From design to production: State-of-the-art web user interfaces to operate ALICE offline and online system

G C Raduta[1]
[1] CERN, Geneva, Switzerland
Email: george.raduta@cern.ch

Abstract. The ALICE Experiment at CERN's Large Hadron Collider is currently undergoing a major upgrade during Long Shutdown; this entails a new Online-Offline computing system and a new set of reliable and performant graphical user interfaces. After deployment, the updated system is expected to read an estimated throughput of 27 Tb/s of raw data and index 800 of Gb/s reconstructed data. The newly developed graphical interfaces are to be used 24 hours a day, 365 days a year in the control room by the subsystem experts and on-call support staff. The interfaces are being built based on a common library developed in-house whose goal is to provide the core functionalities and building blocks, and on modern web technologies meant to ensure support for both Runs 3 and 4. This approach secures a consolidated user experience that is efficient, resilient, easy to maintain, and will bring live, straightforward control of the experiment and useful information to the collaboration. This, in turn, will help in our mission to increase the usage of the LHC beam, maximizing the recorded data and potentially leading to new discoveries. Furthermore, in order to guarantee the stability of the platforms, state-of-the-art technologies and practices were applied to build a continuous-integration and continuous-deployment environment to ensure high-quality code, faster releases, and reduced mean time to resolution. This paper provides an in-depth look at the newly developed web-based components including their features and architecture as well as the automated deployment workflow used for software quality assurance.

1. Introduction

ALICE (A Large Ion Collider Experiment) [1] is a detector at the CERN Large Hadron Collider (LHC) [2] whose general purpose is represented by the heavy ion collisions, proton-proton, nucleus-nucleus, and proton-nucleus collisions. This, in turn, facilitates the physics studies of strongly interacting matter, more specifically the properties of Quark-Gluon Plasma.

Following the successful Run 1 (2010 – 2013) and Run 2 (2015 – 2018), the LHC entered the Long Shutdown 2 period which allowed teams to upgrade their components in preparation of Run 3, due to start in 2022. The ALICE experiment took this opportunity [3] to replace its computing system with a new one, called O^2 (Online-Offline) [4], which will allow for faster data acquisition and online reconstruction.

The new, common system, O^2, will consist of approximately 500 nodes collecting roughly 100x more data than in Run 1, i.e. an estimated value of 27 Tb/s raw data, storing 800 Gb/s reconstructed data. In order for the experiment to be efficiently managed, observed, and controlled on a continous basis by the coordinators in the control room and remotely by detector experts and on-calls, a new set of stable and intuitive user interfaces is being developed.

We will now briefly put into context all the graphical interfaces that will be showcased in this paper. Thus, it all starts with the configuration and creation of a data acquisition workflow through the ALICE Experiment Control System User Interface. This allows users to set up their specific environment per detector and transition it through its life cycle. Next, in order to check that adequate data for physics analysis has been taken, users can visualize data samples with the Quality Control User Interface. Moreover, the collaboration is composed of more than 15 detector teams and multiple support ones. This means that a point of truth repository based on operational metadata is necessary in order to be able to keep track of previous states of the system. Thus, the Bookkeeping User Interface allows either users to manually insert system updates or software components to automatically store operational parameters. Finally, it is imperative for users to follow the chain of running processes in an environment and investigate in case of failures. For that we have developed, the InfoLogger User Interface, which features two running modes (Live and Query) and displays the system logs in an efficient and informative manner.

Experience from LHC Runs 1 and 2 [5] constitutes the base on which the requirements of the new O^2 project were built. An important objective is to deliver a unified look and feel for all User Interfaces (UIs). Such a goal aims to secure an intuitive user experience (UX) when alternating from one application to another, reducing the learning time and chances of errors while enhancing productivity.

Moreover, as the ALICE O^2 Graphical User Interfaces (GUIs) are to be used for the next two Runs (3 and 4), over an estimated time frame of approximately 10 years, they should be developed with effortless maintainability and great reliability in mind. Thus, state-of-the-art technologies and practices were applied to build a Continuous-Integration and Continuous-Delivery (CI/CD) environment which ensures high-quality code, faster releases, and reduced mean time to resolution.

2. Design Overview

2.1. Web Applications Concepts

The ALICE data acquisition facilities at LHC imply live and remote operation of the system by multiple teams, thus the next generation of user interfaces are built based on leading industry standards. This type of environment requires easy to share tools (via URL) as decentralized client applications able to provide remote access without any prior installation or compilation. ALICE O^2 web applications are shipped in individual packages with the design being split into front-end and back-end, each being developed under different methodologies.

The ALICE O^2 Interfaces are being built as Single Page Applications (SPAs) following a Model-View-Controller (MVC) architectural pattern, thus taking full advantage of loading a single web page and dynamically rewriting its content without the need to reload it.

The front-end is being developed with MithrilJS which uses Hyperscript and the concept of virtual DOM; an object which describes a DOM tree [6]. The framework's main advantage is the ability to recreate virtual nodes [7] after each rendering cycle. The template engine then calculates the differences in the body content and only modifies the needed DOM elements. Nowadays, thanks to the powerful JavaScript engines, these can be recreated hundreds of thousands of times in less than a millisecond.

For back-end, Node.JS [8], an open-source, cross-platform JavaScript runtime environment, was chosen. Being designed as single threaded and working with an event loop of callbacks it offers great performance managing concurrent requests and adds value when building large network applications.

2.2. ALICE O^2 WebUI Framework

To ensure a common experience across all ALICE O^2 UIs and to ease development of all the needed tools, an in-house library called aliceo2/web-ui [9] was developed. It provides:

- Core services, controllers, and building blocks that can easily be plugged into web applications;
- A set of CSS Components following ALICE guidelines to ensure a unified look and feel;

- Compatibility across operating systems (Mac, Linux, Windows) and devices (laptops, tablets);
- Real-time data transport;

An HTTP server is provided by the O^2 framework and is developed based on Express (a minimal and flexible Node.js framework). Through this server, we ensured that all interfaces will be kept up to date and secured while providing custom REST and WebSocket Application Programming Interfaces (APIs) which offer a bi-directional communication between client and server through the WebSocket protocol [10]. Moreover, the communication between the two parties is secured via JSON Web Tokens [11]; tokens which are generated after user authentication and validated through yet another module that integrates CERN Open ID.

ALICE O^2 UIs bring users a full set of data gathered from different third-party applications. Thus, various data connectors are also provided such as for Consul (the key-value store that hosts the O^2 system configuration repository) and multiple SQL databases.

3. ALICE O^2 User Interfaces

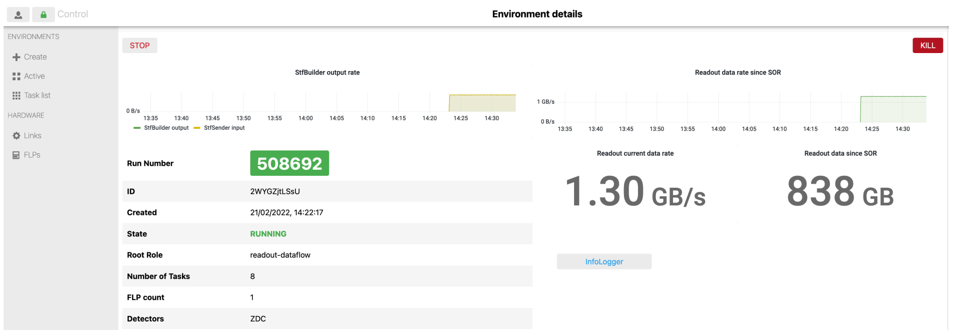
3.1. AliECS User Interface

The AliECS (ALICE Experiment Control System) [12] is a custom built, microservices oriented, integrated solution for the ALICE experiment. It is used to configure, control and launch data-driven workflows inside a computing cluster. Each computer, named FLP, is fitted with multiple Common Readout Units (CRUs) [13] or Common Readout Receiver Cards (C-RORCs) [14] hardware depending on their detector affiliation. The main unit of data collection in AliECS is an environment, i.e. an internally consistent workflow of data processing tasks, along with the detector configuration parameters, hardware settings, and software resources. It represents a data acquisition activity performed by the system, and multiple environments can run in parallel.

The new web-based AliECS GUI supersedes the previous generation of ECS, delivering to users a forthright way of controlling the O^2 data collection. The interface between the two services is setup via gRPC [15], an open source, cross-language RPC (Remote Procedure Call) system.

AliECS GUI enables users to overview the state of the FLP cluster and setup data acquisition workflows. All detector teams can work in parallel via the implemented locking mechanism and detector view. Thus, users first choose a detector and respective servers to work with, after which the interface simplifies the view and filters data based on the selection.

Setting up an environment can be very complex which is why the configuration skeletons for running environments are generated dynamically on page loading based on a YAML template mechanism which allows for quick changes without a need for any of the frameworks to be restarted. The user is expected to configure it via predefined values and intuitive widgets which in turn load step by step based on user input and affiliation. For example, to add the use of a component, users will do it via a toggle which in turn will enable new configurable options. Combinations of configurations can exceed hundreds, thus to further simplify the process, users can save configurations for future use for their colleagues or load an already existing one.



Run Number	508692
ID	2WYGZjtLSsU
Created	21/02/2022, 14:22:17
State	RUNNING
Root Role	readout-dataflow
Number of Tasks	8
FLP count	1
Detectors	ZDC

Readout current data rate: 1.30 GB/s
Readout data since SOR: 838 GB

Figure 1. O^2 AliECS GUI –State transition panel

Once the necessary configuration settings are selected and an environment is created, a redirect will take the user to a new page. As shown in figure 1, this page provides live environment state information via Grafana [16], a transition panel from which the environment can be controlled, and panels per FLP with the state of multiple core tasks running: deployed, configured, running or error.

Moreover, an initial setup page was developed to ease the execution of configuration updates on CRUs by enabling/disabling link components, reducing the number of commands needed to be run in a terminal from a few hundred to a few clicks on the interface. The hardware information used is retrieved from Consul via a custom connector imported from the aforementioned aliceo2/web-ui library.

3.2. Quality Control User Interface

The O^2 Quality Control (QC) [17] replaces and combines the former Online Data Quality Monitoring and Offline Quality Assurance into one system, gathering analysis from user-defined algorithms in both parts of the O^2 System, asynchronous and synchronous. With over 15 detector teams as users and stakeholders, QC is critical in identifying issues early in the data process to provide good quality runs and have the obtained results stored in ROOT histograms. Thus, an interface, QCG, incorporating JSROOT [18] was developed so that operators have a quick way of visualizing the data. QCG converts the histograms into JSON objects after which it plots them; as in figure 2. Users are then able to create, save and share their own custom layouts based on which they filter objects, add ROOT display options and analyse the history of the objects and their evolution.

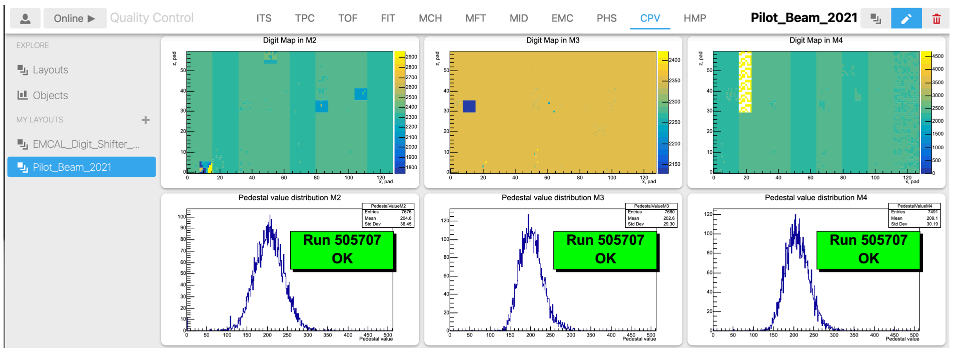


Figure 2. O^2 Quality Control GUI – as used by CPV detector

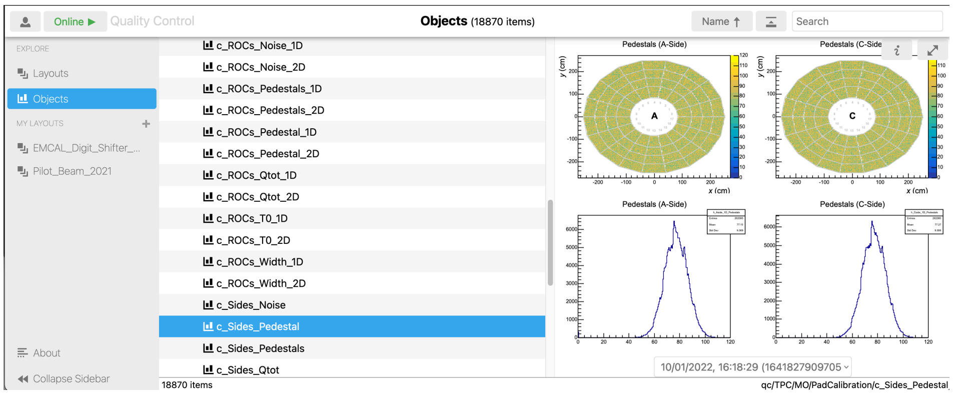


Figure 3. O^2 Quality Control GUI – Virtual list of objects in online mode

Moreover, through the use of the Consul-Service Discovery, it is also possible to view the latest versions of objects coming from QC during the execution of an environment while at the same time viewing its latest representation (figure 3). Due to the high variety of objects as well as the large number of versions per object, a virtual list concept was implemented. Thus, only the objects visible to the user, based on their screen size, will be created while the others will be loaded or removed.

3.3. Bookkeeping User Interface

The ALICE detector is being run in multiple shifts by hundreds of users remotely and in person. To ensure a proper history view and transparency, a new system, Bookkeeping, was developed. This helps users keep track of data acquisition configurations, conditions, and operational interventions in the experimental area. It not only allows users to create entries about the state of the system but it is also capable of receiving and analysing notifications from other components when events are triggered. From an operation point of view, it is expected to be amongst the most used applications from this set of interfaces.

3.4. InfoLogger User Interface

In order to follow the full chain of operations, O^2 benefits from a logging system, InfoLogger, that has its own interface empowering users to follow live updates from the system and investigate accordingly. With a rich set of filters that can be applied and a live mode implemented with the WebSockets protocol, it becomes a core component for understanding and following the whole detector workflow during data acquisition.

The inbound rate from all other components can exceed hundreds of messages per second, and that is why the GUI implements a virtual table, a similar concept like the one used for QGC. The messages are loaded in the background and their corresponding visual elements are created only when the user navigates through them, otherwise they are removed.

4. Implementation and Deployment

4.1. ALICE O^2 User Interfaces in Run 3 detector commissioning

The ALICE O^2 UIs and associated online tools are packaged together in what is known as the "FLP Software Suite" and are deployed on a weekly basis to the experimental area, after an automatic set of integration tests has validated its correctness. Deployment is done via Ansible [19], an open source tool which enables the automation of configuration of systems and deployments.

4.2. Test-Driven-Development and Continous Integration

Test-Driven-Development (TDD) is a software development practice in which developers start the addition of a new feature by implementing a set of unit tests that describe and validate the desired behaviour. Research [20] has shown that while the completion of a feature may take up to 16% longer, the quality of the code will also show an increase of 18%.

ALICE O^2 User Interfaces are expected to be used for both Run 3 & 4 of the experiment, meaning that stability and maintainability are imperative. Thus, the aforementioned GUIs benefit from a constantly updated set of not only unit but also integration and E2E (end-to-end) tests, with an average of 80% code coverage. Moreover, an automated pipeline runs before each code change which analyses the code, searches for potential vulnerabilities, and validates the changes against the tests. This ensured a successful pre-commissioning period over the last couple of years with no critical errors being encountered in the process. The previously mentioned pipeline also ensures a valid packaging of the GUIs followed by automatic delivery to software package repositories hosted at CERN. Once the operation is successfully completed, the FLP suite will run a set of E2E tests, developed with Puppeteer [21], an open source tool provided by Google. The tests validate the compatibility of the software with the rest of the components. Over the years, release validation and production deployment of a new version of the FLP suite including running a compatibility environment through AliECS Core has decreased from 2 weeks to approximately 6 hours, allowing the team to release more confidently on a more regular basis.

5. Conclusion

The ALICE O^2 User Interfaces offer a set of web-based tools which take full advantage of the latest technologies and best practices. These are developed following leading industry standards with the main purpose of bringing intuitive, efficient, and stable interfaces to users for the best part of this decade during the upcoming Runs 3 and 4 of the ALICE experiment.

By creating a custom web library that offers a set of visual components for a unified experience and multiple services which allow a quick and easy integration with other tools, we heavily reduced the setup time when creating a new user interface and we ensured a standard across ALICE O^2.

Future work includes a live notification system for both AliECS and Bookkeeping GUIs to support the oversight of the data acquisition chain, while Quality Control GUI will broaden its current layout capabilities with a richer filtering experience and a comparison view. As development will continue, more tools will be integrated and new features added. The overall aims however will stay the same: constantly ensure high quality code, reduce mean time to resolution and ensure regular releases by complementing the existing CI/CD environment.

All in all, we continue to deliver GUIs based on a consolidated user experience that will be efficient, resilient, easy to maintain, and will bring live, straightforward control of the experiment and useful information to the collaboration. This, in turn, will help in our mission to increase the usage of the LHC beam, maximizing the recorded data and potentially new discoveries.

References

[1] Aamodt K et al. (ALICE Collaboration), Journal of Instrumentation 3, S08002 (2008)
[2] Evans L, Bryant P. LHC machine. Journal of instrumentation. S08001 (2008)
[3] Abelev B et al. (ALICE), Journal of Physics G: Nuclear and Particle Physics 41, 087001 (2014)
[4] Adam J et al. (ALICE), Tech. Rep. CERN-LHCC-2015-006 / ALICE-TDR-019, CERN (2015)
[5] Carena F, Carena W, Chapeland S, Barroso V C, Costa F, Dénes E, Divià R, Fuchs U, Grigore A, Kiss T et al., Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment
[6] World Wide Web Consortium. Document object model (DOM) level 3 core specification (1998).
[7] Chęć D, Nowak Z. The performance analysis of web applications based on virtual DOM and reactive user interfaces. InKKIO Software Engineering Conference 2018 Sep 27 (pp. 119-134). Springer, Cham.
[8] Node.js, https://nodejs.org/en/, accessed: 2022-02-17
[9] O2 WebUI, https://github.com/AliceO2Group/WebUi/tree/dev/Framework, 2022-02-18
[10] Fette I, Melnikov A. The websocket protocol. (2011).
[11] JWT, https://jwt.io/introduction/, accessed: 2022-02-17
[12] Mrnjavac T, Alexopoulos K, Barroso VC, Raduta G. AliECS:a New Experiment Control System for the ALICE Experiment. EPJ Web of Conferences 2020, EDP Sciences.
[13] Mitra J, Khan SA, Mukherjee S, Paul R. Common Readout Unit (CRU)-A new readout architecture for the ALICE experiment. Journal of Instrumentation. 2016
[14] Borga A, Costa F, Crone GJ, Engel H, Eschweiler D, Francis D, Green B, Joos M, Kebschull U, Kiss T, Kugel A. The C-RORC PCIe card and its application in the ALICE and ATLAS experiments. Journal of Instrumentation. 2015
[15] gRPC A high performance, open-source universal RPC framework, https://grpc.io/, 2022-02-18
[16] Grafana - The open observability platform, https://grafana.com/, accessed: 2022-02-18
[17] Konopka P, von Haller B. The ALICE O2 data quality control system. InEPJ Web of Conferences 2020 (Vol. 245, p. 01027). EDP Sciences.
[18] Bellenot B, Linev S. JavaScript ROOT. InJ. Phys.: Conf. Ser. 2015 (Vol. 664, p. 062033).
[19] Ansible IT Automattion, https://www.ansible.com/, accessed: 2022-02-18
[20] George B, & Williams, L (2004). A structured experiment of test-driven development. *Information and software Technology*, *46*(5), 337-342.
[21] Puppeteer-Tools for Developers, https://github.com/puppeteer/puppeteer/, accessed: 2022-02-18