Multimodal Technologies and Interaction

Article

Architecting 3D Interactive Educational Applications for the Web: The Case Study of CrystalWalk

Fernando Bardella [1,*], Rafael Castilho de Moraes [1], Thanos Saringelos [2], Alexandros Karatzaferis [2], Andre Montes Rodrigues [1], Andre Gomes da Silva [1] and Ricardo Mendes Leal Neto [1]

1 Research Group for Scientific Visualization in Materials (GVCM) at the Center for Materials Science and Technology (CCTM), Nuclear and Energy Research Institute (IPEN-CNEN/SP), 2242 Prof. Lineu Prestes Av, Sao Paulo 05508-000, Brazil; rafael@castilho.biz (R.C.d.M.); andre.montes.rodrigues@usp.br (A.M.R.); silva.andregomes@gmail.com (A.G.d.S.); lealneto@ipen.br (R.M.L.N.)

2 School of Electrical and Computer Engineering, Technical University of Crete, Chania 731 00, Greece; asariggelos@isc.tuc.gr (T.S.); ekaratzaferis@isc.tuc.gr (A.K.)

* Correspondence: bardella@ipen.br; Tel.: +55-(11)-3133-9218

Received: 20 August 2018; Accepted: 5 September 2018; Published: 7 September 2018

Abstract: This paper describes the technical development of CrystalWalk, a crystal editor and visualization software designed for teaching materials science and engineering aiming to provide an accessible and interactive platform to students, professors and researchers. Justified by the lack of proper didactic tools, an evaluation of the existing crystallographic software has further shown opportunities for the development of a new software, more focused on the educational approach. CrystalWalk's was guided by principles of free software, accessibility and democratization of knowledge, which was reflected in the application's architecture strategy and the adoption of state-of-the-art technologies for the development of interactive web applications, such as HTML5/WebGL, service-oriented architecture (SOA) and responsive, resilient and elastic distributed systems. CrystalWalk's architecture was successful in supporting the implementation of all specified software requirements proposed by state-of-the-art research and deemed to exert a positive impact in building accessible 3D interactive educational applications for the web.

Keywords: educational platforms; computer graphics; interactive web applications; service-oriented architecture (SOA); distributed systems; crystallographic software

1. Introduction

Motivation behind the development of this research dates back to authors' experience in lecturing materials science and engineering disciplines, where difficulties in understanding fundamental crystal structure topics were broadly reported. Bardella et al. [1] performed a thorough systematic review within existing crystallographic software aiming to verify at what extent it supported the teaching and learning process and the understanding of fundamental materials science topics. This revealed opportunities for the development of a new software focused on students' needs and more broadly a generic online platform for 3D content creation, delivery, editing, and annotation for educational uses. Systematic review highlighted a group of top performing crystallographic software as well indicated gaps and neglected features that were not fulfilled by any evaluated references, revealing enhancement possibilities and opportunities for development of a novel software focused on the educational support to materials science and engineering curricula. Due to the to the domain-specific capabilities of creating and visualizing 3D crystal structure models, only crystallographic software was considered in the review, although interesting didactic opportunities were evaluated from course management systems (CMS) but also potential interface and computer graphics features from computer aided design (CAD) applications. This initial exploratory research was deemed important for determining systematic review's revision protocol, but also the foundational grounds for epistemological aspects of the proposed research strategy [1].

UNESCO's guidelines for education modernization guided CrystalWalk's didactic philosophy, which proposes the use of modern information techniques as enabler for wide availability of quality education resources that can be adaptable customized to specific circumstances [2], capacitating autonomous learning, leadership and creativity [3]. In alignment to such directives, free software [4] and the reactive manifesto [5] philosophies were chosen to guide important strategic aspects of the software architecture, suggesting an interesting perspective in the state of art technologies for the development of interactive web applications such as HTML5/WebGL, service-oriented architecture (SOA) and responsive, resilient and elastic distributed systems [6,7].

The outcomes of several iterations with students and professors reinforced the gaps identified by the systematic review, but also functional requirements deemed significant to any educational context. Aimed at providing students and professors with an easy to use and accessible platform aligned with the material sciences and engineering curricula, a novel educational application was proposed and developed. CrystalWalk is the first web-based 3D interactive crystal editor and visualization software focused on materials science and engineering education [8].

2. Materials and Methods

Due to the complex challenges demanded by application's functional requirements that were identified and confirmed in the systematic review, but also the strict resource limitations imposed by the project, an exploratory research was performed aimed at identifying the most appropriate technologies for the project. Following the approach proposed by Waerner [9] and Pattrasitidech [7], exploratory research aimed at analyzing and comparing main technologies used in 3D software development. Literature was comprehensive and highlighted many flexible and featured-packed technologies, although requirements of application portability narrowed technologies to a smaller subset: Java3D (and its variants), Flash, Stage3D, 3DMLW and HTML5/WebGL. Data summarized in Table 1 consolidate data from this exploratory research stage, indicating that HTML5/WebGL was deemed better aligned with specified requirements of compatibility, portability, support and future perspective. Despite of its risks related to the HTML5 standard maturity at the time of the evaluation [10].

Table 1. Exploratory research summary of Web Graphical Computing technologies (February 2016).

Technology	Compatibility	Set-Up	Support	Integration	Perspective
HTML5/WebGL	★★★	★★★	★★★	★★★	★★★
3DMLW	★★	★★	★	★	★
Java3D	★★	★★	★★	★★	★
Flash	★★	★★	★★★	★★	★
SilverLight	★	★★	★★★	★★	★★
Stage3D	★★	★★	★★★	★★	★★★

Source: adapted from [9] apud [10].

As a natural consequence, exploratory research focus moved towards the selection of a high-level libraries and APIs that could potentially facilitate the application development process, more specifically reducing the laborious efforts traditionally required for the development of a standalone WebGL application [11]. Following the approach proposed by Barbosa [12], exploratory research aimed at libraries that were stable, better documented which project was open source and still active. Data summarized in Table 2 consolidates several WebGL high-level libraries and APIs available, ThreeJS [13] was chosen due to its maturity, available documentation and popularity.

Table 2. Exploratory research summary of HTML5/WebGL high level libraries (February 2016).

Library	Followers	Forks	Documentation	License Type
ThreeJS	1438	6632	★★★	MIT
SceneJS	39	88	★★	MIT/GPL
X3DOM	63	133	★★	MIT/GPL
CopperLicht	3	3	★★★	CopperLicht
CubicVR.js	42	90	★	MIT
GLGE	23	76	★★★	BSD

Source: adapted from [10].

Fundamented in the technologies recommended by the performed exploratory research, the reactive manifesto [5,14] was deemed an adequate paradigm to support functional requirements of accessibility, portability and interaction. This lead to the adoption of other paradigms and technologies in the project such as: service-oriented architectures (SOA), asynchronous module definition specification (AMD), model-view-controller pattern (MVC), publish-subscribe pattern, representational state transfer style (REST) and JavaScript object notation (JSON) [6,7,11,14–17]. Additionally, to the budget impact, this approach also aimed at resource efficiency, enabling the project team to focus on the application development without having to own or manage the infrastructure [14–16]. Lastly, Free Software [4] philosophy was relevant to define and implement some of the project's most important pillars as knowledge democratization and the collaborative development process.

In practical terms, application architecture comprised the integration of 3 modular applications: CWAPP, CWLY, CW4P. CWLY is application's data API responsible for ensuring proper handling of user's querying, loading and storing requests in the database abstraction layer. Implementing a REST architectural style, this API fully integrates application modules with its services as database, domain routing and data storing services. Short URL inputted by the user is handled by Amazon Route 53 service [18], which directs to the CWAPP application. In response to user's performed actions, main web application—CWAPP—initiates a dynamic module loading and communications, enforcing a minimalist allocation of necessary modules and computational resources. CWAPP is an extension of the CW4P framework which effectively implements the event-oriented paradigm for WebGL technology. All applications' lifecycles were designed to run on cloud-based services: they use GitHub as code repository, Amazon Route 53 for domain routing and Heroku as partitioned virtualization provider [19].

Figure 1 illustrates how CWAPP, CWLY, CW4P applications are integrated. An in-depth description of the individual components will be described in the next section.

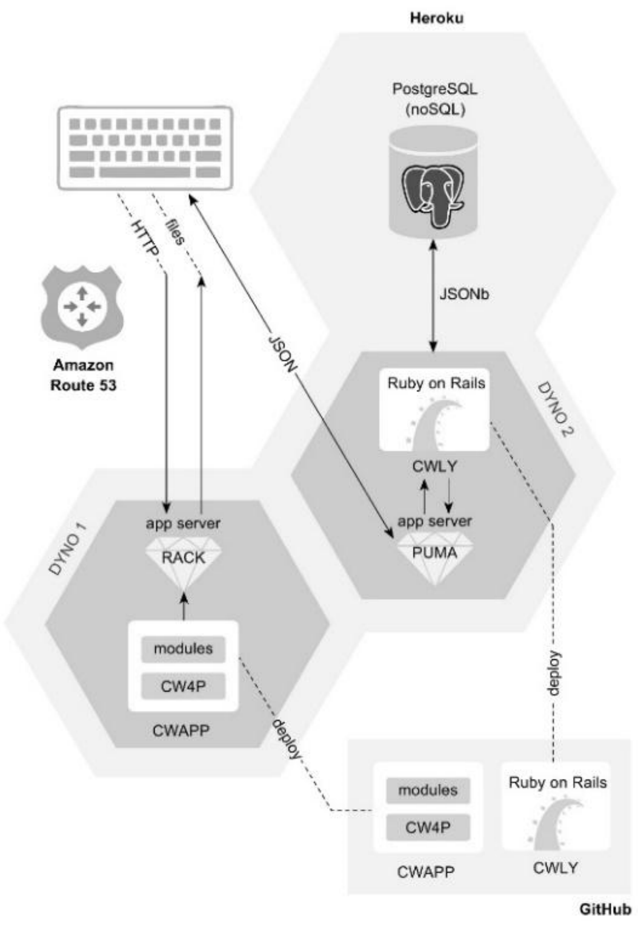

Figure 1. System Architecture. The short URL is handled by Amazon Route 53 service, which directs to the CWAPP application. CWAPP provides necessary modules on demand and CWLY allows access to a database, in order to store or retrieve user created structures. Both applications are hosted on virtualized servers at Heroku. GitHub holds the latest version of the code, which is automatically deployed to Heroku at every update. Source: adapted from [10].

2.1. CrystalWalk Design Pattern (CW4P)

Motivated by best practices in the development of user interfaces, as proposed by Philip [15], Osmani [16], Zakas [17] and the AMD specification [20], CW4P proposes a design pattern that effectively implement the event-oriented paradigm for WebGL technology. This Design Pattern is the foundation of the main CrystalWalk application (CWAPP) and describes overall system architecture and how modules and interconnections are organized. The AMD specification is implemented through the RequireJS library [21]; the publish-subscribe pattern is implemented through the PubSubJS [22] library, which establishes the event-oriented communications component [14,16]; the WebGL technology is implemented through the three.js library. Within this context, the event router module listens and sends messages, coordinating the communications between application modules and 3D objects. Figure 2 represents a summary of this architectural concept and Figure 3 how application modules are organized and how they communicate between each other.

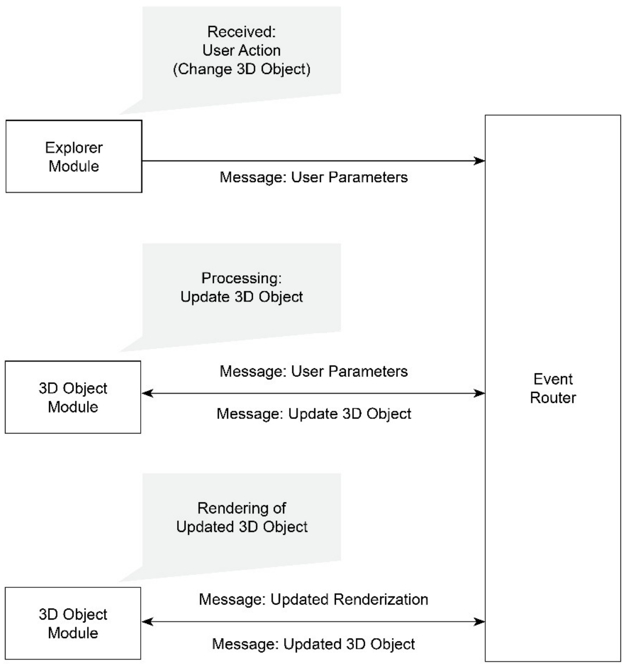

Figure 2. Overview of the event router and event-oriented communications proposed by CW4P. Source: adapted from [10].

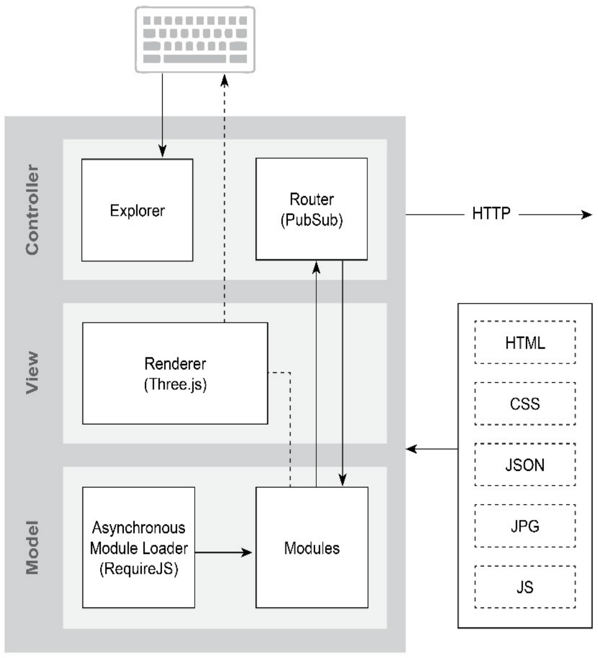

Figure 3. Event flow proposed by CW4P. Source: adapted from [10].

2.2. *CrystalWalk Client Application (CWAPP)*

Main application of the CrystalWalk architecture, CWAPP is an extension of the CW4P framework and is responsible for implementing all user interaction functional requirements. As illustrated on Figure 4, CWAPP is divided in 3 main components: CW4P framework, application's additional modules and a Ruby on Rails interface (deployable) for the delivery of the application files.

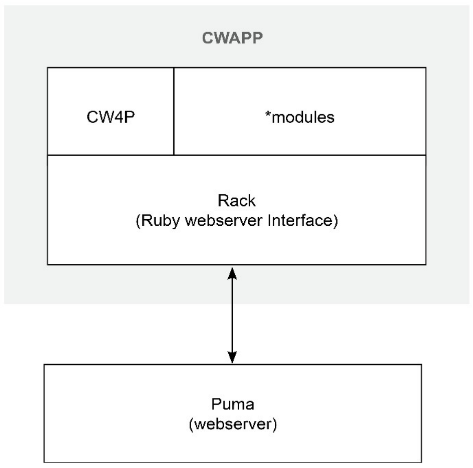

Figure 4. CWAPP application components. Source: adapted from [10].

As proposed by CW4P, overall system architecture implements the Model-View-Controller (MVC) design pattern [6,7,11]. Following the MVC specification, view generates mouse or keyboard input events that are captured by controller. With the help of one or more models, controller interprets the input events and maps user's actions to commands that generate new state events. Once again, controller captures them, generating new events for view, which finally displays the changes back [14–17].

In CWAPP implementation, the initialization of the application modules and data structures that define the logic of the pattern, as well as their dependencies, is performed by the main application module (main.js). Model is responsible for managing the data structures and updating the main scene elements: motif, unit cell, and crystal structure—through their primitive entities: atoms, cells, Miller planes and directions. Finally, view renders the scene elements and displays the user interface, communicating its status to the controller. Figure 5 describes how main modules and interconnections are organized. Figure 6 is an illustrative screenshot of the CWAPP application running in a Browser.

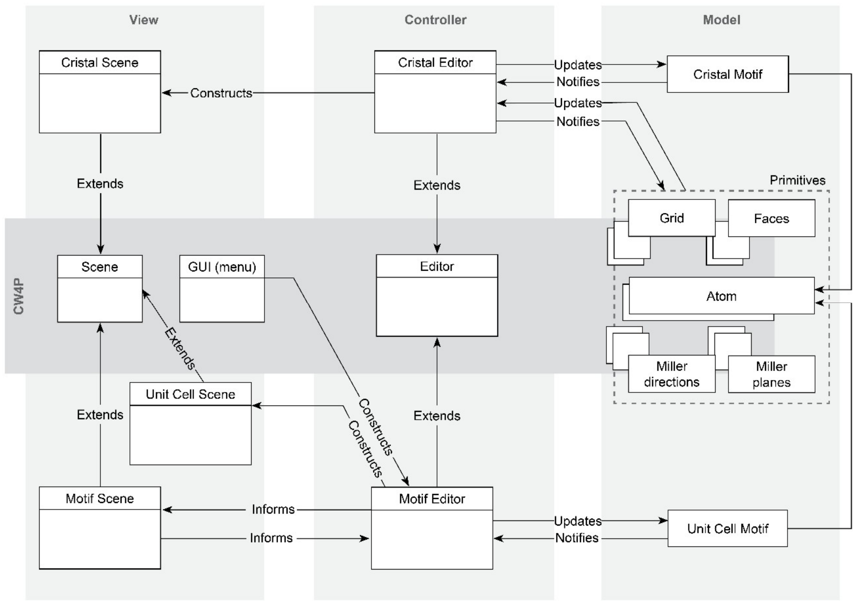

Figure 5. Implementation of the CW4P design pattern in the CWAPP application architecture. Source: adapted from [10].

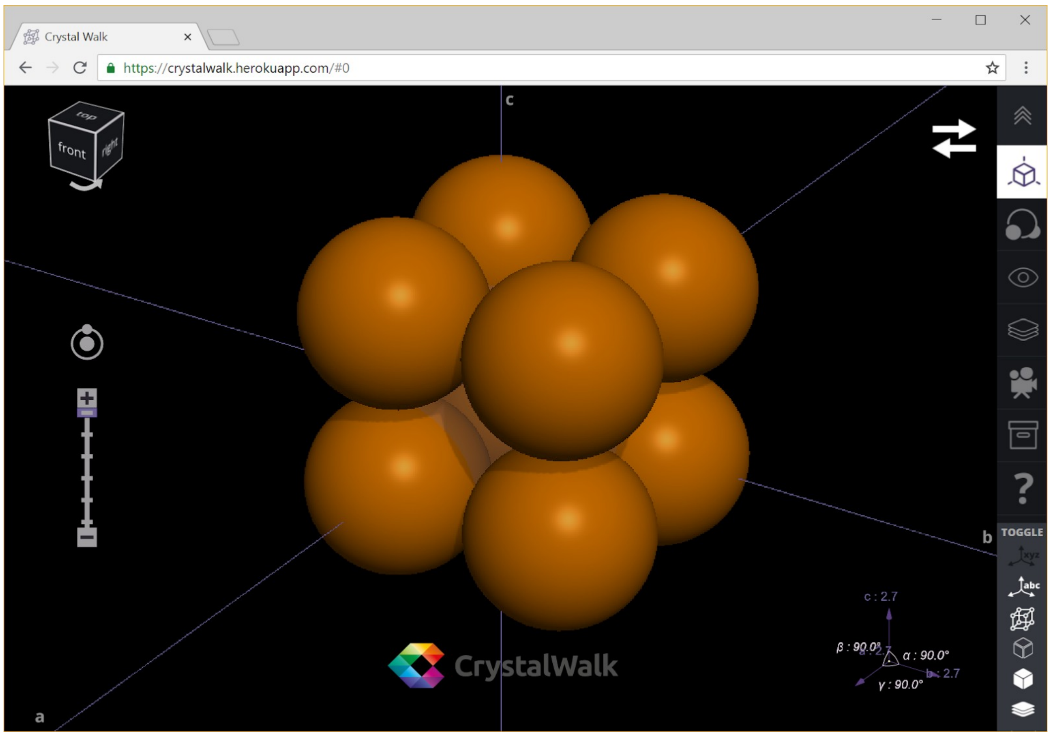

Figure 6. Screenshot of CWAPP interface running on a browser. Source: [8].

2.3. CrystalWalk's URL Shortener and Persistence API (CWLY)

Another component of the CrystalWalk architecture, CWLY is the application responsible for storing, managing, and transferring data between the server and the main client application (CWAPP). Implementation strategy adopted standards and technologies capable of supporting specified CrystalWalk's architecture—such as Heroku cloud development platform, the Ruby on Rails framework, the use of distributed systems (SOA), standardized interfaces for resource management (REST) and high-scalability hybrid (NoSQL/relational) database.

CWLY is divided into three interdependent components: the URL shortener, the persistence API, and the data management interface. The application also has an interface for implementation in Heroku (deployable) developed in Ruby. Figure 7 illustrates this architecture.

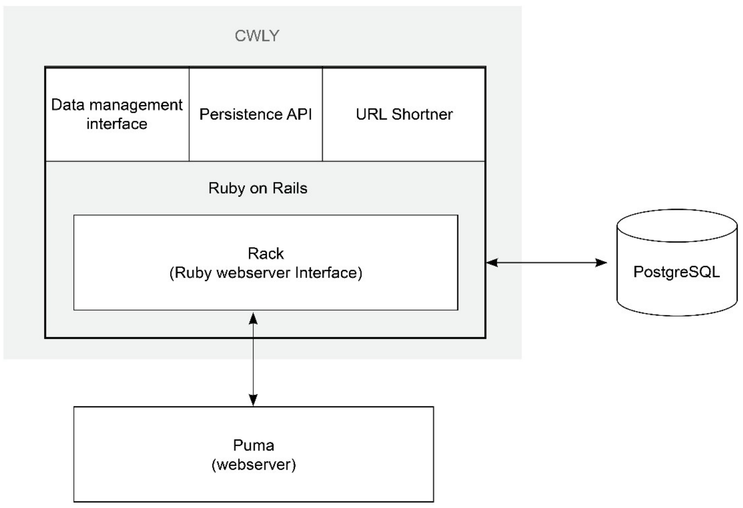

Figure 7. CWLY application components. Source: adapted from [10].

The data persistence API component performs the data storage and retrieves JSON documents in the database. This component has a REST interface [14–17], whose routes are listed and detailed in the Table 3.

Table 3. Description of routes, verbs and parameters of the CWLY REST interface.

Description	Verb	Path	Request Parameters	Response
Stores a JSON document specified by the data parameter	POST	/add	url=server data={"structure": "CubicSimple", "title":"Demo 1"}	{"slug":"$id"
Redirects a shortened URL to the address previously defined in the url parameter, appending the unique identifier $id to the destination address	GET	/$id		HTTP/1.1 301 Moved Permanently Location: server/#$id
Retrieves stored JSON documents that that matches specified search criteria	POST	/qv	qs="search string"	[{"data":{"structure": "CubicSimple", "title":"Demo 1"}}...}
Retrieves a stored JSON document based on a unique identifier $id	GET	/$id.json	JSON	{"data":{"structure": "CubicSimple", "title":"Demo 1"}}

Source: adapted from [10].

Upon receiving a store request, the data persistence API component registers the JSON document in the database, returning a unique identifier for each transaction. The URL shortening component uses this identifier to redirect a shortened URL address to the primary application address, query the database, and retrieve the document. Figure 8 illustrates the operation of this mechanism.

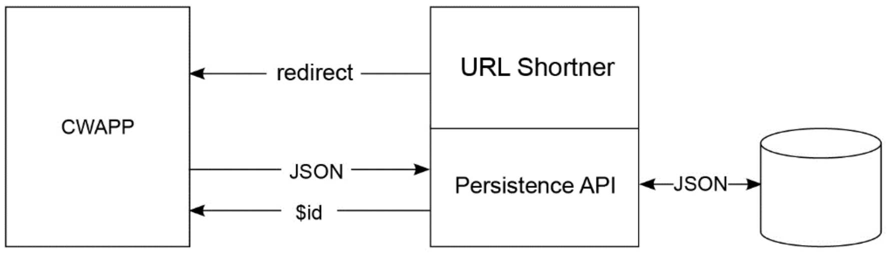

Figure 8. CWLY application flow diagram. Source: adapted from [10].

In alignment with CrystalWalk's architecture, standards and technologies, a hybrid scheme (NoSQL/Relational) was used, adopting specifically PostgreSQL and the jsonb data type due to its superiority in terms of availability and elasticity for the application [23]. The proposed hybrid (NoSQL/Relational) database schema is specified in the Table 4. Finally, the data management component is a restricted interface that allows the location and management of documents stored in a database.

Table 4. Description of the hybrid database schema (NoSQL/ Relational), referring to the application CWLY.

Table	Column	Data Type	Description
documents	url	VARCHAR(255)	Redirect URL for the CWAPP application.
documents	slug (*)	VARCHAR(255)	Unique identifier of the JSON document.
documents	jsonb	JSON/NoSQL	JSON document data structure.

Source: adapted from [10].

Figure 9 demonstrates an illustrative example of CWLY use. The image was exported from CWAPP for illustrating the Fluorite structure to a material sciences class. The image contains a QR code and an URL (http://cw.gl/lw) which when typed or scanned in a browser allows students to access CWAPP application and interact the 3D model online.

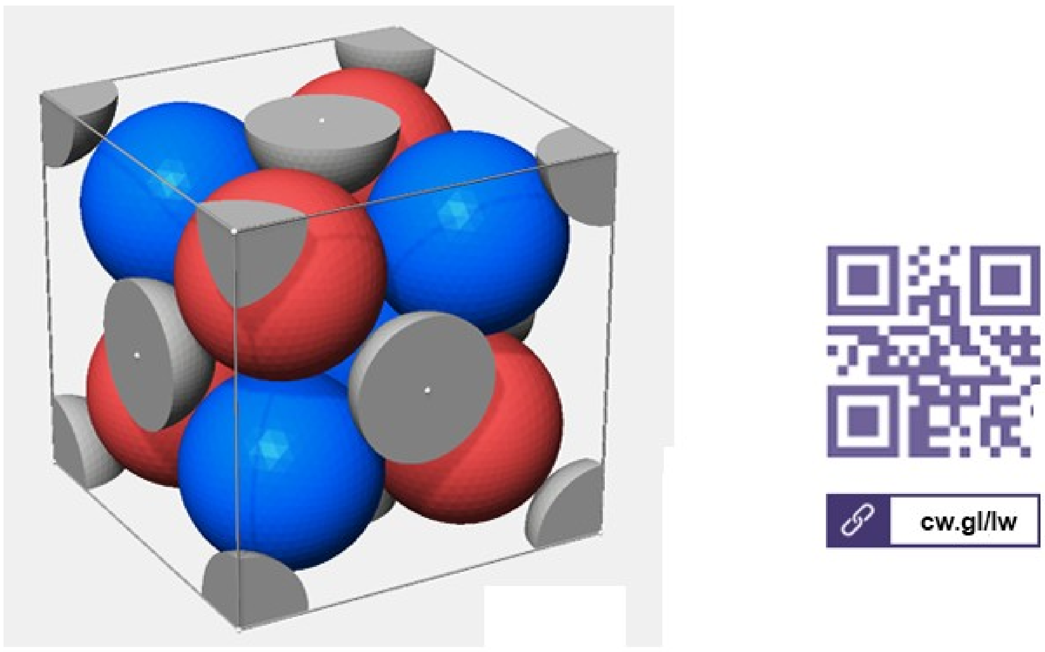

Figure 9. Fluorite structure with a link and QR code to access the online model. Source: [8]

3. Results

Functional instances of all proposed architecture components were deployed and evaluated against specified requirements [1], but also preliminary evaluated against implementation success, performance, and compatibility. A full featured online instance was made available for the preliminary evaluation and reported implementation results can be verified by accessing instance's available URL and using debugging functionalities available on most modern browsers.

From the architectural perspective, as detailed in Section 2, an instance of the CWLY application was implemented using in the PaaS Heroku service, using a free web-type [24] Dyno with Puma application server and PostgreSQL add-on [25] of the Hobby Dev type, setup in a NoSQL/relational hybrid database schema. A specific domain was registered for this purpose, which was successfully configured on the Amazon Route 53 DNS resolution IaaS service using the free entry level category of service. The instance exhibited fluidity in preliminary testing, attesting that the server and domain routing services were correctly configured, as well as the proposed data persistence mechanisms for saving users' models and sessions. All persistence API REST interface routes were verified, demonstrating the storage and recovery of JSON documents from the database. A CWAPP application instance was also deployed on Heroku's PaaS service, using another free web Dyno with a Puma server and, as described in Section 2. As an extension of the CW4P framework, proposed architecture implemented the AMD specification using the RequireJS library; the publish-subscribe pattern through the PubSubJS library and the WebGL technology through Three.js library. The Ruby on Rails interface was correctly implemented, and the application files were properly delivered through the proposed dynamic module loading. The Heroku Dyno instance under the free service category may hibernate after a 30-min inactivity period, requiring additional 10 s to resume instance functions. After the initial base CW4P framework modules and application files delivery (approximately 330 KB), remaining application modules and interface libraries were loaded on demand by the router module, consuming 6.6 MB. A complete usage of application resources calls for the remaining components, summing up to a maximum 8.4 MB, was loaded asynchronously upon user request. Modular design also optimizes browser's cache, reducing data transferring in subsequent accesses. Finally, graphics settings (polygon count and rendering quality) is automatically adjusted according to the processing capabilities of the used device. These optimizations are particularly interesting for smartphone device access, since most recent devices supports 3G access and can provide minimal HTML5/WebGL compatibility.

From the functional requirements perspective, proposed architectural strategy provided a flexible platform capable of addressing all problems identified by students and professors in existing crystallographic software and didactic materials used in materials science and engineering classes. According to the guidelines proposed by UNESCO [2,3], CrystalWalk successfully attended to functional requirements deemed significant for a broader education modernization context.

Following the approach proposed by Deveria [26], user acceptance tests were evaluated against specifications using different browsers across different desktops, tablets and smartphones. Application testing focused on the latest versions of IE, Firefox, Chrome and Safari due to its HTML 5 support consistency. Although HTML 5 support is still incipient on mobile devices, application did perform satisfactory on devices with OpenGL ES 3.0 support. Application compatibility results were consistent with Deveria [26], having most recent Chrome versions providing the best results.

It was observed that for most of the tested operations and functionalities, application did perform in a stable and responsive manner, demonstrating the effectiveness of reactive manifesto's principles of elasticity, resilience, responsiveness and message driven communications [5–7,14]. Although sporadic, crashes were experienced during the execution of features that require the intensive use of computational resources, such as the replication of large numbers of atoms and the computation of differential geometries used in the representation of partial atoms. As per example, generating a empty space representation from an unit cell containing 60 high definition atoms may take up to 4 minutes on the standard testing platform. Performance and instabilities were attributed to the currently immature hardware and software support of WebGL standards, as noted by Deveria [26]. As means to minimize undesirable effects, alerts are displayed to users warning before confirming the execution of these functions.

Although present paper focuses on the architectural aspects of the project, it's worth mentioning that as metric of success from the functional requirements perspective, CrystalWalk was accepted for International Union of Crystallography (IUCR)'s crystallographic software database inclusion [27], and announced as a new tool for teaching concepts of crystallography to non-crystallographers at its official journal [28].

Lastly, advanced users can access the platform's source code repository online at the official CrystalWalk repositories [29–31]. Table 5 contains a summary of project's deployed proof-of-concept instances as well GitHub repositories where the source code and further detailed technical documentation and instructions is available.

Table 5. CrystalWalk's applications, deployed proof-of-concept instances and repositories.

Application	Proof-of-Concept Instance	GitHub Repository
CWAPP	https://crystalwalk.herokuapp.com	https://github.com/gvcm/cwapp
CW4P	https://cw4p.herokuapp.com	https://github.com/gvcm/cw4p
CWLY	https://cw.gl/0	https://github.com/gvcm/cwly

Source: adapted from [10].

4. Conclusions

Justified by the lack of proper didactic tools, state-of-the-art research revealed enhancement opportunities in the development of a novel crystallographic software focused on the educational support to materials science and engineering curricula. CrystalWalk's functional requirements were determined by classroom iterations amongst students and professors, which were later investigated and confirmed by a thorough systematic review of existing crystallographic software. Systematic review revealed opportunities for the development of a new crystallographic software, but also a more broad educational opportunity for an online platform that could be generalized to broader educational uses for 3D content creation, delivery, editing, and annotation.

CrystalWalk's functional requirements led architectural strategy towards an exploratory research aiming at identifying state-of-the-art technologies for the development of interactive web applications, leading to the use of HTML5/WebGL technology and ThreeJS high-level library. Exploratory research led to the adoption of the reactive manifesto, free software philosophy and other paradigms and technologies proposed by state-of-the-art. A solid architectural strategy was seemed fundamental not only to support a complex challenge demanded by application's functional requirements, but also projects strict budget restrictions which demanded team efficiency and the use of free services.

In order to test and verify the proposed application architecture, software components were deployed and evaluated: CWAPP was implemented as an extension of the CW4P framework, implementing the AMD specification using the RequireJS library; the publish-subscribe pattern through the PubSubJS library and the WebGL technology through ThreeJS library. CWLY demonstrated the storage and recovery of JSON documents from the database, implementing specified data persistence mechanisms for saving users' models and sessions. Hybrid (NoSQL/Relational) PostgreSQL implementation using Heroku's partitioned virtualization environment although experimental e non-orthodox at the time of deployment, has satisfactory supported application's requirements of availability, elasticity, costs and performance.

Sporadic instabilities were experienced during certain intensive tasks, although application performed well even on the most resource restricted devices - such as smartphones. User alerts and confirmations were implemented as means to mitigate primitive HTML5/WebGL support and minimize undesirable effects. Despite of these restrictions, all specified software requirements were successfully implemented, demonstrating the effectiveness of architecture's principles of elasticity, resilience, responsiveness and message driven communications.

Performance assessments and preliminary user testing indicated that the proposed architecture was deemed capable of exerting effective and positive impact as an accessible didactic platform for building accessible 3D interactive educational applications for the web, supporting classes with collaborative persistence mechanisms and didactic features. The CrystalWalk application was evaluated under action-research premises and has successfully solved most of the problems identified in the systematic review, delivering a didactic package has effectively enhanced support for teaching and learning activities.

Author Contributions: R.M.L.N. is the head of the research group and idealizer of the CrystalWalk Project. F.B. is the main researcher, idealizer, developer and maintainer of the CrystalWalk Project. A.M.R. is a researcher and contributor of the CrystalWalk Project. R.C.d.M., A.K. and T.S. are technical developers of the CrystalWalk Project. A.G.d.S is a contributor of the CrystalWalk Project. All authors contributed on the writing, reviewing and submission of this manuscript.

Funding: This research had the financial support of the following institutions: the Brazilian National Council for Scientific and Technological Development (CNPq), the National Nuclear Energy Commission (CNEN) and the Institute of Nuclear Power and Energy Research (IPEN) by the use of its facilities, funding of staff and equipment used in this research.

Acknowledgments: The authors would like to thank all the students, professors and researchers involved in the project, the Brazilian National Council for Scientific and Technological Development (CNPq), the National Nuclear Energy Commission (CNEN), the Institute of Nuclear Power and Energy Research (IPEN) by their institutional and financial support to this research.

Conflicts of Interest: The authors declare no conflict of interest.

Availability of Data and Material: CrystalWalk project applications' source codes and documentations are released under the MIT License and is available from its official website at http://gvcm.ipen.br/CrystalWalk.

Ethics Approval and Consent to Participate: Not Applicable to this research.

References

1. Bardella, F.; Montes Rodrigues, A.; Leal Neto, R. The use of crystallographic software as educational support to materials science and engineering. *J. Mater. Ed.* **2018**, *40*. in press.

2. UNESCO. *Information and Communication Technologies in Teacher Education: A Planning Guide*; UNESCO: Paris, France, 1998.
3. UNESCO. *World Education Report: Teachers and Teaching in a Changing World*; UNESCO: Paris, France, 1998.
4. What Is Free Software. Available online: https://www.gnu.org/philosophy/free-sw.en.html (accessed on 3 June 2018).
5. The Reactive Manifesto. Available online: http://www.reactivemanifesto.org (accessed on 3 June 2018).
6. Lampesberger, H. Technologies for web and cloud service interaction: A survey. *Serv. Oritented Comput. Appl.* **2015**, *10*, 71–110. [CrossRef]
7. Pattrasitidech, A. Comparison and Evaluation of 3D Mobile Game Engines. Master's Thesis, University of Gothenburg, Gothenburg, Sweden, 2014.
8. CrystalWalk Official Project Page. Available online: http://gvcm.ipen.br/crystalwalk/ (accessed on 3 June 2018).
9. Waerner, M. 3D Graphics Technologies for Web Applications: An Evaluation from the Perspective of a Real World Application. Master's Thesis, Institute of Technology, Linköping University, Linköping, Sweden, 2012.
10. Bardella, F. Crystalwalk: An Educational Interactive Software for Synthesis and Visualization of Crystal Structures. Ph.D. Thesis, Nuclear and Energy Research Institute (IPEN-CNEN/SP), University of São Paulo (USP), São Paulo, Brazil, 8 July 2016.
11. Parisi, T. *WebGL: Up and Running*; O'Reilly Media: Sebastopol, CA, USA, 2012.
12. Barbosa, J.F.V. Ambientes Urbanos Virtuais Para a Web. Master's Thesis, Universidade do Porto, Porto, Portugal, 2013.
13. Three.js. Available online: https://github.com/mrdoob/three.js (accessed on 3 June 2018).
14. Reactive Programming versus Reactive Systems. Available online: https://www.lightbend.com/reactive-programming-versus-reactive-systems (accessed on 3 June 2018).
15. Philip, G.C. Software design guidelines for event-driven programming. *J. Syst. Softw.* **1998**, *41*, 79–91. [CrossRef]
16. Patterns for Large-Scale JavaScript Application Architecture. Available online: http://addyosmani.com/largescalejavascript (accessed on 3 June 2018).
17. Zakas, N.C. *Maintainable JavaScript: Writing Readable Code*; O'Reilly Media Inc.: Sebastopol, CA, USA, 2012; pp. 53–122.
18. Amazon Route 53 Pricing. Available online: http://aws.amazon.com/route53/pricing/ (accessed on 3 June 2018).
19. Heroku Dyno Types. Available online: https://devcenter.heroku.com/articles/dyno-types (accessed on 3 June 2018).
20. Why AMD? Available online: https://requirejs.org/docs/whyamd.html (accessed on 3 June 2018).
21. RequireJS Official Project Page. Available online: https://requirejs.org (accessed on 3 June 2018).
22. PubSubJS Repository. Available online: https://github.com/mroderick/PubSubJS (accessed on 3 June 2018).
23. PostgreSQL JSON Data Typs. Available online: http://www.postgresql.org/docs/current/static/datatype-json.html (accessed on 3 June 2018).
24. Heroku Dyno Limits. Available online: https://devcenter.heroku.com/articles/limits (accessed on 3 June 2018).
25. Heroku Postgres Addon. Available online: https://addons.heroku.com/heroku-postgresql (accessed on 3 June 2018).
26. Can I Use? WebGL 3D Canvas Graphics. Available online: https://caniuse.com/#feat=webgl (accessed on 17 April 2016).
27. CrystalWalk IUCR Software Database. Available online: https://www.iucr.org/resources/other-directories/software/crystalwalk (accessed on 3 June 2018).
28. Bardella, F.; Montes Rodrigues, A.; Leal Neto, R. CrystalWalk: Crystal structures, step by step. *J. Appl. Cryst.* **2017**, *50*, 949–950. [CrossRef]
29. CWAPP Codebase. Available online: https://github.com/gvcm/CWAPP (accessed on 3 June 2018).
30. CW4P Codebase. Available online: https://github.com/gvcm/CW4P (accessed on 3 June 2018).
31. CWLY Codebase. Available online: https://github.com/gvcm/CWLY (accessed on 3 June 2018).