Conference Management System

Team name: *Dinastia Tudorilor*

Team Leader: *Apostu Sergiu*

Team Members:

-Frontend:

- Tudor Alexa
- Andreas Alexa
- Sergiu Apostu

-Backend:

- Tudor Ardelean
- Robert Amatiesei
- Adrian Ancau
- Tudor Badiu

Team Coordinator: Blaga Dalia

Contents

Introduction

Short Summary

The Conference Management System supports the automatic management of information related to scientific conferences.

The main objective of this project is to provide the solution for organizations or users that need organizing conferences, automatizes the process of initialising one, inviting people into the conference, setting a theme and submitting/reviewing papers.

As for some project conventions, we mostly mention SC(Steering Committee), PC(Program Committee), CMS(Conference Management Systems).

The app is designed as a prototype of a CMS and it sees to the uses of individuals such as Organizations / People that want to initiate a conference, Authors/Writers, Reviewers, Attendees and so on.

The objective of a Conference Management System is to support the automatic management of information related to scientific conferences.This information concerns: the authors submitting proposals, the members of the Program Committee, the submissions' abstract and full papers proposed, meta-information about these, the deadlines for different phases of sending proposals, assigning paper to reviewers, evaluation deadline and announcing the results of paper valuation. Above all, we hope to provide a comfortable user experience.

References

https://www.youtube.com/watch?time_continue=783&v=zid-MVo7M-E&feature=emb_title

https://www.infoq.com/articles/why-architectural-diagrams/

https://openclassrooms.com/en/courses/4191736-design-a-database-with-uml/4191743-learn about-class-diagrams

Stages of implementing the application

Software Engineering Development Activities

Development activities deal with the complexity by constructing and validating models of the application domain or the system. Development activities such as:

Analysis

During analysis, developers aim to produce a model of the system that is correct, complete, consistent, and unambiguous. Developers transform the use cases produced during requirements elicitation into an object model that completely describes the system. During this activity, developers discover ambiguities and inconsistencies in the use case model that they resolve with the client. The result of analysis is a system model annotated with attributes and operations.

Note: Look at the Analysis diagram in the Diagram section

System Design

During system design, developers define the design goals of the project and decompose the system into smaller subsystems that can be realized by individual teams. Developers also select strategies for building the system, such as the hardware/software platform on which the system will run, the persistent data management strategy, the global control flow, the access control policy, and the handling of boundary conditions.

Note: Look at the Architecture diagram in the Diagram section

Object Design

During object design, developers define solution domain objects to bridge the gap between the analysis model and the hardware/software platform defined during system design. This includes precisely describing object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals such as extensibility or understandability, and optimizing the object model for performance.

Note: Look at the Class diagram in the Diagram section

Implementation

During implementation, developers translate the solution domain model into source code. This includes implementing the attributes and methods of each object and integrating all the objects such that they function as a single system. The implementation activity spans the gap between the detailed object design model and a complete set of source code files that can be compiled.

Note: Look at the State Machine diagram in the Diagram section

Technologies Used

1. ***Database server**: PostgreSQL*
2. **Programming language**:

a. **Front-end**: *Typescript(Angular framework)*

b. **Back-end**: *Java(Spring framework)*

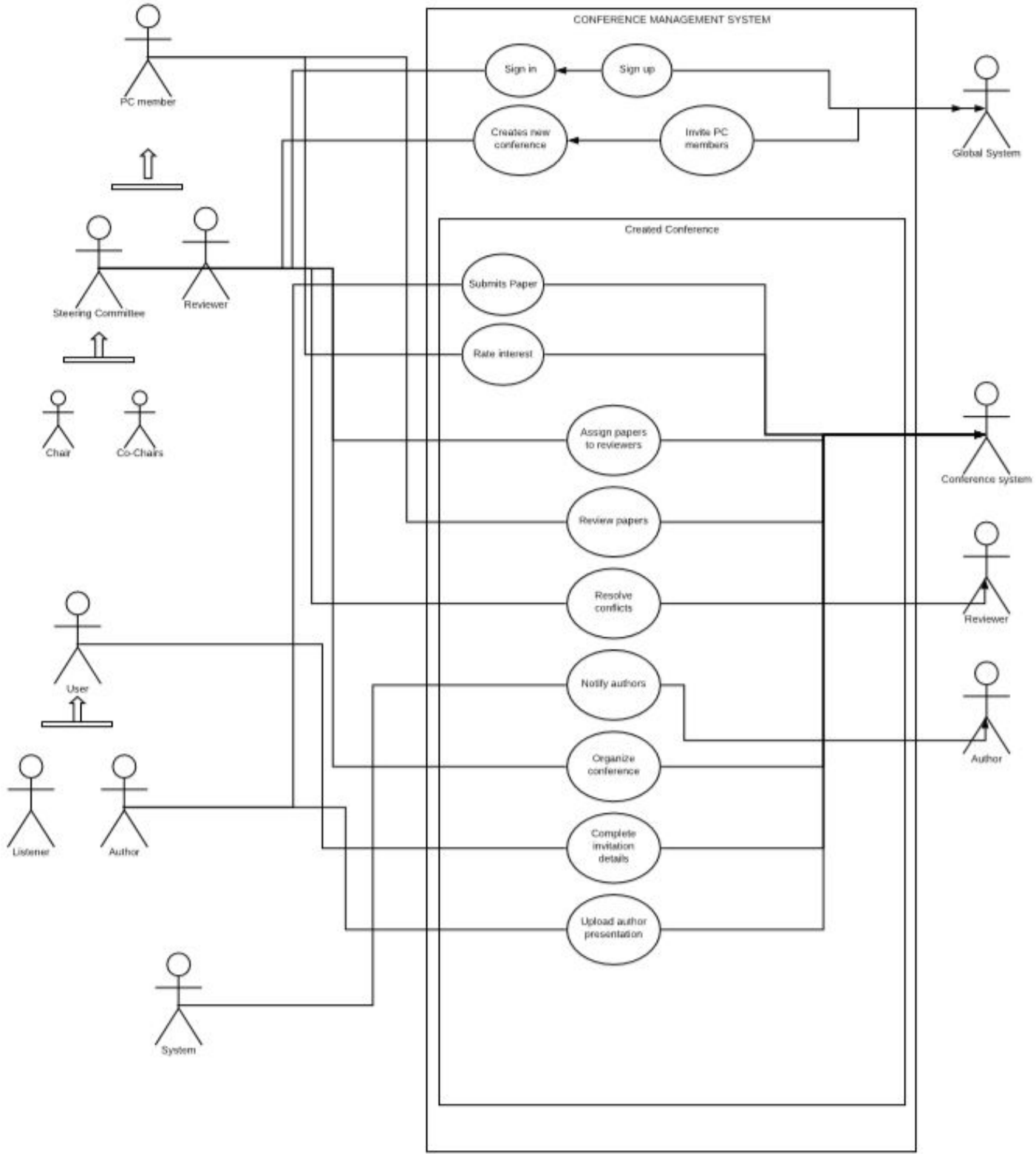
3. **ORM**: *Hibernate*
4. **Tools used for diagram creation**: *draw.io*
5. **Version control**: *Git*
6. **Task management**: *Trello*
7. **GUI creation**: *Moqups*

https://app.moqups.com/bbnYAKZsI6/view/page/ad64222d5

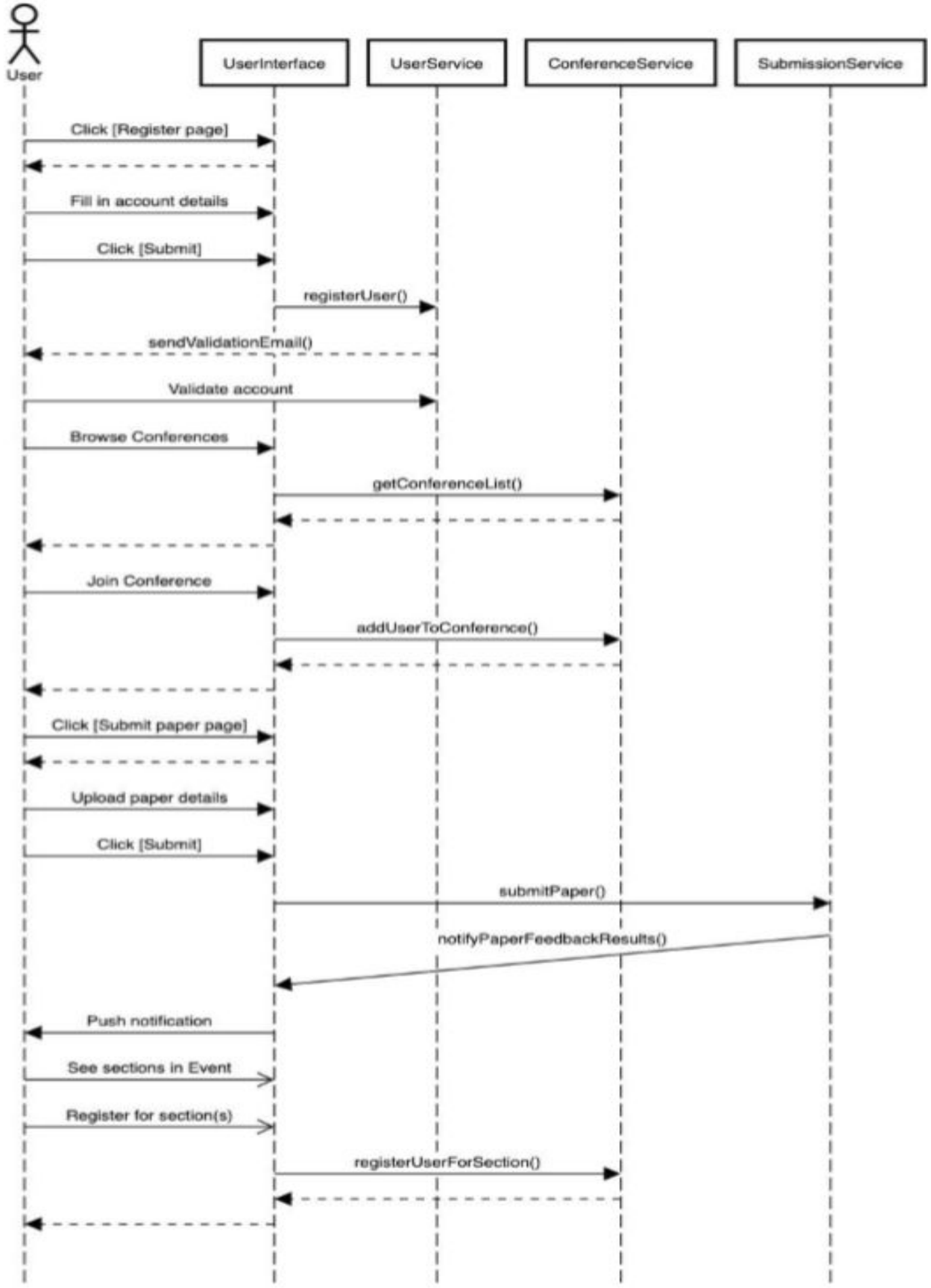
8. **Client-server communication**: *HTTP Request*
9. **Manual testing**: *bug reports in Trello*

Diagrams

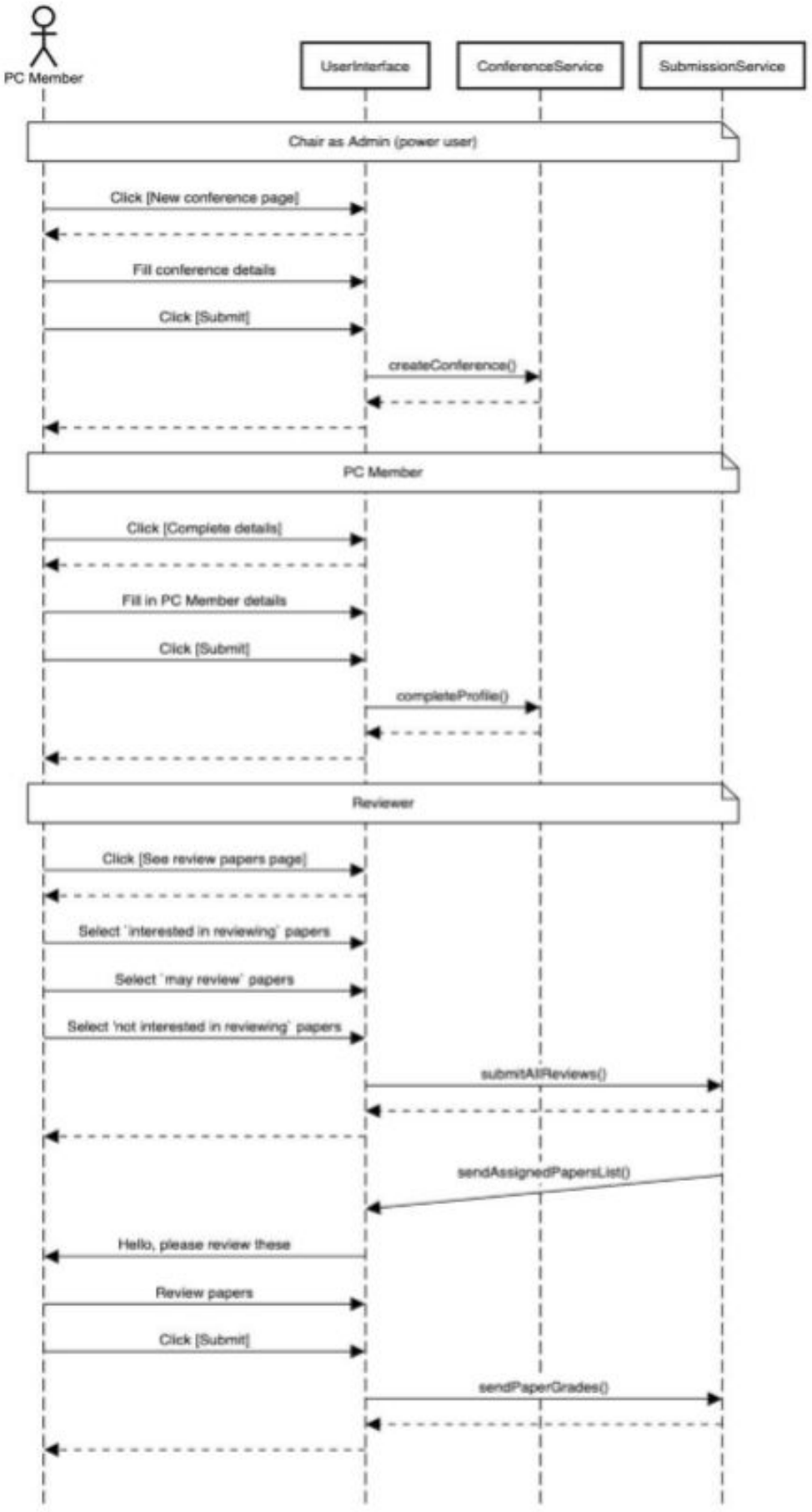
Use case diagram



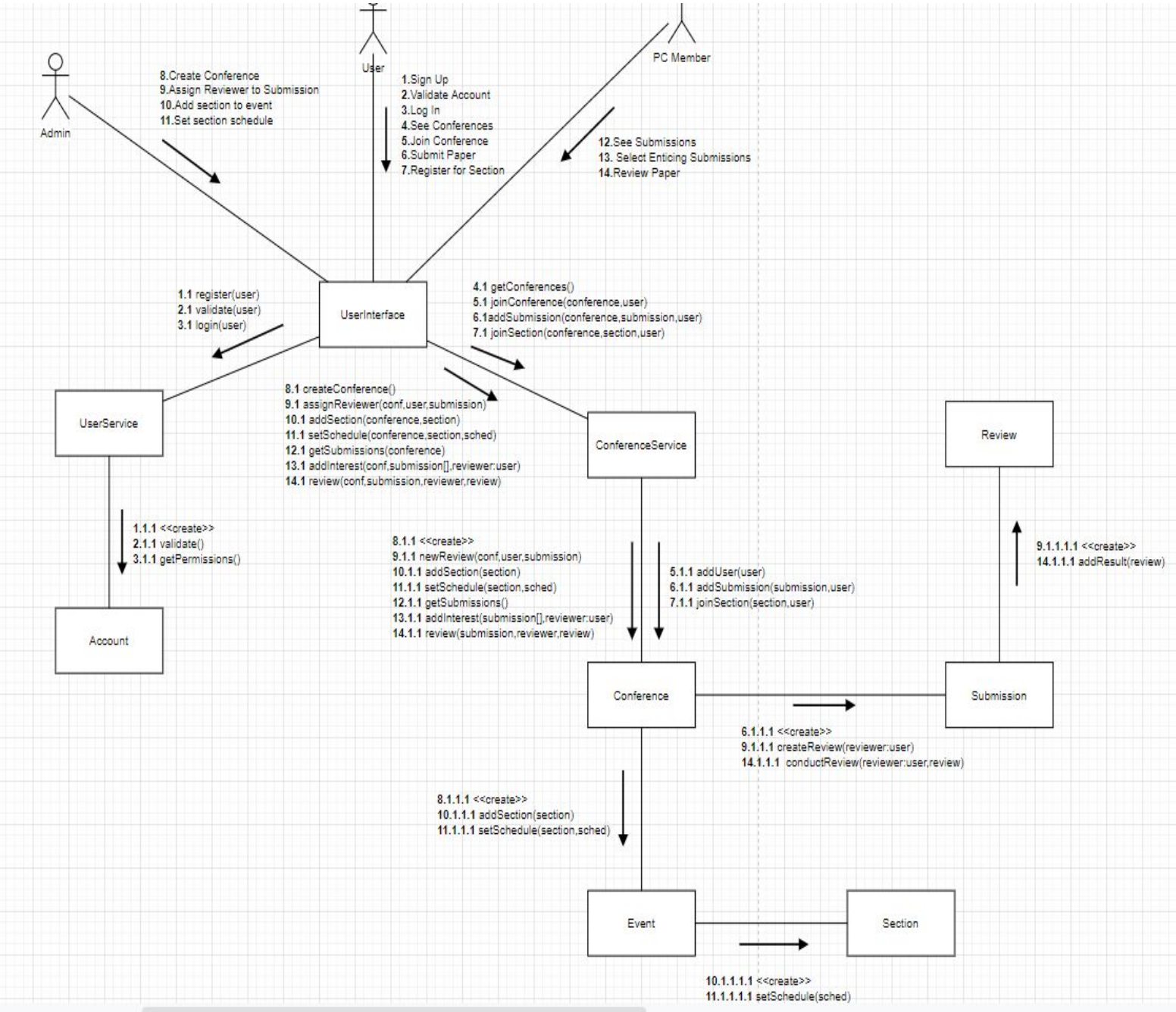
CMS User Sequence Diagram



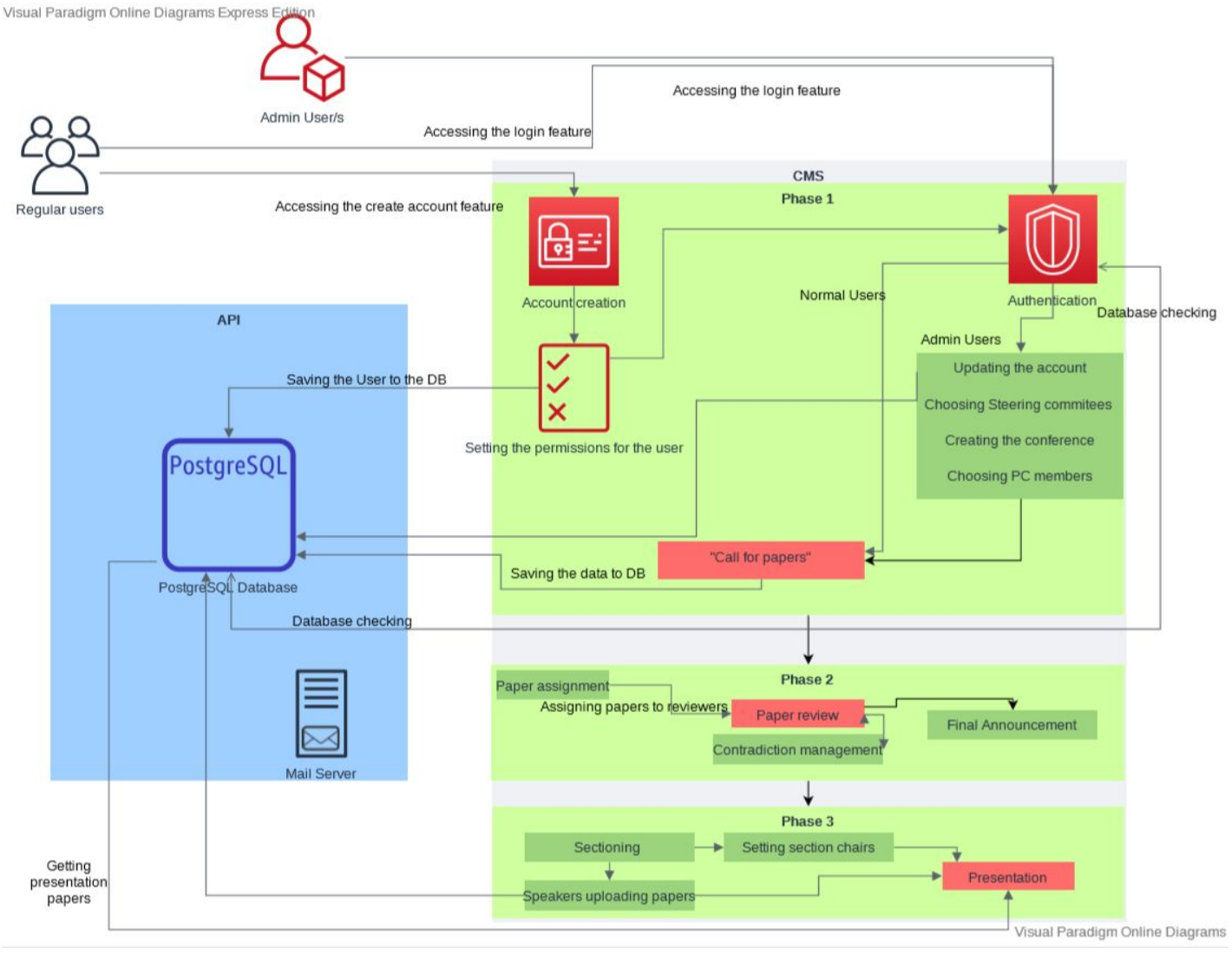
CMS Admin Sequence Diagram



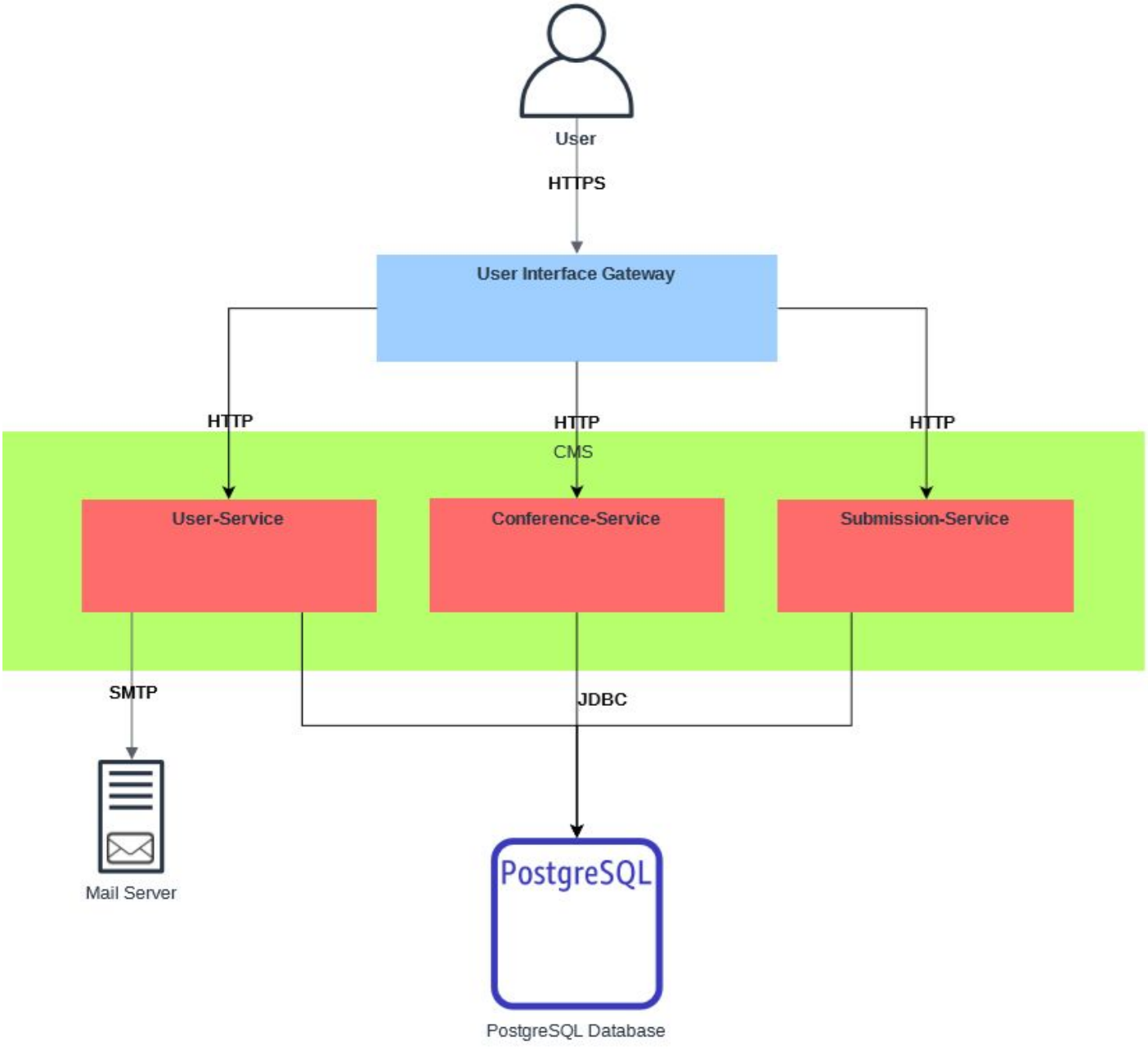
Communication Diagram



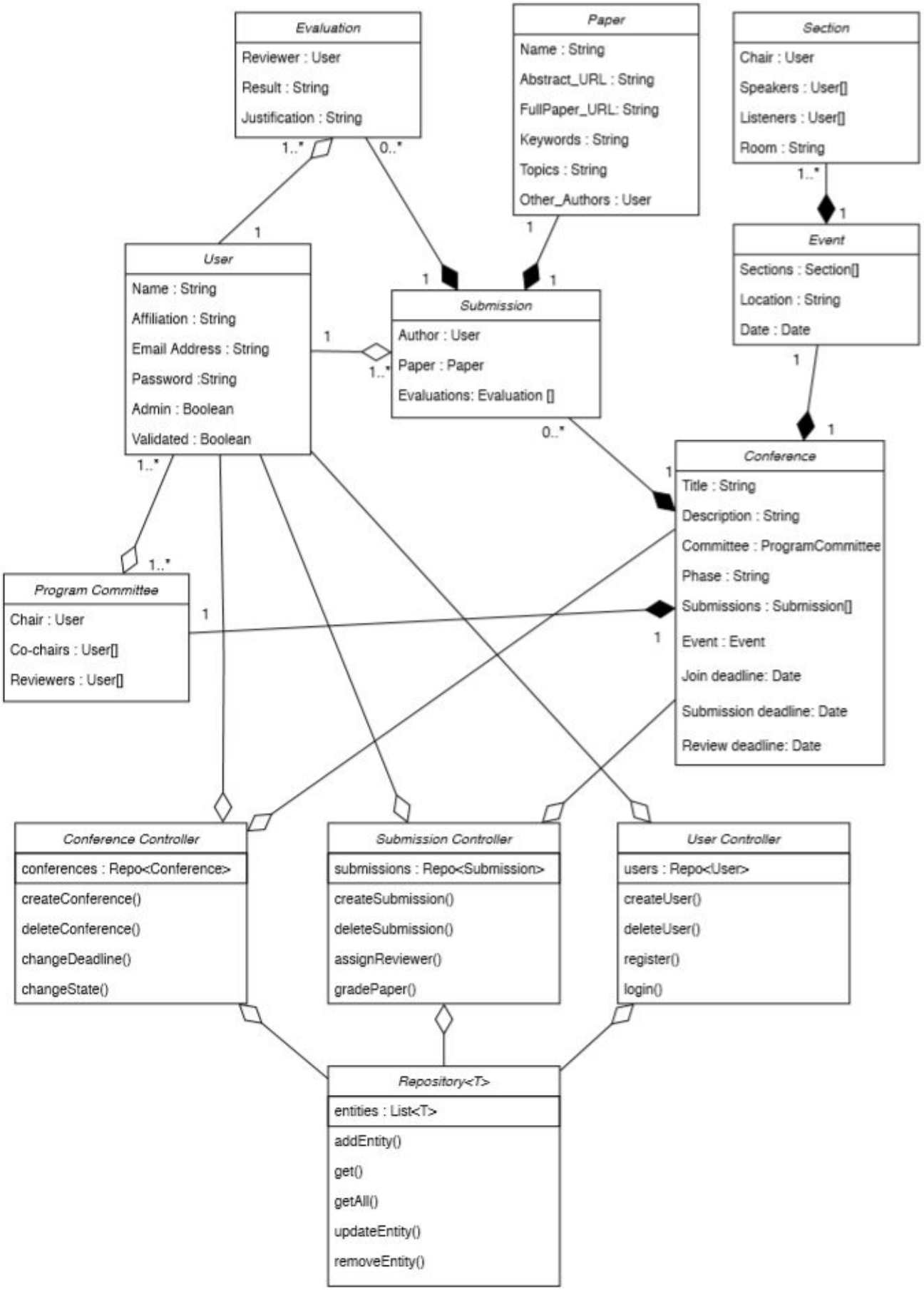
Architecture Diagram



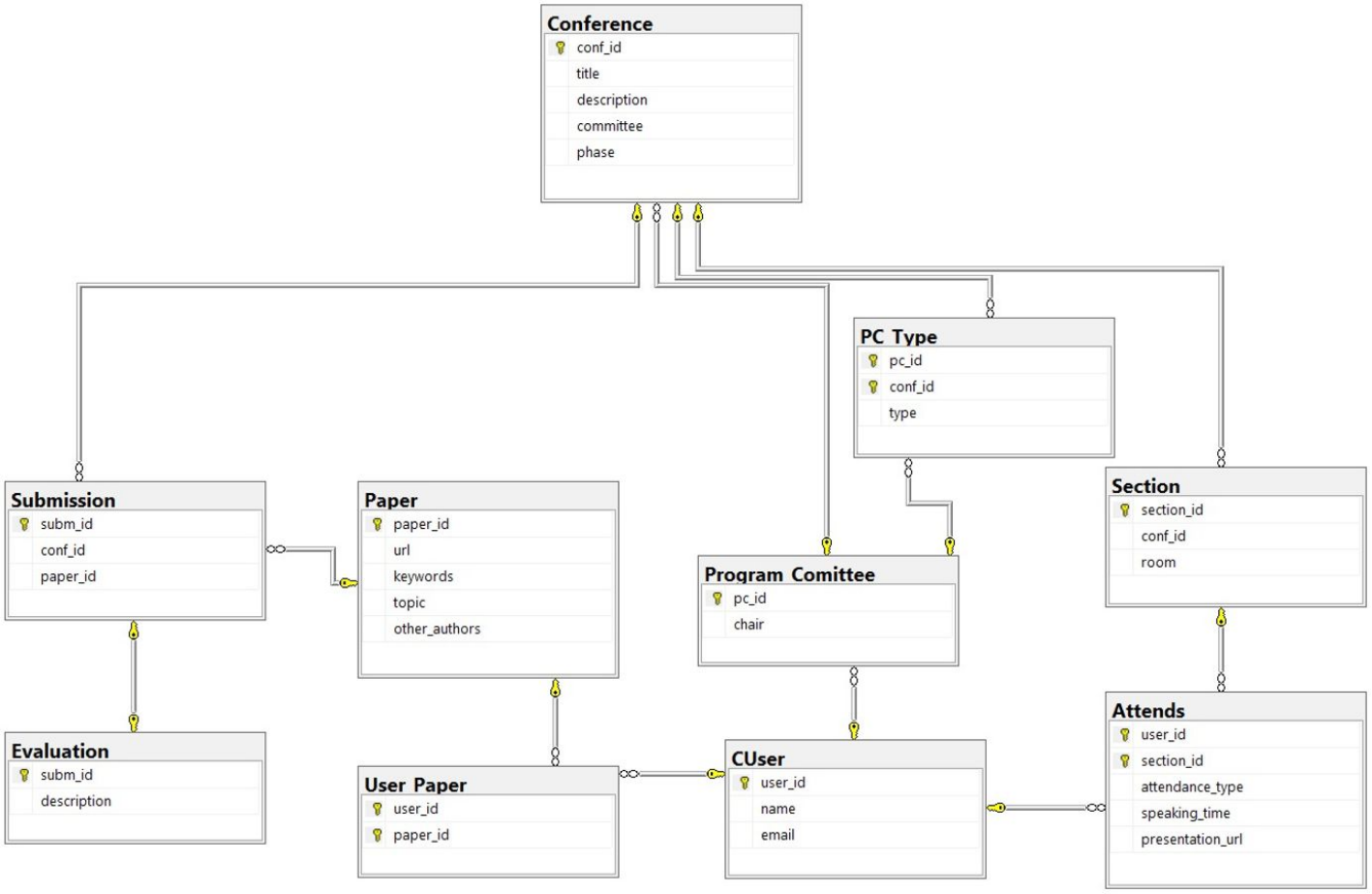
Architecture Diagram



Class Diagram



Database diagram



State machine diagram