

~~~~

Work Flow of Leave Request Management System

Table of Contents

[1. Introduction 5](#_Toc533030164)

[1.1 Purpose 5](#_Toc533030165)

[2. Technologies used 6](#_Toc533030166)

[3. Requirements - Use-Case – Usage Scenarios 7](#_Toc533030167)

[4. High Level Design 9](#_Toc533030168)

[5. Detailed Design 10](#_Toc533030169)

[6. Design Alternatives 14](#_Toc533030170)

[7. Issues, Risk and Dependencies 14](#_Toc533030171)

[8. Future Considerations 14](#_Toc533030172)

[9. References 14](#_Toc533030173)

# Introduction

Our project is a leave management system in company where an employee is created via a REST web service call and leaves are applied, approved or rejected via AMQP, after reading from Queue and storing them on the database a manager is able to approve or reject a leave that is pending.

All employees can apply for any type of leave, an employee can only view his own pending, approved or rejected leaves but a manager can see all pending leaves of all employees and can add comments to them before approving or rejecting

## Purpose

The purpose of this project is to demonstrate the proof of concept in enterprise applications, the concepts that have been implemented are: -

\* How high-volume **Messaging Systems (AMQP)** work

\* How the **REST web service** is implemented

\* How the **aspect-oriented programming (AOP)** is implemented

\* How hibernate handles the mapping of objects to database tables

\* How the security systems (Authentication) work.

\* How the **Group Validation** works

\* N tier model architecture

# Technologies used

To achieve the mentioned functionality, we applied technologies such as

\* Spring REST web service: in Leave Rest Server

\* Spring REST web service Rest Template client: in Leave Rest Client

\* Spring MVC in Leave Rest Client to handle interaction with System user

\* Hibernate Mapping: in Leave Rest Server, Dao Layer

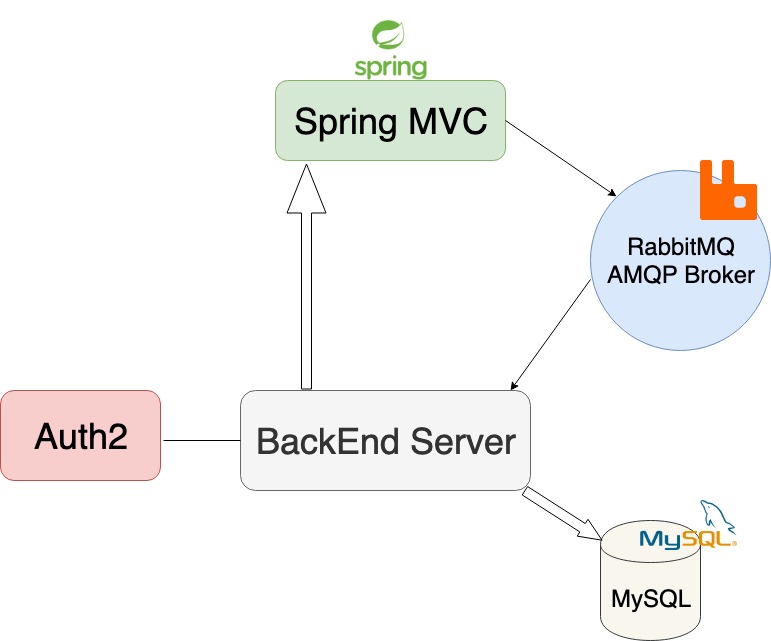
\* AMQP messaging System for in-service communication

\* Spring Enterprise Integration is integrated with Speed Service Application to handle Integration, Transformation and Filtering

\* AOP used in Speed Service App for Logging.

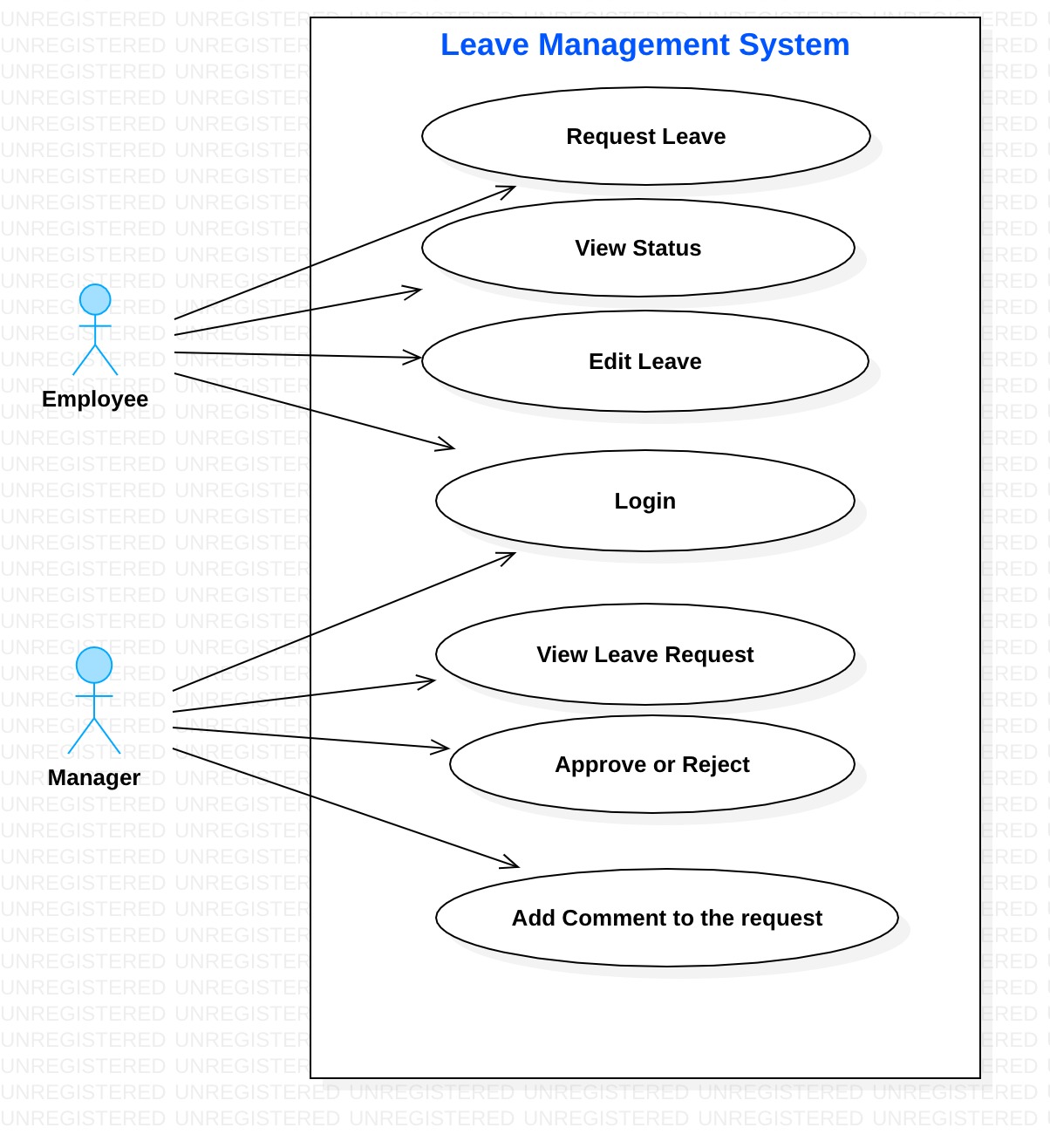
\* Java Mail used to send leave request to manager.

# Software Architecture



# Requirements - Use-Case – Usage Scenarios

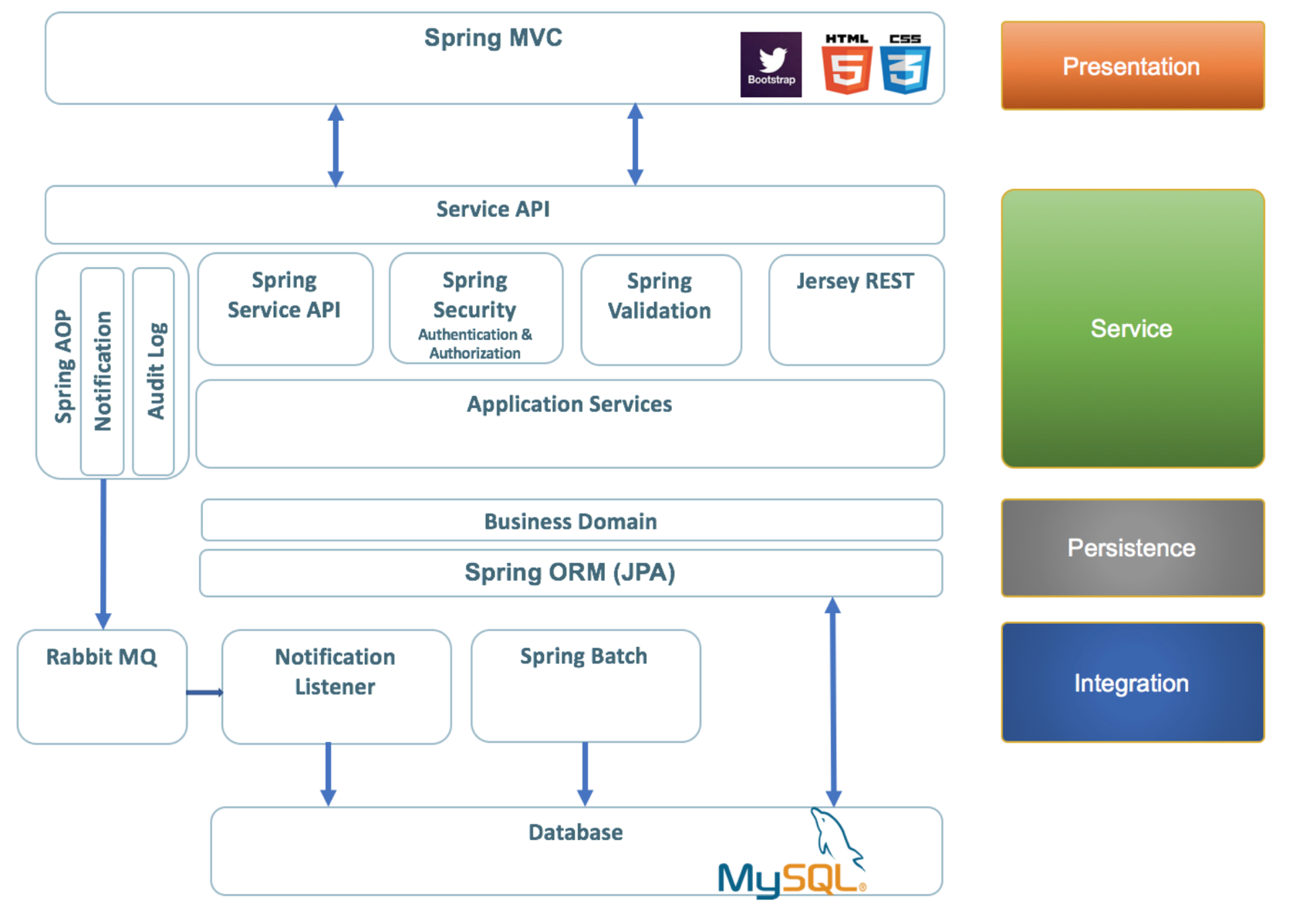
**4.1 Use Case Diagram**



**4.2. Add Member Use Case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | **MAIN FLOW** |
| **Step** |  | **Action** | |  |
| 1 |  | The manager adds member(employee) | | |
| 2 |  | Because it’s a protected resource our system will redirect user to authorization login page | | |
| 3 |  | The user(employee) will enter their credentials | | |
|  |  |  | |  |
| 4 |  | An employee can apply leave request | | |
|  |  |  | | |
| 5 |  | Notification send email to manager | | |
|  |  |  | |  |
| 6 |  | Only the manger can see the leave request of employee | | |
| 7 |  | The manager can approve or reject leave request | | |
| 8 |  | An employee will get notify their request will be approve or reject | | |
|  |  | Use case ends | |  |

# High Level Design



High Level Design

# Detailed Design

6.1 Class Diagram

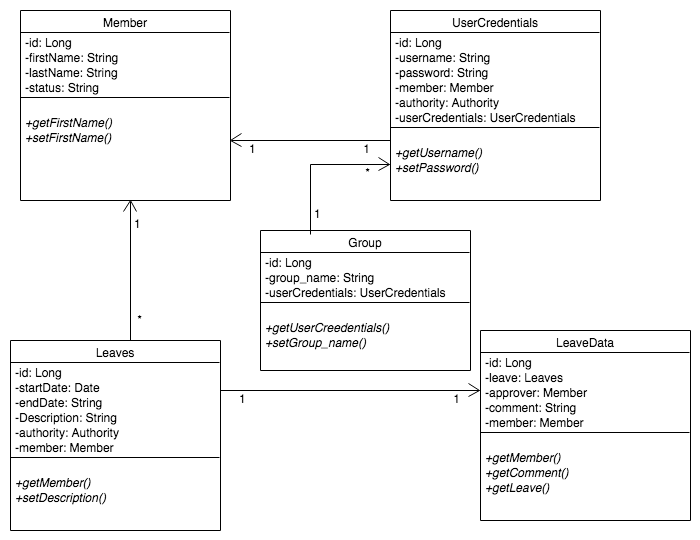
****

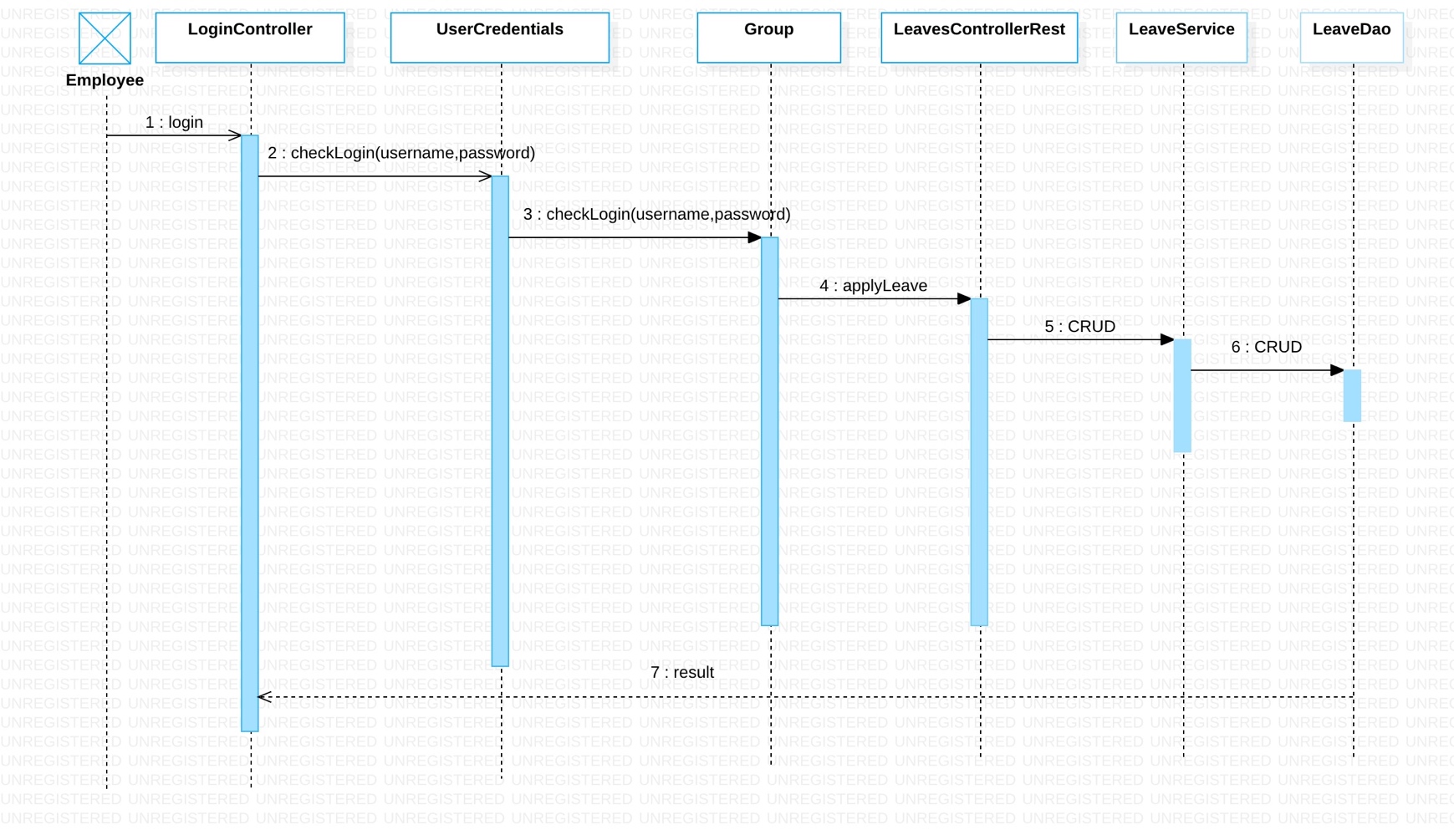
Image 2. Class Diagram

****Interaction Diagram****

5.1.1. Leave Request CRUD operation

Class Diagram

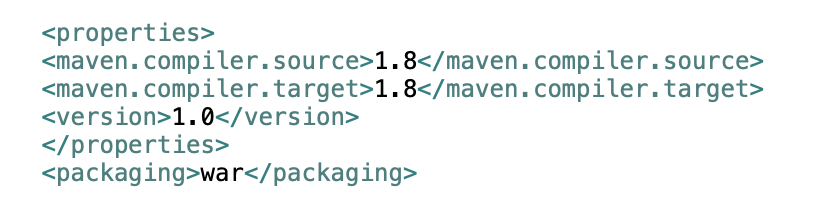
6.2. Sequence Diagram



Sequence Diagram

# Issues, Risk and Dependencies

\* The first issue is the configuration with applicationContext.xml, we almost one day to for this issue, but we solved this problem by add <packaging> war </packaging>



\* In other day, we focused the Relationships between entities issue.

\* Getting lazy loaded objects from a holding object

\* Role based security at method level.

\* Our team issue: Having all four micro services communicate over the network.

# Future Considerations

\* According to the sort time, Auth2 will be implement in order to redirecting to login even after Authentication.

\* We will apply spring batch processing.

\* In future, we can implement this application with leave balance.

# 9. References

Demos from Professor Joe Bruen

<http://www.udemy.com>