Πανεπιστήμιο Αιγαίου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Τεχνολογία Λογισμικού

Διδάσκων: Κυριάκος Κρητικός

Καθηγητής

Conference-User-Management System

Εργαστηριακοί Συνεργάτες: Γεώργιος Χρυσολωράς

icsd20130 Μιχόπουλος Πέτρος

icsd20003 Αϊδίνης Χαράλαμπος

icsd20105 Κωνσταντάρας Ιωάννης

Σάμος, 22-9-2024

Κατάλογος Περιεχομένων

2. Σύνοψη

3. Οργάνωση

3.a. Βασικοί Ρόλοι

3.b. Βασικά Στάδια

4. Απαιτήσεις Συστήματος

4.b. Λειτουργικές

4.2. Μη Λειτουργικές

5. Σχεδίαση

5.a. System Architecture and Context

5.b. Use Cases

5.c. System Behavior

5.d. System Entities

6. Υλοποίηση

6.a. URL GitHub

6.b. Τεκμηρίωση Υλοποίησης

6.c. Τεκμηρίωση Δοκιμής

7. Συμπεράσματα

2. Σύνοψη

Το σύστημα διαχείρισης συνεδρίων που αναπτύσσεται περιλαμβάνει τρεις βασικές οντότητες: Paper (έγγραφο), Conference (συνέδριο) και User (χρήστη), καθένα με ξεχωριστούς ρόλους και ευθύνες. Το backend, υλοποιημένο μέσω RESTful web services και αλληλεπιδρώντας με μια βάση δεδομένων, διευκολύνει την αποτελεσματική διαχείριση συνεδρίων, εγγράφων και χρηστών. Το σύστημα υποστηρίζει έξι ρόλους χρηστών (VISITOR, USER, AUTHOR, PC CHAIR, PC MEMBER, ADMIN), καθέναν με συγκεκριμένα δικαιώματα πρόσβασης και λειτουργιών.

Οι χρήστες μπορούν να δημιουργούν, ενημερώνουν, υποβάλλουν και διαχειρίζονται έγγραφα, ενώ τα συνέδρια προχωρούν μέσα από καταστάσεις όπως CREATED, SUBMISSION, ASSIGNMENT, REVIEW, DECISION, FINAL_SUBMISSION και FINAL. Η διαχείριση χρηστών περιλαμβάνει εγγραφή, πιστοποίηση και έλεγχο πρόσβασης βασισμένο στο ρόλο.

Το σύστημα προτιμά την άμεση εκτέλεση, την αξιοπιστία, την ασφάλεια και τη συναλλαγματικότητα, διασφαλίζοντας συνεπείς ενημερώσεις της βάσης δεδομένων. Σημαντικά χαρακτηριστικά περιλαμβάνουν τις καταστάσεις των εγγράφων και των συνεδρίων, τους ρόλους των χρηστών και έναν πίνακα πρόσβασης βασισμένο στους ρόλους. Το περιβάλλον του συστήματος υπογραμμίζει την αποτελεσματικότητα, την ασφάλεια και τη συμμόρφωση προς τις προκαθορισμένες καταστάσεις και ρόλους.

3. Οργάνωση

3.a.b: Η εργασία χωρίστηκε σε 3 μέρη όσα και τα μέλη της ομάδας. Το πρώτο μέρος της εργασίας με τις λειτουργικές και μη λειτουργικές απαιτήσεις χωρίστηκε ως εξής:

icsd20003: Paper Management

icsd20105: Conference Management

icsd20130: User Management

Στο δεύτερο μέρος με τα διαγράμματα όλα τα μέλη της ομάδας εργάστηκαν ταυτόχρονα μέσω του εργαλείου draw.io

Στο τρίτο μέρος ο κώδικας συγγράφηκε κυριώς ταυτόχρονα από όλα τα μέλη της ομάδας απο κοντά.

4.1 Λειτουργικές Απαιτήσεις

- Paper Creation:

  Το paper πρέπει να είναι μοναδικό για το κάθε conference(συνέδριο).
  Οι χρήστες με το ρόλο AUTHOR μπορούν να δημιουργήσουν ένα νέο paper για ένα συγκεκριμένο conference.
  Ένα paper για να γίνει δεκτό θα πρέπει να έχει έναν μοναδικό τίτλο και να υπάρχουν και όλες οι απαιτούμενες πληροφορίες .
  Το paper ID και το creation date θα πρέπει να δημιουργούνται αυτόματα.
  Ο δημιουργός του paper (requestor) θα παίρνει τον ρόλο του AUTHOR για το conference .

- Paper Update:

  Θα πρέπει να μπορεί να ενημερωθεί ο τίτλος, η περίληψη, οι authors(συγγραφείς), λέξεις-κλειδιά και το περιεχόμενο.

- Co-author Addition:

  Οι συγγραφείς θα μπορούν να ορίσουν χρήστες ως co-authors για το paper .
  Τα ονόματα των χρηστών θα πρέπει να περιλαμβάνονται στα ονόματα των συγγραφέων για να γίνει δεκτό το paper.
  Ο χρήστης, που μετά που γίνει δεκτό το paper, πρέπει να παίρνει ρολό του AUTHOR.

- Paper Submission:

  Μια εργασία μπορεί να υποβληθεί επίσημα σε μια conference αν και αυτή βρίσκεται σε SUBMISSION state.
  Το περιεχόμενο του paper δεν πρέπει να είναι άδειο για να υποβληθεί.

- Reviewer Assignment:

  Ένας reviewer (κριτής) ανατίθεται στο paper αν η conference βρίσκεται σε ASSIGNMENT state.
  Πρέπει να υπάρχει όριο 2 reviewer .

Η ανάθεση επιτρέπεται μόνο εάν ο reviewer είναι μέλος της Program Committee (PC) του conference .

- Paper Review:

  Reviewers, παρέχουν βαθμολογία(0-100) και την αιτιολογία για την βαθμολογία, αν η conference είναι σε REVIEW state.

- Paper Approval:

  Το Paper πρέπει να τροποποιηθεί προκειμένου να ληφθούν υπόψη τα σχόλια των reviewer και μετά μπορεί να γίνει Approved.
  Approval μπορεί να γίνει δεκτή μόνο όταν η conference είναι σε DECISION state.

- Paper Rejection:

  Rejection μπορεί να γίνει δεκτή μόνο όταν η conference είναι σε DECISION state.

- Paper Final Submission:

  Final submission μπορεί να γίνει δεκτό μόνο όταν η conference είναι σε FINAL_SUBMISSION state

- Paper Acceptance::

  Acceptance μπορεί να γίνει δεκτό μόνο όταν η conference είναι σε FINAL state.

- Paper Search::

  Οι χρήστες θα μπορούν να αναζητούν papers με βάση τα ακόλουθα κριτήρια: τίτλος, συγγραφείς και περίληψη.
  Εάν παρέχονται πολλές λέξεις για ένα πεδίο(τίτλος, συγγραφείς, περίληψη), όλες οι λέξεις πρέπει να περιλαμβάνονται στο αντίστοιχο πεδίο για να θεωρηθεί ότι ένα paper ταιριάζει.
  Εάν δεν δίνονται λέξεις για κανένα από τα κριτήρια, όλα τα paper του conference θα θεωρούνται ότι ταιριάζουν.
  Περαιτέρω φιλτράρισμα πρέπει να μπορεί να γίνει ανάλογα με το ρολό του χρήστη που κάνει το search .
  Τα papers που επιστρέφονται στα αποτελέσματα αναζήτησης θα ταξινομηθούν με αύξουσα σειρά με βάση τον τίτλο του paper.
  Η σειρά ταξινόμησης μπορεί να προσαρμόζεται με βάση τις προτιμήσεις του χρήστη.

- Paper View::

  Το σύστημα παρουσιάζει πληροφορίες με βάση το αναγνωριστικό του όταν ένας χρήστης ζητά να δει ένα συγκεκριμένο paper.
  Να μπορεί απορρίψει ορισμένα πεδία βάσει τον ρόλου χρήστη (e.g., AUTHOR, REVIEWER, PC).

- Paper Withdraw::

  Πρέπει ένα paper να μπορούν να το κάνουν withdraw .
  Ένα paper που αποσύρεται(withdrawn) θα διαγραφεί από τα ενεργά paper του conference.

Conference Management Functional Requirements:

1. Conference Creation:

- Conference Name: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να εισάγει το όνομα του συνεδρίου.
- Conference Date: Το σύστημα θα πρέπει να παρέχει επιλογή στο χρήστη να επιλέξει την ημερομηνία του συνεδρίου.
- Conference Location: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να εισάγει την τοποθεσία του συνεδρίου.
- Conference Description: Το σύστημα θα πρέπει να παρέχει ένα πεδίο κειμένου όπου ο χρήστης μπορεί να περιγράψει το συνέδριο.
- Conference Topics: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να καταχωρίζει τα θέματα που θα καλυφθούν στο συνέδριο.
- Speaker Details: Το σύστημα θα πρέπει να παρέχει επιλογή στο χρήστη να εισάγει λεπτομέρειες σχετικά με τους ομιλητές στο συνέδριο.

2. Conference Update:

- Access Conference: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να έχει πρόσβαση στο συνέδριο που θέλει να ενημερώσει.
- Edit Conference Details: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να επεξεργάζεται τα στοιχεία του συνεδρίου, όπως το όνομα, η ημερομηνία, η τοποθεσία και η περιγραφή.
- Update Conference Topics: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να ενημερώνει τη λίστα των θεμάτων που θα καλυφθούν στο συνέδριο.
- Update Speaker Details: Το σύστημα θα πρέπει να παρέχει επιλογή στο χρήστη να ενημερώνει τα στοιχεία σχετικά με τους ομιλητές στο συνέδριο.
- Update Registration Details: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να ενημερώνει τα στοιχεία εγγραφής, όπως η προθεσμία εγγραφής, το κόστος εγγραφής, κ.λπ.
- Update Submission Details: Αν το συνέδριο περιλαμβάνει παρουσιάσεις εργασιών, το σύστημα θα πρέπει να επιτρέπει στο χρήστη να ενημερώνει τα στοιχεία υποβολής, όπως η προθεσμία υποβολής, το μορφότυπο της εργασίας, κ.λπ.
- Save Updates: Αφού ολοκληρωθούν όλες οι ενημερώσεις, το σύστημα θα πρέπει να επιτρέπει στο χρήστη να αποθηκεύει τις αλλαγές.

3. PC Chairs Addition:

- PC Chair Addition: Το σύστημα θα πρέπει να επιτρέπει την προσθήκη PC Chairs εισάγοντας τα στοιχεία τους, όπως όνομα, σύνδεση με οργανισμό, email, και τομείς ειδίκευσης.
- PC Chair Update: Το σύστημα θα πρέπει να επιτρέπει την ενημέρωση των λεπτομερειών του PC Chair. Αυτό περιλαμβάνει την ενημέρωση του ονόματός τους, του οργανισμού τους, του email και των τομέων ειδίκευσής τους.
- PC Chair Removal: Το σύστημα θα πρέπει να επιτρέπει την αφαίρεση των PC Chairs από το συνέδριο.
- PC Chair List: Το σύστημα θα πρέπει να παρέχει μια λίστα με όλους τους PC Chairs που σχετίζονται με ένα συνέδριο.
- PC Chairs Assignment: Το σύστημα θα πρέπει να επιτρέπει την ανάθεση καθηκόντων σε συγκεκριμένους PC Chairs, όπως η αξιολόγηση εργασιών, η διευθέτηση συνεδριάσεων, κ.λπ.

4.PC Members Addition:

- PC Member Addition: Το σύστημα θα πρέπει να επιτρέπει την προσθήκη PC Member εισάγοντας τα στοιχεία τους, όπως όνομα, σύνδεση με οργανισμό, email, και τομείς ειδίκευσης.
- PC Member Update: Το σύστημα θα πρέπει να επιτρέπει την ενημέρωση των λεπτομερειών του PC Member. Αυτό περιλαμβάνει την ενημέρωση του ονόματός τους, του οργανισμού τους, του email και των τομέων ειδίκευσής τους.
- PC Member Removal: Το σύστημα θα πρέπει να επιτρέπει την αφαίρεση των PC Members από το συνέδριο.
- PC Member List: Το σύστημα θα πρέπει να παρέχει μια λίστα με όλους τους PC Members που σχετίζονται με ένα συνέδριο.

5. Conference Search:

- Search Input: Το σύστημα θα πρέπει να παρέχει μια γραμμή αναζήτησης όπου οι χρήστες μπορούν να εισάγουν τους όρους αναζήτησής τους.
- Search Execution: Το σύστημα θα πρέπει να εκτελεί την αναζήτηση όταν ο χρήστης πατάει Enter ή κάνει κλικ σε ένα κουμπί αναζήτησης.
- Search Filters: Το σύστημα θα πρέπει να παρέχει φίλτρα για τον περιορισμό των αποτελεσμάτων αναζήτησης. Τα φίλτρα μπορεί να περιλαμβάνουν ημερομηνία συνεδρίου, τοποθεσία, θέματα κλπ.
- Search Results: Το σύστημα θα πρέπει να εμφανίζει τα αποτελέσματα αναζήτησης με ξεκάθαρο και οργανωμένο τρόπο. Κάθε αποτέλεσμα θα πρέπει να περιλαμβάνει βασικές λεπτομέρειες του συνεδρίου, όπως όνομα, ημερομηνία, τοποθεσία και μια σύντομη περιγραφή.
- Result Selection: Το σύστημα θα πρέπει να επιτρέπει στον χρήστη να επιλέγει ένα αποτέλεσμα αναζήτησης για να δει περισσότερες λεπτομέρειες σχετικά με το συνέδριο.

6. Conference View:

- Conference Selection: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να επιλέγει ένα συνέδριο που θέλει να δει από μια λίστα συνεδρίων.
- Conference Details: Το σύστημα θα πρέπει να εμφανίζει λεπτομερείς πληροφορίες σχετικά με το επιλεγμένο συνέδριο, συμπεριλαμβανομένων του ονόματός του, της ημερομηνίας, της τοποθεσίας, της περιγραφής, των θεμάτων, των ομιλητών, των λεπτομερειών εγγραφής και των λεπτομερειών υποβολής.
- Conference Schedule: Το σύστημα θα πρέπει να εμφανίζει το πρόγραμμα του συνεδρίου, συμπεριλαμβανομένων των ημερομηνιών και ωρών διάφορων συνεδριάσεων και εκδηλώσεων.
- Conference Registration: Εάν ο χρήστης ενδιαφέρεται να παρακολουθήσει το συνέδριο, το σύστημα θα πρέπει να παρέχει επιλογή για εγγραφή στο συνέδριο.
- Conference Updates: Το σύστημα θα πρέπει να εμφανίζει οποιεσδήποτε ενημερώσεις ή ανακοινώσεις σχετικά με το συνέδριο.

7. Conference Deletion:

- Conference Selection: Το σύστημα θα πρέπει να επιτρέπει στο χρήστη να επιλέγει ένα συνέδριο που θέλει να διαγράψει από μια λίστα συνεδρίων που έχει δημιουργήσει.
- Delete Confirmation: Το σύστημα θα πρέπει να ζητά από τον χρήστη να επιβεβαιώσει ότι θέλει να διαγράψει το επιλεγμένο συνέδριο. Αυτό γίνεται για να αποτρέψει τυχόν ακούσιες διαγραφές.
- Conference Deletion: Εάν ο χρήστης επιβεβαιώσει, το σύστημα θα πρέπει να διαγράφει το συνέδριο από το σύστημα.
- Notification: Το σύστημα θα πρέπει να ενημερώνει τον χρήστη ότι το συνέδριο διαγράφηκε με επιτυχία.
- Update List: Το σύστημα θα πρέπει να ενημερώνει τη λίστα των συνεδρίων για να αντικατοπτρίζει τη διαγραφή.

8. Conference State Transitions:

- Submission start, Reviewer assignment start, Review start, Decision making, Final submission start, and Conference end θα επιτρέπονται μόνο σε συγκεκριμένες καταστάσεις του συνεδρίου.

1.User registration :

Το σύστημα θα πρέπει να επιτρέπει στους μη εγγεγραμμένους χρήστες να μπορούν να κάνουν αίτημα για εγγραφή.Τα Usernames θα πρέπει να είναι μοναδικά , θα πρέπει να αρχίζουν πάντα με ένα γράμμα και να αποτελούνται από τουλάχιστον πέντε χαρακτήρες οι οποίοι θα είναι αλφαριθμητικά ή ένας ειδικός χαρακτήρας ‘_’.Οι κωδικοί πρόσβασης θα πρέπει να υπάγονται σε σχετικούς ελέγχους με βάση κάποια κριτήρια ασφάλειας όπως το μήκος αλλά και η πολυπλοκότητα. Ακόμα, θα πρέπει να περιέχουν το λιγότερο 8 χαρακτήρες συμπεριλαμβανομένου κεφαλαίων και πεζών γραμμάτων , αριθμών αλλά και ειδικών χαρακτήρων.Τέλος, θα πρέπει να μετά την εγγραφή του χρήστη ο λογαριασμός του θα πρέπει να μπαίνει σε αναστολή έως ότου κάποιος διαχειριστής να ελέγξει τα στοιχεία του.

2.User Information update:

Οι εγγεγραμμένοι χρήστης και ο Διαχειριστής θα μπορούν να αλλάζουν όλες τις πληροφορίες του χρήστη εκτός από τον κωδικό πρόσβασης.Ο χρήστης που θα εκτελεί το αίτημα θα πρέπει να είναι ήδη authenticated.Τέλος, μετά την αλλαγή του Username το πλέον παλιό Username θα πρέπει να ακυρώνεται.

3.User Password Update :

Ο κάθε Χρήστης θα μπορεί κάνει αίτημα για αλλαγή κωδικού πρόσβασης δίνοντας τον παλιό κωδικό στο σύστημα αλλά και εισάγοντας ταυτόχρονα έναν καινούργιο.Επιπλέον, μετά από τρεις συνεχόμενες αποτυχημένες ενημερώσεις κωδικού του ίδιου χρήστη ο λογαριασμός του θα πρέπει να απενεργοποιείται.Σε κάθε περίπτωση το token του χρήστη θα πρέπει να ακυρωθεί.

4.Account status update:

Ο Διαχειριστής του Συστήματος θα μπορεί να έχει το δικαίωμα να ενεργοποιεί ή να απενεργοποιεί τους Λογαριασμούς των χρηστών.Με την απενεργοποίηση του Λογαριασμού θα πρέπει να ακυρώνεται το Token του συγκεκριμένου λογαριασμού.Η απενεργοποίηση του λογαριασμού αποτρέπει τον χρηστή από το να εκτελέσει συγκεκριμένες ενέργειες στο σύστημα αλλά και το να εκτελεί λειτουργίες στο Paper και στο Conference.

5.User deletion:

Ο Διαχειριστής του συστήματος ή ο εγγεγραμμένος χρήστης θα μπορεί να κάνει αίτημα για διαγραφή του Λογαριασμού του.Όταν γίνει εκτέλεση της συγκεκριμένης λειτουργίας ο Λογαριασμός και το Token που σχετίζεται με τον λογαριασμό θα διαγράφονται.

6.User Authentication:

Οι χρήστες θα πρέπει να ταυτοποιούνται με ένα Όνομα και ένα Κωδικό πρόσβασης.Η επιτυχής ταυτοποίηση θα παράγει ένα Token.Οι τρεις συνεχόμενες ανεπιτυχείς προσπάθειες σύνδεσης στο λογαριασμό από έναν εγγεγραμμένο χρηστή θα απενεργοποιούν τον λογαριασμό.

7.User token validation:

Με την αυθεντικοποίηση ο χρήστης θα μπορεί να περιηγηθεί σε αρκετές λειτουργίες που προσφέρονται από το σύστημα με την αξιοποίηση του εκάστοτε Token.Το σύστημα θα πρέπει να ελέγχει την κατάσταση του εκάστοτε Token εάν είναι Valid, σε ποιον ανήκει αλλά και εάν αυτό έχει λήξει. Τέλος, εάν ένα Token είναι έγκυρο αλλά ανήκει σε ταυτόχρονα δύο χρήστες θα πρέπει οι λογαριασμοί των χρηστών να απενεργοποιηθούν.

4.2 Μη Λειτουργικές Απαιτήσεις

1.Performance: Οι λειτουργίες του συστήματος πρέπει να εκτελούνται γρήγορα (το πολύ 5-10 δευτερόλεπτα ανά request).

2.Reliability:

Δεν πρέπει να παρουσιάζει internal errors.Χρειάζεται να παράγει κατάλληλα μηνύματα σφάλματος σε περίπτωση που χρησιμοποιηθεί εσφαλμένα από τους χρήστες του. Το σύστημα πρέπει να αποφεύγει την πολλαπλή εκτέλεση της ίδιας επιτυχώς εκτελούμενης λειτουργίας εκτός η λειτουργία είναι idempotent.

3.Data Storage:

Το σύστημα πρέπει να αποθηκεύει με ασφάλεια δεδομένα χρήστη, λεπτομέρειες για το paper και πληροφορίες για την conference. Θα πρέπει να εφαρμόζονται διαδικασίες δημιουργίας αντιγράφων ασφαλείας και ανάκτησης για την πρόληψη της απώλειας δεδομένων. Πρέπει να υπάρχει συνέπεια στον τρόπο ενημέρωσης της βάσης δεδομένων, ενώ αυτή η ενημερωμένη έκδοση πρέπει να είναι επιτυχής μόνο εάν η αντίστοιχη λειτουργία του συστήματος (π.χ. Conference UPDATE) έχει εκτελεστεί πλήρως και σωστά.

4.Security:οι κωδικοί θα πρέπει να έχουν αποθηκευτεί σε ασφαλές μέρος και να ακολουθούν συγκεκριμένες τεχνικές κρυπτογραφίας.

5.Authentication: Ο χρήστης του συστήματος θα πρέπει να ταυτοποιείται.

6.Error Handling: Θα πρέπει να υπάρχουν συγκεκριμένα μνήματα σφάλματος για την κάθε περίπτωση λειτουργίας. Ακόμα, σε περίπτωση που ένα Token είναι έγκυρο αλλά ανήκει σε ταυτόχρονα δύο χρήστες θα πρέπει οι λογαριασμοί των χρηστών να απενεργοποιηθούν.

7.**Usability:** Η διεπαφή χρήστη για την εγγραφή, τη διαχείριση λογαριασμού και άλλες λειτουργίες πρέπει να είναι εύχρηστη και φιλική προς τον χρήστη. Τα μηνύματα σφάλματος πρέπει να είναι κατανοητά και να παρέχουν καθοδήγηση στους χρήστες.

8.**Compatibility:** Το σύστημα πρέπει να είναι συμβατό με διάφορες συσκευές για να διασφαλίζεται η προσβασιμότητα για τους χρήστες. Η συμβατότητα με σχετικά πρότυπα και πρωτόκολλα ασφαλείας πρέπει να διατηρείται.

9.**Scalability:**Το σύστημα θα πρέπει να είναι σχεδιασμένο έτσι ώστε να δέχεται μεγάλο αριθμό επισκεπτών και δεδομένων. Η βάση δεδομένων θα πρέπει να είναι επεκτάσιμη.

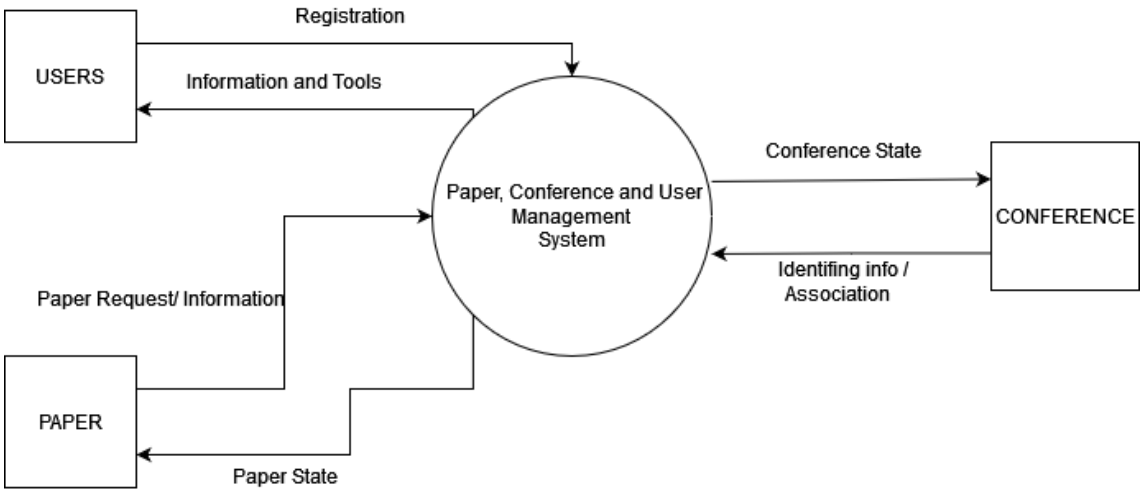
10.**Auditability:** Στο σύστημα θα πρέπει να παρέχονται αρχεία καταγραφής (log Files) στα οποία θα καταγράφονται τα στοιχεία του χρηστή, ημερομηνία , ώρα , οι ενέργειες που εκτέλεσε αλλά και λεπτομέρειες.

11.**User Account Deactivation:** Άμεση απενεργοποίηση λογαριασμού του χρηστή εάν γίνει αίτημα .

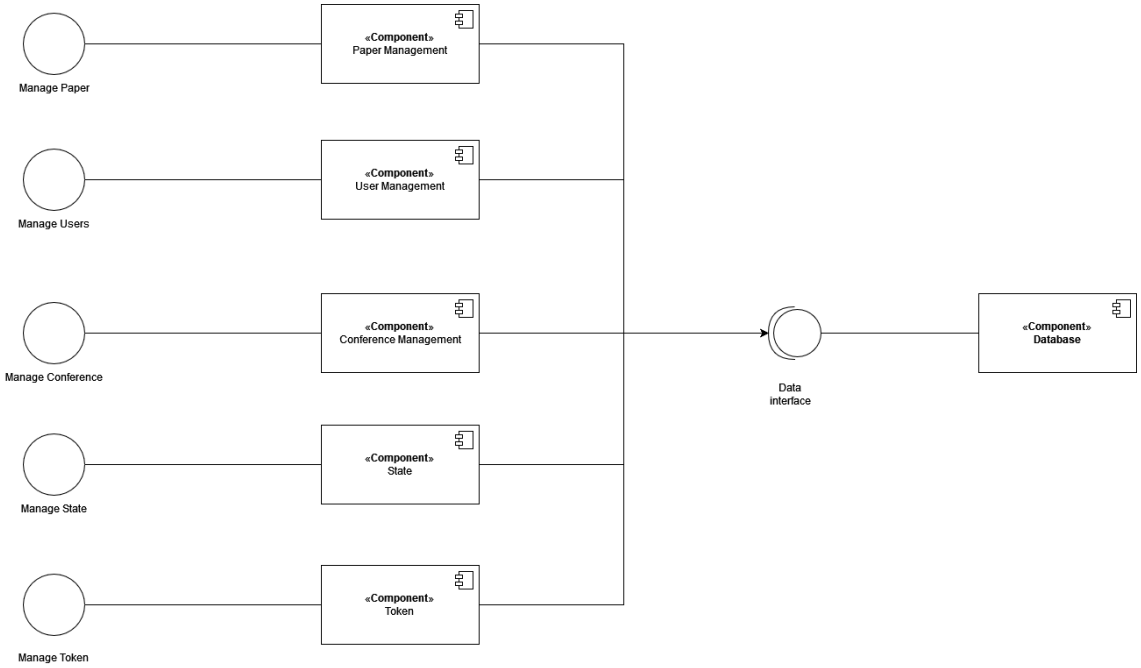
5 Σχεδίαση

5.a System Architecture and Context

Context Diagram:

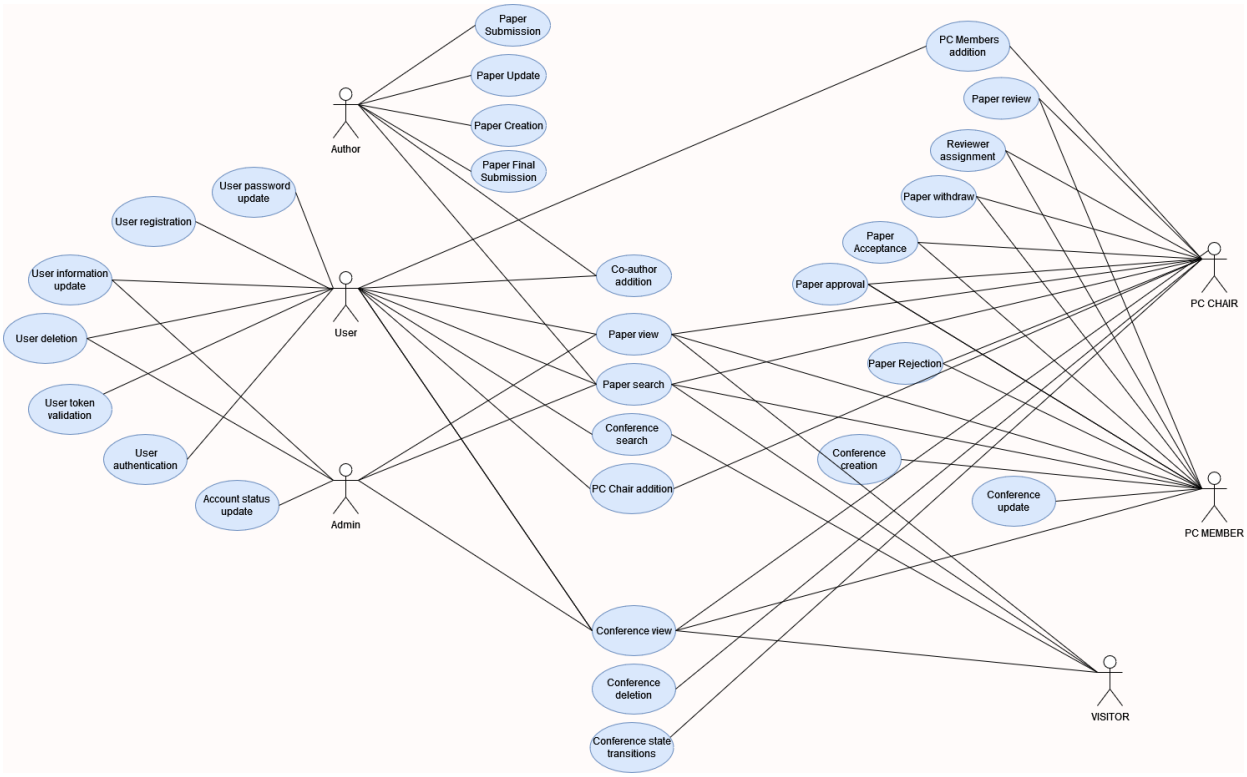


Context Diagram:



5.b Use Cases

USE CASE DIAGRAM



Author μπορεί:

- Να κάνει Paper Submission
- Να αλλαγές σε κάποιο Paper (Paper Update)
- Να δημιουργήσει Paper (Paper Creation)
- Να κάνει την τελική υποβολή (Paper Final Submission)
- Να βάλει στο paper κάποιον ως co-author
- Να αναζητήσει κάποιο Paper

User μπορεί:

- Να κάνει εγγραφή (User registration)
- Να αλλάξει τον κωδικό του (User password update)
- Να αλλάξει τις πληροφορίες του λογαριασμού του (User information update)
- Να διαγράψει τον λογαριασμό του(User Deletion)
- Να κάνει αυθεντικοποίηση (User authentication)
- Να κάνει validate το token του (User token Validation)
- Να διαβάσει κάποιο Paper (Paper View)
- Να αναζητήσει κάποιο Paper (Paper Search)
- Να αναζητήσει κάποιο Conference (Conference Search)
- Να παρακολουθήσει κάποιο conference (Conference View)
- Να γίνει PC Member
- Να γίνει Co-author

Admin μπορεί:

- Να αλλάξει το status λογαριασμών (Account Status Update)
- Να διαγράψει έναν User (User Deletion)
- Να αλλάξει τις πληροφορίες ενός user (User information update)
- Να διαβάσει κάποιο Paper (Paper View)
- Να παρακολουθήσει κάποιο conference (Conference View)

PC Chair μπορεί:

- Να προσθέσει PC Members (PC Members Addition)
- Να κάνει ανασκόπηση σε ένα Paper (Paper Review)
- Να δώσει τον ρολό του Reviewer (Reviewer Assignment)
- Να αποσύρει ένα Paper (Paper Withdraw)
- Να αποδεχτεί ένα Paper (Paper Acceptance)
- Να εγκρίνει ένα Paper (Paper Approval)
- Να απορρίψει ένα Paper (Paper Rejection)
- Να προσθέσει PC Chair (PC Chair Addition)
- Να αναζητήσει κάποιο Paper
- Να διαβάσει ένα Paper
- Να παρακολουθήσει κάποιο conference (Conference View)

- Να διαγράψει ένα conference (Conference Deletion)
- Να αλλάξει το state του conference (Conference State Transition)

PC Member μπορεί:

- Να κάνει ανασκόπηση σε ένα Paper (Paper Review)
- Να δώσει τον ρολό του Reviewer (Reviewer Assignment)
- Να αποσύρει ένα Paper (Paper Withdraw)
- Να αποδεχτεί ένα Paper (Paper Acceptance)
- Να εγκρίνει ένα Paper (Paper Approval)
- Να απορρίψει ένα Paper (Paper Rejection)
- Να αναζητήσει κάποιο Paper
- Να διαβάσει ένα Paper
- Να παρακολουθήσει κάποιο conference (Conference View)

Visitor μπορεί:

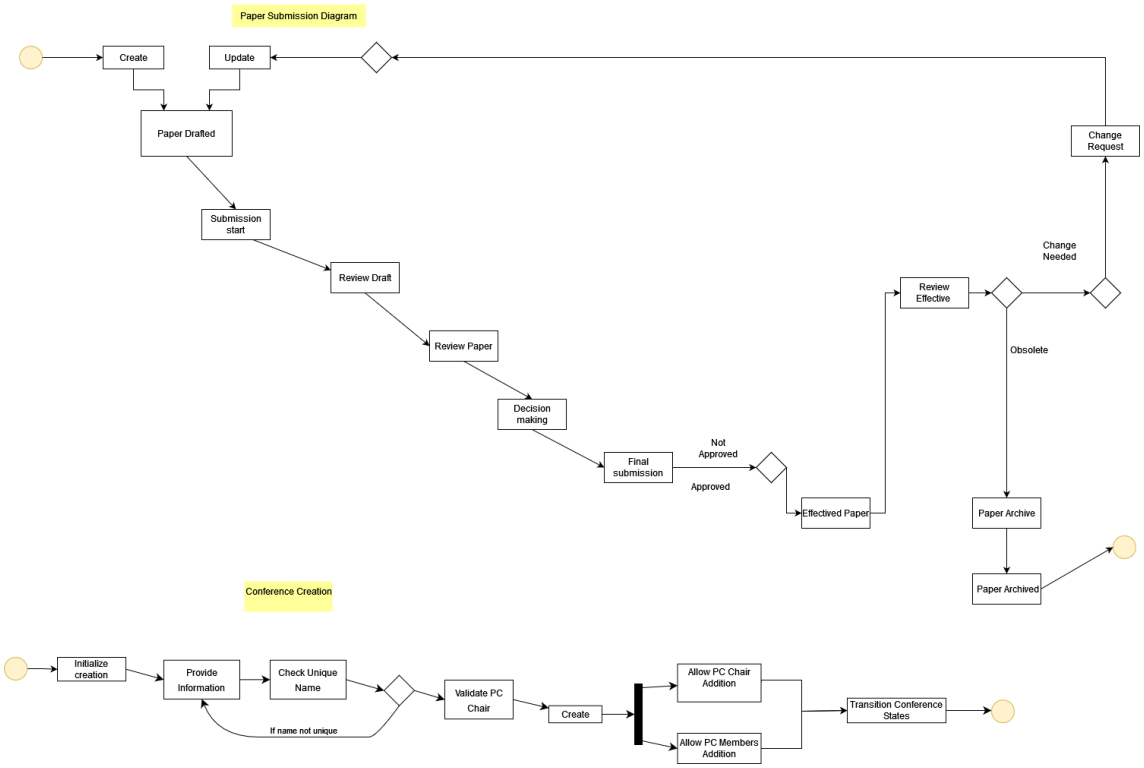
- Να παρακολουθήσει κάποιο conference (Conference View)
- Να αναζητήσει κάποιο Paper
- Να διαβάσει ένα Paper
- Να αναζητήσει κάποιο Conference

5.c: System Behavior

Activity Diagrams:

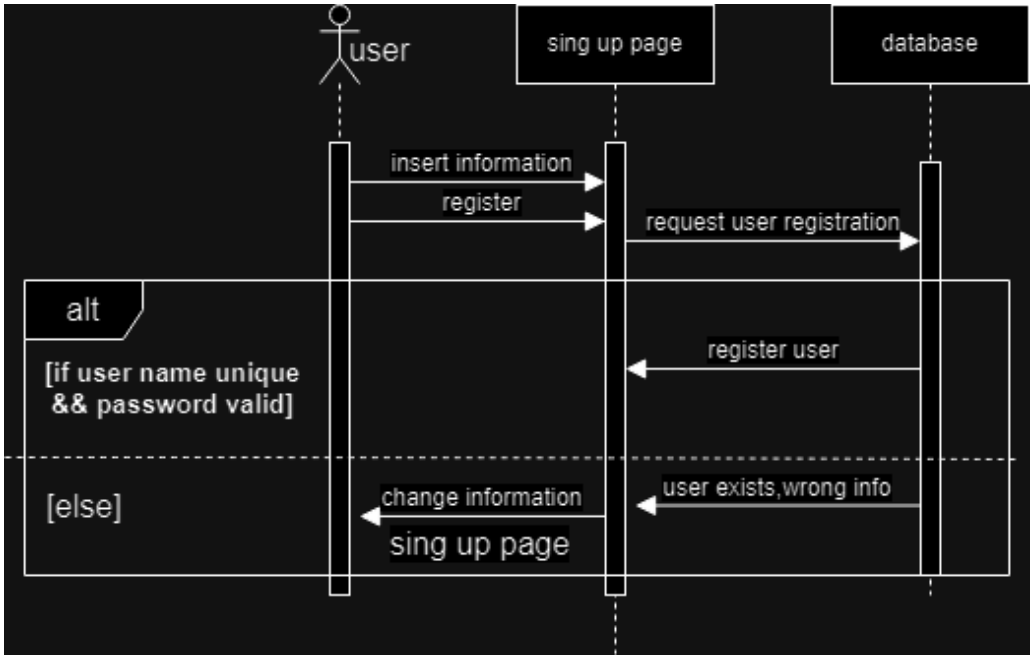
Ο σκοπός των διαγραμμάτων δραστηριοτήτων (activity diagrams) είναι να παρουσιάσουν την λογική κάθε λειτουργίας που πραγματοποιείται στο σύστημα χωρισμένη σε πιο απλά βήματα μέσα από ένα σύνολο ενεργειών και αποφάσεων.

Τα παρακάτω διαγράμματα περιγράφουν της λειτουργίες του παρόντος συστήματος.

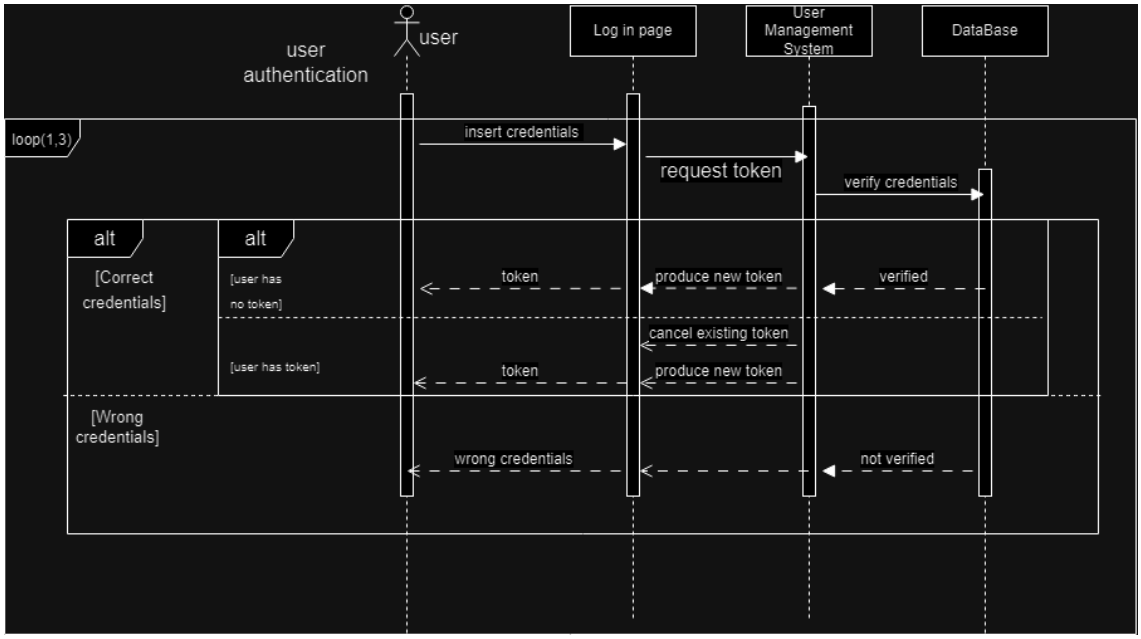


Sequence Diagrams:

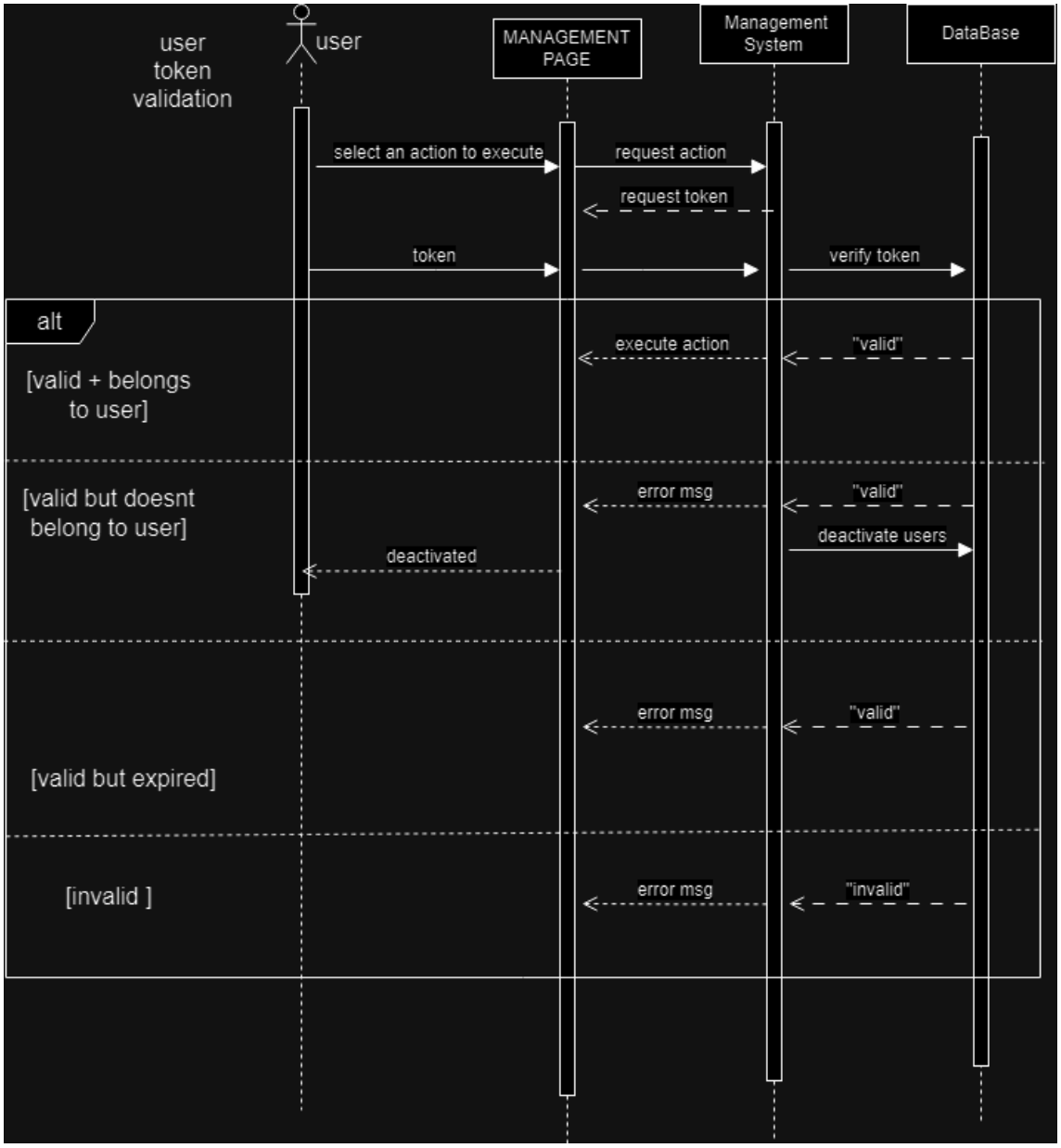
SING UP -- REGISTER



USER AUTHENTICATION

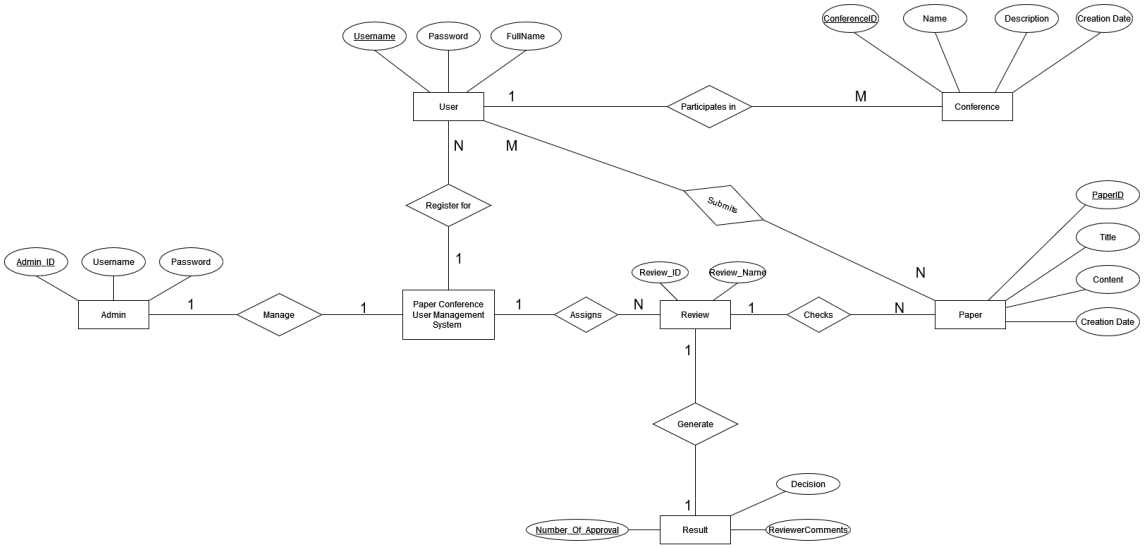


USER TOKEN VALIDATION



5.d: System Entities

Entity-Relationship Diagram



6 Σχεδίαση

6.a. URL GitHub:

https://github.com/CharalamposAdn/UCP-Management

6.b. Τεκμηρίωση Υλοποίησης

Ενδεικτικά κομμάτια κώδικα

Η entity κλάση Conference

```java
@Data
@Entity
public class Conference {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String name;

    @Column(nullable = false)
    private String description;

    @Enumerated(EnumType.STRING)
    private ConferenceState state;

    @Temporal(TemporalType.TIMESTAMP)
    private Date createdDate;

    // Relationship with User (PC Chairs)
    @ManyToMany
    @JoinTable(
        name = "conference_pc_chairs",
        joinColumns = @JoinColumn(name = "conference_id"),
        inverseJoinColumns = @JoinColumn(name = "user_id"))
    private Set<User> pcChairs = new HashSet<>();

    // Relationship with User (PC Members)
    @ManyToMany
    @JoinTable(
        name = "conference_pc_members",
        joinColumns = @JoinColumn(name = "conference_id"),
        inverseJoinColumns = @JoinColumn(name = "user_id"))
    private Set<User> pcMembers = new HashSet<>();

    // Relationship with Papers
    @OneToMany(mappedBy = "conference")
    private Set<Paper> papers = new HashSet<>();
}
```

Η κλάση ConferenceController

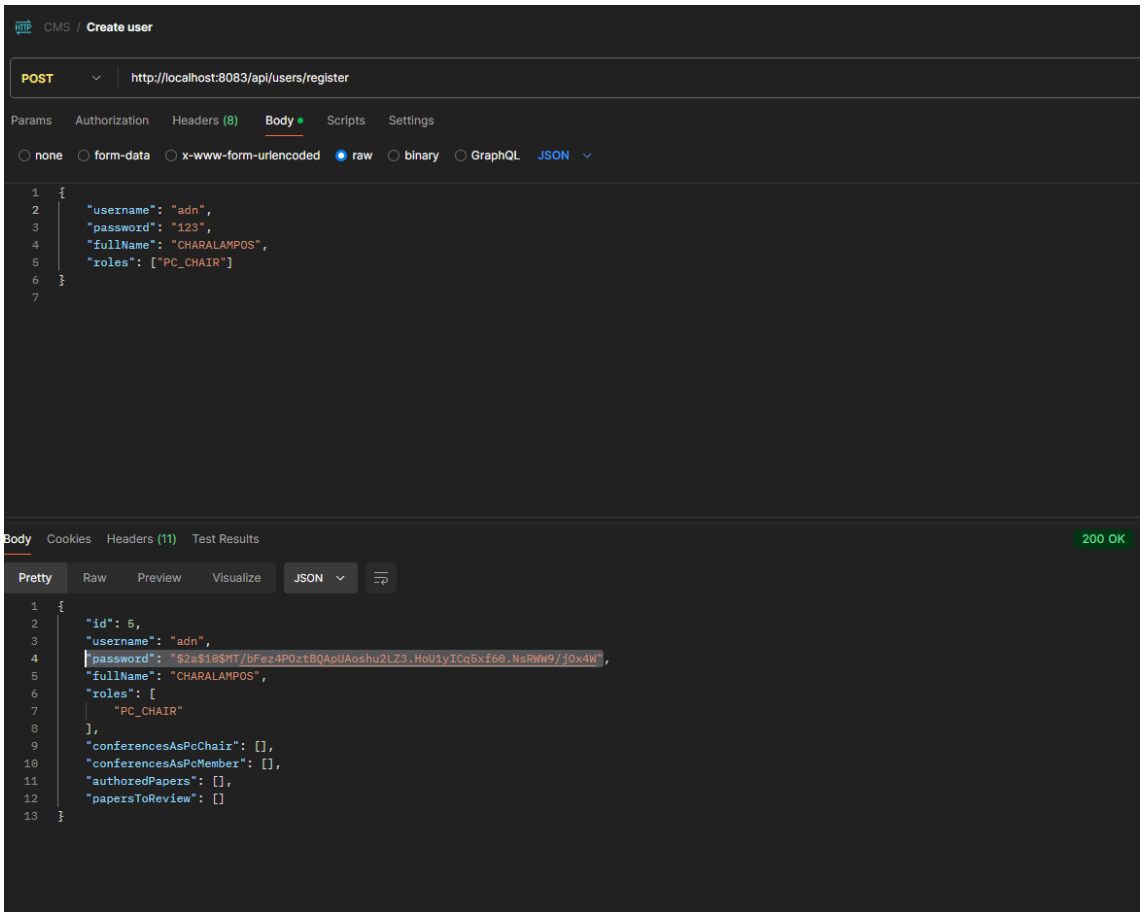
```java
@RequestMapping("/api/conferences")
public class ConferenceController {

    @Autowired
    private ConferenceService conferenceService;

    // Create a new conference
    // Only PC Chair or Admin can create a new conference
    @PreAuthorize("hasRole('ADMIN') or hasRole('PC_CHAIR')")
    @PostMapping
    public ResponseEntity<Conference> createConference(@RequestBody Conference conference) {
        Conference createdConference = conferenceService.createConference(conference);
        return ResponseEntity.ok(createdConference);
    }

    // Get a conference by ID
    // Anyone with PC Chair, PC Member, or Admin role can view conferences
    @PreAuthorize("hasAnyRole('PC_CHAIR', 'PC_MEMBER', 'ADMIN')")
    @GetMapping("/{id}")
    public ResponseEntity<Optional<Conference>> getConferenceById(@PathVariable Long id) {
        Optional<Conference> conference = conferenceService.getConferenceById(id);
        return ResponseEntity.ok(conference);
    }

    // Search for conferences by name (available to all roles)
    @PreAuthorize("hasAnyRole('PC_CHAIR', 'PC_MEMBER', 'ADMIN', 'USER')")
    @GetMapping("/search")
    public ResponseEntity<List<Conference>> searchConferencesByName(@RequestParam String name) {
        List<Conference> conferences = conferenceService.searchConferencesByName(name);
        return ResponseEntity.ok(conferences);
    }

    // Update a conference
    // Only PC Chair or Admin can update a conference
    @PreAuthorize("hasRole('PC_CHAIR') or hasRole('ADMIN')")
    @PutMapping("/{id}")
    public ResponseEntity<Conference> updateConference(@PathVariable Long id, @RequestBody Conference conferenceDetails) {
        Conference updatedConference = conferenceService.updateConference(id, conferenceDetails);
        return ResponseEntity.ok(updatedConference);
    }

    // Delete a conference
    // Only PC Chair or Admin can delete a conference
    @PreAuthorize("hasRole('PC_CHAIR') or hasRole('ADMIN')")
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteConference(@PathVariable Long id) {
        conferenceService.deleteConference(id);
        return ResponseEntity.noContent().build();
    }
```

Η κλάση ConferenceService με όλες τις λειτουργίες του Conference

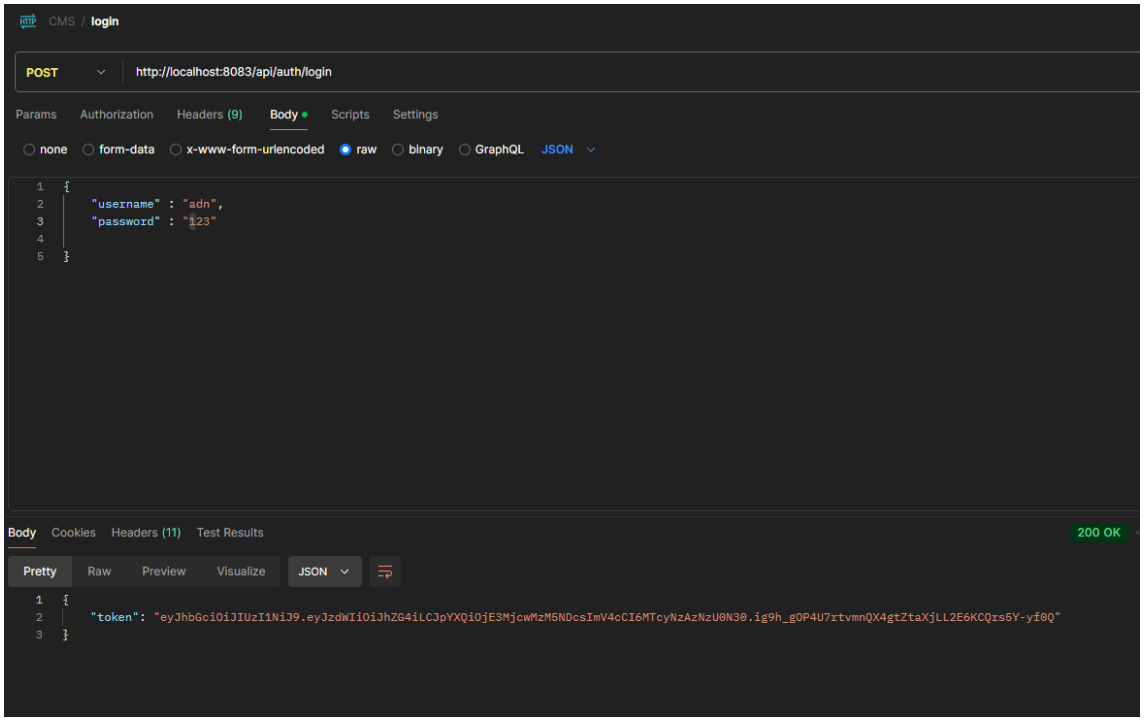
```java
import java.util.List;
import java.util.Optional;

@Service
public class ConferenceService {

    @Autowired
    private ConferenceRepository conferenceRepository;

    public Conference createConference(Conference conference) {…

    public Optional<Conference> getConferenceById(Long id) {…

    public List<Conference> searchConferencesByName(String name) {…

    public Conference updateConference(Long id, Conference conferenceDetails) {…

    public void deleteConference(Long id) {…

    //add PC Chair
    public Conference addPCChair(Long id, Long userId) {…

    //add PC Member
    public Conference addPCMember(Long id, Long userId) {…

    //start Submission Phase
    public Conference startSubmission(Long id) {…

    //start Reviewer Assignment Phase
    public Conference startReviewerAssignment(Long id) {…

    //start Review Phase
    public Conference startReview(Long id) {…

    //start Decision Phase
    public Conference startDecision(Long id) {…

    //start Final Submission Phase
    public Conference startFinalSubmission(Long id) {…

    //end Conference
    public Conference endConference(Long id) {…

    //helper to update conference state
    private Conference updateConferenceState(Long id, ConferenceState newState) {…

    //dummy method to fetch a User by ID (you would implement this based on your system)
    private User fetchUserById(Long userId) {
```

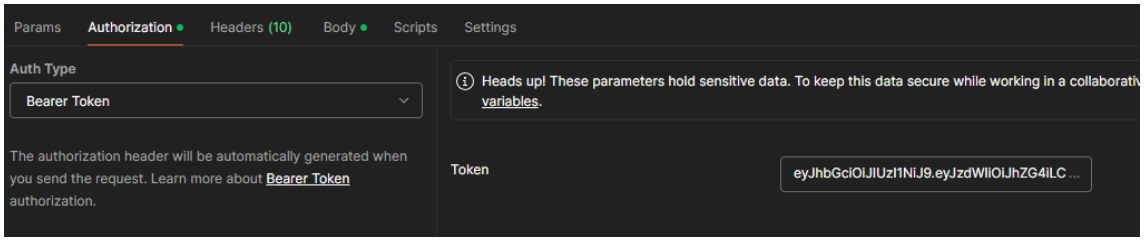
6.c. Τεκμηρίωση Δοκιμής:

Αίτημα POST για εγγραφή χρήστη PC_CHAIR

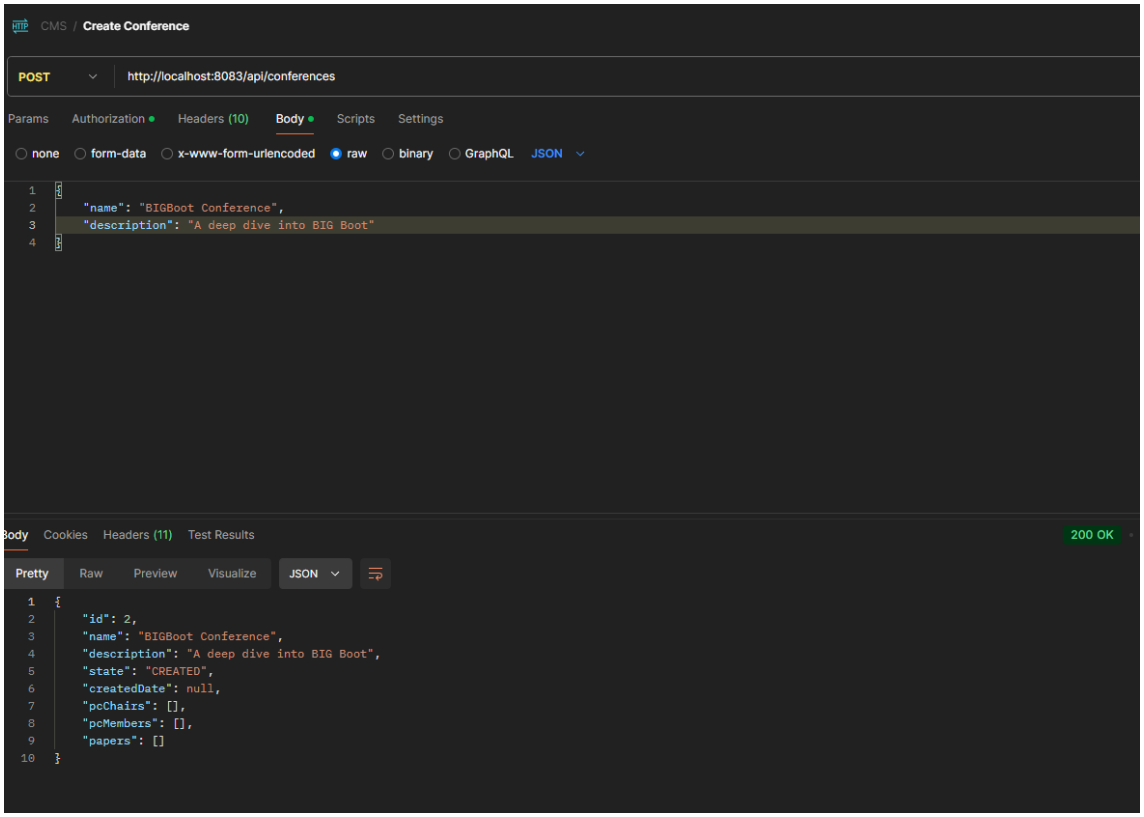


Σύνδεση του χρήστη και δημιουργία Bearer token

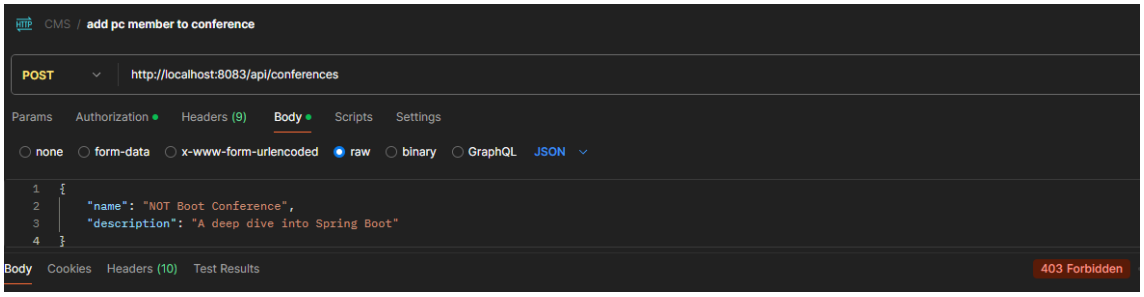




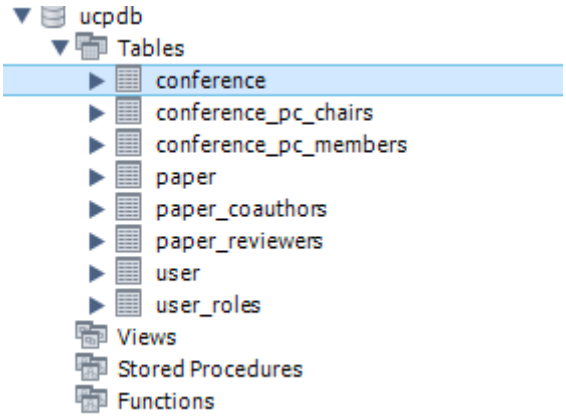
Δημιουργία Conference



Ο PC_MEMBER δεν προστέθηκε γιατί ο συνδεδεμένος χρήστης δεν έχει την συγκεκριμένη άδεια



Φωτογραφία της βάσης στην οποία αποθηκεύονται οι χρήστες αλλά και τα στοιχεία τους .



Συμπεράσματα

Από αυτό το project τα μέλη της ομάδας εξοικειώθηκαν με την υλοποίηση του back-end κώδικα μιας εφαρμογής σε Maven Java και την χρήση της RESTful υπηρεσίας Springboot. Τα προβλήματα τα οποία αντιμετώπισε η ομάδα περιορίστηκαν κυρίως στην υλοποίηση ορισμένων διαγραμμάτων στο 2ο μέρος της εργασίας όπως τα sequence diagrams.