ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Τεχνολογία Λογισμικού

Ομάδα ανάπτυξης:
Νικόλαος Καραγιάννης 3212019080
Σωτήριος Φλασκής 3212019235

Team Project στην Τεχνολογία Λογισμικού
ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2023 – 2024

Πίνακας περιεχομένων

Κεφάλαιο 1. Περίληψη

Το σύστημα που αναπτύχθηκε είναι μια ολοκληρωμένη Πλατφόρμα Διαχείρισης Conference που στηρίχθηκε στην ομαδική δουλεια, δίνοντας έμφαση στην ανάπτυξη λογισμικού. Η υλοποίηση στηριζεται σε εργαλεια οπως το Maven, το Git, το JUnit, και το Jersey framework για την ανάπτυξη υπηρεσιών RESTful. Συγκεκριμένα, το έργο περιλαμβάνει τη δημιουργία ενός backend για το σύστημα διαχείρισης συνεδρίων με RESTful υπηρεσίες που επικοινωνούν με μια βάση δεδομένων.

Σε ό,τι αφορά τη λειτουργικότητα, το σύστημα περιλαμβάνει διάφορες λειτουργίες για τη διαχείριση των conference και paper. Για το paper περιλαμβάνει δημιουργία, ενημέρωση, προσθήκη συν-συγγραφέων, υποβολή, ανάθεση αξιολογητών, αξιολόγηση, έγκριση, απόρριψη, τελική υποβολή, αποδοχή, αναζήτηση, προβολή και απόσυρση. Στη διαχείριση των συνεδριών, οι λειτουργίες περιλαμβάνουν δημιουργία, ενημέρωση, προσθήκη προέδρων και μελών PC, αναζήτηση, προβολή, διαγραφή και μεταβάσεις κατάστασης.

Οι ρόλοι των χρηστών καθορίζονται ως VISITOR (επικεντρωμένος στην προβολή και αναζήτηση), AUTHOR (συμμετέχει σε λειτουργίες που αφορούν τα paper), PC CHAIR (υπεύθυνος για τη διαχείριση των conference) και PC MEMBER (συμμετέχει στις αξιολογήσεις των paper). Οι χρήστες μπορούν να έχουν πολλαπλούς ρόλους, εκτός από το να είναι ταυτόχρονα PC CHAIR και PC MEMBER για το ίδιο conference.

Οι οντότητες που διαχειρίζεται το σύστημα περιλαμβάνουν Papers, Conferences, and Users. Τα Papers μεταβάλλονται ως CREATED, SUBMITTED, REVIEWED, REJECTED, APPROVED, και το ACCEPTED. Με την ίδια λογική τα Conferences τα χειριζόμαστε με το CREATED, SUBMISSION, ASSIGNMENT, REVIEW, DECISION, FINAL_SUBMISSION, και το FINAL . Η πιστοποίηση ταυτότητας των χρηστών είναι προϋπόθεση, και το σύστημα διασφαλίζει τους ρόλους και την ταυτοποίηση των χρηστών.

1.1.1 Πρώτο Μέρος Project

Functional Requirements:

Χρήστης:

- Εγγραφή Χρήστη
- Σύνδεση και Αυθεντικοποίηση
- Ρόλοι χρήστη

Εγγραφά:

- Δημιουργία Εγγράφου
- Ενημέρωση Εγγράφου
- Προσθήκη δεύτερου Συγγραφέα
- Υποβολή Εγγράφου
- Εξέταση Εγγράφου
- Έγκριση Εγγράφου
- Απόρριψη Εγγράφου
- Τελική υποβολή Εγγράφου

- Αποδοχή Εγγράφου
- Αναζήτηση Εγγράφου
- Προβολή Εγγράφου
- Διαγραφή Εγγράφου

Συνεδρίες:

- Δημιουργία Συνεδρίας
- Ενημέρωση Συνεδρίας
- Προσθήκη PC chair
- Προσθήκη PC member
- Αναζήτηση Συνεδρίας
- Προβολή Συνεδρίας
- Διαγραφή Συνεδρίας
- Λειτουργίες για αλλαγή του State της Συνεδρίας

Non-Functional Requirements:

- Ταχύτητα ανταπόκρισης
- Περιορισμός δυνατοτήτων ανά ρόλο
- Μνήματα λάθους
- User-friendly Interface

Κεφάλαιο 2. Οργάνωση

2.1.1 a. Ρόλοι & Δραστηριότητες:

a. Βρισκόμασταν συνεχώς σε επικοινωνία και προχωρούσαμε την εργασία παράλληλα. Γενικά για τον κώδικα ασχολήθηκε κυρίως ο Νίκος και ο Σωτήρης ανέλαβε το conference από τον κώδικα, την αναφορά και τα διαγράμματα. Χωρίς αυτό να σημαίνει ότι ο Νίκος δε βοήθησε στην αναφορά και ο Σωτήρης στον κώδικα απλά είχαμε μοιράσει τα βάρη έτσι όπως κρίναμε εμείς βέλτιστα.

b. Αρχικά την εργασία την ξεκινήσαμε 26/12/23 και δουλεύαμε καθημερινά με 4-5 ώρες την μέρα (μέσω discord) και σήμερα που γράφω την αναφορά 13/1 υπολογίσαμε μέσο όρο 70 ωρών.

Ξεκινήσαμε ανατίθοντας εργασίες στον καθένα και κατανοώντας την εργασία, ένα άλλο σφάλμα που κάναμε ήταν ότι ξεκινήσαμε με τον κώδικα, μόνο με ένα πλάνο στο μυαλό μας για το τι θέλουμε το τελικό αποτέλεσμα να είναι, με το σκεπτικό ότι θα κάνουμε τα διαγράμματα στο τέλος αφού έχουμε ολοκληρώσει τις κλάσεις και την λειτουργικότητα της εφαρμογής μας.

Πρώτο βήμα ήταν η δημιουργία της βάσεις δεδομένων μέσω phpMyAdmin την οποία διαγράψαμε και ξαναδημιουργήσαμε 2 φορές κατά μήκος της εργασίας διότι δεν λειτουργούσε όπως θα θέλαμε. Ξεκινήσαμε την υλοποίηση του κώδικα δημιουργώντας τις βασικές κλάσεις Main, User, Paper, Confernece και Role και μετά προχωρήσαμε με βάση την εκφώνηση, πρώτα το Paper και δημιουργούσαμε κλάσεις και μεθόδους ανάλογα το ζητούμενο κάθε στιγμή.

Κεφάλαιο 3. Απαιτήσεις (πρώτο μέρος του έργου)

3.1 Λειτουργικές Απαιτήσεις:

Αρχιτεκτονική/Directory Map:

```
└─────.vscode
└─────src
|   └─────main
|       └─────java
|           └─────com
|               └─────icsd191080_icsd19235
|                   └─────controller
|                   |           ├─────ConferenceController.java
|                   |           ├─────HomeController.java
|                   |           ├─────PaperController.java
|                   |           ├─────RoleController.java
|                   |           └─────UserController.java
|                   |
|                   └─────model
|                   |           ├─────Conference.java
|                   |           ├─────Paper.java
|                   |           ├─────Role.java
|                   |           └─────User.java
|                   |
|                   └─────repository
|                   |           ├─────ConferenceRepository.java
|                   |           ├─────PaperRepository.java
|                   |           ├─────RoleRepository.java
|                   |           └─────UserRepository.java
|                   |
|                   └─────service
|                   |  |        ├─────ConferenceService.java
|                   |  |        ├─────PaperService.java
|                   |  |        ├─────RoleService.java
|                   |  |        ├─────UserService.java
|                   |  ├─────Main.java
|                   |  └─────SecurityConfig.java
└─────resources
|       └─────templates
|       └─────application.yml
└─────target
└─────pom.xml
└─────README.md
└─────project_makeover.sql
└─────Texnologia Logismikou.postman_collection.json
```

3.1.1 Controller

package com.icsd19080_icsd19235.controller;
Το **controller** είναι υπεύθυνο για τον χειρισμό των αιτήσεων (HTTP requests) από τον χρήστη και την αποστολή απαντήσεων. Οι κλάσεις μέσα σε αυτό το πακέτο, όπως οι ConferenceController.java, PaperController.java, UserController.java και άλλες, είναι υπεύθυνες για τη δρομολόγηση αιτημάτων και την επιστροφή των κατάλληλων δεδομένων (ή views).
Ο **controller** αλληλεπιδρά με τον **service** για να λάβει ή να ενημερώσει δεδομένα στη βάση μέσω του **repository**.

ConferenceController

Request	Method	URL
		*/api/conferences/***
GET	getConference	*/{id}*
GET	getAllConferences	
POST	createConference	
PUT	updateConference	*/{id}*
POST	addPCChairs	*/{id}/add-pc-chairs*
POST	addPCMembers	*/{id}/add-pc-members*
GET	SearchConferences	*/search*
GET	ConferenceView	*/conferences/{id}/view*
DELETE	DeleteConference	*/{id}*
PUT	startSubmission	*/{id}/start-submission*
PUT	startReviewerAssignment	*/{id}/start-reviewer-assignment*
PUT	startReview	*/{id}/start-review*
PUT	startDecisionMaking	*/{id}/start-decision-making*
PUT	startFinalSubmission	*/{id}/start-final-submission*
PUT	endConference	*/{id}/end-conference*

HomeController

Request	Method	URL
GET	showLoginPage	*/login*
POST	Login	*/login*
GET	welcomePage	*/welcome*

PaperController

Request	Method	URL
		/api/papers/*
GET	getAllPapers	
GET	getPaperById	*/{id}*
POST	createPaper	*/{conferenceId}*
PUT	updatePaper	*/{id}*
POST	addCoAuthors	*/{paperId}/add-coauthors*
PUT	submitPaper	*/{paperId}/submit*
PUT	assignReviewer	*/{paperId}/assign-reviewers*
PUT	submitReview	*/{paperId}/submit-review*
PUT	approvePaper	*/{paperId}/approve*
PUT	rejectPaper	*/{paperId}/reject*
PUT	submitFinalPaper	*/{paperId}/final-submit*
PUT	acceptPaper	*/{paperId}/accept*
PUT	searchPapers	*/search*
GET	viewPaper	*/{paperId}/view*
DELETE	withdrawPaper	*/{paperId}/withdraw*

RoleController

Request	Method	URL
		/api/roles/*
GET	getAllRoles	
GET	getRoleById	*/{id}*
POST	createRole	

UserController

Request	Method	URL
		/api/users/*
GET	getAllUsers	
POST	login	*/login*
POST	createUser	

3.1.2 MODEL

package com.icsd19080_icsd19235.model;

Οι κλάσεις στο **model** πακέτο ορίζουν τα αντικείμενα που αντιπροσωπεύουν τα δεδομένα του συστήματος. Κάθε κλάση σε αυτό το πακέτο συνήθως αντιπροσωπεύει έναν πίνακα στη βάση δεδομένων. Τα αντικείμενα αυτά χρησιμοποιούνται για να ενώσουν τα δεδομένα της βάσης με την εφαρμογή μας.

Conference

Το conference αποτελείται από:

conference_id	***creation_date***	***name***	***description***	***state***

Όπου ***state = [CREATED, SUBMISSION, ASSIGNMENT, REVIEW, DECISION, FINAL_SUBMISION, FINAL].***

Επίσης περιέχει τις εξής σχέσεις:

	JoinTable	**Relation**
Many-to-Many	conference_pc_chair	Conference-User
Many-to-Many	conference_pc_members	Conference-User
Many-to-Many	conference_papers	Conference-Papers

Paper

Το paper αποτελείται από:

paper_id	***creation_date***	***title***	***abstract_text***	***content***	***authron_names***	***user_id***	**…**

…	**reviewer 1_id**	**reviewer 2_id**	**reviewer1_com ments**	**reviewer2_com ments**	**reviewer1_ score**	**reviewer2_ score**	**…**

…	**state**	**conference_id**	**submitted**	**needs_modification**

Επίσης περιέχει τις εξής σχέσεις:

	JoinTable	**Relation**
Many-to-Many	Paper_coauthors	Paper-User

Role

Το Role αποτελείται από:

role_id	***user_id***	***role_name***

Επίσης περιέχει τις εξής σχέσεις:

	JoinColumn	**Relation**
Many-to-One	user_id	User-Role

User

Ο χρήστης αποτελείται από:

user_id	***username***	***password***	***fullname***

Επίσης περιέχει τις εξής σχέσεις:

	MappedBy	**Relation**
One-To-Many	user	User-Role
Many-to-Many	coAuthors	User-Paper
Many-to-Many	pcChair	User-Conference
Many-to-Many	pcMembers	User-Conference

Επίσης ο χρήστης κάνει Implement το Interface UserDetails και έχει τις παρακάτω ενέργειες/μεθόδους.

getAuthorities
isAccountNonExpired
isAccountNonLocked
isCredentialsNonExpired
IsEnabled

3.1.3 Repository

package com.icsd19080_icsd19235.repository;

Το πακέτο **repository** περιέχει τις διεπαφές (interfaces) που διαχειρίζονται την επικοινωνία με τη βάση δεδομένων. Αυτές οι διεπαφές επεκτείνουν το **JpaRepository** (ή άλλο repository interface), το οποίο παρέχει βασικές λειτουργίες CRUD (Create, Read, Update, Delete) χωρίς να χρειάζεται να γράψετε SQL.

1. **ConferenceRepository**
   - **findByNameContainingIgnoreCaseAndDescriptionContainingIgnoreCase(String name, String description)** • Επιστρέφει μια λίστα από συνδιασκέψεις που περιέχουν την καθορισμένη ονομασία και περιγραφή, αγνοώντας τη διαφορά κεφαλαίων.

2. **PaperRepository**
   - **findByTitleContainingAndAuthorNamesContainingAndAbstractTextContainingOrderByTitle(String title, String authorNames, String abstractText)** ¨
     Επιστρέφει μια λίστα από papers που περιέχουν τον καθορισμένο τίτλο, τα ονόματα συγγραφέων και το κείμενο της περίληψης, ταξινομημένα κατά τίτλο.
   - **findByTitleContainingAndAuthorNamesContainingOrderByTitle(String title, String authorNames)**
     Επιστρέφει μια λίστα από papers που περιέχουν τον καθορισμένο τίτλο και τα ονόματα συγγραφέων, ταξινομημένα κατά τίτλο.
   - **findByAuthorNamesContainingAndAbstractTextContainingOrderByTitle(String authorNames, String abstractText)**
     Επιστρέφει papers που περιέχουν τα καθορισμένα ονόματα συγγραφέων και το κείμενο της περίληψης, ταξινομημένα κατά τίτλο.

- **findByTitleContainingAndAbstractTextContainingOrderByTitle(String title, String abstractText)**
  Επιστρέφει papers που περιέχουν τον καθορισμένο τίτλο και το κείμενο της περίληψης, ταξινομημένα κατά τίτλο.

- **findByTitleContainingOrderByTitle(String title)**
  Επιστρέφει papers που περιέχουν τον καθορισμένο τίτλο, ταξινομημένα κατά τίτλο.
- **findByAuthorNamesContainingOrderByTitle(String authorNames)**
  Επιστρέφει papers που περιέχουν τα καθορισμένα ονόματα συγγραφέων, ταξινομημένα κατά τίτλο.

- **findByAbstractTextContainingOrderByTitle(String abstractText)**
  Επιστρέφει papers που περιέχουν το καθορισμένο κείμενο της περίληψης, ταξινομημένα κατά τίτλο.

- **findAllByOrderByTitle()**
  Επιστρέφει όλα τα papers, ταξινομημένα κατά τίτλο.

- **findByState(PaperState state)**
  Επιστρέφει papers με βάση την καθορισμένη κατάσταση.

- **findByConferenceAndState(Conference conference, PaperState state)** • Επιστρέφει papers που σχετίζονται με την καθορισμένη συνδιάσκεψη και κατάσταση.

3. RoleRepository

- **findByRoleName(Role.RoleName roleName)** • Επιστρέφει έναν ρόλο με βάση το όνομα του ρόλου.

4. UserRepository

- **findByFullName(String fullName)**
  Επιστρέφει έναν χρήστη με βάση το πλήρες όνομα.

- **findByUsername(String username)**
  Επιστρέφει έναν χρήστη με βάση το όνομα χρήστη, τυλιγμένο σε Optional.

- **findByUsernameAndPassword(String username, String password)**
  Επιστρέφει έναν χρήστη με βάση το όνομα χρήστη και τον κωδικό πρόσβασης.

3.1.4 Service

package com.icsd19080_icsd19235.service;

Στο πακέτο **service**, οι κλάσεις αυτές περιέχουν τη "λογική" της εφαρμογής. Οι υπηρεσίες αλληλεπιδρούν με τα repositories και παρέχουν τις απαραίτητες λειτουργίες στους ελεγκτές. Κάθε υπηρεσία διαχειρίζεται συγκεκριμένες λειτουργίες για ένα οντολογικό μοντέλο, όπως User, Conference, κ.λπ.

ConferenceService

Method	Explanation	Parameters
getConference	*Returns an optional conference by its ID.*	*Long id*
getAllConferences	*Returns a list of all conferences.*	

createConference	*Creates and saves a new conference, ensuring that associated PC chairs, PC members, and papers are valid and properly linked.*	*Conference conference*
updateConference	*Updates the name, description, PC chairs, and PC members of a specific conference by its ID if there are changes.*	*Long conferenceId, String name, String description, Set<User> pcChairs, Set<User> pcMembers*
addPCChairs	*Adds PC chairs to an existing conference by finding and linking the specified users.*	*Long conferenceId, Set<User> pcChairs*
addPCMembers	*Adds PC members to an existing conference by finding and linking the specified users.*	*Long conferenceId, Set<User> pcMembers*
SearchConferences	*Searches and returns a list of conferences that contain the specified name and description (case-insensitive).*	*String name, String description*
ConferenceView	*Returns a simplified view of a specific conference with only the ID and name for a given user.*	*Long conferenceId, User user*
DeleteConference	*Deletes a conference if it is in the CREATED state.*	*Long conferenceId*
startSubmission	*Changes the conference state from CREATED to SUBMISSION.*	*Long conferenceId*
startReviewerAssignment	*Changes the conference state from SUBMISSION to ASSIGNMENT.*	*Long conferenceId*
startReview	*Changes the conference state from ASSIGNMENT to REVIEW.*	*Long conferenceId*
startDecisionMaking	*Changes the conference state from REVIEW to DECISION.*	*Long conferenceId*
startFinalSubmission	*Changes the conference state from DECISION to FINAL_SUBMISSION.*	*Long conferenceId*
endConference	*Changes the conference state from FINAL_SUBMISSION to FINAL, and updates the state of papers associated with the conference to either ACCEPTED or REJECTED.*	*Long conferenceId*

updateConferenceState	*A helper method that updates the conference state if it is in the expected current state and returns whether the state was successfully updated.*	*Long conferenceId, Conference.ConferenceState expectedState, Conference.ConferenceState newState*

PaperService

Method	Explanation	Parameters
getAllPapers	*Returns a list of all papers that exist in the database.*	
getPaperById	*Returns a paper based on its ID.*	*Long Id*
createPaper	*Creates a new paper and assigns it to a specific conference.*	Paper paper
deletePaper	*Deletes a paper.*	*Long Id*
updatePaper	*Updates the details of a paper (title, abstract, authors, content).*	Long paperId, String title, String abstractContent, String authors, String content
addCoAuthors	*Adds co-authors to a paper.*	Long paperId, Set<User> coAuthors
submitPaper	*Submits a paper as long as the conference is in submission state (SUBMISSION).*	Long paperId
assignReviewer	*Assigns a reviewer to a paper as long as the conference is in assignment state (ASSIGNMENT).*	Long paperId, Long reviewerId
submitReview	*Submits a review of a paper by a reviewer..*	Long paperId, Long reviewerId, Integer reviewerScore, String reviewerComments
approvePaper	*Approves a paper when the conference is in decision state (DECISION).*	Long paperId
rejectPaper	*Rejects a paper when the conference is in decision state (DECISION).*	Long paperId
submitFinalPaper	*Submits the final version of the paper and responds to reviewers' comments when the conference is in final submission state (FINAL_SUBMISSION).*	Long paperId, String finalContent, String addressingReviewerComments
acceptPaper	*Accepts a paper as long as the conference is in the final state (FINAL).*	Long paperId
searchPapers	*Searches for papers based on the title, authors, or abstract.*	String title, String authors, String abstractContent
viewPaperId	*Displays the details of a paper based on its ID*	Long paperId, Principal principal

RoleService

Method	Explanation	Parameters
findAllRoles	*Returns a list of all roles that exist in the database.*	

createRole	*Creates a new role for a user and saves it in the database, checking if the user exists before saving the role.*	*Role Role*
updateRole	*Updates an existing role with new details (role name and user).*	*Long Id*, *Role roleDetails*
deleteRole	*Assigns a specific role to a user based on the user ID and the role provided as a parameter.*	*Long Id*
findRoleById	*Returns a role based on its ID.*	*Long Id*
assignRoleToUser	*Assigns a specific role to a user based on the user ID and the role provided as a parameter.*	*Long userId,* *Long confernceId,* *String roleName*

UserService

Method	Explanation	Parameters
findAllUsers	*Returns a list of all users in the system.*	
findUserById	*Returns an optional user by their ID.*	*Long Id*
createUser	*Creates and saves a new user after validating that the full name is provided.*	*User User*
updateUser	*Updates an existing user if the user ID is found, otherwise throws an error.*	*User User*
getUserById	*Retrieves a user by their ID, or throws an error if the user is not found.*	*Long Id*
deleteUser	*Deletes a user by their ID if they exist, otherwise throws an error.*	*Long Id*
loadUserByUsername	*Loads a user by their username for authentication purposes in Spring Security, or throws an error if the user is not found.*	*String Username*
login	*Authenticates a user by their username and password, returning the user if the credentials match, or null if there is a password mismatch.*	*Long Id*

3.1.4 SecurityConfig

package com.icsd19080_icsd19235;

Η κλάση SecurityConfig είναι υπεύθυνη για τη διαμόρφωση της Spring Security, η οποία ελέγχει τον τρόπο πρόσβασης των χρηστών σε διάφορα μέρη της εφαρμογής. Η κλάση φέρει τις αναφορές @Configuration και @EnableWebSecurity, που υποδηλώνουν ότι αυτή η κλάση περιέχει ρυθμίσεις ασφαλείας.

1. **Απενεργοποίηση της Προστασίας CSRF**: Η διαμόρφωση απενεργοποιεί ρητά την προστασία Cross-Site Request Forgery (CSRF) χρησιμοποιώντας το csrf().disable(). Σε πολλές περιπτώσεις, η προστασία CSRF είναι συνιστώμενη για web εφαρμογές, αλλά έχει απενεργοποιηθεί εδώ για να απλοποιηθεί η εφαρμογή και χρήση εργαλείων όπως το Postman.
2. **Επιτρεπόμενη Πρόσβαση σε Συγκεκριμένα Endpoints**: Η μέθοδος authorizeRequests() καθορίζει ποια endpoints είναι δημόσια προσβάσιμα και ποια απαιτούν αυθεντικοποίηση:
    - Η γραμμή antMatchers("/api/**").permitAll() επιτρέπει την ανοικτή πρόσβαση σε κάθε URL που ξεκινά με /api/, δηλαδή οι διαδρομές αυτές μπορούν να προσπελαστούν χωρίς να απαιτείται σύνδεση.
    - Η γραμμή anyRequest().permitAll() επιτρέπει σε όλες τις άλλες διαδρομές στην εφαρμογή να προσπελαστούν χωρίς αυθεντικοποίηση. Σε πιο ασφαλείς ρυθμίσεις, αυτό συνήθως περιορίζεται σε αυθεντικοποιημένους χρήστες.
3. **Προσαρμοσμένη Σελίδα Σύνδεσης**: Αυτή η διαμόρφωση ρυθμίζει μια προσαρμοσμένη σελίδα σύνδεσης που βρίσκεται στη διαδρομή /login χρησιμοποιώντας τη μέθοδο formLogin().loginPage("/login"). Η εντολή permitAll() διασφαλίζει ότι όλοι μπορούν να προσπελάσουν τη σελίδα σύνδεσης χωρίς να απαιτείται αυθεντικοποίηση.
4. **Διαχείριση Αποσύνδεσης**: Η μέθοδος logout() ρυθμίζει τον τρόπο με τον οποίο οι χρήστες μπορούν να αποσυνδεθούν..
5. **Bean για UserDetailsService**: Η μέθοδος userDetailsService() παρέχει στο πλαίσιο ασφαλείας πρόσβαση σε λεπτομέρειες χρήστη. Αυτό επιτυγχάνεται αναθέτοντας την ευθύνη στο UserService, το οποίο υλοποιεί το UserDetailsService. Αυτή η υπηρεσία χρησιμοποιείται για την ανάκτηση πληροφοριών χρηστών (όπως όνομα χρήστη και κωδικός πρόσβασης) για σκοπούς αυθεντικοποίησης.

3.1.5Resources

Ο φάκελος **resources** περιλαμβάνει αρχεία που χρησιμοποιούνται από την εφαρμογή, όπως αρχεία HTML, αρχεία ρυθμίσεων και SQL dump αρχεία. Είναι κοινό μέρος για το **frontend** της εφαρμογής και άλλες παραμετροποιήσεις

Application.yml

Το αρχείο **application.yml** είναι ένα αρχείο ρυθμίσεων για το **Spring Boot**, όπου καθορίζονται βασικές παράμετροι για τη λειτουργία της εφαρμογής, όπως η σύνδεση με τη βάση δεδομένων, η ρύθμιση του Hibernate, η ασφάλεια, το server port και άλλες γενικές ρυθμίσεις.

Section	Property	Value	Explanation
Spring.datasource	url	jdbc:mysql://localhost:3306/project_makeover	Connection string to MySQL database located at localhost, with database project_makeover
	username	root	The username for the MySQL database
	password	“”	The password for the MySQL database (empty in this case)

spring.jpa	Show-sql	true	Enables the display of SQL statements generated by Hibernate.
	hibernate.ddl-auto	none	Disables Hibernate from managing the database schema (no schema generation or updates).
	hibernate.dialect	org.hibernate.dialect.MySQLDialect	Configures Hibernate to use MySQL-specific dialect for SQL generation.
spring.security	enabled	false	Disables Spring Security
spring.server	Port	8080	The application runs on localhost:8080.
Spring.thymleaf	Cache	false	Disables caching of templates for development purposes.
	prefix	classpath:/templates/	Defines the location where Thymeleaf templates are stored.
	suffix	.html	All templates will have the .html file extension.
	mode	HTML	Thymeleaf is configured to operate in HTML mode.
	Encoding	UTF-8	Character encoding for templates is set to UTF-8.
	Content-type	text/html	Content type for Thymeleaf templates is set to text/html.
Spring.main	Allow-circular-references	true	Allows circular references between beans in the application.
Logging.level	org.hibernate.SQL	DEBUG	Enables detailed logging of SQL statements generated by Hibernate.
	org.hibernate.type.descriptor.sql.BasicBinder	TRACE	Enables trace-level logging for binding SQL parameters, useful for detailed debugging.

pom.xml

Πώς όλα αυτά συνδέονται:
Το **pom.xml** παρέχει ένα συνολικό πλαίσιο για τη διαχείριση των εξαρτήσεων, της διαδικασίας build και των ρυθμίσεων της εφαρμογής.
Οι εξαρτήσεις καθορίζουν ποιες βιβλιοθήκες και frameworks θα χρησιμοποιηθούν στην εφαρμογή, όπως το Spring Boot, το JPA, το Thymeleaf και ο JDBC οδηγός για τη MySQL.
Τα plugins, κυρίως το **spring-boot-maven-plugin**, φροντίζουν για τη δημιουργία ενός εκτελέσιμου JAR αρχείου που περιλαμβάνει την εφαρμογή σας και όλες τις εξαρτήσεις της.
Οι ρυθμίσεις build και οι ιδιότητες επιτρέπουν να προσαρμόσουμε πτυχές της διαδικασίας κατασκευής και ανάπτυξης της εφαρμογής, όπως το όνομα του παραγόμενου αρχείου και η έκδοση της Java που θα χρησιμοποιηθεί.
Όλα αυτά τα στοιχεία μαζί βοηθούν στο να οργανώσετε σωστά το project σας και να διασφαλίσετε ότι μπορεί να "χτιστεί", να εκτελεστεί και να συντηρηθεί σωστά.

1. Project Metadata

- **modelVersion**: Η έκδοση του μοντέλου του Maven (4.0.0).
- **groupId**: Μοναδικό αναγνωριστικό της οργάνωσης ή του project. Συνήθως είναι το domain ή το όνομα της οργάνωσης, εδώ **com.example**.
- **artifactId**: Το όνομα του project, εδώ είναι **project-makeover**.
- **version**: Η έκδοση του project, εδώ **1.0-SNAPSHOT**. Το **SNAPSHOT** υποδεικνύει ότι το project είναι υπό ανάπτυξη.

2. Dependencies (Εξαρτήσεις)

Το πιο σημαντικό μέρος του **`pom.xml`** αφορά τις εξαρτήσεις του project, δηλαδή τις βιβλιοθήκες και τα frameworks που απαιτούνται για την εκτέλεση του project σας.

- **spring-boot-starter-web**:
  - Παρέχει όλες τις απαραίτητες βιβλιοθήκες για την ανάπτυξη ενός web project με το **Spring Boot**. Περιλαμβάνει υποστήριξη για REST APIs, τη διαχείριση HTTP αιτήσεων και απαντήσεων, καθώς και το ενσωματωμένο Tomcat web server.
- **spring-boot-starter-data-jpa**:
  - Παρέχει υποστήριξη για την εργασία με βάσεις δεδομένων χρησιμοποιώντας **JPA (Java Persistence API)** και το **Hibernate**. Αυτή η εξάρτηση σας επιτρέπει να αλληλεπιδράτε εύκολα με τη βάση δεδομένων MySQL μέσω αντικειμενοστραφούς προγραμματισμού.
- **mysql-connector-java**:
  - Ο οδηγός (JDBC driver) για τη MySQL. Επιτρέπει στη Java εφαρμογή σας να συνδεθεί με τη MySQL βάση δεδομένων.
- **spring-boot-starter-thymeleaf**:
  - Παρέχει την ενσωμάτωση του **Thymeleaf** template engine, ο οποίος χρησιμοποιείται για την απόδοση (rendering) των HTML views.
- **spring-boot-starter-security**:
  - Παρέχει βασική ασφάλεια για την εφαρμογή, όπως authentication και authorization μέσω του **Spring Security**. Παρόλο που στο application.yml είναι απενεργοποιημένο, αυτή η εξάρτηση είναι ακόμα παρούσα, αν χρειαστεί να ενεργοποιηθεί σε κάποια φάση.

3. Plugins (Πρόσθετα Maven)

Το Maven χρησιμοποιεί plugins για να ορίζει πώς θα γίνεται η εκτέλεση (run), η μεταγλώττιση (compile) και η δημιουργία του build. Τα plugins συνήθως ορίζουν πώς ακριβώς θα "χτιστεί" η εφαρμογή και αν θα υπάρχουν επιπλέον διαδικασίες, όπως δοκιμές ή packaging.

spring-boot-maven-plugin:

- Αυτό το plugin επιτρέπει τη δημιουργία ενός εκτελέσιμου JAR αρχείου για τη Spring Boot εφαρμογή σας. Σας επιτρέπει να "χτίσετε" και να εκτελέσετε την εφαρμογή χρησιμοποιώντας Maven εντολές, όπως mvn spring-boot:run για να εκκινήσετε την εφαρμογή.

4. Repositories (Αποθετήρια Maven)

Τα αποθετήρια είναι οι τοποθεσίες από τις οποίες το Maven κατεβάζει τις εξαρτήσεις που ορίζονται στο **pom.xml.**

central:

- Το Maven Central Repository είναι το προεπιλεγμένο δημόσιο αποθετήριο για Maven εξαρτήσεις. Εδώ αποθηκεύονται όλες οι βιβλιοθήκες που χρειάζεστε για την εφαρμογή σας και το Maven θα τις κατεβάζει από αυτή τη διεύθυνση

5. Build Configuration (Ρυθμίσεις Build)

finalName:

- Καθορίζει το όνομα του τελικού παραγόμενου αρχείου. Εδώ θα ονομαστεί project-makeover.jar.

plugins:

- Όπως αναφέρθηκε παραπάνω, τα plugins καθορίζουν τη διαδικασία του build και το τι ακριβώς πρέπει να γίνει. Σε αυτή την περίπτωση, το spring-boot-maven-plugin φροντίζει να δημιουργηθεί ένα εκτελέσιμο αρχείο JAR που περιέχει την εφαρμογή σας, μαζί με όλες τις εξαρτήσεις.

3.2 Μη - Λειτουργικές Απαιτήσεις:

Ασφάλεια

Θα πρέπει να εφαρμοστούν μηχανισμοί αυθεντικοποίησης και εξουσιοδότησης χρηστών.

Ο πρόσβαση σε ευαίσθητες πληροφορίες (π.χ., αξιολογήσεις εργασιών) θα πρέπει ναπεριορίζεται βάσει των ρόλων των χρηστών.

Απόδοση:

Το σύστημα θα πρέπει να διαχειρίζεται αποτελεσματικά ταυτόχρονες ενέργειες χρηστών.

Ο χρόνος απόκρισης για συνήθεις λειτουργίες (π.χ., υποβολή εργασίας, ανάθεση αξιολόγησης) θα πρέπει να είναι εντός αποδεκτών ορίων.

Αξιοπιστία:

Το σύστημα θα πρέπει να είναι διαθέσιμο για χρήση 24/7 με ελάχιστο χρόνο αδράνειας για συντηρητικά έργα.

Θα πρέπει να διατηρείται η ακεραιότητα των δεδομένων, και θα πρέπει να πραγματοποιούνται τακτικά αντίγραφα ασφαλείας.

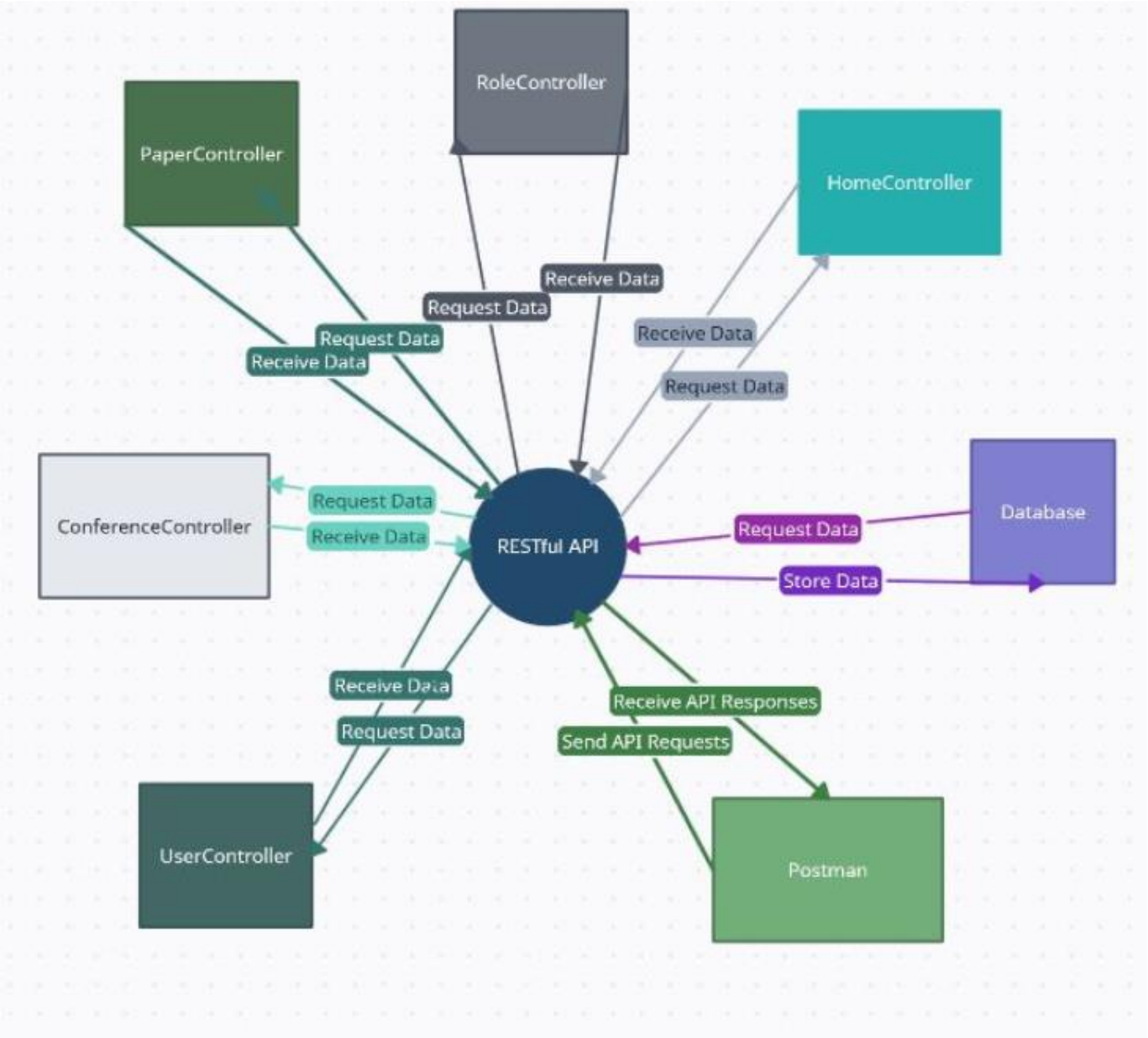
Χρηστικότητα:

Το interface θα πρέπει να είναι ευανάγνωστο και φιλικό προς τον χρήστη.

Θα πρέπει να παρέχονται τεκμηρίωση βοήθειας για να υποστηρίξει τους χρήστες στηνκατανόηση των λειτουργιών του συστήματος.

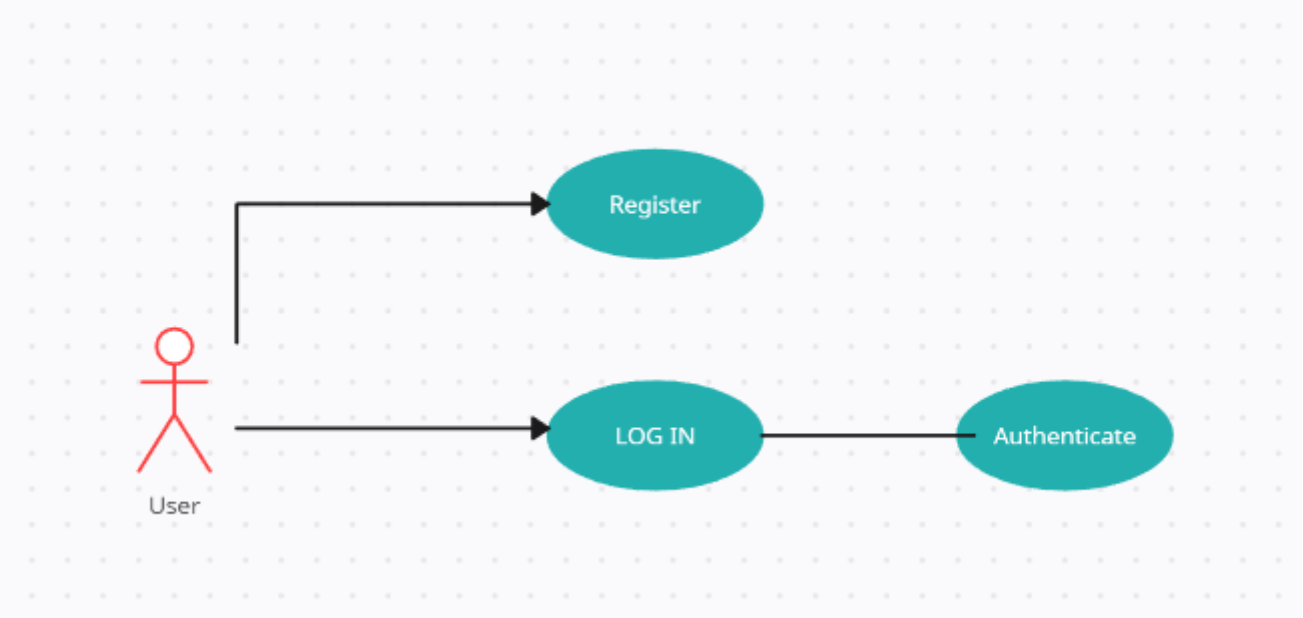
Κεφάλαιο 4. Σχεδίαση (δεύτερο μέρος του έργου)

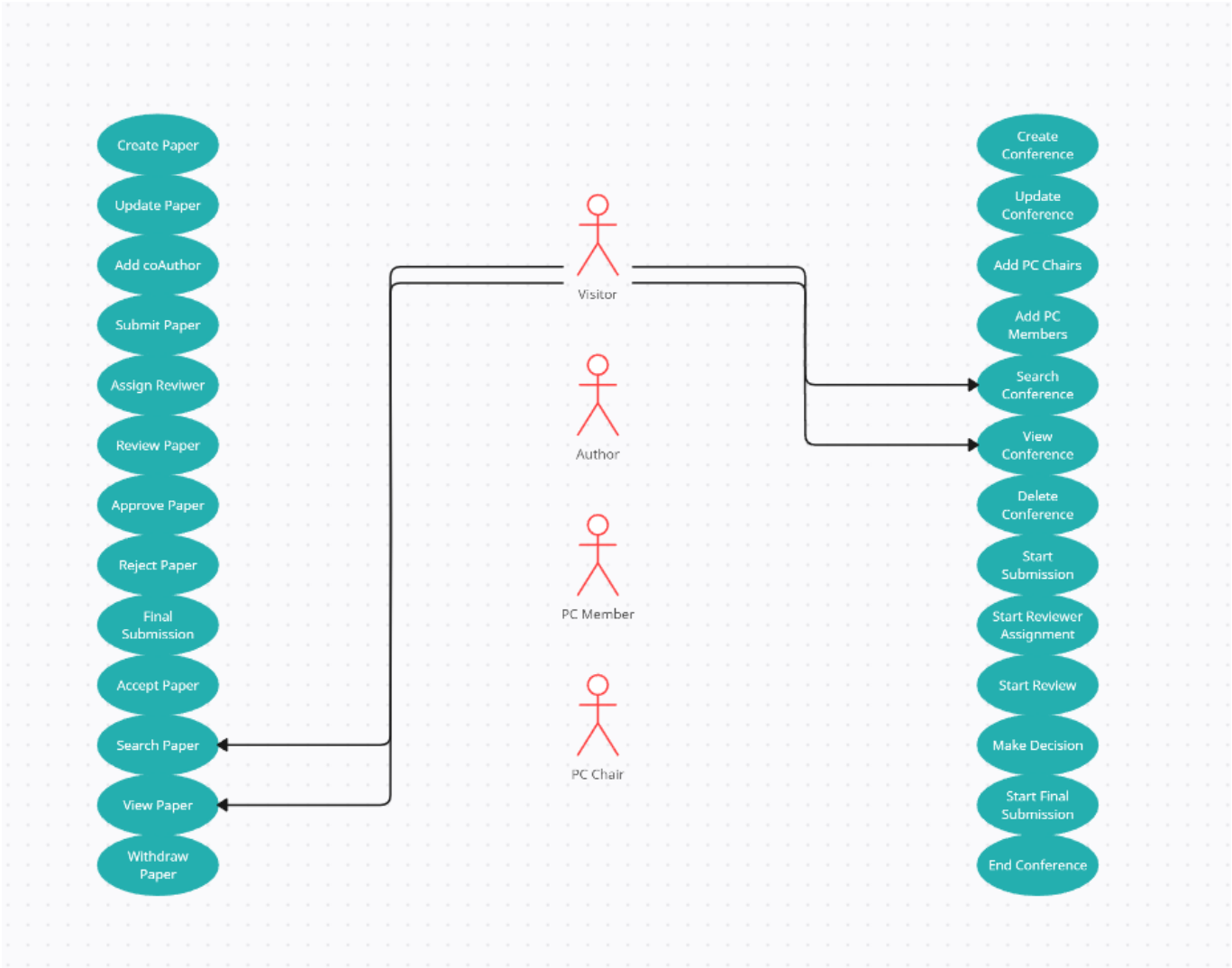
- Context Diagram

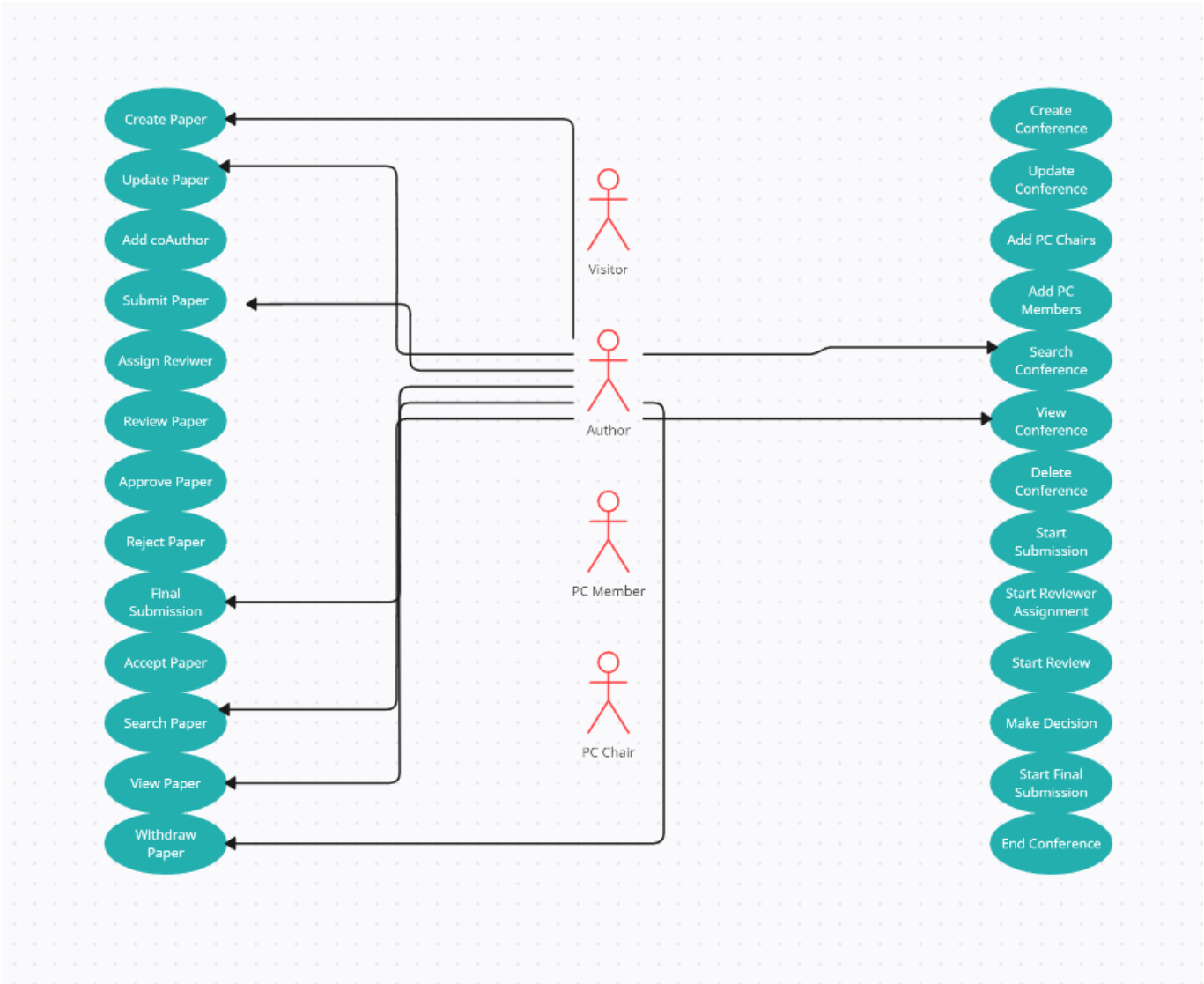


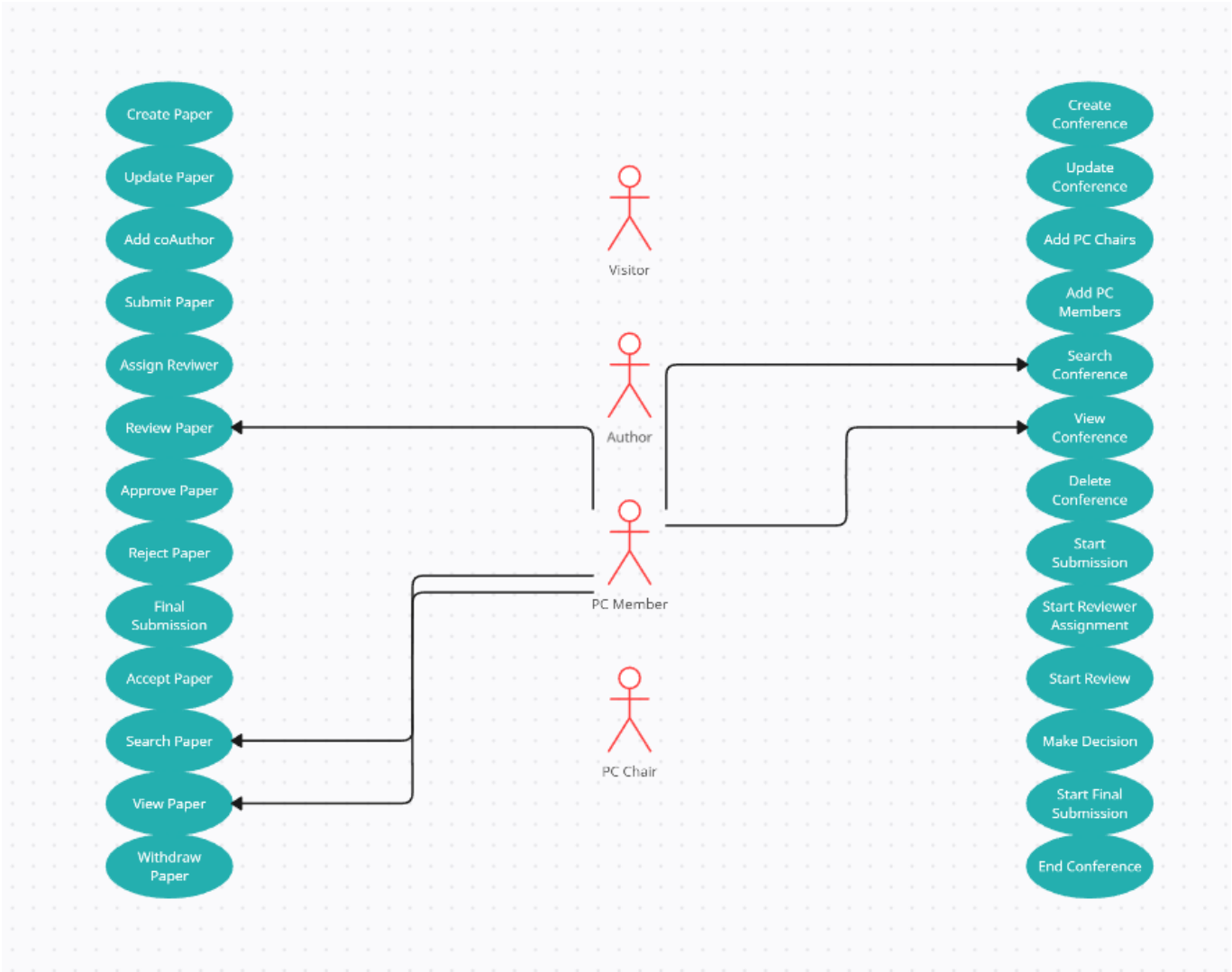
- Use-case Diagrams

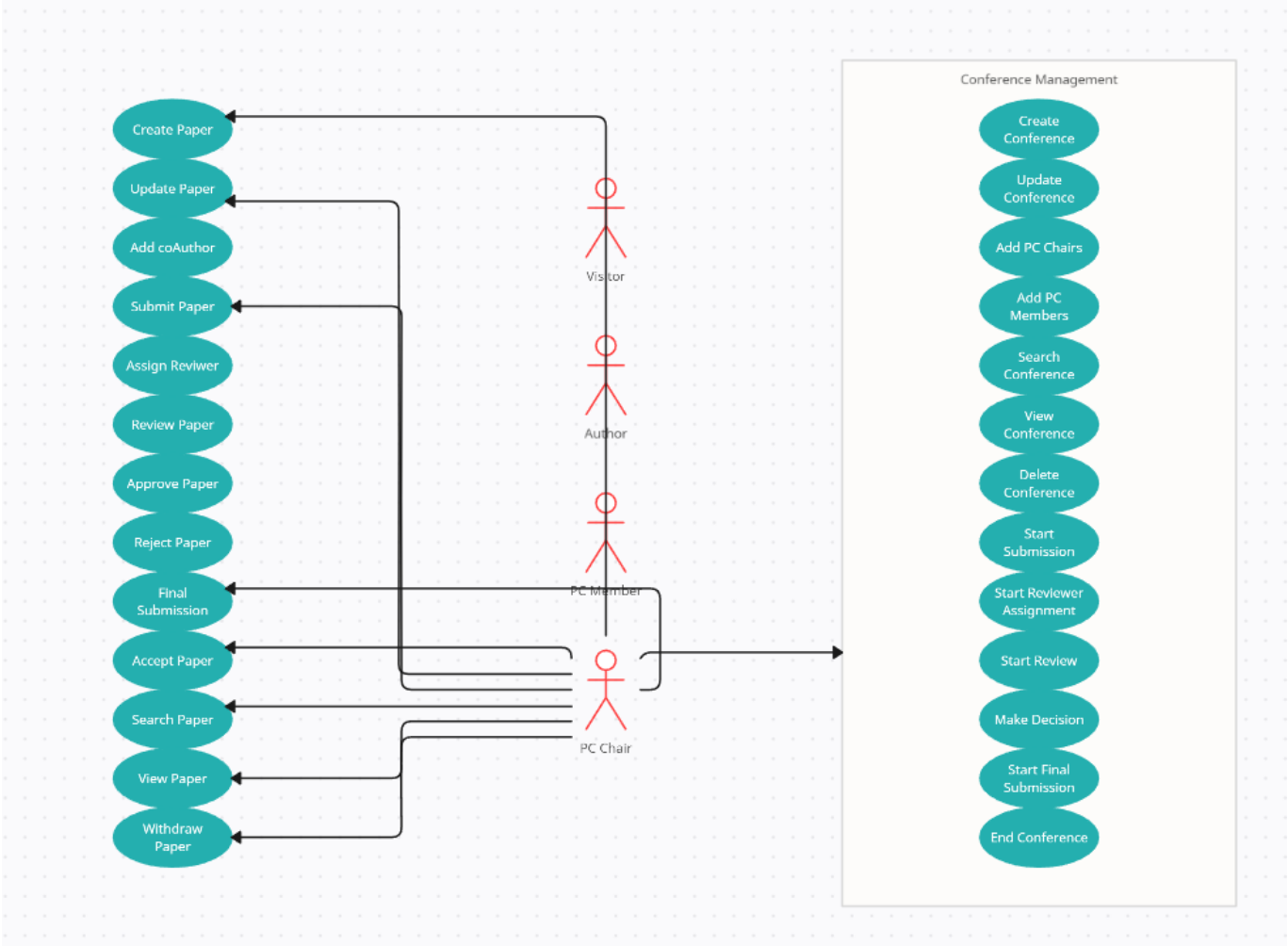
Visitor can perform: View/Search Conferences and Papers.
Author can perform all Visitor actions plus: Create, Update, Submit, Final Submit, and Withdraw Papers.
PC Chair can perform all Author actions plus: Review, Approve, Reject Papers, and manage Conferences.
PC Member can perform all Visitor actions plus: Review Papers.



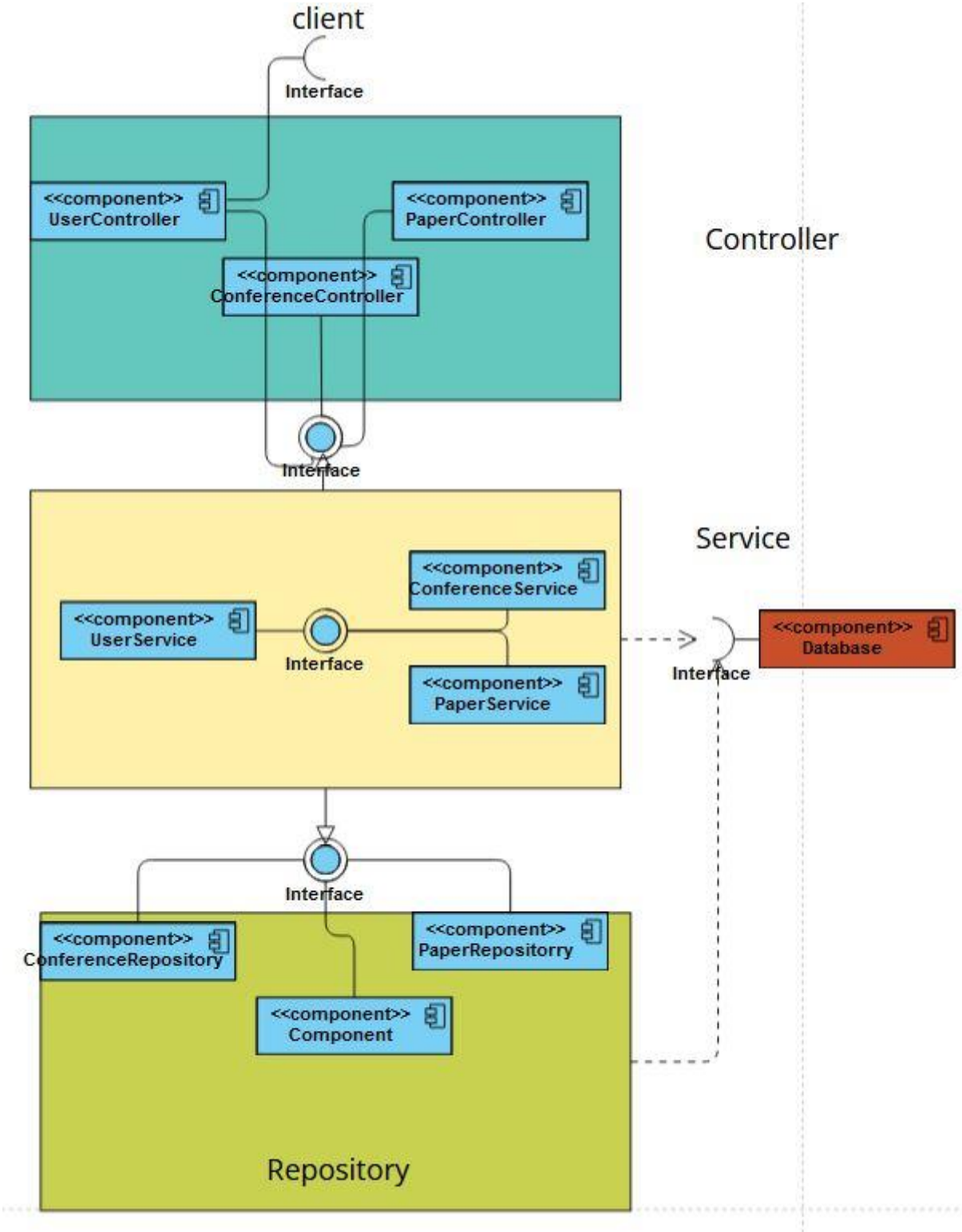

Create Paper
Update Paper
Add coAuthor
Submit Paper
Assign Reviwer
Review Paper
Approve Paper
Reject Paper
Final Submission
Accept Paper
Search Paper
View Paper
Withdraw Paper
Visitor
Author
PC Member
PC Chair
Create Conference
Update Conference
Add PC Chairs
Add PC Members
Search Conference
View Conference
Delete Conference
Start Submission
Start Reviewer Assignment
Start Review
Make Decision
Start Final Submission
End Conference


Create Paper
Update Paper
Add coAuthor
Submit Paper
Assign Reviwer
Review Paper
Approve Paper
Reject Paper
Final Submission
Accept Paper
Search Paper
View Paper
Withdraw Paper
Visitor
Author
PC Member
PC Chair
Create Conference
Update Conference
Add PC Chairs
Add PC Members
Search Conference
View Conference
Delete Conference
Start Submission
Start Reviewer Assignment
Start Review
Make Decision
Start Final Submission
End Conference


Create Paper
Update Paper
Add coAuthor
Submit Paper
Assign Reviwer
Review Paper
Approve Paper
Reject Paper
Final Submission
Accept Paper
Search Paper
View Paper
Withdraw Paper
Visitor
Author
PC Member
PC Chair
Create Conference
Update Conference
Add PC Chairs
Add PC Members
Search Conference
View Conference
Delete Conference
Start Submission
Start Reviewer Assignment
Start Review
Make Decision
Start Final Submission
End Conference

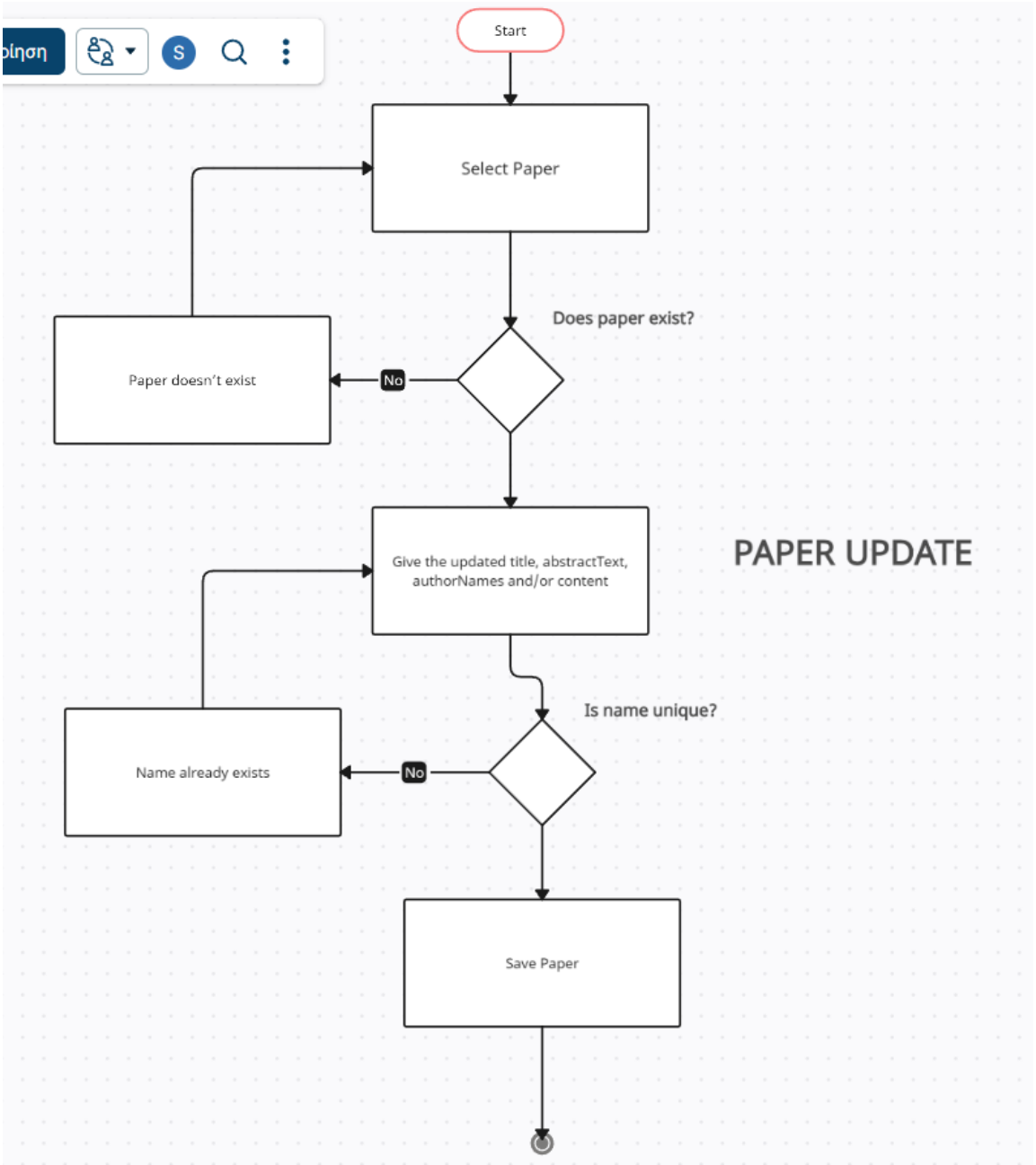

Conference Management
Create Paper
Update Paper
Add coAuthor
Submit Paper
Assign Reviwer
Review Paper
Approve Paper
Reject Paper
Final Submission
Accept Paper
Search Paper
View Paper
Withdraw Paper
Visitor
Author
PC Member
PC Chair
Create Conference
Update Conference
Add PC Chairs
Add PC Members
Search Conference
View Conference
Delete Conference
Start Submission
Start Reviewer Assignment
Start Review
Make Decision
Start Final Submission
End Conference

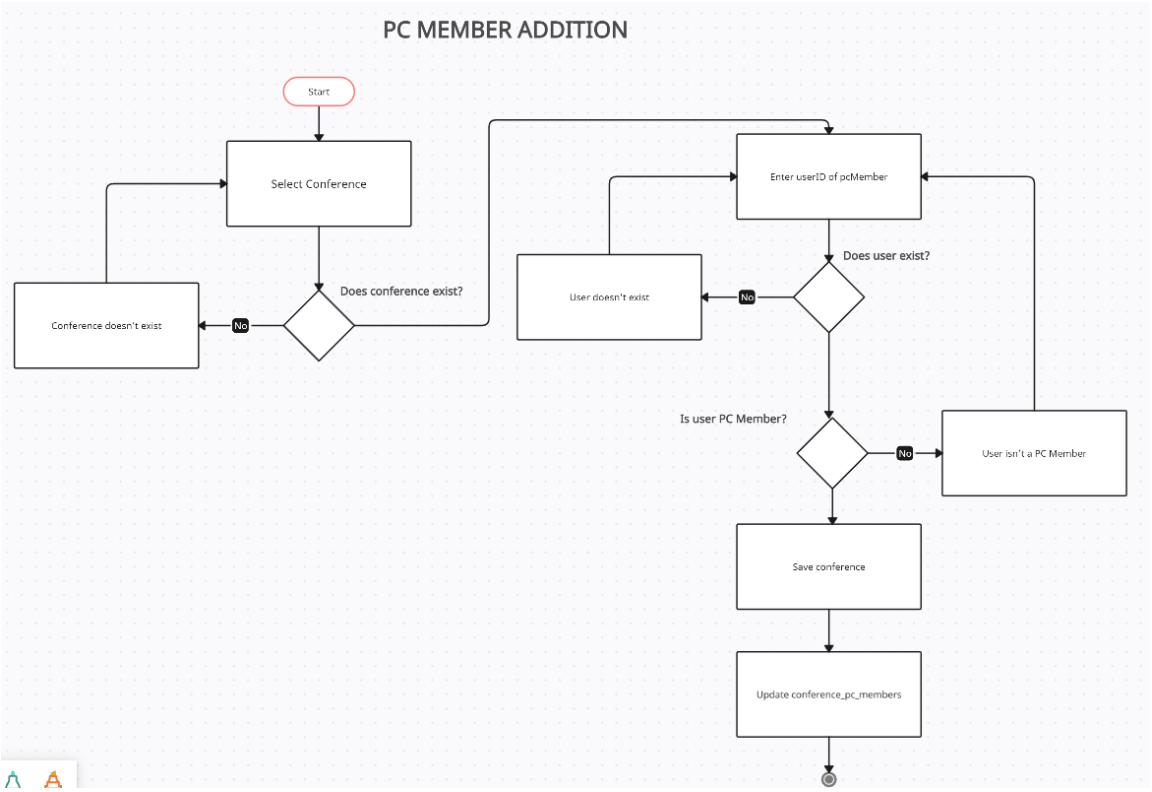
- Component Diagram

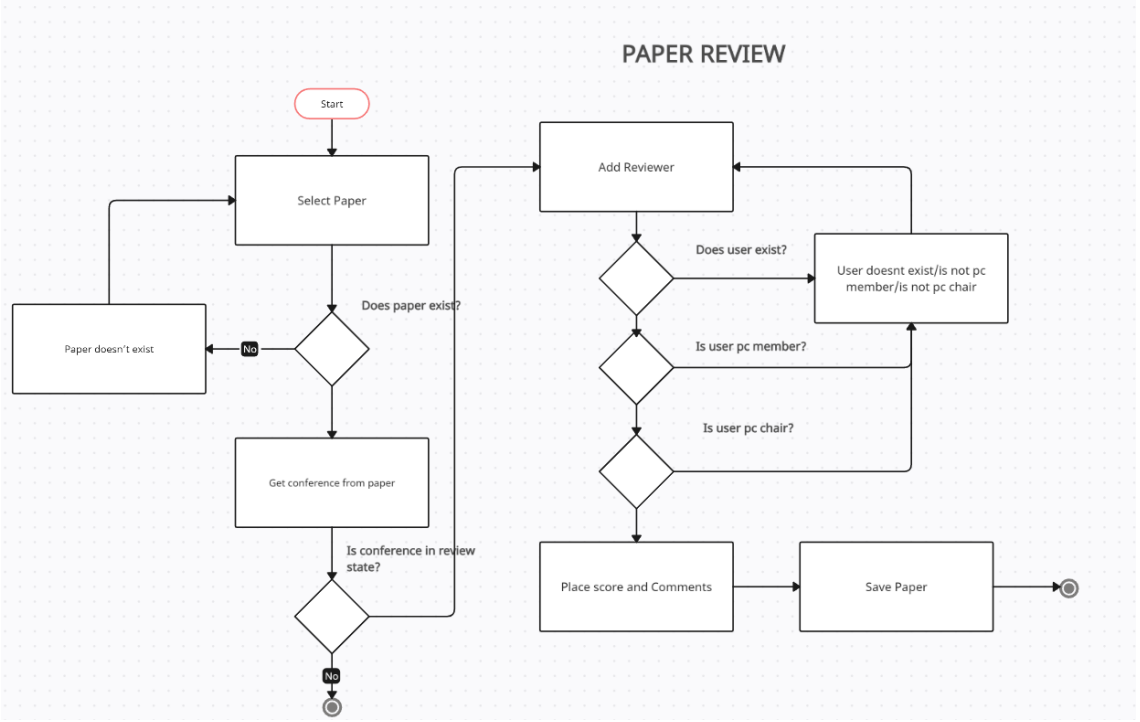


- Activity Diagram

  Σας παραθέτουμε φάκελο με όλα τα διαγράμματα για να μην γεμίζουμε το word (είναι ήδη πολλά). Παραθέτουμε ενδεικτικά μερικά.

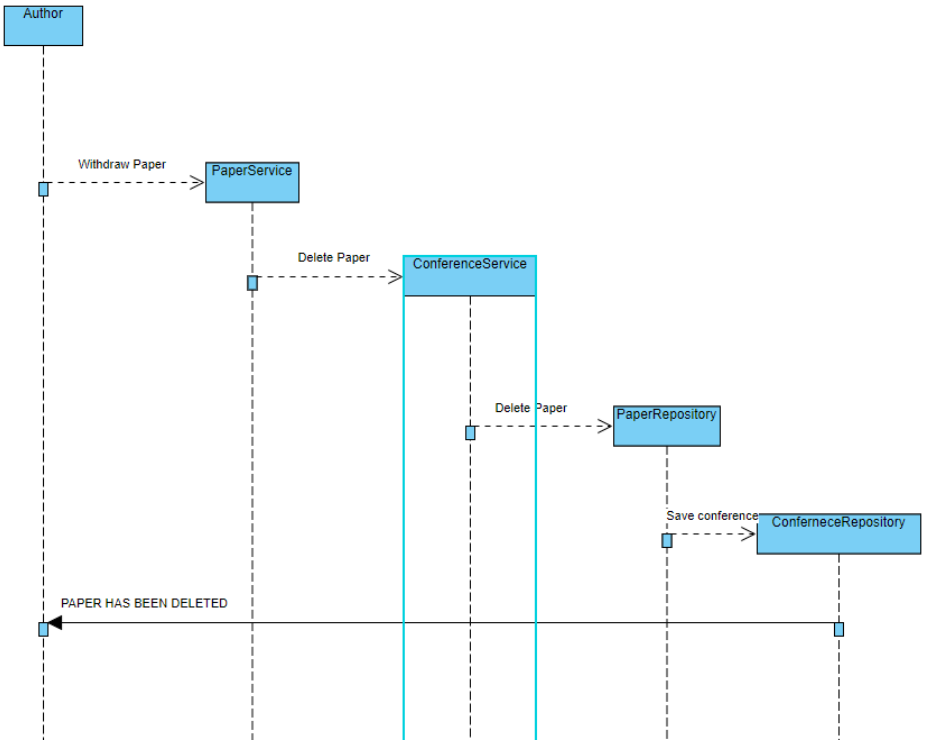


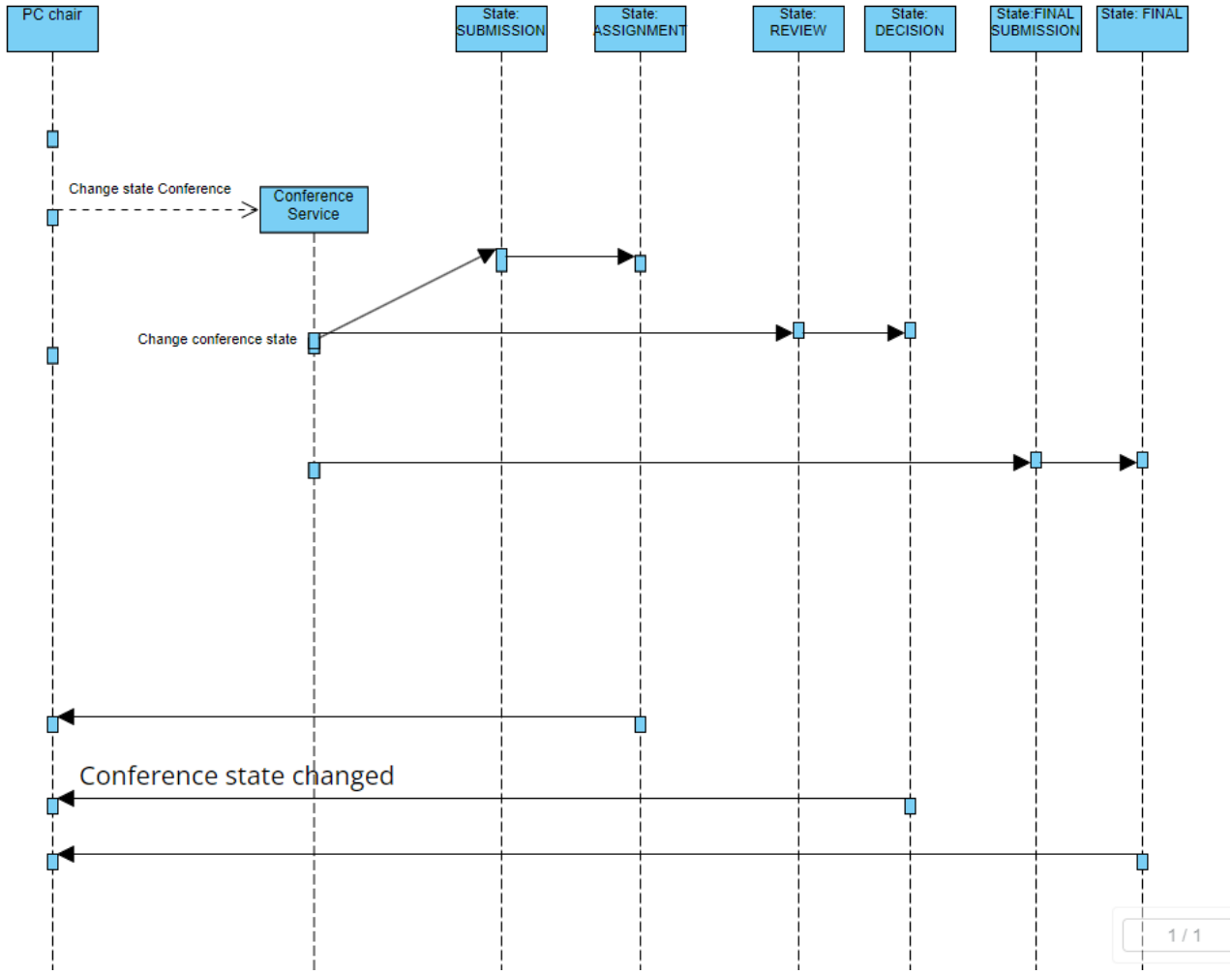




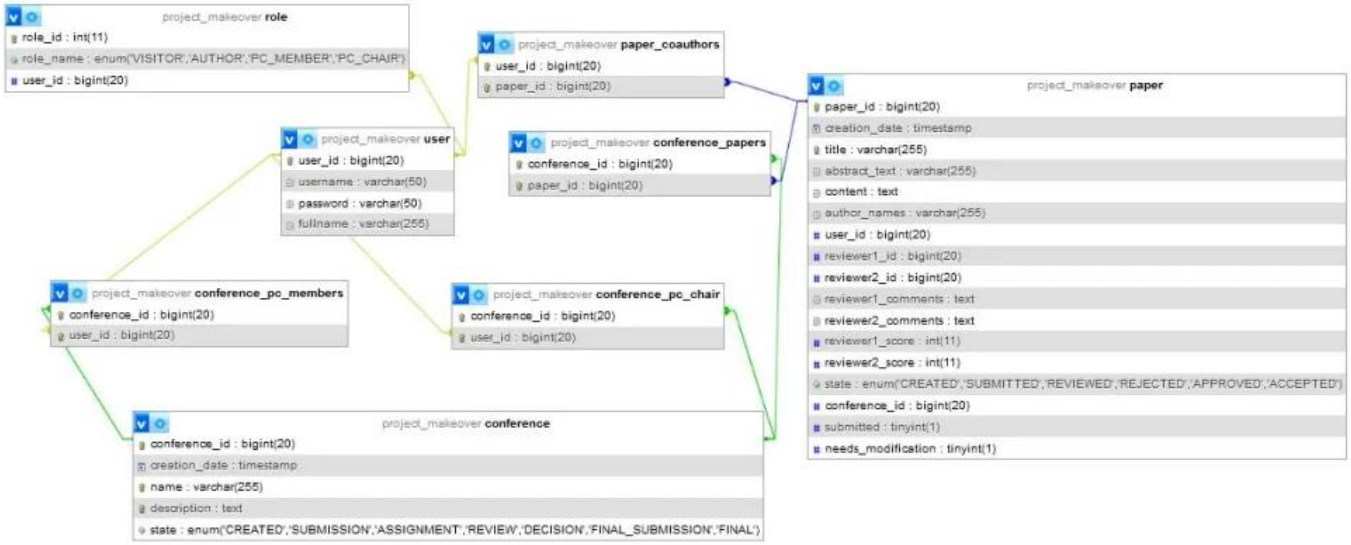
- Sequence Diagram

  Σας παραθέτουμε φάκελο με όλα τα διαγράμματα για να μην γεμίζουμε το word (είναι ήδη πολλά). Παραθέτουμε ενδεικτικά μερικά.


Author
Withdraw Paper
PaperService
Delete Paper
ConferenceService
Delete Paper
PaperRepository
Save conference
ConferneceRepository
PAPER HAS BEEN DELETED


PC chair
State: SUBMISSION
State: ASSIGNMENT
State: REVIEW
State: DECISION
State:FINAL SUBMISSION
State: FINAL
Change state Conference
Conference Service
Change conference state
Conference state changed
1 / 1

- Entity-Relationship Diagram



Κεφάλαιο 5. Υλοποίηση

5.1.1 GitHub

- **URL Αποθετηρίου:** https://github.com/nikoskara90/TexnologiaLogismikou-total_makeover

5.1.2 Τεκμηρίωση Υλοποίησης

Σκοπός της εργασίας ήταν η ανάπτυξη RESTfull API, για την εφαρμογή μας χρησιμοποιήσαμε το Springboot για το backend, MySQL ως βάση δεδομένων με την βοήθεια του Maven και του Git. Ο κώδικας μας όπως αναλύσαμε και στις λειτουργικές απαιτήσεις αφορά τον παρακάτω χάρτη :

Αρχιτεκτονική/Directory Map:

```
└─────.vscode
└─────src
|   └─────main
|       └─────java
|           └─────com
|               └─────icsd191080_icsd19235
|                   └─────controller
|                   |           ├─────ConferenceController.java
|                   |           ├─────HomeController.java
|                   |           ├─────PaperController.java
|                   |           ├─────RoleController.java
|                   |           └─────UserController.java
|                   |
|                   └─────model
|                   |           ├─────Conference.java
|                   |           ├─────Paper.java
|                   |           ├─────Role.java
|                   |           └─────User.java
|                   |
|                   └─────repository
|                   |           ├─────ConferenceRepository.java
|                   |           ├─────PaperRepository.java
|                   |           ├─────RoleRepository.java
|                   |           └─────UserRepository.java
|                   |
|                   └─────service
|                   |           ├─────ConferenceService.java
|                   |           ├─────PaperService.java
|                   |           ├─────RoleService.java
|                   |           └─────UserService.java
|                   ├─────Main.java
|                   └─────SecurityConfig.java
└─────resources
|           └─────templates
|           └─────application.yml
└─────target
```

└─────pom.xml
└─────README.md
└─────project_makeover.sql
└─────Texnologia Logismikou.postman_collection.json

Η εργασία μας υλοποιήθηκε για δεύτερη φορά στηριζόμενοι στο feedback που μας δώσατε και διορθώνοντας τα λάθη που είχαμε τα οποία ήταν πολλά. Την εργασία την ξεκινήσαμε από την αρχή, και την βάση δεδομένων γιατί απλά δεν σωνότανε.
Πρώτο βήμα ήταν χωρίσουμε τα επιμέρους στοιχεία δηλαδή σε Controller, Repository, Service και Model:

DIRECTORY	PACKAGE
DEFAULT	package com.icsd19080_icsd19235;
CONTROLLER	package com.icsd19080_icsd19235.controller;
SERVICE	package com.icsd19080_icsd19235.service;
REPOSITORY	package com.icsd19080_icsd19235.repository;
MODEL	package com.icsd19080_icsd19235.model;

Μετά δημιουργήσαμε τους χρήστες και τους ρόλους και αφετέρου προχωρήσαμε την εργασία όπως στην εκφώνηση, πρώτα το paper και μετά το conference. Το τι περιέχει κάθε κλάση το αναφέραμε στις λειτουργικές απαιτήσεις.

Για να τρέξουμε την εργασία χρειαζόμαστε:

1. Βάση δεδομένων: project_makeover
2. Postman για χρήση
3. Το java project μας (VScode στην περίπτωση μας)
4. Τρέχετε την main,java

5.1.3 Τεκμηρίωση Δοκιμών

Δεν κρατήσαμε σε μορφή screenshot την διαδικασία των τεστ που κάναμε ωστόσο είναι φανερό ότι για ένα τόσο περίπλοκο προτζεκτ χρησιμοποιούσαμε δοκιμές και έξυπνα τεστ ώστε να επιλύνουμε κομμάτι κομμάτι τα προβλήματα που είχαμε.

Κεφάλαιο 6. Σύνοψη και συμπεράσματα

a. Η υλοποίηση ενός τέτοιου συστήματος είναι πολύ χρονοβόρα και δύσκολη αλλά είμαστε ευχαριστημένοι με το αποτέλεσμα που φέραμε εις πέρας παρόλου που είναι λειψό.