Conference Management System (CMS) – Documentation

1. Short brief
2. Stages of implementing the application
3. Technologies

1. **THE BRIEF:**
   Our objective was to design and implement a web application that helps the users in the organization of a conference. We focused on the several stages of what organizing a conference requires:
   a. Creating user accounts which can be assigned different roles by the administrator (SC member) such as PC members or participants; each user is able to change personal profile information;
   b. The administrator (SC member) may create conferences;
   c. The chair/co-chair may change information about the conferences they are assigned to; they can also add more PC members to the conference;
   d. Participants can join various conferences and submit one proposal per conference (which can be or not accepted); they will be further notified about their paper status by email;
   e. Bidding and evaluating the proposals by the PC members:
      - PC members bid the proposals (whether they want or not to review a certain paper)
      - According to the results of the bidding phase, reviewers are assigned to each paper
      - Papers are then evaluated based on the qualifiers received from each reviewer; in case of disagreements, the chair may take the final decision
   f. Publishing the accepted papers + allowing the PC members to assign each accepted paper to a certain section of their conference, before the presentations

2. STAGES OF IMPLEMENTATION:

a. Analyzed the requirements and identified the features that have to be implemented

b. Started the design of the application – abstraction & modelling

- UML diagrams
    1. Structural diagrams
        a. Class diagrams
        b. Object diagrams
    2. Behavioral diagrams
        a. Use case diagrams
        b. Communication diagrams
        c. Sequence diagrams
- Non-UML diagram - Database diagram

The Analysis Model is a model designed in order to better understand the entities of the domain of the problem, but not only the problem itself. In our case we composed the Analysis model by the use of Use Case Diagrams, Class Diagrams, Object Diagrams and Sequence Diagrams.

The Analysis Model is a description of the tasks specified in our spoken language and it is helped by the functional view and documentation realised by the UML diagrams.

In our modelling activity, the process of constructing the models, the abstraction is kept by describing the same model in different situations from different points of view.

Given the definition of the model, it is clear that the views are models. But if we use this whole term (model), we risk confusing it. Yes, the views are models, but models specified through the prism to a single interest / point of view.

To be clear, we call models the descriptions of the application produced in namely periods of time from the realization of the application, so we have: Model of Analysis, Design Model, Implementation Model, Testing Model.

And before the Analysis Model we have the Requirements Model
(Model / Specification Requirements).

Depending on their nature, each model can be described through a number of views.

In our case:

- The functional view represents the function we need for our application to be running, it is described by the Use Case Diagrams.

- The architectural view describes the entities that offer us the functionalities between them, the structure and the relationships between them, it is represented in our case by the Class Diagram.
- The behavioural view is described by the Sequence Diagrams in our case and it shows how entities are connected to each other.

We also decided to structure our web application based on the MVC architectural pattern.

c. Decided upon the technologies used for the implementation (more details in the next section)
d. First trials and error with the technologies, because initially it did not go exactly smoothly
e. Finally stuck with the technologies we used until the end
f. Regular Zoom calls in order to check our progress and discuss&solve bugs

DETAILS REGARDING SYSTEM DESIGN:

DESIGN GOALS

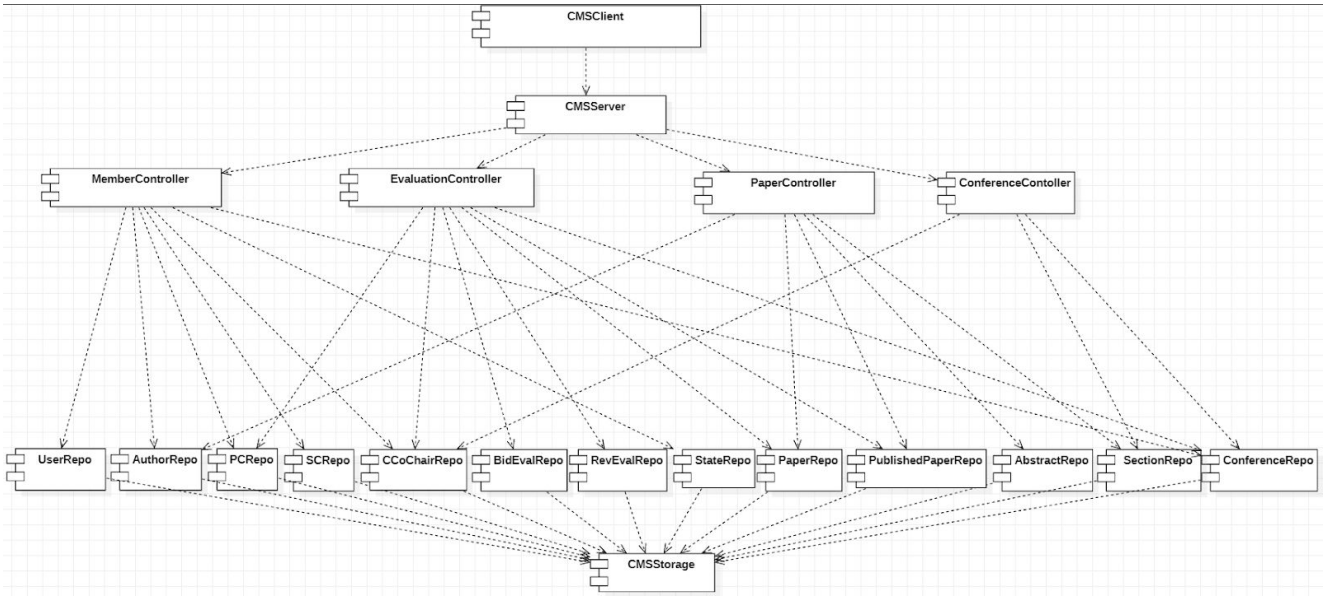
In the process of making our application, we have encountered several design goals that might fulfill our criteria and our desires. Of course that in trying to accomplish our standards, we had some drawbacks and we needed to decide some trade-offs. Our chosen goals are the following:

- Modifiability
- Reusability
- Understandability
- Traceability of requirements
- User-friendliness & Ease of use
- Low cost
- Functionality

Subsystem Decomposition

In this part we have considered the diagrams made in the analysis model, for the functional view, namely the use case diagrams.

In the following diagram, we will divide our system into subsystems identifying the services and repositories that we need to take into consideration.



For the architectural part we considered the Model-View-Controller + the Repository design patterns, as seen above.

Next we will detail the model part given the fact that we are going to use an ORM, Object-Relational Mapping. For each entity we will have some fields and we consider it futile to add to the repositories the getters and setters methods since the framework that we use (Spring + Lombok) will implement this automatically and will get the data from a database.

The only methods that we are going to implement are going to be located in controllers.

MemberController
Login(username, password)
pre:
username – string not null
password – string not null
UserRepo.getAll().filter(p->p.getUsername() == username) != NULL
UserRepo.getAll().filter(p->p.getUsername() == username).getPassword() == password
State == NULL

post:
State = new State(username)

LogOut()
pre:
State != NULL

post:
State == NULL

CreateAccount(username, Password, Email)

pre:

username: string not null
password: string not null
email: string not null
State == NULL
UserRepo.getAll().filter(p->p.getUsername() == username) == NULL
UserRepo.getAll().filter(p->p.getEmail() == email) == NULL

post:

UserRepo.add(username,password,email)

UpdateInfo(Username,Email,WebSite,Affiliation)

pre:

username: string not null
email: string not null
website: string not null
affiliation: string not null
UserRepo.getAll().filter(p->p.getUsername() == username) != NULL

post:

UserRepo.getAll().filter(p->p.getUsername() == username).setEmail(Email)
UserRepo.getAll().filter(p->p.getUsername() == username).setWebsite(Website)
UserRepo.getAll().filter(p->p.getUsername() == username).setAffiliation(affiliation)

SendEmail(Username,Title,Message,ReceveingEmail)

pre:

username: string
title: string not null
message: string not null
receveingEmail: string not null
receivingEmail.contains(‘*.[@]*.[.]*.) <- regex
State != NULL

post:

an email was sent to the receiving email with the Title and Message

Search(searchString)

pre:

searchString: string not null

post:

return
UserRepo.getAll().filter(p->p.getUsername().contains(searchString)).getUsername()

PaperController

addAbstract(author_id, name,Abstract, keyword, topics, additionalAuthors)

pre:

author_id: long not null
abstract: string not null
name: string not null
keyword: string not null
topics: string not null
additionalAuthors: string not null
meta-information: string not null
AuthorRepo.get(author_id) != null

post:

AuthorRepo.add(author_id,name,abstract)

Search(searchString)

pre:

searchString: string not null

post:

return AbstractRepo.getAll().filter(p->p.getName().contains(searchString)).getName()

EvaluationController

BidAbstract(username, abstractName, result, date)

pre:

Username: string not null
abstractName: string not null
result: integer not null
date: Date not null
PcRepo.getAll().filter(p->p.getUserId()==UserRepo.getAll()
.filter(u->u.getUsername() == username).getUserId()) != NULL
AbstractRepo.getAll().filter(p-> p.getName() == abstractName) != NULL
Date < ConferenceRepo.getBidDeadline()
Date > ConferenceRepo.getPaperDeadline()

post:

BidEvalRepo.add(AbstractRepo.getAll().filter(p->p.getName()==abstractName).getId(),
PcRepo.getAll().filter(p->p.getUserId()==UserRepo.getAll().filter(u->u.getUsername()
== username).getUserId()).getId(), result, date)

ReviewPaper(user_id, paper_id, result, date)

pre:

User_id: long not null
Paper_id: long not null
result: integer not null
date: Date not null
PcRepo.getAll().filter(p->p.getUserId() == user_id) != NULL

PaperRepo.getAll().filter(p->p.getId() == paper_id) != NULL

[PcRepo.getAll().filter(p->p.getUserId()==user_id).getId(),paper_id] IN
BidEvalRepo.getAll()
Date < ConferenceRepo.getReviewDeadline()
Date > ConferenceRepo. getBidDeadline ()

post:
BidEvalRepo.getAll().filter(userId==userId AND paperId==PaperId).setResult(result)
BidEvalRepo.getAll().filter(userId==userId AND paperId==PaperId).setDate(date)

AssignReviewEvaluation(user_id, pc_id, paper_id)
pre:
user_id long not null
pc_id long not null
paper_id long not null
CCoChairRepo.getAll().filter(p->p.getUserId() == user_id) != NULL
PCRepo.getAll().filter(p->p.getId() == pc_id) != NULL
PaperRepo.getAll().filter(p->p.getId() == paper_id) != NULL

post:
RevEvalRepo.add(CCoChairRepo.getAll().filter(p->p.getUserId()==user_id).getId(),
pc_id, paper_id,result:NULL,date:NULL)

ConferenceController
ChangePaperDeadline(user_id, date)
pre:
user_id long not null
date Date not null
CChairRepo.getAll().filter(p->p.getUserId() == user_id) != NULL
Date > ConfRepo.getAbstractDeadline()
Date < ConfRepo.getBidDeadline()

post:
ConfRepo.setPaperDeadline(date)

CreateConference(user_id,chair_id, co_chair_id, ConfTitle, abstractDeadline, paperDeadline, bidDeadline,reviewDeadline, conf_end_date)
pre:
User_id long not null
Chair_id long not null
Co_chair_id long not null
ConfTItle string not null
abstractDeadline Date not null
paperDeadline Date not null
bidDeadline Date not null
ReviewDeadline Date not null

Conf_end_date Date not null
UserRepo.getAll().filter(p->p.getId() = chair_id) != NULL
UserRepo.getAll().filter(p->p.getId() = co_chair_id) != NULL
SCRepo.getAll().filter(p->p.getUserId == user_id) != NULL
ConfRepo.getAll().filter(p->p.getName() == confTitle) == NULL
nowDate() < abstractDeadline < paperDeadline < bidDeadline < reviewDeadline < conf_end_date

post:
new Conference(chair_id, co_chair_id, ConfTitle, startDate: now(), conf_end_date, abstractDeadline,paperDeadline,bidDeadline,reviewDeadline)
pc_id1= PCRepo.add(chair_id,conference_id).getId()
ChairRepo.add(pc_id1,conference_id)
pc_id2 = PCRepo.add(co_chair_id,conference_id).getId()
ChairRepo.add(pc_id2,conference_id)

3. TECHNOLOGIES:

a. **Programming languages:**
   - Front-end: JavaScript with Angular, HTML, CSS
   - Back-end: Java (using Spring Framework)

b. **ORM:** Hibernate

c. **Database server:** PostgreSQL

d. **Version control:** GitHub

e. **Tools for diagrams:** draw.io and StarUML

f. **Client-server communication:** REST API (Java)

g. **Testing – manual & automated:**
   - Manual testing: bug reports during Zoom calls (Google Docs)
   - Automated testing: TBA