Large-Scale and Multi-Structured Databases

Project Design

FantaManager

Authors: Edoardo Focacci, Emmanuel Piazza, Matteo Razzai

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

Application Highlights

FantaManager is an application where a user can collect cards resembling his favourite football players, trade them with other users and build his own dream team in order to compete with people online.

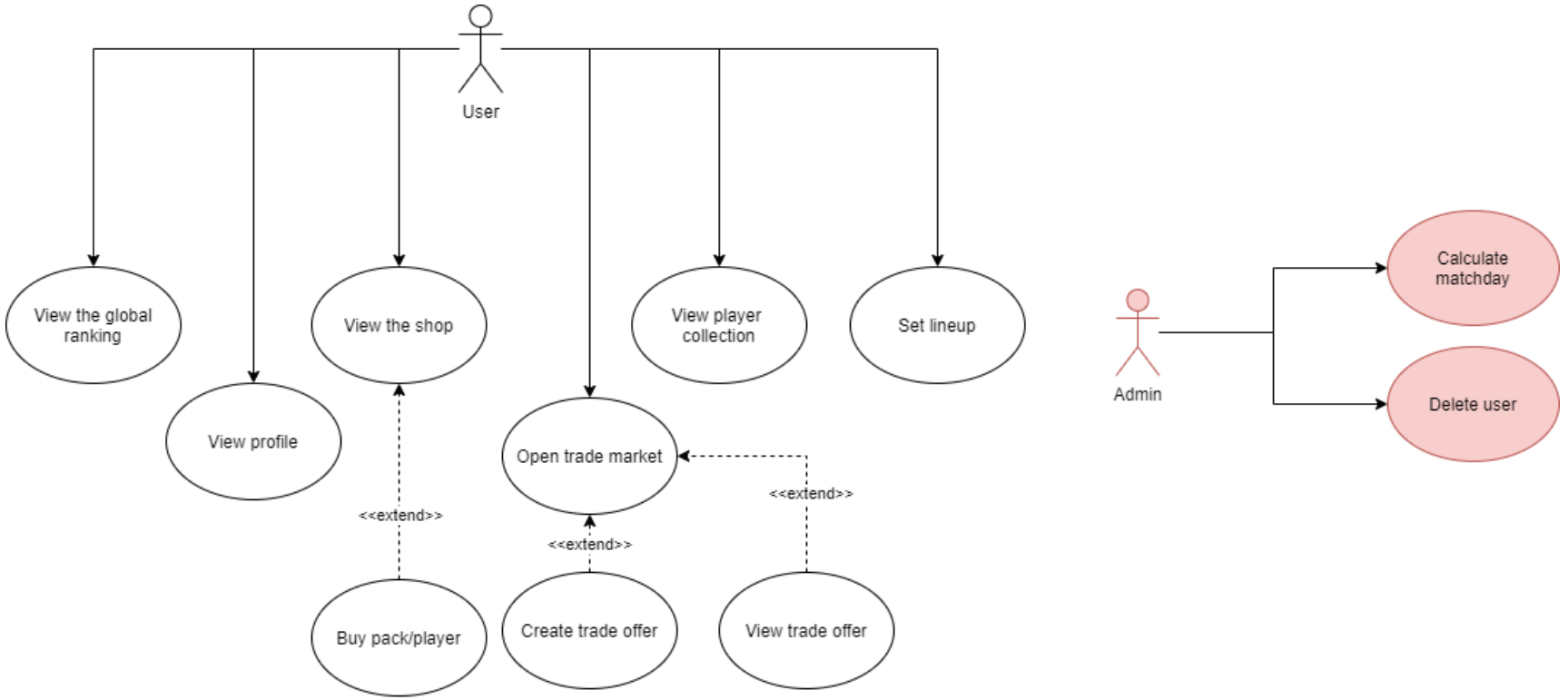
Features

A user can:

- View the list of football players available in the application and their statistics weekly updated.
- Build the team and choose the formation for every matchday in order to earn points and credits.
- Use credits to buy packs of cards or a single card individually.
- Browse the trade requests from other users and accept trades in order to exchange cards.
- View the global ranking.

An admin can: delete other users and calculate the results of the matchday.

Actors and main supported functionalities (simplified)



UML Use Cases (complete)

Dataset Description

Source: Wikipedia – Understats – Kickest - APIFootball

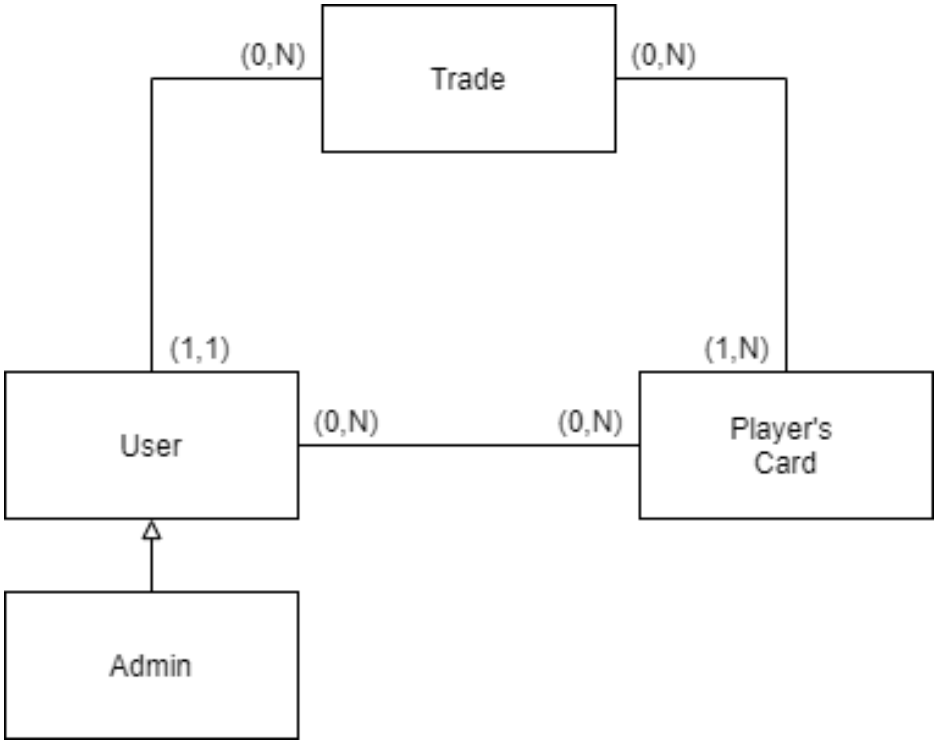
Description: Informations about players retrieved by scraping. Users' informations and trade offers are randomly generated.

Volume: At the beginning, the database is filled up only with players and users informations. Every week the admin will add the rating of the week for every player.

Variety: Statistics and rating of players retrieved from different sites. Wikipedia page of players used for retrieve a brief description of him.

Velocity/Variability: Stats and rate of every player is retrieved at the end of every weekday from different sites. The statistics of the past weeks stays in the database and is calculated in form of average on a different variable.

Preliminary UML Class Diagram



Requirements and Entities handled by Document DB

ENTITIES:

- User (id, surname, name, country, squad[], username, points, credits, etc..)
- Player (id, name, teamName, birthdate, role, statistics[], credit_cost, etc..)
- Trade (id, user, player_from[], player_to[], credits, ts_creation)

CRUD:

- Create/Remove/Modify a user.
- Create/Remove a trade.
- Retrieve player information by name or by team name.
- Retrieve a user by username.
- Retrieve a trade by player's card.

ANALYTICS (3 AGGREGATIONS):

- Best users ranking. (Globally or by region)
- Checking the evolution of a specific statistic of a player. (Goals, assist, dribbling, etc..)
- Check the best players by Serie A team.

Requirements and Entities handled by Key-Value DB

ENTITIES (KeyValues):

- *'user':user_id:'player':player_id:'name'*
- *'user':user_id:'player':player_id:'role'*
- *'user':user_id:'player':player_id:'quantity'*

CRUD:

- Add/Remove a player's card from a user's collection.
- Update quantity of player's cards owned by a user.
- View all players' cards owned by a user.

Software Architecture Preliminary Idea

- Application created using the *Java* language.
- Scraping script written in *Python*.
- *MongoDB* and *Redis* are used for managing the database.
- Front-end system created using the *javafx* library.