University of Piraeus

Department of Digital Systems

Network-centric Information Systems 2020-2021

Semester Work

Panagiotis Karamolegos

E17065

Contents

List of Images

1. System Thematics

1.1. Problem to solve

In the framework of the exculpatory work of the course Network-centric Information Systems 2020-2021, I made the decision to build a Social Media-type Information System. This Social Media refers to people who find it difficult to socialize and as a result cannot enjoy moments within electronic games with a company.

The Information System was called Game Society and its purpose is to bring these people together, as a mediator. Because of this, they will not need to change their character and will be able to socialize easily and quickly to experience many moments of joy through Game Society users.

Game Society is built using RESTful Web Services and MySQL Database.

1.2. System Implementation Process

The following procedure was followed for the implementation of the System:

- Decision of the problem to be solved
- Decision of the functions it should fulfill
- Designing the Database schema
- Creation ofRESTful Web Services using the Postman tool
- Construction ofUI for Information System

The methodology followed was of the Agile type as in each of the above steps (Sprints) of the implementation process:

- Planning
- Designing
- Building
- Testing
- Reviewing + Deploying

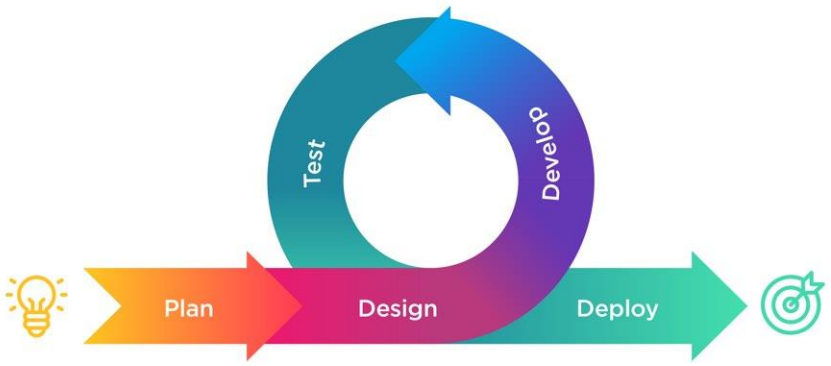


Figure 1: A Sprint in Agile Methodology

2. System Implementation

2.1. Architecture

The System has been created using Service-Oriented Architecture (SOA) in order to reuse services, better maintenance and parallel development of its Services. By using SOA, complex services are broken into smaller and isolated – independent services. Consequently, their creation ends up faster and well organized.

The construction of the Information System was based on REST-based Web Services and specifically on RESTful Web Services for reasons of speed of implementation as well as practice on this type of web services. Due to the security requirement of the Information System in question, it would be useful in the future to recreate it using SOAP-based Web Services.

2.2. Implemented Web Services

All Web Services built are RESTful and exist within the project's src. They are inside the my.restful.web.services package, in the GameSociety class.

Then each Web Service will be presented in a table with the following fields:

- Input: In which there will be all the arguments needed by eachWeb Service.
- Reuse:**(YES | NO)**In which you will report if this theWeb Service is one of those that are reused within the System.
- Use by:**(System | Users | Admins | Users/Admins)**On this field will appear:
    - If this Web Services is intended to be used automatically by the system you are calling**(system)**.
    - If intended for use by all Users of the System **(Users)**.
    - If intended for use by System Administrators only **(Admins)**.
    - If intended for all Users but adds a floating function to Administrators **(Users/Admins)**.
- HTTP METHOD: This field indicates the type of HTTP method of the Web Service.
- Returns: The type of information this returnsWeb Service.
- Description: Its operationWeb Service.
- Endpoint: The endpoint on which the Web Service can be called.
- Use in the System: Theendpoints using the Web Service.
- Comment: Additional useful information.

2.2.1. System Services

testForDB

Entrance	Reuse	Used by	HTTP METHOD
-	YES	System	GET
Returns			
-			
Description			
It builds the Database in case it doesn't exist in the first place. Imports as default Administrator a User with nickname "admin" in case there is not already an Administrator inside.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/testForDB			
Use in the System			
On all endpoints			
Comment			
This Web Service must be called so that the connection to the database can be made and finally all other Web Services of the System can function.			

testForUser

Entrance	Reuse	Used by	HTTP METHOD
User's nickname	NO	System	GET
Returns			
JSON Object			
Description			
Returns the User who owns the entered nickname. It is used as a check so that Users with the same nickname are not entered.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/testForUser/{nickName}			
Use in the System			
http://localhost:8080/GameSociety/Pages/welcome/register.jsp			
Comment			
The specific Web Service closes testForDB and therefore builds (if it does not exist) and connects the Database. As it also initializes the Administrator "admin" in the event that there is no other Administrator within the Information System. You use to introduce new Users within the Information System.			

testForEmail

Entrance	Reuse	Used by	HTTP METHOD
User's email	NO	System	GET
Returns			
JSON Object			
Description			
Returns the User who owns the entered email. It is used as a check so that Users with the same email are not entered.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/testForEmail/{email}			
Use in the System			
http://localhost:8080/GameSociety/Pages/welcome/register.jsp			
Comment			
The specific Web Service closes testForDB and therefore builds (if it does not exist) and connects the Database. As it also initializes the Administrator "admin" in the event that there is no other Administrator within the Information System. You use to introduce new Users within the Information System.			

2.2.2. System User Services

addAUser

Entrance	Reuse	Is used from	HTTP METHOD
User's name User's surname User's nickname User's privileges User's password User's email	NO	System	POST
Returns			
-			
Description			
Adding a new user within the Information System.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/addAUser/{name}/{surname} / {nickName}/{isAdmin}/{password}/{email}			
Use in the System			
http://localhost:8080/GameSociety/Pages/welcome/register.jsp			
Comment			
-			

deleteAUser

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	YES	Users/Admins	DELETE
Returns			
-			
Description			
Deletes a user from the Information System. In the event that the last Administrator has been deleted, then a User with the nickname "admin" is entered as the default Administrator.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/deleteAUser/{userID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/UserProfile.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/EditProfile.jsp			
Comment			
Use in the profile settings of each User in case he wants to delete himself. You also use as a floating function for Administrators, so that they can from the profile of a single User delete him from the Information System.			

getUsersByNickName

Entrance	Reuse	Is used from	HTTP METHOD
User's nickname	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all users whose nickname contains the entered alphanumeric character.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getUsersByNickName/{nickName}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Search/SearchSelection.jsp			
Comment			
If a user's nickname is Panagiotis, and we give "n" as input, then it will also return Panagiotis, because of the -n- within the alphanumeric.			

getUser

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	YES	System	GET
Returns			
JSON Object			
Description			
Returns the details of the User who owns the entered ID.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getUser/{userID}			
System Usage - in all but			
http://localhost:8080/GameSociety/Pages/HomePage/LogOut.jsp			
http://localhost:8080/GameSociety/Pages/welcome/login.jsp			
http://localhost:8080/GameSociety/Pages/welcome/register.jsp			
Comment			
This Web Service exists mainly for the presentation of the User who is Logged in on Social Media. However, it is also used to obtain information about users who are not logged in to the browser the User is using.			

changePrivileges

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	NO	Admins	PUT
Returns			
-			
Description			
Changes a User's rights from Administrator to Simple User and vice versa. Also, in case there is no more Administrator after its use, then it creates the default Administrator with the nickname "admin".			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/changePrivileges/{userID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/UserProfile.jsp			
Comment			
This function can only be used by System Administrators, in order to create new Administrators, or demote unnecessary administrators. The Information System is built so that an Administrator cannot demote himself.			

getAllUsers

Entrance	Reuse	Is used from	HTTP METHOD
-	NO	Admins	GET
Returns			
JSON Array			
Description			
Returns all users of the Information System.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getAllUsers			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Admin/ShowAllUsers.jsp			
Comment			
This function can only be used by System Administrators, in order to easily find other Users so that they can tamper with their rights or possibly delete problematic material from them.			

changeProfileInfo

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	NO	Users	PUT
User's name			
User's surname			
User's profile Picture Path			
User's nickname			
User's password			
User's email			
Returns			
-			
Description			
Changes a user's password, first name, and last name with the imported Service data.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/changeProfileInfo/{userID}/{n ame}/{surname}/{profilePicturePath}/{nickName}/{password}/{email}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/EditProfile.jsp			
Comment			
The rest of the information is not given through the UI as an option for update as it would interfere with the security of the System and would also contradict regulations, such as the regulation that two Users cannot have the same nickname.			

2.2.3. Playlist Services

addAUserGame

Entrance	Reuse	Is used from	HTTP METHOD
User's ID Game's name	NO	Users	POST
Returns			
-			
Description			
Adds a game title to a User's list to the List.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/addAUserGame/{userID}/{na me}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/MyGames/MyGames.jsp			
Comment			
These games are used to enable Users to find people with common interests within Social Media.			

deleteAUserGame

Entrance	Reuse	Is used from	HTTP METHOD
User's ID Game's ID	NO	Users	DELETE
Returns			
-			
Description			
Deletes a game from a User's List.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/deleteAUserGame/{userID}/ { gameID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/MyGames/MyGames.jsp			
Comment			
Each User will be able to delete from the List games that he probably does not play anymore.			

getAllUserGames

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	YES	Users	GET
Returns			
JSON Array			
Description			
Returns the entire List of games of a user.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getAllUserGames/{userID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/MyGames/MyGames.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/OtherUser/SeeGames.jsp			
Comment			
This function, apart from displaying the games that the connected User has added to his List, is also used to display the games of other Users' Lists within the System.			

getUsersPlayingTheGame

Entrance	Reuse	Is used from	HTTP METHOD
Game's name	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all Users who have Listed the given game title.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getUsersPlayingTheGame/{name}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Search/SearchSelection.jsp			
Comment			
This Web Service, you use to find people with common interests, therefore brings the Users together resulting in easier finding a company for the game that each User likes.			

2.2.4. Publications Services

addAPost

Entrance	Reuse	Is used from	HTTP METHOD
User's ID Post's content Post's date and time	NO	Users	Post
Returns			
-			
Description			
Adds a new User Post to the Information System.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/addAPost/{userID}/{content}/{dateTime}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/UserProfile.jsp			
Comment			
Each User will, through his profile, upload Posts, which all Game Society Users will be able to see.			

getAllPosts

Entrance	Reuse	Is used from	HTTP METHOD
User's ID	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all Posts of a User sorted in descending order relative to the time they were uploaded.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getAllPosts/{userID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/UserProfile.jsp			
Comment			
Each User will have all their Publications collected in their profile.			

getHomeScreenPosts

Entrance	Reuse	Is used from	HTTP METHOD
-	YES	Users	GET
Returns			
JSON Array			
Description			
Returns all Posts all Posts of all Users within the Information System sorted in descending order relative to the time they were uploaded.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getHomeScreenPosts			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/HomePage.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
This feature will be used as the Home Page for the Game Society. So that all Users have the opportunity to be visible to the rest of the public and to socialize. It will also be used to find a Post that a particular User wants to see.			

deletePost

Entrance	Reuse	Is used from	HTTP METHOD
Post's ID	YES	Users/Admins	DELETE
Returns			
-			
Description			
Deletes the Post that has the ID given as input.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/deletePost/{postID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/HomePage.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/UserProfile.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
The function of deleting a Post is allowed to ordinary Users wherever they come across their own Post. But Administrators have this option, in all Information System Publications.			

2.2.5. Feedback Services

addAComment

Entrance	Reuse	Is used from	HTTP METHOD
User's ID (commenting) Post's ID	NO	Users	Post
Comment's text			
Comment's date and time			
Returns			
-			
Description			
Adds a comment, of a user, to a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/addAComment/{userID}/{pos tID}/{text}/{dateTime}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
Any User may comment on any Post on Game Society.			

deleteComment

Entrance	Reuse	Is used from	HTTP METHOD
Comment's ID	NO	Users/Admins	DELETE
Returns			
-			
Description			
Deletes a user's comment from a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/deleteComment/{commentID }			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
Each User can delete any of his Comments. But an Administrator can delete any comment of any User.			

showComments

Entrance	Reuse	Is used from	HTTP METHOD
Post's ID	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all comments of a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/showComments/{postID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
This function is used so that Users can see the comments that exist on the Information System publications.			

2.2.6. Upvote (Likes) Services

likeOrDislike

Entrance	Reuse	Is used from	HTTP METHOD
User's ID (liking)	NO	Users	POST
Post's ID			
Returns			
-			
Description			
Adds (if none) or removes (if any) a User's Like on a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/likeOrDislike/{userID}/{postID }			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
Every user can like every publication, but can also take it back.			

getAllUsersLiked

Entrance	Reuse	Is used from	HTTP METHOD
Post's ID	YES	Users	GET
Returns			
JSON Array			
Description			
Returns all Users who have Liked a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getAllUsersLiked/{postID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Likes.jsp			
Comment			
This function is used by the system in Post.jsp to find whether the user who is logged in has Liked the Post that is displayed or not. But it is also used in Likes.jsp to show all users who have Liked the Post they were looking at grouped together.			

getAmountOfLikes

Entrance	Reuse	Is used from	HTTP METHOD
Post's ID	NO	Users	GET
Returns			
JSON Object			
Description			
Returns the total number of likes for a Post.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/getAmountOfLikes/{postID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Post/Post.jsp			
Comment			
This function is used so that the User does not need to count the total number of people who liked a Post.			

2.2.7. Messaging Services

sendAMessage

Entrance	Reuse	Is used from	HTTP METHOD
User's ID (logged in)	NO	Users	Post
User's ID (friend user)			
Message's content			
Message's date and time			
Returns			
-			
Description			
Sends a message from a User to Another User of the Information System.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/sendAMessage/{theUserID}/{ friendUserID}/{content}/{dateTime}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/OtherUser/Chat.jsp			
Comment			
Each User can send messages to other Users of the Information System, through the profile of the other Users, in order to consult them about their plans for their common interests or even for their own possible free discussions.			

showMessages

Entrance	Reuse	Is used from	HTTP METHOD
User's ID (logged in)	NO	Users	GET
User's ID (friend user)			
Returns			
JSON Array			
Description			
Returns the last 10 messages of two Information System Users, sorted in descending order.			
Endpoint			
http://localhost:8080/GameSociety/rest/GameSociety/showMessages/{theUserID}/{ friendUserID}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/Profile/OtherUser/Chat.jsp			
Comment			
Each User will be able to view their last messages with other Game Society Users so that they can remember what they have been up to in their conversation.			

2.3. External Web Services/ Microservices

The source of the external Services Used in the System is the following:

https://www.freetogame.com/api-doc

Free To Game

Entrance	Reuse	Is used from	HTTP METHOD
-	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all free video games			
Endpoint			
https://www.freetogame.com/api/games			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/ShowFreeGames/FreeGamesSelection.jsp			
Comment			
This feature exists so that Users can potentially find new and easily accessible games, as this is one of their hobbies, since they participate in the Game Society.			

Free To Game For Platforms

Entrance	Reuse	Is used from	HTTP METHOD
Game's platform	NO	Users	GET
Returns			
JSON Array			
Description			
Returns all free video games for the input platform			
Endpoint			
https://www.freetogame.com/api/games?platform={platform}			
Use in the System			
http://localhost:8080/GameSociety/Pages/HomePage/ShowFreeGames/FreeGamesSelection.jsp			
Comment			
This feature exists so that Users can potentially find new and easily accessible games, as this is one of their hobbies, since they participate in the Game Society. The difference with the previous function is that now they can search faster for games that will be on their preferences. In case no game is found for the imported platform, then this Web Service will return a JSON Object.			

3. System User Manual

3.1. Technologies

3.1.1. Languages

The following technologies were used for the implementation of the Information System:

Java Server Pages (JSP)

.jsp pages were used to produce the front-end part of the information system. These pages contain html and java code as they make closures in .css files.

Java

The Java language was used in order to connect the Database and create and call the various Web Services that exist within the Information System.

Hypertext Markup Language (HTML)

The html language was used in order to display information to the Users of the System.

Cascading Style Sheets (CSS)

This language is the stylistic part of the information system.

3.1.2. Libraries

In Game Society various libraries were used to complete the Information System.

*java.sql.**

This library was used for the purpose of creating SQL Queries, as well as SQL Statements and executing them in the implemented Web Services.

*java.ws.rs.**

This library was used to build the endpoints of the implemented Web Services of the Information System. It was also used to create java annotations to provide metadata.

*org.json.**

This library was used so that JSON objects and tables can be provided through Web Services for better organization and management of the information transferred through them.

java.time.LocalDateTime

This library was used to find the real world time to be added as data to User comments and posts.

java.time.format.DateTimeFormatter

This library exists to manage the time and date format obtained from the "java.time.LocalDateTime" library.

3.1.3. Programs

Eclipse IDE for Enterprise Java Developers
Used Eclipse to Create Dynamic Web Project

Postman
Postman was used to check the correct operation of the implemented Web Services as well as the Third Party Web Services.

MySQL Workbench
The MySQL Workbench was used for reasons of speed and ease in checking the correct operation of the Database in accordance with the implemented Web Services.

3.2. Database – MySQL

MySQL was used as the Database for Game Society. MySQL was chosen in relation to MongoDB as with its use there is a SCHEMA and it is possible to predict the fields of the records that are given in response to Queries. The Information System Database is shown in Figure 2.

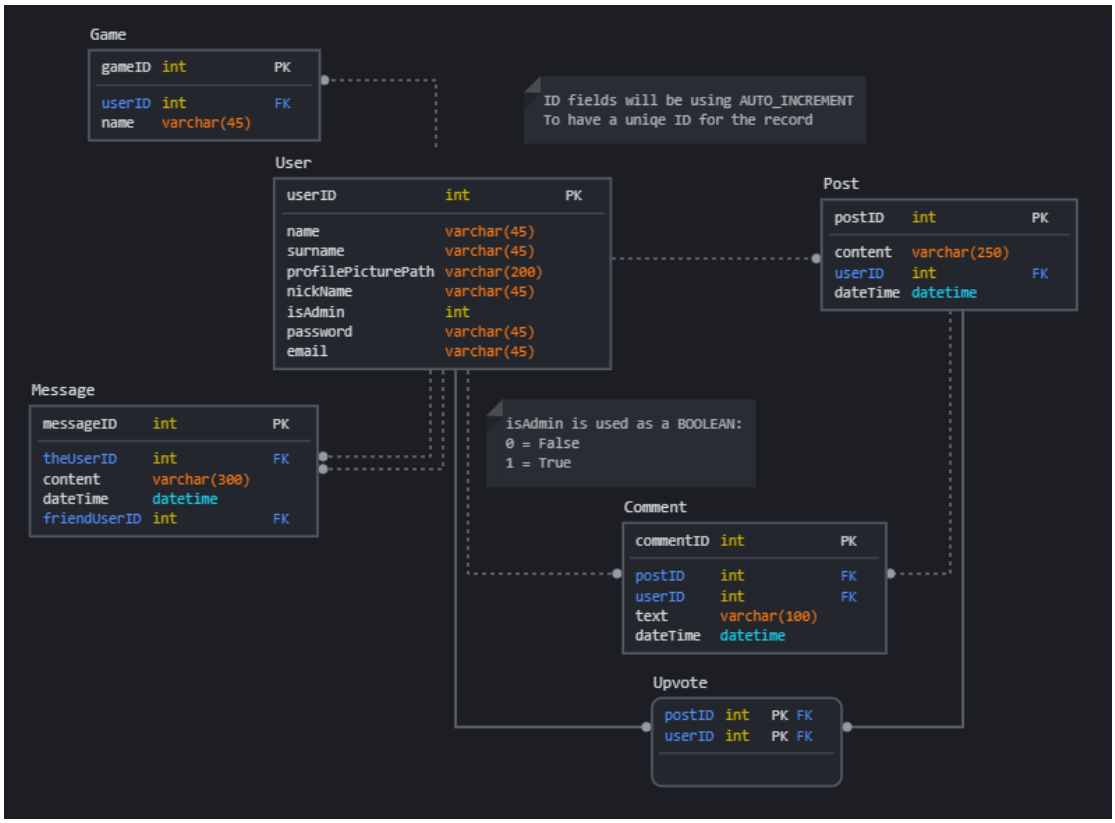


Figure 2: Game Society Database (gamesocietydb)

Below is an explanation of what you are representing in Figure 2

The Tables of the Base are shown in a rectangular parallelogram and the name of the corresponding Table is displayed on the top left of them (Parallelograms that do not have a name are notes). Each table is divided into two parts, the primary keys and the rest of its children. To the left of the Tables are the names of the fields and to the right their types respectively. The lines indicate a 1:N relationship – where N is the side where the white dot is present. Solid lines mean that the foreign key participates in the unique key of the table, while dashed lines mean that the unique key does not contain the foreign key.

3.2.1. User

Users of the Information System will have the following information:

- userID: A unique identifier for each User (Integer)
- name: The name of each User (Alphanumeric – size 45)
- surname: The surname of each User (Alphanumeric – size 45)
- profilePicturePath: The path of each User's profile picture (Alphanumeric – size 200)
- nickName: The nickname of each User (Alphanumeric – size 45)
- isAdmin: A flag indicating whether the User is an Administrator or not (Integer 0/1)
- password: The password of each User for the System (Alphanumeric – size 45)
- email: The email of each User (Alphanumeric – size 45)

3.2.2. Game

Each User can have multiple games . Each game consists of:

- gameID: A unique identifier for each Game (Integer)
- userID: The unique ID of the User who has entered this game –**foreign key** (Integer)
- name: The title of the respective game (Alphanumeric – size 45)

3.2.3. Message

Each User can send multiple messages. Each User can receive multiple messages . Each message contains the following information:

- messageID: A unique identifier for each Message (Integer)
- content: The content of the Message (Alphanumeric – size 300)
- dateTime: The time and day the message was sent by the sender (Datetime)

- theUserID: The unique ID of the User who has sent this Message –**foreign key**(Integer)
- friendUserID: The unique ID of the User receiving this Message –**foreign key** (Integer)

3.2.4. Post

Each User can upload multiple Posts. So Posts have the following fields:

- postID: A unique identifier for each Post (Integer)
- content: The content of the Post (Alphanumeric – size 250)
- dateTime: The time and day that each Post was uploaded (Datetime)
- userID: The unique identifier of the User who uploaded the Post – **foreign key**(Integer)

3.2.5. Comment

Each User can make several Comments. Each Post can contain multiple Comments . Therefore, the Information System Comments will contain the following information:

- commentID: A unique identifier for each Comment (Integer)
- postID: The unique Post ID that this Comment exists on –**foreign key** (Integer)
- text: The content of the Comment (Alphanumeric – size 100)
- dateTime: The time and day the Comment was uploaded (Datetime)
- userID: The unique identifier of the User who uploaded the Post – **foreign key**(Integer)

3.2.6. Upvote

Each User can make many likes. Each Post can get likes from many Users . For this, the Upvotes information will contain as a key the following external keys only:

- postID: The unique Post ID that exists this Upvote –**foreign key**(Integer)

- userID: The unique ID of the User who made the Upvote –**foreign key** (Integer)

3.3. User Manual

The following User Guide contains images from the Google Chrome browser.

3.3.1. Install, Run and Open

Game Society is installed by following the steps below:

1. Installation of the given Database (It is not a mandatory step in the case that we want the System to start only with the default Administrator)

2. Import the Game Society folder into Eclipse
3. Open the following path:
   Game Society/WebContent/Pages/welcome/login.jsp
4. Run login.jsp
5. Then suggest opening a browser such as Google Chrome or Mozilla Firefox so that the styles of the system pages can be seen.
6. Go to the following URL: http://localhost:8080/GameSociety/Pages/welcome/login.jsp

Now that we see Figure 3 below in the browser, then we have opened the System normally.

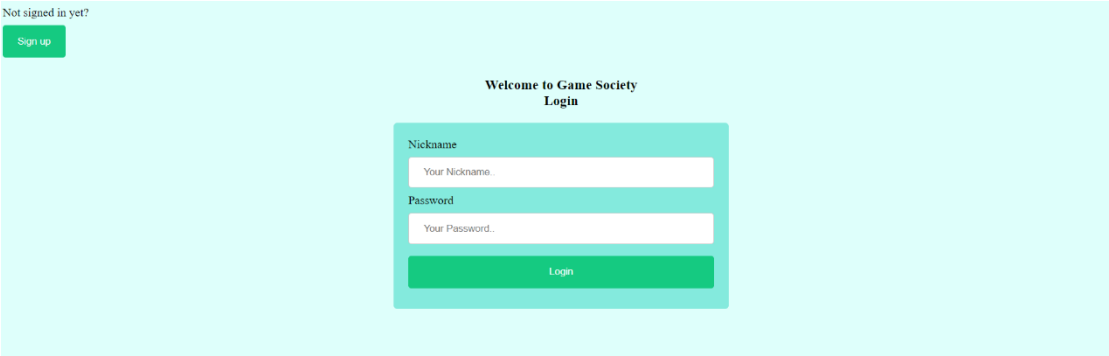


Figure 3: Log in page

3.3.2. The User's Area

Game Society Users can perform the following functions:

- Realizing their registration in the System.
- Introduction to the system with their details.
- Exit the System
- They can see all Posts of all Users in the System.
- They can search for Users based on: o
  of their nickname.
  - o of the games they play.
- They can see all the known free games available on Internet.

- They can search all known free games that exist online based gaming platform.

- Adding game titles that are playing on the System.
- He can see the game titles he has added to the System.
- Delete game titles he has added to the System.
- They can see their profile.
- They can see the profile of other Users.
- Changing their details in the System.
- Deleting their Account from the System.
- Create a Post within the System.
- Deleting some of their Posts from the System.
- They can see all their Posts on their profile collected.
- They can see all of another User's posts collected at his profile.

- Open Post they want to see more details about it
- Create Comments on Posts.
- Deleting their own Comments from Posts.
- They can see all Comments on Posts.
- They can doLike Posts.
- They can take it offLike them from already Liked Posts.

- They can see how many have doneLike a Post.
- They can see who has doneLike a Post.
- They can send a message to any other User.
- They can see their messages with other Users.
- They can see what game titles another User is playing which he has added within the System.

Login to Game Society

From the Log in page we can, if we have not registered, press the "Sign up" button on the top left and be transferred to the page for our registration which is shown in Figure 4. After successfully entering our information, we will be transferred to Log in page again to enter the System.

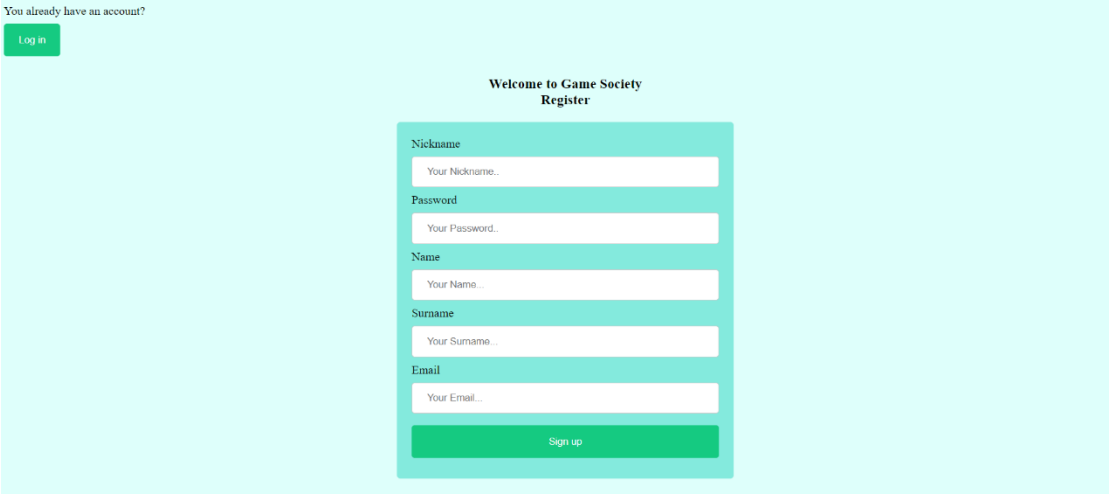


Figure 4: Registration Page

Homepage

After entering our information correctly on the Log in page (Figure 3), then the Homepage will be shown to us which is also shown in Figure 5. On this page we can see all the Publications made by Game Society Users. The posts are sorted so that they are the most recent to the top. At the top of each page, as long as we are connected to an account, the rest of the pages we have access to as Users (Home Page, Search Users, Free Games, Games I Play, Log out, Profile) are shown.

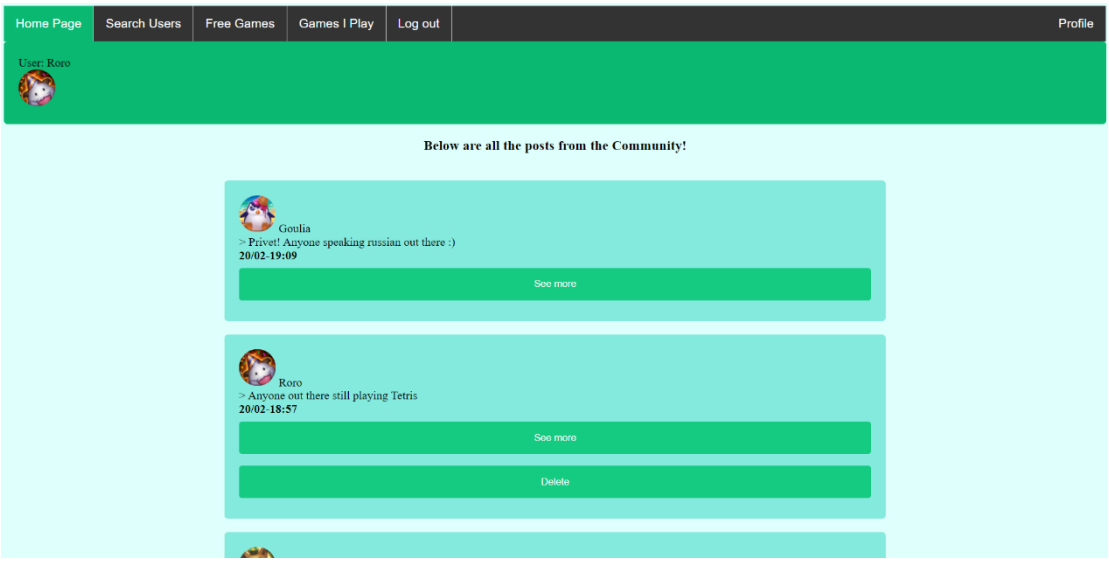


Figure 5: Homepage

On the top left, we see some basic elements of the Account from which we are connected (Nickname and profile picture).

Search Users

By clicking on the Search Users tab, a form appears (Figure 6) which gives us the following two options:

- Search all Users similar to a nickname that will we import
- Search all Users playing a game title that will we import

After giving our input and pressing the corresponding button, the results are shown below the form.

Searching for example the English letter "n" in "Search by Nickname" shows us the results of Figure 7. The Users that have appeared contain the letter "n" in them.

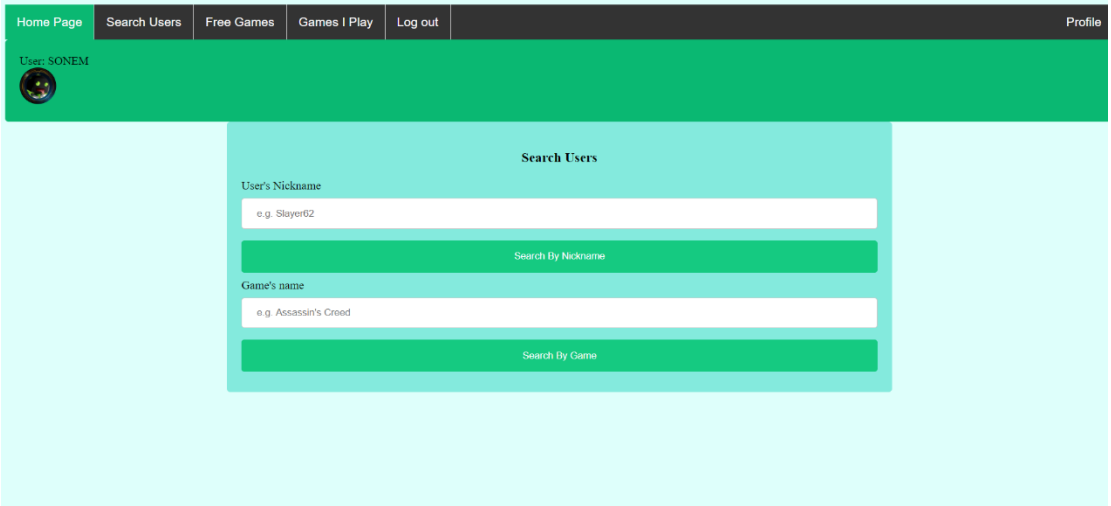


Figure 6: Search Users

For each User that appears in the searches, by clicking on the "Go to Profile" button we can go to the profile of this User and perform functions explained in its chapter**" Profile»**.

The search by game title is based on the titles that each User has added to their list, which list you explain in the chapter**"Games I Play»**.

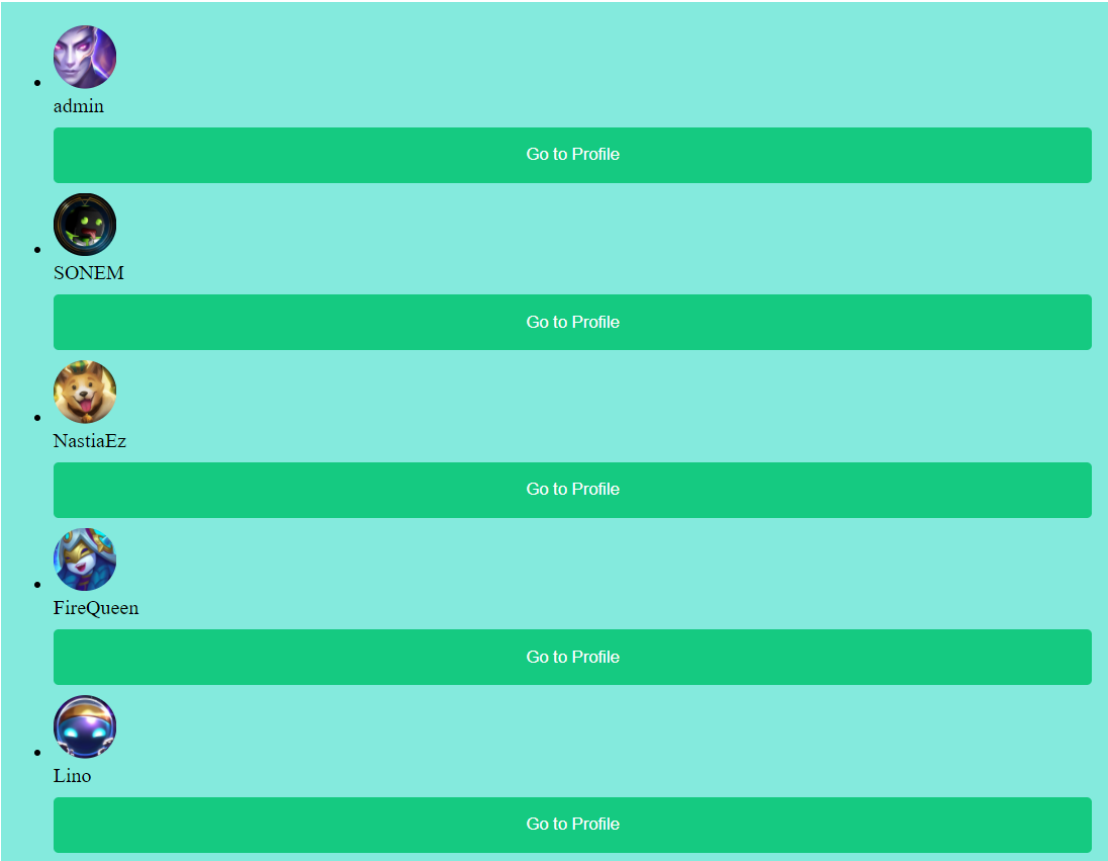


Figure 7: Nickname Search Result with the English letter "n"

Free Games

By clicking on the "Free Games" tab we go to a form (Figure 8) in which we have the following two options:

- See all popular free games available on the internet
- To find for a platform that we will introduce, which popular ones free games are there in it.

Similarly to "Search Users", after pressing the corresponding button, our results will appear below the form. For each game it will show:

- Onehis thumbnail.
- Onelink for the game page.
- The title of the game.

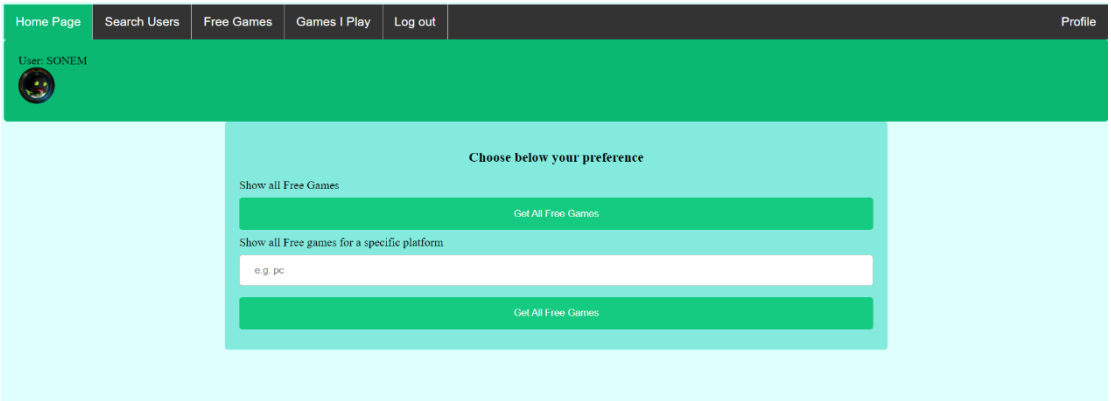


Figure 8: Free Games

Games I Play

In the "Games I Play" tab, the List of the games we have added appears, which each User declares to be playing (Figure 9). Specifically, User SONEM in the image plays games such as "League of Legends", "Minecraft" etc.

The User has the option for each entry, by pressing the "DELETE" button, to delete it from his List. Still in the form that appears to him, he can enter games in his List. These games are the ones he plays and wants other Users to be able to search for him through them in the tab**"Search Users»**.

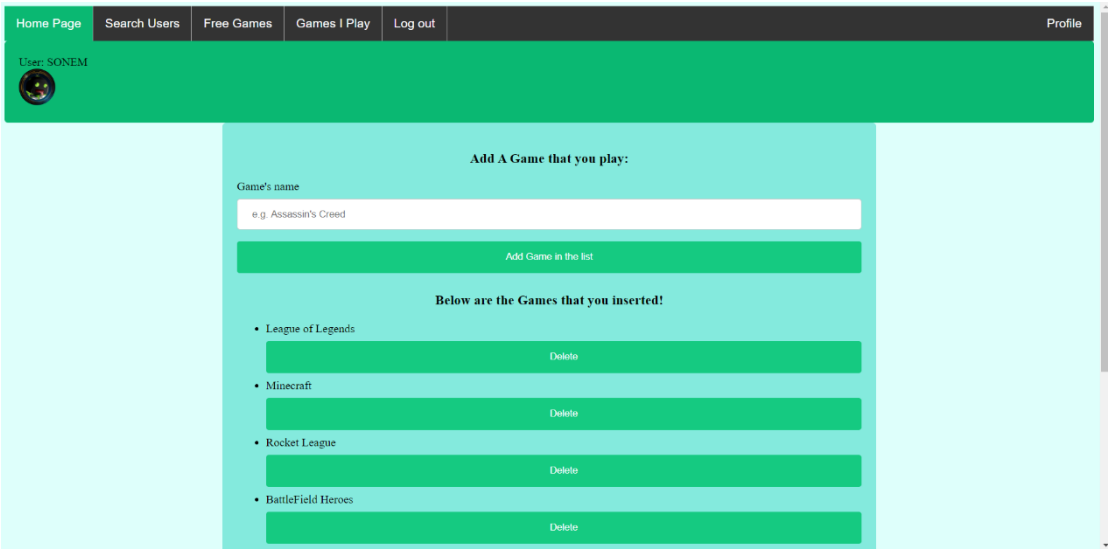


Figure 9: Game I Play

Log out

By pressing the "Log out" tab, the User exits from his connection to the system and is shown the form to log in.

Profile

When the User clicks on the "Profile" tab (top right), he can see his profile, along with the Publications he has made (by scrolling down). This page is also displayed for other Users when we click on "Go to Profile" on the page**"Search Users»**.

In the case that we have entered our own Profile then we see a page, as in Figure 10. On it we can create our own Publication and make it a Post so that the rest of the Game Society Users can see it. We can still click on "Edit Profile" to change information about our Account which will be explained further in the chapter**"Edit Profile»**.

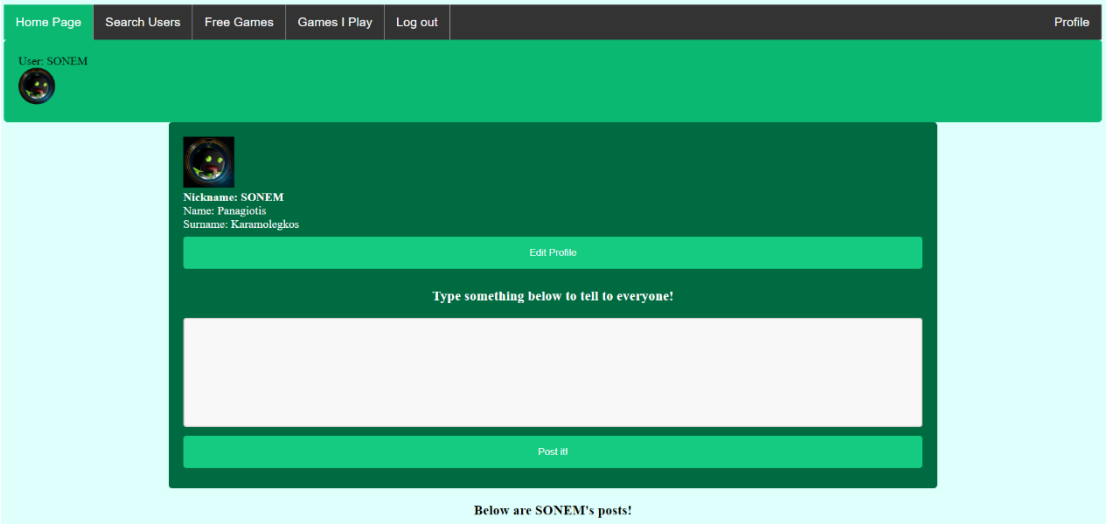


Figure 10: Our Profile

If we have entered the profile of another User, then we see a page, like Figure 11. On this page we can choose whether to chat with this User or if we want to see all the games that he has declared that he is playing. The chat will be analyzed in the chapter**"Chat»**.

To see what games the other User is playing, it is enough to click on the "See what Games this User is playing!" button. This will take us to a page, like Figure 12. Through this page, we can see the games that this User sees, but also go back to his Profile, by pressing the "Go Back" button.

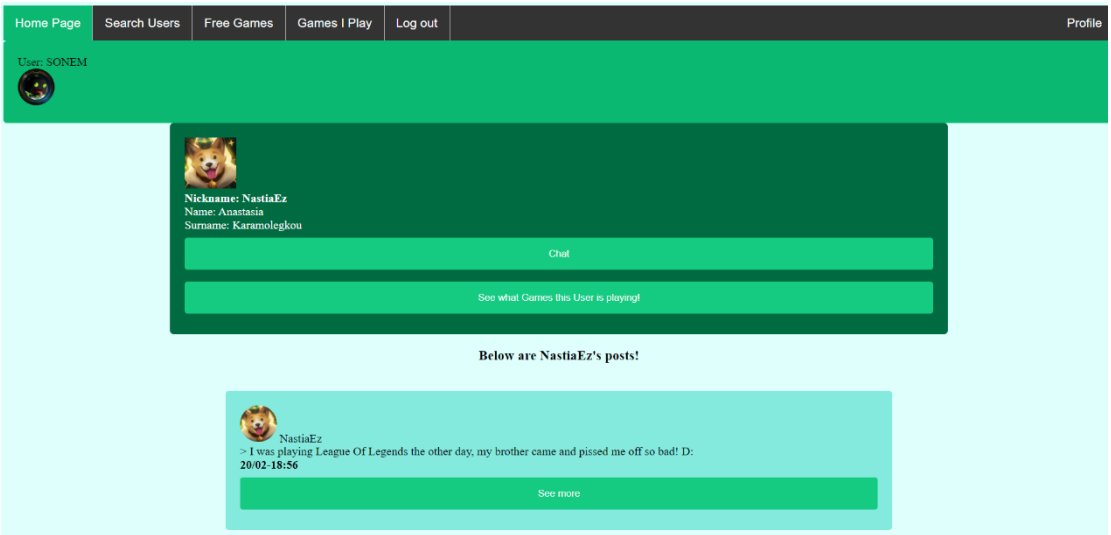


Figure 11: Another User's Profile

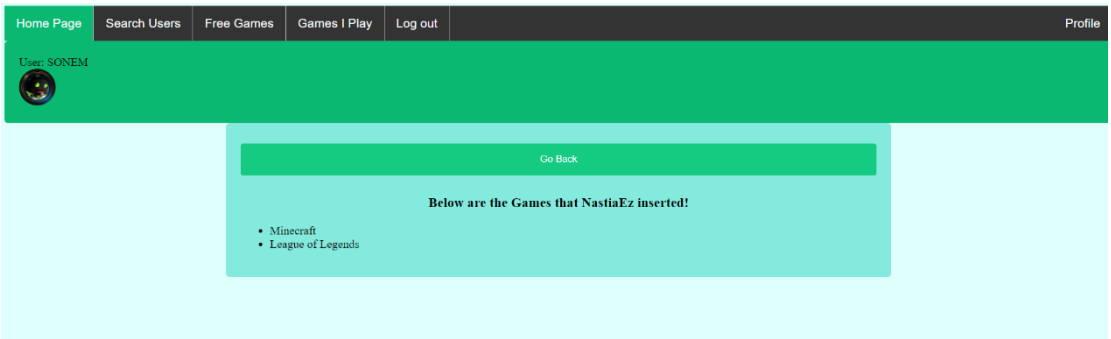


Figure 12: The games another User is playing

Edit Profile

After pressing the "Edit Profile" button from our Profile, we go to a page like Figure 13. In it we see information that we have entered into the System during registration. Here we can change the password, first name and last name that we have declared. Even by clicking on one of the images shown, we can change our profile image. If we scroll down to the bottom of the page, we can see the "Delete My Account" button (Figure 14) to delete all our comments, discussions, posts, games we have added, and also the profile us from the Game Society.

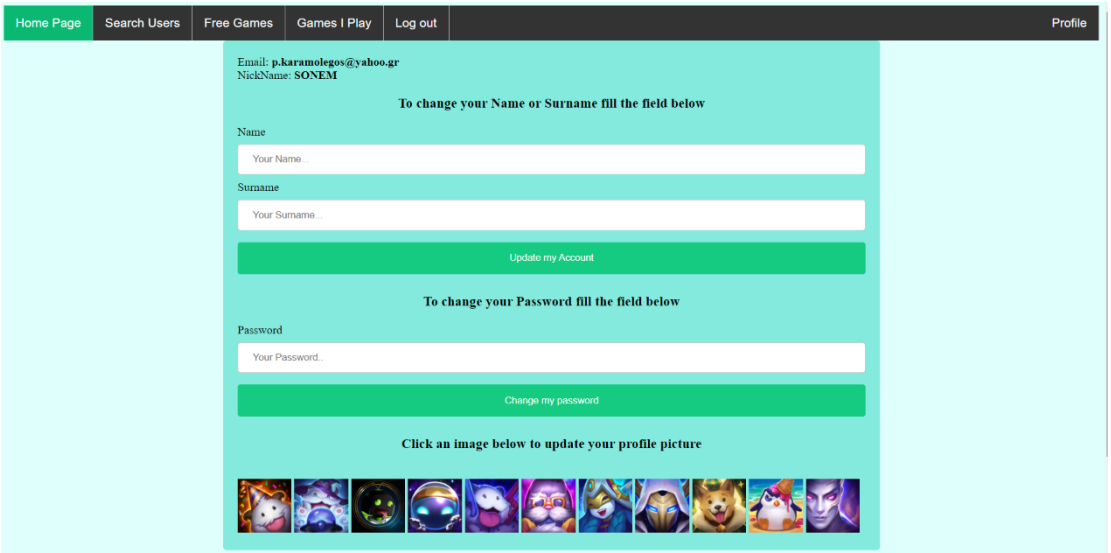


Figure 13: Edit Profile

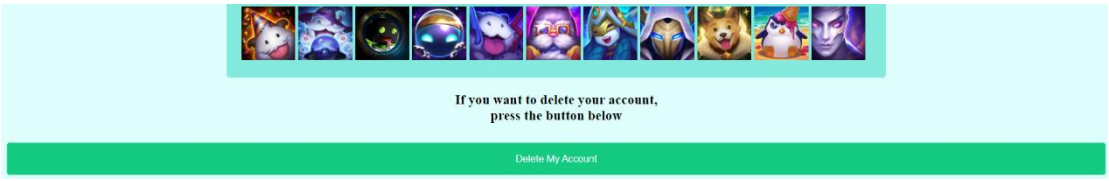


Figure 14: "Delete My Account" at the bottom of "Edit Profile"

Chat

After clicking on the "Chat" button from another User's Profile, we are transferred to a page, as shown in Figure 15. On this page we can see our chat with this User and we can go back to his Profile by pressing the button " Go Back'. This page will show our last 10 messages with the other User. By Scrolling down (unless we haven't sent messages yet with this User) we can create a message and send it to the other User (Figure 16). Users' messages are sorted so that the most recent message appears at the bottom of the conversation, so that Users can refer back to the latest messages if they wish and read them from top to bottom.

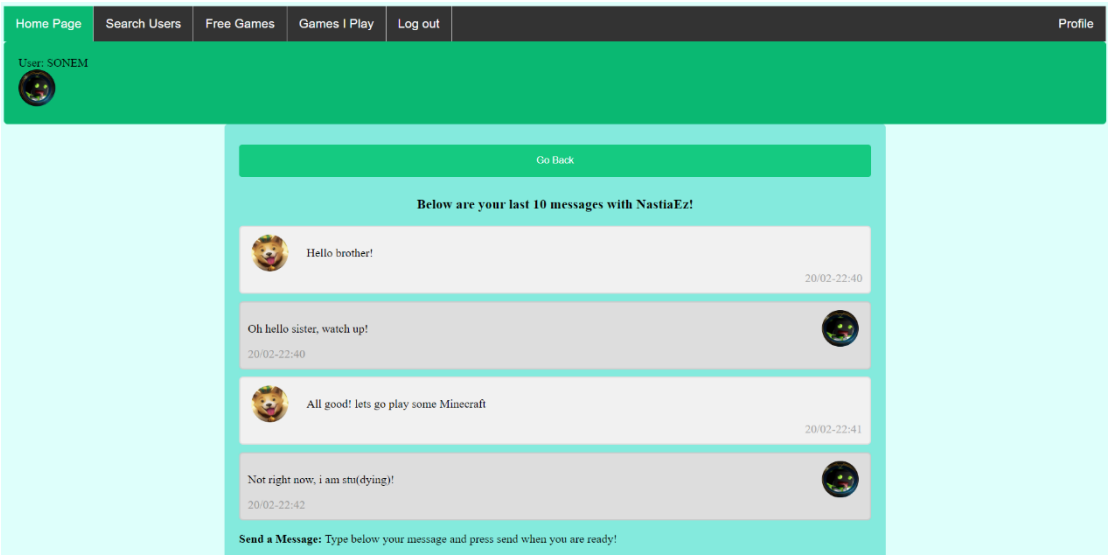


Figure 15: Chat

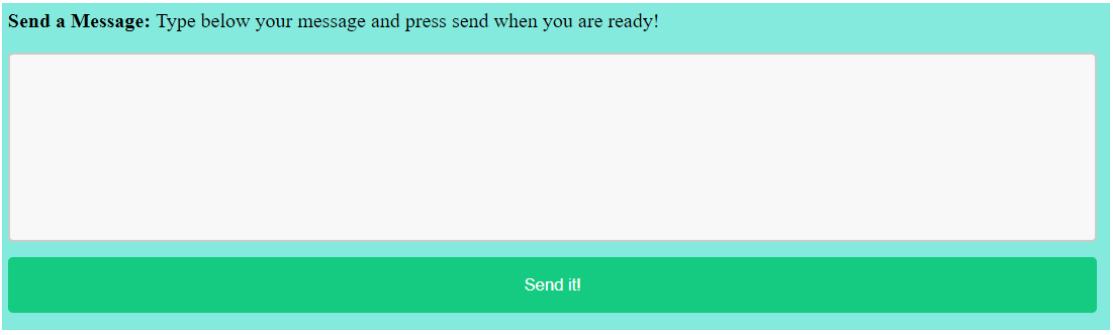


Figure 16: The form for creating and sending a message

Posts

In the**"Homepage»**and in every**"Profile"**of Users the Posts are shown their. Where Posts are displayed there is a 'See more' button within the post (Figure 17). Clicking this button will take you to this Post's page to see more information about it. However, whenever we come across our own publication, the "Delete" button (Figure 18) appears along with it so that by pressing it we can delete all the content of this Publication.

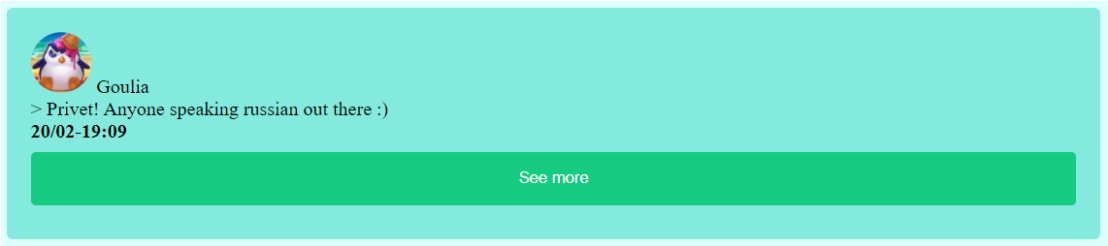


Figure 17: One Post

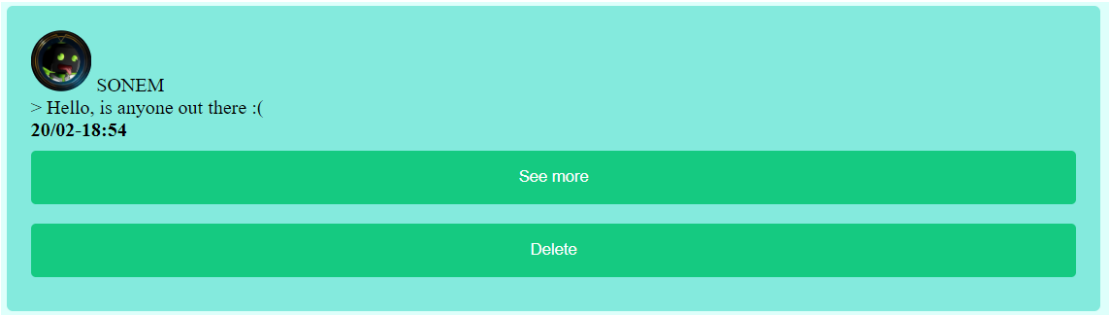


Figure 18: A Post of the User connected to the Game Society

After we have clicked on "See more" a page will appear, like the one shown in Figure 19. In the case that the Post is not ours, then there will be no "Delete" button, as it would be absurd to be able to delete it Another User's Post.

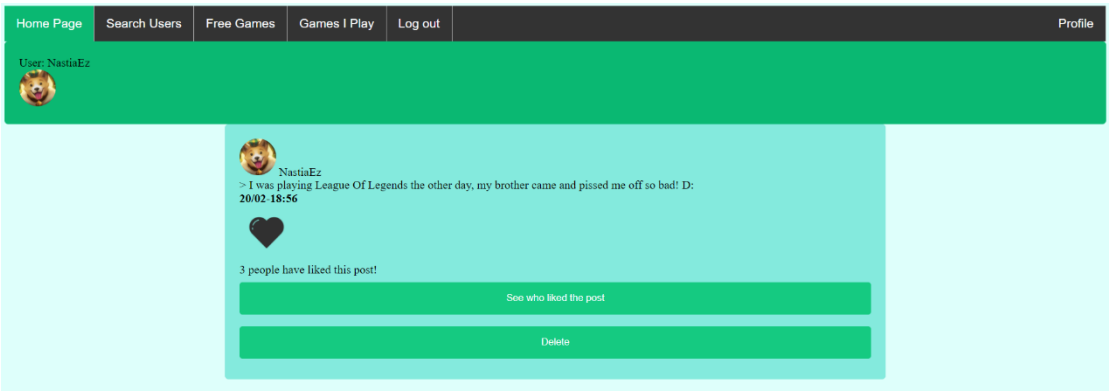


Figure 19: The page of a Post

On this page we can do the following:

- To doLike or "take back" our Like from the Post - by clicking on the heart image.
    - Black heart means we haven't Liked. Red heart
    - means we have Liked.
- Let's see how many people have doneLike the publication (in Figure 19 it is three)
- Let's see who has doneLike the post – by clicking on the "See who liked the post" button (explain further in the chapter**"See who liked the post»**)
- View all Post Comments (explain further in chapter **"Comments"**)

Comments

By scrolling down on the page of a Post, the Comments appear (Figure 20). There we can see the comments of the Post, delete our own comments, as well as add new comments. In Figure 20, the comment that has the profile image on the right side belongs to the account from which we have connected to the System and therefore we are allowed to delete it by pressing the "Delete this comment" button.

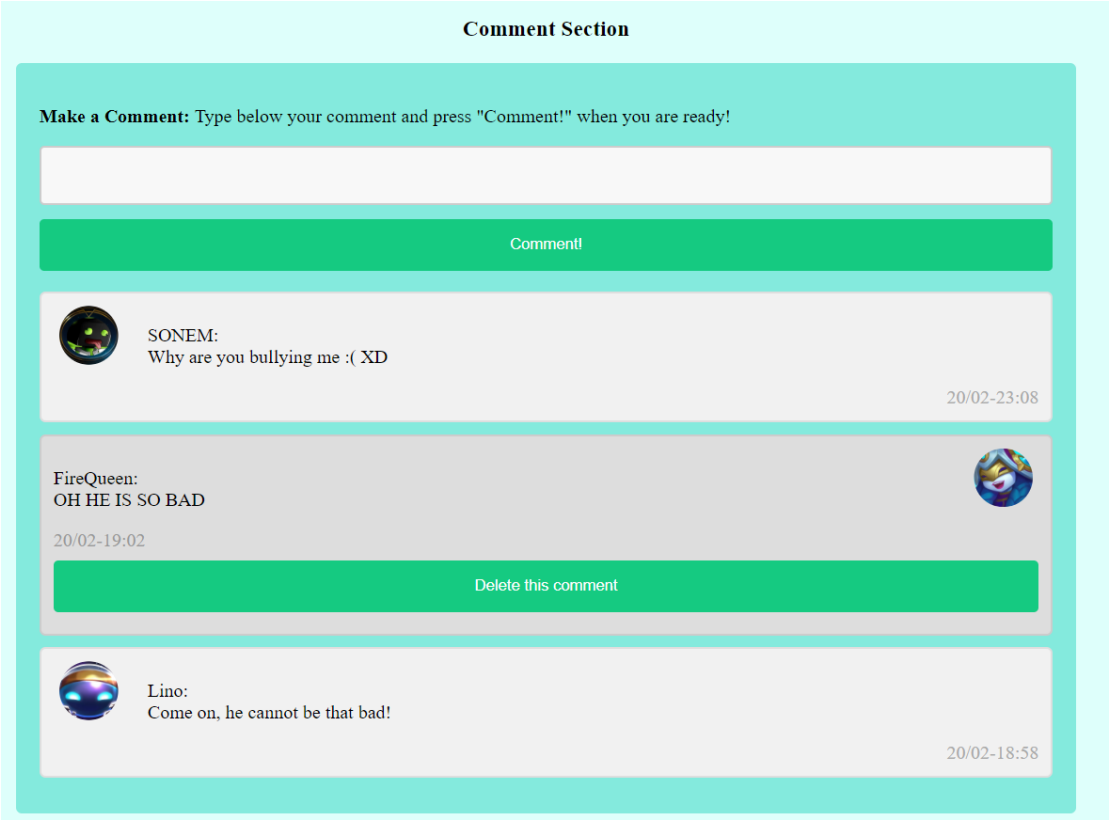


Figure 20: Comment Section

See who liked the post

By pressing the button "See who liked the post" from the page of a Post we are directed to a page like Figure 21. On it, we can see all the people who have liked the Post and by pressing "Go Back" we can go back to Publication page (Figure 19).

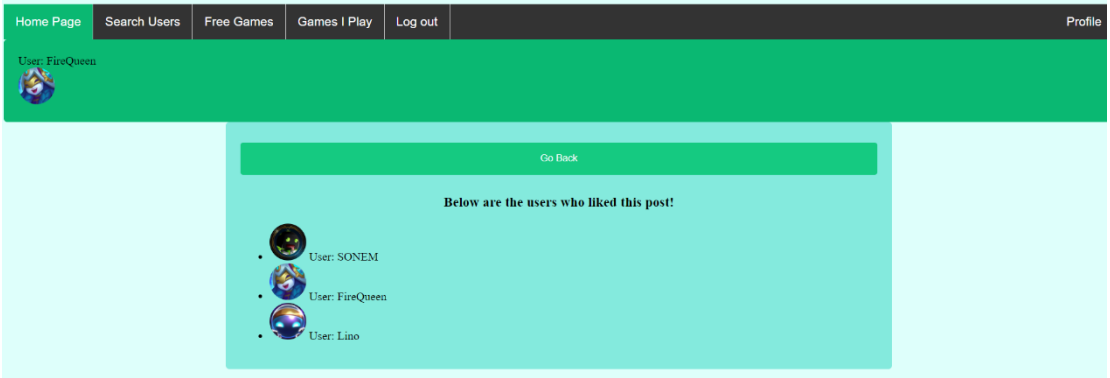


Figure 21: See who liked the post

3.3.3. The Admin Area

System Administrators can do anything that regular Users can do, but they have some additional abilities such as:

- Delete other Users fromGame Society.
- Deleting Posts of other Users from the System.
- Deleting Comments of other Users from the publications.
- Change the rights of ordinary Users from ordinary Users to Administrators and vice versa.
- They can see all System Users together.

In the case that the Database does not already exist, then an Administrator is automatically built with the following elements as Default:

- Name:admin
- Last name:admin
- Pen name:admin
- Code:admin
- E-mail:admin@admin.com

The above Administrator will be rebuilt whenever there is no other Administrator within the System automatically.

Administrators, unlike Users, can always have the following buttons, even on other Users' objects:

- Delete the Post «Delete»
- Delete Comment «Delete this Comment»

Show All Users

After logging in with an admin account, a floating tab is added to the Game Society menu so we can see all the Users together. This tab is 'Show All Users' and is the penultimate one from the left (shown in Figure 22).

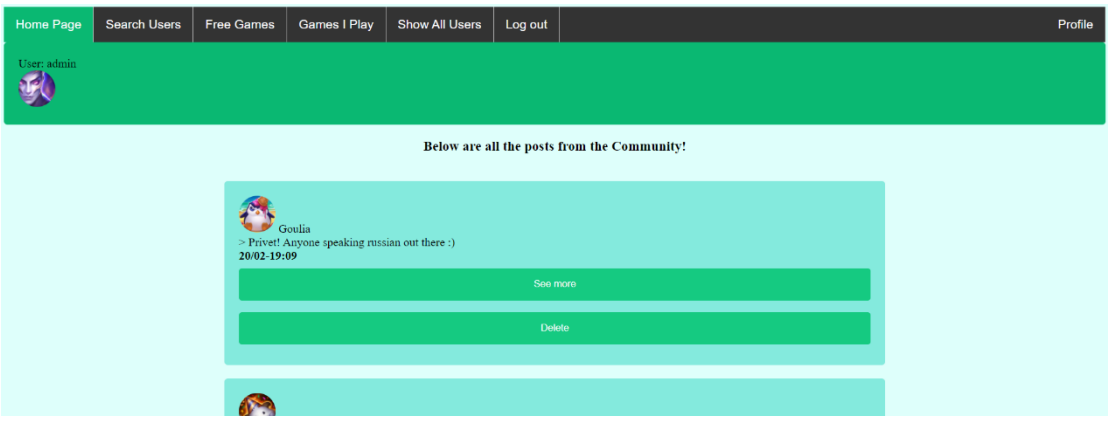


Figure 22: Show All Users tab

After clicking on this tab, a page like Figure 23 appears. From there we can quickly go to the profile of any User we want.

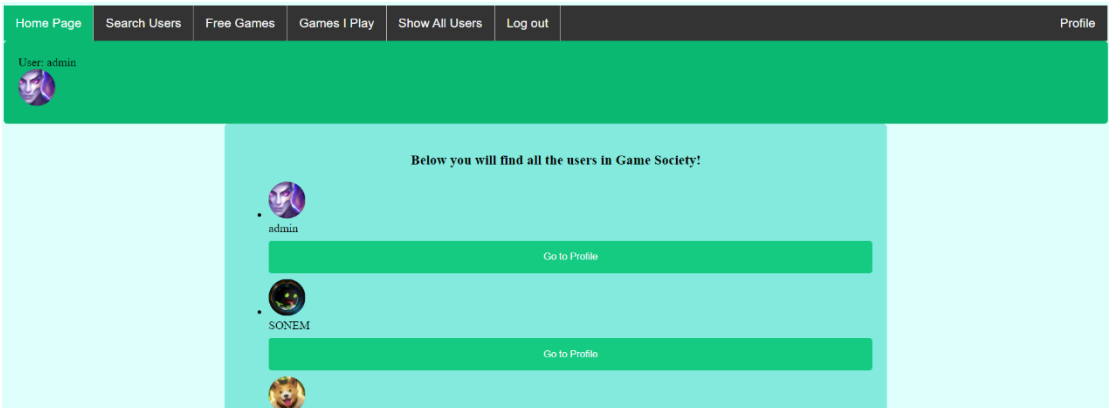


Figure 23: Show All Users

Profile Viewing – By Admins

When an Admin looks at another User's profile, they see a page like Figure 24. So we have two floating buttons.

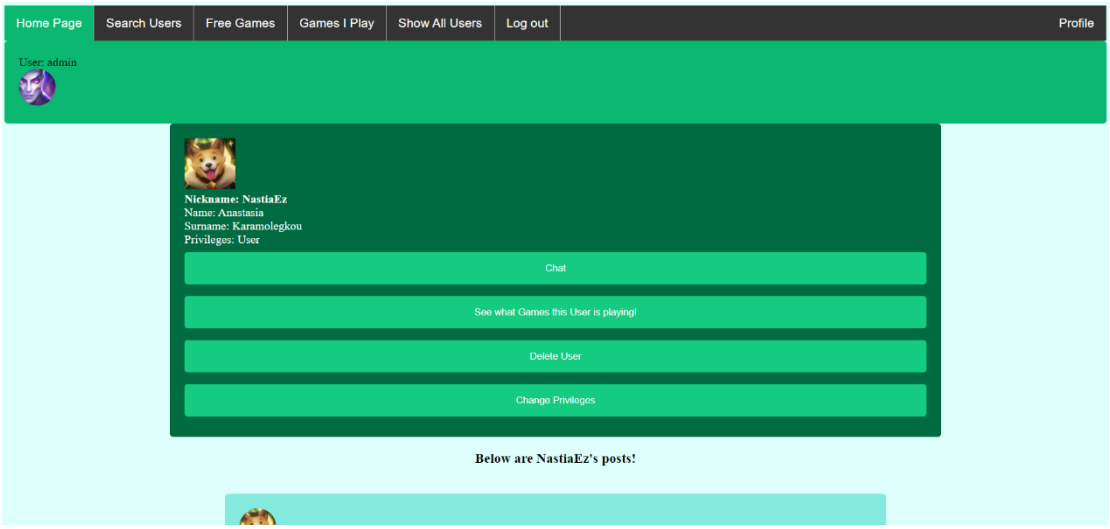


Figure 24: Profile Viewing – By Admins

If we press the button "Delete User" then that User will be deleted from Game Society, deleting all his discussions, his comments, the games he has in his List, his Posts and his profile.

By pressing the "Change Privileges" button, the User will:

- He will become an Administrator if he is a simple User
- He will become a simple User if he is an Administrator

3.4. Manual

Then the System will be explained in its technical aspect. All pages within the - Project 'GameSociety' folder located in the following path (which also appears open in Figure 25) will be analyzed for the Web Services they use:

GameSociety/WebContent/Pages

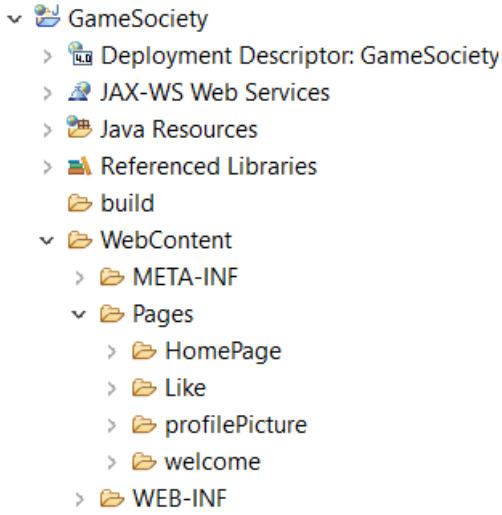


Figure 25: The path to the System pages

As mentioned in**User Manual**, to open the System correctly we must run login.jsp from the following path:

GameSociety/WebContent/Pages/welcome/login.jsp

Therefore the explanation of the System will start from that page.

Web Services that are used on multiple pages for the same reason (such as displaying the logged-in User's details at the top left) will not be explained on all pages, except those that first appear in the following manual.

All Web Services used below are explained in **chapter 2**of this report.

3.4.1. welcome

Starting from the path below, we see within it three files (Figure 26):

GameSociety/WebContent/Pages/welcome

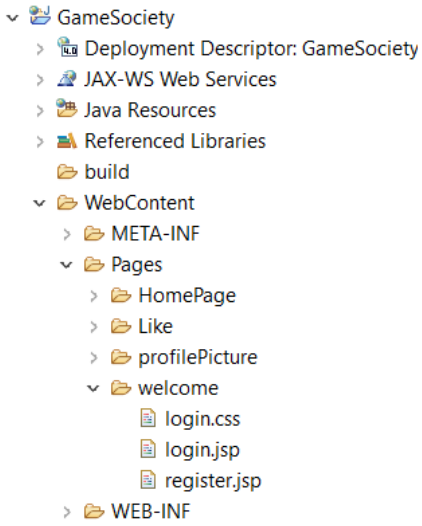


Figure 26: welcome

These three files are login.css, login.jsp and register.jsp. Any files ending in .css are simply the style selected for the system, so they will not be referenced from here on out.

login.jsp

login.jsp starts by "opening" the Database so that the User can interact with it (Figure 27). Also, in case the Base does not exist, then you create it and the default admin is also built with it. Most pages do this to ensure that the Database will be available to the User from system pages whenever he uses them.

```
7  <%
8      Client client = Client.create();
9      WebResource webResource = client.resource("http://localhost:8080/GameSociety/rest/GameSociety/testForCellarDB");
10     ClientResponse myresponse = webResource.get(ClientResponse.class);
11 %>
```

Figure 27: Opening the Bases

Figure 28 shows the Users when the Database is automatically built by the Web Service for the first time. The Administrator shown will be re-created whenever there is no other Administrator within the Base.

	userID	name	surname	profilePicturePath	nickName	isAdmin	password	email
▶	1	admin	admin	1	admin	1	admin	admin@admin.com
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 28: Building the Base

Continuing, Game Society pages are designed to move the end user to the correct page. For example, a User who is already logged in should not be able to go to login unless they log out. One such type of management is shown in Figure 29.

```
12 <%
13
14     if(session.getAttribute("userID")!= null){
15         response.sendRedirect("../HomePage/HomePage.jsp");
16     }
17
18 %>
```

Figure 29: Checking already logged in User

Before entering a User into the system, checks are made regarding the correctness of the data he provides in the form in front of him. Using the Web Service **testForUser**, the page fetches the User's data from the database based on the nickname he gives to arrange his password (Figure 30).

```
28     if(request.getParameter("login")!=null){
29         String nickName = request.getParameter("nickName");
30         String password = request.getParameter("password");
31
32         if(nickName.equals("") || password.equals("")){
33             error = "You must fill all the fields!";
34         }
35         else if(nickName.contains(" ") || password.contains(" ")){
36             error = "You must not use spaces!";
37         }
38         else{
39             client = Client.create();
40             String link = "http://localhost:8080/GameSociety/rest/GameSociety/testForUser/";
41             link+=nickName;
42             webResource = client.resource(link);
43             myresponse = webResource.accept("application/json").get(ClientResponse.class);
44             JSONObject user = new JSONObject(myresponse.getEntity(String.class));
45             if(user.length()>0){
46                 if(password.equals(user.getString("password"))){
47                     session.setAttribute("userID",user.getInt("userID"));
48                     if(user.getInt("isAdmin")==1){
49                         session.setAttribute("adminID",user.getInt("userID"));
50                     }
51                     response.sendRedirect("../HomePage/HomePage.jsp");
52                 }
53                 else{
54                     error = "The nickname or password is wrong!";
55                 }
56             }
57             else{
58                 error = "The nickname or password is wrong!";
59             }
60         }
61     }
```

Figure 30: Authentication

register.jsp

The registration of each User in Game Society is done from the welcome folder, in the register.jsp file:

On this page, after checks as to whether the User has filled out the form in all its fields correctly, a check is made of the User's existing existence in the Information System through the**testForUser**(for the possible existence of his nickname) and **testForEmail**(for the possible existence of thise-mail). After it is detected that this User does not exist and has filled in the form correctly, then it is entered into the system through the Service**addAUser**. These calls are shown in Figure 31.

Figure 32 shows the introduction of a new user to the Information System. When a User registers, they are initially given the status of a simple User and the image named 1.png as their profile picture from the following folder:

GameSociety/WebContent/Pages/profilePicture

```
46          else{
47              client = Client.create();
48              link = "http://localhost:8080/GameSociety/rest/GameSociety/testForUser/";
49              link+=nickName;
50              webResource = client.resource(link);
51              myresponse = webResource.accept("application/json").get(ClientResponse.class);
52              JSONObject user = new JSONObject(myresponse.getEntity(String.class));
53              if(user.length()==0){
54                  client = Client.create();
55                  link = "http://localhost:8080/GameSociety/rest/GameSociety/testForEmail/";
56                  link+=email;
57                  webResource = client.resource(link);
58                  myresponse = webResource.accept("application/json").get(ClientResponse.class);
59                  JSONObject userByEmail = new JSONObject(myresponse.getEntity(String.class));
60                  if(userByEmail.length()==0){
61
62                      client = Client.create();
63                      link = "http://localhost:8080/GameSociety/rest/GameSociety/addAUser/";
64                      link+=name+"/";
65                      link+=surname+"/";
66                      link+=nickName+"/";
67                      link+=0+"/";
68                      link+=password+"/";
69                      link+=email;
70                      webResource = client.resource(link);
71                      myresponse = webResource.post(ClientResponse.class);
72
73
74                      session.setAttribute("registered",true);
75                      response.sendRedirect("login.jsp");
76                  }
77                  else{
78                      error = "This email already exist!";
79                  }
80              }
```

Figure 31: The closures of register.jsp

	userID	name	surname	profilePicturePath	nickName	isAdmin	password	email
▶	1	admin	admin	1	admin	1	admin	admin@admin.com
	2	Panagiotis	Karamolegkos	1	SONEM	0	12345	p.karamolegos@yahoo.gr
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 32: Inserting a new User into the Base

3.4.2. HomePage

Refer this subhead to the folder in the following path, which also appears open in Figure 33:

GameSociety/WebContent/Pages/HomePage

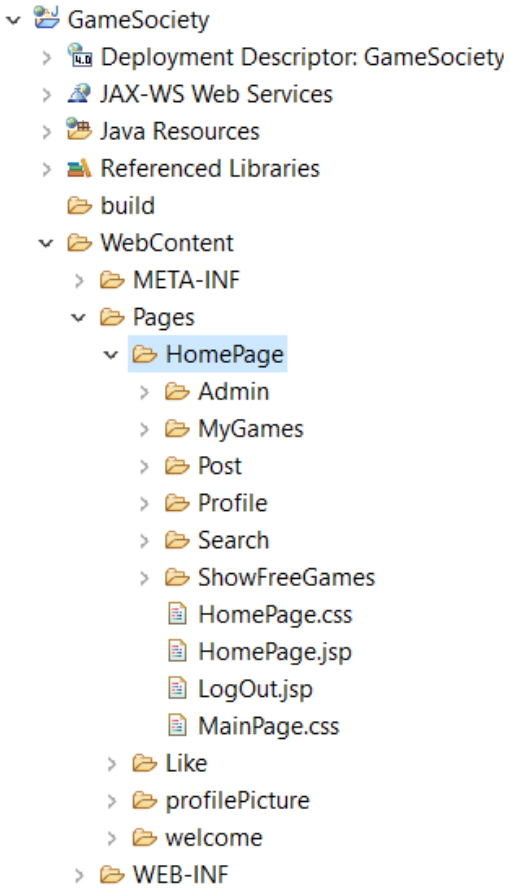


Figure 33: HomePage

In 3.4.2. HomePage.jsp and LogOut.jsp will be mentioned and the rest of the contents of this folder will be explained in chapters 3.4.3. to 3.4.8.

LogOut.jsp

This page works as a simple deletion of the user from the browser session and transfer to the log in page.

HomePage.jsp

The HomePage.jsp page is used as the "Home" of Game Society. To display the User's nickname and profile picture, the **getUser**from the lines of code shown in the Image34.

```
22      else{
23          userID = (int)session.getAttribute("userID");
24          client = Client.create();
25          String link = "http://localhost:8080/GameSociety/rest/GameSociety/getUser/";
26          link+=userID;
27          webResource = client.resource(link);
28          myresponse = webResource.accept("application/json").get(ClientResponse.class);
29          JSONObject user = new JSONObject(myresponse.getEntity(String.class));
30          nickName = user.getString("nickName");
31          isAdmin = user.getInt("isAdmin");
32          profilePicturePath = user.getString("profilePicturePath");
33      }
34  %>
```

Figure 34: Obtaining details of the connected User

All the operations that the User can perform on the Game Society pages are given in the form of a form, therefore they are detected with Selection Structures (if-else if-else) to perform their management.

For better management and easier debugging of the system, each form calls the same .jsp in which it is located so that it itself manages what will direct the user at the end of the execution of each function.

Figure 35 shows the rest of the Web Services that HomePage.jsp calls.

```
35  <%
36      if(request.getParameter("deletePost")!=null){
37          int postID = Integer.parseInt(request.getParameter("postID"));
38          client = Client.create();
39          String link = "http://localhost:8080/GameSociety/rest/GameSociety/deletePost/";
40          link+=postID;
41          webResource = client.resource(link);
42          myresponse = webResource.delete(ClientResponse.class);
43      }
44
45      JSONArray allPosts = new JSONArray();
46      if(session.getAttribute("userID") != null){
47          client = Client.create();
48          String link = "http://localhost:8080/GameSociety/rest/GameSociety/getHomeScreenPosts/";
49          webResource = client.resource(link);
50          myresponse = webResource.accept("application/json").get(ClientResponse.class);
51          allPosts = new JSONArray(myresponse.getEntity(String.class));
52      }
53
54      if(request.getParameter("goToThePost")!=null){
55          int postID = Integer.parseInt(request.getParameter("postID"));
56          session.setAttribute("postViewingID",postID);
57          response.sendRedirect("Post/Post.jsp");
58      }
59  %>
```

Figure 35: Web Services of HomePage.jsp

Whenever the deletePost parameter is detected, then the corresponding form on the page gives the ID of the corresponding Post and finally it is deleted, along with all the comments and Likes related to it through the Web Service **deletePost**(Image 36).

Whenever it is detected within the browser session that there is a User inside, then all Posts of all Users are requested so that they can then be displayed to him through**getHomeScreenPosts**.

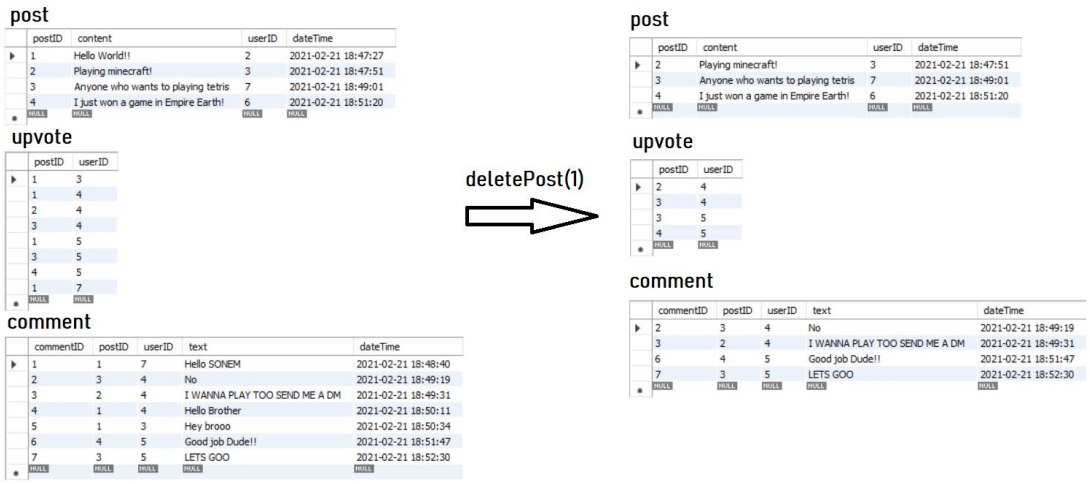


Figure 36: Database before and after Post deletion

3.4.3. Admin

ShowAllUsers.jsp

Within the following path, the ShowAllUsers.jsp page appears:

GameSociety/WebContent/Pages/HomePage/Admin

This page is displayed only for Information System Administrators and exists so that they can see all Users gathered in one place, in order to access them faster. This operation is carried out through the Web Service**getAllUsers**which returns data for all System Users, its call is shown in Figure 37.

```
<%
    if(session.getAttribute("adminID") != null){
        client = Client.create();
        link = "http://localhost:8080/GameSociety/rest/GameSociety/getAllUsers/";
        webResource = client.resource(link);
        myresponse = webResource.accept("application/json").get(ClientResponse.class);
        allUsers = new JSONArray(myresponse.getEntity(String.class));
    }
```

Figure 37: Calling getAllUsers

3.4.4. MyGames

MyGames.jsp

Within the path below, the MyGames.jsp page appears

GameSociety/WebContent/Pages/HomePage/MyGames

On this page Users can use Web Services **deleteAUserGame**,**addAUserGame**and **getAllUserGames**for its management List of their game titles. The calls of these Services are shown in Figure 38.

```
36 <%
37     if(request.getParameter("deleteGame")!=null){
38         int gameID = Integer.parseInt(request.getParameter("gameID"));
39         client = Client.create();
40         link = "http://localhost:8080/GameSociety/rest/GameSociety/deleteAUserGame/";
41         link+=userID+"/";
42         link+=gameID;
43         webResource = client.resource(link);
44         myresponse = webResource.delete(ClientResponse.class);
45 
46     }
47 
48     if(request.getParameter("addGame")!=null){
49         String name = request.getParameter("name");
50         if(name.equals("")){
51             error = "You must fill the field";
52         }
53         else{
54             client = Client.create();
55             link = "http://localhost:8080/GameSociety/rest/GameSociety/addAUserGame/";
56             link+=userID+"/";
57             link+=name.replace(' ','_');
58             webResource = client.resource(link);
59             myresponse = webResource.post(ClientResponse.class);
60         }
61     }
62 
63     if(session.getAttribute("userID") != null){
64         client = Client.create();
65         link = "http://localhost:8080/GameSociety/rest/GameSociety/getAllUserGames/";
66         link+=userID;
67         webResource = client.resource(link);
68         myresponse = webResource.accept("application/json").get(ClientResponse.class);
69         UserGames = new JSONArray(myresponse.getEntity(String.class));
70     }
71 %>
```

Figure 38: The Games I Play page Services

Figure 39 shows the import of a game named "Battlefield 3" to user ID number 2 using**addAUserGame**and then the delete it by using it**deleteAUserGame**.

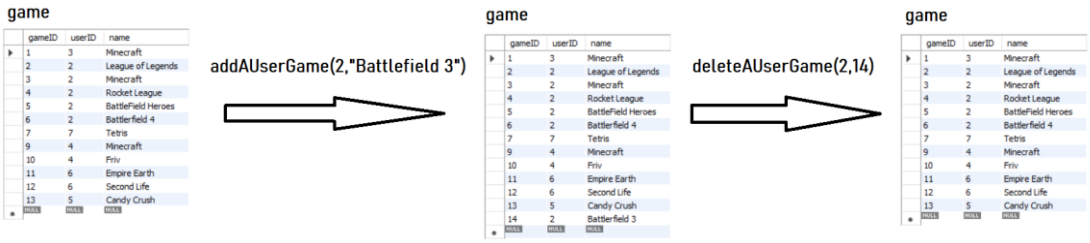


Figure 39: Insert and Delete a game

3.4.5. Post

Within the path below,

GameSociety/WebContent/Pages/HomePage/Post

the following pages appear:

- Post.jsp
- Likes.jsp

These pages display all the information about all the Publications of the System Users.

Post.jsp

In the Post.jsp page Users can:

- View a Post
- Delete the Post if they have the necessary rights
- To see if they have done it themselvesLike the displayed Post
- See how many have doneLike the displayed Post
- See all Post comments
- Add comments to the existing Post
- To delete comments if they have the necessary rights

In the event that the User proceeds to delete the Post, comment or Like, similar modifications are made to the Database as shown in Figure 36 using the Services **deletePost**(Image40), **deleteComment**(Image41),**likeOrDislike**(Image42) respectively.

```
66      if(request.getParameter("deletePost") != null){
67          int postID = Integer.parseInt(request.getParameter("postID"));
68          client = Client.create();
69          String link = "http://localhost:8080/GameSociety/rest/GameSociety/deletePost/";
70          link+=postID;
71          webResource = client.resource(link);
72          myresponse = webResource.delete(ClientResponse.class);
73      }
```

Figure 40: Deleting a Post from its page

```
75      if(request.getParameter("deleteComment") != null){
76          int postID = Integer.parseInt(request.getParameter("postID"));
77          int commentID = Integer.parseInt(request.getParameter("commentID"));
78          client = Client.create();
79          String link = "http://localhost:8080/GameSociety/rest/GameSociety/deleteComment/";
80          link+=commentID;
81          webResource = client.resource(link);
82          myresponse = webResource.delete(ClientResponse.class);
83          session.setAttribute("postViewingID",postID);
84      }
```

Figure 41: Deleting a Comment

```
55      if(request.getParameter("likeDislike") != null){
56          int postID = Integer.parseInt(request.getParameter("likeDislike"));
57          client = Client.create();
58          String link = "http://localhost:8080/GameSociety/rest/GameSociety/likeOrDislike/";
59          link+=userID+"/";
60          link+=postID;
61          webResource = client.resource(link);
62          myresponse = webResource.post(ClientResponse.class);
63          session.setAttribute("postViewingID",postID);
64      }
```

Figure 42: Add and Delete Like

When the user adds a Like, then you call the same Service as in Figure 42. To add a Comment, the lines of code shown in Figure 43 are used using the**addAComment**. Figure 44 shows the Database before and after adding a Comment.

```
94      if(request.getParameter("addAComment") != null){
95          int postID = Integer.parseInt(request.getParameter("postID"));
96          String content = request.getParameter("content");
97          if(content.equals("")){
98              error = "You must fill the field!";
99          }
100         String newline = System.getProperty("line.separator");
101         boolean hasNewline = content.contains(newline);
102         if(hasNewline){
103             error = "You may not use enter!";
104         }
105         else{
106             content = content.replace(" ","_");
107             DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
108             LocalDateTime now = LocalDateTime.now();
109             String dateTime = dtf.format(now);
110             dateTime = dateTime.replace(" ","_");
111             client = Client.create();
112             String link = "http://localhost:8080/GameSociety/rest/GameSociety/addAComment/";
113             link+=userID+"/";
114             link+=postID+"/";
115             link+=content+"/";
116             link+=dateTime;
117             webResource = client.resource(link);
118             myresponse = webResource.post(ClientResponse.class);
119         }
120         session.setAttribute("postViewingID",postID);
121     }
```

Figure 43: Adding a Comment

comment

commentID	postID	userID	text	dateTime
2	3	4	No	2021-02-21 18:49:19
3	2	4	I WANNA PLAY TOO SEND ME A DM	2021-02-21 18:49:31
6	4	5	Good job Dude!!	2021-02-21 18:51:47
7	3	5	LETS GOO	2021-02-21 18:52:30
NULL	NULL	NULL	NULL	NULL

addAComment(2,3,Lol no, 2021-02-21 21:08:31)

comment

commentID	postID	userID	text	dateTime
2	3	4	No	2021-02-21 18:49:19
3	2	4	I WANNA PLAY TOO SEND ME A DM	2021-02-21 18:49:31
6	4	5	Good job Dude!!	2021-02-21 18:51:47
7	3	5	LETS GOO	2021-02-21 18:52:30
8	3	2	LoL no	2021-02-21 21:08:31
NULL	NULL	NULL	NULL	NULL

Figure 44: Adding a Comment to the Database

The is used**getHomeScreenPosts**to be found within the results of the Post to be displayed to the User (Figure 45).

Web Services is used**showComments**and**getAllusersLiked**for view all comments but also to find if the User who has connected has pressed Like on the specific Post (Figure 46).

As you also use the**getAmountOfLikes**to be displayed to the User the sum of all Likes of the Post (Figure 47).

```
myPostID = (int)session.getAttribute("postViewingID");
session.removeAttribute("postViewingID");
client = Client.create();
String link = "http://localhost:8080/GameSociety/rest/GameSociety/getHomeScreenPosts/";
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
allPosts = new JSONArray(myresponse.getEntity(String.class));
for(int i=0; i<allPosts.length(); i++){
    JSONObject jsonObject = allPosts.getJSONObject(i);
    if(jsonObject.getInt("postID")==myPostID){
        myPost = allPosts.getJSONObject(i);
        break;
    }
}
myUserID = myPost.getInt("userID");
myProfilePicturePath = myPost.getString("profilePicturePath");
myContent = myPost.getString("content");
myDateTime = myPost.getString("dateTime");
myNickName = myPost.getString("nickName");
```

Figure 45: Finding the Post view

```
client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/showComments/";
link+=myPostID;
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
allComments = new JSONArray(myresponse.getEntity(String.class));

client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/getAllUsersLiked/";
link+=myPostID;
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
allLikes = new JSONArray(myresponse.getEntity(String.class));

for(int i=0; i<allLikes.length(); i++){
    JSONObject jsonObject = allLikes.getJSONObject(i);
    if(jsonObject.getInt("userID")==userID){
        userLikesThisPost = true;
        break;
    }
}
```

Figure 46: Getting Comments and detecting User Likes

```
client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/getAmountOfLikes/";
link+=myPostID;
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
JSONObject jsonObject = new JSONObject(myresponse.getEntity(String.class));
amountOfLikes = jsonObject.getInt("amount");
```

Figure 47: Getting the number of Likes of a Post

Likes.jsp

This page uses the Web Service**getAllUsersLiked**so that it appears all Users who have Liked the displayed Post of the Post.jsp page. It is called on the lines shown in Figure 48.

```
39      if(session.getAttribute("viewingLikesOfPostID") != null){
40          postID = (int)session.getAttribute("viewingLikesOfPostID");
41          session.removeAttribute("viewingLikesOfPostID");
42          client = Client.create();
43          link = "http://localhost:8080/GameSociety/rest/GameSociety/getAllUsersLiked/";
44          link+=postID;
45          webResource = client.resource(link);
46          myresponse = webResource.accept("application/json").get(ClientResponse.class);
47          allLikes = new JSONArray(myresponse.getEntity(String.class));
48      }
49      else if(request.getParameter("GoToPost")==null){
50          response.sendRedirect("../../welcome/login.jsp");
51      }
52  %>
```

Figure 48: Calling all Users who have Liked a Post

3.4.6. Profile

The following path shows the pages related to the User profile pages, which can also be seen in Figure 49:

GameSociety/WebContent/Pages/HomePage/Profile

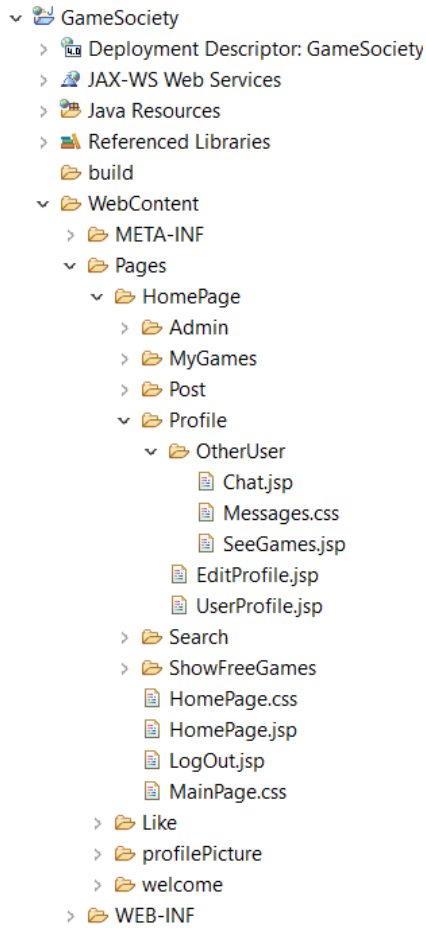


Figure 49: Profile pages

UserProfile.jsp

From the UserProfile.jsp, a user can delete the Publications - if he has the corresponding rights - as explained in chapter 3.4.2. using the Web Service **deletePost**(Image50)

```
134     if(request.getParameter("deletePost")!=null){
135         int postID = Integer.parseInt(request.getParameter("postID"));
136         client = Client.create();
137         link = "http://localhost:8080/GameSociety/rest/GameSociety/deletePost/";
138         link+=postID;
139         webResource = client.resource(link);
140         myresponse = webResource.delete(ClientResponse.class);
141
142         if(request.getParameter("profileUserID")!=null){
143             int profileUser = Integer.parseInt(request.getParameter("profileUserID"));
144             session.setAttribute("profileUserID",profileUser);
145             response.sendRedirect("UserProfile.jsp");
146         }
147     }
```

Figure 50: Deleting a Post through a profile

Administrators, through the code shown in Figure 51, can delete the profile user, using the**deleteAUser**. In Figure 52, a deletion of a user from the Database (with ID number 3) is shown. When deleting a user, all their Posts, all their messages, all the games they have added as well as all their Likes and comments are deleted.

```
70     if(request.getParameter("deleteUser")!=null){
71         int userIDToDelete = Integer.parseInt(request.getParameter("profileUserID"));
72         client = Client.create();
73         link = "http://localhost:8080/GameSociety/rest/GameSociety/deleteAUser/";
74         link+=userIDToDelete;
75         webResource = client.resource(link);
76         myresponse = webResource.delete(ClientResponse.class);
77         response.sendRedirect("../HomePage.jsp");
78     }
```

Figure 51: Deleting a User through their Profile

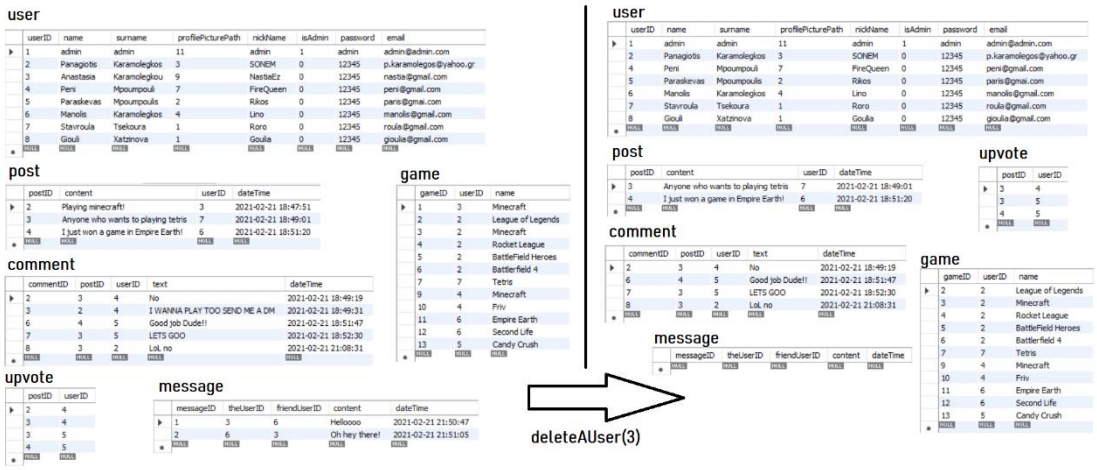
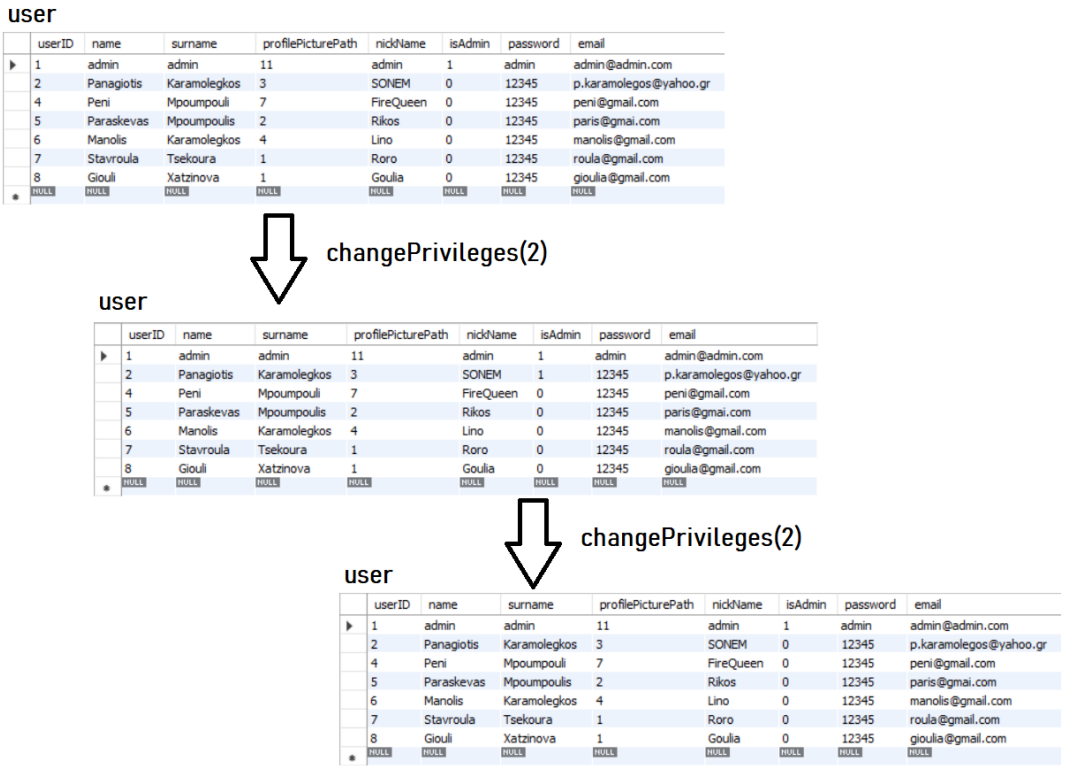


Figure 52: Deleting a User from the Database

You are also using the Web Service on this page**changePrivileges** to change the rights of a User from Administrator to simple User and vice versa. Its use is shown in Figure 53 and the change it causes to the Database is shown in Figure 54, using User ID number 2.

```
96      if(request.getParameter("changePrivileges")!=null){
97          int profileUser = Integer.parseInt(request.getParameter("profileUserID"));
98          client = Client.create();
99          link = "http://localhost:8080/GameSociety/rest/GameSociety/changePrivileges/";
100         link+=profileUser;
101         webResource = client.resource(link);
102         myresponse = webResource.put(ClientResponse.class);
103         session.setAttribute("profileUserID",profileUser);
104         response.sendRedirect("UserProfile.jsp");
105     }
```

Figure 53: Code for changing permissions



userID	name	surname	profilePicturePath	nickName	isAdmin	password	email
1	admin	admin	11	admin	1	admin	admin@admin.com
2	Panagiotis	Karamolegkos	3	SONEM	0	12345	p.karamolegos@yahoo.gr
4	Peni	Mpoumpouli	7	FireQueen	0	12345	peni@gmail.com
5	Paraskevas	Mpoumpoulis	2	Rikos	0	12345	paris@gmai.com
6	Manolis	Karamolegkos	4	Lino	0	12345	manolis@gmail.com
7	Stavroula	Tsekoura	1	Roro	0	12345	roula@gmail.com
8	Giouli	Xatzinova	1	Goulia	0	12345	gioulia@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 54: Changing User rights in the Database

To insert Posts from System Users, use the code in Figure 55 using the Service **addAPost**. An example of this usage in the Database is shown in Figure 56.

```
            content = content.replace(" ","_");
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
            LocalDateTime now = LocalDateTime.now();
            String dateTime = dtf.format(now);
            dateTime = dateTime.replace(" ","_");
            client = Client.create();
            link = "http://localhost:8080/GameSociety/rest/GameSociety/addAPost/";
            link+=userID+"/";
            link+=content+"/";
            link+=dateTime;
            webResource = client.resource(link);
            myresponse = webResource.post(ClientResponse.class);
```

Figure 55: Post input code

Figure 56: Inserting a Post into the Database

EditProfile.jsp

In the EditProfile.jsp page it similarly uses the**deleteAUser**such as and in UserProfile.jsp. In addition, this page successively uses the **chageProfileInfo**in order to change any information that the respective User wants. THE this function simply changes elements in the Base.

```
    if(request.getParameter("changePassword")!=null){
        String newPassword = request.getParameter("password");
        if(newPassword.equals("")){
            passwordError = "You must fill the field!";
        }
        else if(newPassword.contains(" ")){
            passwordError = "You may not use spaces!";
        }
        else{
            client = Client.create();
            link = "http://localhost:8080/GameSociety/rest/GameSociety/changeProfileInfo/";
            link+=userID+"/";
            link+=name+"/";
            link+=surname+"/";
            link+=profilePicturePath+"/";
            link+=nickName+"/";
            link+=newPassword+"/";
            link+=email;
            webResource = client.resource(link);
            myresponse = webResource.put(ClientResponse.class);
            passwordError = "The update is done!";
        }
    }
```

Figure 57: User data change code

Chat.jsp

The page inside the OtherUser folder named Chat.jsp exists for communication between Users. It is being used**showMessages**so that the messages of the conversation between the respective two Users (Figure 58).

```
92          client = Client.create();
93          link = "http://localhost:8080/GameSociety/rest/GameSociety/showMessages/";
94          link+=userID+"/";
95          link+=profileUserID;
96          webResource = client.resource(link);
97          myresponse = webResource.accept("application/json").get(ClientResponse.class);
98          messages = new JSONArray(myresponse.getEntity(String.class));
```

Figure 58: Message acquisition code

Also, the Web Service is used**sendMessage**(Image59) for sending messages between the two Users. An example in the Database is in Figure 60 where User with ID number 2 sends the message "Hello" to User with ID number 3.

```
50              content = content.replace(" ","_");
51              DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
52              LocalDateTime now = LocalDateTime.now();
53              String dateTime = dtf.format(now);
54              dateTime = dateTime.replace(" ","_");
55              client = Client.create();
56              link = "http://localhost:8080/GameSociety/rest/GameSociety/sendAMessage/";
57              link+=userID+"/";
58              link+=profileUser+"/";
59              link+=content+"/";
60              link+=dateTime;
61              webResource = client.resource(link);
62              myresponse = webResource.post(ClientResponse.class);
```

Figure 59: Web Service for sending messages

Figure 60: Sending a message between two Users

SeeGames.jsp

As on the page for the "Games I Play" tab, it is used similarly **getAllUserGames**and to display all games of another User (Figure 61).

```
client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/getAllUserGames/";
link+=profileUserID;
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
userGames = new JSONArray(myresponse.getEntity(String.class));
```

Figure 61: Web Service for obtaining a User's game

3.4.7. Search

Within the following path, we locate the SearchSelection.jsp file:

GameSociety/WebContent/Pages/HomePage/Search/SearchSelection.jsp

In this file, the "Search Users" tab is displayed, where searches are made for Users through the games they have declared that they are playing, as well as their nickname.

Figure 62 shows its use**getUsersByNickName**in order to make the appearance of users who have a nickname similar to the one the User has entered as input.

A similar use is made for the**getUsersPlayingTheGame**(Image63) in order to display the Users who play the game that the User gives as input.

```
client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/getUsersByNickName/";
link+=searchedNickname;
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
usersByNickName = new JSONArray(myresponse.getEntity(String.class));
```

Figure 62: User Acquisition Alias similar to login

```
client = Client.create();
link = "http://localhost:8080/GameSociety/rest/GameSociety/getUsersPlayingTheGame/";
link+=searchedGame.replace(' ','_');
webResource = client.resource(link);
myresponse = webResource.accept("application/json").get(ClientResponse.class);
usersByGame = new JSONArray(myresponse.getEntity(String.class));
```

Figure 63: Acquiring Users playing a given game

3.4.8. ShowFreeGames

Within the following path, we locate the FreeGamesSelection.jsp file:

GameSociety/WebContent/Pages/HomePage/ShowFreeGames/
FreeGamesSelection.jsp

In this file, the "Free Games" tab appears, where searches are made for free games that exist on the internet. The Web Services used are Third Party Web Services. Their use is shown in Figure 64.

```
37 <%
38     if(request.getParameter("allFreeGames")!=null){
39         client = Client.create();
40         String link = "https://www.freetogame.com/api/games";
41         webResource = client.resource(link);
42         myresponse = webResource.accept("application/json").get(ClientResponse.class);
43         allFreeGames = new JSONArray(myresponse.getEntity(String.class));
44     }
45
46     if(request.getParameter("platformSearch")!=null){
47         String platform = request.getParameter("platform");
48         if(platform.equals("")){
49             error = "You must fill the field!";
50         }
51         else if(platform.contains(" ")){
52             error = "You must not use spaces!";
53         }
54         else{
55             client = Client.create();
56             String link = "https://www.freetogame.com/api/games?platform=";
57             link+=platform;
58             webResource = client.resource(link);
59             myresponse = webResource.accept("application/json").get(ClientResponse.class);
60             try{
61                 platformFreeGames = new JSONArray(myresponse.getEntity(String.class));
62             }
63             catch(JSONException ex){
64
65             }
66         }
67     }
68 %>
```

Figure 64: Third Party Web Services

Figures 65, 66 and 67 show three returns of these Services via the Postman tool.

GET https://www.freetogame.com/api/games Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (25) Test Results Status: 200 OK Time: 222 ms Size: 161.59 KB Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "title": "Dauntless",
        "thumbnail": "https://www.freetogame.com/g/1/thumbnail.jpg",
        "short_description": "A free-to-play, co-op action RPG with gameplay similar to Monster Hunter.",
        "game_url": "https://www.freetogame.com/open/dauntless",
        "genre": "MMORPG",
        "platform": "PC (Windows)",
        "publisher": "Phoenix Labs",
        "developer": "Phoenix Labs, Iron Galaxy",
        "release_date": "2019-05-21",
        "freetogame_profile_url": "https://www.freetogame.com/dauntless"
    },
    {
        "id": 2,
        "title": "World of Tanks",
        "thumbnail": "https://www.freetogame.com/g/2/thumbnail.jpg",
        "short_description": "If you like blowing up tanks, with a quick and intense game style you will love this game!",
        "game_url": "https://www.freetogame.com/open/world-of-tanks",
        "genre": "Shooter",
        "platform": "PC (Windows)",
        "publisher": "Wargaming",
        "developer": "Wargaming",
        "release date": "2011-04-12",
```


Figure 65: Proper Use of Free To Game

GET https://www.freetogame.com/api/games?platform=pc Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (25) Test Results Status: 200 OK Time: 602 ms Size: 130.38 KB Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "title": "Dauntless",
        "thumbnail": "https://www.freetogame.com/g/1/thumbnail.jpg",
        "short_description": "A free-to-play, co-op action RPG with gameplay similar to Monster Hunter.",
        "game_url": "https://www.freetogame.com/open/dauntless",
        "genre": "MMORPG",
        "platform": "PC (Windows)",
        "publisher": "Phoenix Labs",
        "developer": "Phoenix Labs, Iron Galaxy",
        "release_date": "2019-05-21",
        "freetogame_profile_url": "https://www.freetogame.com/dauntless"
    },
    {
        "id": 2,
        "title": "World of Tanks",
        "thumbnail": "https://www.freetogame.com/g/2/thumbnail.jpg",
        "short_description": "If you like blowing up tanks, with a quick and intense game style you will love this game!",
        "game_url": "https://www.freetogame.com/open/world-of-tanks",
        "genre": "Shooter",
        "platform": "PC (Windows)",
        "publisher": "Wargaming",
        "developer": "Wargaming",
        "release_date": "2011-04-12",
        "freetogame_profile_url": "https://www.freetogame.com/world-of-tanks"
```


Figure 66: Proper Use of Free To Game For Platforms

GET https://www.freetogame.com/api/games?platform=wrongPlatform Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (25) Test Results Status: 404 Not Found Time: 127 ms Size: 1.19 KB Sav

Pretty Raw Preview Visualize JSON


```
{
    "status": 0,
    "status_message": "No platform found, please check the correct parameters."
}
```


Figure 67: Incorrect Use of Free To Game For Platforms