TU Vienna

Software Architecture

SWAG - Assignment 3

Authors:
Frieder Ulm, *0527031*, 0527031@student.tuwien.ac.at
Matthias Steinboeck, *0527943*, e0527943@student.tuwien.ac.at
Hubert Hirsch, *0625008*, e0625008@student.tuwien.ac.at
Eugen Dahm, *0625325*, e0625325@student.tuwien.ac.at
Philipp Raich, *0404014*, e0404014@student.tuwien.ac.at

Group: **swa043**

June 12, 2011

Contents

1 Use Cases **2**

2 Issues and Decisions **3**

2.1 Implementation Details 3

3 Component Diagram **4**

4 Database Model **5**

1 Use Cases

The following Use Cases where identified and implemented

Use Case	**Create User Account**
Goal in Context	User can create a account
Additional Notes	Starting base is initialized, user gets an e-mail with his generated password

Use Case	**Delete User Account**
Goal in Context	User can delete his account
Additional Notes	Bases are abandoned and can be taken by other players

Use Case	**Login/Logout**
Goal in Context	User can Login/Logout on/off his account
Additional Notes	Game State must be saved correctly until next Login, while the game continues and already taken actions continue

Use Case	**Send Ingame Message**
Goal in Context	User can send ingame messages to other users

Use Case	**Receive Ingame Message**
Goal in Context	User can receive ingame messages from other users

Use Case	**Build Base**
Goal in Context	User can build base on a map

Use Case	**Build Building**
Goal in Context	User can build building on a base

Use Case	**Upgrade Building**
Goal in Context	User can upgrade his existing buildings

Use Case	**Navigate Maps**
Goal in Context	User can navigate through existing maps

Use Case	**Build Troops**
Goal in Context	User can build troops

Use Case	**Form Squad**
Goal in Context	User can form squads with built troops

Use Case	**Move Squads**
Goal in Context	User can move his sqauds to other squares
Additional Notes	Squares occupied by foreign squads will be attacked by moving there, the "stronger" squads win, defeated troops disappear, if only foreign buildings remain on the square, they are attacked and will dissapear, remaning resources are assigned to the attacker

2 Issues and Decisions

Issue	The Application must be able to scale according to a unknown player count
Decision	Server load is minimized through gui-rendering executed client-side (html-prefetch & JavaScript)
Status	Implemented
Constraints	None
Related Principles	Thick Client
Related Artifacts	GUI

Issue	Reliable and fast communication between GUI and Business Logic to minimize overhead and loading times
Decision	JSON-Format mesaging between GUI (Client) and Business Logic (Server)
Status	Implemented
Constraints	None
Related Principles	Thick Client
Related Artifacts	All GUI-related Components

Issue	The application must be extensible
Decision	New modules and Services can easily be implemented in the exisiting structure
Status	Implemented
Constraints	None
Related Principles	Extensibility
Related Artifacts	All

Issue	All game-related information should be logged
Decision	An Interceptor is implemented that calls the Logger-Service explicitly
Status	Proposal
Constraints	None
Related Principles	Interceptor
Related Artifacts	LoggingInterceptor

Issue	All messages and transactions should be in the right order and the delivery reliable
Decision	A Message Queue is implemented that relays all messages and transactions
Status	Proposal
Constraints	None
Related Principles	Message Queue
Related Artifacts	MessageQueue

2.1 Implementation Details

Application Server: **Glassfish 3.1**
Database: **PostgreSQL 8.4**
Server-Side (Business Logic): **JAVA/JSP/AJAX**
Client-Side (Rendering): **JavaScript**

The following JavaScript Libraries were used:

- jquery
- requirejs
- jquery.cookie
- jquery.bind
- jquery.ui
- md5

3 Component Diagram

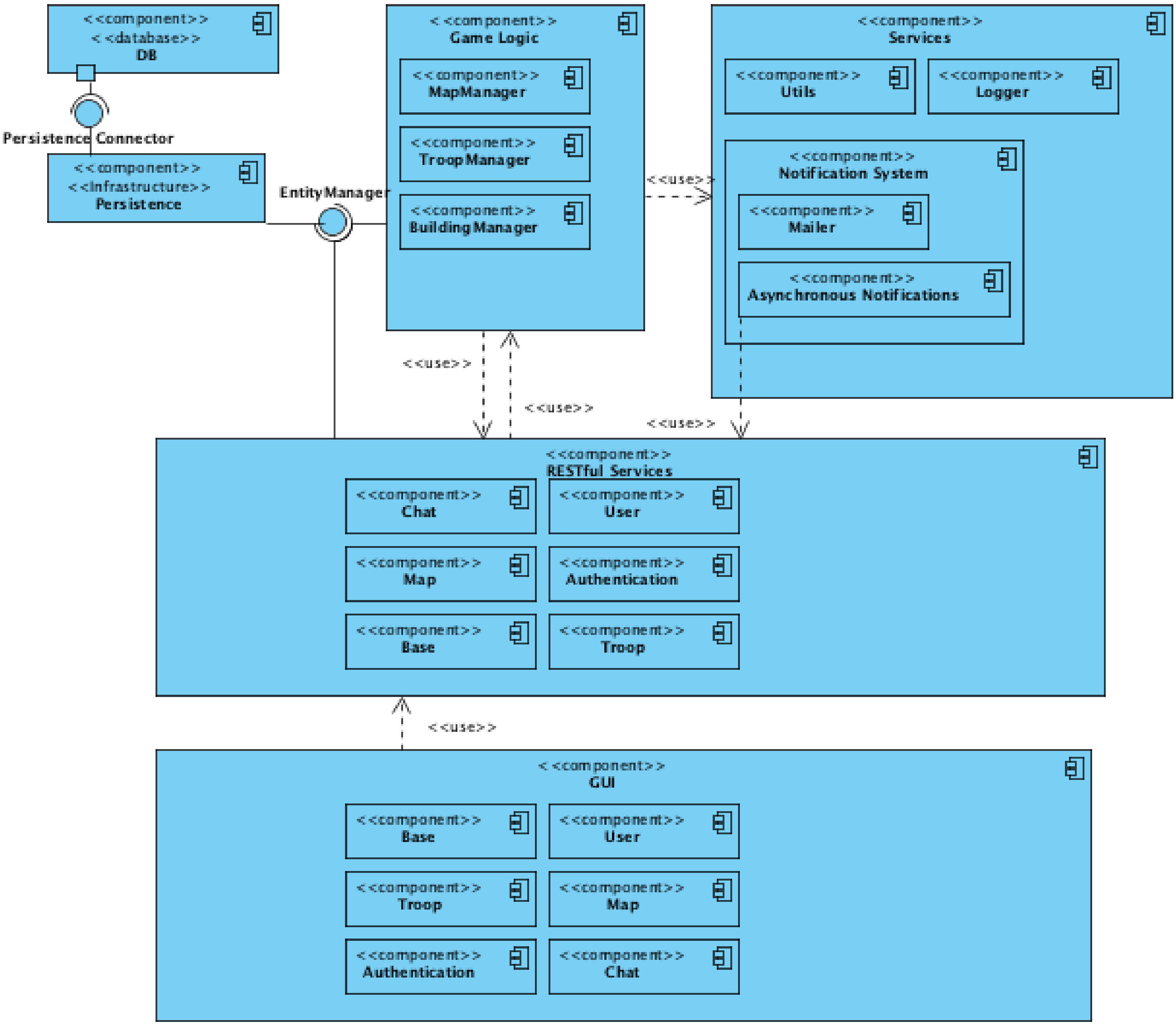

Figure 1: SWAG Component Diagram

The component diagram may be observed in figure 3. The main feature of the architecture is the speed that achieved by minimizing the communication between the RESTful Services at the server side and the GUI at the client side. This is achieved by prefetching HTML-content at once and loading the corresponding HTML content client side by JavaScript. The communication overhead is minimized by using JSON-formatted messages for all Services.

The components visible at the client side are thus only JavaScript components, rendering the information returned by the RESTful Services.

4 Database Model

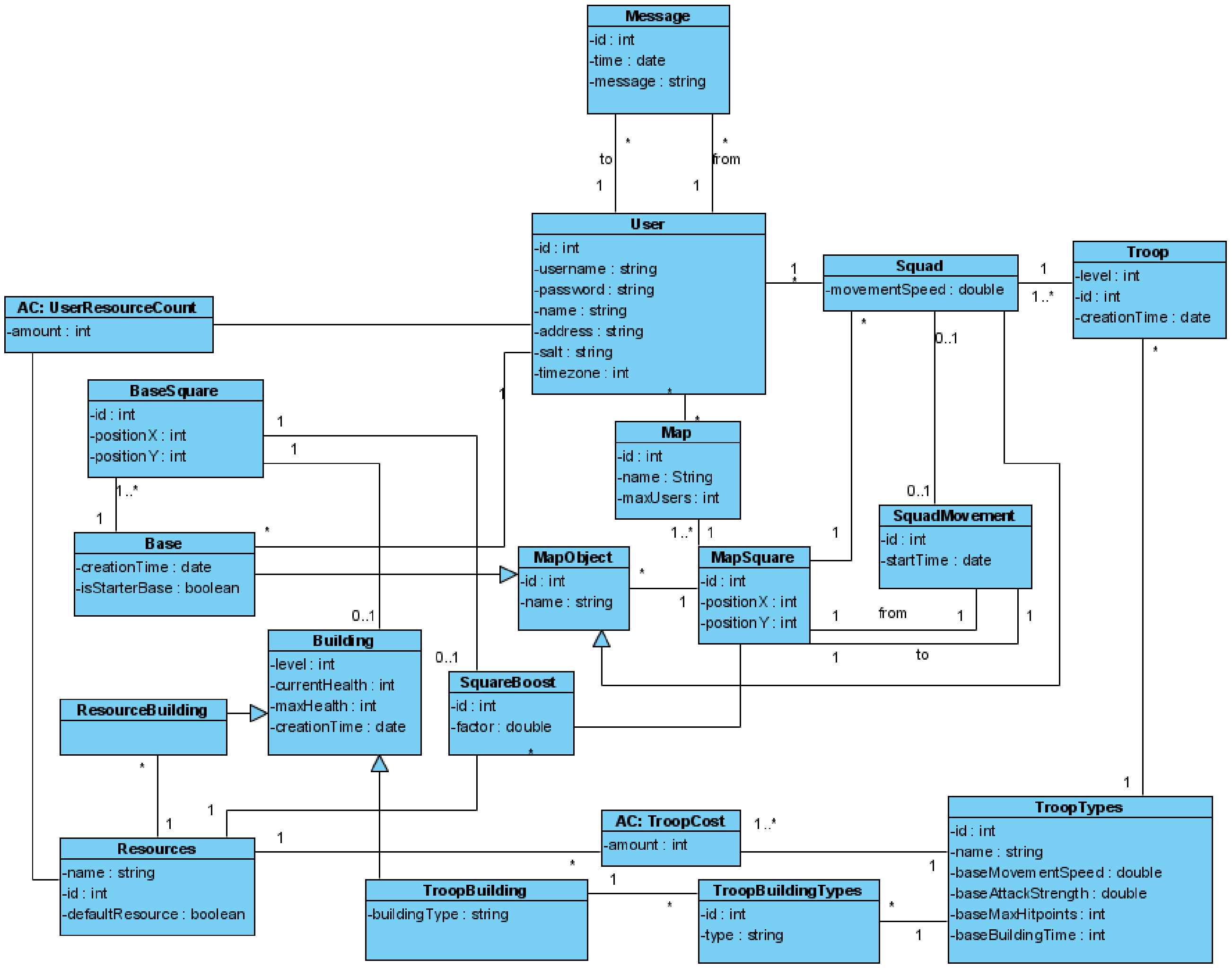

Figure 2: SWAG Database Model