Assignment 2

Student: Danila Lucia-Diana
Group:30433

Table of Contents

1. Requirements Analysis

- **Assignment Specification**

Use JAVA/C# API to design and implement an application for a ping-pong association that organizes tournaments on a regular basis. Every tournament has a name and exactly 8 players (and thus 7 matches). A match is played best 3 of 5 games. For each game, the first player to reach 11 points wins that game, however a game must be won by at least a two point margin.

The application should have two types of users: a regular user represented by the player and an administrator user. Both kinds of uses have to provide an email and a password in order to access the application.

- **Functional Requirements**

The regular user can perform the following operations:

- View Tournaments
- Enroll
- Search Tournaments

- Update the score of their current game. (They may update the score only if they are one of the two players in the game. The system detects when games and matches are won)

The administrator user can perform the following operations:

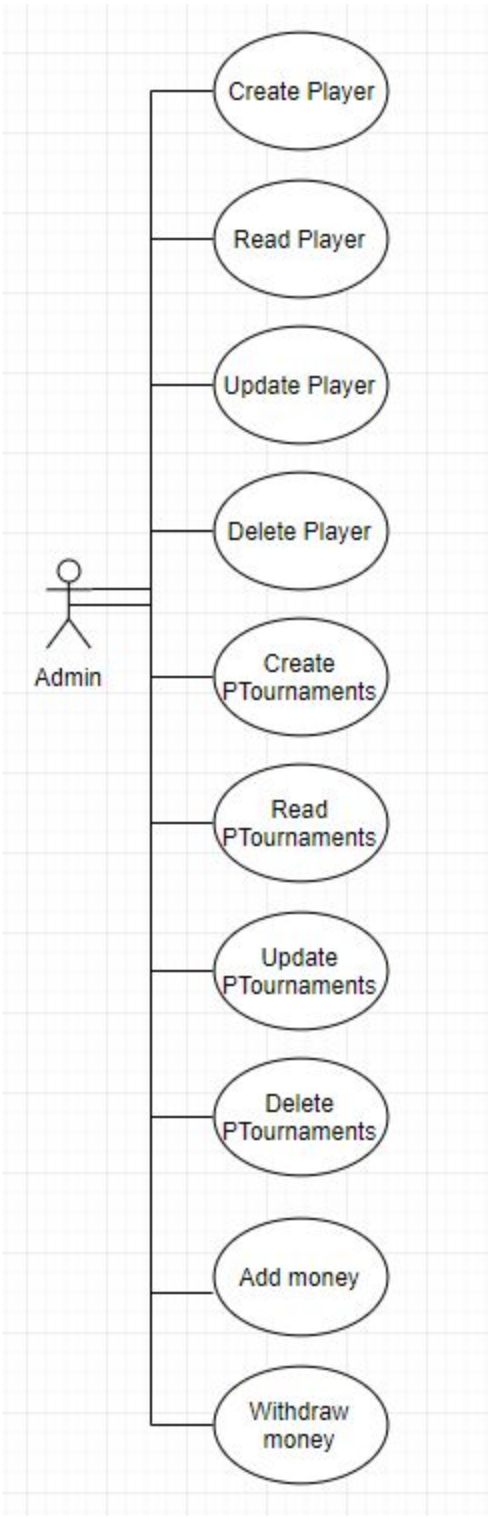
- CRUD on paid tournaments
- Add money to account
- withdarw money

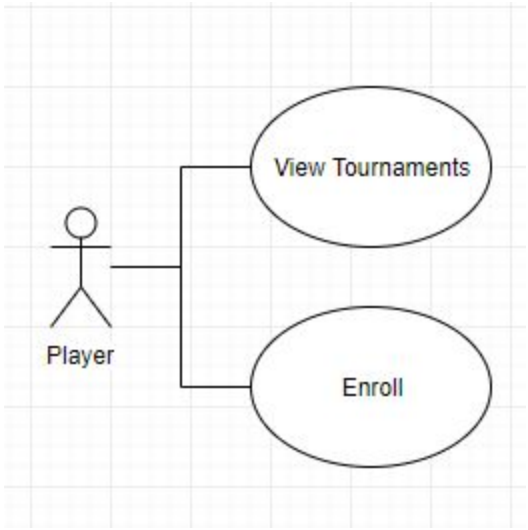
- **Non-functional Requirements**

The data will be stored in a database. Use the MVC pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application.

All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.

2. Use-Case Model


Create Player
Read Player
Update Player
Delete Player
Admin
Create PTournaments
Read PTournaments
Update PTournaments
Delete PTournaments
Add money
Withdraw money



Use case: delete player
Level: user-goal level
Primary actor: admin
Main success scenario: admin logs in -> views the player table -> choses the id of the player to be deleted -> insert id -> player deleted
Extensions: failure -> to log in or invalid id for player to be deleted

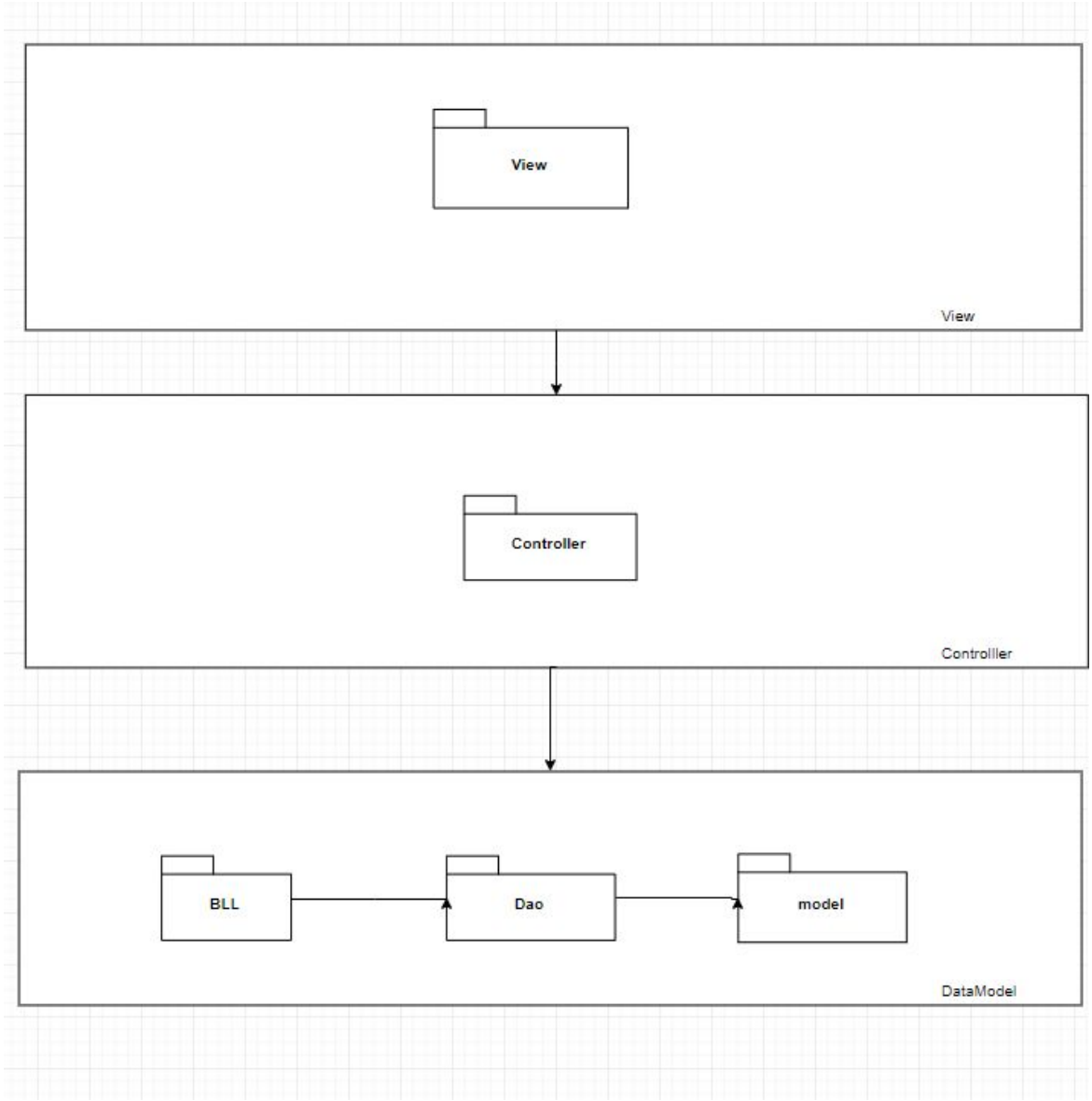
3. System Architectural Design

3.1 Architectural Pattern Description

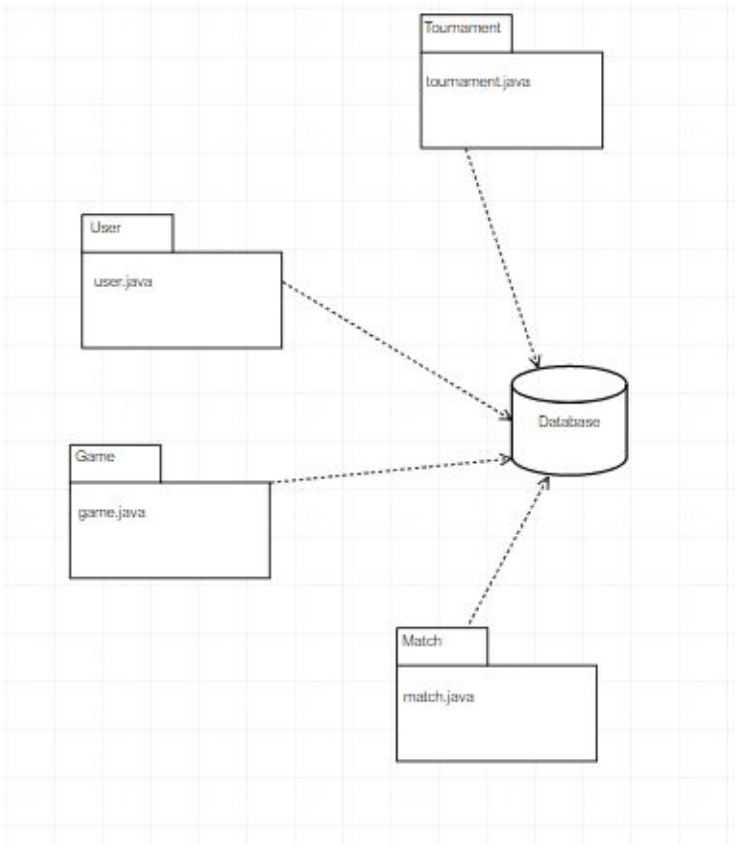
The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.
Hibernate is a high-performance Object/Relational persistence and query service, which is licensed under the open source GNU Lesser General Public License (LGPL) and is free to download. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities. This tutorial will teach you how to use Hibernate to develop your database based web applications in simple and easy steps.

3.2 Diagrams

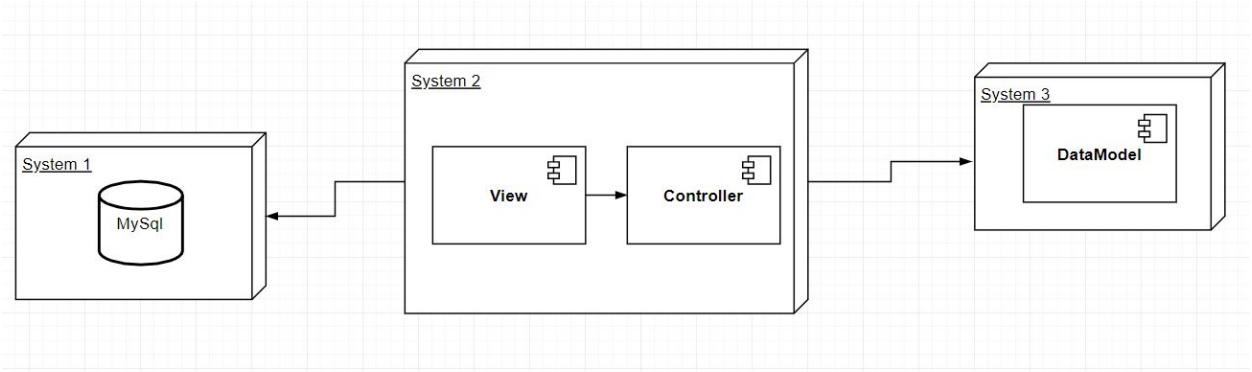
Package diagram



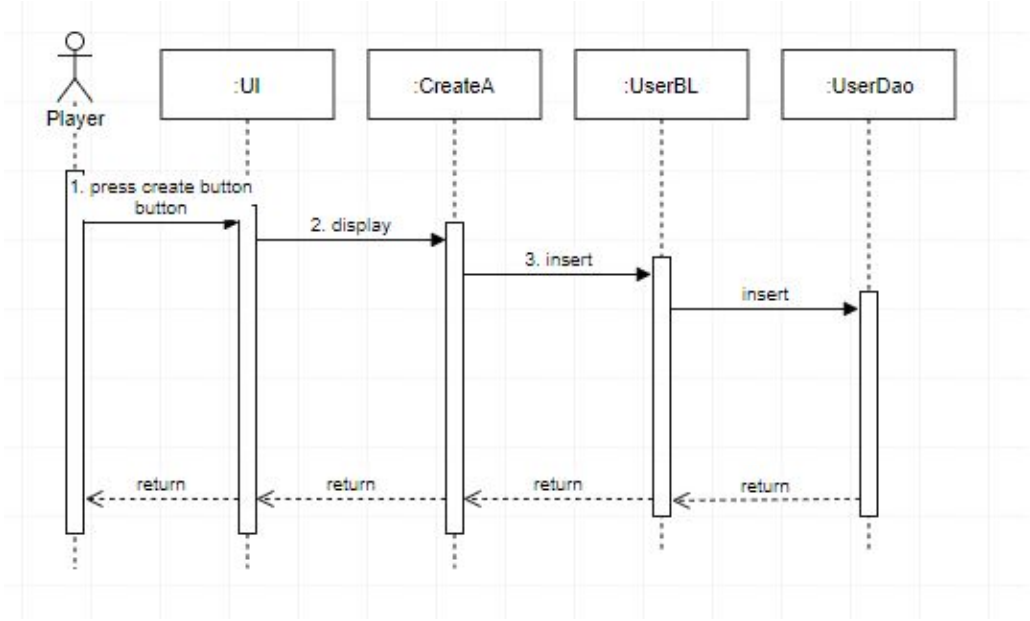
Component Diagram



Deployment Diagram



4. UML Sequence Diagrams



5. Class Design

5.1 Design Patterns Description

The used design Patterns are MVC, DAO, Table Module and 3-tier architecture.

Dao: access to data varies depending on the source of the data. Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation.

Table Module: A single instance that handles the business logic for all rows in a database table or view.

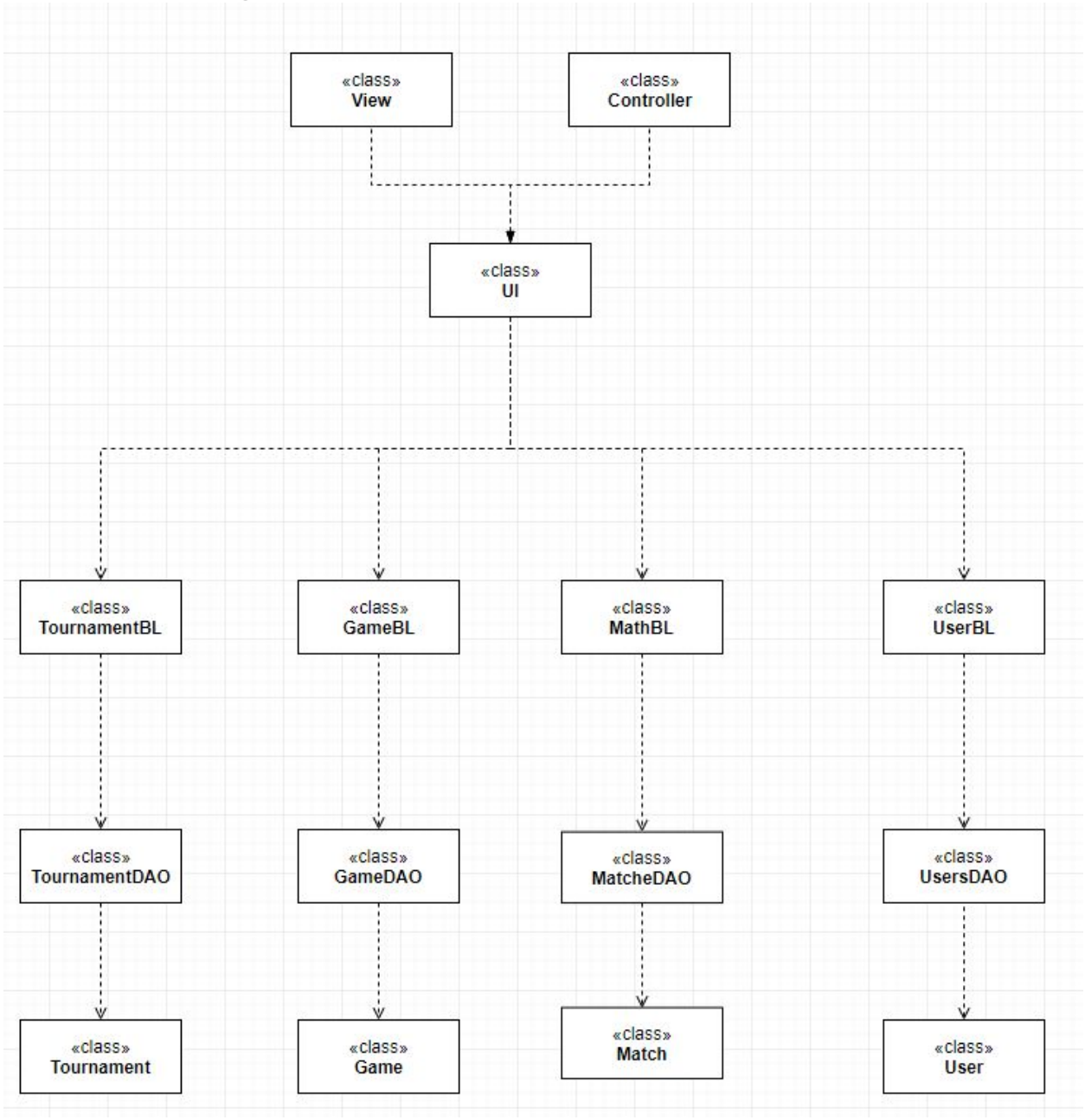
MVC stands for Model, View and Controller. MVC separates application into three components - Model, View and Controller.

Model: Model represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database.Model is a data and business logic.

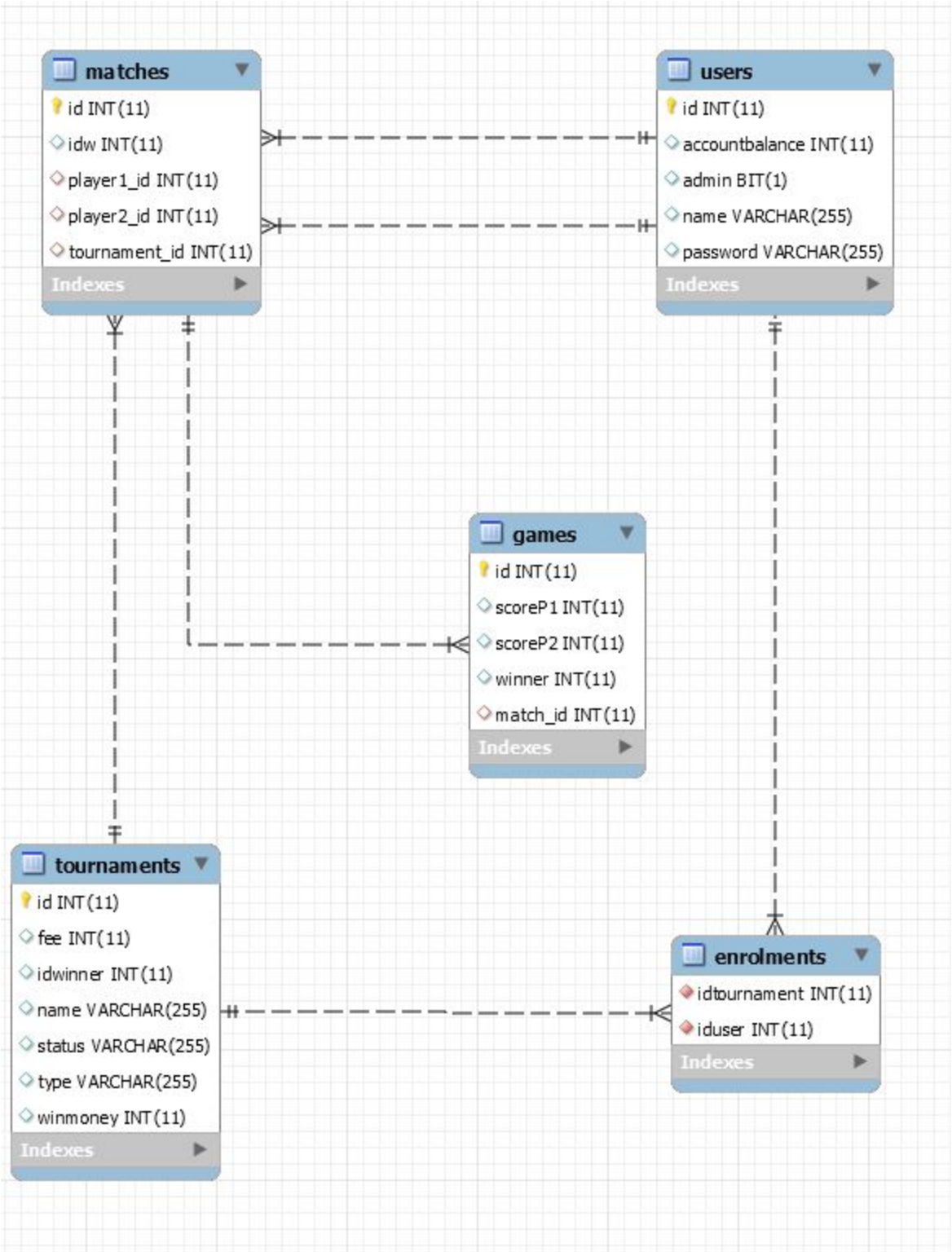
View: View is a user interface. View display data using model to the user and also enables them to modify the data.View is a User Interface.

Controller: Controller handles the user request. Typically, user interact with View, which in-tern raises appropriate URL request, this request will be handled by a controller. The controller renders the appropriate view with the model data as a response.Controller is a request handler.

5.2UMLClassDiagram



6.DataModel


matches
id INT(11)
idw INT(11)
player1_id INT(11)
player2_id INT(11)
tournament_id INT(11)
Indexes
users
id INT(11)
accountbalance INT(11)
admin BIT(1)
name VARCHAR(255)
password VARCHAR(255)
Indexes
games
id INT(11)
scoreP1 INT(11)
scoreP2 INT(11)
winner INT(11)
match_id INT(11)
Indexes
tournaments
id INT(11)
fee INT(11)
idwinner INT(11)
name VARCHAR(255)
status VARCHAR(255)
type VARCHAR(255)
winmoney INT(11)
Indexes
enrolments
idtournament INT(11)
iduser INT(11)
Indexes

7. System Testing

Junit tests implemented for find by id.

8. Bibliography

http://searchsoftwarequality.techtarget.com/definition/3-tier-application
http://www.oracle.com/technetwork/java/dataaccessobject-138824.html
https://martinfowler.com/eaaCatalog/tableModule.html