Online ISSN: 2367-5357
DOI: 10.7546/itc-2019-0010

Development of a Browser Based Strategy Game Application

V. Dimitrov, K. Kolev

Key Words: *Kinvey; JavaScript; jQuery; HTML; CSS.*

Abstract. *The proposed application discussed in the present article is a browser based strategy game. The programmable languages and tools used for its development including JavaScript, jQuery, HTML, CSS, Kinvey, Sublime Text, etc. The application is build using the 3-Tier Software Architecture. Some of the benefits of this architecture are scalability, speed of development, performance, availability, etc.*

1. Introduction

PC games are listed among the best entertainments suitable not only for kids but for adults, too. Currently, there are available games and games in development, which are ideal for almost any part of the human society. Some game players dream of being a cook, while others – to experience a real fight – and due to this reason new and so different games have been designing all the time, even right now, to satisfy the individual needs and preferences. There are, indeed, lots of game types, but one of the most common and popular ones are the multiplayer games or those, where players can compete against each other, as well as socialize and have fun with people from all over the world. This is the main goal in the game project made by the authors of the material. It is a project of a browser-based strategy game.

What is a browser game? It is a game, which does not require the player to either download or install it on his computer. The game can be loaded and played directly through an internet browser. This game is inspired by an old one by BIZZARD Company – StarCraft. This game allows the players to take control of an army with a maximum population of up to 200 people, while the game of the project permits the players to have at their disposals unlimited armies to use in vast and massive fights. In this game, players strive for necessary resources, spy the rest of the players, arrange armies aiming to protect their territories or to attack the opponents.

2. Description of the used technologies

In this project HTML (Hypertext Markup Language) methods are used to build and design the website where the game can be played. It is a browser game with no 3D graphics or animations, but HTML elements for visualization. HTML files come with attachments of many JavaScript files. In addition to these, inside some of the HMTL files there is an inner implementation of JavaScript content. Each HTML file has its own CSS (Cascading Style Sheets) file that defines the visual presentation of the HTML elements. A certain part of the HTML code is generated by JavaScript through JQuery. The application uses the following HTML files:

Index.html
Index.html is the home page that contains information about the game and its background history, as well as fields for login and registration. After completing the registration process, the players are redirected to Home.html.

Admin.html
Admin.html is an admin panel through which the players with some resources are generated depending on their units and buildings. This page is accessible only by the administrator. The admin panel contains a button through which resources can be manually added.

Home.html
Home.html is the main page through which the app is used, and the game is played. It is where HTML for several pages is opened and constantly replaced with one another depending on what the players what to see and do. This page has a drop-down menu from where we can select which HTML to be active, respectively shown.

JavaScript
Within the application, JavaScript serves one of the most essential roles. The entire functionality of the browser game is executed by different scripts and their inner functions. JavaScript serves the role of the connection between the game and the database. Thanks to JavaScript

procedures like login and registration, transition from one to another panel in home.html, adding new resource, building and making new units, as well as the most crucial thing in the game – the attacks between the players – are executed.

The application uses the following scripts divided into 3 groups:

Library (libs)
These scripts are of an open-source type and each of them comes with different function:

- **bootstrap.min.js**
- **jquery.min.js**
- **q.js**

Subsidiary (helpers)
These scripts are made with a general purpose to support the rest scripts being pieces of content for constant files (constants.js) and requesters (requester.js).
Requerster.js – this is script through which POST, GET, PUT и DELETE queries are executed.

Module (modules)
These scripts are linked with requester.js and call through functions for certain cases the orders from requester.js for the available tables. Each of the scripts serves its own table, which is why their names are: User.js, Building.js, Unit.js and Upgrade.js

General (main)
These are the main scripts where the final functions that are requested in the game are coded. These scripts integrate with almost any of the previously listed scripts:

- **admin.js**
  This is the script that is executed in the admin panel, and its goal is to load the players with certain resources depending on their units and buildings.
- **index.js**
  This is the script that is executed on the home page and contains functions for login and registration.
- **home.js**
  This is the script that is executed on the main page for information refreshment and to provide the necessary functions for the command we want to perform.
- **war.js**
  This script is executed on the main page and serves the role to present what is going on in War Room.

jQuery
jQuery is implemented through the previously mentioned jQuery.min.js. It is a common alternative of JavaScript published at the beginning of 2006 by John Resig [5]. In its essence jQuery simplifies the access to each element of a particular website and thus, allows the easy establishment of a dynamic functionality within the pages. Query also simplifies the call of the functions through the buttons and supports the creation of a dynamic HTML code that is added in certain situations. It is used in all the main scripts, plus – it has implantation inside home.html.

Kinvey
This game uses Kinvey as a Cloud database (DB) where the information for the registered users, as well as their units, building, etc. is stored.

2.1. Application architecture

The main architecture of the application consist of 3 layers (*figure 1*), while other architectures are offered in [1, 4].

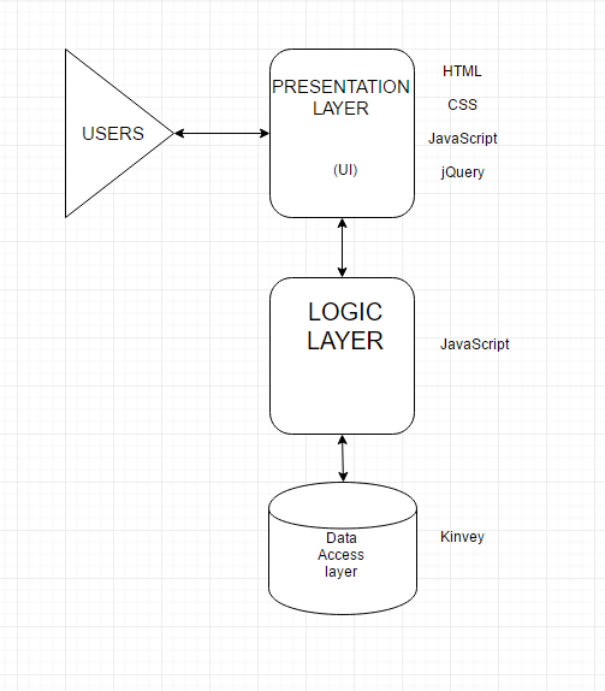

Figure 1. Application architecture

Presentation layer
The communication between the users and the application is performed in the presentation layer. The users have direct access to this layer, which represents the web pages and through which they call the commands that are performed within the next layer. It is built with HTML and CSS, as well as with a bit of JavaScript for responsive design.

Logic layer
The logic layer, also known as business logic, is where the processes and the functions which the user triggers, but does not have a direct access of their execution. The logic layer is also responsible for the execution of the functions that are necessary to complete a certain required task. It is where data/information from the database can be downloaded to either process or to be used in a certain function execution.

Data access layer

This is the layer, which contains the application data in a "raw" source or the so called database. Specifically, in this case the database is available in Cloud space and it is sorted in 4 tables linked with each other using a specific key. The database is accessible through the logic layer or manually by the administrator.

Scheme execution

The players access the application through their web browser and by clicking on "Register" button through the presentation layer, the function for registration is started to be executed by a script in the logic layer. Once the user data is accepted, the user performs the required tasks to register. This is how an order of a POST type is send to the database with information for a new user request alongside with a table where the user will be registered. This app allows the creation of other tables for a certain user to be created during the registration process.

3. An architecture of a combat between two players

The main function in the game is the combat between the players (*figure 2*). The other approaches are reviewed in [2, 3].

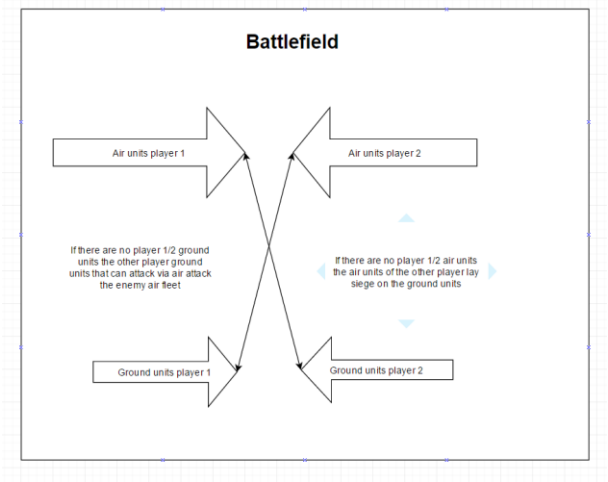

Figure 2. Architecture of the combat between two players

The combat is performed on two different fronts as the game has both – ground and flying units. This is why the logic of the combat begins with the hypothesis that the ground units of the players get into fight and during this moment their flying units are in a combat, too. As there is more than a unit per unit type, each of these units has an attack priority or a certain position in the army. The first line that is going to be taken aback consists of a certain unit. Once a player wins in the ground combat against other player and having units for flying fights, his or her ground army directs to the flying unit of the opponent and vice versa. The general priorities are listed in *figure 3*.

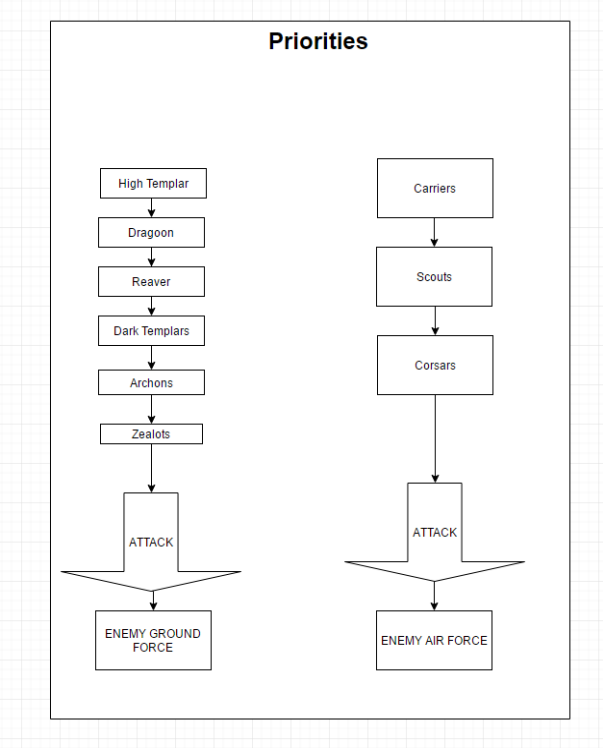

Figure 3. The priorities of the units during a combat

3.1. An architecture of dependencies between buildings, units and upgrades

In the game in order to build certain buildings, units or perform upgrades, the player will need the dependencies between them, i.e. which unit needs which building to be able to be built (*figure 4*).

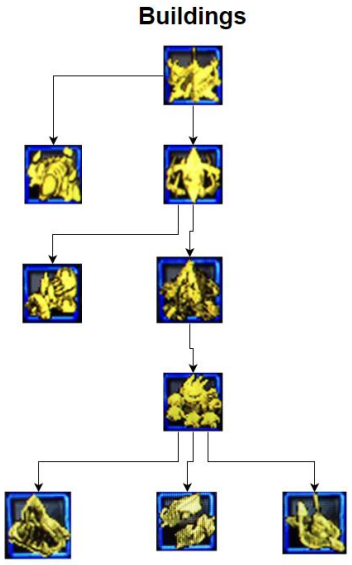

Figure 4. Dependence of buildings on buildings

Dependencies on units (*figure 5*):

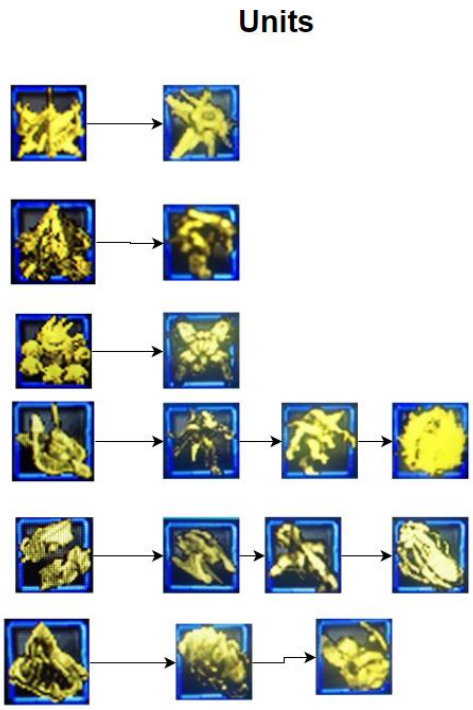

Figure 5. Dependencies of units on buildings

Dependencies of upgrades on buildings (*figure 6*):

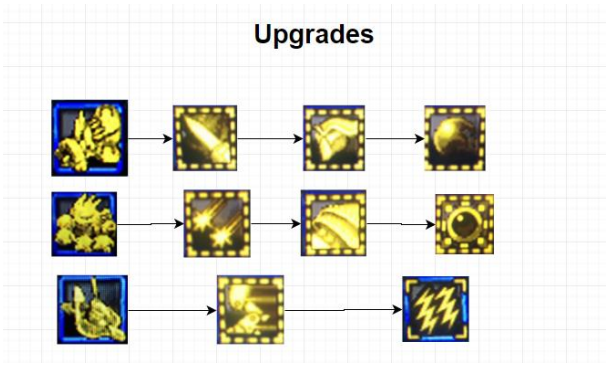

Figure 6. Dependencies of upgrades on buildings

4. HTML architecture

The HTML architecture consists of 3 HTML files: index.html, home.html and admin.html; shown in the *figure 7*.

Admin.html – it is detached by the rest HTML files with no connection with them. It has an administrative purpose. Admin.html is accessible only for the administrator of the game. Through Admin.html the admin refreshes the resources of all the players on the base of the formula and each player's units and buildings.

Index.html – represents the home page with the options for login and new registration, as well as for the game history and the war in which the users will participate after complete registrations. After registration or successful login, the page automatically sends the user to home.html, and the information is once again refreshed.

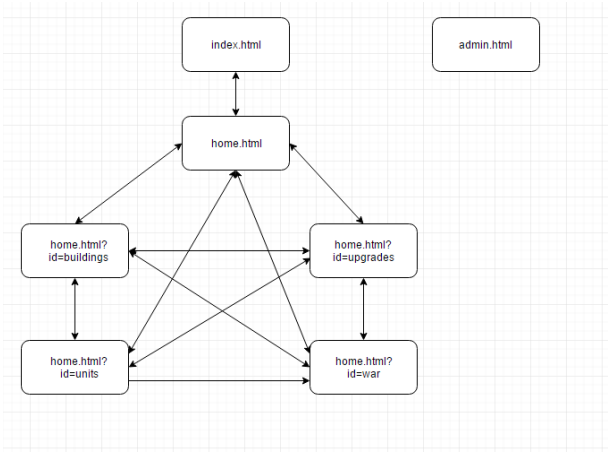

Figure 7. HTML architecture

Home.html – this is one HTML file where the html code for 4 more pages is stored, but when it comes to the design and actions, the entire information is actually placed in one single html file. It is just more preferable to set certain pages in hidden mode to show only the pages we request. Here is an example: if we have home.html?id=units, we will get loaded only the page where we build the units.

4.1. JavaScript architecture

The scripts in the application are attached to HTML through jQuery. Some of them start at the moment a certain page gets accessible, while others require clicking on a certain button. Most of the functions have them the same cycle but are with different tasks. They start from html "onClick" or "load" (*figure 8*). The functions for each html are carried in detached main scripts. It is where the direction to a certain module for the request is performed. The module determines the request of a certain function and then, sends the data in requester.js from where the request to the database is sent. Once there is a response, it is transfigured into a result (result set) in the main script from where the data process starts according to the goal of the process/function. Last, but not least, the result is shown in html to be visible for the user.

4.2. Kinvey DB architecture

The database of the application is located at "Cloud Storage" in Kinvey [6]. The information inside it can be modified through the application functions. If a manual action is required, the admin can enter it to change the information in the Cloud-space. The DB contains one table with users that with each login in the app start a new session

and get unique ID through which they connect with the rest of the tables. In the DB of the app there is also information for each user – what resource he/she owns, what units, buildings and improvements he/she has made. Each user has a single line per table from where the individual user's data is extracted through the user's unique ID (*figure 9*).

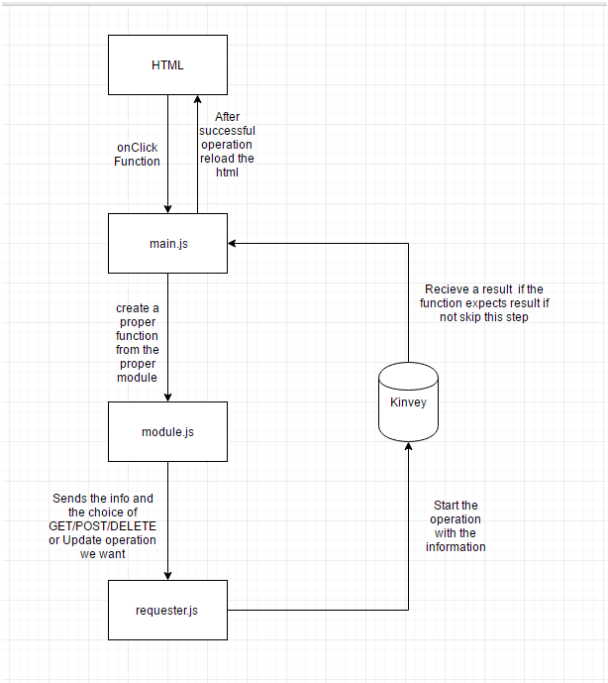

Figure 8. JavaScript architecture

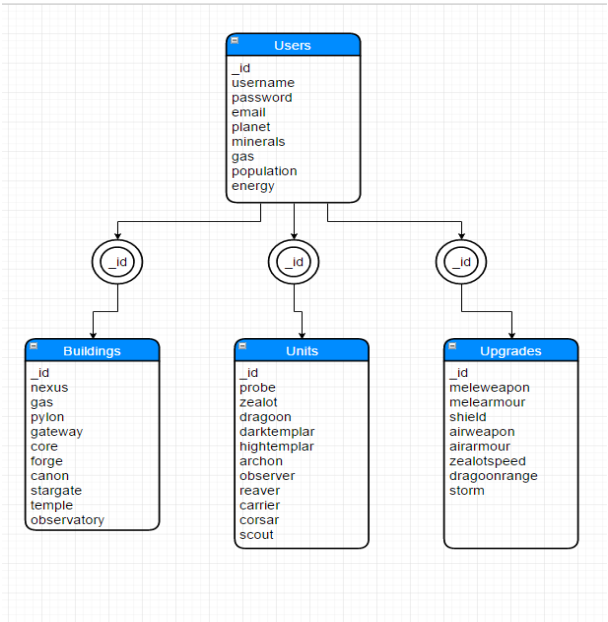

Figure 9. Architecture of DB and the connection between the tables

Example of unit purchase execution

In this example the user accesses the html part for unit purchase and starts onClick function (*figure 10*). The function is executed in home.js from where two requests are sent. One of the requests is sent to the table with the users aiming to execute the user identification (user id). The other one is sent to the table of the purchased unit where the data is refreshed according to the executed purchase. Then, two different functions of a request type are started. One of them aims to refresh the number of player units of a certain type, while the other refreshes the player's resources after the completed purchase. In this example both of the requests are of a POST type that alongside with the data, are sent from the module scripts to requester.js, which, meanwhile, refreshes the DB. In case of a correct execution a refresh of the website follows aiming to show the new number of the units.

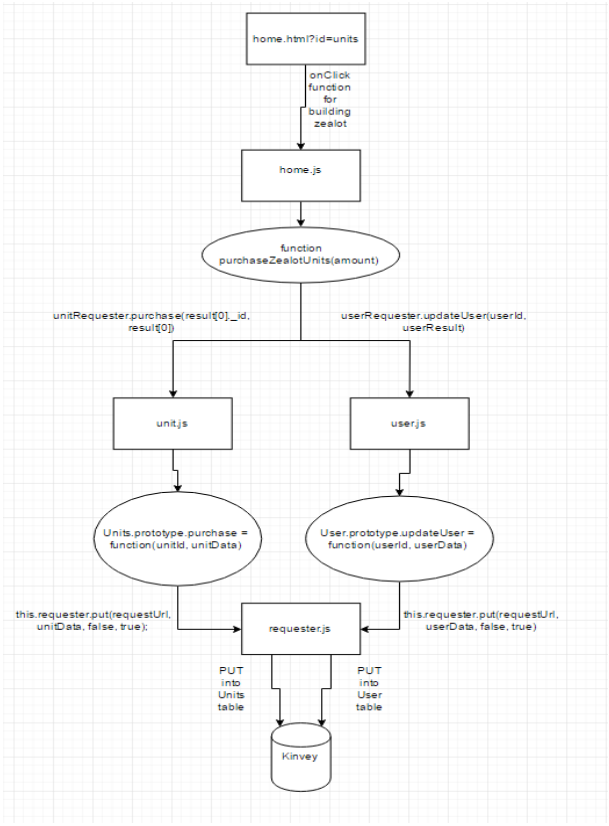

Figure 10. Example of unit purchase architecture

5. Conclusion

The project could be developed further with additional functionalities such as:

- Adding function for messages between the players and saving war reports;
- Password change and password recover;
- Better game balance;
- Adding more races and history expansion.

Other similar games used for ideas when developing this project: www.bulfleet.com, www.hanovete.com, www.sco.trs10.com.

References

1. Baek, I., K. Kim. Web-Based Interface for Data Labeling in StarCraft. 2018 IEEE Conference on Computational Intelligence and Games (CIG), Maastricht, Netherlands, 14-17 August 2018, doi: 10.1109/CIG.2018.8490451.
2. Nguyen, T., K. Nguyen, R. Thawonmas. Potential flow for unit positioning during combat in StarCraft. 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE), Tokyo, Japan, 1-4 October 2013, doi: 10.1109/GCCE.2013.6664759.
3. Rooijackers, M., M. Winands. Wall Building in the Game of StarCraft with Terrain Considerations. 2018 IEEE Conference on Computational Intelligence and Games (CIG), Maastricht, Netherlands, 14-17 August 2018, doi: 10.1109/CIG.2018.8490413.
4. Smolyakov, I., S. Belyaev. Design of the Software Architecture for StarCraft Video Game on the Basis of Finite State Machines. 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, Russia, 28-31 January 2019, doi: 10.1109/EIConRus.2019.8656866.
5. https://api.jquery.com/
6. https://devcenter.kinvey.com/rest/guides/datastore/

Manuscript received on 23.10.2018

Vladimir Dimitrov *graduated in 2016 with a Ph.D. degree from Technical University – Sofia, Computer Systems Department with thesis about IPTV. Holds the highly respective CCNP Security certification. Teaches on a freelance basics Computer Networks courses and also is a supervisor of B.Sc. and M.Sc. students. Works as a Network Security Engineer. Main scientific interests in computer networks, network security and programming.*

Contacts:
Dianabad, bl. 21, Sofia, Bulgaria
e-mail: vldimitrov85@gmail.com

Kolio Kolev *is a Ph.D. student at Institute of Information and Communication Technologies – Bulgarian Academy of Sciences. His research interest includes systems for digital TV broadcast, telecommunication technologies and programming.*

Contacts:
Institute of Information and Communication Technologies
Bulgarian Academy of Sciences
Acad. Georgi Bonchev Str., block 25A, 1113 Sofia, Bulgaria
e-mail: kkolev@iit.bas.bg