BetGame

-Design Specification-

Contents

[Overview 3](#_Toc399004600)

[Business Background 3](#_Toc399004601)

[Architecture 3](#_Toc399004602)

[Front end/View 4](#_Toc399004603)

[Controllers 4](#_Toc399004604)

[Services 4](#_Toc399004605)

[Logical View 5](#_Toc399004606)

[Login/Registration 5](#_Toc399004607)

[User Home 5](#_Toc399004608)

[Game creation 5](#_Toc399004609)

[Admin Portal 6](#_Toc399004610)

[User Portal 6](#_Toc399004611)

[Software Stack View 6](#_Toc399004612)

[Data View 7](#_Toc399004613)

[Components 7](#_Toc399004614)

[Authentication 8](#_Toc399004615)

[Error Handling 8](#_Toc399004616)

[Data Access layer 8](#_Toc399004617)

[Archiving 8](#_Toc399004618)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 2014.09.20 | Geza Nagy | v1.0 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

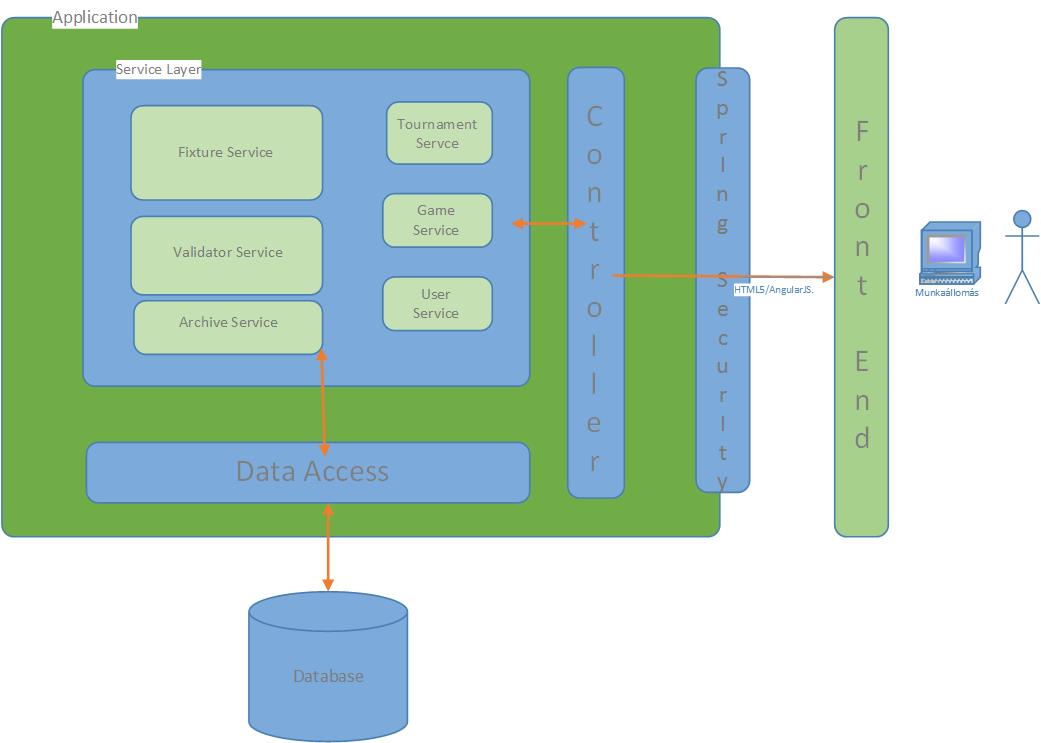
# Overview

This document is going to describe the high level architecture and technical choices which have been made for the implementation of the BetGame application.

## Business Background

Since the old ways of betting for tournaments – like via facebook or e-mails -are really uncomfortable the BetGame application is make the whole process easier and provide more fun for the players. Every step of the game is handled and everything is automated where it’s possible.

Architecture



The basic purpose of this site is to show information in a very comfort and fast way to the users. This is the main goal of the solution/architecture above.

## Front end/View

The UI which the users can see is a static HTML and content and the dynamic part is provided by AngularJS’s Client side MVC solution.

## Controllers

Every page/module on the screen has its own REST controller which will provide information and interface for apply modifications on the back-end. The details of every controller is not part of this document.

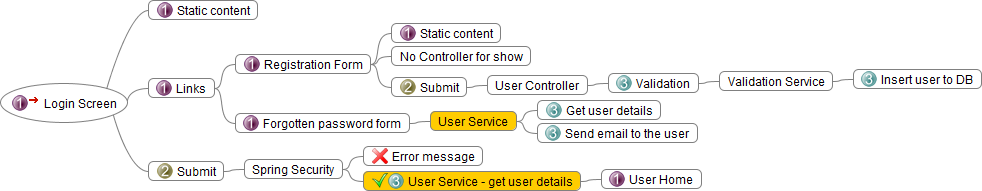
## Services

The engine of the application is this service layer. These components placed behind the controllers so they are not reachable directly from the front end. The planned Services:

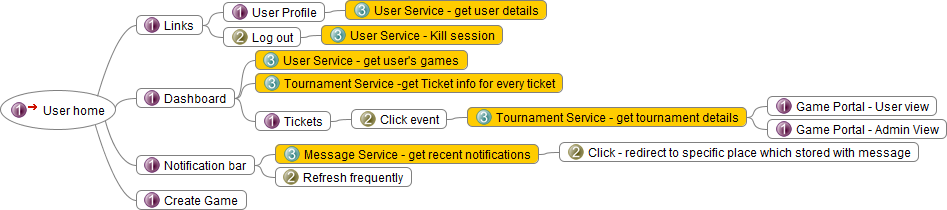
* ***Tournament Service:*** The scope of this component is to get information and in some cases do modifications related to the tournament. The tournament part is different from the Game part in the meaning of everything happens to the teams/groups during the Championship or Cup is a tournament event. These events has consequences and these have effects of the players as they are betting and their scores are changes.
* ***Game Service:*** Everything which is excluded from the tournament but related to the players and their competition belongs to this service. E.g.: Rules, Scores etc.
* ***User Service:*** This component is handle the logged in user actions, or handles the registration of a new one and also helps in the “Attendees” part when creating a new game.
* ***Fixture Service:*** Maybe the most complicated part of the application. The concept of the service is calculate/recalculate the fixtures of a tournament by its structure or after a change which has made by user interaction.
* ***Archive Service:*** The detailed model structure below can cause really huge table sin the database so when a tournament has ended and not visited for a while all the information of it will go to an archive table and it will be also reachable until the game itself is deleted but it’s obvious the information providing can be slower in this case.
* ***Validation Service:*** Every action the users can make – which requires - will go through a validation process. These will be detailed when they are come up during the implementation.

# Logical View

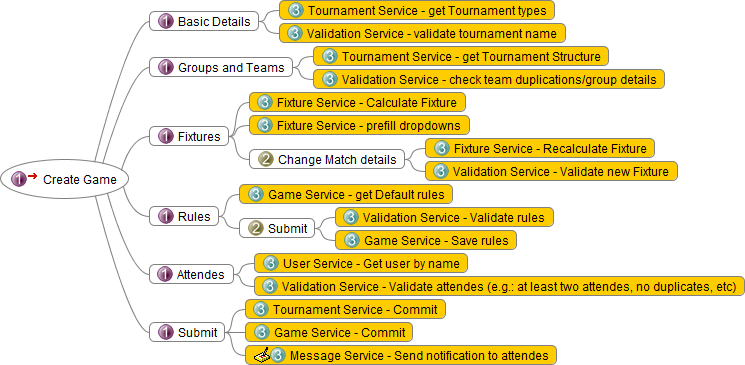
## Login/Registration



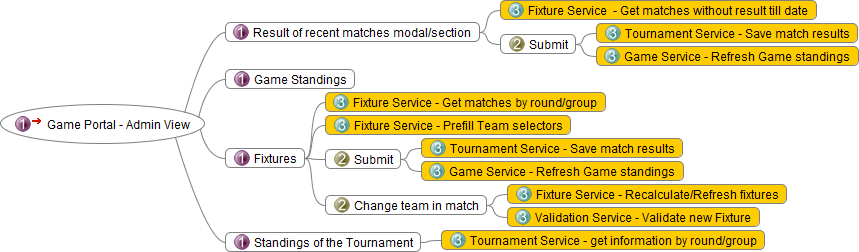
## User Home



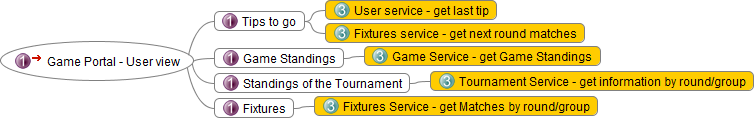
## Game creation



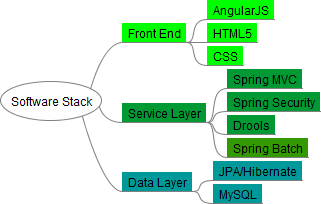
## Admin Portal



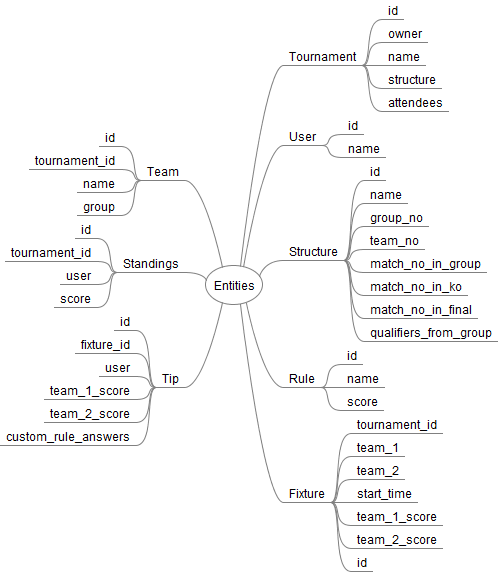
## User Portal



## Software Stack View



## Data View



The entity structure planned above can easily cause very huge tables. To avoid performance issues there will be an archive process configured and there will be views for the frequently shown but not so often modified data, e.g. the fixtures.

Components  
Users

The users has basically two roles. Admin or Player. Only one view makes different among users and that’s the Game portal. Because This is the only place where matters if the current user is the owner of the specific game or not.

Authentication

Authentication happens when the user provide his/her username and password and submit to login. A session will be created then and it lives until the user logs off.

## Error Handling

The errors what occurs in the back end will be logged. If an error comes up because a user interaction (for example the provided data from the user is invalid) then the user will be warned by an error message on the current page.

## Data Access layer

The reach of the Database will be handled by JPA Hibernate and the tools it provides. Entities, Entity Manager, Session Factory. Daos created for the entities will be available for the Service Layer components.

## Archiving

[TBD]