GameBase Application Project Proposal

Table of Contents

Introduction and Requirements

The application project proposed in this paper consists in a software solution (preliminarily named "*GameBase*") whose core functionality is collecting, organizing, and offering information related to videogames, allowing its users to search and browse for games using different filters (e.g. title, release date, platform, etc…) and view detailed information on such games.
In order to enhance the application user experience, users will be allowed some limited interaction, on the one side regarding the games, such as adding a title to their *favourites* and view their favourite games list, and on the other concerning other users, such as the possibility to *follow* another user and view his favourite games list.
In addition, in order for the application to produce an income (or at least, to ensure self-sustainability) on the one hand users will be offered a one-time payment option allowing them to perform more advanced queries on the dataset whose results will be presented as statistics, a solution which should appeal to professional-oriented users such as data analysts, and on the other the games should present, when applicable, links to affiliated digital stores allowing users to purchase them.

Functional Requirements

The following list summarizes the projected functional requirements of the application:

- The videogames dataset should be conspicuous in size, and should be composed of data of different nature, such as text, images, video, hyperlinks to related content, etc… (hypermedia).
- Access to the application is to be granted to registered users only through a login system based on the *username/password* model, in order to also distinguish the different user categories in the application (standard users, analysts and administrators) with their related privileges.
  A sign-up form allowing new users to register within the application as standard users must also be provided.
- The following functions must be offered to the application's administrators:
  - Update the entire games dataset or the information relative to a single game (where a dynamic scraping mechanism will be considered).
  - Delete a game from the dataset.
  - Delete a user from the application.
- The following functions must be offered to the application's standard users:
  - Search and browse the list of games using different filters, such as title, release date, platform, genre, etc…
  - View the detailed information relative to a game.
  - Add/remove a game from their favourites list.
  - Rate a game.
  - Follow/unfollow another user.
  - View the favourite games list of followed users.
  - Perform a one-time payment to upgrade their account to an analyst account.
  - Enter and update their personal information, including name, gender, age, favourite game genre, etc…
  - Follow links to affiliated digital stores in order to purchase games.
- The analysts must be allowed to perform a set of advanced queries on the games' and users' datasets, whose results must be presented in the form of statistics.

Non-Functional Requirements

As for the non-functional requirements, given that the success and profitability of the application strictly relies on the number of active users, a modern structured graphical interface must be provided in order to enhance the user experience with the application.

In addition, since a professional-grade quality of service (QoS) is desired, both in terms of responsiveness and tolerance to single points of failure, the dataset of the application should be replicated and possibly sharded among different servers, a solution that allows for load-balancing purposes and future-proofs a possible development towards a content distribution network (CDN) architecture.

Working Hypotheses

The design of the application will also rely on the following working hypotheses:

- The details of how the payment relative to a user upgrading their account to an analyst account is processed are not taken into consideration.
- Existing regulations and concerns relative to the users' data privacy are not taken into account.
- Advanced aspects relative to data replication, such as load balancing and disaster recovery, are not taken into account.

Specification

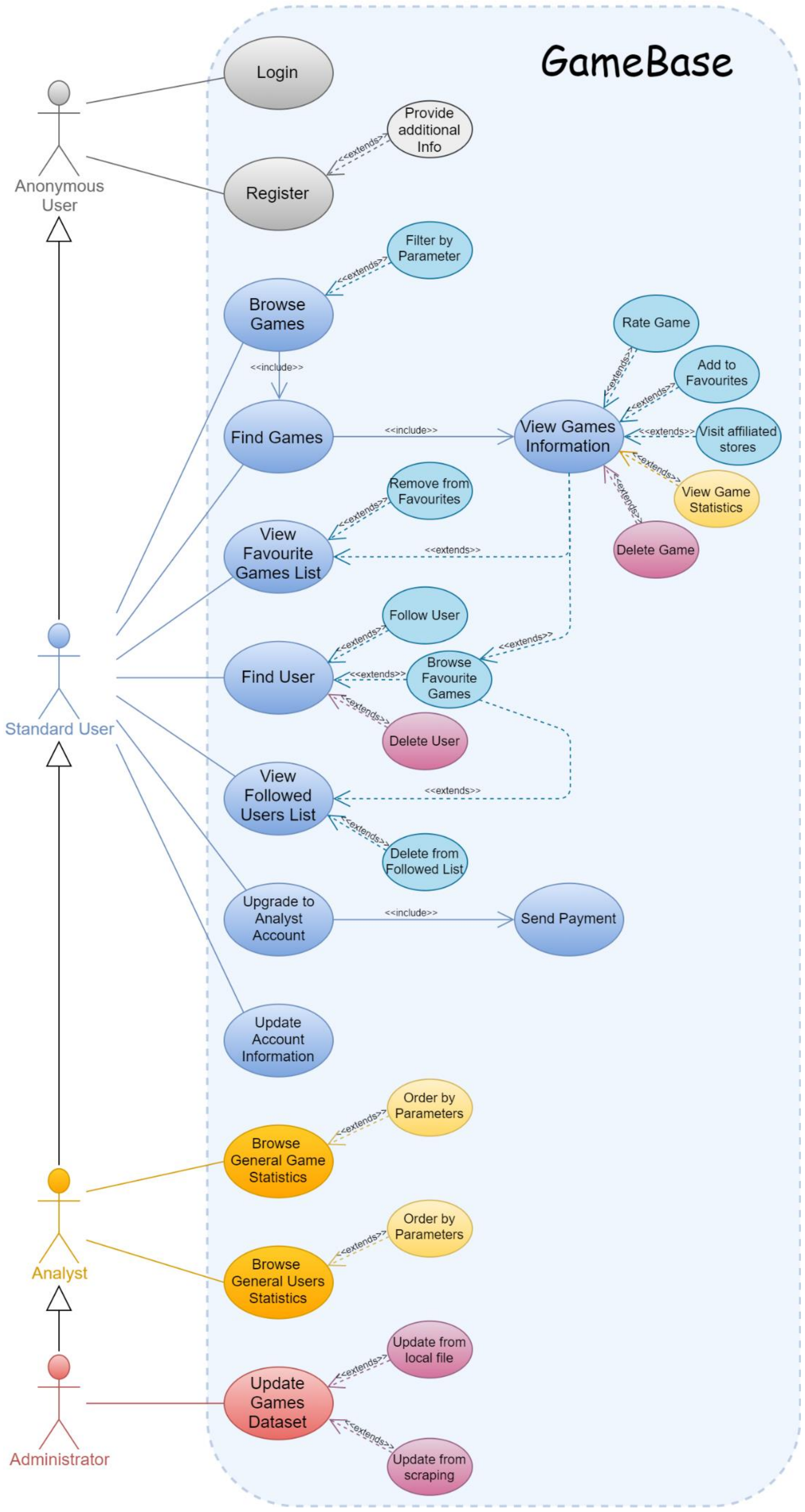
Actors and Use Cases Diagram

From the software functional requirements, the application is meant to be used by four actors:

- The Anonymous Users, who are allowed only to log in or register within the application, in the latter case possibly providing additional personal details.
- The Standard Users, who are allowed to interact with most of the application's features such as browsing, finding and viewing the detailed information regarding games, add a game to their favourites list, rate a game, follow other users, see the followed users' favourites lists, perform a one-time payment to upgrade their account to an analyst account, update their personal account information, and follow affiliated links to digital stores in order to purchase games
- The Analysts, who in addition to the features offered to the standard users are also allowed to perform a set of advanced queries on the games' and users' datasets, whose results will be presented as statistics.
- The Administrators, who are allowed to perform every operation in the application, including the ones requiring a high level of privilege such as updating the entire games dataset or the information relative to a single game, and delete a user or a game from the application.

The set of actors with their use cases is shown in the diagram below:

Use Cases Diagram



Software Architecture

As for its architecture, the software will be divided into three tiers according to the *Front-End – Middleware – Back-End* paradigm, where *Java* 8 will be used as the core programming language:

Front-End

The Front-End will consist in a graphical interface based on the *Swing* API, which will allow users to interact with the application, along with the modules required to exchange data with the Middleware.

Middleware

The Middleware will comprise the logic and the data routing aspects of the application, where:

- The *Games* dataset will be managed using the *MongoDB* 4.4.2 document database and will be partitioned into two separate replicated shards, where the middleware will host a *mongos* instance acting as a router between the application logic and the shards and a *mongod* instance acting as configuration server for the latter.

- The *Users* dataset will be managed using a single instance of the *Neo4j* 3.5 graph database, which will also contain the games of the *Games* dataset as stub entities in order to store their relations with the Users (a feature which is further discussed later in the document).

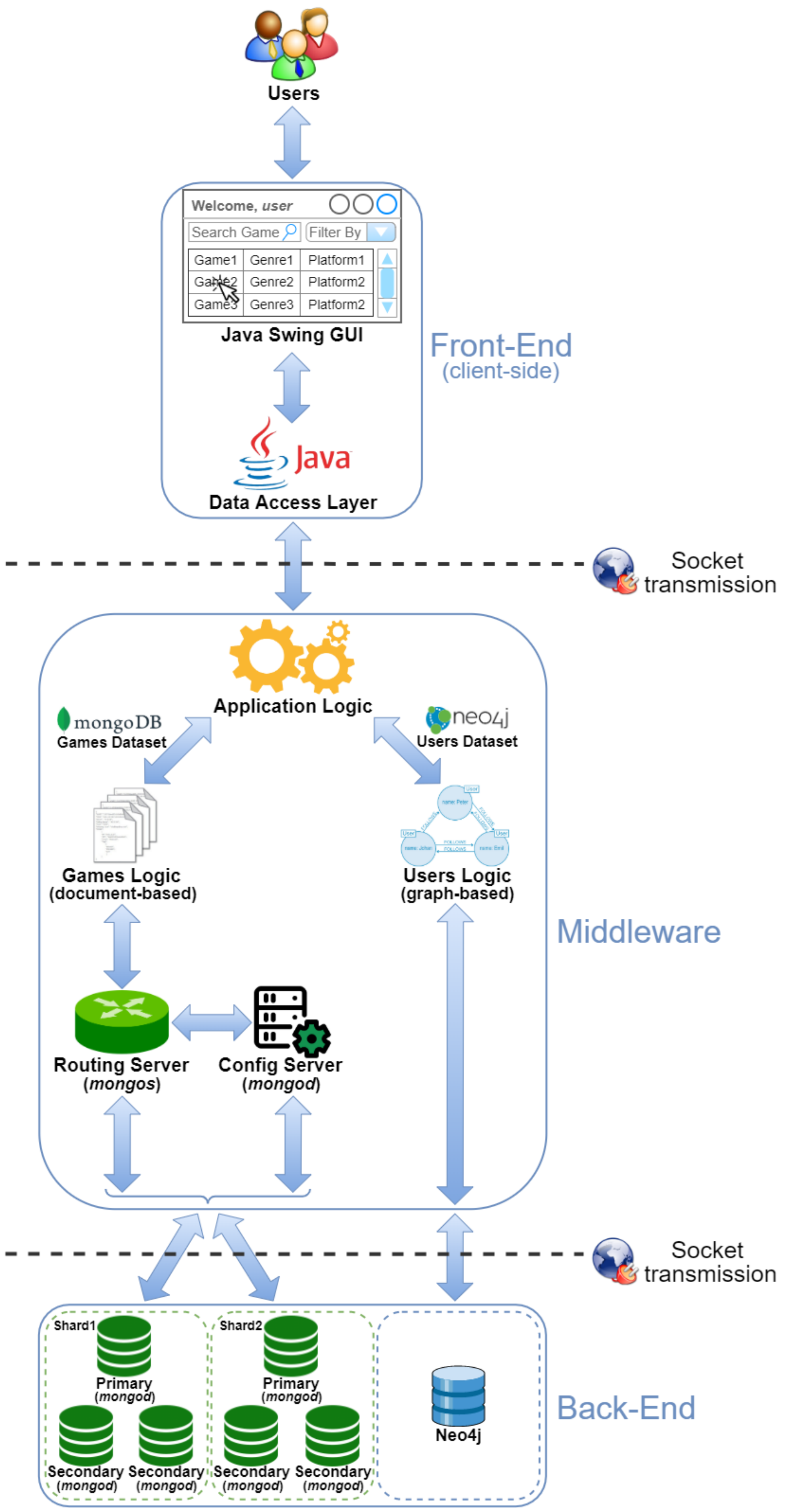
Back-End

The Back-End will comprise the data persistence aspects of the application where, as discussed above:

- The *Games* dataset will be partitioned into two *MongoDB* shards, each consisting of a replica set composed of 3 *mongod* instances (1 primary and 2 secondary).

- The *Users* dataset will be stored in a single *Neo4j* instance.

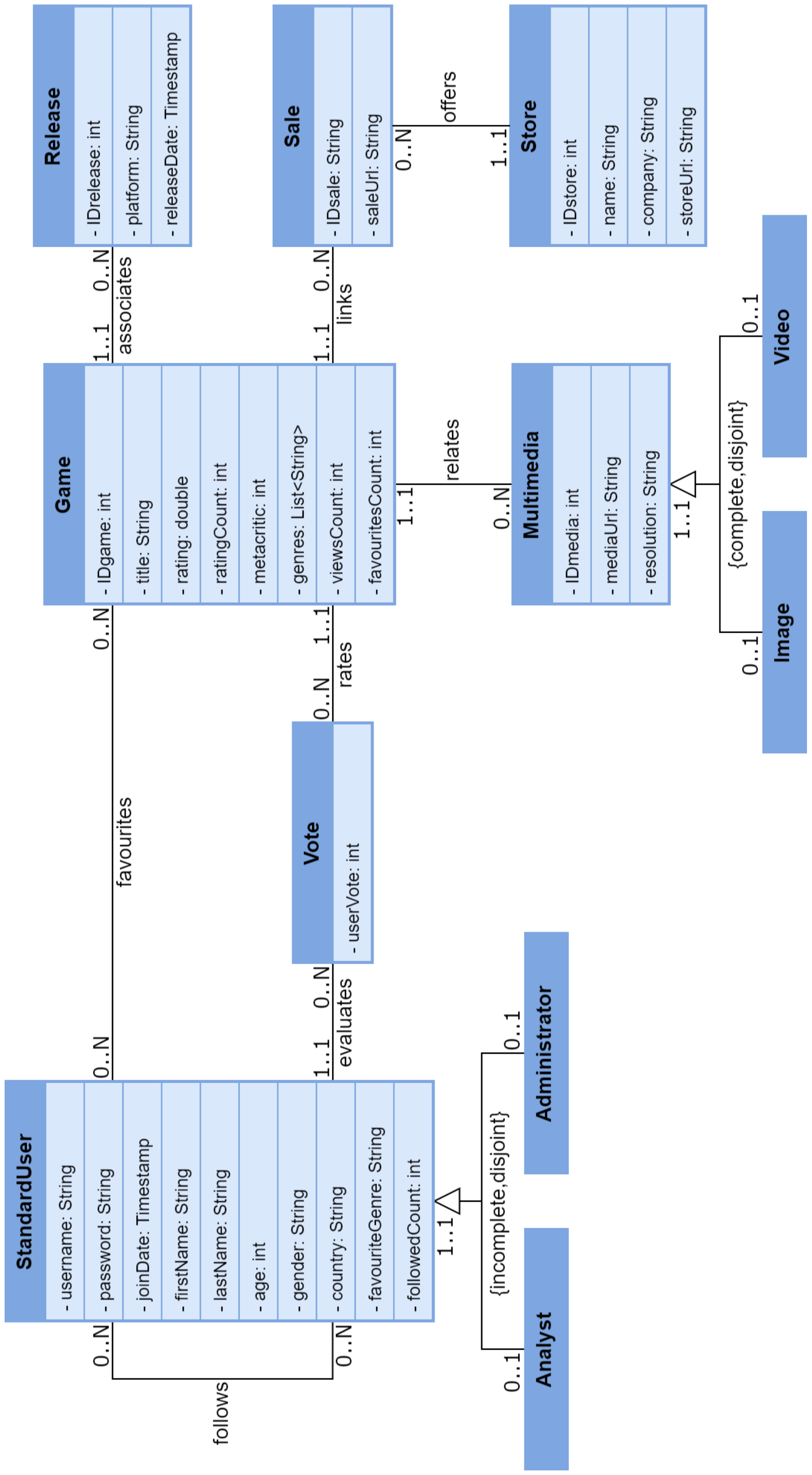
The following diagram summarizes the overall software architecture of our application:



Users
Welcome, user
Search Game
Filter By
Game1 Genre1 Platform1
Game2 Genre2 Platform2
Game3 Genre3 Platform2
Java Swing GUI
Java
Data Access Layer
Front-End
(client-side)
Socket transmission
Application Logic
mongoDB
Games Dataset
neo4j
Users Dataset
Games Logic (document-based)
Users Logic (graph-based)
Middleware
Routing Server (mongos)
Config Server (mongod)
Socket transmission
Shard1
Primary (mongod)
Secondary (mongod)
Secondary (mongod)
Shard2
Primary (mongod)
Secondary (mongod)
Secondary (mongod)
Neo4j
Back-End

Analysis Classes Diagram

From the software's requirements and the additional working hypotheses the following analysis classes can be identified within the application:



Classes Definitions

CLASS	DESCRIPTION
StandardUser	A user with no special privileges within the application, who is nevertheless allowed to interact with most of its functionalities
Analyst	A special user who is allowed to perform advanced queries on the users' and games' datasets, whose results will be presented as statistics
Administrator	A super user who is allowed to perform every operation in the application
Game	A videogame whose information is offered by the application
Vote	A user's evaluation of a game
Release	A release of a particular game for a specific platform
Multimedia	A multimedia content related to a particular game, which may consist in an image or a video clip
Image	An image related to a particular game
Video	A video clip related to a particular game
Store	An affiliated digital store where users can purchase games
Sale	A sale offer of a particular game in a specific store

Classes Attributes

StandardUser		
Attribute	**Type**	**Description**
username	String	A unique string identifying the user, which is also used by them to access the application
password	String	The password required for the user to access the application
joinDate	Timestamp	The date and time the user registered within the application
firstName	String	The user's first name (optional)
lastName	String	The user's last name (optional)
age	int	The user's age (optional)
gender	String	The user's gender (optional)
country	String	The user's country (optional)
favouriteGenre	String	The user's favourite videogame genre (optional)
followedCount	int	The number of the user's followers

Analyst		
Attribute	**Type**	**Description**
none (same as the superclass)		

Administrator		
Attribute	**Type**	**Description**
none (same as the superclass)		

Game		
Attribute	**Type**	**Description**
IDgame	int	The game's unique identifier
title	String	The game's title
rating	double	The mean of all users' evaluations of the game, expressed as a decimal number between 1.0 and 5.0
ratingCount	int	The number of unique users' evaluations of the game
metacritic	int	The Metacritic® rating of the game
genres	List<String>	The list of genres the game belongs to
viewsCount	int	The number of times the game was viewed by a user
favouritesCount	int	The number of users who have added the game to their favourites list

Vote		
Attribute	**Type**	**Description**
userVote	int	A user's evaluation of a specific game expressed as an integer between 1 and 5

Release		
Attribute	**Type**	**Description**
IDrelease	int	The game release's unique identifier
platform	String	The platform on which the game was released
releaseDate	Timestamp	The release date of the game for the platform

Multimedia		
Attribute	**Type**	**Description**
IDmedia	int	The multimedia content's unique identifier
mediaUrl	String	The URL from which the multimedia content can be retrieved
resolution	String	The multimedia content's resolution (e.g. 1920x1080)

Image		
Attribute	**Type**	**Description**
none (same as the superclass)		

Video		
Attribute	**Type**	**Description**
none (same as the superclass)		

Store		
Attribute	**Type**	**Description**
IDstore	int	The store's unique identifier
name	String	The store's name
company	String	The store's owner company
storeUrl	Timestamp	The URL of the store's homepage

Sale		
Attribute	**Type**	**Description**
IDsale	int	The sale's unique identifier
saleUrl	String	The game's URL within the store

Application Datasets Preliminary Design

Games Dataset (MongoDB)

The *Games* dataset of our application will be organized in a single MongoDB collection whose documents represent games, and will be built from a partition of the dataset offered by the RAWG service, which allows its contents to be scraped directly in JSON format through a set of HTTP-based API.
It should be noted, however, that the structure of the scraped documents will not exactly reflect that envisaged for the documents of our database in terms of both data organization and relevance, and so the scraped documents will have to undergo a "normalization" process before being added to our *Games* database.

The projected structure of the documents in the *Games* dataset is outlined below:

```
{
 _id: ObjectId("5df0af2d958481ae0bf94862"),
 title: "The Witcher 3: The Wild Hunt",
 rating: 4.87,
 ratingCount: 283319,
 metacritic: 93,
 genres: ["RPG","Action"],
 viewsCount: 324619,
 favouritesCount: 16281,
 releases:
  [
   {platform: "PC", releaseDate: 18-05-2015},
   {platform: "XboxOne", releaseDate: 18-05-2015},
   {platform: "Switch", releaseDate: 18-05-2015},
   {platform: "PS4", releaseDate: 06-06-2019}
  ],
 sales:
  [
   {store: "Steam", company: "Valve", saleUrl: "http://steam.com/TheWitcher3"},
   {store: "GOG", company: "GOG", saleUrl: "http://gog.com/TheWitcher3"},
   {store: "XboxStore", company: "Microsoft", saleUrl: "http://xboxstore.com/TheWitcher3"},
   {store: "PlayStationStore", company: "Sony", saleUrl: "http://psstore.com/TheWitcher3"}
  ],
 multimedia:
  {
   images:
    [
     {resolution: "640x480", mediaUrl: "http://TheWitcher3/images/lowres.png"},
     {resolution: "1980x1080", mediaUrl: "http://TheWitcher3/images/medres.png"},
     {resolution: "2440x1440", mediaUrl: "http://TheWitcher3/images/highres.png"}
    ],
   videos:
    [
     {resolution: "640x480", mediaUrl: "http://TheWitcher3/videos/lowres.png"},
     {resolution: "1980x1080", mediaUrl: "http://TheWitcher3/videos/medres.png"},
    ]
  }
}
```

Users Dataset (Neo4j)

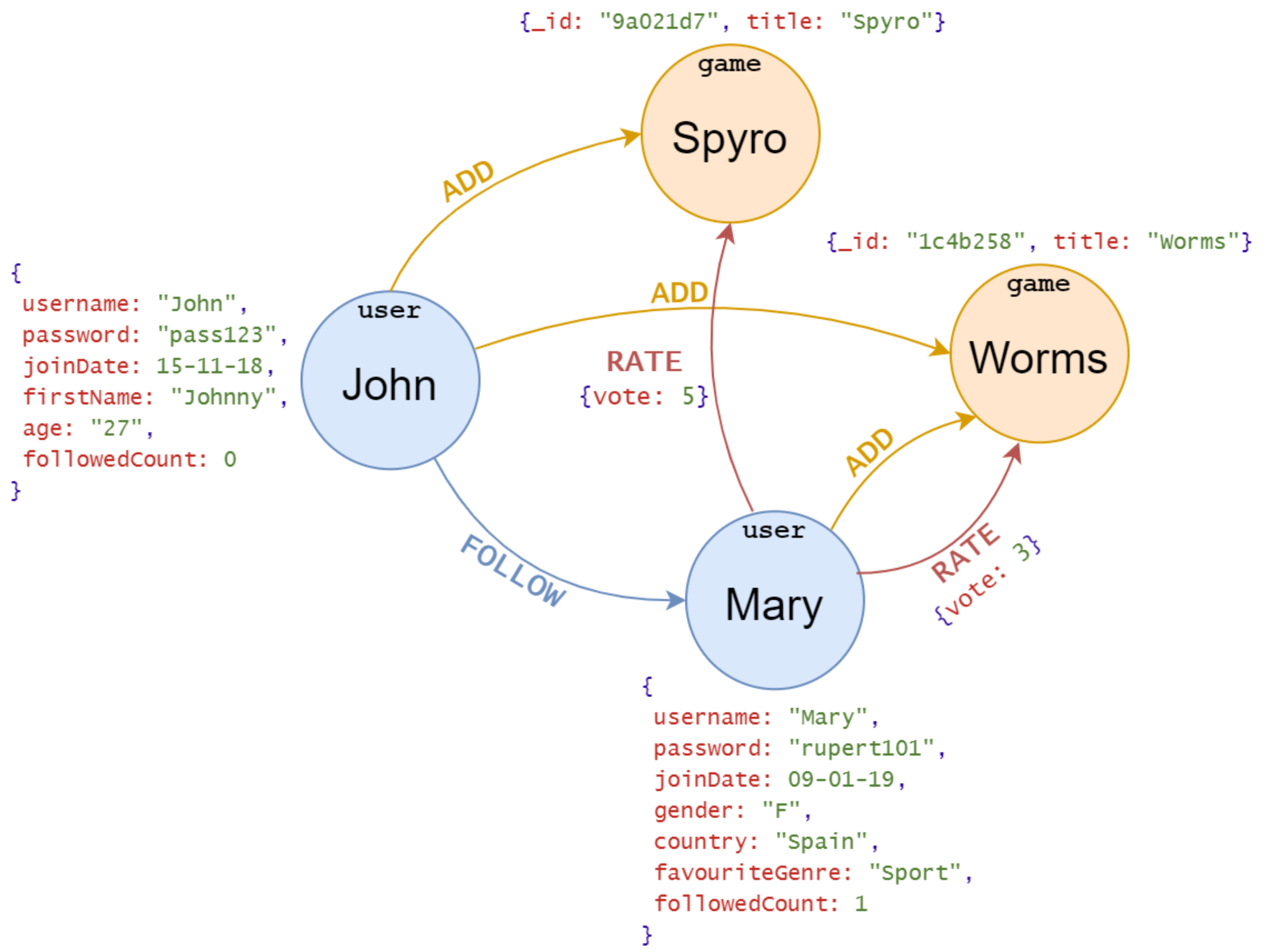
The *Users* dataset of our application will consist in a graph stored in a single Neo4j instance, where the nodes can belong to two different categories:

- The **Users**, representing the users registered within the application regardless of their role (Standard User, Analyst, Administrator), and having as attributes the ones specified in their account information as for the "StandardUser" class attributes (pag. 8), where the *username*, *password*, *joinDate* and the *followedCount* are mandatory.
- The **Games**, which are "stub" nodes that are created whenever a new game is added to the *Games* database for the purpose of storing the relationships between the Users and the Games in the application, and have as attributes the *_id* of the related document in the *Games* database and the game *title*.

The following types of relationships are envisaged in the database:

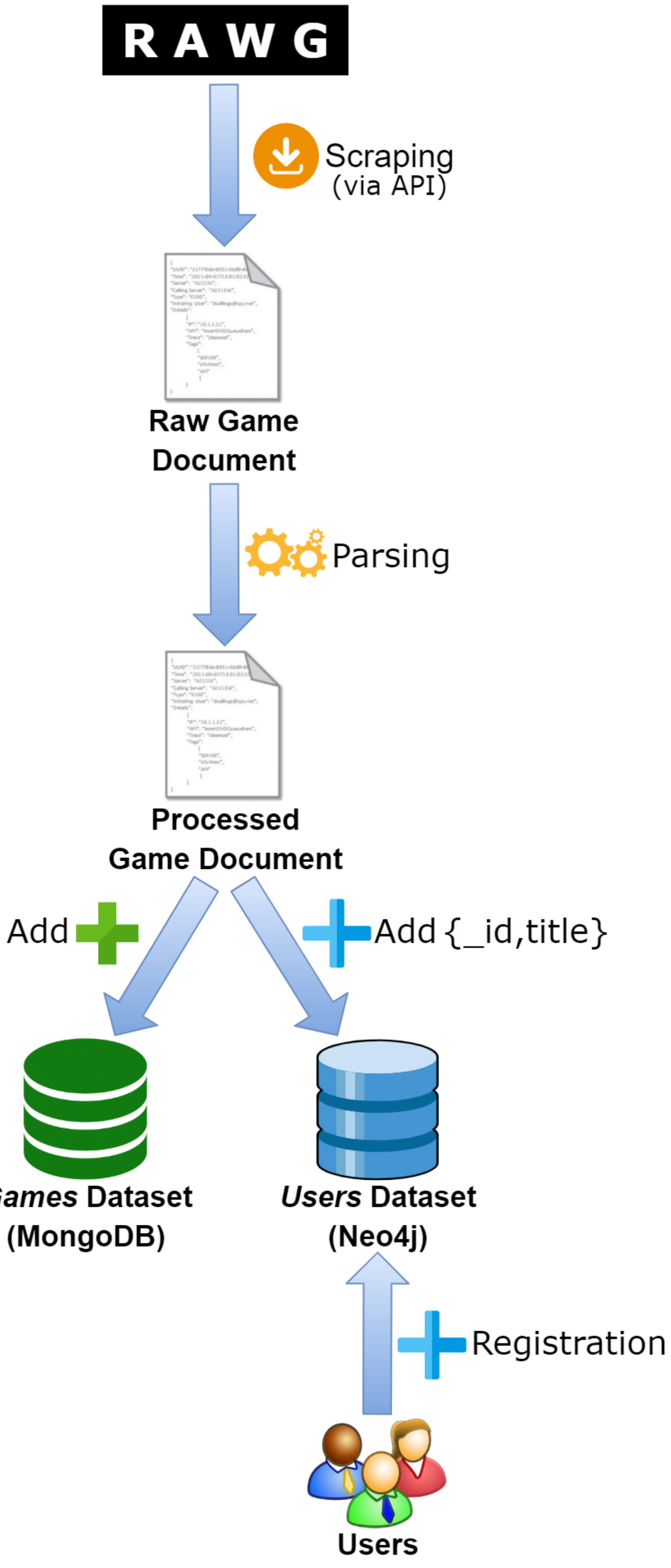
- **User FOLLOW User**, which is created/removed when a User adds/removes another User to his followed list, and has no attributes.
- **User ADD Game**, which is created/removed when a User adds/removes a Game to his favourites list, and has no attributes.
- **User RATE Game**, which is created when a User rates a game, has as attribute the *vote* consisting in an integer between 1 and 5, and cannot be removed but only updated with a different vote.

An example of the type of nodes and relationships used in the graph database is shown below:



Datasets Populations Summary

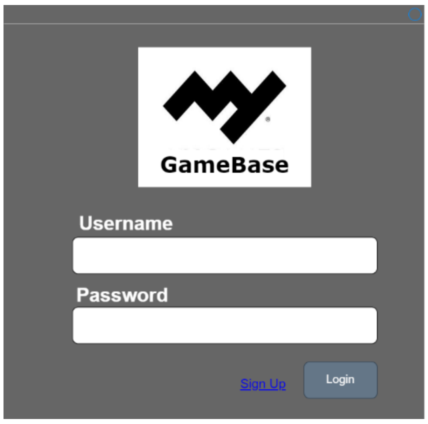
A summary of how the *Games* and *Users* datasets are populated within the application is shown below:



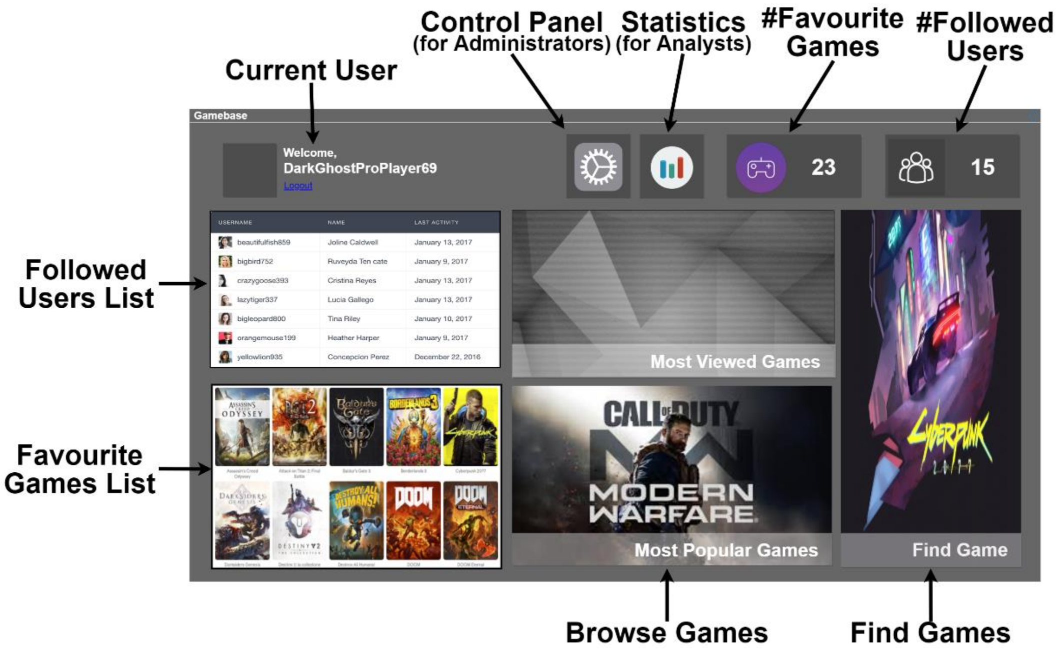
Graphical User Interface Mock-ups

As the last part of this application proposal, we present some mock-ups of the graphical user interface:

Login Screen

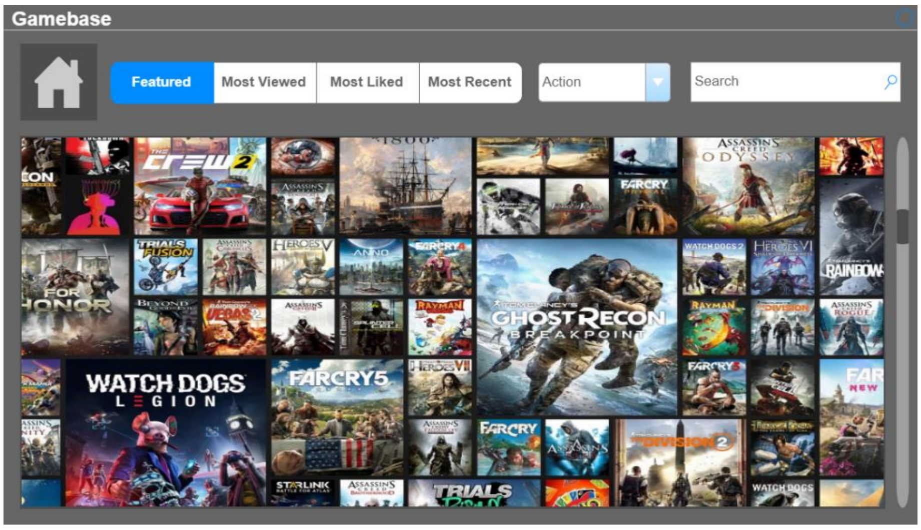


Main Window

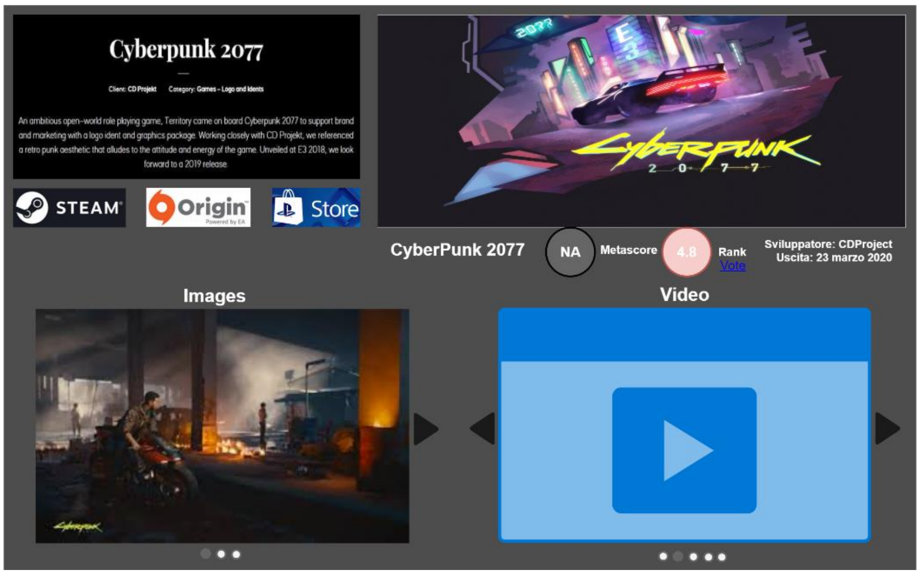


USERNAME	NAME	LAST ACTIVITY
beautifulfish859	Joline Caldwell	January 13, 2017
bigbird752	Ruveyda Ten cate	January 9, 2017
crazygoose393	Cristina Reyes	January 13, 2017
lazytiger337	Lucia Gallego	January 13, 2017
bigleopard800	Tina Riley	January 10, 2017
orangemouse199	Heather Harper	January 9, 2017
yellowlion935	Concepcion Perez	December 22, 2016

Browse/Search Game Window



Game View



Cyberpunk 2077

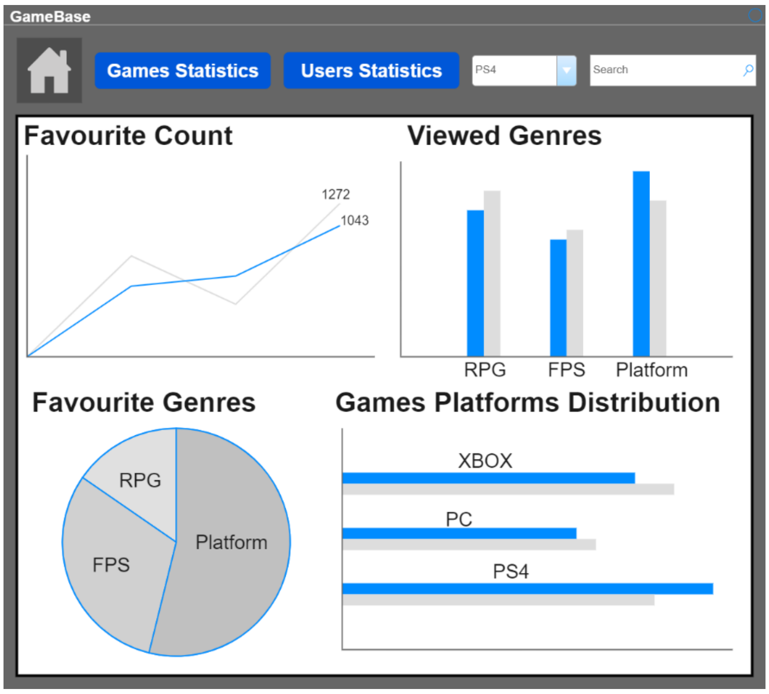
Client: CD Projekt Category: Games – Logo and Idents

An ambitious open-world role playing game, Territory came on board Cyberpunk 2077 to support brand and marketing with a logo ident and graphics package. Working closely with CD Projekt, we referenced a retro punk aesthetic that alludes to the attitude and energy of the game. Unveiled at E3 2018, we look forward to a 2019 release.

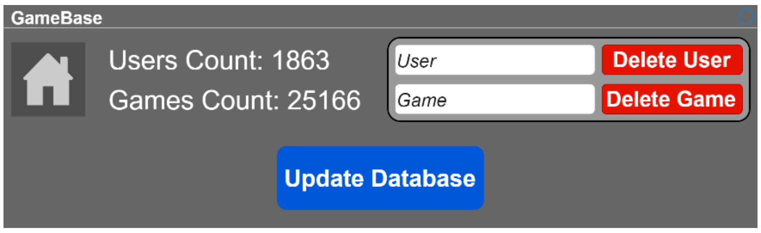
CyberPunk 2077 NA Metascore 4.8 Rank Vote

Sviluppatore: CDProject
Uscita: 23 marzo 2020

Statistics Page



Administration Panel



Users Count: 1863
Games Count: 25166