КУРСОВ ПРОЕКТ

Тема: Браузър игра

Изготвил:

*Александър Александров Вербовский*

***№ 471218008, 76 група***

СОФИЯ

2020

# ГЛАВА I:

**Проучване на проблемната област**

Браузър игрите се играят онлайн, чрез Интернет. Различават се от обикновените видео- и компютърни игри по това, че не изискват потребителят да ги инсталира като софтуер на своя компютър. Някои игри разчитат изцяло на технологии "от страна на клиента" като уеббраузър и обичаен [плъгин](https://bg.wikipedia.org/wiki/%D0%9F%D0%BB%D1%8A%D0%B3%D0%B8%D0%BD) от рода на Java или Flash, докато други използват [скриптове](https://bg.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) "от страна на сървъра". Последните обикновено са (масови) мултиплейър игри, докато игрите "от страна на клиента" са само за един играч. Често игрите, които се играят в браузър, се наричат и браузър-базирани игри (*browser-based games*).

Плъгин игрите разчитат на потрибителския браузър да изтегли и изпълни играта. Това позволява по-лесно модифициране на параметрите на играта и нечестна игра. Поради това повечето такива игри все още са предназначени за индивидуална игра. Много разпространени са онлайн флаш игрите, които доста често пресъздават някои от старите игри за Sega конзолата или игровите машини.

Съществуват редица сайтове които предлагат Браузърни игри или още познати като Флаш игри. Също така не са малко и известните онлайн браузър игри, създадени от българи. Такива са Imperia Onlien, Khan Wars, Lady Popular и много други.

# ГЛАВА II:

Цел и задачи на проекта

## Цел на проекта

1.1 Да се създаде мултиплейър онлайн браузър игра. Всеки потребител трябва да бъде свързан към обща база данни, така че всички промени, направени от конкретния потребител, да се отразяват и при останалите потребители.

1.2 Всеки потребител притежава отбор, който може да попълва с играчи. Играчите са общи за всички потребители и се различават по тяхната функция. Всяка вечер базата от данни се попълва с нови (случайно генерирани) играчи.

1.3 Всеки потребител може да се изправя в битка (да напада) отборите на другите потребители, като така печели допълнително средства (чрез които купува нови играчи) и точки (чрез които се изкачва в общото класиране на играчите).

## Задачи на проекта

2.1 Да се проучат и изберат технологии, съответстващи на целите на проекта

2.2 Да се проектира база данни, отговаряща на изискванията и функционалностите в проекта

2.3 Да се създаде примерна функционалност на WEB приложението (онлайн браузър играта)

2.4 Да се създаде, вече проектираната, база от данни, на базата на избраните технологии

2.5 Да се създаде модул, който да бъде връзката между базата данни и WEB приложението (Database Project)

2.6 Да се създаде WEB приложение, отговарящо на изискванията и примерната функционалност

2.7 Да се свържат WEB приложението, Database проектът и базата данни.

2.8 Да се тества завършената система

# 

# ГЛАВА III:

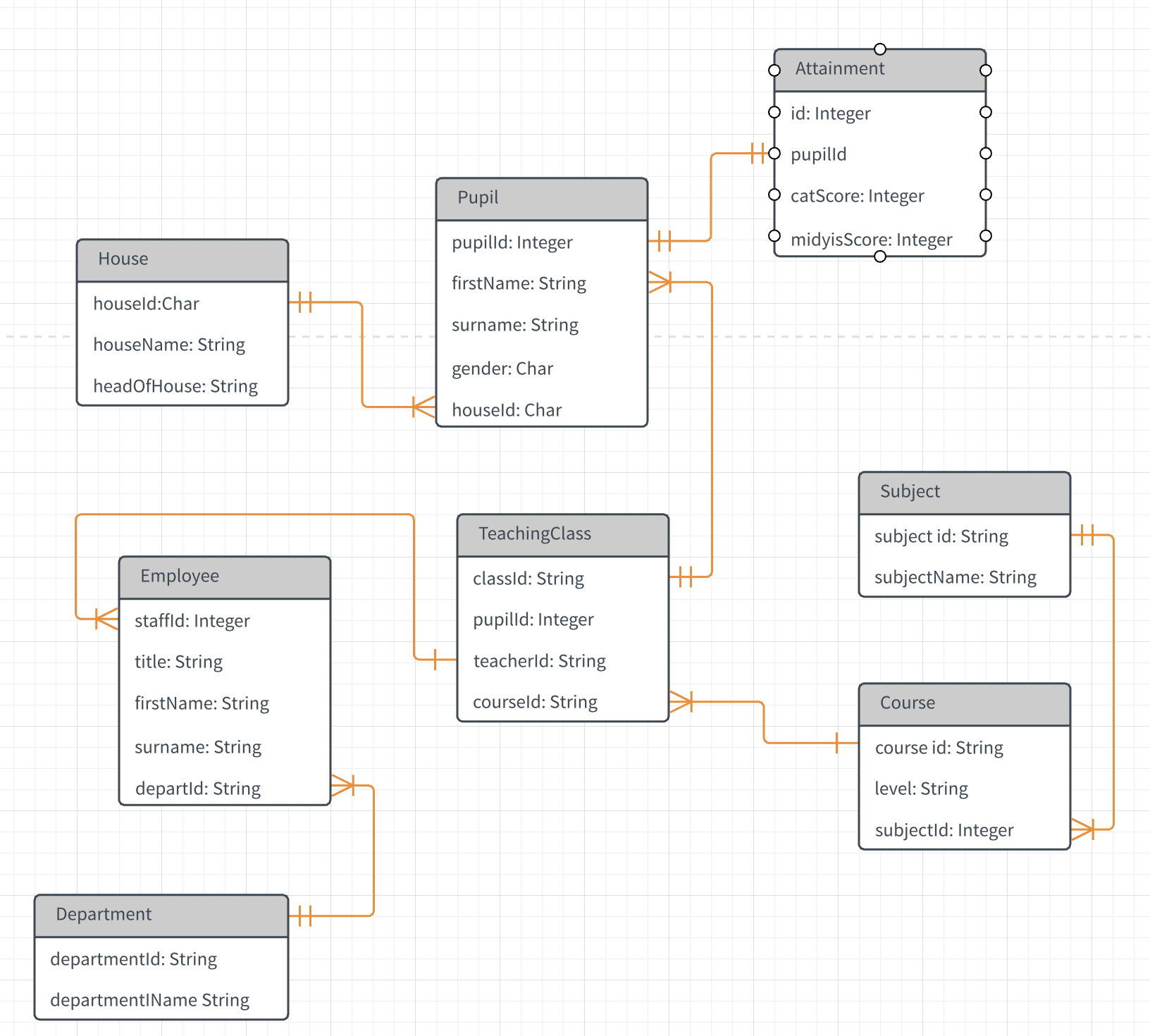
**Избор на технологии за реализация**

## База данни

**1.1 Релационни бази данни**

Базата данни представлява колекция от данни. Организацията и подредбата на данните може да следва определен модел. Според модела може да се опишат два типа бази данни - релационни и нерелационни.

Най-известният и използван до момента тип бази данни е релационният (SQL бази данни). Другият тип бази данни, използващ нерелационен модел за данните, се нарича NoSQL .



(**фиг. 3.1**) Релационни бази данни

Релационните бази съхраняват данните по предварително структуриран начин - в таблици, подредени в редове (записи) и колони. Между отделните данни и таблици може да се създават връзки (релации). Таблиците и връзките между тях образуват структура, която може да се представи като схема. Отделните единици информация се съдържат в единични полета, подредени в редове в таблица.

SQL базите се използват масово - при малки обеми информация например за уеб сайт с две страници, до големи уеб или мобилни приложения, блогове, онлайн магазини и други. Най-известните готови системи за управление на съдържанието (CMS) поддържат и използват релационни бази данни - WordPress, Joomla, Drupal, Magento и други. По-малко са тези обаче, които поддържат NoSQL бази данни (като Drupal).

**1.2 PostgreSQL**



(**фиг. 3.2**) PostgreSQL

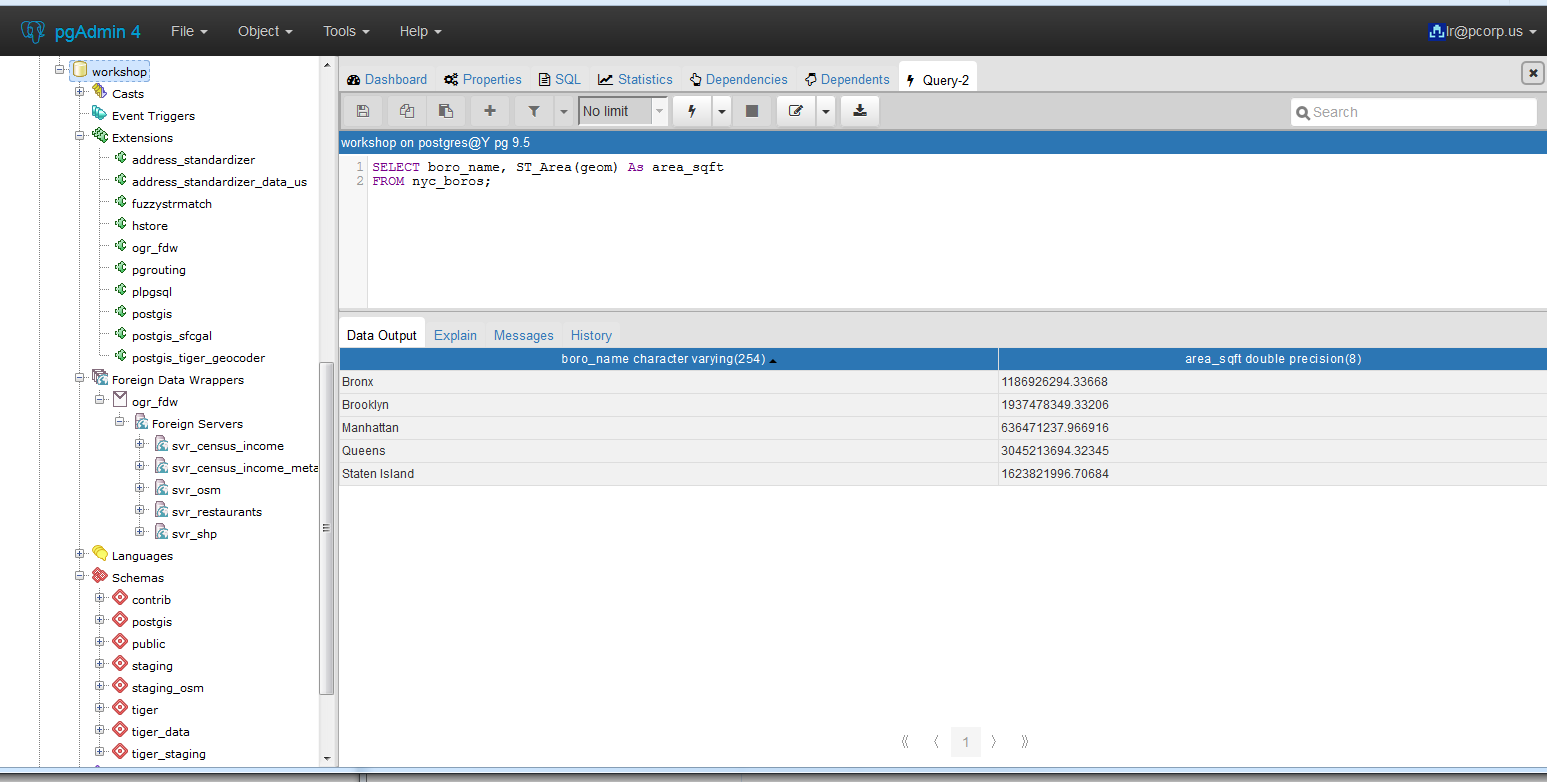
PostgreSQL е обектно-релационна база данни (ORDBMS) с отворен код. Поддържа се на всички известни операционни системи като Linux, Unix и Windows. Тя е една от водещите системи за управление на бази данни, редом с MySQL и SQLite.

PostgreSQL е често предпочитана от разработчиците и проектите, за които основен приоритет при боравенето с данните е надеждността, способностите за скалируемост и допълнителните възможности на базата данни.

PostgreSQL има по-усложнен, но усъвършенстван начин на работа с базите данни, като в същото време достига другите системи по леснота на управлението и администрацията.

Характеристики:

* транзакционен модел с пълно ACID изпълнение и MVCC (multiversion concurrency control) контрол при едновременно писане и четене на данните);
* способност за разширяване от потребителя (чрез добавяне на нови типове за данни, функции, оператори, методи за индексиране, процедурни езици и други);
* различни методи за индексиране (не всички от които се предлагат в други системи за бази данни);
* използва един единствен storage engine (PostgreSQL);
* комплексни заявки;



(**фиг. 3.3**) Работа с PostgreSQL

**1.3 DBeaver**



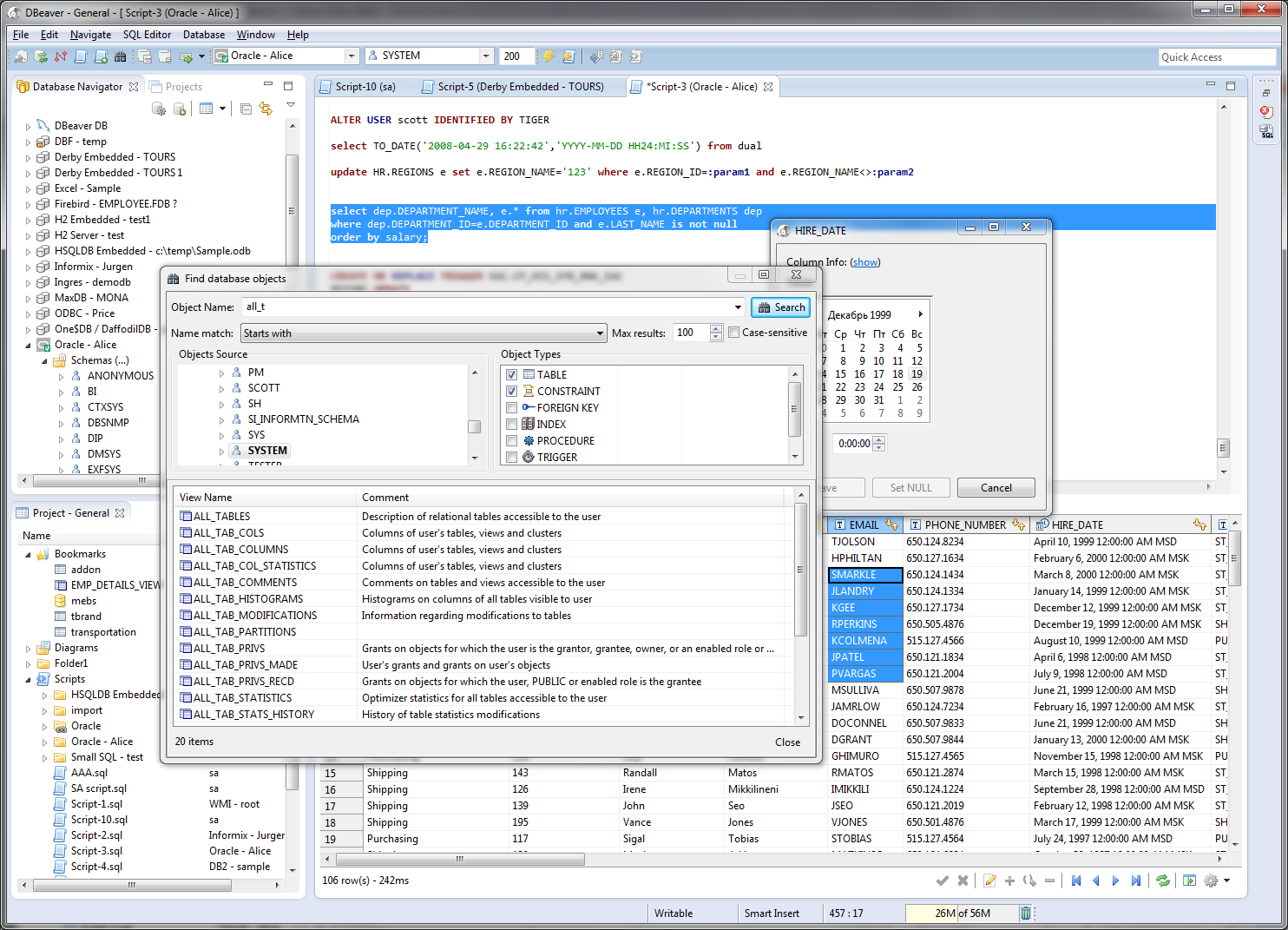
(**фиг. 3.4**) DBeaver

**DBeaver** е отворен, многоплатформен и напълно свободен графичен софтуер, реализиран в Java / Eclipse и проектиран от самото начало, за да действа като SQL клиент и универсален мениджър на базата данни за операционни системи GNU / Linux.

Сред акцентите му можем да споменем редактор на метаданни, изпълнение на SQL изявления, скриптове, управление на транзакции, скролируеми резултати, управление на SQL скриптове, ER диаграми, както и модерен и внимателно проектиран потребителски интерфейс.

Приложението се основава на рамка с отворен код и разполага с мощна архитектура на приставки, която позволява на потребителя лесно да разшири своята функционалност по подразбиране, като създава разширения / приставки.

Приложението поддържа широка гама от сървъри / двигатели за бази данни, включително MySQL, Oracle, PostgreSQL, IBM DB2, Microsoft SQL Server, Sybase, ODBC, Java DB, Derby, Firebird, HSQLDB, SQLite, H2, IBM Informix, SAP MAX DB, Cache, Ingres, Linter, Teradata, Vertica, както и всеки източник на данни, съвместим с JDBC.



(**фиг. 3.5**) DBeaver UI

## WEB проект

**2.1 Eclipse**

****

(**фиг. 3.6**) Eclipse

Eclipse е среда за разработване на софтуер, която се състои от IDE (интегрирана среда за разработка) и от plugin система. Тази среда е написана основно на Java и също така е най-често използвана за Java приложения.  
 Еclipse предоставя на потребителите си възможността да инсталират или да създадат свои плъгини и така да разширят функционалността на средата. Всеки плъгин се интегрира в Eclipse средата по един и същ начин. Всичко в Eclipse, с малки изключения, са плъгини. Част от тях предоставят на потребителя да разработва и приложения на други езици.

Eclipse е безплатен и с отворен код. Появява се през 2001 година, а от 2004 за разработката и поддръжката му се грижи Eclipse Foundation – организация, част от която са част от най-големите технологични компании (IBM, Oracle и т.н).

**2.2 Java**

****

(**фиг. 3.7**) JavaI

За език за програмиране беше избран Java, който е обектно-ориентиран език за програмиране появил се през 1995г. За да се изпълни дадена програма, написана на Java, е необходима виртуална машина JVM (Java Virtual Machine). Това се налага поради факта, че Java кодът не се компилира до машинен код, а до т.н. байт код – специфичен за езика код. Тази реализация има много предимства, които значително надвишават недостатъците.   
 Някой от предимствата на Java са лесната ѝ преносимост между различните платформи, защото веднъж компилирана, една програма може да бъде стартирана на всяко устройство с виртуална машина, независимо от архитектурата или операционната му система. Тази виртуална машина осигурява и други предимства като garbage collector (освобождаване на паметта от обекти, които вече не се използват и към тях няма повече референции), heap (предварително заделяне на динамична памет), възможност за контрол на правата на потребителя на ниво виртуална машина, висока степен на сигурност (няма директен достъп до паметта) и т.н.   
 Основният недостатък на езика е необходимостта от допълнителни ресурси, като процесорно време и памет, за изпълнението на самата виртуална машина, както и по-ниската производителност.  
Java е език от високо ниво и всичко в него се дефинира в класове, които са основна концепция на езика. Класовете имат свойства и методи (определящи поведението на един клас). Тук не съществуват глобални променливи.

Всички тези предимства на езика, както и голямата общност, използваща го, прави Java едно от най-добрите решения за разработването на приложения.

**2.3 ZK Framework**

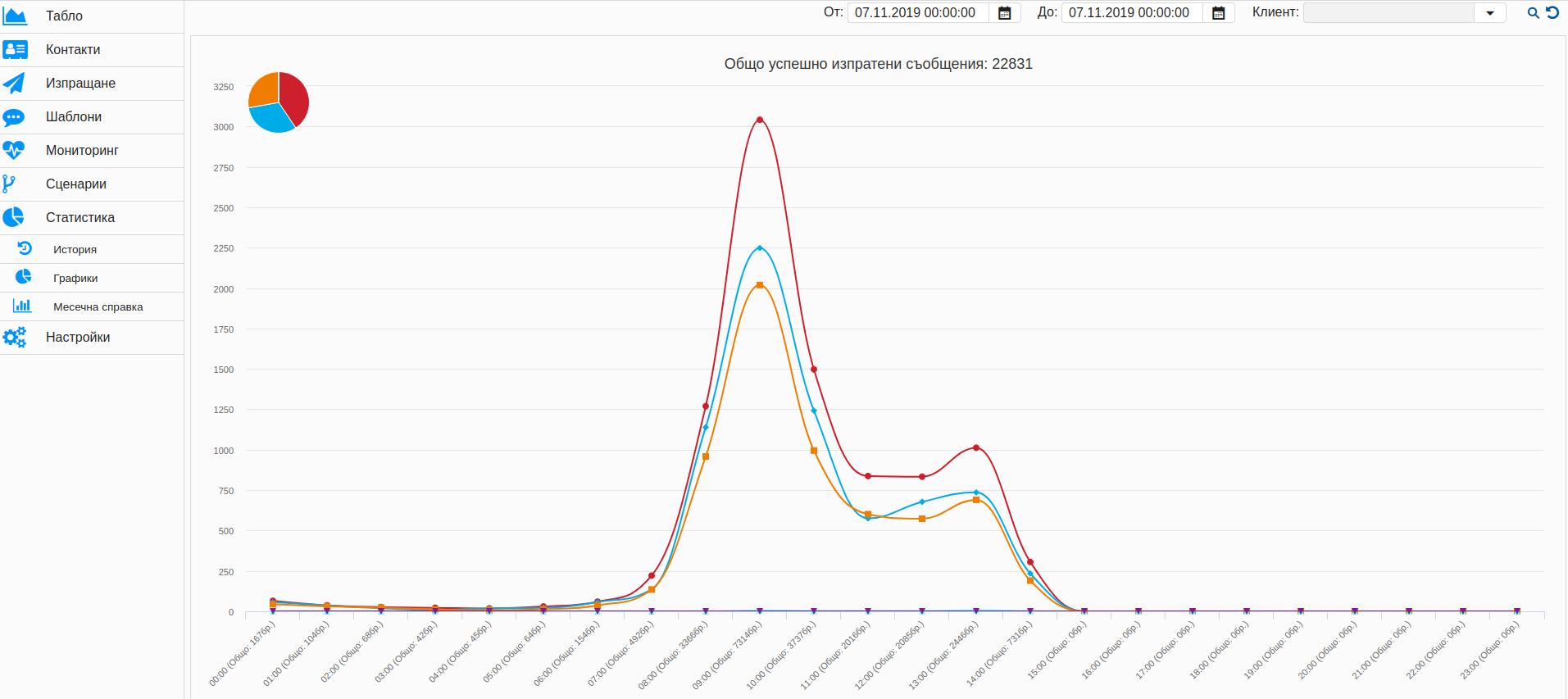
****

(**фиг. 3.8**) ZK Framework

ZK представлява AJAX базирана, технологична рамка с отворен код, разработена на Java. Тя подпомага създаването на front-end приложения.

Чрез ZK могат да се създават приложения с минимални умения в програмирането и без да е нужно изучаването на JavaScript и AJAX.

ZK предоставя множество готови компоненти, което прави създаването на потребителско изживяване много лесно. Стилът, поведението и функционалността на всеки компонент може да бъде конфигуриран спрямо специфичните изисквания на разработчика.

****

(**фиг. 3.9**) ZK Project

**2.4 Spring**

****

(**фиг. 3.10**) Spring Framework

Spring представлява технологична рамка с отворен код, която предоставя множество функции, улесняващи разработването на Java EE (Enterpise Edition) приложения. Появява се през 2002г. и включва различни модули, предоставящи голям брой различни функционалности, като управление на транзакциите, аспектно-ориентирано програмиране, контейнер на зависимостите, достъп до данни и др.

## Архитектура на приложението

Шаблонът Model-View-ViewModel (MVVM) може да се използва при всички XAML платформи. Неговата цел е да се осигури чисто разделяне на връзките между контролите от потребителския интерфейс и тяхната логика. В MVVM шаблона има 3 основни компонента: model (модел), view (изглед) и view model (изглед-модел). Всеки от тях изпълнява различна и отделна роля.

*Компонентите са отделени един от друг, което от своя страна позволява:*

* Размяна/подмяна на компоненти
* Вътрешната имплементация на компонент да бъде променена без това да засегне останалите
* Работата по компонентите да се извършва независимо
* Индивидуално тестване на компонентите

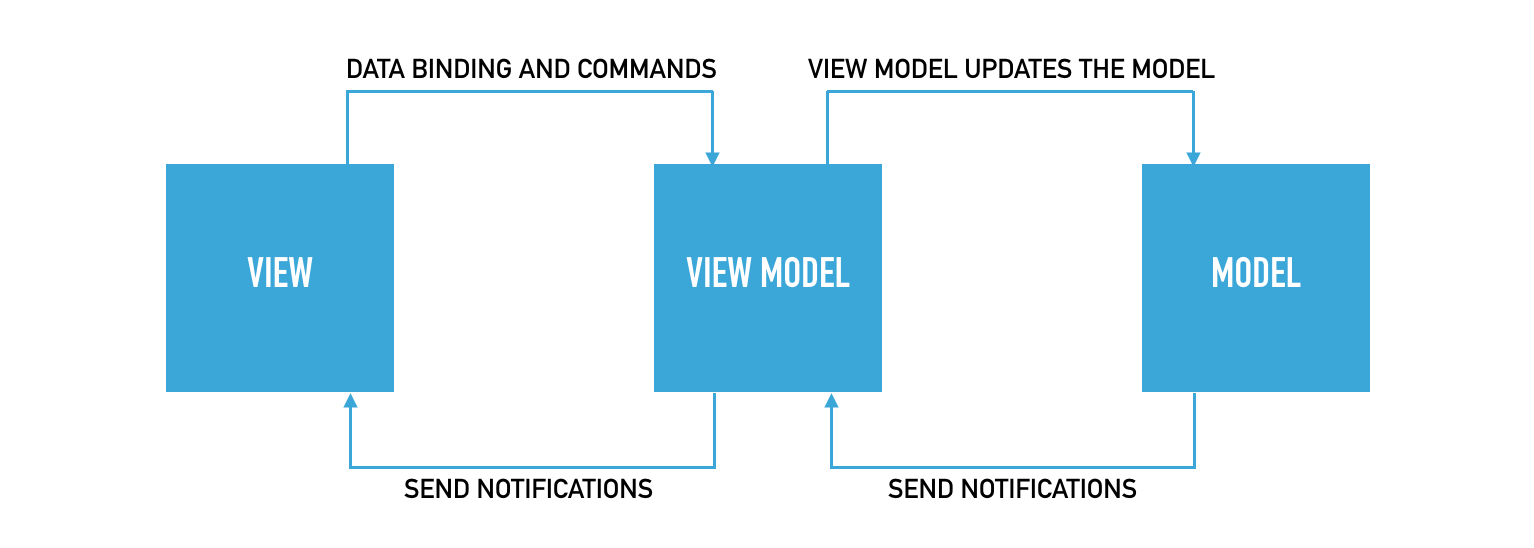
*Основната мотивация за имплементирането на приложение, използвайки MVVM шаблона, е:*

1. Предвижда чисто разделение между логиката на приложението и на потребителския интерфейс. Това прави приложението по-лесно за тестване, поддържа и развитие. Той подобрява възможностите за преизползване на код.

2. Това е естествения модел за XAML платформи. Благодарение на широките възможности за data binding и dependency свойствата, се осигуряват средства за свързване на UI към view model.

3. Позволява разделението на работата между разработчика и дизайнер.

4. Отделянето на логиката в отделни класове, позволява по-лесното ѝ тестване.



(**фиг. 3.11**) MVVM Pattern

# 

# 

# ГЛАВА IV:

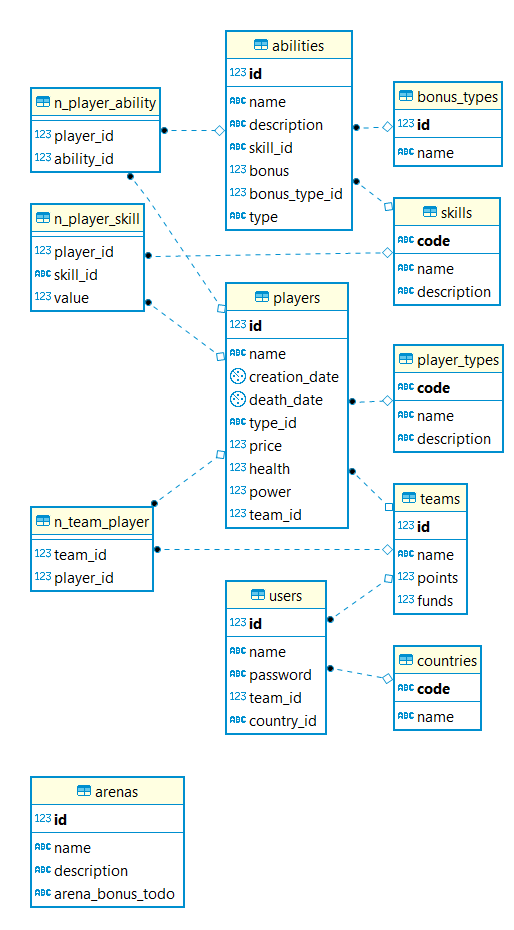
Програмна реализация

## 

## Модели в проекта

Проектът съдържа 9 модела:

* User (Потребител) - съдържа потребителското име, паролата, *Отборът* и *Държавата* на всеки регистриран потребител.
* Team (Отбор) - всеки *Потребител* притежава *Отбор*. Отборът има име, точки и средства.
* Player (Играч) - всеки *Отбор* има по 5 *Играча*. Играчът има име, дати на създаване и смърт, *Тип*, цена, живот, сила и връзка към *Отбор*.
* Player Type (Тип играч) - това е типът играч. Типовете са предефинирани от началото на играта и не могат да се променят. Има 5 типа - **D01**, **T01**, **T012**, **T02**, **T013**. Моделът съдържа име и описание на типа.
* Ability (Способност) - показва способностите, които всеки играч притежава. Те му носят различни бонуси в битка. Моделът съдържа име, описание, *Умение*, бонус, *Тип бонус*, тип на способността (бонус в атака или в защита).
* Skill (Умение) - показва умението, което носи бонус *(напр. Двама играчи имат едно и също Умение* *- “Точен изстрел”, но то носи различен бонус на всеки един от тях)*. Съдържа име и описание.
* Bonus Type (Тип бонус) - показва какъв точно е бонусът за дадена *Способност* - процент или абсолютна стойност.
* Arena (Арена) - *Арената* е мястото, на което се провеждат битките. Тя се избира на случаен принцип при започване на битка. Съдържа име и описание.
* Country (Държава) - *Държавата*, от която е даден *Потребител*. Съдържа трибуквен код и име.

****

(**фиг. 4.1**) Диаграма на базата данни

## VM и .zul

В шаблона Model-View-ViewModel (MVVM) - *ViewModel* превръща информацията, идваща от *Model*, в стойности, които могат да се визуализират във *View* компонента. Всяка функционалност в проекта съдържа по един *ViewModel*, който е свързан със *.zul* файл, който играе ролята на *View*.

Във *ViewModel* класа има *Init* метод, който се изпълнява при инициализация на дадения клас. В него могат да се добавят и методи анотирани с анотацията *@Command*, които се извикват от *.zul* файла. Чрез тях може да се извърши връзката между *View* и *ViewModel*.

В проекта има един основен *ViewModel* клас - *IndexVM*. Този клас се инициализира при влизане в играта и е активен до затваряне на страницата. Върху *.zul* файла на този клас - *index.zul* се визуализират всички страници, които потребителят може да извика впоследствие.



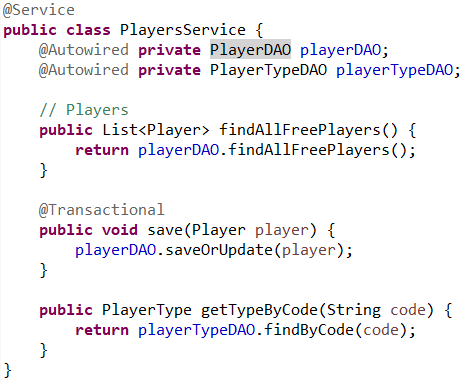
(**фиг. 4.2**) Пример за ViewModel клас



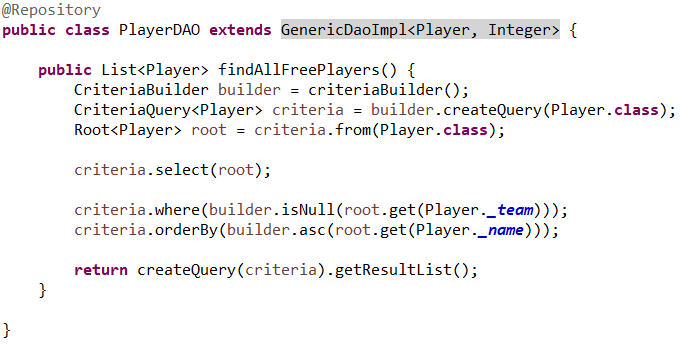
(**фиг. 4.3**) Пример за .zul файл

## Services

Всеки *ViewModel* клас извиква определен брой *Service* класове. В тях се съдържат различни методи за запис, търсене, изтриване (*и др.*) на даден модел. От своя страна, тези класове извикват методи в *DAO* (*Data Access Object*) класове, които са директната връзка с на обекта с базата.



(**фиг. 4.4**) PlayersService клас



(**фиг. 4.5**) PlayerDAO клас

## Битка

Основна функционалност на проекта е битката. В меню “*Класиране*”, потребителят може да избере да нападне отбор на друг потребител. Ако даден отбор няма пълен брой играчи или потребителят избере да нападне себе си, то на екрана ще се изскочи прозорец за грешка (с различно съобщение). В случай, че грешка няма, то потребителят преминава в екрана за битка.

При стартиране на битка, информацията за двамата играчи се изпраща на класа *BattleUtil*, който започва генерирането на битката.

Първоначално се извежда информация за отборите, часа на начало и мястото на провеждане на битката. След което започват рундовете, в които играчи на двата отбора нападат противниковите играчи. Класът избира случаен играч от единия отбор, който напада случаен играч от другия отбор. При атака има две възможности - попадение и блокиран удар от страна на противника.

Битката продължава до момента в който даден отбор няма останали живи играчи. В този момент на екрана се извежда отборът победител, който печели точки и средства от битката.

## Utilities

В проекта има 4 “utility” класа, които спомагат различни функционалности в проекта:

* ***BattleUtil***- подават му се параметри като арена и два отбора, а този клас генерира цялата битка, рунд по рунд.
* ***PlayersUtil*** - генерира играч със случайно име, точки атака и защита и тип.   
   Името се генерира на базата на предварително зададени множества от имена и фамилии.  
   Точките се генерират като се взема случайна стойност между **0** и **предварително конфигурирана максимална стойност** в *Config* файла.
* ***ZulLocator*** - на база на подадени параметри - зарежда желан компонент (*.zul* файл).
* ***ZulUtil*** - спомагателен клас, с методи, подпомагащи функционалностите в *.zul* файловете (*напр. на база на подаден String параметър - визуализира икона на екрана*).



(**фиг. 4.6**) PlayersUtil клас



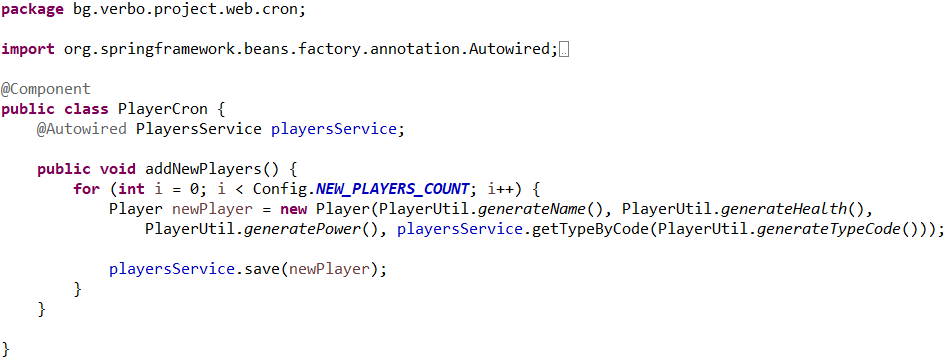
(**фиг. 4.7**) ZulLocator клас

## Cron

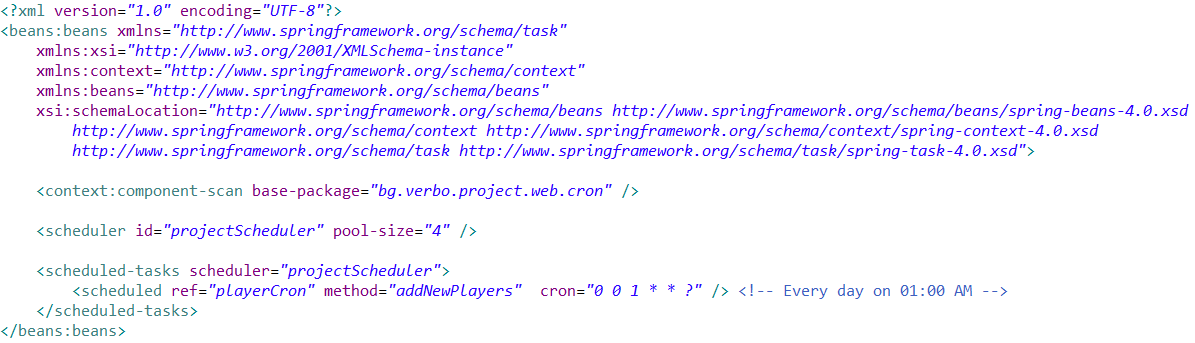
Spring позволява създаването на “планирани” методи, които се извикват през определено време. Благодарение на тях, в проекта е добавена функционалност за добавяне на играчи всяка вечер. Броят играчи, които да се добавят се конфигурира през *Config.java* файла.

Всяка вечер (*в 01:00 през нощта*) се извиква методът *addNewPlayers()* в класа *PlayersCron.java*. Чрез този метод се извиква *PlayerUtil* класът, който генерира играч със случайно име и други параметри.

Този *cron* е дефиниран в cron.xml файла, който пък от своя страна е добавен в *applicationContext.xml*.



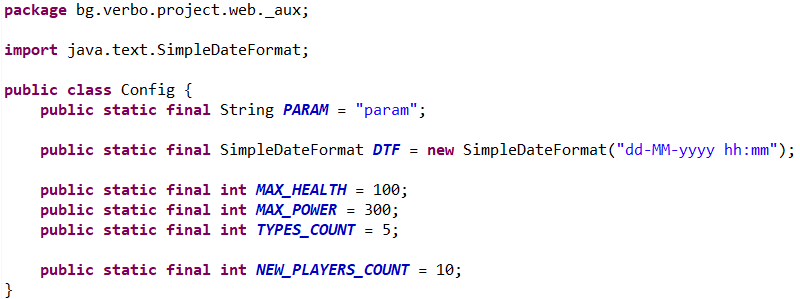
(**фиг. 4.8**) PlayerCron клас



(**фиг. 4.9**) cron.xml файл

## Конфигурационен файл

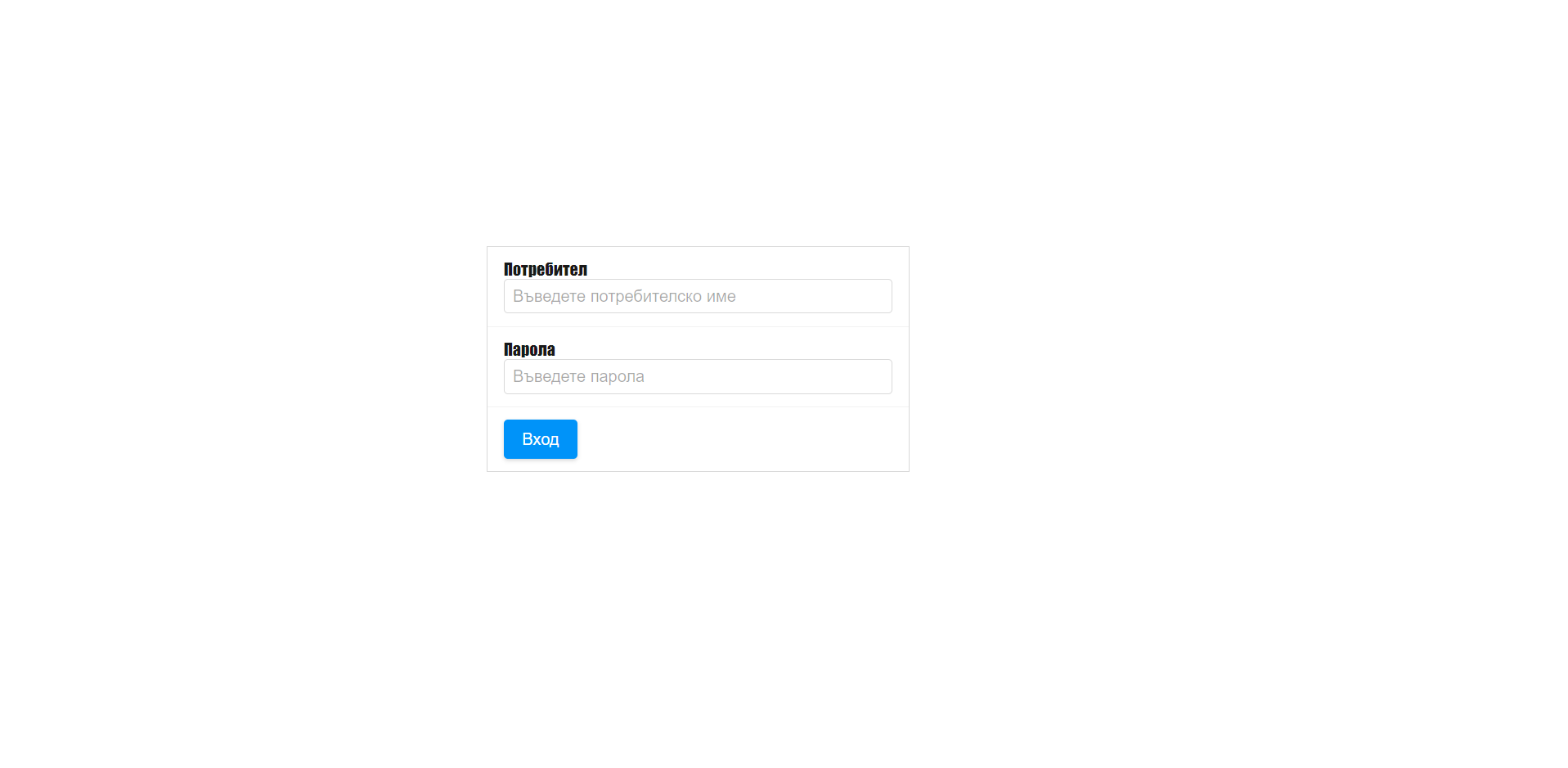
За да се избегне преписването на ръка на едни и същи данни - е добавен конфигурационен файл - Config.java, в който се съхраняват всички константи в проекта. Благодарение на този файл - при рефакториране на дадена константа, тя ще се промени автоматично във всички файлове, които я използват. Така могат да се избегнат множество грешки.



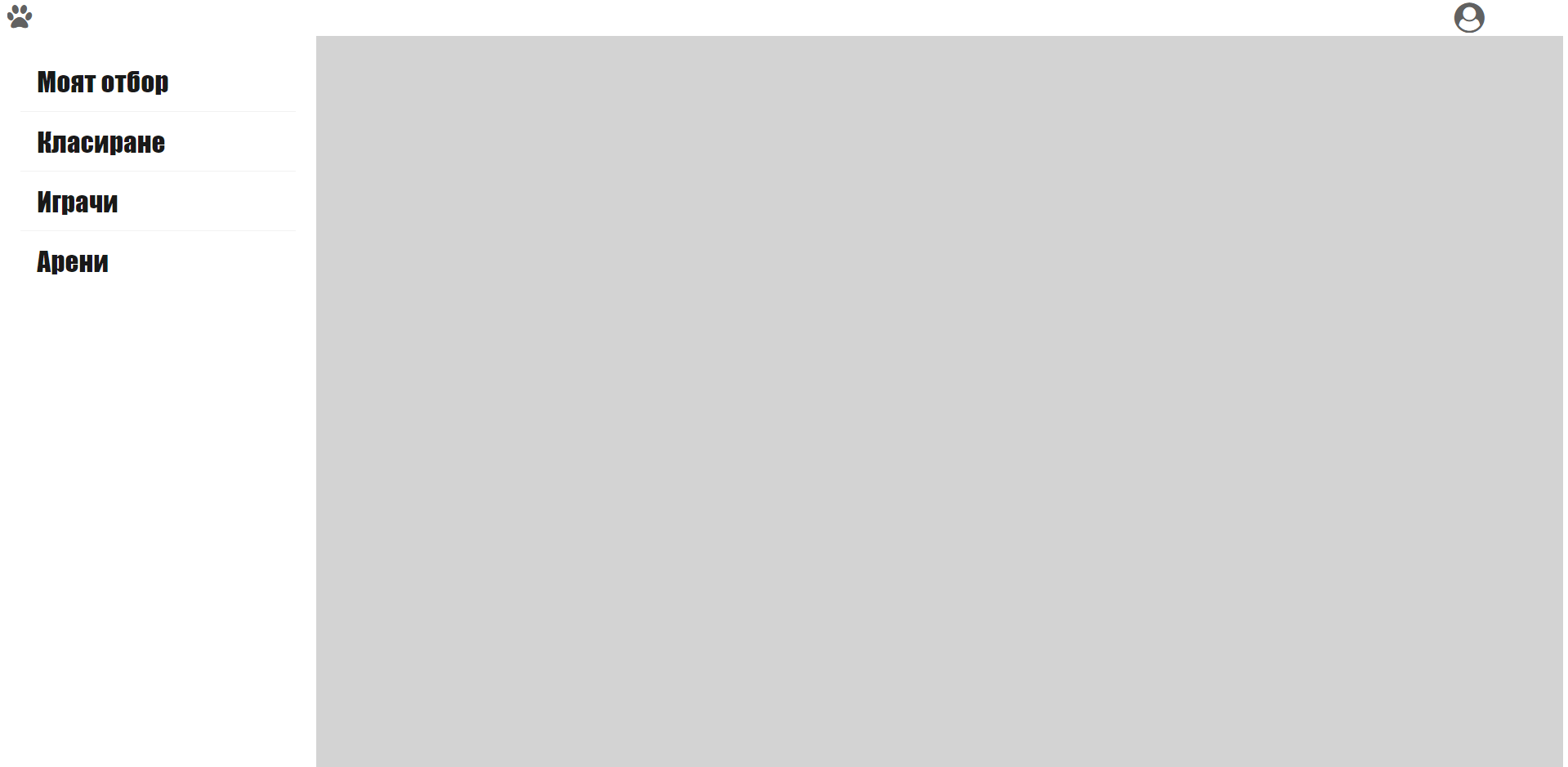
(**фиг. 4.10**) Конфигурационен файл

# ГЛАВА V:

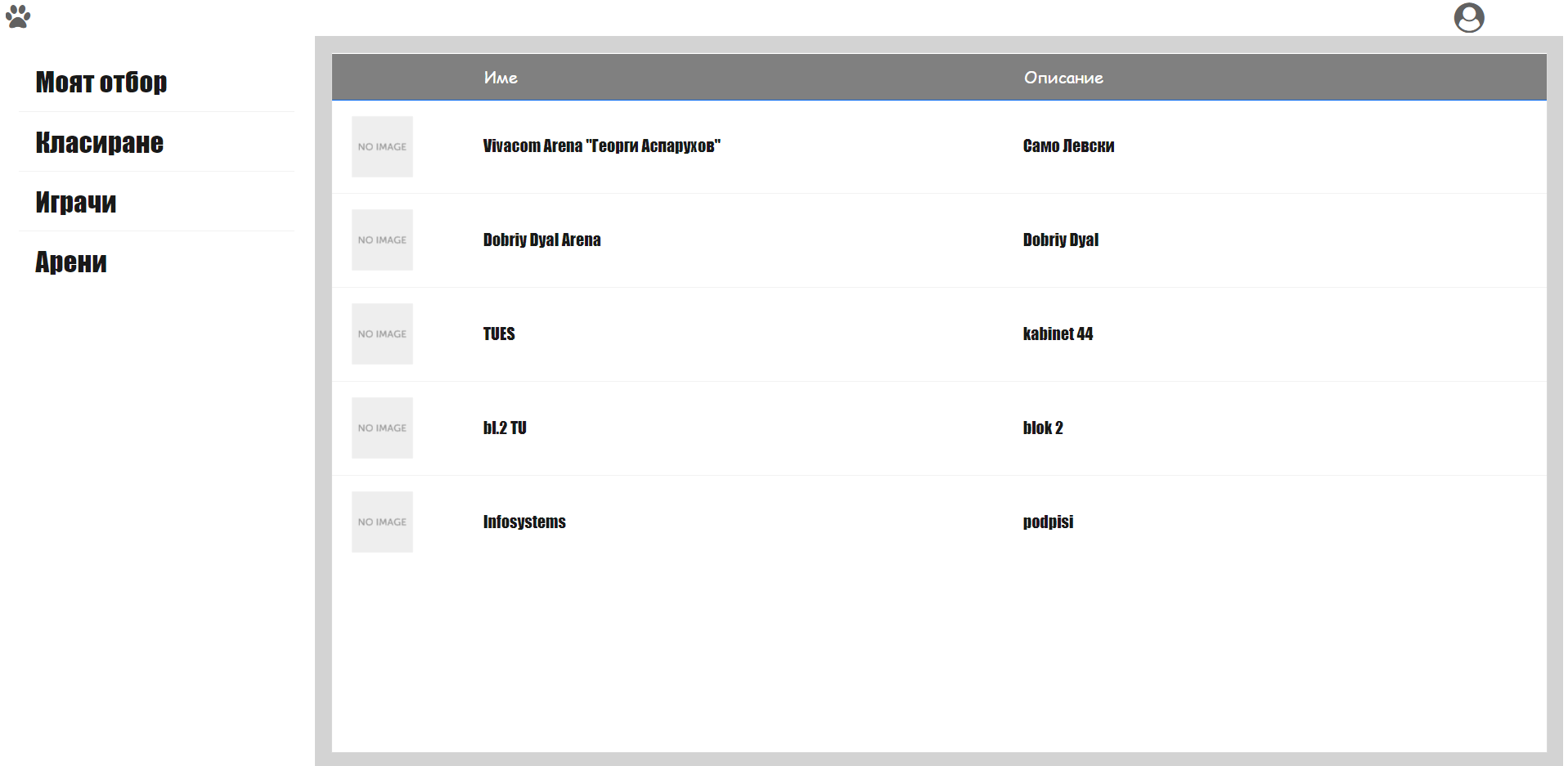
Експериментални резултати



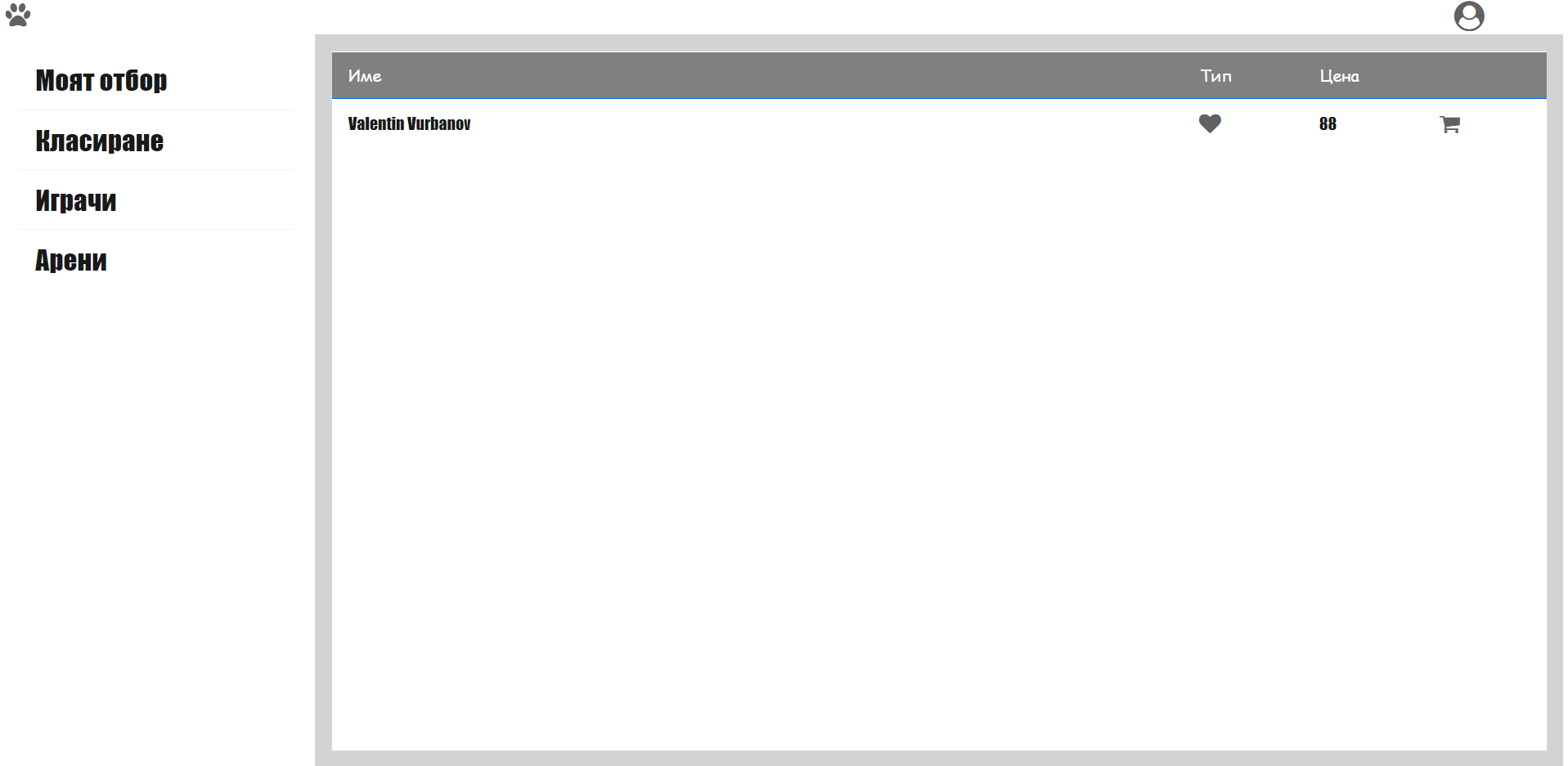
(**фиг. 5.1**) Login form



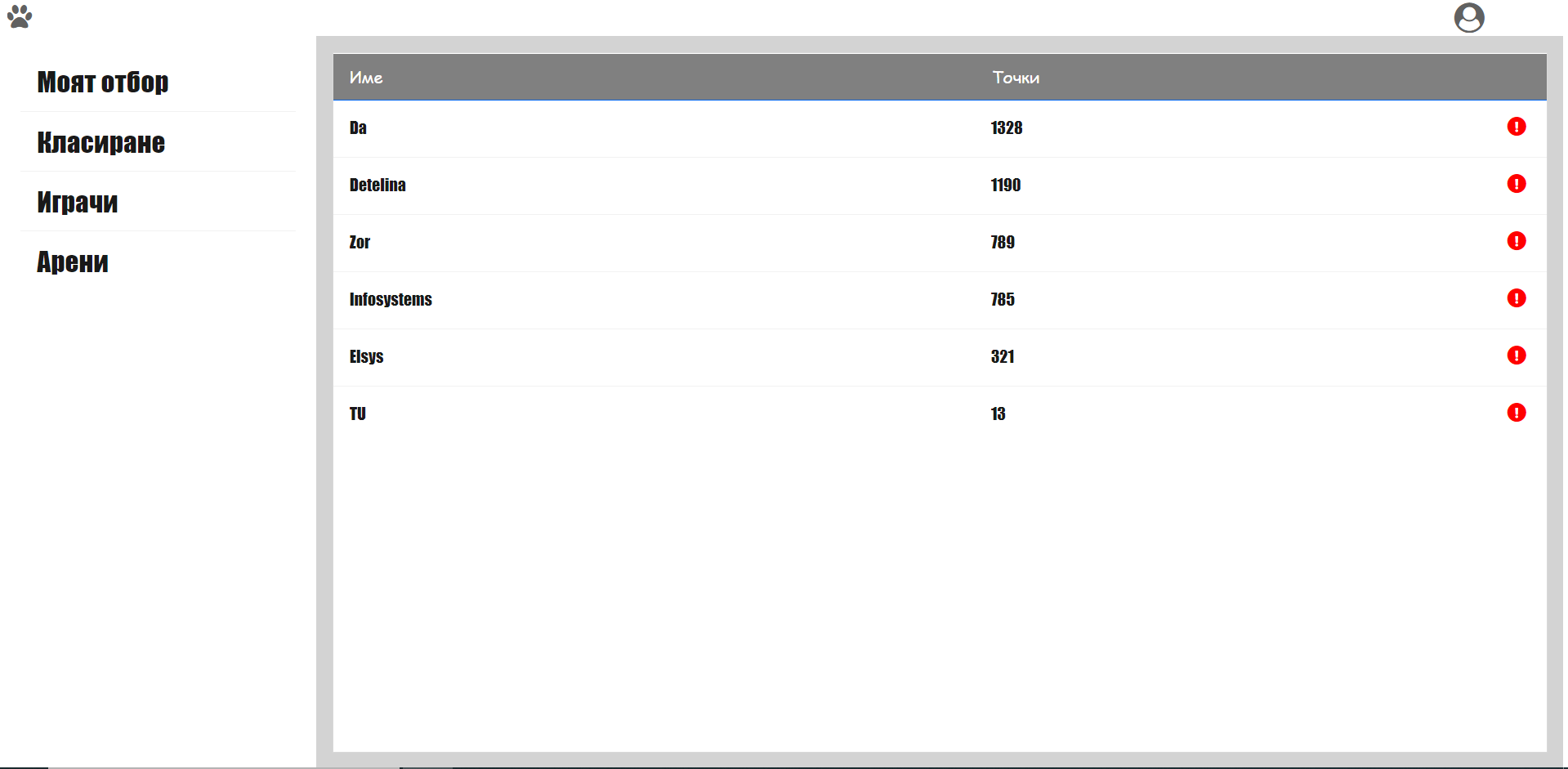
(**фиг. 5.2**) Main Menu



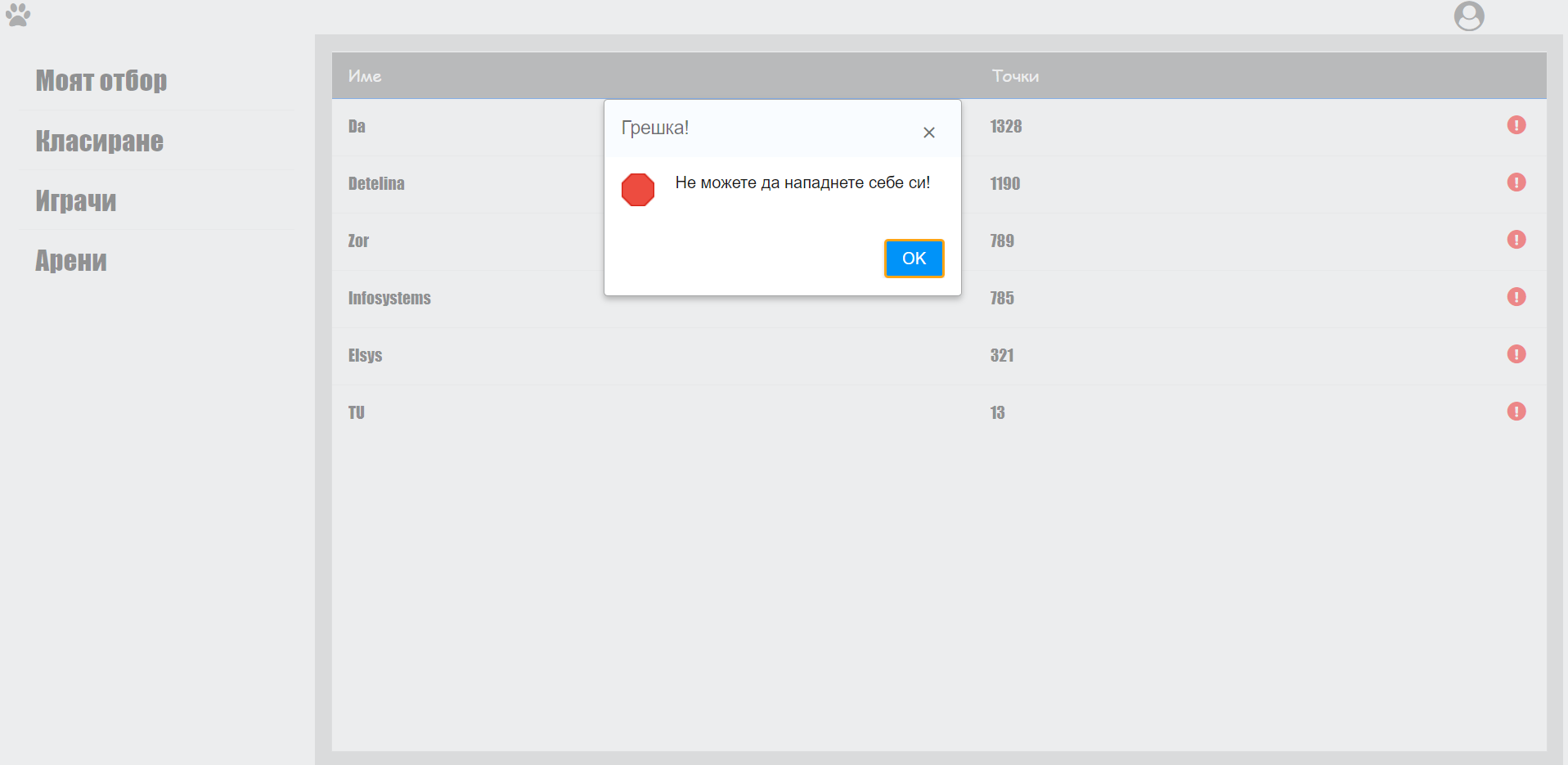
(**фиг. 5.3**) Arenas Menu



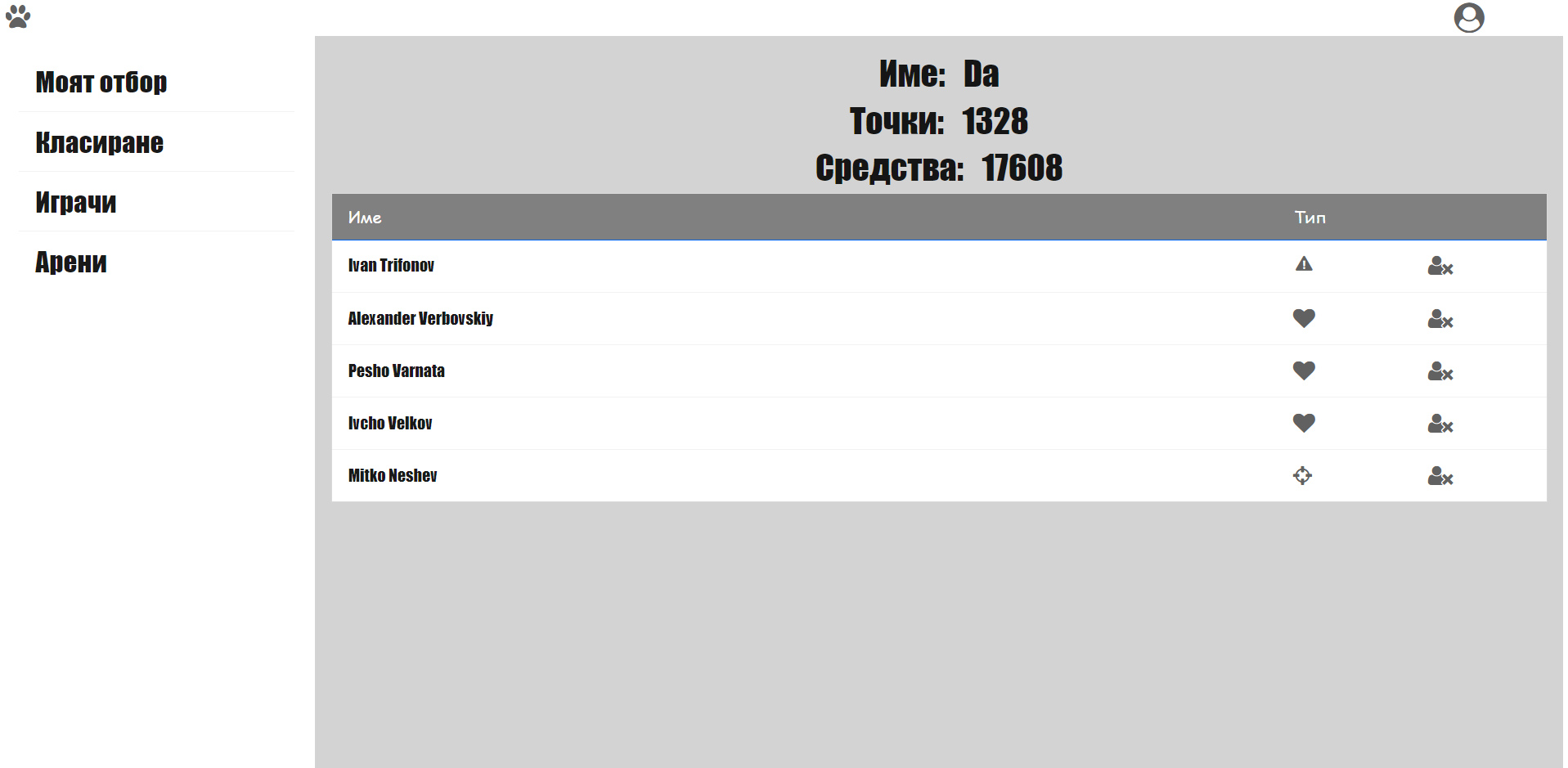
(**фиг. 5.4**) Players Menu



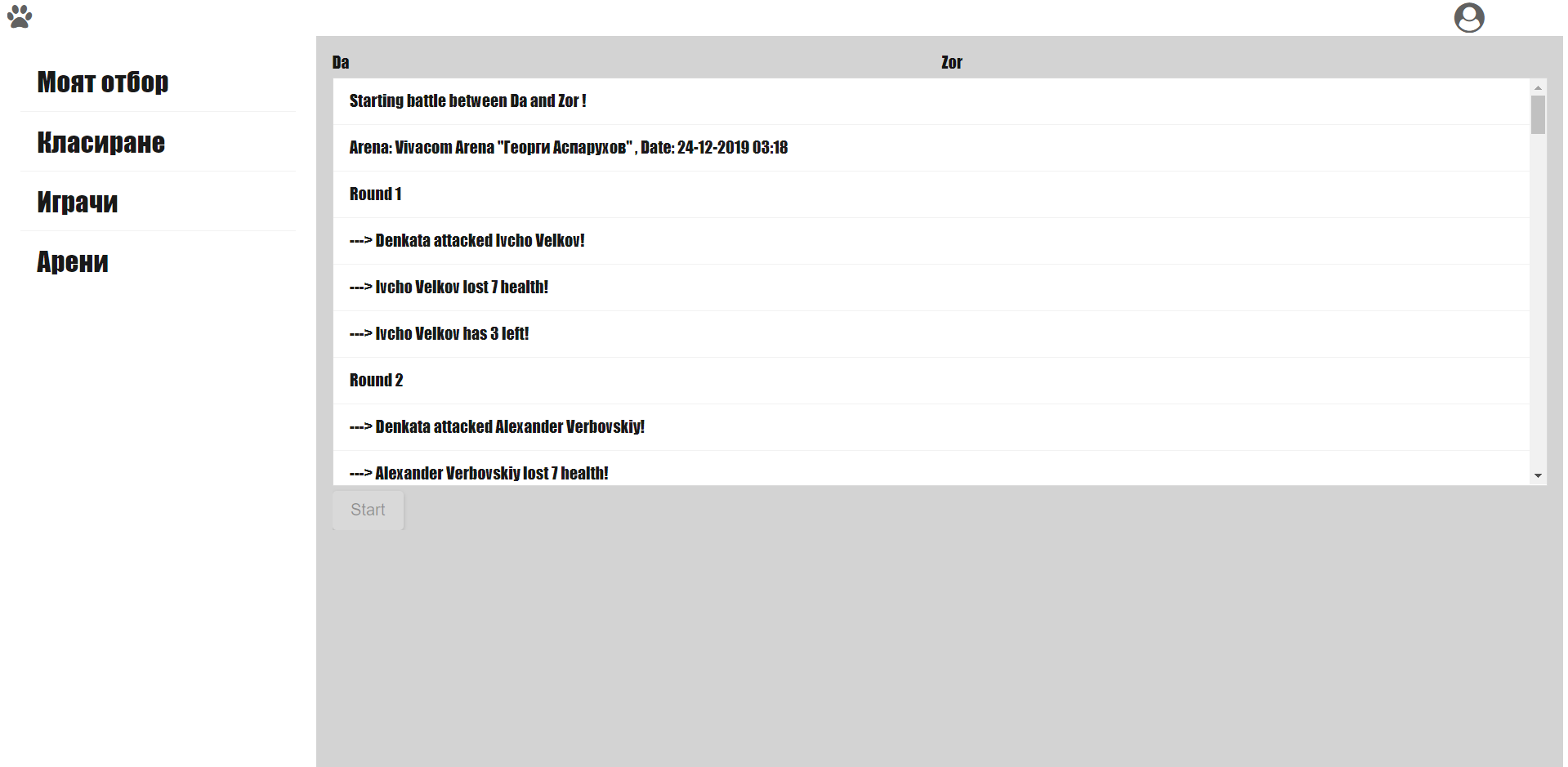
(**фиг. 5.5**) Ranking Menu



(**фиг. 5.6**) Validation Message



(**фиг. 5.7**) My Team Menu



(**фиг. 5.8**) Battle Visualization

# ИЗТОЧНИЦИ

1. <http://www.tasheva.info/PS/Upr3_PS_2016.pdf>
2. RefPRO (Система за футболно съдийство и оценка на съдиите) 2018г. - Александър Вербовский - ТУЕС към ТУ-София
3. <https://dbeaver.io/>
4. <https://www.postgresql.org/>
5. <https://www.zkoss.org/>
6. <https://spring.io/>

# СЪДЪРЖАНИЕ

[ГЛАВА I:](#_xn35ta6la4oy)

[ГЛАВА II:](#_7905429wryux)

[Цел на проекта](#_e2ddutt8cytt)

[Задачи на проекта](#_ni6ih880tdbc)

[ГЛАВА III:](#_1kqjzsdpvtnc)

[База данни](#_uluqk8880r55)

[WEB проект](#_m1h1fce81wr6)

[Архитектура на приложението](#_ox1xav32l1uo)

[ГЛАВА IV:](#_rejw40p66ru2)

[Модели в проекта](#_asv7nvq42n16)

[VM и .zul](#_45rpf15s4vmy)

[Services](#_dgss0o67dmx5)

[Битка](#_2ysqy4og05da)

[Utilities](#_s5ga5c3x6ocp)

[Cron](#_arv53796kjv6)

[Конфигурационен файл](#_yrpc8ynzzl5x)

[ГЛАВА V:](#_vz1m0f4kzt9h)

[ИЗТОЧНИЦИ](#_baet6gygfweh)

[СЪДЪРЖАНИЕ](#_uxoyvzkuw88y)

# ПРИЛОЖЕНИЕ

## Линк към GitHub repository:

<https://github.com/Verbo1806/ZK-Browser-game-Java-course-project->