A Collaborative Web Computer-Aided Design Application

R. L. Soares[1], D. S. Bomfim[1], L. F. Bez[2], P. C. F. Lopes[2], A. M. B. Pereira[2], L. F. Martha[1]

[1] *Dept. of Civil and Environmental Engineering, Pontifical Catholic University of Rio de Janeiro*
St. Marquês de São Vicente, 225 - Gávea, 22541-041, RJ / Rio de Janeiro, Brazil
rodrigolucassoares@gmail.com, dsbomfim2@hotmail.com, lfm@tecgraf.puc-rio.br
[2] *Institute of Computing, Fluminense Federal University*
Av. Gal. Milton Tavares de Souza, s/n - São Domingos, 24210-310, RJ/Niterói Brazil
luizf.bez@gmail.com, pedrocortez@id.uff.br, andremaues@gmail.com

Abstract. Computer-aided design (CAD) applications have been helping engineers to design and understand better the structure and the behavior of models. These applications usually have a complex data structure and are stand-alone. However, the advance in cloud-based systems has made applications scale better and brought the possibility of real-time collaborative applications. This study presents a proof of concept of a collaborative web CAD application. The application uses a client-server architecture. The server is responsible for keeping the data structure and crossing data between clients using the collaborative functionality while the client has a user interface and a canvas component to display data and the model. The application uses WebSocket to communicate between client and server.

Keywords: CAD, Collaboration, Web architecture.

1 Introduction

In spite of technology and web-designed applications have evolved, most CAD/CAE applications are developed as standalone and single-user applications [1]. Nowadays engineering and manufacturing industries teams have spread to different locations, this geographical barrier consists in a problem when information needs to be shared with different teams in different locations. On the other hand, as the method of analysis becomes more sophisticated more computational power is needed, consequently the engineers need computers with high power of processing, which might not be available locally for all users. In addition, in many areas, the process of decision-making cannot be isolated and collaboration aids problem-solving [2].

More flexibility is provided to CAD/CAE software giving them cloud functionalities, some of the benefits of this approach are:

- There is no need to install the software on each machine: users only need to access the URL of the software and the software is going to work in the user's browser, and software updates will be working soon as the new version is uploaded to production.
- Simplification on file-sharing: usually projects files are shared by paper or electronic means (email, cloud file sharing systems).
- Employment of processing power on cloud servers: as the software works as a thin layer on the user's browser, all the processing is made on the servers that means that computers with less processing power could still be used.

Several works about visualization and collaboration in web environments have been published in a wide range from engineering and manufacturing industries to medicine, such as [1-3]. But there is still a lack of published works discussing simultaneous drawing and visualization tools in collaboration.

In cloud computing there are three service models available as follows [1]:

- *Infrastructure as a Service (IaaS):* Provides processing, storage, and other computational resources and allows the user to deploy programs. The user has control over deployed applications and storage.

- *Platform as a Service (PaaS):* The user has the capability to deploy applications but does not has the ability to manage the cloud infrastructure except for a few environment configurations.
- *Software as a Service (SaaS):* The user has access to use the provider's application; the application is accessible from various client interfaces. The user has only access to user-specific configurations.

In this work is presented a Software as a Service (SaaS) interactive web CAD/CAE application, which has been named by Zissis et al [1] as CAD/CAE as a service (CAAS). The application consists of a client that works in the user's browser and uses the WebSocket protocol to connect with the server interface that stores model data structure and connects multiple users to a single model data structure for collaboration. The main purpose of the application discussed in this work is for modeling and visualization of Engineering applications such as Finite Element Analysis (FEA). However, the way the architecture was developed allows developers to build their own client applications in a way that better fits the use case.

2 Architecture

The architecture consists of a client-server environment, in which a client application works in the user's browser and is used to process input, manipulate the data, and visualize data output. The client application is developed using JavaScript language and the user interface is developed using Facebook's user interface library React. The server application is responsible to process and store user's data in memory and is developed using Python. It is worth mentioning that the case FEA server application has a half-edge data structure to ease the modeling task, but describing this is out of the scope of this paper. In the next sections, it is explained the communication between client and server applications as well as the architecture and the technologies that are used.

2.1 Communication

The communication between client and server applications uses the WebSocket protocol as an application layer of the TCP/IP model. The TCP/IP reference model was first described by Cerf and Kahn (1974), and later defined as a standard in the internet community. It has 4 layers as the protocol stack – Link Layer, Internet Layer, Transport Layer, and Application Layer [4]. The Application Layer and the Transport Layer are the highest-level layers of the protocol stack and, in most cases, are the layers that the developer can implement or choose which protocol to use.

The Transport Layer is designed to allow peer entities on the source and destination hosts to have a channel for communication, it can operate over two protocols: TCP (Transmission Control Protocol), which is a reliable connection-oriented protocol or UDP (User Datagram Protocol), which is an unreliable connectionless protocol [4]. The TCP protocol fits better for the purpose of this work because it provides reliability to data transmission avoiding modeling interruptions.

The Application Layer defines the patterns that are used by applications to communicate with each other. Here it is worth mentioning the HTTP (HyperText Transmission Protocol), which is the most used protocol for common web applications and the WebSocket protocol.

The WebSocket protocol works through the HTTP protocol and provides a full-duplex channel. Thus, after the connection is established, either the client or the server can send messages to each other in any order [5]. This approach is especially useful to CAAS applications because it allows the server to start data transmission to all clients. Therefore, all client instances of a shared model are updated simultaneously, in contrast to HTTP in which, after one user interaction, other users will not have a simultaneous update. This issue was mentioned in Zissis et al [1].

Figure 1 exemplifies the WebSocket Protocol mentioned above. Multiple clients can start the connection with the server through the HTTP method GET, containing a header rule to update the connection to WebSocket. The server responds by accepting or refusing the connection. After that, if the server accepted the connection, the clients and the server have a full-duplex communication channel. In addition, the libraries used as an interface to WebSocket protocol on the client and on the server (Socket.io and Flask-socket.io, respectively) allow clients to freely join and leave rooms. The clients have their own socket, but share the same room, since the room is used to broadcast information between a subset of clients.

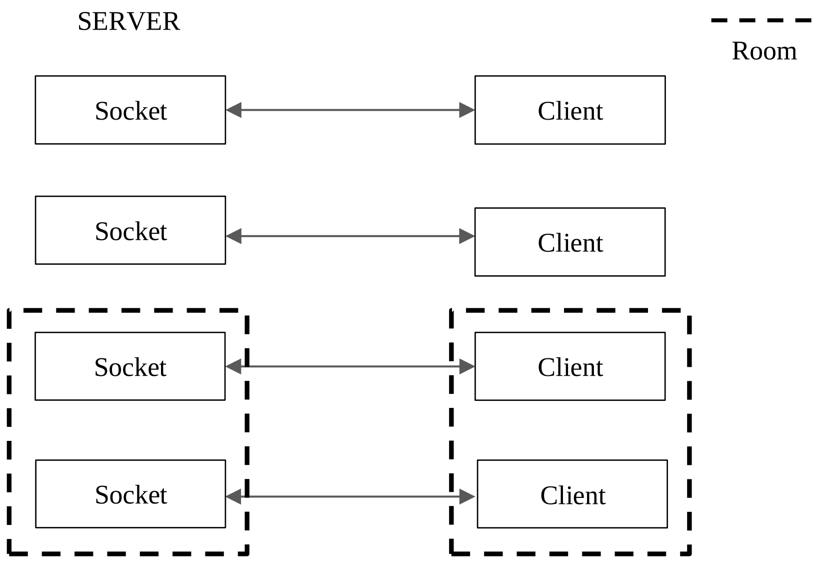


Figure 1 – Client – Server architecture using WebSocket.

2.2 Overall architecture

After having briefly discussed the communication process, this section discusses the overall architecture of the application. The server application is designed to be applicable to many different types of applications. Consequently, one design principle taken into consideration was to separate the server application to run as an Application Program Interface (API) that receives and sends requests to client applications after having processed the requests. Therefore, the designed application has two servers: one server to send the static files to be interpreted by the user's browser and the other server to run the server application (the API). This approach allows users to design their own client applications that better fit their case. The main architecture is showed in Figure 2. First, the user makes a request by the HTTP protocol GET and by response receives the static files which are used to interpret the client application in user's browser, after that the application starts a WebSocket connection to the server application that is hosted in another domain.

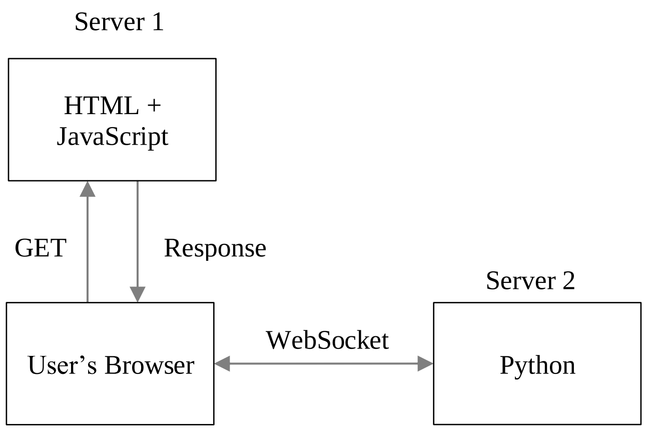


Figure 2 - Application architecture.

After the client application has been connected to the server application the user can start to model. Figure 3 shows a use-case diagram to elucidate the behavior of the application in the case when there are two users working in collaboration.

The user can model with or without collaboration. The modeling task is basic geometric entities insertion, deletion, and attribute insertion. To collaborate with other users, one needs to create a room and share the room's token with other users. With that token, the users can connect to the room and model in collaboration.

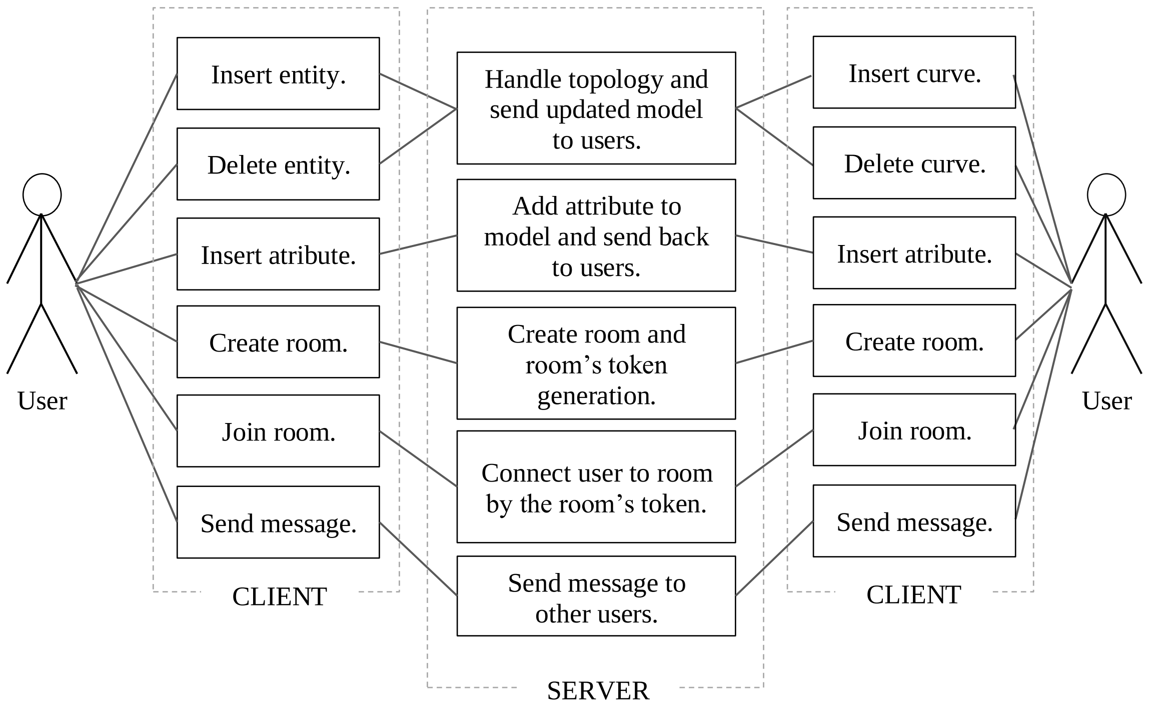

Insert entity.
Delete entity.
Insert atribute.
Create room.
Join room.
Send message.
CLIENT
User
Handle topology and send updated model to users.
Add attribute to model and send back to users.
Create room and room's token generation.
Connect user to room by the room's token.
Send message to other users.
SERVER
Insert curve.
Delete curve.
Insert atribute.
Create room.
Join room.
Send message.
CLIENT
User

Figure 3 – Use-case diagram.

2.3 User interface.

The user interface was developed using the React library, a JavaScript library developed by Facebook and based on components that manage their own state in such a way that when associated permits the creation of complex user interfaces [7]. React components constitute a tree structure and components occupying a level above share their state with components at a level below [8]. The component that occupies the top level is considered the root component, from which all other components derive.

Component state management was performed in a distributed way, that is, the state is not managed only in the root component, avoiding the re-rendering of all components that share the state when it is modified, which could cause performance problems. In Figure 4, the component tree for the developed application is presented.

Components were designed to perform specific functions. The App component is the root component from which the other components derive, and it stores the application states that are common to other components. The Canvas component was created to perform the iterative modeling task. It is an HTML5 canvas element that uses WebGL. The latter is a cross-platform web standard for 3D graphics generation [9] that offers a JavaScript API to execute the shader codes on the client's GPU [1]. An attribute manager component was also developed. Attributes are entities that can define loads, supports, and material properties. In addition, a messaging component was developed that allows users to communicate when connected to a room, and a command line to execute commands without the need for mouse iteration. The other components complement an interface to make it user-friendly, including headers, options menus.

Figure 5 shows the application's user interface. As a user starts to draw a model the changes are updated simultaneously to remaining users, the same happens with sent messages. To illustrate that, in Figure 5b a user draws a triangle by inserting lines and sends a message to other users using the message component. Figure 5c shows another user connected to the same room, it is possible to notice the model and the room's messages being displayed in canvas and messages components, respectively. In addition, each model has individual control of the camera, so a user can pan and zoom without interfering with others user's canvas.

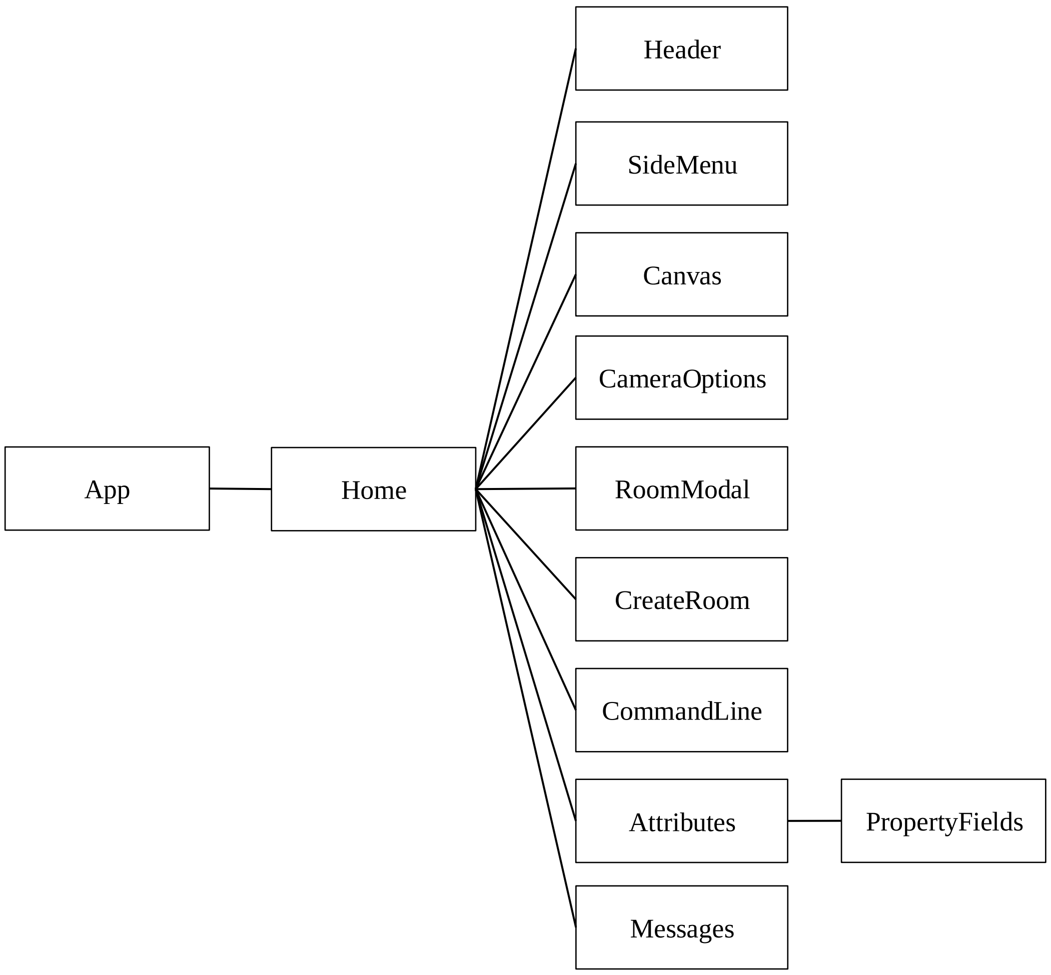
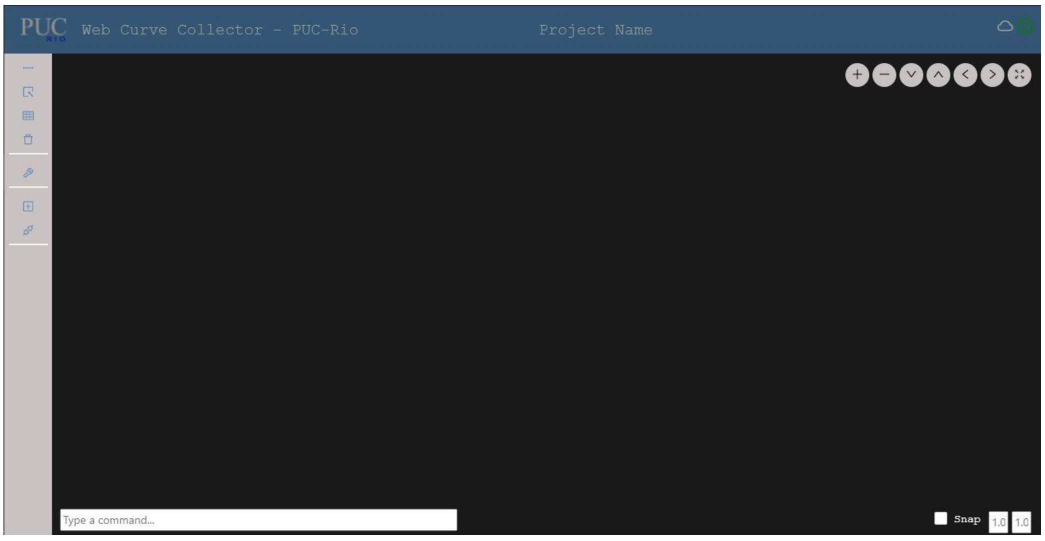
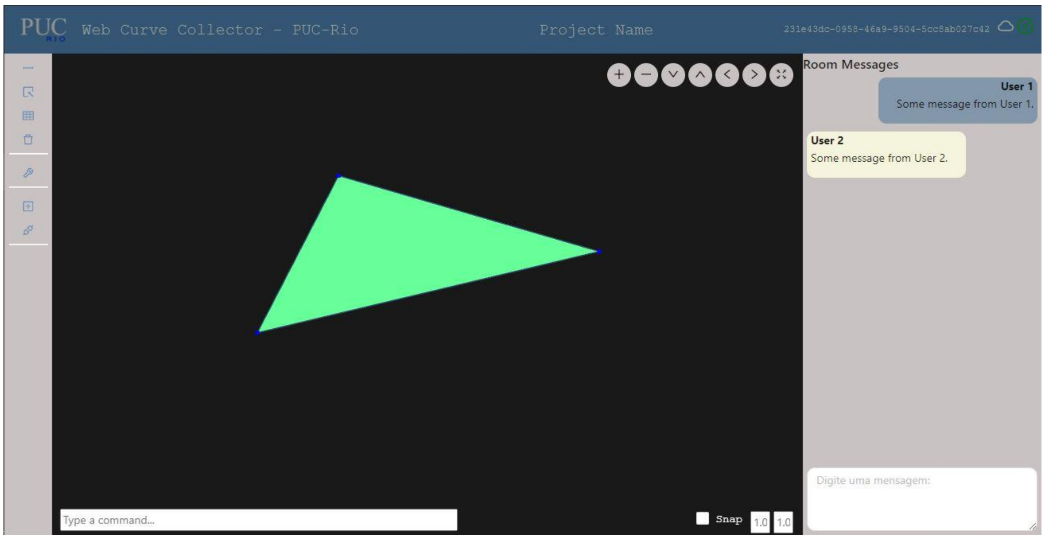


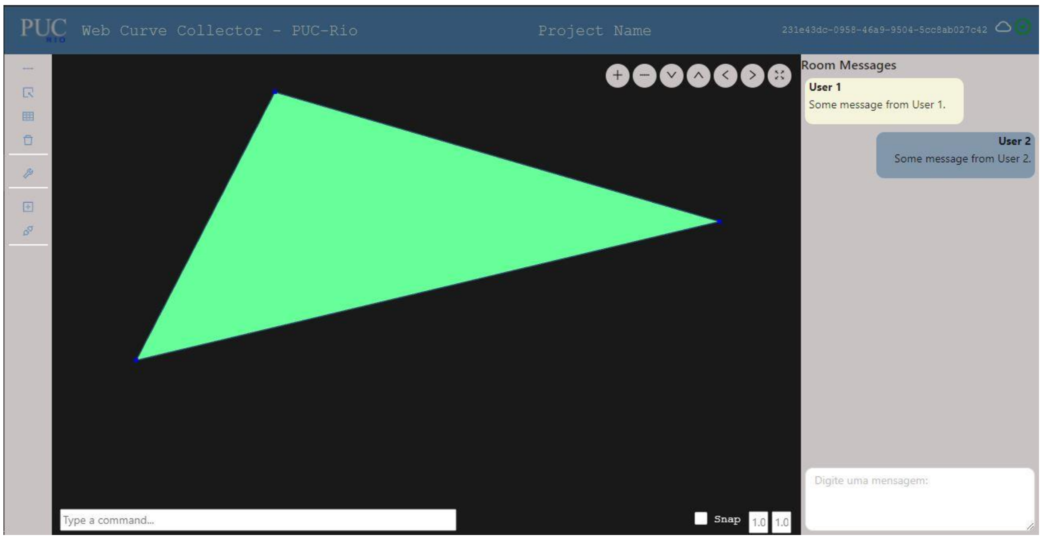
Figure 4 – Component's tree.



(a)


PUC Rio
Web Curve Collector - PUC-Rio
Project Name
231e43dc-0958-46a9-9504-5cc8ab027c42
Room Messages
User 1
Some message from User 1.
User 2
Some message from User 2.
Digite uma mensagem:
Type a command...
Snap
1.0
1.0

(b)


PUC Rio
Web Curve Collector - PUC-Rio
Project Name
231e43dc-0958-46a9-9504-5cc8ab027c42
Room Messages
User 1
Some message from User 1.
User 2
Some message from User 2.
Digite uma mensagem:
Type a command...
Snap
1.0
1.0

(c)

Figure 5 – Default user interface (a) User 1 interface (b) User 2 interface (c).

3 Conclusions

This work presented a simple web-based architecture to support CAAS by developing a proof-of-concept application. The application has the capability of designing simultaneously and cooperatively and discussing with other users through a chat system. It can be applied to educational FEA applications to ease the task of teaching as everyone will have access to the same information at the same time and avoiding compatibilities problems with different application versions and licenses. The application is still a proof of concept. More work needs to be done to improve its performance. Some suggestions are inserting a load balancer like NGINX to avoid issues in time-response as the application flow increases. It is also interesting to do a benchmark to know better the application performance and compatibility with different browsers.

Acknowledgements. This work was carried out with the support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Financing Code 001. The first author is funded by the grant # 132786/2020-6, Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq).

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] D. Zissis, D. Lekkas, P. Azariadis, P. Papanikos and E. Xidias. *Collaborative CAD/CAE as a service.* International Journal of Systems Science: Operations & Logistics, 2016.
[2] P. Lavric, C. Bohak, M. Maholt. *Collaborative view-aligned annotations in web-based 3D medical data visualization*. 40th International Convention on Information and Communication Technology, Electronics and Microelectronics. Croatia, 2017.
[3] P. Nyamsuren, S. Lee, H. Hwang, T. Kim. *A Web-based collaborative framework for facilitating decision making on a 3D design developing process*. Journal of Computational Design and Engineering, 2015.
[4] A. S. Tanembaum and D. J. Whetherall. *Computer Networks*. Prentice Hall, 2011.
[5] Lombardi. *A first course in Blab La Blab.* Publisher, 2021.
[6] Socket.io. Rooms. Accessed 17 July 2021, < https://socket.io/docs/v4/rooms/>.
[7] Facebook Inc. React – A JavaScript library for building user interfaces, 2021. Accessed 14 July 2021, < https://reactjs.org/>.
[8] A. Banks and E. Porcello. *Learning React : Modern Patterns for Developing React Apps*. O'Reilly, 2020.
[9] Khronos Group. WebGL Overview. Accessed 14 July 2021, < https://www.khronos.org/webgl/>.