήστη (πχ. πάλι μπορούν να εμφανίζονται τα threads με την μεγαλύτερη συμμετοχή και ένας σύνδεσμος σε όλα τα γενικά threads), μόλις αυτός συνδεθεί στο σύστημα. Εκτός από την συμμετοχή σε και την εμφάνιση threads, θα πρέπει να επιτρέπεται η αναζήτηση των threads. Η διαγραφή threads θα μπορεί να γίνεται είτε από τον κάτοχό του ή από έναν επιμελητή του συστήματος.
- **FR23:** *Friends*: ένας εγγεγραμμένος χρήστης μπορεί να βλέπει πολύ βασικές πληροφορίες για άλλους εγγεγραμμένους χρήστες (πχ. όνομα χρήστη και ονοματεπώνυμο). Αλλά εφόσον κάνει αίτηση φιλίας, αν αυτή γίνει δεκτή, τότε ο χρήστης θα μπορεί να δει περισσότερες πληροφορίες για τον φίλο του (πχ. email, ηλικία, threads στα οποία συμμετέχει, βαθμολογίες που έχει κάνει). Εφόσον γίνει αίτηση φιλίας σε έναν χρήστη, ο χρήστης αυτός θα πρέπει να ειδοποιείται από το σύστημα. Έπειτα, θα μπορεί να δει όλες τις αιτήσεις φιλίας που του έχουν γίνει και να αποφασίζει ποιες θα κάνει δεκτές. Ένας χρήστης μπορεί να βλέπει όλους τους φίλους του και οποιαδήποτε στιγμή μπορεί να αποφασίσει αν θα διαγράψει κάποιον από αυτούς από φίλο του. Λογικά, ένας χρήστης θα μπορεί να πηγαίνει στη λίστα των

φίλων του και στις αιτήσεις φιλίας του (είτε που έχει κάνει αυτός είτε που έχουν γίνει προς αυτόν) από την αρχική σελίδα του συστήματος.

4.1.4 Μη Λειτουργικές Απαιτήσεις

- **NFR1:** *Χρηστική & αποκριτική διεπαφή χρήσης (UI) (Web, Mobile, Laptop, Tablet)* με
    - Ενιαίο look-n-feel
    - Κατάλληλο user-orientation
    - Καλό επίπεδο πλοηγότητας (navigability)
    - Καλό επίπεδο δυναμικότητας με ισορροπημένη χρήση Ajax & Δυναμικής HTML
- **NFR2:** *Διαχωρισμένο API (Back-End) και Front-End*
    - Level 3 REST maturity για το backend
    - Προαιρετικό: αυτόματη παραγωγή και διάθεση της τεκμηρίωσης του REST API (π.χ., μέσω swagger)
- **NFR3:** Προαιρετικό: *υποστήριξη πολλαπλών γλωσσών (Ελληνικά & Αγγλικά)*
- **NFR4:** *Καλή απόδοση συστήματος*
    - Οι απλές λειτουργίες συστήματος (πχ. εμφάνιση φορμών, αποστολή δεδομένων, κλπ.) δεν θα πρέπει να διαρκούν πάνω από 5 δευτερόλεπτα
- **NFR5:** *Καλή διαθεσιμότητα συστήματος*
    - Ικανότητα υποστήριξης πολλαπλών χρηστών (πχ. τουλάχιστον 10)
- **NFR6:** *Υψηλή αξιοπιστία συστήματος*
    - Ελάχιστος αριθμός λαθών/σφαλμάτων ανά μήνα λειτουργίας
    - Κατάλληλος χειρισμός λαθών/εξαιρέσεων με αυτο-επεξηγούμενα μηνύματα (και άλλους τρόπους οπτικοποίησης, όπου είναι δυνατόν)
    - Κατάλληλη επικύρωση φορμών τόσο στην πλευρά του πελάτη όσο και του εξυπηρετητή
- **NFR7:** *Καλό επίπεδο ασφάλειας* (αυθεντικοποίηση βασιζόμενη σε tokens, RBAC/ABAC, κατακερματισμός κωδικών, υποστήριξη TLS, προστασία CSRF κα.)
- **NFR8:** Προαιρετικό: *καλή κάλυψη δοκιμής* με χρήση δοκιμών μονάδων, ενοποίησης και συστήματος, οι οποίες μπορούν να εκτελούνται αυτόματα κατά την υποβολή κώδικα (CI/CD)

4.2 Σχεδίαση

Η σχεδίαση του συστήματος αποτελεί το επόμενο κρίσιμο βήμα μετά την ανάλυση απαιτήσεων, καθώς καθορίζει τη δομή, τη λειτουργικότητα και τη συμπεριφορά των διαφόρων συστατικών μερών του. Η σχεδίαση βασίζεται σε αρχές **Modular Design (Αρθρωτής Σχεδίασης)**, διασφαλίζοντας την ευκολία συντήρησης, την επεκτασιμότητα και την ορθολογική διαχείριση πόρων.

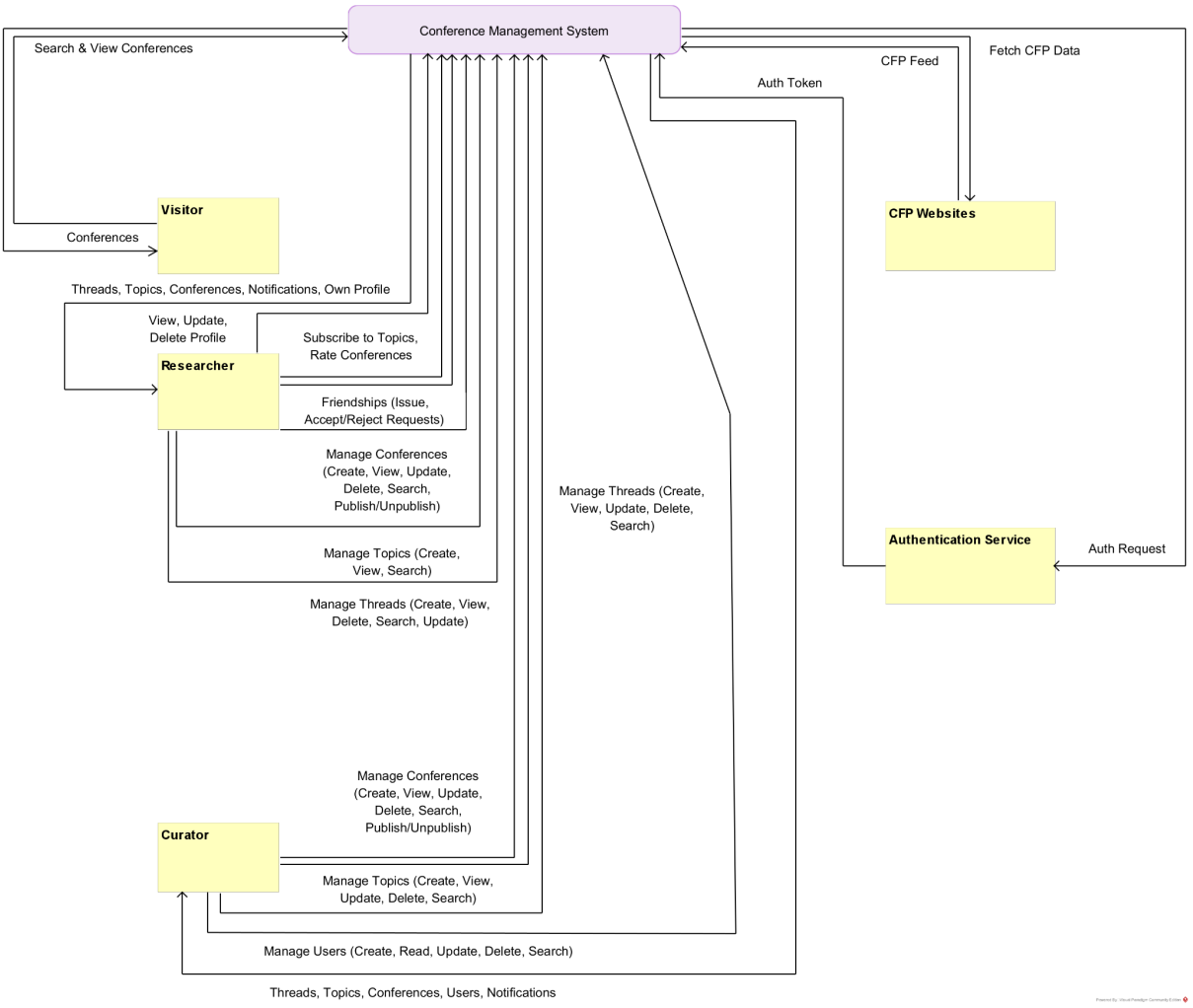
Το **Modular Design** είναι μια αρχιτεκτονική προσέγγιση στην οποία το σύστημα αναλύεται και οργανώνεται σε διακριτές, ανεξάρτητες μεταξύ τους ενότητες (modules), καθεμία από τις οποίες επιτελεί μια συγκεκριμένη λειτουργία. Αυτή η μεθοδολογία προσφέρει σημαντικά πλεονεκτήματα, όπως ευκολία στη συντήρηση και αναβάθμιση, επαναχρησιμοποίηση κώδικα και βελτιωμένη διαλειτουργικότητα. Επιπλέον, τα αρθρωτά συστήματα είναι πιο ανθεκτικά σε αλλαγές, καθώς κάθε ενότητα μπορεί να τροποποιηθεί ή να αντικατασταθεί χωρίς να επηρεάζει τη συνολική λειτουργικότητα του συστήματος [8].

Σε αυτή την ενότητα παρουσιάζονται τα κύρια σχεδιαστικά μοντέλα που χρησιμοποιήθηκαν για τη σχεδίαση της εφαρμογής μας, συνοδευόμενα από σύντομες σχετικές περιγραφές.

4.2.1 Διάγραμμα Περιβάλλοντος (Context Diagram)

Το **Διάγραμμα Περιβάλλοντος** αποτυπώνει τις κύριες αλληλεπιδράσεις του συστήματος με εξωτερικές οντότητες, όπως χρήστες (επισκέπτης, ερευνητής, επιμελητής) και εξωτερικά συστήματα (π.χ., CFP ιστοσελίδες, υπηρεσίες αυθεντικοποίησης, όπως Google Accounts, κα.). Το διάγραμμα περιβάλλοντος για το δικό μας σύστημα φαίνεται στην Εικόνα 2. Ακολουθεί μια σύντομη ανάλυσή του.

- **Κύριες Οντότητες:**
    - Χρήστες: Επισκέπτης (Visitor), Ερευνητής (Researcher), Επιμελητής (Curator)
    - Εξωτερικά Συστήματα: CFP Websites, Authentication Service (Υπηρεσία Αυθεντικοποίησης)
- **Κύριες Αλληλεπιδράσεις:**
    - Εγγραφή & Αυθεντικοποίηση Χρηστών
    - Διαχείριση Συνεδρίων (Εισαγωγή, Ενημέρωση, Διαγραφή, Δημοσίευση, Θέαση)
    - Διαχείριση Θεμάτων (Εισαγωγή, Ενημέρωση, Θέαση, Διαγραφή)
    - Αναζήτηση Θεμάτων, Συνεδρίων και Χρηστών
    - Διαχείριση Συνδρομών, Threads και Ειδοποιήσεων
    - Διαχείριση Χρηστών (Διαγραφή, Ενημέρωση, Επαναφορά Κωδικού, Αλλαγή Κωδικού)
    - Διαχείριση Threads (Δημιουργία, Θέαση, Ενημέρωση, Διαγραφή, Αναζήτηση)
    - Εισαγωγή Σχολίων σε Threads, Διαχείριση Αιτημάτων Φιλίας & Αξιολόγηση Συνεδρίων



Εικόνα 2: Διάγραμμα Περιβάλλοντος του Συστήματος

4.2.2 Διάγραμμα Συστατικών (Component Diagram)

Το **Διάγραμμα Συστατικών** παρουσιάζει τις κύριες (αρχιτεκτονικά) ενότητες (components) του συστήματος και τον τρόπο αλληλεπίδρασής τους.

- **Frontend (Περιβάλλον Χρήστη)**
  - Web Interface (React.js): Παρέχει μια αποκριτική (responsive) διεπαφή χρήστη (user interface - UI) που ανταποκρίνεται τόσο σε επιτραπέζιες όσο και σε κινητές συσκευές. Δεν υπάρχει ξεχωριστό συστατικό για τη διεπαφή με κινητές συσκευές, καθώς η υπάρχουσα διεπαφή χρήστη διαμορφώνεται δυναμικά ανάλογα με την πλατφόρμα του χρήστη.
- **Backend (Λογική Εφαρμογής)**
  Το backend είναι οργανωμένο σε Controllers, Services και Repositories, σύμφωνα με το αρχιτεκτονικό μοτίβο MVC (Model-View-Controller) (ουσιαστικά το μοτίβο Controller-Service-Repository / CSR που είναι μια επέκταση του MVC).
  - Συστατικό User Management:
    - *User Management Controller* (REST API για διαχείριση χρηστών).

        - *User Management Service* (επεξεργασία και επιχειρησιακή λογική διαχείρισης χρηστών).
        - *User Repository* (αλληλεπίδραση με τη βάση δεδομένων ως προς τη διαχείριση αντικειμένων χρηστών).
    - Συστατικό Conference Management:
        - *Conference Management Controller* (REST API για διαχείριση συνεδρίων).
        - *Conference Management Service* (επεξεργασία και επιχειρησιακή λογική διαχείρισης συνεδρίων).
        - *Conference Repository* (πρόσβαση στη βάση δεδομένων για τη διαχείριση αντικειμένων συνεδρίων).
    - Συστατικό Topic Management:
        - *Topic Management Controller* (REST API για διαχείριση θεμάτων).
        - *Topic Management Service* (επεξεργασία και επιχειρησιακή λογική διαχείρισης θεμάτων).
        - *Topic Repository* (αποθήκευση, ενημέρωση, διαγραφή και ανάκτηση (αντικειμένων) θεμάτων).
    - Συστατικό Notification Management:
        - *Notification Controller* (REST API για διαχείριση ειδοποιήσεων χρηστών).
        - *Notification Service* (επεξεργασία και επιχειρησιακή λογική διαχείρισης ειδοποιήσεων χρηστών)
        - *Notification Repository* (αποθήκευση ειδοποιήσεων μέσω διεπαφής με υποκείμενη βάση δεδομένων).
    - Συστατικό Social Networking:
        - *Social Networking Controller* (REST API για κοινωνική δικτύωση).
        - *Social Networking Service* (επιχειρησιακή λογική κοινωνικής δικτύωσης).
        - *Social Networking Repository* (αποθήκευση σχετικών δεδομένων κοινωνικής δικτύωσης όπως αιτημάτων φιλίας, φιλίες, αξιολογήσεις συνεδρίων, κα.).
- **Βάση Δεδομένων:**
  Το σύστημα χρησιμοποιεί δύο τύπους βάσεων δεδομένων για τη βέλτιστη αποθήκευση και ανάκτηση (εν γένει διαχείριση) δεδομένων:
    - Relational DB (πχ. SQL - PostgreSQL):
        - **Αποθήκευση δομημένων δεδομένων**, όπως:
            - **Χρήστες** (προφίλ χρηστών & διαπιστευτήρια).
            - **Συνέδρια**
            - **Θέματα (Topics)**: εφόσον συνδέονται άμεσα με συνέδρια, είναι πιο λογικό να αποθηκεύονται σε **σχεσιακή βάση.**
    - NoSQL DB (πχ. MongoDB):
        - **Αποθήκευση ημι-δομημένων ή δυναμικών δεδομένων**, όπως:
            - **Ειδοποιήσεις (Notifications)** Διαφορετικές μορφές ειδοποιήσεων ανά χρήστη.
            - **Κοινωνικές Δραστηριότητες (Social Interactions)** Σχόλια, βαθμολογίες, συζητήσεις

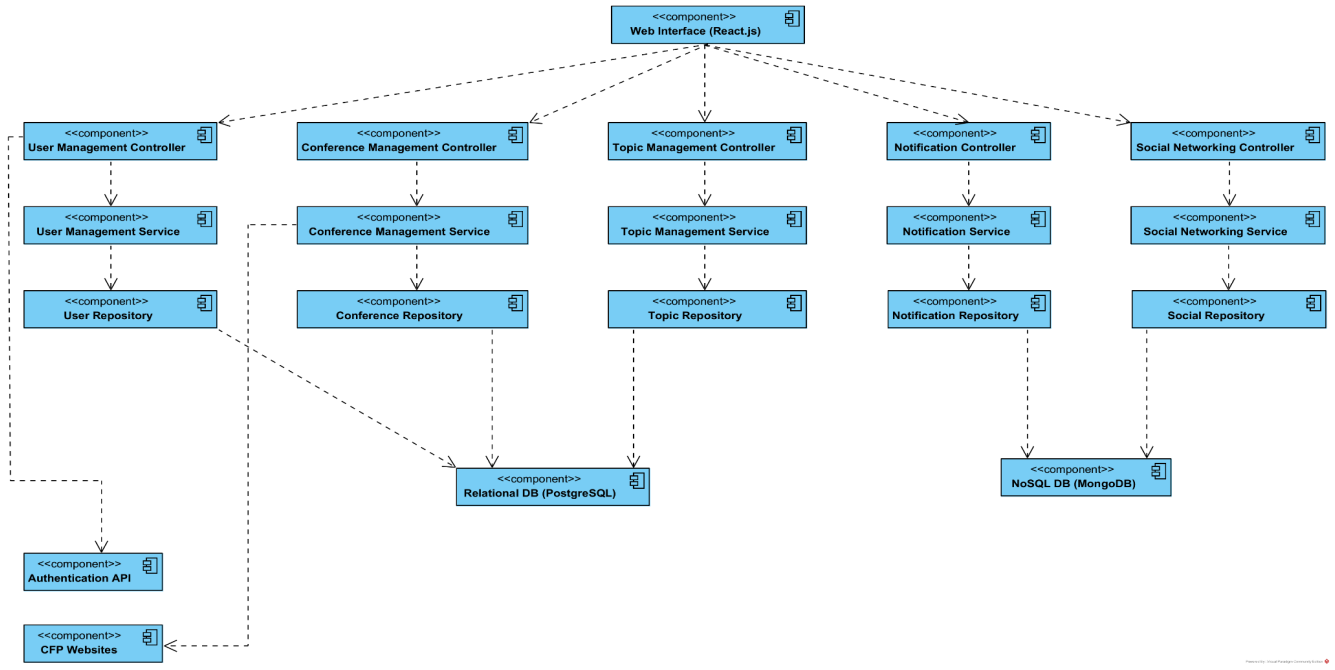
Ο διαχωρισμός SQL και NoSQL εξασφαλίζει αποδοτική διαχείριση των δεδομένων ανάλογα με τις απαιτήσεις της εφαρμογής.

- **Εξωτερικά APIs / Υπηρεσίες / Συστήματα (External Services)**
  Το σύστημα αλληλεπιδρά με εξωτερικές υπηρεσίες και συστήματα για τη διασφάλιση της ασφάλειας και την απόκτηση δεδομένων:
  - Authentication API: Παρέχει έλεγχο ταυτότητας χρηστών μέσω OAuth.
  - CFP Websites (Εξωτερικές Ιστοσελίδες Συνεδρίων): Παρέχουν τα δεδομένα των συνεδρίων. Το σύστημα ανακτά και ενημερώνει πληροφορίες μέσω **εσωτερικού μηχανισμού crawling** που υλοποιείται από την υπηρεσία Crawling Service

Το **Frontend** επικοινωνεί με το **Backend** μέσω των **RESTful APIs** που παρέχει το τελευταίο, διασφαλίζοντας μια σαφή διαχωριστική γραμμή μεταξύ των δύο επιπέδων της αρχιτεκτονικής.

Η **επικοινωνία του Backend με τη βάση δεδομένων** δεν γίνεται μέσω της απευθείας εκτέλεσης επερωτήσεων (queries), αλλά υλοποιείται μέσω της **τεχνολογίας ORM (Object-Relational Mapping)**, η οποία εφαρμόζεται χρησιμοποιώντας την προγραμματιστική έννοια των **Repositories**. Τα τελευταία είναι ειδικά συστατικά που επιτρέπουν την αφηρημένη διαχείριση των δεδομένων (μέσω αντικειμένων), αποκρύπτοντας τις λεπτομέρειες αλληλεπίδρασης με την υποκείμενη βάση δεδομένων, βελτιώνοντας με αυτό τον τρόπο την ασφάλεια, αξιοπιστία και επεκτασιμότητα του συστήματος.

Η ενσωμάτωση εξωτερικών APIs και υπηρεσιών στο σύστημα πραγματοποιείται μέσω HTTP clients ή ειδικών connectors (SDKs), ανάλογα με τις απαιτήσεις της εκάστοτε (εξωτερικής) υπηρεσίας/API και το πρωτόκολλο επικοινωνίας που αυτή υποστηρίζει.



Εικόνα 3: Διάγραμμα Συστατικών του Συστήματος

4.2.3 Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram)

Ένα **διάγραμμα Περιπτώσεων Χρήσης** απεικονίζει τις βασικές λειτουργίες που παρέχει το σύστημα στους χρήστες του μέσω σχετικών αλληλεπιδράσεων υψηλού επιπέδου μεταξύ του συστήματος και των χρηστών. Στις αλληλεπιδράσεις αυτές ενδέχεται να συμμετέχουν και εξωτερικά συστήματα με υποστηρικτικό ρόλο στην υλοποίηση της σχετικής λειτουργίας (που αναπαρίσταται από την εκάστοτε αλληλεπίδραση).

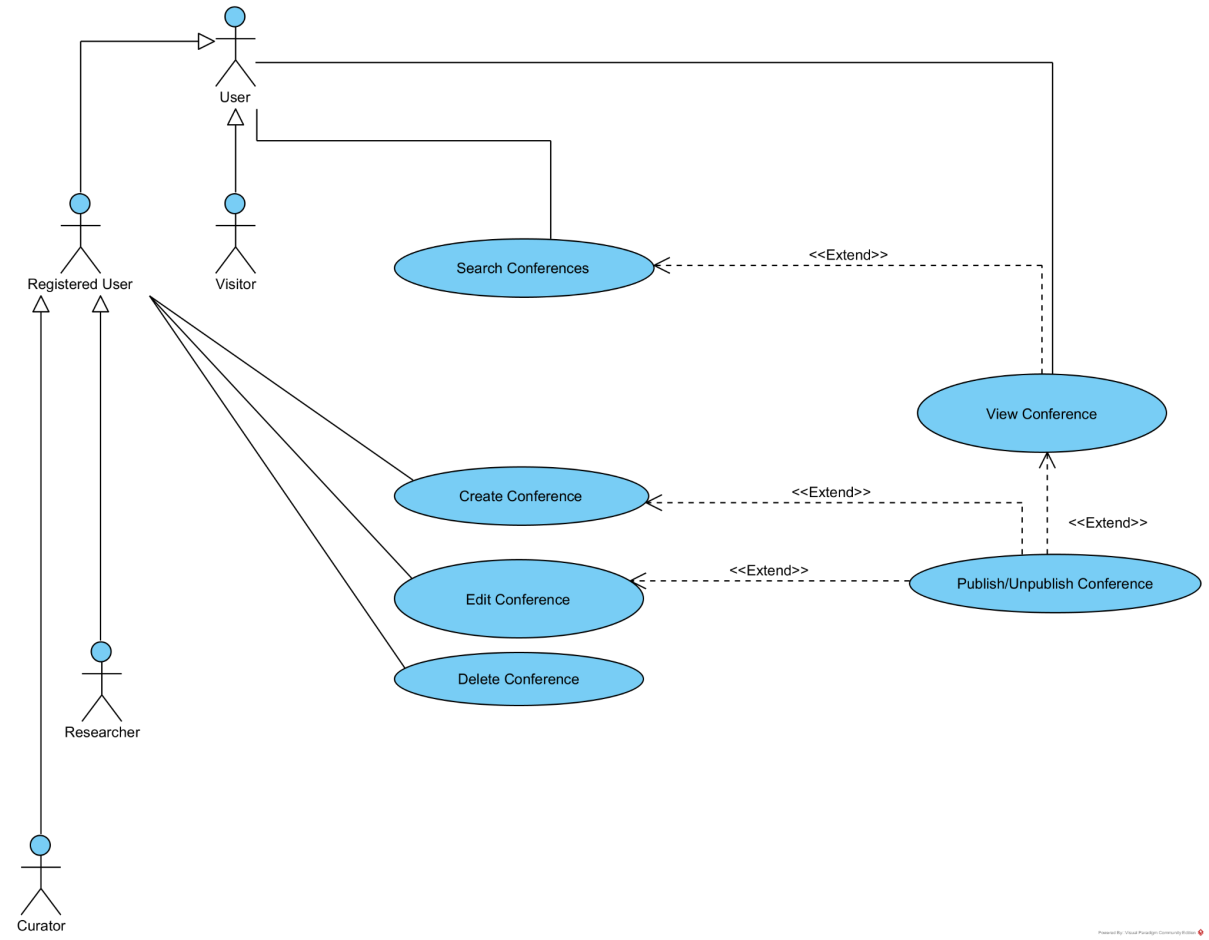
Παρακάτω παραθέτουμε τα διαγράμματα περιπτώσεων χρήσης που προέκυψαν από τη σχεδίαση του συστήματός μας, τα οποία εστιάζουν έκαστο στη διαχείριση συγκεκριμένης οντότητας (πχ. συνεδρίου, θέματος ή χρήστη). Επιπλέον, με περιγραφικό και δομικό τρόπο, οργανώνουμε τις περιπτώσεις χρήσης που περιλαμβάνονται στα διαγράμματα αυτά ανάλογα με το είδος χρήστη που αφορούν (ουσιαστικά το είδος χρήστη που τις εκκινεί και είναι ο βασικός αποδέκτης των λειτουργιών που αυτές αναπαριστούν).

Κύριες Περιπτώσεις Χρήσης:

- **Επισκέπτης (Visitor)**
    - Αναζήτηση συνεδρίων
    - Προβολή συνεδρίου
    - Εγγραφή στο σύστημα
- **Ερευνητής (Researcher)**
    - Σύνδεση
    - Διαχείριση λογαριασμού:
        - Ενημέρωση στοιχείων λογαριασμού
        - Διαγραφή λογαριασμού
        - Προβολή λογαριασμού
        - Ενημέρωση κωδικού πρόσβασης
        - Επαναφορά κωδικού πρόσβασης
    - Διαχείριση συνεδρίων:
        - Δημιουργία νέου συνεδρίου
        - Ενημέρωση συνεδρίου
        - Διαγραφή συνεδρίου
        - Δημοσίευση/Αποδημοσίευση συνεδρίου
        - Προβολή συνεδρίου
        - Αναζήτηση συνεδρίων
    - Διαχείριση θεμάτων:
        - Εισαγωγή θέματος
        - Εγγραφή (subscribe) σε θέματα
        - Διαγραφή από θέμα
        - Αναζήτηση θεμάτων
        - Ενημέρωση θέματος
        - Προβολή θέματος
    - Κοινωνική Δικτύωση:
        - Βαθμολόγηση συνεδρίων
        - Διαχείριση συζητήσεων
            - Προβολή συζήτησης
            - Δημιουργία συζήτησης

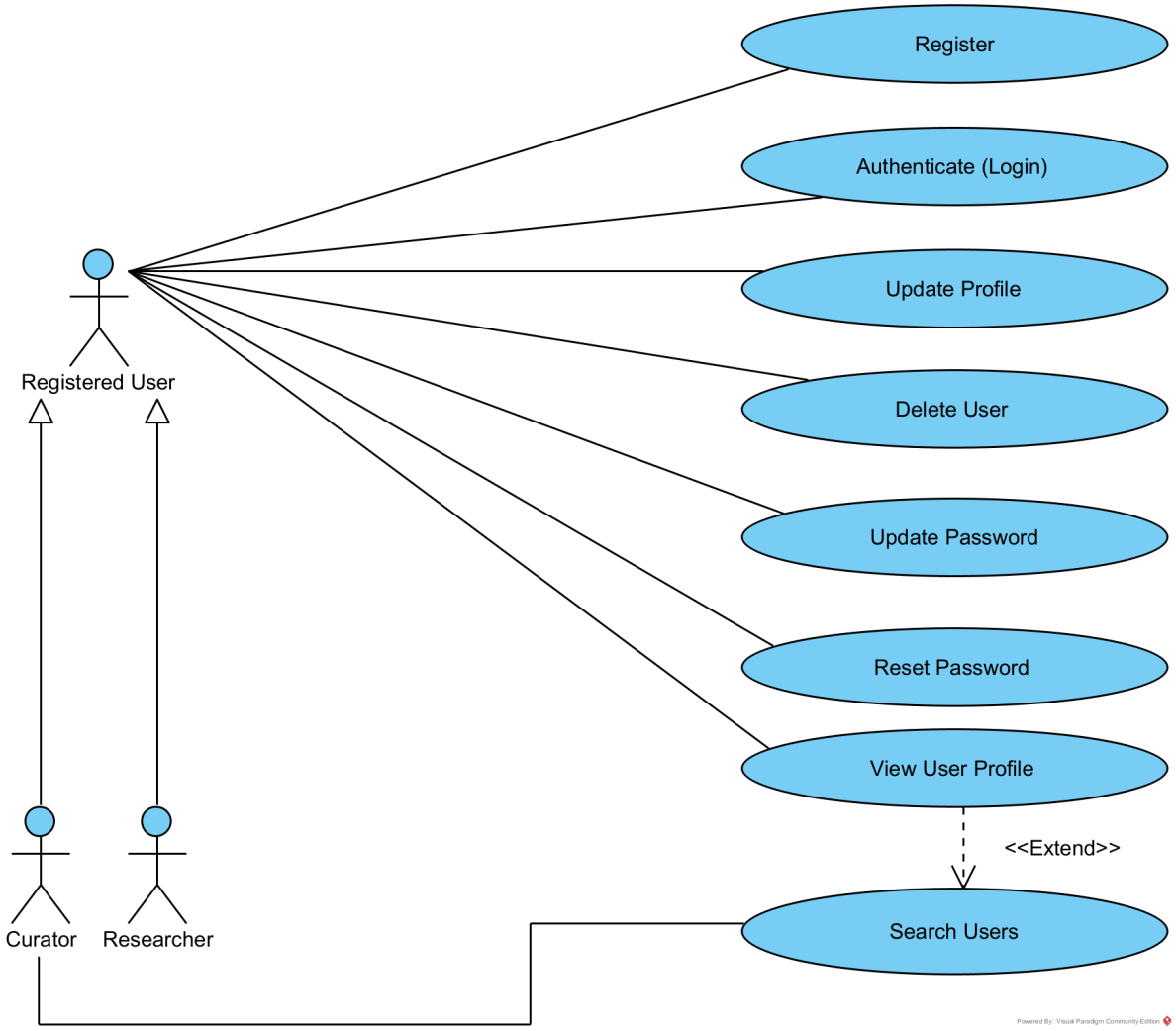
                - Αποστολή σχολείου σε συζήτηση
                - Διαγραφή συζήτησης
                - Αναζήτηση συζήτησης
                - Ενημέρωση συζήτησης
            - Διαχείριση φίλων
                - Προβολή λίστας φίλων
                - Υποβολή αίτησης φιλίας
                - Αποδοχή/Απόρριψη αίτησης φιλίας
                - Διαγραφή φιλικού χρήστη
- Επιμελητής (Curator)
    - Διαχείριση λογαριασμού:
        - Ενημέρωση στοιχείων λογαριασμού
        - Προβολή λογαριασμού
        - Ενημέρωση κωδικού πρόσβασης
        - Επαναφορά κωδικού πρόσβασης
    - Διαχείριση χρηστών:
        - Προβολή χρηστών
        - Αναζήτηση χρηστών
        - Διαγραφή χρηστών
        - Ενημέρωση χρηστών
        - Εισαγωγή χρήστη
    - Διαχείριση συνεδρίων:
        - Δημιουργία νέου συνεδρίου
        - Ενημέρωση συνεδρίου
        - Διαγραφή συνεδρίου
        - Δημοσίευση/Αποδημοσίευση συνεδρίου
        - Προβολή συνεδρίου
        - Αναζήτηση συνεδρίων
    - Διαχείριση θεμάτων:
        - Προβολή θεμάτων
        - Ενημέρωση θεμάτων
        - Διαγραφή θεμάτων
        - Δημιουργία θεμάτων
        - Αναζήτηση θεμάτων
        - Εγγραφή σε / διαγραφή από θέμα
    - Έλεγχος και επιμέλεια συζητήσεων/σχολίων
        - Τροποποίηση συζητήσεων/σχολίων
        - Διαγραφή συζητήσεων/σχολίων/αξιολογήσεων συνεδρίων μόνο αν υπάρχει παραβίαση κανόνων
    - Σύνδεση

Διάγραμμα Διαχείρισης Συνεδρίων



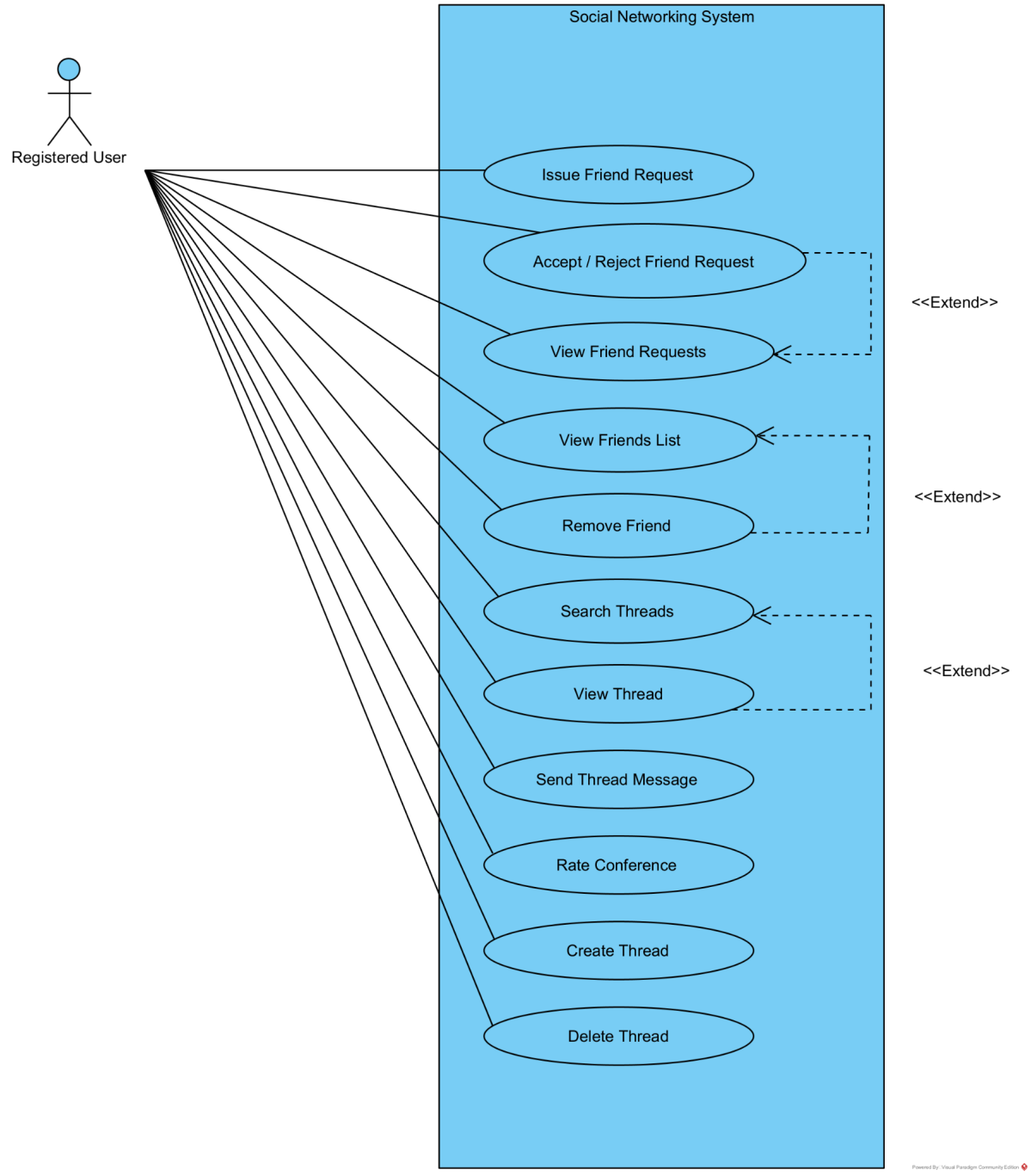
Εικόνα 4: Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Συνεδρίων

Διάγραμμα Διαχείρισης Λογαριασμού Χρήστη



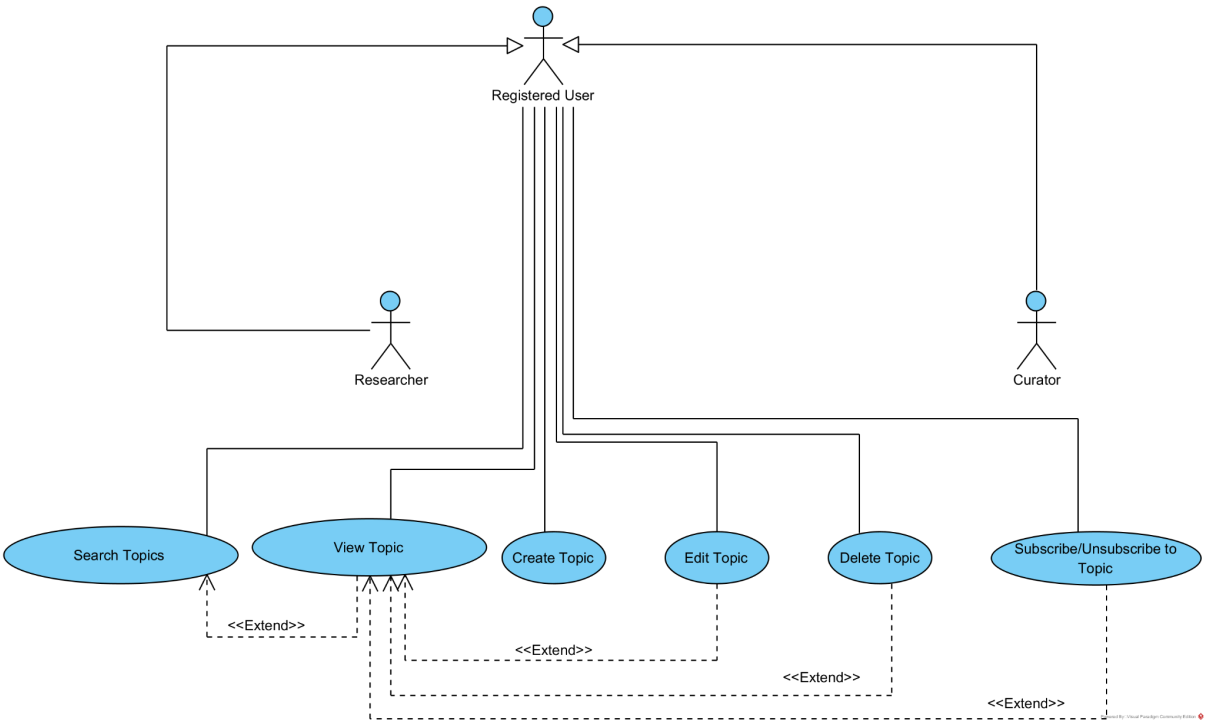
Εικόνα 5: Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Λογαριασμού Χρήστη

Διάγραμμα Κοινωνικής Δικτύωσης Χρηστών



Εικόνα 6: Διάγραμμα Περίπτωσης Χρήσης Κοινωνικής Δικτύωσης Χρηστών

Διάγραμμα Διαχείρισης Θεμάτων



Εικόνα 7: Διάγραμμα Περίπτωσης Χρήσης για τη Διαχείριση Θεμάτων

4.2.4 Διάγραμμα Ακολουθίας (Sequence Diagram)

Το **Διάγραμμα Ακολουθίας** περιγράφει τη ροή αλληλεπιδράσεων μεταξύ των συστατικών του συστήματος και των χρηστών, στο πλαίσιο μιας συγκεκριμένης περίπτωσης χρήσης. Στη συνέχεια παρουσιάζουμε με δομικό τρόπο πολλά από τα διαγράμματα ακολουθίας που προέκυψαν από τη σχεδίαση του συστήματός μας, οργανωμένα ανάλογα με την βασική οντότητα που αφορούν (συνέδριο, θέμα, κλπ.). Σημειώνεται πως για λόγους μειωμένης πολυπλοκότητας, τα διαγράμματα που παρουσιάζονται στη συνέχεια καλύπτουν τη βασική, θετική ροή για την κάθε περίπτωση χρήσης / βασική λειτουργία συστήματους που υλοποιούν. Συνεπώς, εναλλακτικές ή αρνητικές ροές (χειρισμού λαθών) δεν καλύπτονται.

4.2.4.1 Συνέδρια

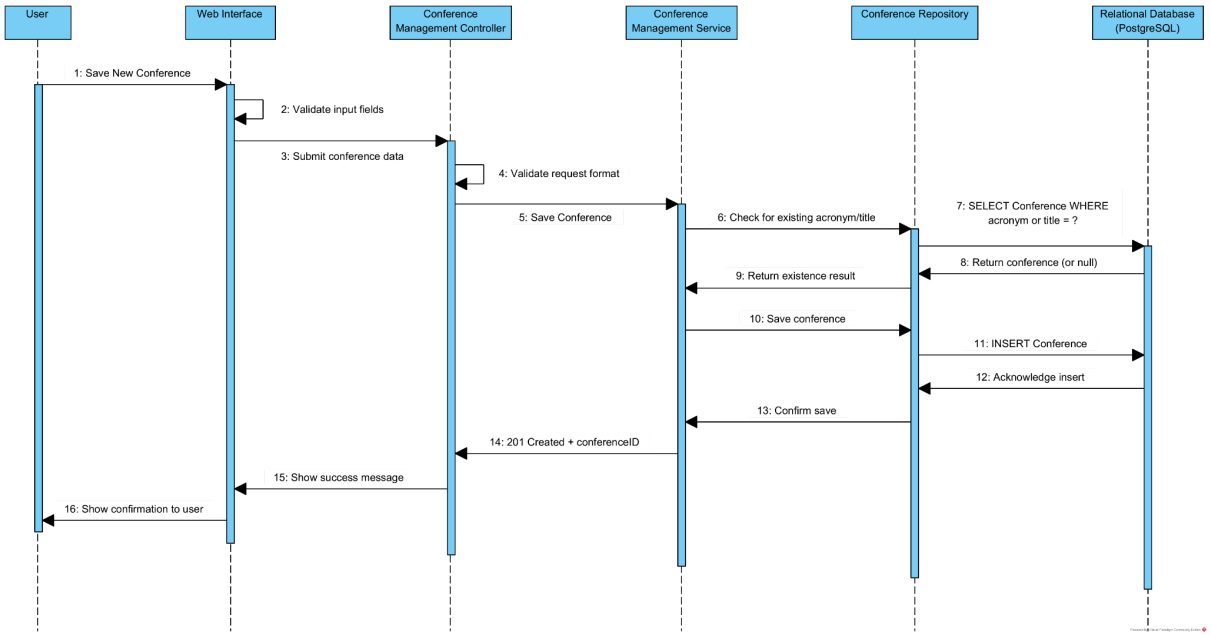
Δημιουργία Νέου Συνεδρίου

Η διαδικασία δημιουργίας συνεδρίου ξεκινά από τον χρήστη, ο οποίος επιλέγει να αποθηκεύσει ένα νέο συνέδριο. Τα δεδομένα που εισάγονται ελέγχονται αρχικά στο Web Interface για να διαπιστωθεί αν είναι σωστά συμπληρωμένα. Αν όλα είναι εντάξει, η πληροφορία αποστέλλεται στον Conference Management Controller, ο οποίος πραγματοποιεί έναν επιπλέον έλεγχο για τη μορφή και την πληρότητα των στοιχείων.

Στη συνέχεια, το αίτημα προωθείται στο Conference Management Service, όπου γίνεται έλεγχος αν υπάρχει ήδη συνέδριο με ίδιο τίτλο ή ακρωνύμιο. Ο έλεγχος αυτός γίνεται μέσω του Conference Repository, το οποίο διαβάζει από τη βάση δεδομένων τη σχετική

πληροφορία, εφόσον υπάρχει. Αν δεν βρεθεί αντίστοιχο συνέδριο, τα δεδομένα καταχωρίζονται στη βάση.

Μόλις ολοκληρωθεί επιτυχώς η αποθήκευση, επιστρέφεται από το σύστημα μήνυμα επιβεβαίωσης, το οποίο περιλαμβάνει και το μοναδικό αναγνωριστικό του συνεδρίου (`conferenceID`). Το μήνυμα εμφανίζεται στη διεπαφή χρήστη προς ενημέρωση του αιτούντος χρήστη.



Εικόνα 8: Διάγραμμα Ακολουθίας Δημιουργίας Νέου Συνεδρίου

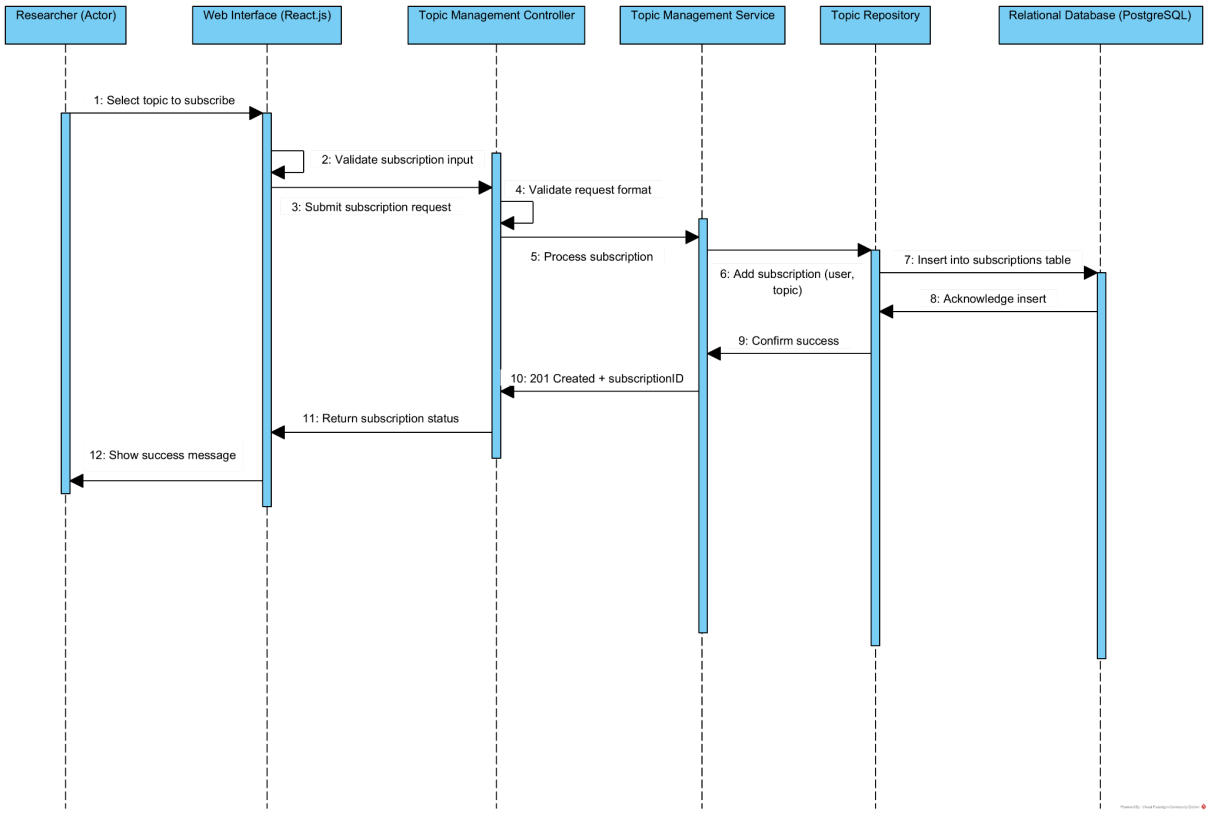
Εγγραφή Ερευνητή σε Θέμα

Η διαδικασία εγγραφής χρήστη σε ένα θέμα (topic) ξεκινά όταν ο χρήστης επιλέγει κάποιο διαθέσιμο θέμα από τη σχετική λίστα στην πλατφόρμα. Η επιλογή αυτή ενεργοποιεί τη διεπαφή (Web Interface), η οποία ελέγχει τα δεδομένα της επιλογής και διασφαλίζει ότι η αίτηση είναι έγκυρη.

Εφόσον όλα είναι σωστά, η αίτηση εγγραφής αποστέλλεται στον Topic Management Controller, ο οποίος εκτελεί επιπλέον έλεγχο στη δομή του αιτήματος και στη συνέχεια το προωθεί στην αντίστοιχη υπηρεσία (Topic Management Service). Η υπηρεσία από τη μεριά της αναλαμβάνει να επεξεργαστεί την εγγραφή και να τη μετατρέψει σε εντολή προς το αντίστοιχο repository για την αποθήκευση της σχέσης (εγγραφής) χρήστη–θέματος.

Το repository εκτελεί το σχετικό INSERT στη βάση δεδομένων, ενημερώνοντας τον πίνακα των εγγραφών (subscriptions). Εφόσον η εισαγωγή γίνει με επιτυχία, η βάση επιστρέφει επιβεβαίωση. Η τελευταία φθάνει στη σχετική υπηρεσία (Topic Management Service), η οποία περιλαμβάνει στην απάντηση επιτυχίας (201 Created) το μοναδικό `subscriptionID` της καταχωρημένης σχέσης εγγραφής. Η απάντηση αυτή διαβιβάζεται πίσω στον controller και από εκεί στη διεπαφή του χρήστη (Web Interface).

Τέλος, ο χρήστης ενημερώνεται στην οθόνη ότι η εγγραφή του στο επιλεγμένο θέμα ολοκληρώθηκε επιτυχώς.



Εικόνα 9: Διάγραμμα Ακολουθίας Εγγραφής Ερευνητή σε Θέμα

4.2.4.2 Χρήστες

Είσοδος Χρήστη / Αυθεντικοποίηση

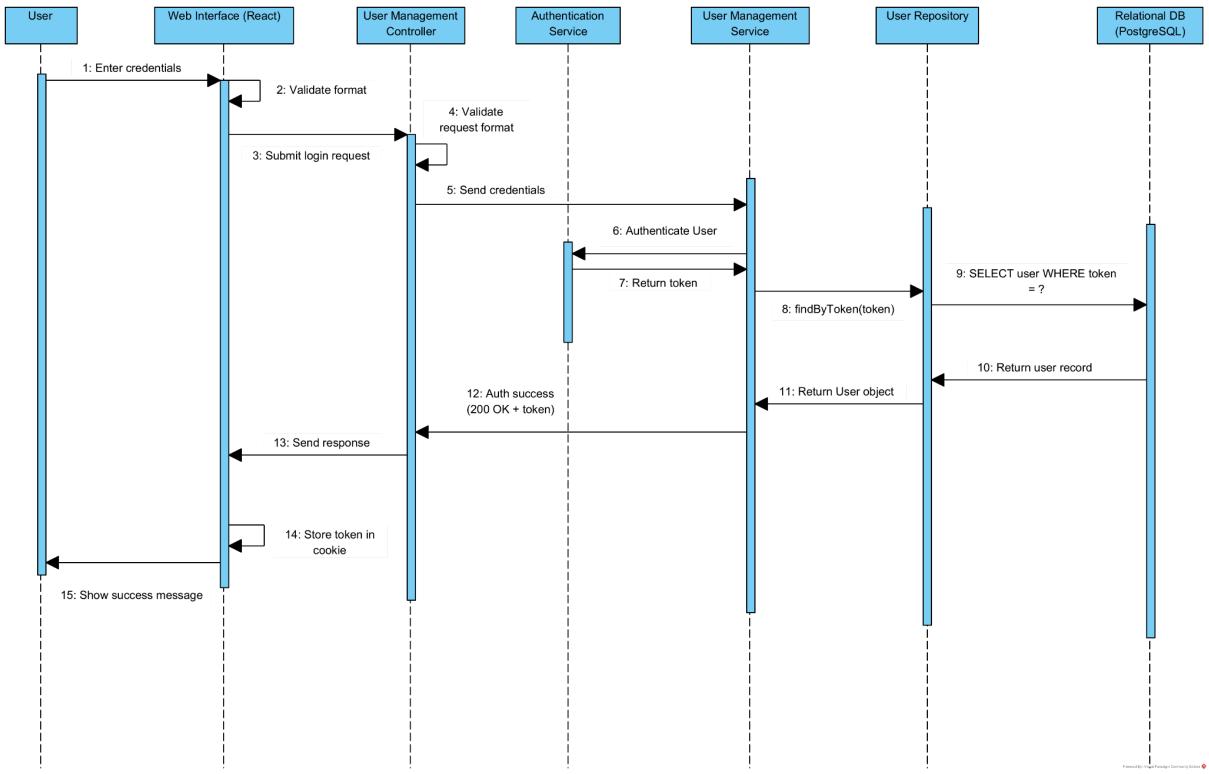
Η διαδικασία εισόδου στο σύστημα ξεκινά όταν ο χρήστης συμπληρώνει το όνομα χρήστη και τον κωδικό πρόσβασης στη φόρμα σύνδεσης. Τα στοιχεία που καταχωρεί ελέγχονται πρώτα τοπικά στο Web Interface για να διαπιστωθεί αν έχουν τη σωστή μορφή (π.χ. δεν είναι κενά, έχουν σωστό μήκος κ.λπ.).

Εφόσον τα στοιχεία είναι έγκυρα, η αίτηση σύνδεσης προωθείται στον User Management Controller, ο οποίος ελέγχει τη δομή του αιτήματος και στέλνει τα στοιχεία σύνδεσης στην (εξωτερική) υπηρεσία Authentication Service. Εκεί γίνεται ο βασικός έλεγχος ταυτοποίησης, δηλαδή ελέγχονται εάν τα στοιχεία του χρήστη είναι σωστά. Αν η ταυτοποίηση πετύχει, η υπηρεσία επιστρέφει ένα ειδικό αναγνωριστικό (token), το οποίο χρησιμοποιείται για να συνεχιστεί η διαδικασία.

Το token προωθείται στην υπηρεσία User Management Service, η οποία αναζητά το αντίστοιχο προφίλ χρήστη μέσω του User Repository. Το repository εκτελεί σχετικό ερώτημα

προς τη βάση δεδομένων και επιστρέφει τα δεδομένα του χρήστη που αντιστοιχούν στο συγκεκριμένο token.

Αφού ολοκληρωθεί με επιτυχία η ταυτοποίηση και ληφθούν τα στοιχεία του χρήστη, επιστρέφεται στο frontend ένα μήνυμα επιτυχίας (HTTP 200 OK) μαζί με το token. Το token αποθηκεύεται στον browser του χρήστη (π.χ. σε cookie) και το σύστημα εμφανίζει κατάλληλο μήνυμα που επιβεβαιώνει την επιτυχή είσοδο του χρήστη.



Εικόνα 10: Διάγραμμα Ακολουθίας Εισόδου Χρήστη και Αυθεντικοποίησης

Εγγραφή Χρήστη

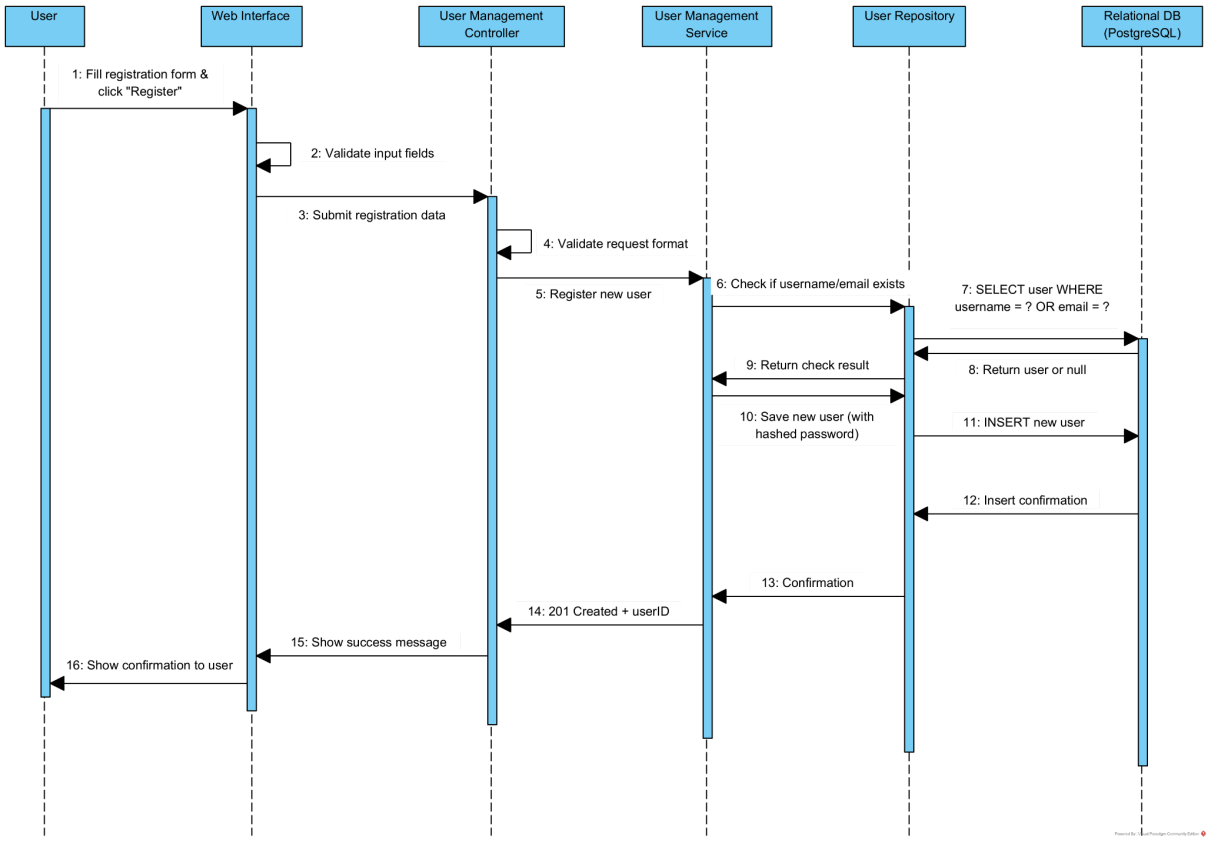
Η διαδικασία εγγραφής ξεκινά όταν ένας χρήστης συμπληρώνει τη φόρμα εγγραφής με τα βασικά στοιχεία του, όπως όνομα χρήστη, email και κωδικό πρόσβασης. Η εισαγωγή των δεδομένων γίνεται από το Web Interface, το οποίο αρχικά ελέγχει τοπικά τη μορφή και την πληρότητα των παρεχόμενων δεδομένων.

Εφόσον όλα τα δεδομένα είναι έγκυρα, αποστέλλονται στον User Management Controller για επιπλέον έλεγχο. Στη συνέχεια, το αίτημα προωθείται στο User Management Service, το οποίο ελέγχει πρώτα αν υπάρχει ήδη χρήστης με το ίδιο username ή email. Ο έλεγχος αυτός γίνεται μέσω του User Repository, το οποίο εκτελεί σχετικό ερώτημα στη βάση δεδομένων.

Αν δεν εντοπιστεί ήδη καταχωρημένος χρήστης, τότε το User Service προχωρά στην αποθήκευση του νέου λογαριασμού. Ο κωδικός του χρήστη αποθηκεύεται με ασφάλεια, αφού πρώτα γίνει (κρυπτογραφικός) κατακερματισμός του (hashing). Τα δεδομένα

καταχωρούνται στη βάση και επιστρέφεται επιβεβαίωση ότι η εγγραφή ολοκληρώθηκε επιτυχώς, μαζί με το μοναδικό αναγνωριστικό του νέου χρήστη (`userID`).

Το τελικό μήνυμα επιτυχίας προβάλλεται στον χρήστη μέσω της διεπαφής και τον ενημερώνει ότι η εγγραφή του στο σύστημα ολοκληρώθηκε.



Εικόνα 11: Διάγραμμα Ακολουθίας Εγγραφής Χρήστη

4.2.4.3 Θέματα

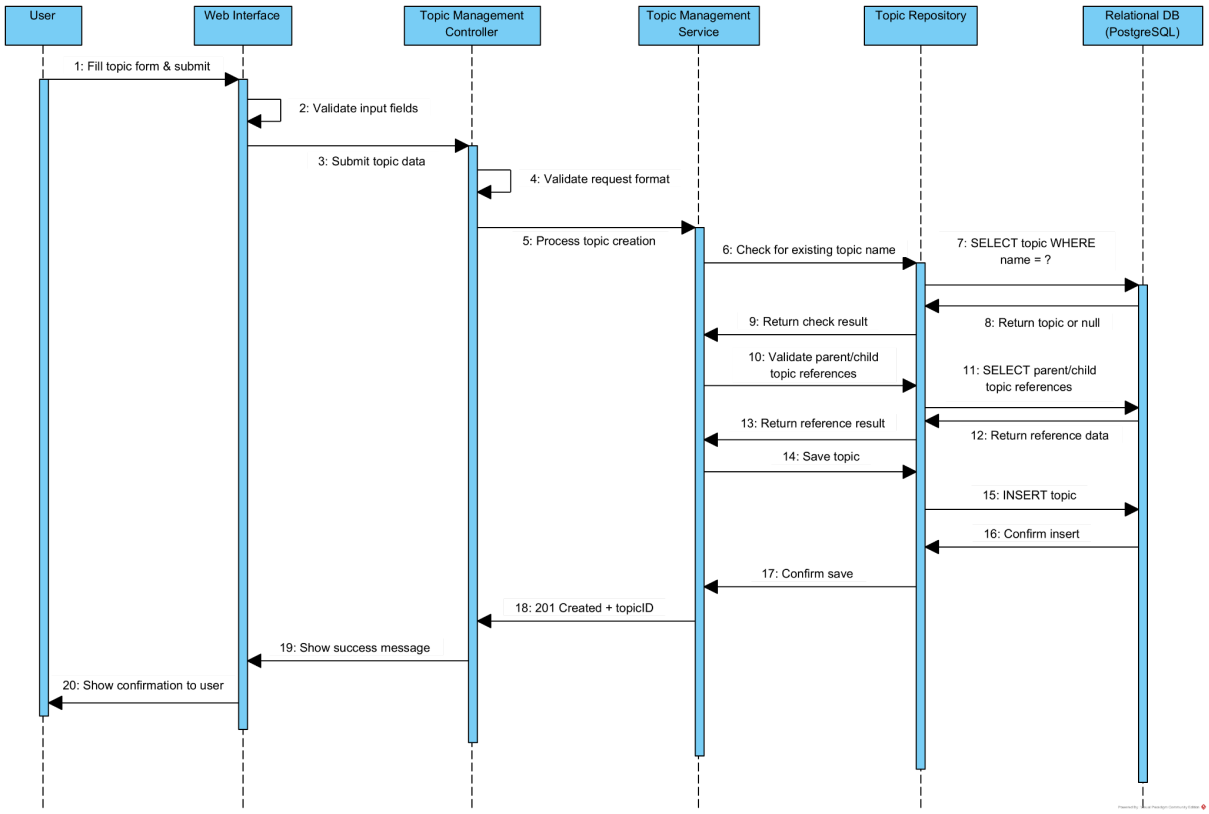
Εισαγωγή Νέου Θέματος

Η διαδικασία δημιουργίας νέου θέματος ξεκινά όταν ο χρήστης συμπληρώνει μια φόρμα με τα στοιχεία του θέματος, όπως το όνομα, η περιγραφή και, εφόσον υπάρχει, η συσχέτιση με πατρικό ή θυγατρικό θέμα. Αρχικά, το Web Interface πραγματοποιεί έναν βασικό έλεγχο στα δεδομένα για να διασφαλίσει ότι είναι σωστά συμπληρωμένα.

Εφόσον όλα είναι εντάξει, η αίτηση αποστέλλεται στον Topic Management Controller, ο οποίος ελέγχει επιπλέον τη δομή της πληροφορίας και τη διαβιβάζει στο Topic Management Service για επεξεργασία. Το service αρχικά ελέγχει αν υπάρχει ήδη καταχωρημένο θέμα με το ίδιο όνομα, χρησιμοποιώντας το Topic Repository που προχωρά στη σχετική επερώτηση στη βάση.

Αν δεν υπάρχει ήδη ίδιο θέμα, το σύστημα προχωρά στον έλεγχο των σχέσεων με πατρικά και θυγατρικά θέματα για να διασφαλίσει ότι οι σχετικές αναφορές είναι έγκυρες. Εφόσον όλα είναι σωστά, το νέο θέμα αποθηκεύεται στη βάση δεδομένων.

Μόλις ολοκληρωθεί η αποθήκευση, το σύστημα δημιουργεί ένα μοναδικό αναγνωριστικό (`topicID`) και το περιλαμβάνει στην απάντηση. Ο χρήστης ενημερώνεται μέσω του Web Interface ότι η καταχώρηση του θέματος ολοκληρώθηκε με επιτυχία.



Εικόνα 12: Διάγραμμα Ακολουθίας Εισαγωγής Νέου Θέματος

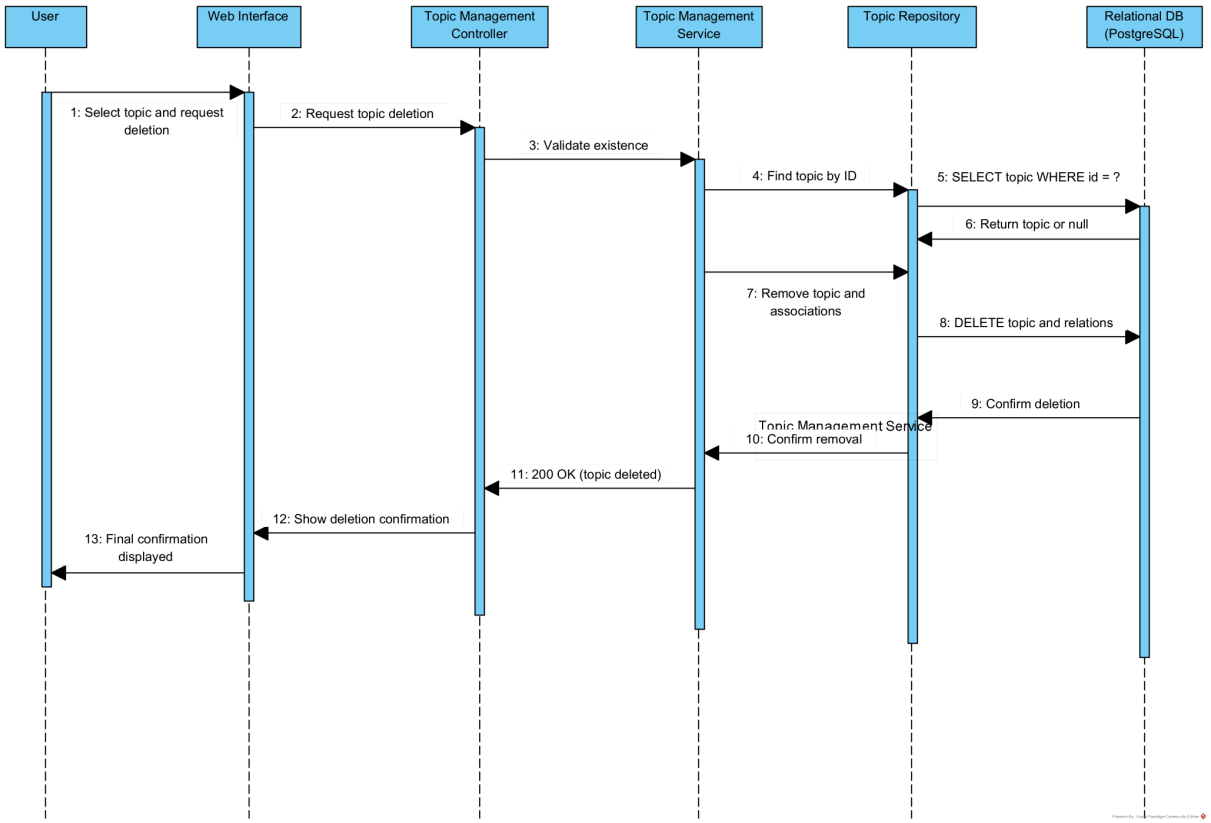
Διαγραφή Θέματος

Η διαδικασία ξεκινά όταν ο χρήστης επιλέγει να διαγράψει ένα συγκεκριμένο θέμα από το σύστημα. Μέσα από το περιβάλλον του Web Interface, η αίτηση αυτή υποβάλλεται και ελέγχεται αν τα στοιχεία είναι σωστά συμπληρωμένα.

Εφόσον η αίτηση είναι έγκυρη, προωθείται στον Topic Management Controller, ο οποίος αναλαμβάνει να επικυρώσει περαιτέρω την εγκυρότητά της. Κατόπιν, η πληροφορία περνά στο Topic Management Service, το οποίο χρησιμοποιεί το Topic Repository για να εντοπίσει αν το συγκεκριμένο θέμα υπάρχει πραγματικά στη βάση δεδομένων.

Εφόσον το θέμα εντοπιστεί, τότε το σύστημα προχωρά στη διαγραφή του μαζί με τις σχετικές του συσχετίσεις (όπως συνδέσεις με πατρικά ή θυγατρικά θέματα). Η διαγραφή εκτελείται στη βάση, η οποία επιστρέφει επιβεβαίωση.

Μετά την ολοκλήρωση της ενέργειας, το σύστημα ενημερώνει τον χρήστη ότι η διαγραφή πραγματοποιήθηκε με επιτυχία και το αντίστοιχο μήνυμα εμφανίζεται στην οθόνη του.



Εικόνα 13: Διάγραμμα Ακολουθίας Διαγραφής Θέματος

4.2.4.4 Social Media

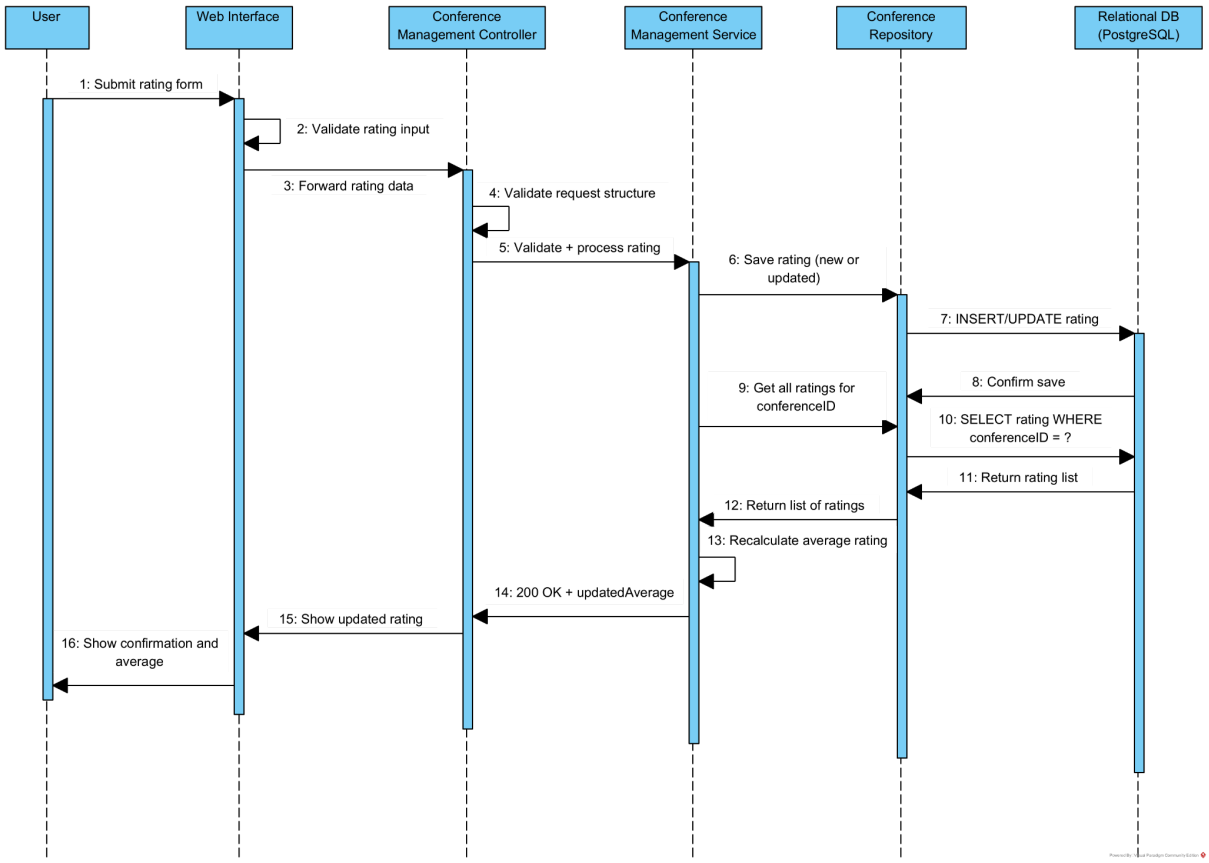
Βαθμολόγηση Συνεδρίου

Η διαδικασία ξεκινά όταν ο χρήστης υποβάλει μια βαθμολογία για κάποιο συνέδριο μέσω της φόρμας αξιολόγησης στο Web Interface. Τα στοιχεία που έχει συμπληρώσει ελέγχονται τοπικά ώστε να διασφαλιστεί ότι είναι σωστά και πλήρη.

Αφού περάσει ο έλεγχος, η αίτηση προωθείται στον Conference Management Controller, ο οποίος ελέγχει περαιτέρω τη δομή των δεδομένων και προωθεί τη βαθμολογία στο Conference Management Service. Εκεί, αποθηκεύεται είτε ως νέα βαθμολογία είτε ως ενημέρωση παλαιότερης.

Στη συνέχεια, το Conference Management Service ζητά από το Conference Repository όλες τις βαθμολογίες που σχετίζονται με το ίδιο συνέδριο και τις ανακτά από τη βάση δεδομένων. Με βάση αυτές, υπολογίζεται εκ νέου ο μέσος όρος της βαθμολογίας του συνεδρίου.

Το αποτέλεσμα επιστρέφεται στον χρήστη με ένα μήνυμα επιτυχίας, το οποίο τον ενημερώνει και για τον νέο μέσο όρο.



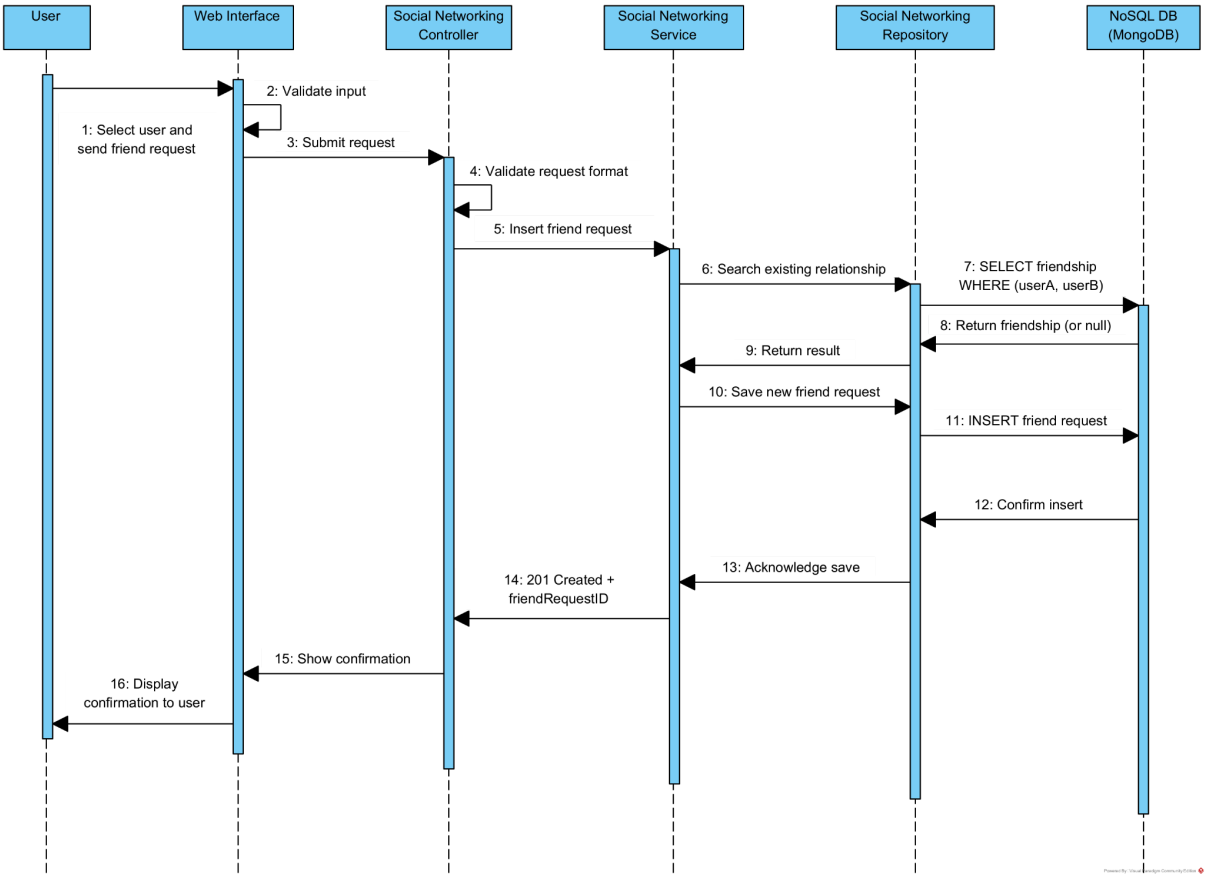
Εικόνα 14: Διάγραμμα Ακολουθίας Βαθμολόγησης Συνεδρίου

Αποστολή Αιτήματος Φιλίας

Ο χρήστης επιλέγει έναν άλλο χρήστη και ξεκινά τη διαδικασία αποστολής αιτήματος φιλίας μέσω της διεπαφής χρήστη (Web Interface). Το αίτημα αυτό ελέγχεται αρχικά για τη σωστή μορφή των δεδομένων από τη διεπαφή χρήστη και προωθείται στον Social Networking Controller, ο οποίος εκτελεί επιπλέον επικυρώσεις.

Στη συνέχεια, το αίτημα προωθείται στο Social Networking Service, το οποίο ελέγχει εάν υπάρχει ήδη σχέση φιλίας ή εκκρεμές αίτημα μεταξύ των δύο χρηστών. Ο έλεγχος αυτός πραγματοποιείται μέσω του Social Networking Repository, το οποίο προχωρά στη σχετική αναζήτηση στη βάση δεδομένων.

Εφόσον δεν υπάρχει προηγούμενη σχέση, δημιουργείται νέα εγγραφή με το αίτημα φιλίας. Μετά την επιβεβαίωση εισαγωγής στη βάση, ένα μήνυμα επιβεβαίωσης εμφανίζεται στον χρήστη.



Εικόνα 15: Διάγραμμα Ακολουθίας Αποστολής Αιτήματος Φιλίας

4.2.5 Διάγραμμα Δραστηριοτήτων (Activity Diagram)

Το **Διάγραμμα Δραστηριοτήτων** απεικονίζει τη ροή εργασιών στο πλαίσιο μιας συγκεκριμένης βασικής λειτουργίας του συστήματος (κι άρα περίπτωσης χρήσης). Στη συνέχεια, παρουσιάζουμε ορισμένα διαγράμματα δραστηριοτήτων του συστήματός μας, τα οποία οργανώνονται με βάση τη βασική οντότητα συστήματος που αφορούν.

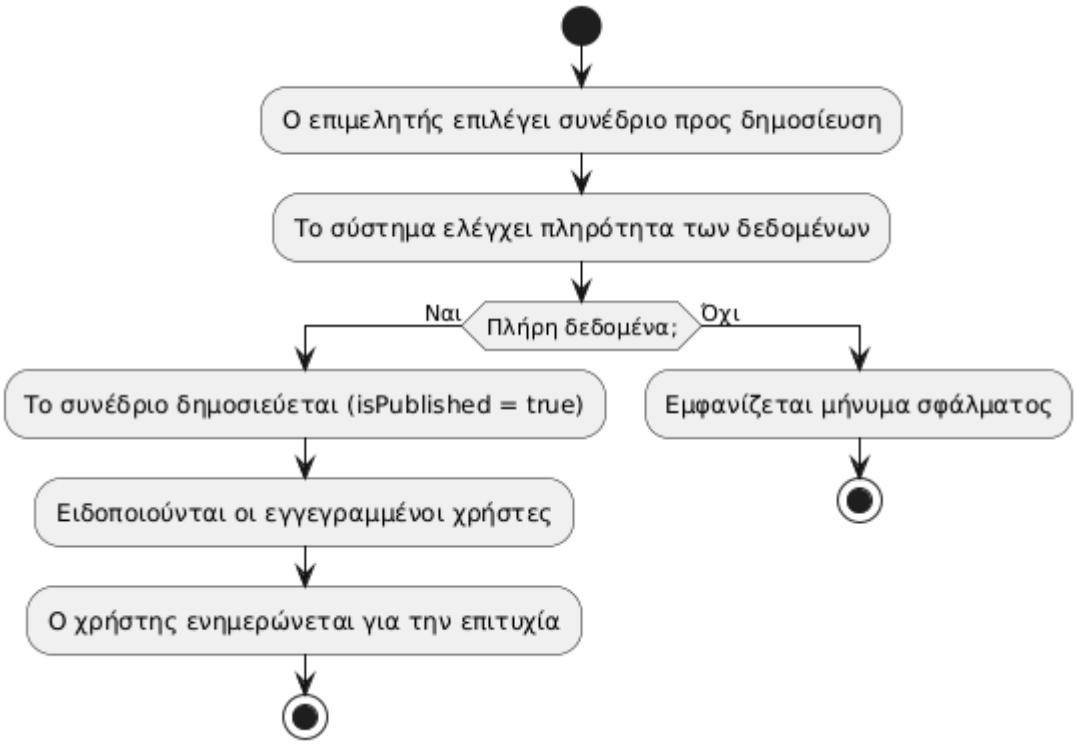
4.2.5.1 Συνέδριο

Δημοσίευση Συνεδρίου από Επιμελητή (Curator)

Ο επιμελητής επιλέγει ένα συνέδριο με σκοπό να το δημοσιεύσει. Το σύστημα ελέγχει εάν έχουν συμπληρωθεί όλα τα απαιτούμενα πεδία για την ολοκλήρωση της διαδικασίας.

Αν τα δεδομένα είναι πλήρη και έγκυρα, το συνέδριο δημοσιεύεται, δηλαδή η κατάστασή του αλλάζει ώστε να θεωρείται διαθέσιμο (`isPublished = true`). Κατόπιν, αποστέλλονται ειδοποιήσεις σε όλους τους χρήστες που είναι εγγεγραμμένοι σε θέματα σχετικά με το δημοσιευμένο συνέδριο, ώστε να ενημερωθούν για τη νέα δημοσίευση. Τέλος, ο επιμελητής ενημερώνεται για την επιτυχή ολοκλήρωση της διαδικασίας.

Εφόσον τα δεδομένα δεν είναι επαρκή, εμφανίζεται σχετικό μήνυμα σφάλματος.



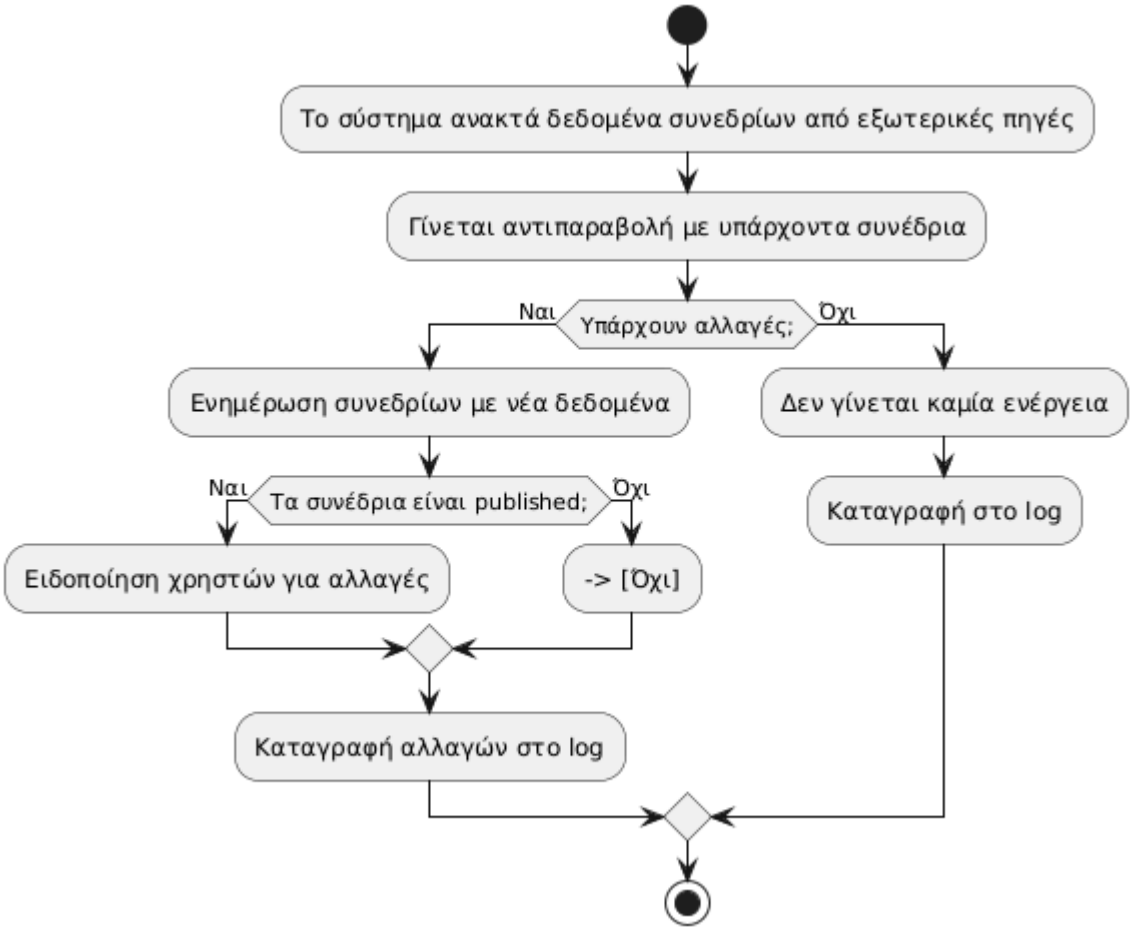
Εικόνα 16: Διάγραμμα Δραστηριοτήτων Δημοσίευσης Συνεδρίου από Επιμελητή

Αυτόματη Ενημέρωση Συνεδρίων μέσω Crawling

Το σύστημα εκτελεί προγραμματισμένα (scheduled task) μια διαδικασία ανάκτησης δεδομένων συνεδρίων από εξωτερικές πηγές. Τα δεδομένα που λαμβάνονται συγκρίνονται με τις ήδη καταχωρημένες εγγραφές για να εντοπιστούν τυχόν διαφορές.

Εφόσον εντοπιστούν αλλαγές, το σύστημα ενημερώνει τις εγγραφές των συνεδρίων με τα νέα δεδομένα. Στη συνέχεια, αν το κάθε συνέδριο είναι ήδη δημοσιευμένο, τότε αποστέλλονται σχετικές ειδοποιήσεις στους χρήστες που είναι εγγεγραμμένοι στα θέματα με τα οποία σχετίζεται το συνέδριο. Σε κάθε περίπτωση, οι αλλαγές που πραγματοποιήθηκαν καταγράφονται στο αρχείο καταγραφής (log).

Αν δεν εντοπιστούν αλλαγές, δεν πραγματοποιείται καμία τροποποίηση, και καταγράφεται στο log ότι δεν υπήρξε ανάγκη για ενημέρωση.



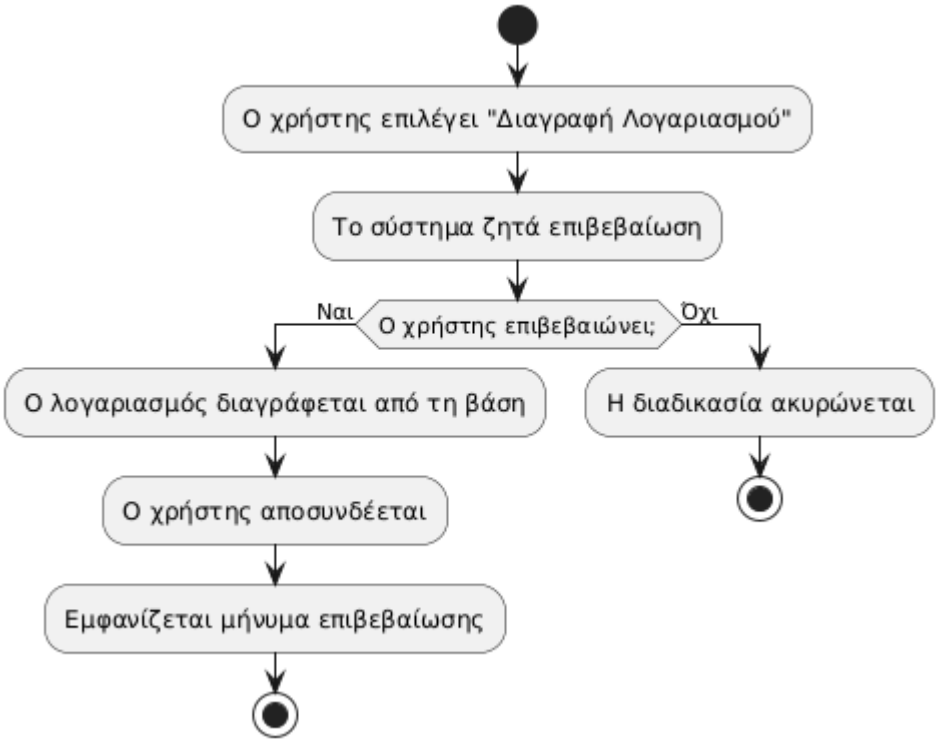
Εικόνα 17: Διάγραμμα Δραστηριοτήτων Αυτόματης Ενημέρωσης Συνεδρίων μέσω Crawling

4.2.5.2 Χρήστες

Διαγραφή Λογαριασμού Χρήστη

Ο χρήστης επιλέγει τη λειτουργία «Διαγραφή Λογαριασμού» μέσω της διεπαφής. Το σύστημα ζητά επιβεβαίωση για την εκτέλεση της ενέργειας. Εφόσον ο χρήστης ακυρώσει, η διαδικασία σταματά και δεν γίνεται καμία ενέργεια.

Εναλλακτικά, εφόσον ο χρήστης επιβεβαιώσει, ο λογαριασμός του διαγράφεται από τη βάση δεδομένων. Στη συνέχεια, το σύστημα πραγματοποιεί αυτόματη αποσύνδεση του χρήστη και εμφανίζει σχετικό μήνυμα επιβεβαίωσης για την επιτυχή ολοκλήρωση της διαδικασίας.

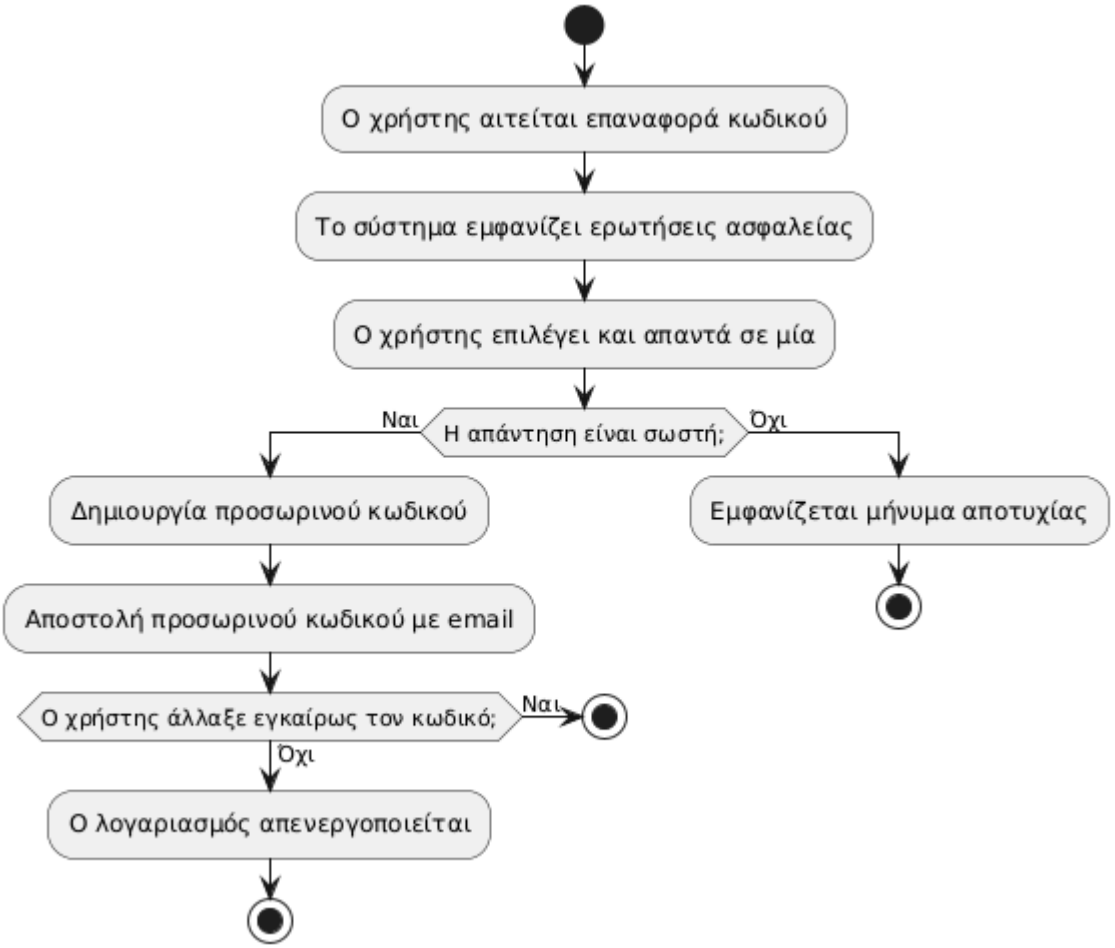


Εικόνα 18: Διάγραμμα Δραστηριοτήτων Διαγραφής Λογαριασμού Χρήστη

Επαναφορά Κωδικού Χρήστη

Ο χρήστης ζητά την επαναφορά του κωδικού πρόσβασης. Το σύστημα προβάλλει ερωτήσεις ασφαλείας, από τις οποίες ο χρήστης επιλέγει μία και την απαντά. Εφόσον η απάντηση είναι λανθασμένη, εμφανίζεται μήνυμα αποτυχίας και η διαδικασία ολοκληρώνεται.

Διαφορετικά, αν η απάντηση είναι σωστή, δημιουργείται προσωρινός κωδικός και αποστέλλεται στο email του χρήστη. Ο χρήστης καλείται να αλλάξει τον κωδικό μέσα σε συγκεκριμένο χρονικό διάστημα. Αν δεν το κάνει εγκαίρως, ο λογαριασμός του απενεργοποιείται για λόγους ασφαλείας.



Εικόνα 19: Διάγραμμα Δραστηριοτήτων Επαναφοράς Κωδικού Χρήστη

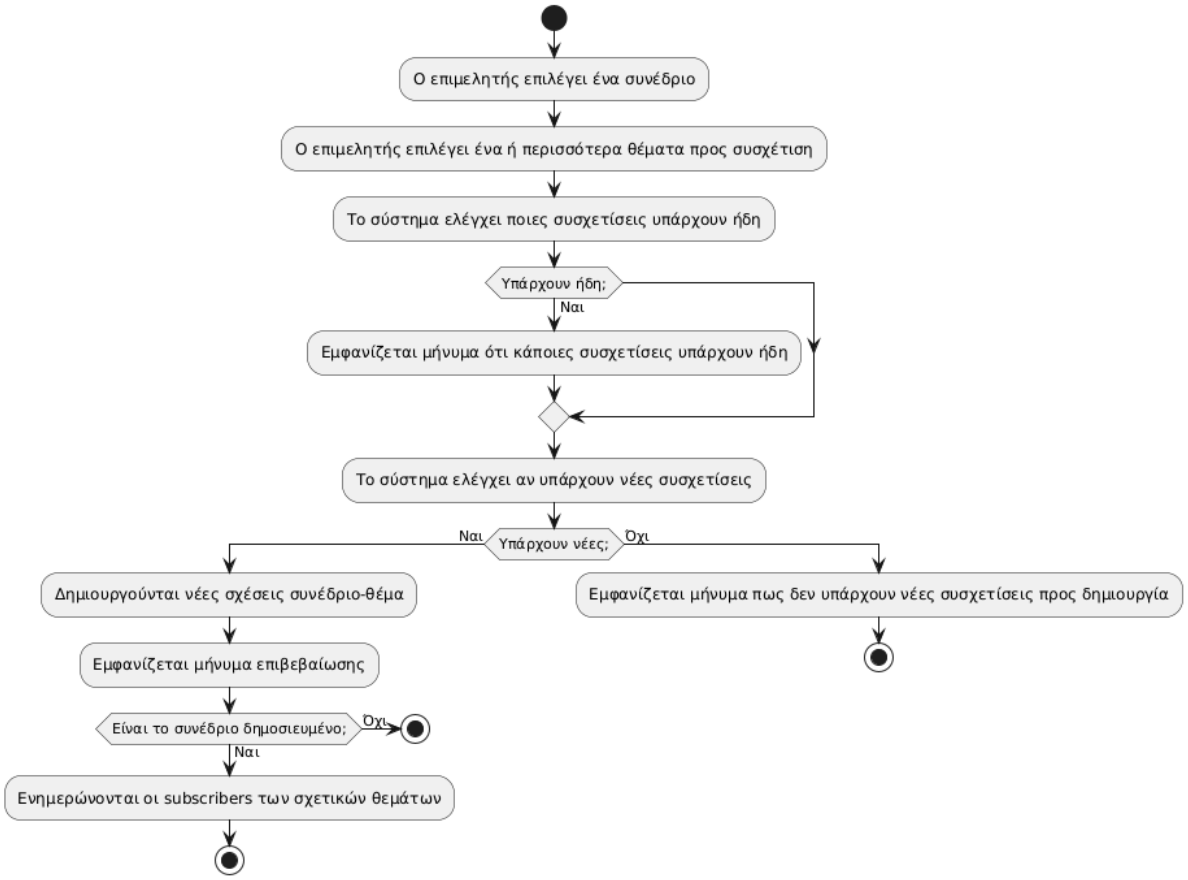
4.2.5.3 Θέματα

Συσχέτιση Θέματος με Συνέδριο από Επιμελητή

Ο επιμελητής επιλέγει το συνέδριο και ένα ή περισσότερα θέματα που επιθυμεί να συσχετίσει. Το σύστημα ελέγχει ποιες από τις επιλεγμένες συσχετίσεις υπάρχουν ήδη. Αν υπάρχουν, εμφανίζεται σχετικό μήνυμα.

Στη συνέχεια, ελέγχεται αν υπάρχουν νέες συσχετίσεις. Εφόσον δεν υπάρχουν, ενημερώνεται ο χρήστης και η διαδικασία ολοκληρώνεται. Διαφορετικά, αν υπάρχουν, αυτές δημιουργούνται και εμφανίζεται μήνυμα επιβεβαίωσης.

Τέλος, εφόσον το συνέδριο έχει ήδη δημοσιευθεί, ειδοποιούνται όλοι οι χρήστες που έχουν εγγραφεί στα θέματα με τα οποία σχετίστηκε τώρα το συνέδριο.

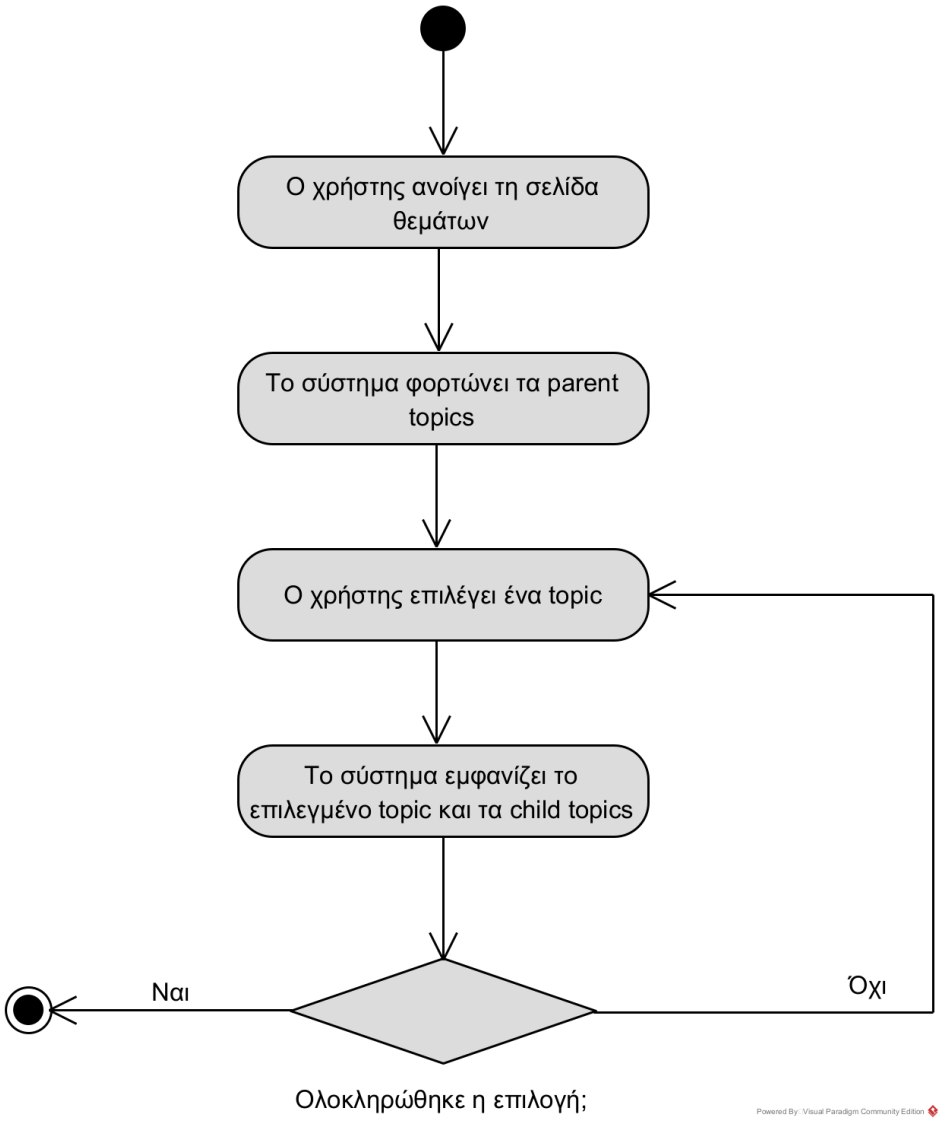


Εικόνα 20: Διάγραμμα Δραστηριοτήτων Συσχέτισης Θέματος με Συνέδριο από Επιμελητή

Θέαση Θέματος

Ο χρήστης μεταβαίνει στη σελίδα των θεμάτων, όπου το σύστημα φορτώνει αρχικά τις θεματικές ενότητες που δεν έχουν γονικό θέμα (parent topics). Ο χρήστης επιλέγει μία από αυτές και το σύστημα εμφανίζει το επιλεγμένο θέμα μαζί με τα υπο-θέματά (child topics) του, εφόσον υπάρχουν.

Η διαδικασία επαναλαμβάνεται όσο ο χρήστης επιθυμεί να συνεχίσει την πλοήγηση σε υπο-θεματικές ενότητες. Όταν θεωρήσει πως έχει φτάσει στο θέμα που τον ενδιαφέρει, ολοκληρώνει την πλοήγηση.



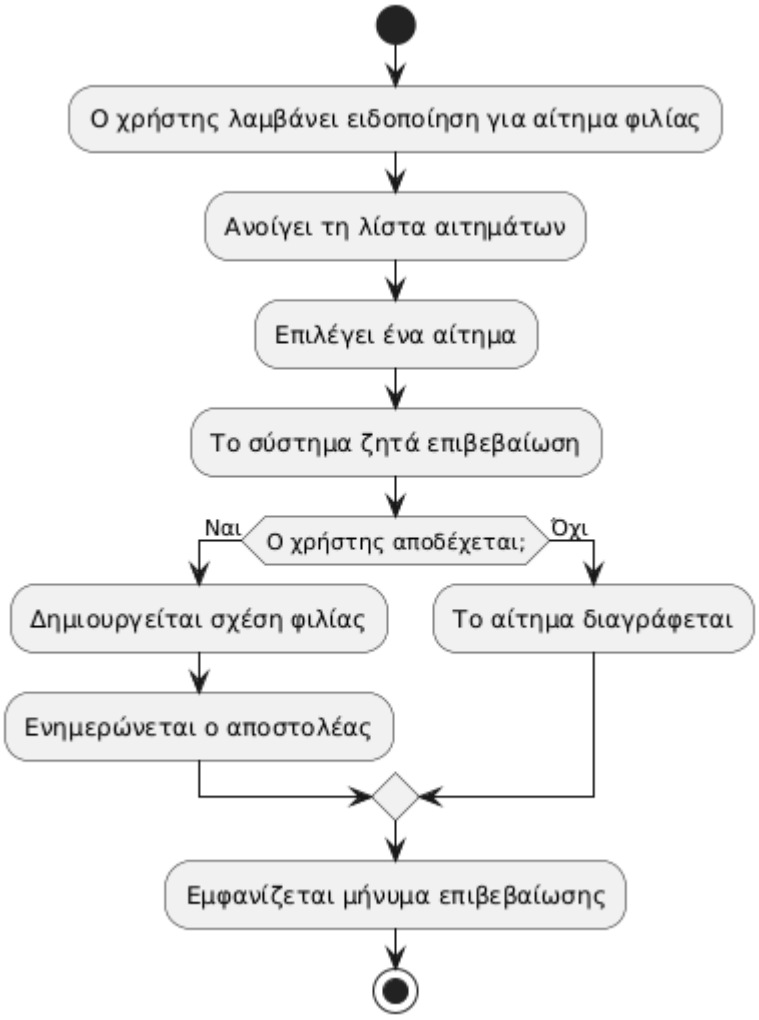
Εικόνα 21: Διάγραμμα Δραστηριοτήτων Θέασης Θέματος

4.2.5.4 Social Media

Αποδοχή ή Απόρριψη Αιτήματος Φιλίας

Ο χρήστης λαμβάνει ειδοποίηση ότι έχει μια νέα αίτηση φιλίας και ανοίγει τη λίστα με τα εισερχόμενα αιτήματα. Από εκεί επιλέγει μία αίτηση και το σύστημα ζητά επιβεβαίωση για το αν την αποδέχεται ή απορρίπτει.

Εφόσον την αποδεχτεί, δημιουργείται η σχέση φιλίας και ο αποστολέας της αίτησης ενημερώνεται για την αποδοχή. Διαφορετικά, αν την απορρίψει, το αίτημα διαγράφεται. Σε κάθε περίπτωση, το σύστημα εμφανίζει ένα τελικό μήνυμα επιβεβαίωσης της ενέργειας.

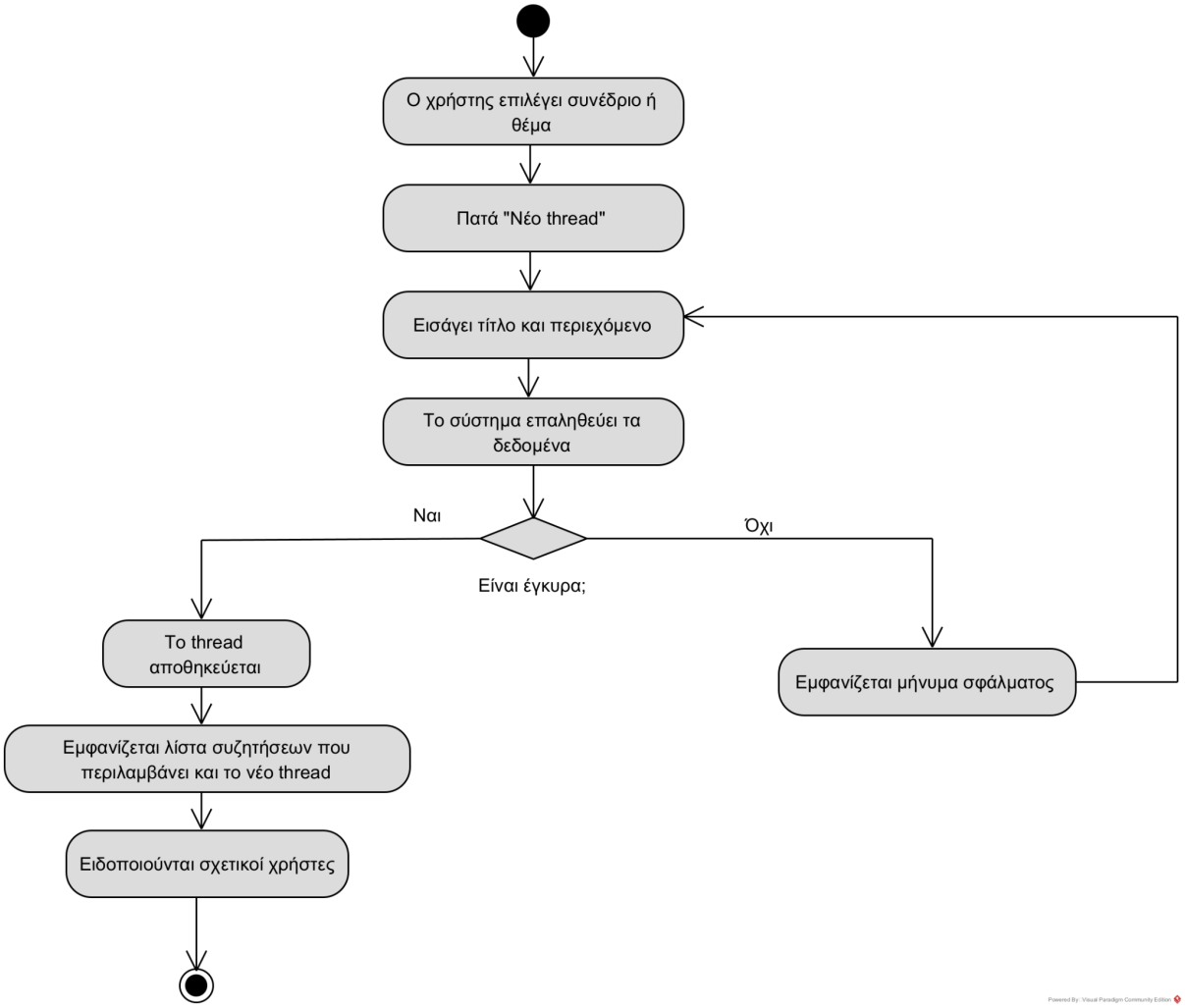


Εικόνα 22: Διάγραμμα Δραστηριοτήτων Αποδοχής ή Απόρριψης Αιτήματος Φιλίας

Δημιουργία Νέου Thread (Συζήτησης) σε Συνέδριο ή Θέμα

Ο χρήστης επιλέγει ένα συνέδριο ή ένα θέμα και στη συνέχεια προχωράει στην επιλογή για δημιουργία νέου thread. Στη φόρμα που εμφανίζεται, εισάγει έναν τίτλο και σχετικό περιεχόμενο. Το σύστημα επαληθεύει την εγκυρότητα των δεδομένων.

Εφόσον τα δεδομένα δεν είναι σωστά, εμφανίζεται μήνυμα σφάλματος και ο χρήστης μπορεί να τα διορθώσει. Διαφορετικά, αν είναι έγκυρα, το νέο thread αποθηκεύεται και εμφανίζεται μαζί με τα υπόλοιπα στη σχετική λίστα. Τέλος, οι χρήστες που έχουν εγγραφεί στο θέμα ή συνέδριο λαμβάνουν σχετική ειδοποίηση.



Εικόνα 23: Διάγραμμα Δραστηριοτήτων Δημιουργίας Νέου Thread σε Συνέδριο ή Θέμα

4.2.6 Διάγραμμα Κλάσεων (Class Diagram)

Το **Διάγραμμα Κλάσεων** περιγράφει τις βασικές οντότητες του συστήματος (με τη μορφή κλάσεων αντικειμένων), τις ιδιότητές τους και τις μεταξύ τους σχέσεις.

Βασικές Κλάσεις:

- **Χρήστης (User):**
  Αντιπροσωπεύει έναν εγγεγραμμένο χρήστη του συστήματος.
  - **Ιδιότητες:**
    - `userID: String` - το αναγνωριστικό του χρήστη
    - `username: String` - το όνομα χρήστη
    - `email: String` - το email του χρήστη
    - `password: String` - ο κωδικός του χρήστη
    - `role: Enum {RESEARCHER, CURATOR}` - ο ρόλος (ερευνητής ή επιμελητής) που παίζει ο χρήστης στο σύστημα

- `organization: String` - ο οργανισμός στον οποίο εντάσσεται ή εργάζεται ο χρήστης
- `title: String` - ο τίτλος του χρήστη (πχ. Δρ., Καθηγητής, κλπ.)

➢ **Σχέσεις:**

- Δημιουργεί *Conferences* (1-N) - χρήστης μπορεί να δημιουργήσει πολλά συνέδρια αλλά ένα συνέδριο ανήκει σε έναν χρήστη
- Δημιουργεί *Threads* (1-N) - ο χρήστης μπορεί να δημιουργήσει πολλά threads αλλά ένα thread ανήκει σε έναν χρήστη
- Γράφει *Comments* (1-N) - ο χρήστης μπορεί να αναρτήσει πολλά σχόλια για ένα θέμα - ένα σχόλιο αφορά έναν μόνο χρήστη
- Αξιολογεί *Conferences* μέσω *ConferenceRanking* (N-N) - ο χρήστης μπορεί να αξιολογήσει πολλά συνέδρια ενώ ένα συνέδριο μπορεί να αξιολογηθεί από πολλούς χρήστες
- Εγγράφεται σε *Topics* (N-N) - ο χρήστης μπορεί να εγγράφεται σε πολλά θέματα και σε ένα θέμα μπορούν να έχουν εγγραφεί πολλοί χρήστες
- Στέλνει αιτήματα φιλίας σε άλλους χρήστες (N-N με `sendsFriendRequestTo`) - ο χρήστης μπορεί να στείλει αιτήματα φιλίας σε πολλούς χρήστες καθώς και να λάβει πολλά αιτήματα από αυτούς

❖ **Συνέδριο (Conference):**
Αντιπροσωπεύει ένα επιστημονικό συνέδριο.

➢ **Ιδιότητες:**

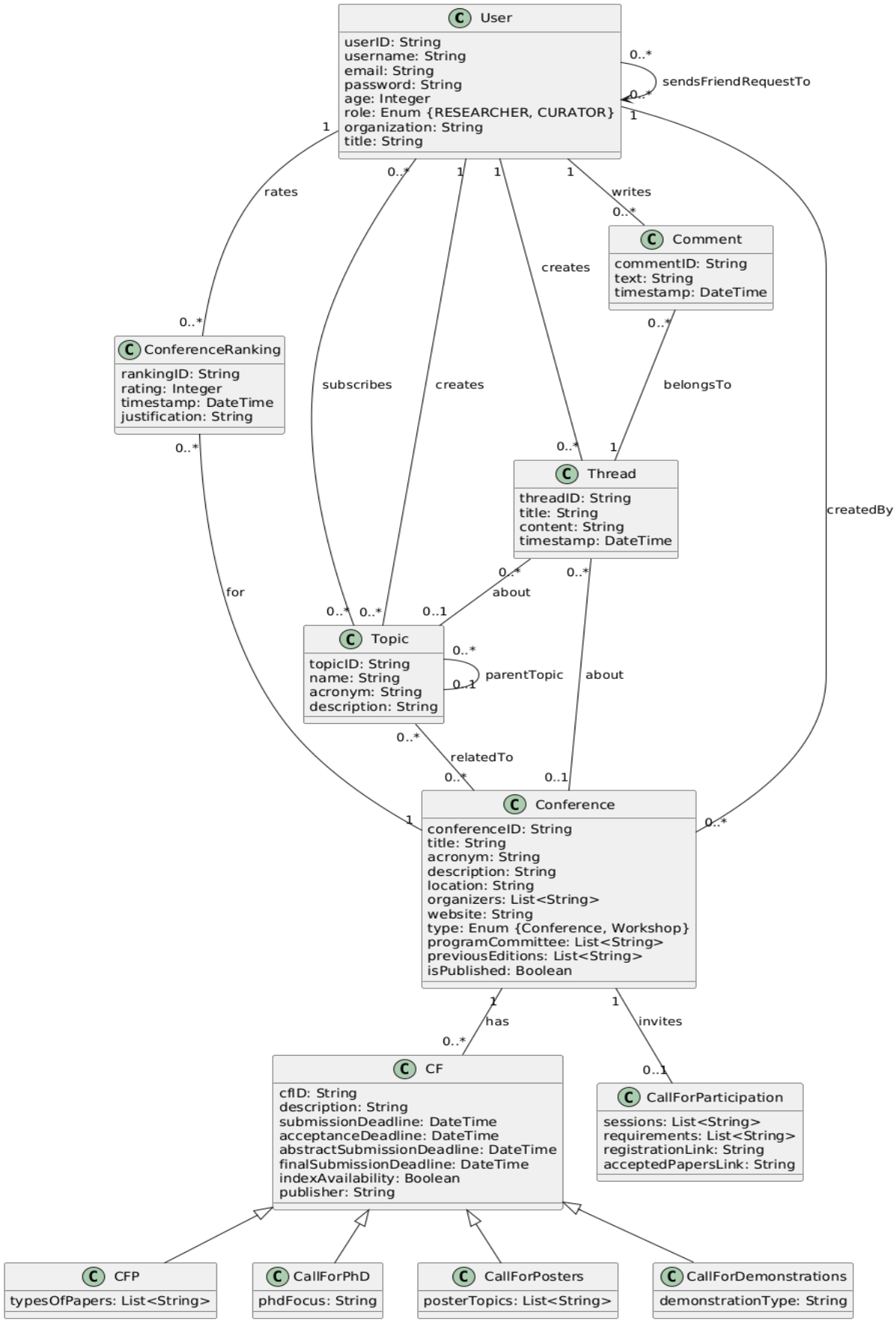
- `conferenceID: String` - το αναγνωριστικό του συνεδρίου
- `title: String` - ο τίτλος του συνεδρίου
- `acronym: String` - το ακρωνύμιο του συνεδρίου
- `description: String` - μια (κειμενική) περιγραφή του συνεδρίου
- `location: String` - η τοποθεσία όπου θα διεξαχθεί το συνέδριο
- `organizers: List<String>` - λίστα με διοργανωτές του συνεδρίου
- `website: String` - URL προς τον ιστότοπο του συνεδρίου
- `type: Enum {Conference, Workshop}` - τύπος συνεδρίου (κανονικό ή workshop)
- `programCommittee: List<String>` - λίστα με μέλη της επιτροπής προγράμματος του συνεδρίου
- `previousEditions: List<String>` - λίστα από συνδέσμους προς προηγούμενες εκδόσεις του συνεδρίου

        - `isPublished: Boolean` - αν το συνέδριο είναι δημοσιευμένο ή όχι
    - ➢ **Σχέσεις:**
        - Συσχετίζεται με πολλά *Topics* (N-N) - ένα συνέδριο συσχετίζεται με πολλά θέματα ενώ ένα θέμα μπορεί να αφορά πολλά συνέδρια
        - Έχει πολλές προσκλήσεις τύπου *CF* (1-N) - ένα συνέδριο μπορεί να έχει πολλές προσκλήσεις αλλά μια πρόσκληση αφορά πολλά συνέδρια
        - Έχει *CallForParticipation* (0-1) - ένα συνέδριο μπορεί να έχει το πολύ μόνο μια πρόσκληση προς συμμετοχή ενώ μια τέτοια πρόσκληση αφορά ένα μόνο συνέδριο
- ❖ **Θέμα (Topic):**
  Αντιπροσωπεύει θεματική περιοχή.
    - ➢ **Ιδιότητες:**
        - `topicID: String` - αναγνωριστικό του θέματος
        - `name: String` - το όνομα του θέματος
        - `acronym: String` - ακρωνύμιο του θέματος
        - `description: String` - μια κειμενική περιγραφή του θέματος
    - ➢ **Σχέσεις:**
        - Έχει πατρικό θέμα (1) - ένα θέμα μπορεί να έχει το πολύ ένα πατρικό θέμα
        - Έχει παιδικά θέματα (0-N) - ένα θέμα μπορεί να έχει πολλά θέματα παιδιά
- ❖ **Thread (Συζήτηση):**
  Αντιπροσωπεύει μία συζήτηση σχετική με θέμα ή συνέδριο.
    - ➢ **Ιδιότητες:**
        - `threadID: String` - αναγνωριστικό συζήτησης
        - `title: String` - τίτλος συζήτησης
        - `content: String` - κειμενική περιγραφή συζήτησης
        - `timestamp: DateTime` - χρονοσφραγίδα δημιουργίας συζήτησης
    - ➢ **Σχέσεις:**
        - Αφορά σε *Topic* ή *Conference* - μια συζήτηση αφορά είτε μόνο ένα συνέδριο είτε μόνο ένα θέμα
- ❖ **Comment (Σχόλιο):**
    - ➢ **Ιδιότητες:**
        - `commentID: String` - αναγνωριστικό σχολίου
        - `text: String` - περιεχόμενο σχολίου
        - `timestamp: DateTime` - χρονοσφραγίδα ανάρτησης
    - ➢ **Σχέσεις:**
        - Ανήκει σε *Thread* - ένα σχόλιο ανήκει σε μια συζήτηση ενώ μια συζήτηση μπορεί να έχει πολλά σχόλια
- ❖ **ConferenceRanking (Βαθμολογία/Αξιολόγηση Συνεδρίου):**
    - ➢ **Ιδιότητες:**
        - `rankingID: String` - αναγνωριστικό αξιολόγησης

        - `rating: Integer` - βαθμός αξιολόγησης
        - `timestamp: DateTime` - χρονοσφραγίδα αξιολόγησης
        - `justification: String` - δικαιολόγηση βαθμολογίας συνεδρίου
    - **Σχέσεις:**
        - Για *Conference* (N-1) - μια βαθμολογία/αξιολόγηση αφορά ένα μόνο συνέδριο αλλά ένα συνέδριο μπορεί να έχει πολλές αξιολογήσεις
- **CF (Call For...):**
  Αντιπροσωπεύει πρόσκληση για υποβολή (papers, posters, PhDs, demos).
    - **Ιδιότητες:**
        - `cfID: String` - αναγνωριστικό πρόσκλησης
        - `description: String` - περιγραφή πρόσκλησης
        - `submissionDeadline: DateTime` - διορία υποβολής
        - `acceptanceDeadline: DateTime` - διορία ενημέρωσης αποδοχής
        - `abstractSubmissionDeadline: DateTime` - διορία υποβολής abstracts
        - `finalSubmissionDeadline: DateTime` - διορία υποβολής τελικής μορφής αποδεκτών άρθρων
        - `indexAvailability: Boolean` - αν υπάρχει δυνατότητα ευρετηριασμού των άρθρων που γίνονται αποδεκτά στο συνέδριο
        - `publisher: String` - εκδότης του συνεδρίου
    - **Υποκλάσεις:**
        - `CFP: typesOfPapers: List<String>` - αναπαριστά πρόσκληση υποβολής για άρθρα και εμπεριέχει την ιδιότητα των τύπων των άρθρων
        - `CallForPhD: phdFocus: String` - αναπαριστά πρόσκληση υποβολής για προτάσεις / εργασίες PhD και εμπεριέχει μια ιδιότητα που προσδιορίζει τους θεματικούς τομείς ενδιαφέροντος για τις PhD εργασίες
        - `CallForPosters: posterTopics: List<String>` - αναπαριστά πρόσκληση υποβολής για πόστερ και εμπεριέχει μια ιδιότητα για τα αποδεκτά θέματα των πόστερ
        - `CallForDemonstrations: demonstrationType: String` - αναπαριστά πρόσκληση υποβολής για επιδείξεις με μια ιδιότητα σχετιζόμενη με τα είδη επιδείξεων που είναι αποδεκτά
- **CallForParticipation:**

  Αντιπροσωπεύει μια πρόσκληση συμμετοχής σε συνέδριο

- **Ιδιότητες:**
  - `sessions: List<String>` - θεματικές συνόδοι εντός του συνεδρίου
  - `requirements: List<String>` - λίστα από απαιτήσεις για τη συμμετοχή
  - `registrationLink: String` - URL προς την ιστοσελίδα εγγραφής για το συνέδριο
  - `acceptedPapersLink: String` - URL προς την ιστοσελίδα όπου αναφέρονται τα άρθρα που έχουν γίνει αποδεκτά στο συνέδριο



Εικόνα 24: Διάγραμμα Κλάσεων Συστήματος

4.3 Υλοποίηση

Η υλοποίηση του συστήματος βασίζεται σε σύγχρονες τεχνολογίες και αρχιτεκτονικές που διασφαλίζουν **απόδοση, επεκτασιμότητα και ευκολία συντήρησης**. Σε αυτή την ενότητα περιγράφονται οι γλώσσες προγραμματισμού, τα πλαίσια και οι βιβλιοθήκες που χρησιμοποιήθηκαν, η δομή του κώδικα, οι βασικές κλάσεις της εφαρμογής, καθώς και ο τρόπος δοκιμής του συστήματος.

4.3.1 Γλώσσα Προγραμματισμού και Τεχνολογίες

Η υλοποίηση αναπτύχθηκε με χρήση **σύγχρονων τεχνολογιών και αρχιτεκτονικών**, διαχωρίζοντας το **Frontend, Backend και Database Layer**.

4.3.1.1 Frontend (Περιβάλλον Χρήστη)

- **Γλώσσα:** JavaScript / TypeScript
- **Framework:** React.js
- **Βιβλιοθήκες:**
  - `React Router` (Διαχείριση δρομολόγησης)[4]: Το React Router είναι η πιο διαδεδομένη βιβλιοθήκη για τη διαχείριση δρομολόγησης σε εφαρμογές React, επιτρέποντας τη δημιουργία πολλαπλών σελίδων με δυναμικά URLs χωρίς πλήρη ανανέωση της σελίδας. Πώς χρησιμοποιήθηκε:
    - **Διαχείριση διαδρομών** (`Routes`) για σελίδες όπως `Home`, `Conferences`, `Topics`, `Profile`.
    - **Δυναμικές διαδρομές** (`:id`) για προβολή συγκεκριμένων συνεδρίων ή θεμάτων.
    - **Προστατευμένες διαδρομές** (`Private Routes`) για περιοχές που απαιτούν σύνδεση χρήστη.
  - `Axios` (Αιτήματα API)[5]: Το Axios είναι μια βιβλιοθήκη για διαχείριση HTTP αιτημάτων που επιτρέπει την εύκολη επικοινωνία μεταξύ του Frontend και του Backend API. Πώς χρησιμοποιήθηκε:
    - **GET αιτήματα** για φόρτωση δεδομένων (από βασικές οντότητες όπως συνέδρια, θέματα, χρήστες).
    - **DELETE αιτήματα** για διαγραφή αντικειμένων βασικών οντοτήτων (όπως χρηστών, συνεδρίων ή θεμάτων).
    - **POST/PUT αιτήματα** για εγγραφή σε θέματα, δημιουργία συνεδρίων/θεμάτων.
    - Προσθήκη `interceptors` (μηχανισμός της Axios για παρεμβολή σε αιτήματα/απαντήσεις, χρήσιμος για authentication).
    - Χειρισμός λαθών με `try/catch blocks`.
  - `Tailwind CSS` (Responsive Design)[6]: Το Tailwind CSS είναι ένα utility-first CSS πλαίσιο (framework) που επιτρέπει τη γρήγορη δημιουργία responsive

[4] React Router (2024). React Router: Declarative Routing for React.js Applications
[5] Axios (2024). Promise-based HTTP client for the browser and Node.js
[6] Tailwind CSS (2024). A utility-first CSS framework for rapid UI development

(αποκρίσιμων) UI χωρίς να χρειάζονται custom CSS αρχεία. Πώς χρησιμοποιήθηκε:

- **Βελτιστοποίηση UI** για κινητές συσκευές (`sm, md, lg, xl breakpoints`).
- **Εύκολη διαχείριση στυλ** απευθείας μέσα στα συστατικά (components) του React.
- **Χρήση προκατασκευσμένων κλάσεων (pre-built classes)** (`flex, grid, px-4, py-2, bg-blue-500`) για βελτιστοποίηση εμφάνισης.

- **Χαρακτηριστικά:**
  - Ανταποκρίσιμο UI (Responsive)
  - Διαχείριση καταστάσεων με React Hooks[7] που επιτρέπουν την αποδοτική διαχείριση της κατάστασης μιας React εφαρμογής χωρίς την ανάγκη χρήσης κλάσεων.
  - Δυναμική φόρτωση περιεχομένου (Lazy Loading[8]) ώστε να φορτώνονται δυναμικά τμήματα του UI μόνο όταν χρειάζεται, μειώνοντας τον αρχικό χρόνο φόρτωσης της εφαρμογής

4.3.1.2 Backend (Λογική Εφαρμογής)

- **Γλώσσα:** Java
- **Framework:** Spring Boot[9], ένα πλαίσιο της Java που επιτρέπει την ταχεία ανάπτυξη backend εφαρμογών με ελάχιστη διαμόρφωση. Βασίζεται στο Spring Framework και προσφέρει έτοιμες ρυθμίσεις (convention over configuration) για εφαρμογές και υπηρεσίες ιστού, μειώνοντας την ανάγκη για εκτεταμένη παραμετροποίηση.
- **Βιβλιοθήκες:**
  - `Spring Security` (Authentication & Authorization)[10]: Το Spring Security χρησιμοποιήθηκε για την ασφαλή διαχείριση χρηστών και την προστασία των τελικών σημείων (endpoints) του API. Πιο συγκεκριμένα:
    - **JWT Authentication:** Υλοποιήθηκε πλήρως η παραγωγή, η επαλήθευση και ο έλεγχος των JWT tokens, τα οποία περιέχουν πληροφορίες ταυτότητας και ρόλου του χρήστη.
    - **Role-Based Access Control (RBAC):** Ορίστηκαν ρόλοι (Researcher, Curator) με αντίστοιχα δικαιώματα πρόσβασης.
    - **Hashing κωδικών:** Χρησιμοποιήθηκε BCryptPasswordEncoder για ασφαλή αποθήκευση των κωδικών.
    - **Προστασία endpoints:** Χρησιμοποιήθηκαν annotations όπως @PreAuthorize και @Secured για τον έλεγχο πρόσβασης στα δεδομένα.
  - `Spring Data JPA` (Αλληλεπίδραση με Βάση Δεδομένων): Χρησιμοποιήθηκε το Spring Data JPA για την απλοποίηση της επικοινωνίας με τη βάση δεδομένων, με:

[7] React Hooks (2024). Managing Component State Efficiently
[8] React Lazy Loading (2024). Optimizing React Apps with Code Splitting
[9] Spring Boot (2024). Spring Boot: Rapid Development for Java Applications
[10] Spring Security (2024). Secure Spring Applications

        - Βασικές CRUD λειτουργίες μέσω JpaRepository (findAll(), findById(), save(), deleteById()).
        - Δηλώσεις σχέσεων οντοτήτων (OneToMany, ManyToMany, OneToOne) μεταξύ των βασικών οντοτήτων (User, Conference, Topic).
        - Υποστήριξη Lazy Loading και Fetching strategies για βελτιστοποίηση της απόδοσης κατά τις επερωτήσεις.
    - `Spring Boot Actuator` (Monitoring)[11]: Ενσωματώθηκε το Spring Boot Actuator για την παρακολούθηση της κατάστασης του συστήματος μέσω:
        - Health Checks (/actuator/health).
        - Metrics monitoring (/actuator/metrics) για παρακολούθηση πόρων και αιτημάτων HTTP.
        - Logging & tracing (/actuator/loggers).
    - `Lombok` (Αυτόματη δημιουργία getter/setter): Χρησιμοποιήθηκε η βιβλιοθήκη Lombok για την αυτόματη δημιουργία getter/setter μεθόδων, constructors, και equals/hashCode, μειώνοντας σημαντικά τον boilerplate κώδικα στα μοντέλα δεδομένων (User, Conference, Topic).

- **Χαρακτηριστικά:**
    - RESTful API
    - MVC/CSR αρχιτεκτονική
    - JWT authentication που επιτρέπει ασφαλή και κρυπτογραφημένη ταυτοποίηση και εξουσιοδότηση χρηστών, χωρίς αποθήκευση session στον server, αλλά με όλα τα στοιχεία ενσωματωμένα στο token.

4.3.1.3 Database Layer (Βάση Δεδομένων)

- **Σχεσιακή βάση δεδομένων:** PostgreSQL[12] (Για στατικά δεδομένα, όπως χρήστες και συνέδρια). Η PostgreSQL επιλέχθηκε επειδή είναι μια ισχυρή, αξιόπιστη και επεκτάσιμη σχεσιακή βάση δεδομένων, ιδανική για δομημένα δεδομένα, όπως χρήστες, συνέδρια και σχέσεις μεταξύ αυτών, παρέχοντας ACID συμμόρφωση και υποστήριξη για πολύπλοκα ερωτήματα.
- **NoSQL βάση δεδομένων:** MongoDB[13] (Για δυναμικά δεδομένα, όπως ειδοποιήσεις και κοινωνική δικτύωση). Η MongoDB επιλέχθηκε επειδή είναι μια ευέλικτη NoSQL βάση δεδομένων, ιδανική για δυναμικά και μη δομημένα δεδομένα, όπως ειδοποιήσεις και κοινωνικές αλληλεπιδράσεις, επιτρέποντας ταχεία ανάκτηση πληροφοριών και εύκολη κλιμάκωση μέσω αποθήκευσης δεδομένων σε μορφή JSON-like εγγράφων (documents).

4.3.1.4 Εξωτερικές Υπηρεσίες & APIs

- Google OAuth 2.0 (Authentication) – Χρησιμοποιείται για την ασφαλή ταυτοποίηση χρηστών, επιτρέποντας την ελεγχόμενη πρόσβαση μέσω μηχανισμού OAuth.

---

[11] Spring Boot Actuator (2024). Monitor Your Spring Boot Applications
[12] PostgreSQL (2024). The World's Most Advanced Open Source Relational Database
[13] MongoDB (2024). The NoSQL Database for Modern Applications

4.3.2 Δομή Κώδικα

4.3.2.1 Δομή UI (Frontend)

Το UI έχει αναπτυχθεί με React.js και ακολουθεί μια component-based αρχιτεκτονική. Η βασική δομή οργανώνεται σε σελίδες (Pages) και επαναχρησιμοποιήσιμα στοιχεία (Reusable Components).

Components:

Τα βασικά components που επαναχρησιμοποιούνται σε πολλαπλές σελίδες περιλαμβάνουν:

- **Navbar:** εμφανίζεται σε όλες τις βασικές σελίδες, παρέχοντας πλοήγηση, είσοδο/εγγραφή και σύνδεσμο στο προφίλ (του χρήστη).
- **ConferenceCard**: παρουσιάζει συνοπτική πληροφορία για συνέδρια σε λίστες αποτελεσμάτων.
- **NotificationList**: εμφανίζει ειδοποιήσεις χρηστών και χρησιμοποιείται τόσο στη σελίδα ConferenceDetails όσο και στη σελίδα Προφίλ.
- **ProfileCard**: προβάλλει τα βασικά στοιχεία χρήστη στη σελίδα Προφίλ.
- **CommentsSection**: χρησιμοποιείται για την εμφάνιση σχολίων σε συζητήσεις ή αξιολογήσεις συνεδρίων.
- **SearchBar**: ανεξάρτητο component για την αναζήτηση συνεδρίων, θεμάτων ή χρηστών, το οποίο ενσωματώνεται τόσο στη HomePage όσο και σε εξειδικευμένες σελίδες.
- **TopicCard**: παρουσιάζει πληροφορία για ένα θέμα.
- **ThreadPreview**: προβολή αρχικής πληροφορίας για συζητήσεις (threads).

Pages:

- **HomePage.js:** Αρχική σελίδα με αναζήτηση συνεδρίων, προβολή ειδοποιήσεων, λίστα δημοφιλών συνεδρίων και βασική πλοήγηση.
- **ConferenceDetails.js**: Σελίδα λεπτομερειών συνεδρίου με ενότητες όπως θεματικές, βαθμολόγηση, σχόλια και συναφείς συζητήσεις.
- **Profile.js**: Σελίδα Προφίλ με πληροφορίες χρήστη, ειδοποιήσεις και διαχείριση φίλων.
- **TopicsPage.js**: Εμφανίζει τη δομή των θεμάτων και παρέχει δυνατότητα εγγραφής σε αυτά.
- **ThreadsPage.js**: Σελίδα με συζητήσεις (γενικές ή ανά συνέδριο/θέμα), δημιουργία και απάντηση σε συζητήσεις (threads).

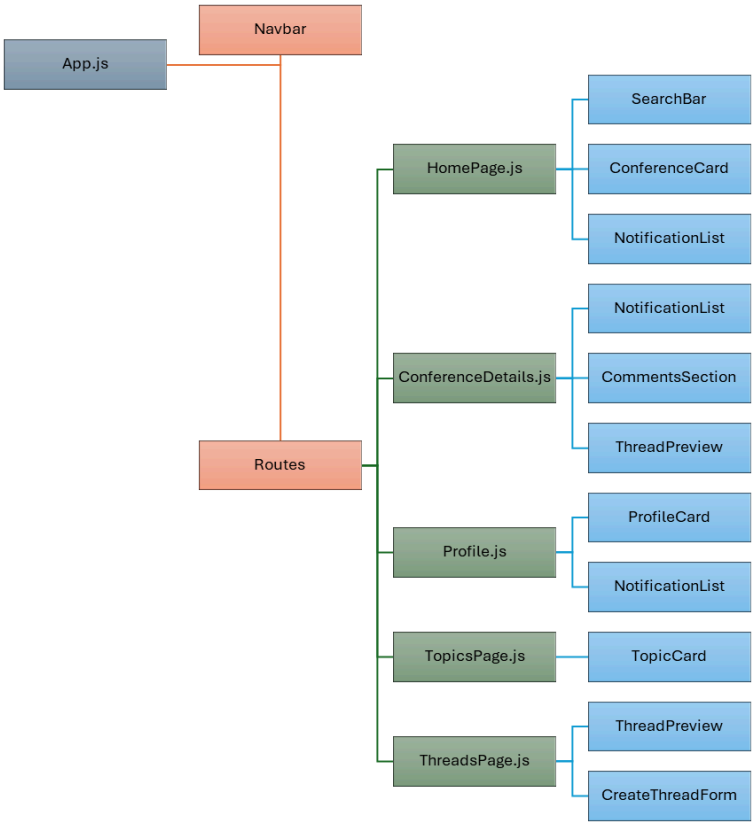
Λογική Εφαρμογής:

- **State Management**: Χρησιμοποιείται το React Context API για την κεντρική διαχείριση της κατάστασης της εφαρμογής (χρήστης, ειδοποιήσεις, θέματα).
- **Routing**: Η διαχείριση δρομολόγησης γίνεται μέσω του React Router, με υποστήριξη για δυναμικές διαδρομές και προστατευμένες σελίδες (π.χ. μόνο για συνδεδεμένους χρήστες).
- **Styling**: Εφαρμόζεται με χρήση της βιβλιοθήκης Tailwind CSS, επιτρέποντας responsive σχεδιασμό χωρίς custom CSS.

Παρότι έχει ξεκινήσει η ανάπτυξη του frontend, το υφιστάμενο UI αποτελεί ένα απλό πρωτότυπο (mockup) που παρέχει τη βασική δομή και λειτουργική ροή της εφαρμογής. Δεν έχει ολοκληρωθεί η πλήρης διασύνδεση με το backend, ούτε έχουν υλοποιηθεί όλες οι λεπτομέρειες των περιπτώσεων χρήσης.

Λόγω περιορισμένου χρόνου και της πολυπλοκότητας του συνολικού έργου, η υλοποίηση του frontend δεν προχώρησε σε τελικό στάδιο. Συνεπώς, η παρούσα εργασία έχει εστιάσει στην ολοκληρωμένη ανάπτυξη και δοκιμή του backend, ενώ το frontend παρέμεινε σε επίπεδο σχεδιασμού και αρχικής δοκιμής.

Αυτή η προσέγγιση επιτρέπει την περαιτέρω επέκταση και ολοκλήρωση του frontend σε επόμενο στάδιο, βασισμένη στη δομή και τα APIs που έχουν ήδη σχεδιαστεί και υλοποιηθεί.



Εικόνα 25: Ιεραρχική δομή React σελίδων UI

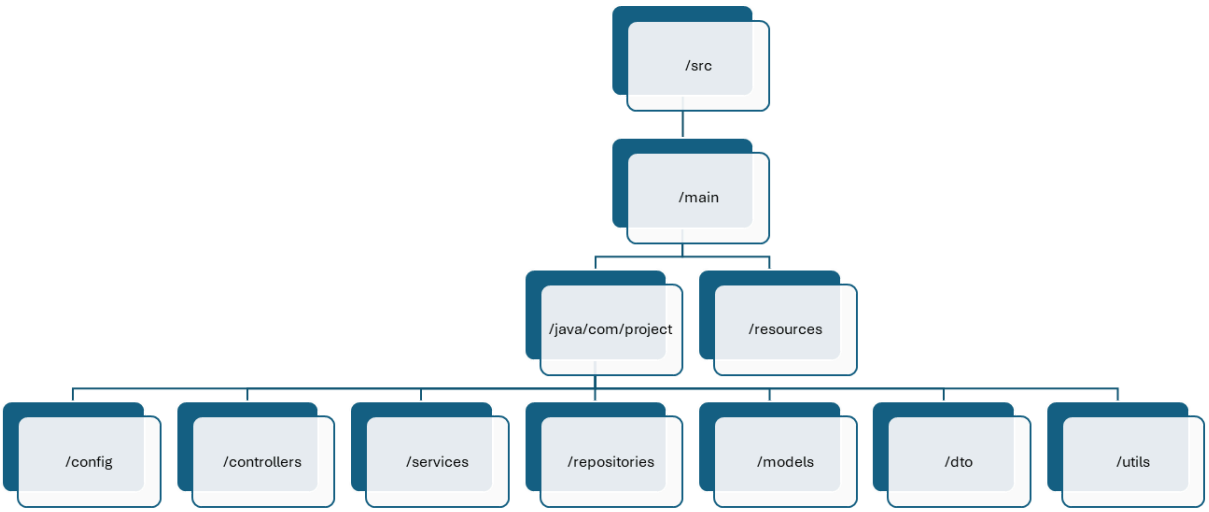
4.3.2.2 Δομή Backend

Η δομή του backend οργανώνεται σύμφωνα με την αρχιτεκτονική MVC (Model-View-Controller), ακολουθώντας τη βέλτιστη πρακτική για εφαρμογές Spring Boot.

- Ο βασικός φάκελος `/java/com/project` περιέχει τα κύρια υποσυστήματα του backend, δομημένα σε διακριτούς φακέλους για **καθαρό κώδικα και εύκολη συντήρηση**:
  - **/config**: Περιέχει αρχεία ρυθμίσεων (π.χ. διαμόρφωση ασφάλειας, CORS, database connections).
  - **/controllers**: Περιέχει κώδικα που αφορά κλάσεις τύπου ελεγκτή (controller), οι οποίες διαχειρίζονται τα HTTP αιτήματα στο πλαίσιο κλήσεων API προς την εφαρμογή.
  - **/services**: Περιλαμβάνει κώδικα που υλοποιεί την επιχειρησιακή λογική, επεξεργάζοντας δεδομένα πριν αποθηκευτούν ή επιστραφούν.
  - **/repositories**: Περιέχει κώδικα υπεύθυνο για την αλληλεπίδραση με τη βάση δεδομένων, χρησιμοποιώντας τη τεχνολογία **Spring Data JPA**.

- **/models**: Περιέχει κλάσεις που μοντελοποιούν τις βασικές (πληροφοριακές) οντότητες της εφαρμογής (π.χ. `User`, `Conference`, `Topic`).
- **/dto**: Περιλαμβάνει Data Transfer Objects (DTOs), τα οποία χρησιμοποιούνται για τη μεταφορά δεδομένων μεταξύ του backend και του frontend ή μεταξύ διαφορετικών επιπέδων της εφαρμογής (ή μεταξύ του backend & εξωτερικών εφαρμογών που ενδέχεται να επαναχρησιμοποιούν τη λειτουργικότητά του). Τα DTOs ορίζουν σαφή σχήματα δεδομένων για λειτουργίες όπως:
  - Εγγραφή ή ενημέρωση χρηστών (π.χ. RegisterUserDTO, UpdateUserDTO)
  - Αναζήτηση συνεδρίων (ConferenceSearchDTO)
  - Αποστολή ή προβολή ειδοποιήσεων (NotificationDTO)
  - Διαχείριση θεμάτων και συνδρομών (TopicDTO, SubscriptionDTO)
- **/utils**: Βοηθητικές κλάσεις και αντίστοιχες μέθοδοι που χρησιμοποιούνται σε όλη την εφαρμογή.

- **/resources**: ο φάκελος αυτός περιλαμβάνει αρχεία ρυθμίσεων και στατικών πόρων



Εικόνα 26: Δομή Φακέλων Backend Εφαρμογής

4.3.2.3 Κλάσεις/Μοντέλα Τομέα (Domain Models)

Οι κλάσεις του τομέα (domain models) υλοποιήθηκαν σύμφωνα με το διάγραμμα κλάσεων που παρουσιάστηκε στην ενότητα **4.2.6 Διάγραμμα Κλάσεων (Class Diagram)**. Στην παρούσα ενότητα δεν παρατίθενται εκ νέου, καθώς έχουν ήδη αναλυθεί στη φάση της σχεδίασης.

4.2.2.4 Κλάσεις Backend (Business Logic & API Controllers)

Το backend της εφαρμογής ακολουθεί την αρχιτεκτονική MVC (Model–View–Controller), ιδιαίτερα το μοτίβο CSR, και διαχωρίζει την ευθύνη των λειτουργιών σε τρία επίπεδα:

- Controller classes - Αναλαμβάνουν την επεξεργασία HTTP αιτημάτων από το frontend και χαρτογραφούν τις κλήσεις σε endpoints της εφαρμογής. Κάθε controller

σχετίζεται με μια βασική (πληροφοριακή/επιχειρησιακή) οντότητα του συστήματος προς διαχείριση.
- Service classes - Ενσωματώνουν την επιχειρησιακή λογική για κάθε λειτουργία διαχείρισης οντοτήτων. Οι controller κλάσεις καλούν τις αντίστοιχες υπηρεσίες που εντοπίζονται σε αυτό το επίπεδο.
- Repository interfaces - Παρέχουν πρόσβαση στη βάση δεδομένων για κάθε (πληροφοριακή) οντότητα, αξιοποιώντας το Spring Data JPA.

Αναλυτικά:

- **`UserController`** – Διαχειρίζεται αιτήματα για χρήστες.
- **`ConferenceController`** – Υποστηρίζει δημιουργία, ενημέρωση, διαγραφή, θέαση και αναζήτηση συνεδρίων.
- **`TopicController`** – Χειρίζεται θέματα και εγγραφές χρηστών σε αυτά.
- **`NotificationController`** – Διαχειρίζεται την αποστολή ειδοποιήσεων.
- **`UserService`**, **`ConferenceService`**, **`TopicService`**, **`NotificationService`** – Περιλαμβάνουν την επιχειρησιακή λογική για κάθε αντίστοιχη λειτουργία διαχείρισης πάνω στις σχετικές, προαναφερόμενες πληροφοριακές οντότητες.
- **`UserRepository`**, **`ConferenceRepository`**, **`TopicRepository`** – Παρέχουν λειτουργίες καταχώρησης, ανάκτησης, τροποποίησης και διαγραφής δεδομένων των αντίστοιχων (πληροφοριακών) οντοτήτων.

Με αυτόν τον τρόπο, το σύστημα είναι επεκτάσιμο και διατηρεί καθαρό διαχωρισμό ευθυνών ανάμεσα στο API, τη λογική και την αποθήκευση των δεδομένων.

4.4 Δοκιμή του Συστήματος

4.4.1 Εισαγωγή

Η διαδικασία δοκιμής που ακολουθήσαμε για το σύστημά μας περιλάμβανε **δοκιμές μονάδων (Unit Testing), δοκιμές ενοποίησης (Integration Testing) και δοκιμές UI (Frontend Testing)**.

- Unit Testing (Μονάδες Δοκιμών - Backend)
    - **Εργαλεία:** JUnit, Mockito - Τα εργαλεία **JUnit** και **Mockito** προσφέρουν διαφορετικές δυνατότητες για τον έλεγχο της λειτουργικότητας του συστήματος και ενσωματώθηκαν ως **βιβλιοθήκες** στο έργο. Πιο συγκεκριμένα:
        - JUnit
            - Είναι το πιο διαδεδομένο πλαίσιο για τη συγγραφή δοκιμών μονάδων (unit tests) σε Java.
            - Επιτρέπει τη συγγραφή αυτοματοποιημένων δοκιμών για μεμονωμένες μεθόδους και κλάσεις.
            - Παρέχει **annotations** όπως `@Test`, `@BeforeEach`, `@AfterEach`, `@BeforeAll`, `@AfterAll`, για τη σωστή εκτέλεση και προετοιμασία των δοκιμών.

            - Χρησιμοποιείται για την αυτοματοποιημένη δοκιμή: της επιχειρησιακής λογικής (services), των κλάσεων οντοτήτων (π.χ. validation, getters/setters), και των ρυθμίσεων της εφαρμογής (π.χ. configuration classes).
        - Mockito
            - Είναι ένα **mocking framework**, που επιτρέπει τη δημιουργία εικονικών αντικειμένων (mocks) για εξαρτήσεις σε δοκιμές.
            - Χρησιμοποιείται για: την απομόνωση του κώδικα υπό δοκιμή από εξαρτώμενα συστατικά (π.χ. βάση, APIs), τη δημιουργία mocks για εσωτερικά αντικείμενα (π.χ. μια υπηρεσία μέσα σε έναν controller), και τη ρύθμιση προσδοκώμενων τιμών με when(...).thenReturn(...) για την προσομοίωση (των ιδανικών) απαντήσεων σε ερωτήματα προς τις εξαρτήσεις από τις κλάσεις που δοκιμάζονται.
            - Χρησιμοποιείται κυρίως για δοκιμές υπηρεσιών (services), repositories, καθώς και controller-classes μέσω mocking των εξαρτήσεών τους.
    - Υλοποιημένες Δοκιμές Μονάδων:
        - **Δοκιμές σε Αποθετήρια (Repositories):** Ελέγχθηκαν οι βασικές CRUD λειτουργίες καθώς και custom επερωτήσεις σε αποθετήρια, με χρήση in-memory βάσης δεδομένων (H2) για απομόνωση από την παραγωγική βάση και την αξιόπιστη επαλήθευση των αποτελεσμάτων.
        - **Δοκιμές σε Υπηρεσίες (Services):** Πραγματοποιήθηκαν δοκιμές επιχειρησιακής λογικής, όπως η δημιουργία και η επεξεργασία χρηστών, με απομόνωση εξαρτήσεων μέσω τεχνικών mocking. Έγινε επαλήθευση των αναμενόμενων αποτελεσμάτων.
        - **Δοκιμές σε Controllers (REST API):** Ελέγχθηκαν τα HTTP endpoints της εφαρμογής μέσω προσομοίωσης αιτημάτων, επαληθεύοντας την ορθότητα των status codes και των μορφών απάντησης, με mocking των αντίστοιχων υπηρεσιών (services) ώστε να ελεγχθεί η συμπεριφορά υπό διαφορετικές συνθήκες.
        - **Δοκιμές Κλάσεων Ωφέλειας (Utility Classes):** Υλοποιήθηκαν δοκιμές για τη σωστή δημιουργία και επαλήθευση JWT tokens, καλύπτοντας σενάρια επιτυχούς και αποτυχημένης αυθεντικοποίησης.
        - **Δοκιμές Διαμόρφωσης (Configuration):** Ελέγχθηκε η σωστή φόρτωση και συμπεριφορά των beans ασφαλείας της εφαρμογής, διασφαλίζοντας ότι οι ρυθμίσεις ασφαλείας ανταποκρίνονται στις προδιαγραφές του συστήματος.
    - Integration Testing (Δοκιμές Ενοποίησης)
        - Περιλαμβάνει την ολοκληρωμένη λειτουργική δοκιμή της συνεργασίας των υποσυστημάτων του backend, εστιάζοντας στις αλληλεπιδράσεις μεταξύ controller, service, repository και βάσης δεδομένων.
        - Οι δοκιμές κάλυψαν πλήρως τη ροή των δεδομένων από τα REST API endpoints έως και την αποθήκευση, επαληθεύοντας την ορθότητα της επιχειρησιακής λογικής και τον χειρισμό εξαιρέσεων.

        - Ιδιαίτερη έμφαση δόθηκε στον έλεγχο της συνέπειας των δεδομένων και στη σωστή συμπεριφορά της εφαρμογής υπό ρεαλιστικές συνθήκες χρήσης.
    - Frontend Testing (Δοκιμές UI)
        - Λόγω περιορισμένου χρόνου, το frontend δεν ολοκληρώθηκε πλήρως, οπότε οι δοκιμές UI περιορίστηκαν σε βασικό έλεγχο της λειτουργίας μέσω mocking δεδομένων και manual δοκιμών.

4.4.2 Δοκιμές Μονάδων

Στο πλαίσιο της διαδικασίας Unit Testing υλοποιήθηκαν δοκιμές μονάδων σε διάφορα επίπεδα του backend, όπως:

- **Repositories**, π.χ. με τη κλάση δοκιμών `UserRepositoryTest` που εστιάζει στον έλεγχο του `UserRepository`

Επεξήγηση Δοκιμής

Η δοκιμή `testFindByEmail` έχει ως στόχο να ελέγξει τη βασική λειτουργία του `UserRepository` για την ανάκτηση ενός χρήστη με βάση το email του. Συγκεκριμένα:

- **Ρύθμιση Test Περιβάλλοντος**: Χρησιμοποιείται το annotation `@DataJpaTest` που δημιουργεί ένα απομονωμένο περιβάλλον για δοκιμές JPA, με in-memory βάση δεδομένων (π.χ. H2). Το προφίλ `test` επιτρέπει την ειδική διαμόρφωση για το περιβάλλον δοκιμών.
- **Δημιουργία Δοκιμαστικών Δεδομένων**: Κατασκευάζεται ένα νέο αντικείμενο `User` με συγκεκριμένα πεδία (username, email, password, role) και αποθηκεύεται στη βάση με την μέθοδο `save`.
- **Εκτέλεση της Δοκιμαζόμενης Μεθόδου**: Καλείται η μέθοδος `findByEmail` του `UserRepository` για να βρεθεί ο χρήστης με το συγκεκριμένο email.
- **Έλεγχος Αποτελέσματος**: Με τις μεθόδους `assertTrue` και `assertEquals` ελέγχεται ότι:
    - Η αναζήτηση επιστρέφει αποτελέσματα (δηλαδή ο χρήστης βρέθηκε).
    - Όλα τα δεδομένα του χρήστη που επιστράφηκαν συμφωνούν με τα αρχικά δεδομένα.

Η δοκιμή δεν χρησιμοποιεί mock αντικείμενα γιατί είναι δοκιμή επιπέδου repository με πραγματική in-memory βάση, και ελέγχει την ακεραιότητα των λειτουργιών persistence. Δεν θεωρείται δοκιμή ενοποίησης διότι δεν χρησιμοποιείται κανονική βάση δεδομένων, όπως σχεσιακή.

```
package com.project.repositories;

import com.project.models.User;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
import org.springframework.test.context.ActiveProfiles;
```

```
import java.util.Optional;

import static org.junit.jupiter.api.Assertions.*;

@DataJpaTest
@ActiveProfiles("test")
public class UserRepositoryTest {

    @Autowired
    private UserRepository userRepository;

    @Test
    public void testFindByEmail() {
        // Δημιουργία και αποθήκευση ενός νέου χρήστη
        User user = new User();
        user.setUserID("1");
        user.setUsername("Alice");
        user.setEmail("alice@example.com");
        user.setPassword("pass123");
        user.setRole(com.project.models.Role.RESEARCHER);
        user.setTitle("Dr.");  // Προσθήκη τιμής για το title
        user.setOrganization("TechCorp");  // Προσθήκη τιμής για το
organization

        userRepository.save(user);

        // Ανάκτηση του χρήστη βάσει email
        Optional<User> found =
userRepository.findByEmail("alice@example.com");

        // Επαλήθευση ότι βρέθηκε χρήστης
        assertTrue(found.isPresent());

        // Επαλήθευση όλων των βασικών πεδίων
        User retrievedUser = found.get();
        assertEquals("Alice", retrievedUser.getUsername());
        assertEquals("alice@example.com", retrievedUser.getEmail());
        assertEquals("pass123", retrievedUser.getPassword());
        assertEquals(com.project.models.Role.RESEARCHER,
retrievedUser.getRole());
        assertEquals("Dr.", retrievedUser.getTitle());  // Επαλήθευση
του title
        assertEquals("TechCorp", retrievedUser.getOrganization());  //
Επαλήθευση του organization
```

```
    }
}
```

Παράθεμα Κώδικα 1: Μονάδα ελέγχου του repository UserRepository με χρήση JUnit και @DataJpaTest, η οποία επαληθεύει τη σωστή λειτουργία της μεθόδου findByEmail βάσει καταχωρημένου χρήστη.

- **Υπηρεσίες (Services)**, π.χ. με τη κλάση δοκιμών `UserServiceTest` που εστιάζει στον έλεγχο της υπηρεσίας `UserService`

Επεξήγηση Δοκιμής

Η δοκιμή `testCreateUser` ελέγχει τη λειτουργία της υπηρεσίας `UserService` για τη δημιουργία νέου χρήστη, απομονώνοντας τις εξαρτήσεις μέσω mock αντικειμένων:

- **Mocking**: Η εξάρτηση `UserRepository` γίνεται mock με τη χρήση του Mockito. Με την εντολή `when(userRepository.save(user)).thenReturn(user);` ορίζεται ότι όταν καλείται η μέθοδος `save` με το συγκεκριμένο αντικείμενο `user`, θα επιστρέφεται το ίδιο αντικείμενο. Αυτό αποτρέπει την πραγματική πρόσβαση στη βάση δεδομένων και επιτρέπει τον έλεγχο της λογικής της υπηρεσίας ανεξάρτητα.
- **Προετοιμασία**: Δημιουργείται ένα αντικείμενο `User` με συγκεκριμένα πεδία που χρησιμοποιείται ως είσοδος για τη μέθοδο `createUser`.
- **Κλήση Υπηρεσίας**: Η μέθοδος `createUser` εκτελείται, καλώντας ενδοεπιχειρησιακή λογική της υπηρεσίας και το mocked repository.
- **Επαλήθευση Αποτελέσματος**: Γίνεται έλεγχος ότι το αποτέλεσμα δεν είναι null και ότι όλα τα πεδία του χρήστη που επιστράφηκαν είναι όπως αναμένονται.
- **Επαλήθευση Συμπεριφοράς**: Με την `verify` επιβεβαιώνεται ότι η μέθοδος `save` του repository κλήθηκε ακριβώς μία φορά, αποδεικνύοντας πως η υπηρεσία προσπαθεί να αποθηκεύσει το χρήστη στη βάση.

Αυτή η δοκιμή είναι χαρακτηριστική για δοκιμή μονάδων υπηρεσιών, όπου η εστίαση είναι στη λογική της υπηρεσίας και όχι στις εξωτερικές εξαρτήσεις. Τα mock αντικείμενα παρέχουν πλήρη απομόνωση ώστε να μπορεί να ελεγχθεί μόνο η υπηρεσία.

```
package com.project.services;

import com.project.models.User;
import com.project.repositories.UserRepository;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;

public class UserServiceTest {
```

```
    @Mock
    private UserRepository userRepository;

    @InjectMocks
    private UserServiceImpl userService;

    @BeforeEach
    public void setUp() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    public void testCreateUser() {
        User user = new User();
        user.setUserID("1");
        user.setUsername("John Doe");
        user.setEmail("john@example.com");
        user.setPassword("password");
        user.setRole(com.project.models.Role.RESEARCHER);

        when(userRepository.save(user)).thenReturn(user);

        User createdUser = userService.createUser(user);

        assertNotNull(createdUser);
        assertEquals("John Doe", createdUser.getUsername());
        assertEquals("john@example.com", createdUser.getEmail());
        assertEquals("password", createdUser.getPassword());
        assertEquals(com.project.models.Role.RESEARCHER,
createdUser.getRole());

        verify(userRepository, times(1)).save(user);
    }
}
```

Παράθεμα Κώδικα 2: Μονάδα ελέγχου της υπηρεσίας UserServiceImpl με χρήση Mockito, η οποία ελέγχει ότι η μέθοδος createUser αποθηκεύει και επιστρέφει σωστά έναν νέο χρήστη μέσω του UserRepository.

- **Controllers**, π.χ. η κλάση δοκιμών `UserControllerTest` εστιάζει στην δοκιμή του UserController

Επεξήγηση

Η δοκιμή `testCreateUserEndpoint` ελέγχει το REST API endpoint δημιουργίας νέου χρήστη (`/api/users/register`) στον `UserController`.

- Χρησιμοποιώντας το `MockMvc` προσομοιώνεται ένα HTTP POST αίτημα, αποστέλλοντας το σώμα της αίτησης ως JSON που αντιστοιχεί σε αντικείμενο `RegisterUserDTO`, περιλαμβάνοντας πεδία όπως όνομα, email, τίτλος, οργανισμός και κωδικός.
- Με το `Mockito` γίνεται mocking της `UserService`, έτσι ώστε όταν κληθεί η μέθοδος `createUser` με οποιοδήποτε αντικείμενο `User`, να επιστραφεί προκαθορισμένος mock χρήστης.
- Επαληθεύεται ότι το αποτέλεσμα της αίτησης είναι HTTP 201 (Created), που υποδηλώνει επιτυχές αίτημα δημιουργίας.
- Επιπλέον, ελέγχεται ότι η JSON απόκριση περιέχει τα πεδία `username`, `email` και `role` με τις αναμενόμενες τιμές.
- Ο έλεγχος του ρόλου είναι σημαντικός, καθώς διασφαλίζει ότι ο ρόλος του χρήστη μεταβιβάζεται και αποθηκεύεται σωστά καθώς και εμφανίζεται στην τελική απόκριση του API.

Με αυτόν τον τρόπο, η δοκιμή καλύπτει τόσο την αναμενόμενη λειτουργικότητα όσο και τη δομή της απόκρισης, διασφαλίζοντας όχι μόνο τη βασική δημιουργία αλλά και την ορθή συμπεριφορά της υπηρεσίας.
Το endpoint `/api/users` υλοποιήθηκε προσωρινά για λόγους σαφήνειας, ωστόσο, συμφωνούμε ότι μια περισσότερο RESTful προσέγγιση προτείνει τη χρήση του `/api/users` με `POST` αίτημα, το οποίο υποδηλώνει από μόνο του την πρόθεση δημιουργίας ενός νέου πόρου (ενός χρήστη στη προκειμένη περίπτωση).

```
package com.project.controllers;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.project.dto.RegisterUserDTO;
import com.project.models.Role;
import com.project.models.User;
import com.project.services.UserService;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMoc
kMvc;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import static org.mockito.ArgumentMatchers.any;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post
;
```

```
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

@SpringBootTest
@AutoConfigureMockMvc(addFilters = false)
public class UserControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private UserService userService;

    private final ObjectMapper objectMapper = new ObjectMapper();

    @Test
    public void testCreateUserEndpoint() throws Exception {
        RegisterUserDTO dto = new RegisterUserDTO();
        dto.setUsername("Bob");
        dto.setEmail("bob@example.com");
        dto.setPassword("secret123");
        dto.setOrganization("TestOrg");
        dto.setTitle("PhD");

        User mockUser = new User();
        mockUser.setUserID("1");
        mockUser.setUsername("Bob");
        mockUser.setEmail("bob@example.com");
        mockUser.setPassword("secret123");
        mockUser.setRole(Role.RESEARCHER);


Mockito.when(userService.createUser(any(User.class))).thenReturn(mockUse
r);

        mockMvc.perform(post("/api/users")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(dto)))
                .andExpect(status().isCreated())
                .andExpect(jsonPath("$.username").value("Bob"))
                .andExpect(jsonPath("$.email").value("bob@example.com"))
                .andExpect(jsonPath("$.role").value("RESEARCHER"));
    }
}
```

Παράθεμα Κώδικα 3: Έλεγχος του REST endpoint /api/users του UserController με χρήση MockMvc, ο οποίος επαληθεύει την επιτυχή δημιουργία χρήστη και την ορθότητα των επιστρεφόμενων JSON πεδίων.

- **Utility κλάσεις**, π.χ. η κλάση δοκιμής `JwtUtilsTest` εστιάζει στον έλεγχο της utility κλάσης `JwtUtils`

Επεξήγηση

- Στη πρώτη δοκιμή, προστέθηκε assertion για να ελεγχθεί ρητά ότι το χρονικό διάστημα μεταξύ `issuedAt` και `expiration` είναι ακριβώς 1000ms (1 δευτερόλεπτο), όπως ορίζεται στην υλοποίηση του token.
- Στη δεύτερη δοκιμή, το `assertThrows` πλέον ελέγχει συγκεκριμένα για `ExpiredJwtException`, που είναι η εξαίρεση που ρίχνει το jwt όταν το token έχει λήξει. Αυτό κάνει το τεστ πιο ακριβές και αντιπροσωπευτικό.

```
package com.project.utils;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;
import org.junit.jupiter.api.Test;

import javax.crypto.SecretKey;
import java.util.Date;

import static org.junit.jupiter.api.Assertions.*;

public class JwtUtilsTest {

    private final SecretKey secretKey =
Keys.secretKeyFor(SignatureAlgorithm.HS512);

    // Μέθοδος για δημιουργία token για δοκιμές
    private String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() +
1000)) // 1 δευτερόλεπτο
                .signWith(secretKey)
                .compact();
    }

    @Test
    public void testGenerateAndValidateToken() {
        String token = generateToken("testUser");
```

```
        Claims claims = Jwts.parserBuilder()
                .setSigningKey(secretKey)
                .build()
                .parseClaimsJws(token)
                .getBody();

        assertEquals("testUser", claims.getSubject());
        assertNotNull(claims.getIssuedAt());
        assertNotNull(claims.getExpiration());

        // Νέο assertion για έλεγχο ότι το expiration είναι 1 δευτερόλεπτο
μετά το issuedAt
        long issuedAtMillis = claims.getIssuedAt().getTime();
        long expirationMillis = claims.getExpiration().getTime();
        assertTrue(expirationMillis - issuedAtMillis == 1000,
"Expiration should be 1 second after issuedAt");
    }

    @Test
    public void testTokenExpiration() throws InterruptedException {
        String token = generateToken("testUser");

        Thread.sleep(1100); // αναμονή 1.1 δευτ. για να λήξει το token

        // Πιο συγκεκριμένη εξαίρεση από τη jjwt για expired token
        assertThrows(io.jsonwebtoken.ExpiredJwtException.class, () -> {
            Jwts.parserBuilder()
                    .setSigningKey(secretKey)
                    .build()
                    .parseClaimsJws(token);
        });
    }
}
```

Παράθεσμα Κώδικα 4: Έλεγχος των βασικών λειτουργιών δημιουργίας και επαλήθευσης JWT tokens με χρήση της βιβλιοθήκης jjwt. Περιλαμβάνει δοκιμές για την εγκυρότητα του subject, των χρονικών σφραγίδων και την ανίχνευση ληγμένων tokens.

- **Ρυθμίσεις εφαρμογής**, π.χ. η κλάση δοκιμής `SecurityConfigTest` που ελέγχει τις διαμορφώσεις ασφάλειας του backend

Επεξήγηση

Η κλάση `SecurityConfigTest` υλοποιεί βασικές δοκιμές για την επιβεβαίωση της σωστής φόρτωσης και διαμόρφωσης της ασφάλειας στο backend. Συγκεκριμένα, περιλαμβάνει τις εξής δοκιμές:

- **contextLoads()**: Ελέγχει αν το bean της κλάσης `SecurityConfig` έχει φορτωθεί στο Spring Application Context. Αυτό διασφαλίζει ότι η βασική ρύθμιση ασφαλείας υπάρχει και έχει καταχωρηθεί σωστά, αποτρέποντας σφάλματα που θα προέκυπταν αν το bean απουσίαζε.
- **securityFilterChainExists()**: Ελέγχει αν το `SecurityFilterChain` bean, που περιλαμβάνει τον βασικό μηχανισμό ασφαλείας (όπως φίλτρα για αυθεντικοποίηση, εξουσιοδότηση και CSRF), είναι παρόν. Αυτό το bean είναι κρίσιμο για τη λειτουργία της ασφάλειας, καθώς εφαρμόζει τους κανόνες που έχουμε ορίσει.
- **testCsrfEnabled()**: Ελέγχει ότι το CSRF (Cross-Site Request Forgery) είναι ενεργοποιημένο στο `SecurityFilterChain`. Αυτό είναι σημαντικό για την προστασία του API από επιθέσεις τύπου CSRF, όπου κακόβουλες αιτήσεις που θα μπορούσαν να εκτελεστούν εκ μέρους του χρήστη χωρίς τη γνώση του.

Συνολικά, οι δοκιμές αυτές επιβεβαιώνουν ότι η βασική διαμόρφωση ασφαλείας είναι ενεργή και λειτουργεί όπως προβλέπεται, χωρίς να ελέγχουν συγκεκριμένες λειτουργίες ή endpoints. Για πιο λεπτομερείς δοκιμές λειτουργίας της ασφάλειας, μπορούν να χρησιμοποιηθούν και integration tests που ελέγχουν την πρόσβαση σε προστατευμένα endpoints με και χωρίς αυθεντικοποίηση.

```
package com.project.config;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.context.ApplicationContext;
import org.springframework.security.web.SecurityFilterChain;

import static org.junit.jupiter.api.Assertions.*;

@SpringBootTest
public class SecurityConfigTest {

    @Autowired
    private ApplicationContext applicationContext;

    @Autowired
    private SecurityFilterChain securityFilterChain;

    @Test
    public void contextLoads() {
        // Ελέγχει αν το bean SecurityConfig φορτώθηκε στο context
        assertNotNull(applicationContext.getBean("securityConfig"));
    }

    @Test
```

```
    public void securityFilterChainExists() {
        // Ελέγχει αν το SecurityFilterChain bean έχει φορτωθεί
        assertNotNull(securityFilterChain);
    }

    @Test
    public void testCsrfEnabled() throws Exception {
        // Εδώ αναμένουμε το CSRF να είναι disabled, άρα false
        assertFalse(securityFilterChain.toString().contains("csrf"));
    }
}
```

Παράθεμα Κώδικα 5: Έλεγχος της σωστής φόρτωσης του bean SecurityConfig και της ύπαρξης του SecurityFilterChain στο Spring Security context. Περιλαμβάνει και έλεγχο για απενεργοποίηση του μηχανισμού CSRF.

4.4.3 Integration Testing (Δοκιμές Ενοποίησης)

Χρησιμοποιήθηκε η βιβλιοθήκη **Spring Boot Test** σε συνδυασμό με το εργαλείο **MockMvc** για την εκτέλεση δοκιμών σε περιβάλλον που προσομοιάζει το πραγματικό runtime, με απομονωμένα mock αντικείμενα όπου απαιτείται. Η βάση δεδομένων αντικαταστάθηκε από **in-memory βάση (H2)** κατά τη διάρκεια των δοκιμών, διασφαλίζοντας ότι η αλληλεπίδραση με το persistence layer γίνεται σε ρεαλιστικό πλαίσιο χωρίς παρενέργειες. Με αυτόν τον τρόπο ελέγχθηκε η συνεργασία μεταξύ **controller**, **service** και **repository**, καθώς και η σωστή απόκριση των REST endpoints με βάση τα αναμενόμενα δεδομένα εισόδου και εξόδου.

Αντιπροσωπευτικό παράδειγμα
Μια χαρακτηριστική δοκιμή ενοποίησης είναι η δοκιμή δημιουργίας χρήστη μέσω POST αιτήματος στο endpoint `/api/users`. Το σενάριο ελέγχει την πλήρη διαδρομή των δεδομένων:

- Ο χρήστης αποστέλλει αίτημα όπου το σώμα (αιτήματος) έχει μορφή JSON και αντιστοιχεί σε ένα αντικείμενο RegisterUserDTO, το οποίο αναπαριστά τα δεδομένα ενός νέου χρήστη με πεδία, όπως username, email, password, τίτλος και οργανισμός.
- Το αίτημα φτάνει στον `UserController`, ο οποίος το προωθεί στην `UserService`.
- Η υπηρεσία μετασχηματίζει το DTO (RegisterUserDTO) σε οντότητα `User` και καλεί το αντίστοιχο repository για αποθήκευση.
- Η βάση δεδομένων (H2) αποθηκεύει το νέο χρήστη και επιστρέφει το αντικείμενο στην υπηρεσία.
- Ο controller επιστρέφει ως JSON την τελική αναπαράσταση του χρήστη στον client, συμπεριλαμβανομένων των πεδίων `username`, `email` και `role`.
- Η απόκριση του endpoint επαληθεύεται από τη δοκιμή ως προς:
    - τον **HTTP status code (201 Created)**,
    - την **ακρίβεια του περιεχομένου της JSON απόκρισης**, και

- την **ορθή ανάθεση ρόλου** στον χρήστη.

```
package com.project.controllers;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.project.dto.RegisterUserDTO;
import com.project.models.Role;
import com.project.models.User;
import com.project.services.UserService;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMoc
kMvc;
import org.springframework.boot.test.mock.mockito.MockBean;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import static org.mockito.ArgumentMatchers.any;
import static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post
;
import static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

@SpringBootTest
@AutoConfigureMockMvc(addFilters = false)
public class UserControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private UserService userService;

    private final ObjectMapper objectMapper = new ObjectMapper();

    @Test
    public void testCreateUserEndpoint() throws Exception {
        RegisterUserDTO dto = new RegisterUserDTO();
        dto.setUsername("Bob");
        dto.setEmail("bob@example.com");
        dto.setPassword("secret123");
```

```
        dto.setOrganization("TestOrg");
        dto.setTitle("PhD");

        User mockUser = new User();
        mockUser.setUserID("1");
        mockUser.setUsername("Bob");
        mockUser.setEmail("bob@example.com");
        mockUser.setPassword("secret123");
        mockUser.setRole(Role.RESEARCHER);


Mockito.when(userService.createUser(any(User.class))).thenReturn(mockUse
r);

        mockMvc.perform(post("/api/users")
                .contentType(MediaType.APPLICATION_JSON)
                .content(objectMapper.writeValueAsString(dto)))
                .andExpect(status().isCreated())
                .andExpect(jsonPath("$.username").value("Bob"))
                .andExpect(jsonPath("$.email").value("bob@example.com"))
                .andExpect(jsonPath("$.role").value("RESEARCHER"));
    }
}
```

Παράθεμα Κώδικα 6: Δοκιμή ενοποίησης με χρήση MockMvc για την επαλήθευση του endpoint δημιουργίας χρήστη μέσω POST.

Η δοκιμή αυτή δεν επικεντρώνεται μόνο σε μια μεμονωμένη μονάδα (unit), αλλά εξετάζει τη **συνεργασία όλων των επιπέδων** του backend, αποδεικνύοντας την **ενοποιημένη λειτουργία** του συστήματος σε ένα ολοκληρωμένο σενάριο χρήσης.

4.4.4 UI Testing (Δοκιμές Frontend)

Το frontend δεν ολοκληρώθηκε, επομένως δεν πραγματοποιήθηκαν πλήρεις δοκιμές διεπαφής χρήστη (UI testing).
Ωστόσο, πραγματοποιήθηκαν **mocked integration tests** με **Jest** και **React Testing Library (RTL)** για βασικές προσομοιωμένες δοκιμές, όπως ο έλεγχος αν το πεδίο αναζήτησης συνεδρίων αποδίδεται σωστά.
Οι δοκιμές αυτές βασίζονται σε mock δεδομένα και δεν περιλαμβάνουν πλήρη end-to-end εκτέλεση, καθώς το backend δεν είναι πλήρως ενσωματωμένο με το frontend.

4.5 Βαθμός Ικανοποίησης Απαιτήσεων

Ο βαθμός ικανοποίησης των απαιτήσεων υπολογίστηκε με βάση την υλοποίηση του backend, καθώς δεν υλοποιήθηκε πλήρως το frontend και η βασική λειτουργικότητα του συστήματος εξαρτάται κυρίως από το backend. Ως εκ τούτου, η αξιολόγηση του βαθμού

ικανοποίησης αφορά αποκλειστικά το backend κομμάτι του συστήματος, που είναι το πιο κρίσιμο για τη λειτουργία του.

Πίνακας 2: Βαθμός Ικανοποίησης Λειτουργικών Απαιτήσεων

Κωδ. Απαίτησης	Περιγραφή Απαίτησης	Ικανοποίηση	Δικαιολόγηση
FR1	Εγγραφή χρήστη	✔️ Πλήρης	Υλοποιήθηκε πλήρως στο backend μέσω REST API `/api/users/register`. Δοκιμάστηκε με μονάδες (unit) και ολοκλήρωσης (integration) tests.
FR2	Διαγραφή χρήστη	✔️ Πλήρης	Υλοποιήθηκε στο backend, μεμονωμένα unit tests. Δεν έγιναν ολοκληρωμένα integration tests για λόγους περιορισμένου χρόνου.
FR3	Ενημέρωση στοιχείων χρήστη	✔️ Πλήρης	Υλοποιήθηκε και ελέγχθηκε με unit tests στο backend.
FR4	Επαναφορά/ενημέρωση κωδικού	⚠️ Μερική	Υλοποιήθηκε βασική λογική στο backend, δεν έχει ολοκληρωθεί πλήρως (π.χ. διαχείριση προσωρινών κωδικών, χρονικών ορίων).
FR5	Ταυτοποίηση χρήστη	✔️ Πλήρης	Υλοποιήθηκε πλήρως με JWT Authentication. Δοκιμάστηκε εκτενώς με μονάδες και integration tests.
FR6	Εμφάνιση χρήστη	✔️ Πλήρης	Υλοποιήθηκε στο backend. Δοκιμάστηκε με REST API tests και unit tests.
FR7	Αναζήτηση χρήστη	✔️ Πλήρης	Υλοποιήθηκε στο backend με custom queries (repository). Δοκιμάστηκε με unit tests. Δεν εκτελέστηκαν ολοκληρωμένα integration tests.

FR8	Εισαγωγή συνεδρίου	⚠️ Μερική	Υλοποιήθηκε στο backend. Δεν υλοποιήθηκε η αυτόματη εισαγωγή (crawling). Δοκιμάστηκε με μονάδες και API tests.
FR9	Ενημέρωση συνεδρίου	⚠️ Μερική	Υλοποιήθηκε στο backend με αντιδραστικό τρόπο (χειροκίνητη ενημέρωση). Προληπτική ενημέρωση (crawling) δεν υλοποιήθηκε.
FR10	Διαγραφή συνεδρίου	✔️ Πλήρης	Υλοποιήθηκε και ελέγχθηκε με unit και API tests.
FR11	Αναζήτηση συνεδρίου	⚠️ Μερική	Υλοποιήθηκε απλή αναζήτηση. Εξεζητημένη αναζήτηση δεν υλοποιήθηκε. Δοκιμάστηκε με API tests.
FR12	Δημοσίευση συνεδρίου	⚠️ Μερική	Υλοποιήθηκε στο backend με χειρωνακτική δημοσίευση. Δεν υπάρχει αυτοματοποιημένη δημοσίευση.
FR13	Εμφάνιση συνεδρίου	✔️ Πλήρης	Υλοποιήθηκε πλήρως στο backend και δοκιμάστηκε με REST API tests.
FR14	Εισαγωγή θέματος	✔️ Πλήρης	Υλοποιήθηκε στο backend και δοκιμάστηκε με δοκιμές μονάδων και integration tests.
FR15	Ενημέρωση θέματος	✔️ Πλήρης	Υλοποιήθηκε στο backend και δοκιμάστηκε με δοκιμές μονάδων και integration tests.
FR16	Διαγραφή θέματος	✔️ Πλήρης	Υλοποιήθηκε στο backend και δοκιμάστηκε με δοκιμές μονάδων.
FR17	Εγγραφή σε θέμα	✔️ Πλήρης	Υλοποιήθηκε στο backend. Βασική υποστήριξη στο frontend με mock δεδομένα. Δοκιμάστηκε η εγγραφή και ενημέρωση χρηστών.
FR18	Διαγραφή από θέμα	✔️ Πλήρης	Υλοποιήθηκε στο backend. Δοκιμάστηκε μέσω unit tests.
FR19	Αναζήτηση θέματος	✔️ Πλήρης	Υλοποιήθηκε στο backend και δοκιμάστηκε με unit tests.

FR20	Εμφάνιση θέματος	✔️ Πλήρης	Υλοποιήθηκε στο backend και δοκιμάστηκε με REST API tests.
FR21	Βαθμολόγηση συνεδρίων	⚠️ Μερική	Υποστηρίζεται πλήρως στο Backend, το frontend υποστηρίζει μόνο εμφάνιση βαθμολογίας. Δεν έχουν γίνει πλήρεις δοκιμές.
FR22	Threads / συζητήσεις	✔️ Πλήρης	Υλοποιήθηκε πλήρως στο backend. Δοκιμάστηκαν με integration tests.
FR23	Friends (Διαχείριση φίλων και αιτημάτων)	✔️ Πλήρης	Υλοποιήθηκε στο backend. Δοκιμάστηκε με μονάδες και integration tests.

Πίνακας 3: Βαθμός Ικανοποίησης Μη Λειτουργικών Απαιτήσεων

Κωδ. Απαίτησης	**Περιγραφή Μη Λειτουργικής Απαίτησης**	**Ικανοποίηση**	**Δικαιολόγηση**
NFR1	Χρηστική & αποκριτική διεπαφή χρήσης	⚠️ Μερική	Βασικό UI υλοποιημένο με React και Tailwind CSS, χωρίς πλήρη λειτουργικότητα και ολοκλήρωση διεπαφής χρήστη.
NFR2	Διαχωρισμένο API και Front-End	⚠️ Μερική	Υλοποιήθηκε διαχωρισμός backend/frontend, αλλά το frontend δεν είναι ολοκληρωμένο ή πλήρως ενσωματωμένο.
NFR3	Υποστήριξη πολλαπλών γλωσσών (Ελληνικά & Αγγλικά)	☒	Δεν υλοποιήθηκε.
NFR4	Καλή απόδοση συστήματος	– (Μη Αξιολογημένο)	Δεν πραγματοποιήθηκαν επίσημες μετρήσεις απόδοσης στο backend ή frontend.
NFR5	Καλή διαθεσιμότητα συστήματος	– (Μη Αξιολογημένο)	Δεν εκτελέστηκαν ολοκληρωμένες δοκιμές διαθεσιμότητας. Δεν υπάρχουν στοιχεία για πλήρη αξιολόγηση.

NFR6	Υψηλή αξιοπιστία συστήματος	⚠️ Μερική	Υπάρχουν βασικοί χειρισμοί λαθών και εξαιρέσεων καθώς και επικυρώσεις δεδομένων, αλλά δεν καλύπτονται πλήρως όλες οι περιπτώσεις.
NFR7	Καλό επίπεδο ασφάλειας	⚠️ Μερική	Υλοποιήθηκε JWT Authentication, RBAC, κρυπτογράφηση κωδικών, υποστήριξη HTTPS. Δεν καλύπτονται ακόμα όλα τα θέματα ασφάλειας.
NFR8	Κάλυψη δοκιμών με χρήση unit, integration, system tests	⚠️ Μερική	Υλοποιήθηκαν βασικές δοκιμές μονάδων και ενοποίησης (κυρίως backend), με κάλυψη ~50-60% του κώδικα.

5. Εγκατάσταση & Επίδειξη Συστήματος

5.1 Εγκατάσταση

5.1.1 Προαπαιτούμενα

Για την τοπική εγκατάσταση και εκτέλεση του συστήματος απαιτούνται τα παρακάτω εργαλεία και τεχνολογίες:

Απαιτούμενα Εργαλεία

- **Java JDK 17+**
  Απαιτείται για την εκτέλεση του backend που είναι υλοποιημένο με Spring Boot.
  Εγκατάσταση:
  - Oracle JDK ή OpenJDK
- **Maven**
  Για τη διαχείριση των εξαρτήσεων και τη μεταγλώττιση/εκτέλεση του backend.
  Εγκατάσταση:
  - Apache Maven Installation Guide
- **Node.js & npm**
  Για την ανάπτυξη και εκτέλεση του frontend που είναι υλοποιημένο με React.js.
  Εγκατάσταση:
  - Node.js Official Site
- **PostgreSQL** και **MongoDB**
  Βάσεις δεδομένων για την αποθήκευση των δεδομένων της εφαρμογής.
  Εγκατάσταση:
  - PostgreSQL: https://www.postgresql.org/download/
  - MongoDB: https://www.mongodb.com/try/download/community
- **Git**
  Για τη διαχείριση της έκδοσης του κώδικα και την κλωνοποίηση από το αποθετήριο GitHub.
  Εγκατάσταση:
  - Git Official Site
- **Docker**
  Για την εύκολη εκτέλεση της εφαρμογής μέσω δοχείων (containers) σε οποιαδήποτε πλατφόρμα, χωρίς να απαιτείται τοπική εγκατάσταση όλων των εργαλείων.
  Εγκατάσταση:
  - Docker

Λήψη Κώδικα

- Ιδιωτικό Αποθετήριο
  Ο πηγαίος κώδικας στο Github (https://github.com/zeimpekis9/cfpzeimpekis) είναι διαθέσιμος μετά από αίτηση πρόσβασης.
  Παρακαλούμε επικοινωνήστε για να σας σταλεί πρόσκληση στο αποθετήριο.

5.1.2 Οδηγίες Εγκατάστασης και Εκτέλεσης

1. *Κλωνοποίηση αποθετηρίου*
   Κλωνοποιήστε το αποθετήριο του έργου από το GitHub στον τοπικό σας υπολογιστή με την εντολή:

```
git clone https://github.com/zeimpekis9/cfpzeimpekis
```

2. *Μετάβαση στο ριζικό φάκελο του έργου*
   Αφού ολοκληρωθεί η κλωνοποίηση, μεταβείτε στον ριζικό φάκελο του έργου:

```
cd cfpzeimpekis
```

3. *Backend - Εκτέλεση*
   Μεταβείτε στον φάκελο backend:

```
cd conference-cfp-platform-backend
```

Εκτελέστε το backend με Docker:

```
docker-compose up --build
```

4. Ανοίξτε τον browser στη διεύθυνση:

```
http://localhost:8080/swagger-ui/index.html
```

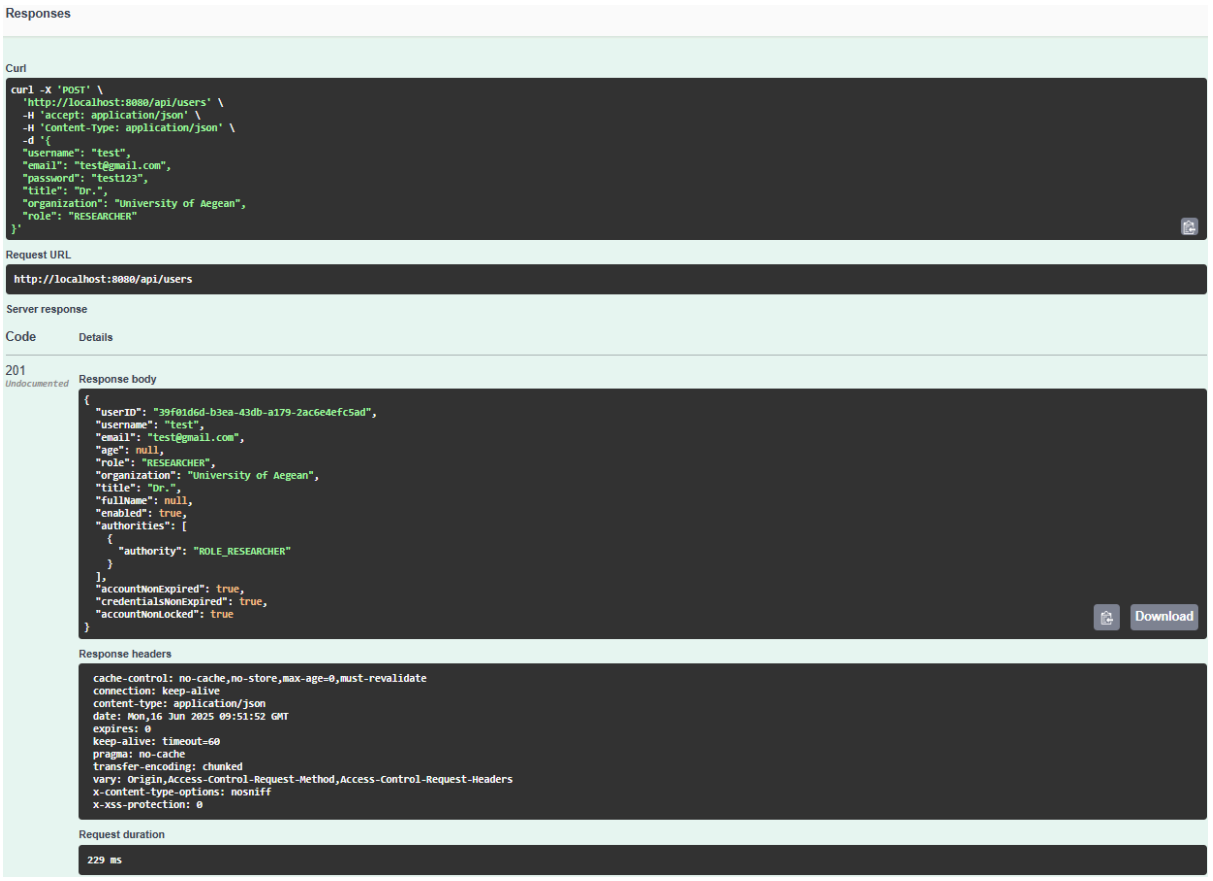
   όπου μπορείτε να δείτε και να δοκιμάσετε τα REST API endpoints.

5.2 Επίδειξη

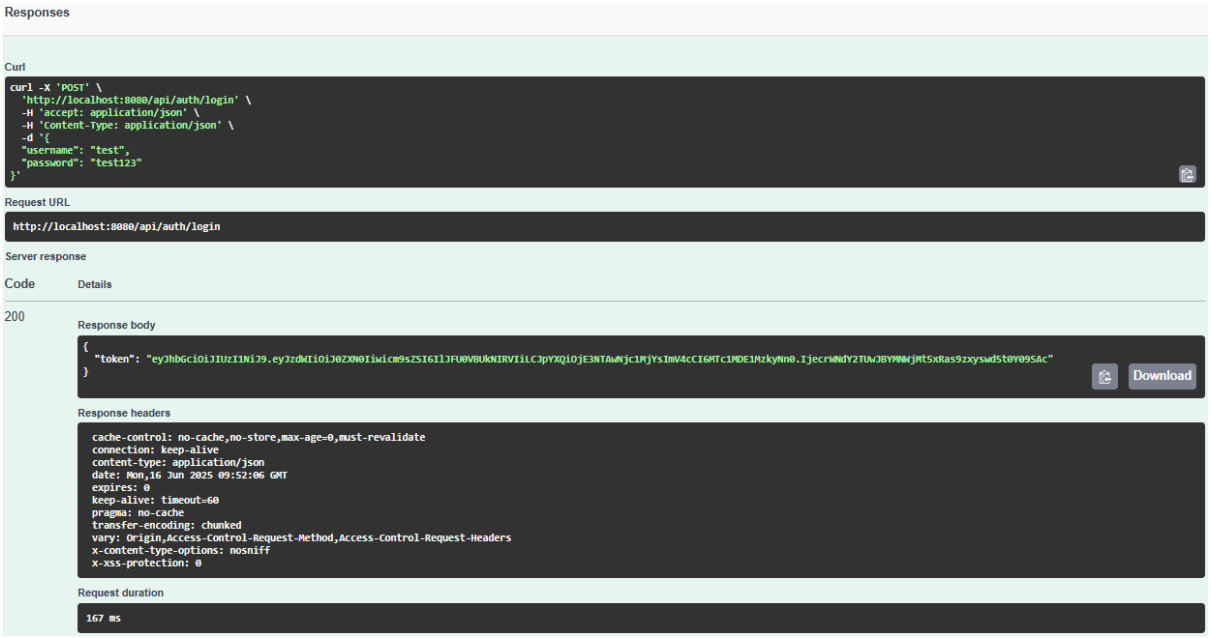
Παρακάτω παρουσιάζονται βασικά σενάρια χρήσης του (backend του) συστήματος, που αναδεικνύουν τη λειτουργικότητά του και τον τρόπο αλληλεπίδρασης των πελατών/χρηστών με το σύστημα.

- *Σενάριο 1: Εγγραφή Χρήστη, Σύνδεση Χρήστη, Εμφάνιση Στοιχείων Χρήστη, Ενημέρωση Στοιχείων Χρήστη και Αυτοδιαγραφή*
  - Ο χρήστης εγγράφεται στο σύστημα μέσω του endpoint (δείτε Εικόνα 27) `/api/users`, παρέχοντας τα βασικά στοιχεία (`username`, `email`, `password`).
  - Μετά την επιτυχή εγγραφή, πραγματοποιεί **είσοδο (login)** και λαμβάνει **JWT token** για ασφαλή πρόσβαση στα υπόλοιπα endpoints (δείτε Εικόνα 28). Αυτό γίνεται μέσω POST αιτήματος προς το endpoint /api/auth/login που περιλαμβάνει τα διαπιστευτήρια του χρήστη. Το token αυτό μπορεί έπειτα να ρυθμιστεί ώστε να περιλαμβάνεται στις αιτήσεις του χρήστη προς διαχείριση των πληροφοριακών οντοτήτων του συστήματος. Για παράδειγμα, αυτό επιτρέπεται στο κουμπί Authorize του Swagger UI (δείτε Εικόνα 29).

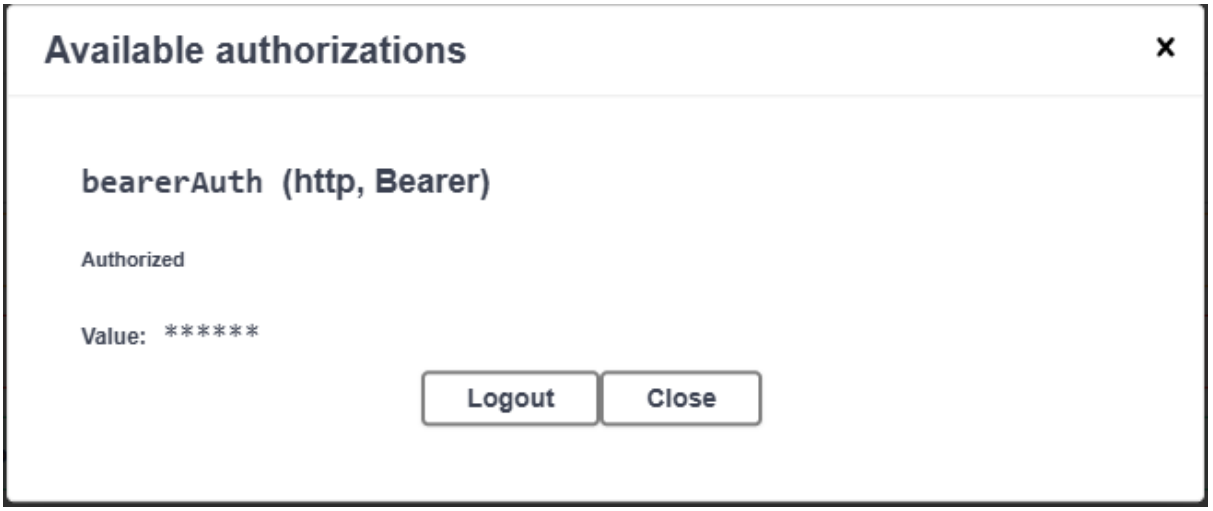
- Ο χρήστης μπορεί να **προβάλει τα στοιχεία του** με `GET` αίτηση βάσει του `username` του μέσω του endpoint `/api/users/{username}`, όπου η απόκριση θα περιλαμβάνει τα αποθηκευμένα δεδομένα χρήστη (δείτε Εικόνα 30).
- Μπορεί επίσης να **ενημερώσει τα στοιχεία του** (π.χ. email, ονοματεπώνυμο, οργανισμό, τίτλο) μέσω `PUT` αίτησης στο τελικό σημείο του REST API `/api/users/{id}`, στέλνοντας ενημερωμένο JSON σώμα και χρησιμοποιώντας το token που έλαβε (δείτε Εικόνα 31).
- Ο χρήστης με δικαιώματα Ερευνητή μπορεί να **διαγράψει τον λογαριασμό του** μέσω `DELETE` αίτησης στο endpoint `/api/users/me`, στέλνοντας μήνυμα επιτυχίας 204 που σημαίνει πως ο λογαριασμός διαγράφηκε με επιτυχία (δείτε Εικόνα 32).



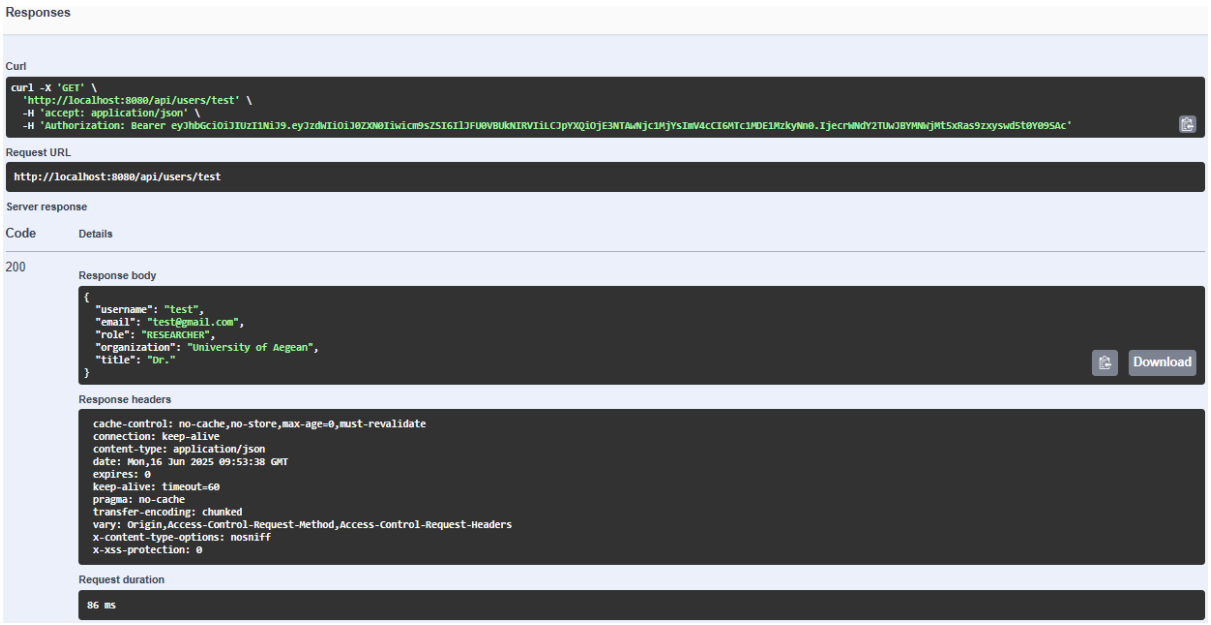
Εικόνα 27: Αίτημα εγγραφής χρήστη μέσω του endpoint `/api/users` με βασικά στοιχεία εγγραφής



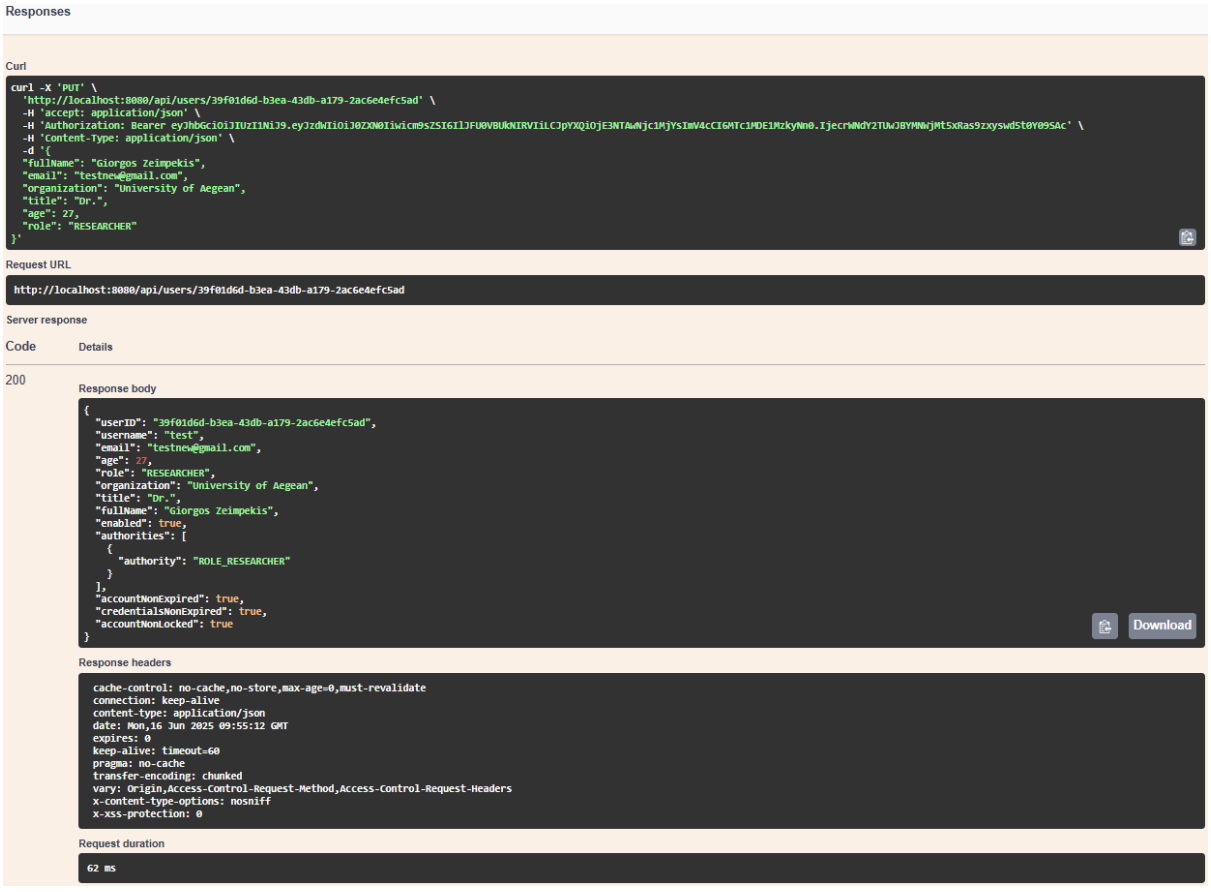
Εικόνα 28: Επιτυχής είσοδος χρήστη μέσω του endpoint /api/auth/login και παραλαβή JWT token για ασφαλή πρόσβαση



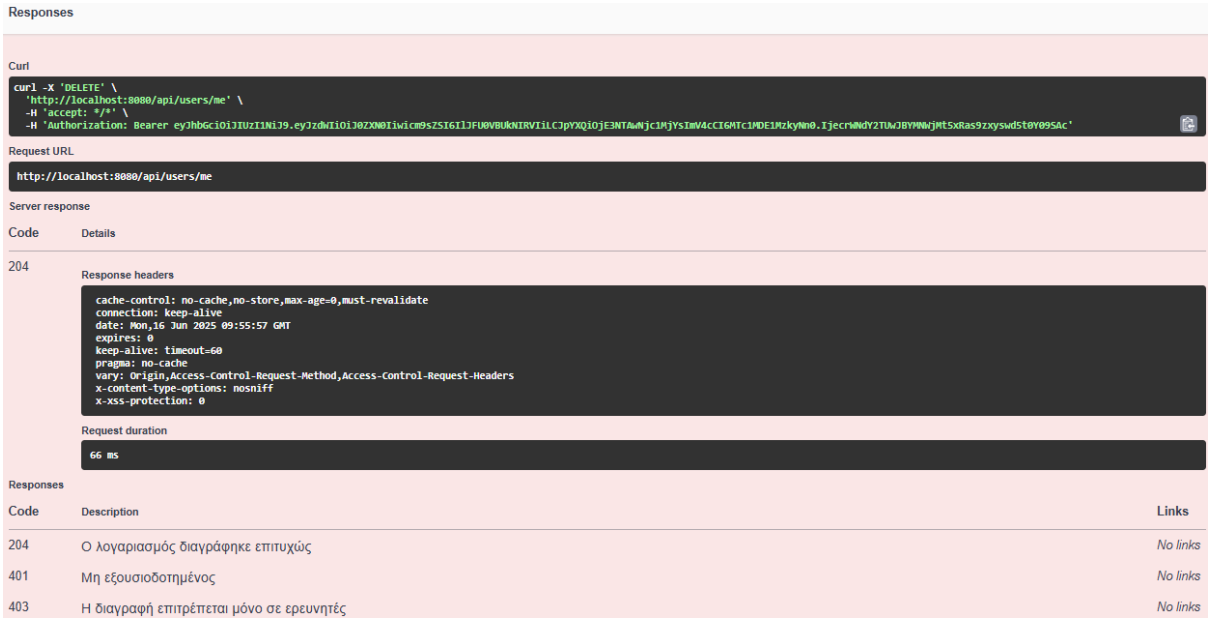
Εικόνα 29: Χρήση του JWT token στο κουμπί Authorize του Swagger UI για αυθεντικοποιημένη πρόσβαση στα endpoints



Εικόνα 30: Εμφάνιση των στοιχείων χρήστη με βάση το username, μέσω του endpoint /api/users/{username}. Το σύστημα επιστρέφει το προφίλ χωρίς τον κωδικό πρόσβασης



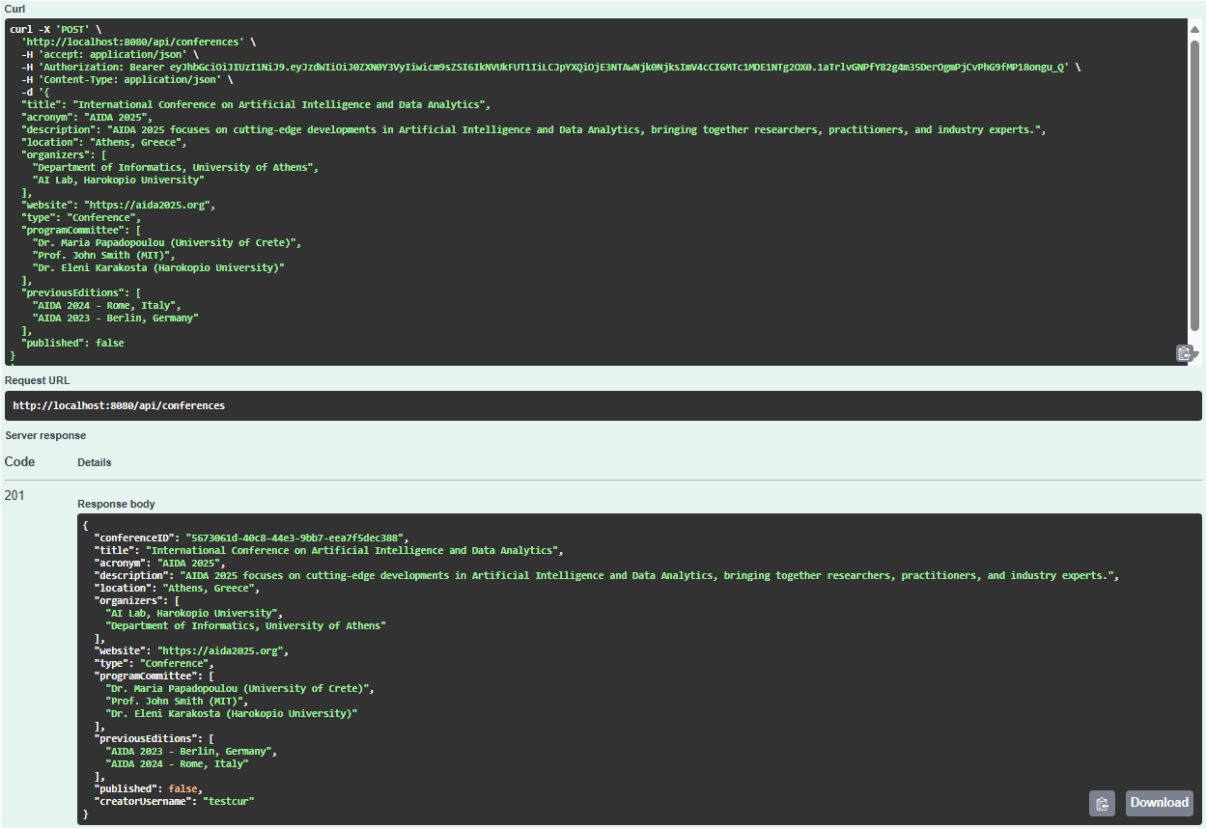
Εικόνα 31: Ενημέρωση των στοιχείων χρήστη μέσω του endpoint /api/users/{id} με PUT αίτηση



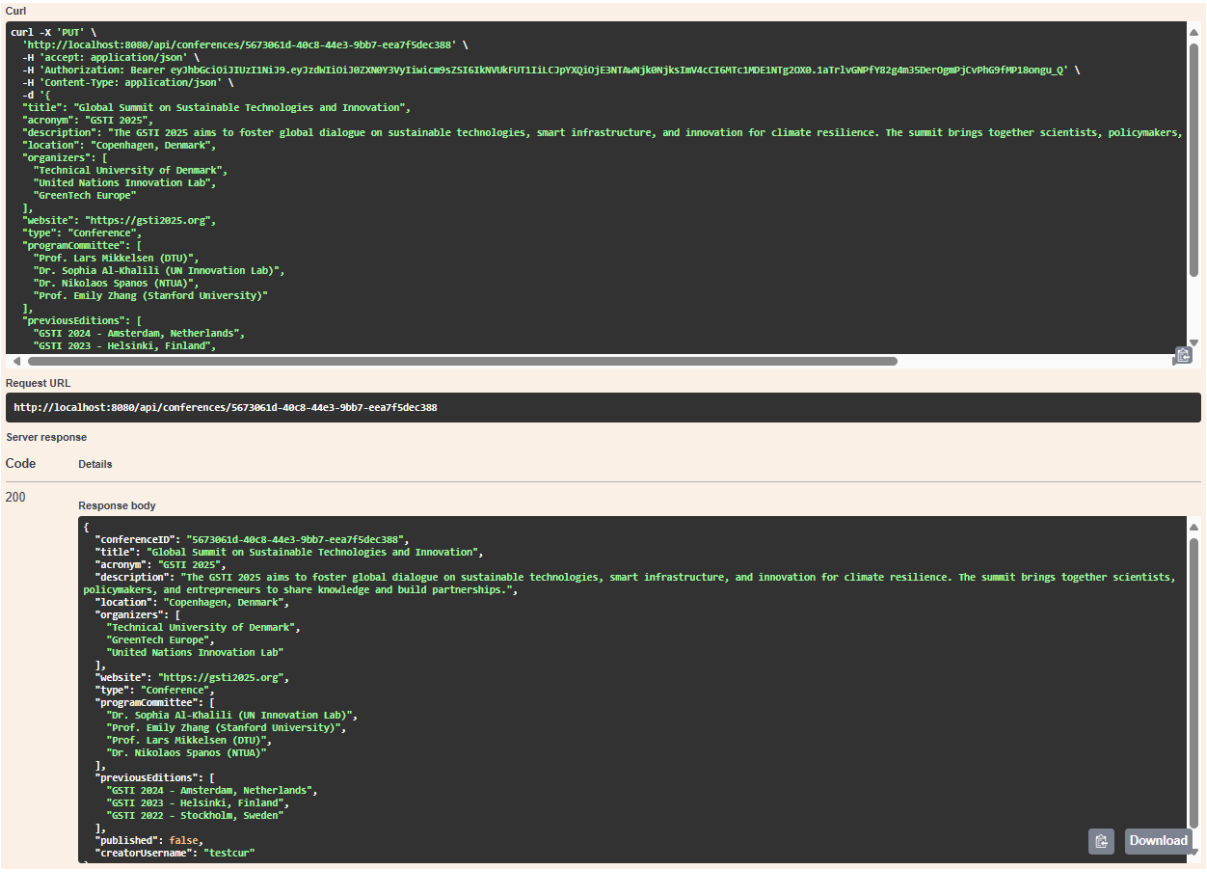
Εικόνα 32: Αυτοδιαγραφή λογαριασμού ερευνητή μέσω του endpoint /api/users/me με DELETE αίτηση

Για τα σενάρια 2, 3 και 4 χρησιμοποιήθηκε λογαριασμός με χρήστη επιμελητή για μια καλύτερη και ενοποιημένη παρουσίαση των σεναρίων με βάση τον ίδιο χρήστη. Προφανώς, ο χρήστης αυτός θα πρέπει να έχει ήδη συνδεθεί στο σύστημα και λάβει ένα έγκυρο token αυθεντικοποίησης, το οποίο έπειτα θα χρησιμοποιεί κατά την εκτέλεση των λειτουργιών διαχείρισης που περιλαμβάνονται στα σενάρια αυτά.

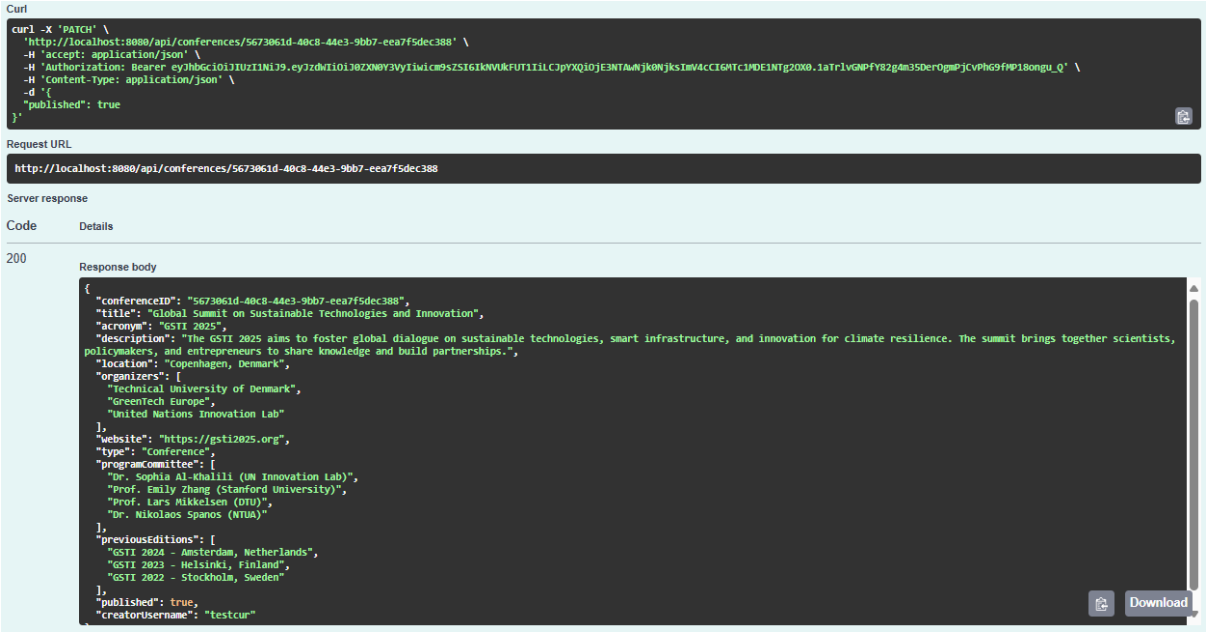
- Σενάριο 2: Δημιουργία, Ενημέρωση και Δημοσίευση Συνεδρίου
    - Ο χρήστης δημιουργεί ένα νέο συνέδριο χρησιμοποιώντας το endpoint **`/api/conferences`** με **`POST`** αίτηση όπου παρέχει όλες τις απαραίτητες πληροφορίες (δείτε Εικόνα 33).
    - Ενημερώνει στοιχεία του συνεδρίου (π.χ. ημερομηνίες, περιγραφή) μέσω του endpoint **`/api/conferences/{id}`** είτε με **`PUT`** αίτηση (δείτε Εικόνα 34) για αλλαγές πολλαπλών πεδίων είτε με **`PATCH`** για αλλαγές ενός πεδίου (δείτε Εικόνα 35), όπου `{id}` είναι το μοναδικό αναγνωριστικό του συνεδρίου.
    - Όταν το συνέδριο είναι έτοιμο προς δημοσίευση, ο χρήστης μπορεί να το δημοσιεύσει (ή να αλλάξει την κατάσταση δημοσίευσης), κάνοντας το ορατό στους υπόλοιπους χρήστες. Αυτό γίνεται και πάλι με αίτηση **`PATCH`** στο προαναφερόμενο endpoint (δείτε Εικόνα 35).



Εικόνα 33: Αίτημα δημιουργίας νέου συνεδρίου μέσω του endpoint /api/conferences. Ο χρήστης στέλνει όλα τα απαραίτητα στοιχεία του συνεδρίου (τίτλος, περιγραφή, τύπος κλπ)

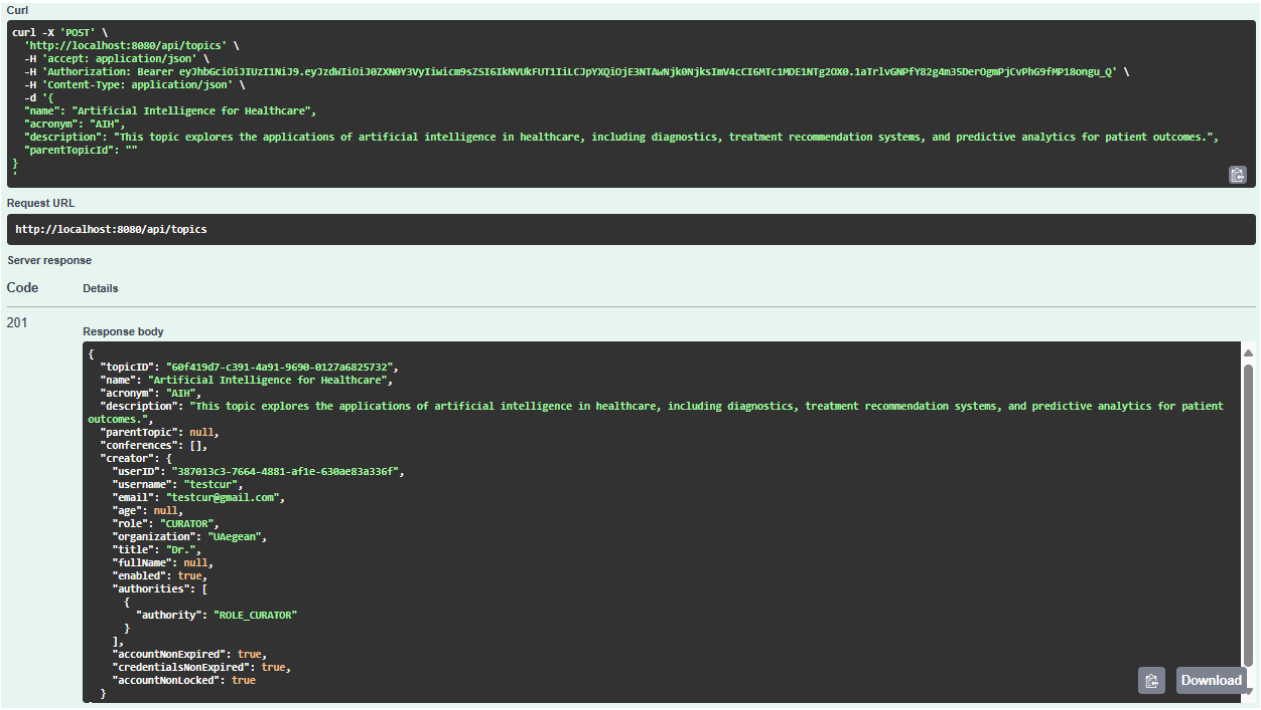


Εικόνα 34: Ενημέρωση στοιχείων συνεδρίου μέσω του endpoint /api/conferences/{id} με PUT αίτηση

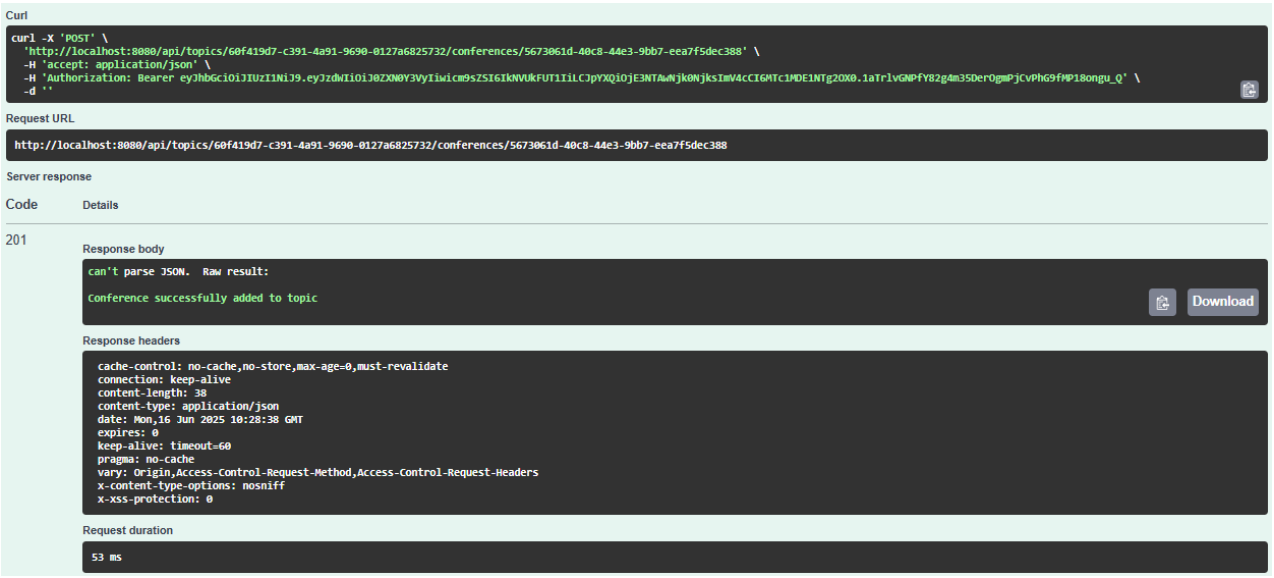


Εικόνα 35: Ενημέρωση ενός στοιχείου συνεδρίου μέσω του endpoint /api/conferences/{id} με PATCH αίτηση

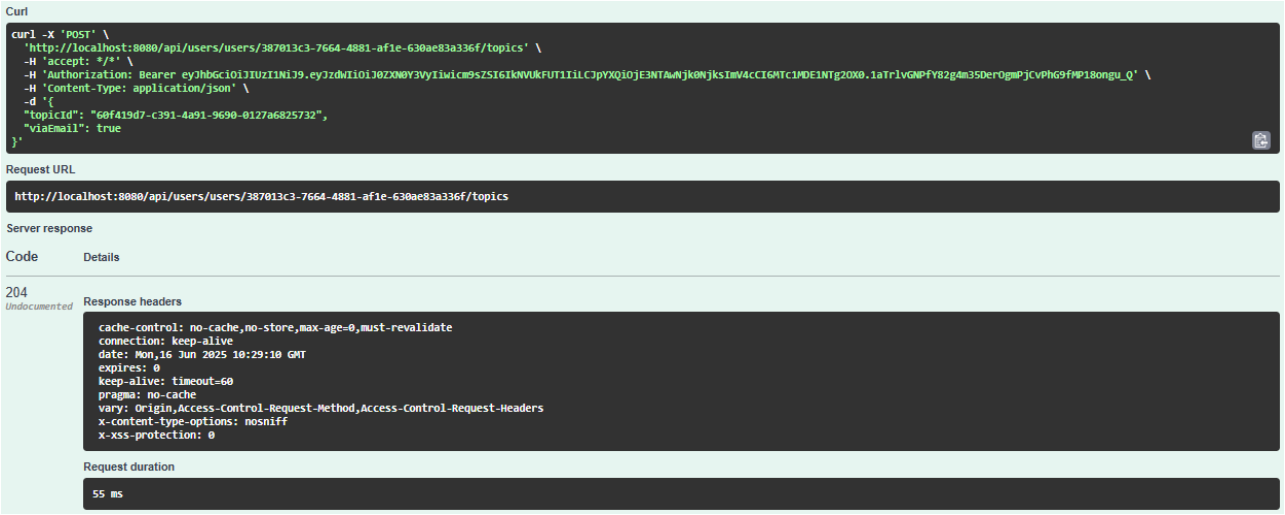
- *Σενάριο 3: Δημιουργία Θέματος, Συσχέτιση με Συνέδριο και Εγγραφή σε Θέμα*
  - Ο χρήστης αποστέλλει `POST` αίτημα στο endpoint `/api/topics`, αποστέλλοντας τα απαιτούμενα δεδομένα για το νέο θέμα (όνομα, ακρωνύμιο περιγραφή και πατρικό θέμα). Στην Εικόνα 36 παρατηρούμε πως το θέμα προς δημιουργία είναι ριζικό, οπότε δεν έχει πατρικό θέμα.
  - Με `POST` αίτημα στο endpoint `/api/topics/{topicId}/conferences/{conferenceId}` γίνεται η συσχέτιση του νέου θέματος με ένα συγκεκριμένο συνέδριο (δείτε Εικόνα 37). Η υπηρεσία ενημερώνει το αντικείμενο `Conference`, προσθέτοντας σε αυτό το σχετικό `Topic`.
  - Τέλος, με `POST` αίτημα στο endpoint `/api/users/{userId}/subscribe-topic/{topicId}`, ένας χρήστης μπορεί να εγγραφεί στο συγκεκριμένο θέμα (δείτε Εικόνα 38). Το σύστημα ενημερώνει τη λίστα μελών του θέματος με τον χρήστη αυτόν προς μελλοντική ειδοποιήσή του (πχ. όσον αφορά ενημερώσεις δημοσιευμένων συνεδρίων που σχετίζονται με το θέμα αυτό).



Εικόνα 36: Παράδειγμα POST αίτησης για τη δημιουργία νέου θέματος (topic) μέσω του endpoint /api/topics



Εικόνα 37: Παράδειγμα αποστολής POST αιτήματος στο endpoint /api/topics/{topicId}/conferences/{conferenceId} για τη σύνδεση ενός υπάρχοντος θέματος με ένα συνέδριο



Εικόνα 38: Παράδειγμα αποστολής POST αιτήματος για την εγγραφή (subscribe) ενός χρήστη σε συγκεκριμένο θέμα (topic)

- *Σενάριο 4: Αναζήτηση, Ενημέρωση και Διαγραφή Χρήστη από Επιμελητή*
  - Ο **επιμελητής** αποστέλλει **GET** αίτημα στο endpoint **`/api/users/search?q={όνομα ή username}`**, για την αναζήτηση χρηστών βάσει ονοματεπώνυμου ή username (δείτε Εικόνα 39).
  - Με **PATCH** ή **PUT** αίτημα στο endpoint **`/api/users/{userId}`**, γίνεται η ενημέρωση στοιχείων χρήστη (π.χ. email, τίτλος, οργανισμός) μέσω του κατάλληλου DTO από τον χρήστη με ρόλο επιμελητή (δείτε Εικόνα 40).
  - Ο επιμελητής μπορεί να διαγράψει έναν άλλον χρήστη με **DELETE** αίτημα στο endpoint **`/api/users/{userId}`** (δείτε Εικόνα 41)

Εικόνα 39: Παράδειγμα GET αιτήματος στο endpoint /api/users/search για την αναζήτηση χρηστών με βάση το ονοματεπώνυμο ή το username

Εικόνα 40: Παράδειγμα PUT αιτήματος στο endpoint /api/users/{id} όπου ένας επιμελητής μπορεί να ενημερώσει τα στοιχεία ενός χρήστη

Εικόνα 41: Παράδειγμα DELETE αιτήματος στο endpoint /api/users/{id} όπου ένας επιμελητής μπορεί να διαγράψει έναν ερευνητή

6. Συμπεράσματα & Μελλοντική Εργασία

Στο πλαίσιο της παρούσας διπλωματικής εργασίας αναπτύχθηκε ένα έξυπνο σύστημα διαχείρισης επιστημονικών συνεδρίων, το οποίο υποστηρίζει βασικές λειτουργίες, όπως:

- προσωποποίηση περιεχομένου βάσει των θεμάτων ενδιαφέροντος του χρήστη,
- αξιολόγηση συνεδρίων,
- δυνατότητες κοινωνικής δικτύωσης (π.χ. φιλικές συνδέσεις, επικοινωνία),
- και ψηφιακή διαχείριση συνεδρίων, θεμάτων και χρηστών μέσω RESTful API.

Παρότι το σύστημα δεν είναι πλήρως ολοκληρωμένο, υλοποιήθηκαν και δοκιμάστηκαν βασικά υποσυστήματα του backend, τα οποία οργανώνονται σε ένα συνεκτικό RESTful API με επίκεντρο τις κύριες επιχειρησιακές/πληροφοριακές οντότητες (χρήστες, συνέδρια, θέματα). Η διεπαφή χρήστη (frontend) δεν υλοποιήθηκε πλήρως λόγω περιορισμένου χρόνου, ωστόσο το backend είναι λειτουργικά σταθερό και παρέχει ισχυρό τεχνικό υπόβαθρο για περαιτέρω ανάπτυξη.

Στο μέλλον, το σύστημα μπορεί να επεκταθεί με την προσθήκη των εξής λειτουργιών:

- **Ολοκληρωμένοι μηχανισμοί αυθεντικοποίησης και εξουσιοδότησης (authentication & authorization) όπως:**
  - Υποστήριξη Two-Factor Authentication (2FA) για ενισχυμένη ταυτοποίηση,
  - **Καταγραφή ενεργειών χρηστών (logging/monitoring)** για λόγους audit και ανίχνευσης κακόβουλων ενεργειών,
  - **Rate limiting** για αποτροπή επιθέσεων τύπου brute-force, βελτιώνοντας τη συνολική ασφάλεια και τη διαχείριση πρόσβασης.
- **Αυτόματη άντληση δεδομένων συνεδρίων και CFPs (Call for Papers):** Η μελλοντική ενσωμάτωση μηχανισμών crawling από εξειδικευμένες πηγές (π.χ. WikiCFP, ConfRef, Conference Alerts) θα επιτρέψει την αυτόματη συγκέντρωση και ενημέρωση των διαθέσιμων συνεδρίων, αυξάνοντας την πληρότητα και την επικαιρότητα της πληροφορίας που προσφέρεται στον χρήστη.
- **Αναλυτικά στατιστικά και πίνακες ελέγχου (dashboards):** Η προσθήκη εργαλείων οπτικοποίησης της δραστηριότητας του χρήστη (π.χ. συμμετοχές, αξιολογήσεις, εγγραφές σε θέματα) και η παρουσίαση τάσεων συνεδρίων θα ενίσχυε σημαντικά τη χρηστικότητα της πλατφόρμας και θα παρείχε στους ερευνητές υποστήριξη λήψης αποφάσεων.
- **Εξελιγμένα συστήματα συστάσεων (recommendation systems):** Αν και υπάρχει δυνατότητα προσωποποίησης βάσει επιλεγμένων θεμάτων στο σύστημά μας, η ενσωμάτωση τεχνικών, όπως collaborative filtering (συνεργατικό φιλτράρισμα) ή content-based recommendation (σύσταση βασιζόμενη στο περιεχόμενο), θα προσφέρει δυναμικές και στοχευμένες προτάσεις συνεδρίων, βελτιώνοντας την εμπειρία χρήσης και τη συνάφεια της πληροφόρησης.

Συνολικά, το σύστημα που αναπτύχθηκε θέτει σταθερές βάσεις για την παροχή προηγμένων υπηρεσιών υποστήριξης ερευνητών στην εύρεση, διαχείριση και παρακολούθηση επιστημονικών συνεδρίων, με σημαντικές δυνατότητες για μελλοντική επέκταση και προσαρμογή στις ανάγκες της ερευνητικής κοινότητας.

Βιβλιογραφία

1. **Srinivasa, S., & Rachakonda, A. R. (2008).** *Silverfish: A Contextual Knowledge Extraction System.* International Conference on Management of Data.
2. **Lazarinis, F. (2003).** *Combining Information Retrieval with Information Extraction for Efficient Retrieval of Calls for Papers.* University of Glasgow.
3. **Yadav, Y. O., & Desai, B. C. (2023).** *ConfSys - An Intelligent Conference Management System.* International Database Engineered Applications Symposium (IDEAS).
4. **Vahdati, S., Arndt, N., Auer, S., & Lange, C. (2016).** *OpenResearch: Collaborative Management of Scholarly Communication Metadata.* Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW), 778-793.
5. **Omar, R., Vahdati, S., Lange, C., Vidal, M.-E., & Behrend, A. (2018).** *SAANSET: Semi-Automated Acquisition of Scholarly Metadata Using OpenResearch.org Platform.* IEEE International Conference on Semantic Computing, 41-42.
6. **Zhao, K. (2012).** *ConfSys3: An Online Academic Conference & eJournal System.* Master's Thesis, Concordia University, Montreal, Quebec, Canada.
7. **Arshad, N., Soroya, S. H., Safder, I., Haider, S., Hassan, S. U., Aljohani, N. R., Alelyani, S., & Nawaz, R. (2022).** *Extracting Scientific Trends by Mining Topics from Call for Papers.* Library Hi Tech, 40(1), 115-132.
8. **Defense Acquisition University (DAU).** (n.d.). *Modular Design*. Retrieved from https://www.dau.edu/cop/se/resources/modular-design
9. Pivotal Software, Inc. (2023). *Spring Boot Reference Documentation*. Retrieved from https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
10. Apache Maven Project. (2023). *Introduction to the Build Lifecycle*. Retrieved from https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html
11. Bloch, J. (2018). *Effective Java* (3rd ed.). Addison-Wesley Professional.