Project overview - Kaos Chat

Group name: Kaos Development

Group members:
Johannes Uhr
19931114-0058
uhrj@student.chalmers.se

David Fogelberg
19940309-3819
fodavid@student.chalmers.se

David Johnsson
19940830-5879
davjohns@student.chalmers.se

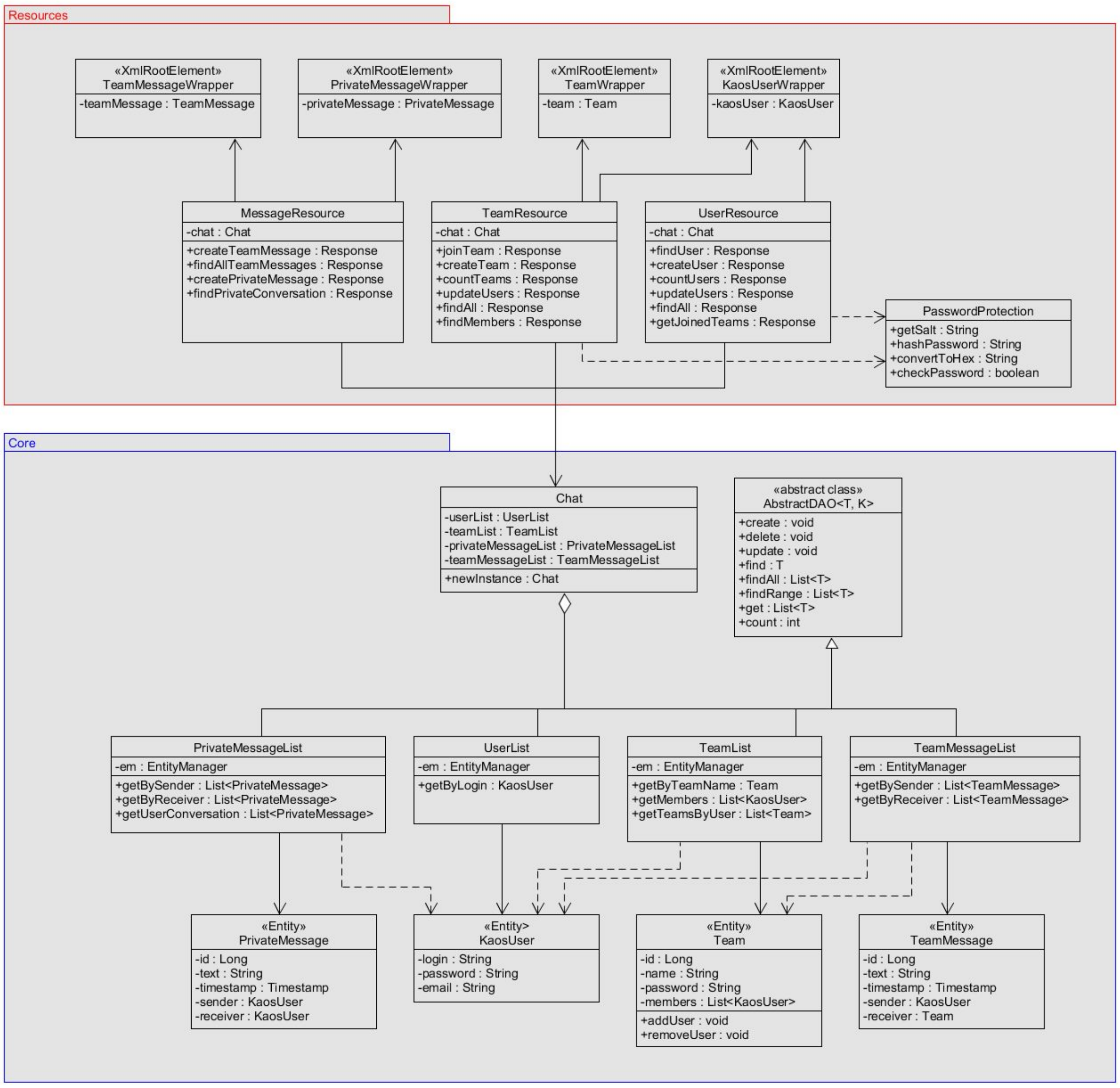
Henrik Numé
19900711-1454
numeh@student.chalmers.se

General overview
Our web application is a chat site where users can chat in different groups. It's inspired by the popular chat app "Slack". This app allows users to form groups and write messages that all group members can see, as well as private messages to members of the group. Anyone can create new teams, but to join an already existing team, the user has to type the correct password of that team.

Use cases
- Register a new user
- Log in
- Create a new team
- Join an existing team
- Write in team chat
- Write in private chat
- Log out

Technical design

Object oriented model UML



Selected approach

We decided to use a service based approach. The core is made in java with the Java Persistence API and the backend uses the REST API to supply the angular frontend with necessary data. A deciding factor to our choice of approach is the fact that AngularJS is very popular and widely used today and therefore an important skill for us to learn.

Participating software components

The web-application is deployed on a Glassfish 4.1 server running Java DB as a database with the Derby jdbc driver. Java EE 7 is used for the business logic. Our physical setup is non-distributed. All of the functionality is located on single server.

The presentation layer is using jQuery, AngularJS and Bootstrap libraries. It is communicating with the RESTful service using JSON objects.

Frontend modules
The frontend is made with angular and is made up of two modules: *app* and *chat*.

App is the main module and contains controllers for registering users and logging in, as well as all services, i.e. userService, authenticationService, teamService and messageService. *App* is also responsible for using $routeProvider to direct the user to the right partials.

Chat contains controllers for the main chat partial, which is the main interface of the app, as well as controllers for creating and joining teams.

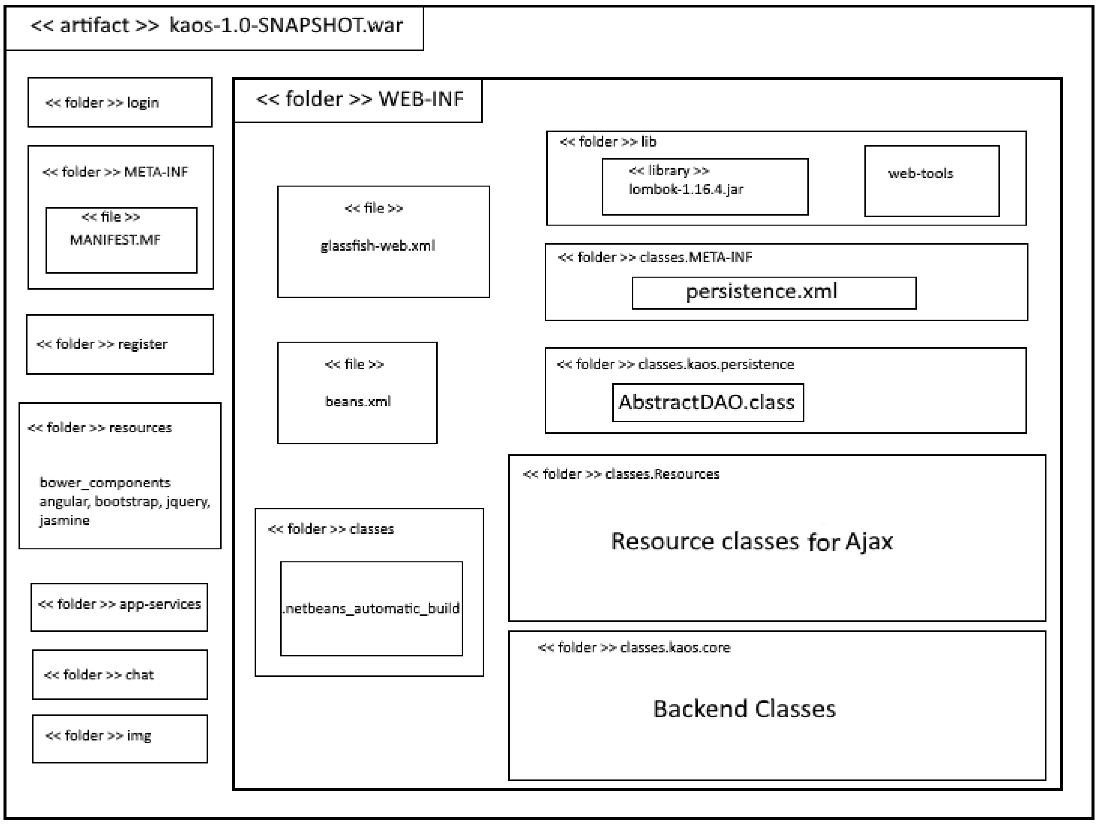
Backend modules
The backend consists of two java modules: *core* and *resources.*

The *core* module holds the database entity classes and the classes that are used to insert and read from the database.

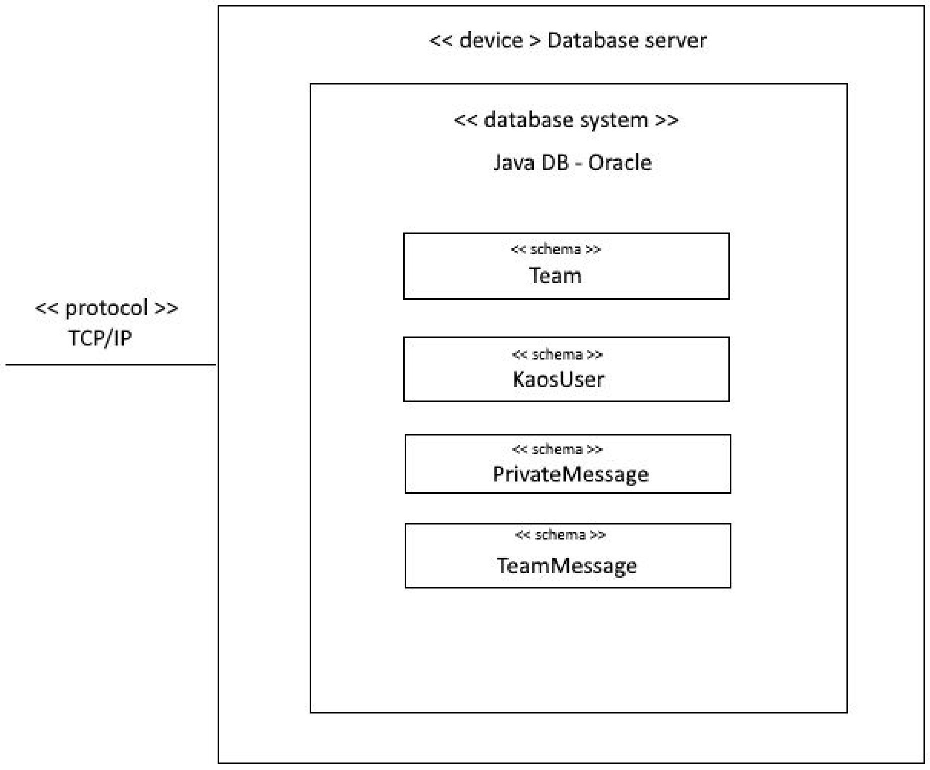
The *resources* module contains all the classes that are used by the frontend to GET, POST, PUT or DELETE from the database. The module also contains classes for password protection and wrapping objects before sending them.

Database Schema
KaosUser(login, email, password)
Team(name, password, members)
members -> List<KaosUser>
TeamMessage(id, text, timestamp, sender, receiver)
sender -> KaosUser
receiver -> Team
PrivateMessage(id, text, timestamp, sender, receiver)
sender -> KaosUser
receiver -> KaosUser

Deployment diagram:



<< artifact >> kaos-1.0-SNAPSHOT.war
<< folder >> login
<< folder >> META-INF
<< file >> MANIFEST.MF
<< folder >> register
<< folder >> resources
bower_components angular, bootstrap, jquery, jasmine
<< folder >> app-services
<< folder >> chat
<< folder >> img
<< folder >> WEB-INF
<< file >> glassfish-web.xml
<< file >> beans.xml
<< folder >> classes
.netbeans_automatic_build
<< folder >> lib
<< library >> lombok-1.16.4.jar
web-tools
<< folder >> classes.META-INF
persistence.xml
<< folder >> classes.kaos.persistence
AbstractDAO.class
<< folder >> classes.Resources
Resource classes for Ajax
<< folder >> classes.kaos.core
Backend Classes



<< device > Database server
<< database system >>
Java DB - Oracle
<< schema >> Team
<< schema >> KaosUser
<< schema >> PrivateMessage
<< schema >> TeamMessage
<< protocol >> TCP/IP