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

Scopo di questo lavoro quello di presentare uno strumento di progettazione architettonica basato su web. Dal punto di vista architetturale, abbiamo introdotto una serverless architecture (vedi citazione [1]). Unarchitettura di questo tipo si basa massivamente su servizi di terze parti (typically in the cloud) o su funzioni invocate allinterno di ephemeral containers (may only last for one invocation) per la gestione dello stato e della logica server-side. Questa scelta ci ha permesso di ridurre la complessit dellinfrastruttura.

(AGGIUNGERE QUALCHE MINIMO DETTAGLIO SUL FATTO CHE GIRO NEL BROWSER SU QUALSIASI DISPOSITIVO E SISTEMA)

In effetti i vantaggi di queste scelte tecniche sono molteplici e variano a seconda del punto di vista che si vuole adottare:

Punto di vista dell'utente: non vi sono complicate installazioni del sistema o procedure di aggiornamento. Il browser fornisce un ambiente compatibile con diverse macchine e con diversi sistemi operativi

Punto di vista dello sviluppatore: possibile propagare nuove versioni del software in maniera istantanea verso gli utenti

Punto di vista dello sviluppo: nessuna preoccupazione sulla gestione del carico o delluptime in quanto lapplicazione principale gira su client

Entrando nel dettaglio del software realizzato, questo strumento di progettazione architettonica si propone di realizzare modelli di edifici attraverso lintroduzione di uninterfaccia semplificata che mira ad evitare complicate interazioni con i modelli tridimensionali cercando invece di rimpiazzarle con interazioni su rappresentazioni bidimensionali simboliche delledificio. La natura web ci ha permesso inoltre di sperimentare API per la collaborazione tra utenti remoti, che possono cos progettare un edificio contemporaneamente su pc diversi. Molta cura stata poi posta nellespandibilit del sistema, in modo che gli sviluppatori che intendessero offrire questa piattaforma a loro volta potessero personalizzarlo a piacere. A questo fine stato introdotto un **catalogo**, sempre basato su un servizio esterno, che potesse fornire oggetti da inserire per la modellazione con delle **propriet** specifiche per il tipo di applicazione. Le sezioni che seguono mostrano nel dettaglio i concetti espressi qui, mostrandone la realizzazione e ponendo laccento sugli aspetti architetturali. (UNA VOLTA COMPLETATO METTERE UN ELENCO PUNTUALE DEL CONTENUTO DELLE SEZIONI)

2. Literature review

In commercio esistono molti software per la realizzazione di edifici. Si pu pensare ai classici prodotti CAD quali AutoCAD o veri e propri BIM come Revit della Autodesk. Questi sono sicuramente i pi diffusi e completi, tuttavia essendo desktop apps, dal punto di vista dellutente la loro installazione pu essere difficoltosa e gli aggiornamenti pi complicati. Il nostro obiettivo era invece quello di creare una piattaforma semplificata che potesse invece raggiungere praticamente la totalit degli utenti anche su piattaforme differenti. Anche qui vi sono degli esempi, basti pensare al software floorplan di autodesk (http://www.homestyler.com/floorplan/) o anche a strumenti di modellazione tridimensionale vera e propria (Bak3d).

https://www.shapespark.com

Uno dei contributi di questo lavoro consiste nellesplorazione del nuovo standard dei Web Components e del pattern uniflow. Il risultato atteso quello di riuscire a realizzare non solo una completa infrastruttura per il disegno ma anche una piattaforma estendibile da parte degli utenti mediante lintroduzione di nuovi componenti. In effetti questa la vera e propria differenza con gli strumenti citati precedentemente, perch larchitettura proposta non si limita a risolvere il problema della modellazione ma offre uno strumento di produzione di modellatori architettonici.

AGGIUNGI CITAZIONI ED ESPANDI PARTE SUI WEB COMPONENTS

3. Methodology

3.1. Centralized state

La realizzazione del sistema stata effettuata utilizzando alcune tra le tecnologie ed i pattern pi in auge nello sviluppo di applicazioni web-oriented. Nello specifico sono stati utilizzati i pattern Unidirectional Data Flow e Immutability attraverso limplementazione messa a disposizione rispettivamente dalle librerie ReduxJS e ImmutableJS.

Lo sviluppo del progetto si focalizzato sull'individuazione di una struttura gerarchica in grado di rappresentare in modo esaustivo linsieme delle componenti che compongono la planimetria, costituendo di fatto lo stato dellapplicazione. Lo stato stato gestito attraverso limmutabilità, un principio che prevede lapplicazione di nuove modifiche attraverso la generazione di un nuovo stato, in primis identico al precedente, ma sul quale vengono applicate le modifiche richieste. Dal punto di vista della gestione della memoria questo meccanismo richiederebbe un importante dispendio di risorse. Per evitare ci limplementazione ImmutableJS adotta dei meccanismi che simulano il principio di immutabilità evitando la copia in profondità della memoria (deep clone) e alleggerendo cos il carico in termini di utilizzo di memoria e tempo macchina.

3.1.1. Application as Finite State Machine. Per dominare la complessità dellapplicazione lintero sistema stato

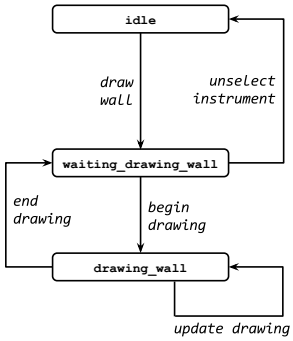

Figure 1. Bla bla bla.

modellato su una macchina a stati. Sfruttando il modello basato su azioni e reducer messo a disposizione dal pattern unidirectional data flow stato individuato un grafo che rappresenta le possibili evoluzioni dello stato dellapplicazione. Il nodo corrente in cui si trova la macchina a stati stato mappato attraverso lintroduzione di una variabile globale che rappresenta la modalità in cui si trova lapplicazione, gli eventi del browser sono stati mappati con gli archi uscenti del grafo.Ad esempio unoperazione di creazione di un nuovo muro viene mappata attraverso un grafo composto da 4 nodi.

3.2. Collaboration

COLLABORAZIONE

3.3. Building elements

Il catalogo del programma un insieme, strutturato in categorie, nel quale sono contenuti tutti gli elementi costruttivi che possibile inserire allinterno della planimetria. Lattribuzione della categoria viene effettuata sulla base delle caratteristiche dellelemento e determina i casi duso permessi allutente. Le categorie sono:

Walls, rientrano in questa categoria tutti i tipi di muro (perimetrali, interni, portanti). La creazione avviene specificando il punto di inizio e fine dellelemento. La rappresentazione interna dei muri viene ricondotta a quella di un grafo in cui i nodi, che corrispondono ai punti geometrici in cui si intersecano pi muri, hanno delle coordinate che li collocano nello spazio e gli archi sono larea visibile del muro.

Openings, rientrano in questa categoria gli elementi che bucano i muri come porte, finestre e archi. La creazione viene effettuata attraverso uno snap sui muri precedentemente creati. Lutente pu agire sulla posizione

ed il sistema garantisce che questa venga preservato il legamente con il muro.
Areas, rientrano in questa categoria i pavimenti. La creazione viene effettuata automaticamente attraverso unanalisi della disposizione dei muri. Lalgoritmo individuato viene di seguito descritto.
Objects, rientrano in questa categoria tutti gli oggetti posizionabili sulle aree. Lutente pu agire sulla disposizione sia in termini di posizione che rotazione.

3.4. UI components

L'intera applicazione è stata pensata in maniera modulare, in modo da poterla estendere in maniera semplice. Dal punto di vista web la tecnologia applicata stata quella dei **Web Components**. L'idea di base quello di definire l'applicazione frontend come una collezione di componenti, che vengono renderizzati in modo diverso a seconda dei valori assegnati allo stato. In particolare, vi una classe di componenti delegati alla rappresentazione delle proprietà dello stato stesso: i **viewers**. Esempi sono i visualizzatori per il 2D e per il 3D. In generale, la struttura a componenti permette di definire una qualunque visualizzazione dello stato (ad esempio in forma tabellare) semplicemente inserendo un nuovo componente. Una schematizzazione del concetto evidenziata nella figura 2.

Come possiamo vedere nello schema, siamo in grado di identificare tre macro blocchi. Il primo il catalogo, che come abbiamo visto (CITAZIONE AL PARAGRAFO) contiene tutti i building elements del sistema e le loro properietà. Il secondo è il core vero e proprio e si occupa della gestione dello stato e contiene le funzionalit di disegno. Questo comunica con il catalogo prendendo le proprietà dei building elements. Infine abbiamo i visualizzatori veri e propri, che vengono scelti in base alla `mode` property inside the state.

Al momento sono stati implementati i visualizzatori 2D e 3D.

2D Viewer. This viewer creates a 2D view of the building model. Dato lo stato globale, in grado di sfruttare il **Virtual DOM** per aggiornare solo le parti che vengono modificate evitando aggiornamenti globali continui

3D Viewer. Sfrutta la libreria di modellazione per WebGL **ThreeJS** per creare una vista 3D del building model. Per evitare di dover continuamente ricreare l'intero modello quando cambiano porzioni dello stato, è stato implementato un sistema di *diff* e *patch* dello stato (usando la libreria `immutable-js-diff`) che a sua volta si riconduce allo stato citato in [1]. In pratica stata creata una struttura parallela che mappa gli oggetti di ThreeJS con i relativi building elements memorizzati nello stato. Ogni volta che viene lanciata un'azione redux, viene calcolata la diff tra il vecchio stato e quello nuovo e si ricreano solo gli

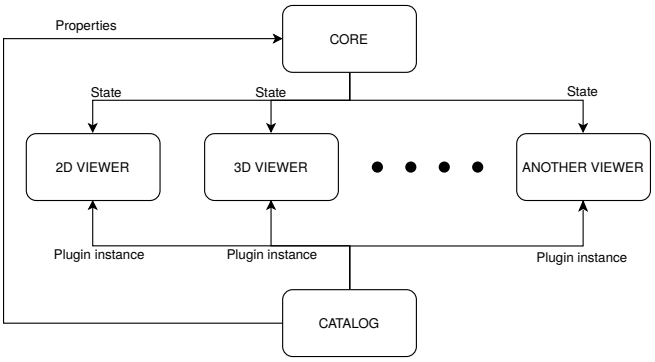

Figure 2. The architectural schema for the viewers. Here we can see that the core can instanziate several different viewers giving them the state for the representation

oggetti che sono stati modificati. In particolare possiamo avere tre possibili *operations*: (i) **add**, (ii) **replace** and (iii) **remove**. For every operation, viene determinato un diverso tipo di comportamento in base al particolare building element cambiato, coinvolgendo eventualmente building elements correlati

3.4.1. 3D renderer component.

threeJS immutablediff

3.5. Architettura serverless

CARICO FRONTEND
CARICAMENTO REMOTO GEOMETRIA PLUGIN

4. Results

E' possibile identificare uno stato centralizzato e consistente che descrive lo stato del progetto
Grazie all'algoritmo del virtual dom il 2d viene manutenuto allineata sulla base dello stato in modo automatico
Grazie all'algoritmo di diff e patch il 3d viene mutuato senza effettuare un rendering completo, ma applicando le modifiche dello stato
Il sistema di lock su layer permette la modifica contemporanea su parti diverse del progetto
Servizio di database realtime permette di sincronizzare i vari client sulle modifiche anche in modo concorrente
Definizione dei plugin possibile estendere gli elementi geometrici che arricchiscono il progetto architettonico
Con l'utilizzo dei componenti possibile comporre l'interfaccia ed esporre il servizio in modo modulare

4.1. Future work

Aggiungere nuovi componenti Servizi che generano remotamente building elements

Acknowledgments

The authors would like to thank Stefano..... and Prof....

References

[1] P. Bryan and M. Nottingham, "JavaScript Object Notation (JSON) Patch," Internet Requests for Comments, RFC Editor, Tech. Rep. 6902, April 2013.