A Web Serverless Architecture for Buildings Modeling

Enrico Marino*, Danilo Salvati†, Federico Spini*, Christian Vadalà†
*Department of Engineering, Roma Tre University, Rome, Italy
Email:{marino,spini}@ing.uniroma3.it
†Department of Mathematics and Physics, Roma Tre University, Rome, Italy
Email:{salvati,vadala}@ing.uniroma3.it

***Abstract*—The motivations of relentless migration of software products toward services accessible via Web must be sought in the undeniable benefits in terms of accessibility, usability, maintainability and spreadability granted by the Web medium itself. It is the case of office suites, beforehand thought as resilient desktop applications, nowadays made available as Web applications, often equipped with real-time collaboration features and with no need for the user to explicitly install or upgrade them anymore. Although it could not be easy to envisage a Web-based graphic application due to its inherent complexity, after the recent and significant enrichment of the HTML5 APIs, a few first attempts appeared online in the form of vectorial drawing collaborative editors or VR oriented interior design environments. This paper introduces an effective Web architecture for buildings modeling that leverages the serverless pattern to dominate the developing complexity. The resulting front-end application, powered by Web Components and based on unidirectional data flow pattern, is extremely customizable and extendible by means the definition of plugins to augment the UI or the application functionalities. As regards the modeling approach, it offers (a) to model the building drawing the 2D plans and to navigate the building in a 3D first person point of view; (b) to collaborate in real-time, allowing to work simultaneously on different layers of the project; (c) to define and use new building elements, that are furnitures or architectural components (such as stairs, roofs, etc.), augmenting a ready to use catalog. This work suggests a path for the next-coming BIM online services, matching the professionals collaboration requirements typical of the BIM approach with the platform which supports them the most: the Web.**

1. Introduction

Nowadays we are seeing a relentless migration of software products toward services accessible via the Web medium. This is mainly due to the undeniable benefits in terms of accessibility, usability, maintainability and spreadability granted by the Web medium itself. Nevertheless these benefits don't come without a cost: performance and development complexity become major concerns in the Web environment.

In particular, due to the several introduced abstraction layers it is not always feasible to "port" a desktop application into the Web realm, an aspect to be taken into account even for the relevant hardware differences among all the devices equipped with a Web Browser. It can be even more arduous to tackle the inherent distributed software architecture (a client/server one at least) induced by the Web platform.

Nevertheless increasingly rich and complex Web applications began to appear, supported by the enriched HTML5 APIs, which thanks to the WebGL [1] (which enables direct access to GPU), Canvas [2] (2D raster APIs) and SVG [3] (vectorial drawing APIs), has paved the way for the entrance of Web Graphic Applications.

In this work we report about our endeavor toward the definition of a Web based buildings modeling tool which overcomes the aforementioned performance and development difficulties relying on a unidirectional data flow design pattern and on a serverless architecture [4], respectively.

A serverless architecture, on the contrary of what the name may suggest, actually employs many different specific servers, whose operation and maintenance don't burden on the project developer(s). These several servers can be seen as third party services (typically cloud-based) or functions executed into ephemeral containers (may only last for one invocation) to manage the internal state and server-side logic. Realtime interaction among users jointly working on the same modeling project, is for example achieved via a third party APIs for remote users collaboration.

The tool user interface, entirely based on web components pattern, has been kept as simple as possible: the user is required to interact mainly with two-dimensional symbolic placeholders representing parts of the building, thus avoiding complex 3D interactions. The modeling complexity is thus moved from the modeler to the developer which fills out an extendible *catalog* of customizable *building elements*. The modeler has only to select the required element, place and parametrize it according to the requirements. It is obvious that a large number of building elements has to be provided to ensure the fulfillment of the most modeling requirements.

The remainder of this document is organized as follows. Section 2 provides an overview of related work. Section 3 reports about the application user experience. Section 4 presents adopted architectural solutions. Finally, Section 5

contains some conclusive remarks.

2. Related work

In this section we highlight some remarkable experiences aligned with the aim of our project. There are plenty of Desktop applications worth to be mentioned and analyzed, but in the following we deliberately focus on Web based works.

Shapespark[1] offers a web viewer of remarkable quality that allow the user to move inside a synthetic 3D indoor environment. Modeling phase is served in the form of plugins for different Desktop proprietary solutions.

Playcanvas[2] is a complete and powerful web based game creation platform which offers an integrated physical engine and a whole set of functionalities to support modeling. Although powerful and relatively simple to use, it doesn't focus on buildings modeling.

Floorplan[3] has been developed by Autodesk specifically for the architectural field, and for indoor renewal projects in particular. It is a 2D modeling tool which offer also a 3D walk-through mode.

Spini et al. [5] introduced a Web modeling and baking service for indoor environments. The modeling tools exposes a 3D interaction the user may not be accustomed to, an hitch we tried to outflank by avoiding 3D modeling interaction and let the user only face a "metaphoric" 2D interface.

As regards support for users collaboration it worths to be mentioned the *Operational Transformation* (OT) approach [6]: a group of nodes exchanges messages without a central control point. Two main properties hold in this setup: (i) changes are relative to other user's changes (it works on "diffs") and (ii) no matter in which order concurrent changes are applied, the final document is the same. In our serverless architecture however, external (third parties) central synchronization point are allowed, making complexity introduced by protocols like OT less effective.

3. Application Experience

The aim of this application is to lead the user to the realization of a building model through description of the 2D plan in order to simplify the user work, as he does not need to do difficult 3D interactions but only 2D drawings. To follow this idea, we chose to create our buildings through **symbolic modeling** which consists in the creation of placeholders for a particular object, leaving to the platform developer the modeling interactions needed to transform the placeholders into the wanted 2D/3D model. So the the user only need to drag and drop an object into the drawing area or to draw the wireframe, depending on the type of object he is interested in. All this objects are also grouped on **layers**, which can have different altitudes and opacity.

1. https://www.shapespark.com/
2. https://playcanvas.com/
3. http://www.homestyler.com/floorplan/

3.1. Building Elements Catalog

As we have seen earlier, the software relies on a library of symbolic elements to replace 3D interactions, they are collected in a **building elements catalog** containing four element typologies (from now we will refer to an element contained in the catalog as **building element**). Each typology has been identified studying the various user interactions and the way an object can be placed in the space:

Lines Each line is drawn by selection of a start point and of an end point and we can have two ways to move this type of building element. The first one is by dragging one of the points, the other one is by dragging the entire line. For example walls are elements belonging to this category

Openings Each opening is an element that is linked to a line, making an hole on it. The user create a new opening dragging it on a chosen line. Doors and windows on walls are elements belonging to this category

Areas Each area is an element which is automatically generated from walls. Basements belongs to this category

Objects Each element which is freely inserted into space with a drag and drop interaction is an object

3.2. User Interface

blabla

4. Serverless Architecture

We used a Serverless Architecture to deploy all the work.

To offer the application to end-users with high availability and high performance we deployed all Javascript files that are part of the system into a CDN (Content Delivery Network). This single point of deploy offered us a centralized method to upgrade all customers applications, whereby there is a new version, without any customer explicit action.

To perform CPU intensive operations, necessary to make complex geometric element, we used a third party FaaS system that run Python functions and that generate 3D elements. Thank to this we can scale up and down and adjust the CPU load on the traffic.

The application is distributed as frontend web application and so it's mostly executed into customer's browser. This has the following advantages: (i) to avoid user hard installation or updating typically performed by not technical users; (ii) to offer a good abstraction level that makes the application platform and operation system independent; (iii) to avoid a server overloading by moving heavy computations on the user client.

The application state is mostly inside the customer's browser and it is represented as a tree structured object. This state can be serialized and saved on a BaaS system or downloaded as JSON file. The customer can restore the application state in any time using a preceding downloaded serialized version of the state. The system will adjust the

user interface and the scene by means of the Virtual DOM and the diff and patch algorithm.

The collaboration part is based on a DBaaS, with the common state stored in an external DB provided by Firebase. Following this methodology we could also add user authentication features based on external BaaS providing all user management functionalities.

In Figure 1 there is the architectural schema described above

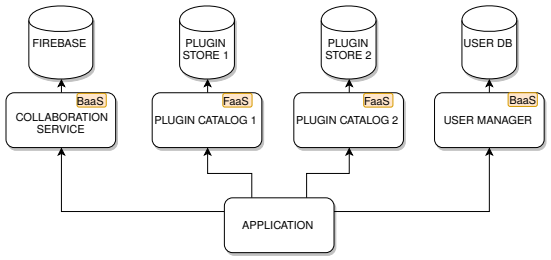

Figure 1. The serverless architecture for our application.

4.1. Centralized Application State

blabla

4.2. Component Based UI

blabla

5. Conclusions

In this work we outlined a serverless architecture to support buildings modeling in a Web environment. The serverless architecture that gives benefits in terms of availability, reliability, scalability, easiness of deployment, maintainability and upgradability is obtained by implementing the application logic as a client-side only centralized state Web application exploiting the unidirectional data flow pattern. This approach allow for a easy-to-serialize state (in the form of a JSON document) that can be pushed on a third party document oriented DB-as-a-Service and loaded back in the frontend reactive architecture, which transparently reload the state once its serialized version is passed in. The application itself is served by a CDN (Content Delivery Network) thus avoiding any need for web server. Offline routines rely on Function-as-a-Service platform as well as users management and collaboration features.

This architecture has been successfully employed by the authors in the *Metior* project [7], a tool to support selective deconstruction of buildings in the pursuit of a "zero waste" model.

Acknowledgments

Authors would like to thank GEOWEB S.p.A., a web service company owned by Sogei S.p.A. and CNGeGL Italian National Board of Quantity Surveyors, for supporting this work. Thanks are extended to Stefano Perrone for developing the models shown in the Figures **??**.

References

[1] D. Jackson, "WebGL Specification," Khronos, Khronos Recommendation, Oct. 2014, https://www.khronos.org/registry/webgl/specs/1.0.3/.

[2] J. Munro, J. Mann, I. Hickson, T. Wiltzius, and R. Cabanier, "HTML Canvas 2D Context," W3C, W3C Recommendation, Nov. 2015, http://www.w3.org/TR/2015/REC-2dcontext-20151119/.

[3] D. Jackson, E. Dahlström, J. Ferraiolo, A. Grasso, C. McCormack, P. Dengler, J. Fujisawa, D. Schepers, C. Lilley, and J. Watt, "Scalable Vector Graphics (SVG) 1.1 (Second Edition)," W3C, W3C Recommendation, Aug. 2011, http://www.w3.org/TR/2011/REC-SVG11-20110816/.

[4] M. Roberts, "Serverless Architectures." [Online]. Available: http://martinfowler.com/articles/serverless.html

[5] F. Spini, E. Marino, M. D'Antimi, E. Carra, and A. Paoluzzi, "Web 3D Indoor Authoring and VR Exploration via Texture Baking Service," in *Proceedings of the 21st International Conference on Web3D Technology*, ser. Web3D '16. New York, NY, USA: ACM, 2016, pp. 151–154. [Online]. Available: http://doi.acm.org/10.1145/2945292.2945309

[6] C. A. Ellis and S. J. Gibbs, "Concurrency Control in Groupware Systems," *SIGMOD Rec.*, vol. 18, no. 2, pp. 399–407, Jun. 1989. [Online]. Available: http://doi.acm.org/10.1145/66926.66963

[7] E. Marino, F. Spini, D. Salvati, C. Vadalà, M. Vicentino, A. Paoluzzi, and A. Bottaro, "Modeling semantics for building deconstruction," in *Proceedings of the 12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - GRAPP*, 2017, to appear.