MICRO FRONTENDS - A NEW MIGRATION PROCESS FOR MONOLITHIC WEB APPLICATIONS

A. Asko*, S. Deghaye, E. Galatas, A. Kustra, C. Roderick, B. Urbaniec, CERN, Geneva, Switzerland

Abstract

Numerous standalone web applications have been developed over the last 10 years to support the configuration and operation of the CERN accelerator complex. These applications have different levels of complexity, but they all support hundreds of users for essential activities. A monolithic architecture has been utilised so far, tailoring the standalone applications to specific accelerator needs.

The global GUI technology landscape continues to evolve quickly, with most GUI technologies typically reaching end-of-life within 1-to-5 years. Keeping up-to-date with technologies presents a major challenge for the GUI application maintainers, with larger monolithic applications requiring long migration cycles which impede the introduction of new functionalities during the migration phase.

To tackle the above issues within the CERN Controls domain, a new Micro Frontend architecture has been introduced and is being used to gradually migrate a large and complex AngularJS-based web application to Angular. This paper introduces the new generic architecture, which is not tied to any specific web framework. The development workflow, challenges, and lessons learned so far will be covered. The differences of this approach, particularly when compared to monolithic application technology migrations, will also be discussed.

INTRODUCTION

Since its inception, web development has been characterized as a rapidly evolving field, continuously advancing with fresh tools and solutions for organizations and businesses. Over time, numerous web applications have been created to facilitate the control, monitoring, and configuration of various components within CERN's accelerator complex. In numerous instances, given the intricacies inherent to a setting like CERN, the development of these applications has extended over long periods, often spanning years.

Given the volatile nature of web technologies and the industry as a whole, these applications would ultimately rely on already outdated technologies, shortly after they were deemed feature-complete. In turn, this would trigger a significant migration effort, transitioning to the latest "standard" web technology stack, a process that could also span several years. Simultaneously, fresh requirements emerging from the accelerator complex sometimes require the development of new applications, sometimes using yet another technology stack.

The development of these applications in this manner, can lead to being trapped in a repetitive cycle, endeavoring to deliver the necessary functionalities while also remaining aligned with the swiftly progressing market technology landscape. This raises the question, why bother to try to align with modern technologies? There are two main reasons for this:

1. To profit from the global industry investment in modern, supported technologies and deliver solutions that meet user requirements and user experience expectations.
2. To use technologies that recent generations of engineers are both familiar with and aspire to use, and in turn provide them with transferable skills. This aligns with one of CERN's core missions, which entails training and preparing the upcoming generations of engineers to confront the challenges that will inevitably arise in the future.

Despite the aforementioned reasoning, it was clearly desirable to find an alternative approach.Rather than constantly embarking on repetitive, technology-driven, lengthy re-development upgrades, a path was sought to be able to introduce new technologies in a more gradual manner. The aim being to avoid lengthy full application re-development cycles and be able to add new functionality and improvements to existing applications, leveraging new technology in the process. The solution has been to adopt a new architectural approach, known as Micro Frontends, gradually replacing outdated, monolithic applications. Leveraging this new development style has the potential to redefine the landscape of web applications for accelerator controls.

OBJECTIVES AND GUIDING PRINCIPALS

Currently, CERN accelerator controls rely on more than 30 web applications that have diverse levels of complexity and are based on various technology stacks. To reach a more maintainable situation, it is imperative to tackle this heterogeneity and strive for standardization. Several web application frameworks are currently in use, including AngularJS (which reached end-of-life in January 2022), Angular, and VueJS. This presents an additional challenge for development teams, requiring competence in multiple frameworks and their unique approaches or methodologies. The pursuit of standardization on a single framework will facilitate the concurrent migration of multiple applications, enable more flexible allocation of development teams, and promote the reuse of common solutions across the overall application portfolio.

* anti.asko@cern.ch,
stephane.deghaye@cern.ch,
epameinondas.galatas@cern.ch,
ajob.kustra@cern.ch,
chris.roderick@cern.ch,
bartek.urbaniec@cern.ch

To initiate the micro frontends migration process effectively, it is imperative to have well-defined objectives, that serve as guiding principles upon which decisions can be made regarding the applications concerned.

Objectives

The paramount objective for the new migration process is to achieve a smooth coexistence between new and existing applications. During the incremental upgrade, newly migrated components of the applications must integrate smoothly with the old application, under the same domain. In this way, end-users are not exposed to the complexity of managing multiple domains in what can already be a complex application landscape.

The next objective is to ensure that it remains feasible to concurrently develop both the legacy and newly-migrated sections of the applications. It is essential to anticipate that certain elements cannot be migrated instantaneously, and there may be justifiable reasons for further extensions in the legacy applications. Likewise, the capability to deliver essential bug fixes to both segments of the applications should also be maintained. This dual development approach ensures a smooth transition without compromising critical functionality or stability.

Guiding Principals

The frontend is inherently the most dynamic part of an application and subject to more frequent technology changes. Therefore, prioritizing lightweight solutions is recommended, limiting business logic in frontend components, and separating responsibilities among application components for a modular ”divide and conquer” approach. This lays the foundations for implementation of future changes, with minimal impact on other parts of the system.

All of the architectural and development aspects outlined above should have a minimal impact on development team performance or application maintainability. Optimizing the application development cycle, by as decribed, should result in the swift delivery of maintainable code, that can be extended and adapted for future needs.

ARCHITECTURE

Overview

Micro frontend architecture, similar to the microservices architecture, advocates for independent, deliverable applications that are composed into a greater whole. For the plethora of monolithic applications that are part of the CERN accelerators controls, adopting a micro frontend architecture opens the door to begin incrementally upgrading them. In practice, it allows development teams to replace potentially obsolete applications bit-by-bit, and eventually eradicate them completely. By definition, this involves replacing large complicated applications with smaller composed ones, which can results in simpler applications that are actually easier to develop. Even in the most basic terms, smaller applications will most likely have a small code base, which in-turn, provides a friendlier environment for testing, continuous integration and delivery. Lastly, teams can potentially become more autonomous and decide what technologies and development style best fits the smaller application needs, without impacting the larger overall application composed for end-user consumption.

It is important to understand that there is no silver bullet in any architectural style. The aforementioned advantages can be countered with potential pitfalls. Some micro frontend implementations can lead to a duplication of effort in the development process, increased complexity for application integration, and additional communication needs between different application teams.

There are multiple technical ways to implement the micro frontends architecture [1], according to the environment and the expected results. For CERN accelerator controls, two distinct approaches were investigated.

Single Page Applications

A Single Page Application (SPA) (Fig. 1) combines multiple smaller applications into one. By creating smaller applications, the different parts e.g. header, main content (different action pages), footer etc. can be developed separately and then be assembled together. This can be accomplished using different sets of tools and frameworks that best suit the use case of the smaller applications or distinct deveopment teams involved.

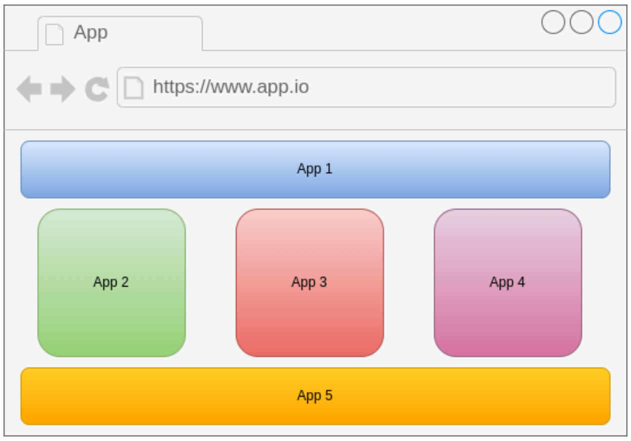


Figure 1: SPA Micro frontends architecture.

Autonomous Applications

The autonomous applications architectural style pulls the separation one level higher. Instead of slicing the application on each page section, it advocates that each application is a complete entity on its own. Multiple autonomous micro applications (Fig. 2) share common functionalities whenever needed and are still served to the end-user as one. Each application uses a different URI, forcing the refresh of the page on entering each application.

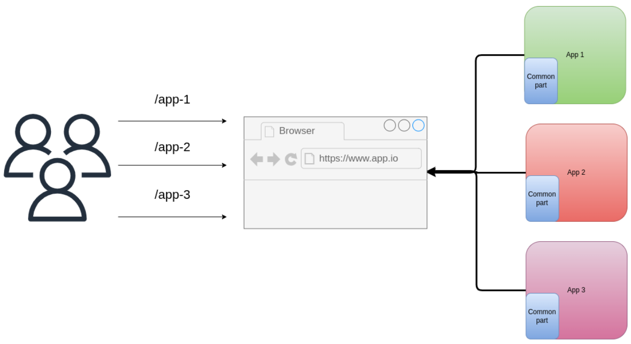


Figure 2: URI Micro frontends architecture.

ADOPTION AT CERN

Both of the aforementioned architectural solutions were evaluated and validated in the current infrastructure. Instead of constructing isolated basic prototypes for testing, real usage scenarios were selected within existing applications. Once developments were completed, the final results were deployed in production to perform a full and real-world validation.

Initially, it was chosen to evaulate the first approach, using a SPA architecture, as it would provide the smoothest end-user experience. It was decided to migrate the "fault creation" page of the Accelerator Fault Tracking (AFT) application [2] to the new technology stack (Angular) and integrate it to the existing solution (AngularJS). The new application representing the "fault creation" page was developed as an autonomous web component that could be integrated into the pre-existing solution (Fig. 3) . In detail, a custom HTML component was created based on the new technology stack and encapsulating its own HTML structure, styles and behavior.

Web components come with a variety of advantages. They are known for their re-usability, enabling the encapsulation of specific functionality, for deployment across different sections of an application or even across multiple applications. Additionally, web components impose isolation, which helps prevent unintended conflicts with other elements in the application. Their interoperability is also a notable feature, allowing integration with various frontend libraries and frameworks. Lastly, web components can be seen as contributing to maintainability, by allowing the encapsulation of complex functionality into separate, manageable pieces, thereby improving the organization and sustainability of the overall codebase.

To evaluate the second approach, using autonomous applications, it was decided to migrate the Controls Middleware (CMW) [3] configuration module of the Controls Configuration Data Editor (CCDE) [4]. The new part of the web application was built and deployed as a separate, independent frontend module, accessible through its own URI. Once developed, the module was integrated with the monolithic application, creating the end-user impression that it is actually part of it.

The Micro frontends architecture using different URIs offers multiple advantages. It empowers independent and isolated development on discrete modules by autonomous teams, leading to faster development cycles and granular deployments. It also ensures agility and scalability with dynamic loading and isolation mechanisms in place between the micro frontends. Sharing of common functionalities is achieved through shared libraries, in the form of dependencies, which become part of each application.

CHALLENGES

Digging deeper into the development phase, various limitations and challenges were encountered. Some were common to both architectural approaches, while others were unique to each approach.

Application Size

It quickly became apparent that the total size of the applications nearly doubled. For the SPA web components approach, the growth in size was attributed to the combination of two major frameworks (AngularJS and Angular), within a single application. Each framework introduced additional libraries that contributed to the overall increased size. In contrast, the expansion in size when using distinct applications and URIs was primarily due to the content downloads associated with each microfrontend. In both scenarios, the increase in size had the potential to result in significant delays in the initial loading time of the applications, particularly if users had limited bandwidth available for their internet connection, therefore deteriorating the user experience significantly.

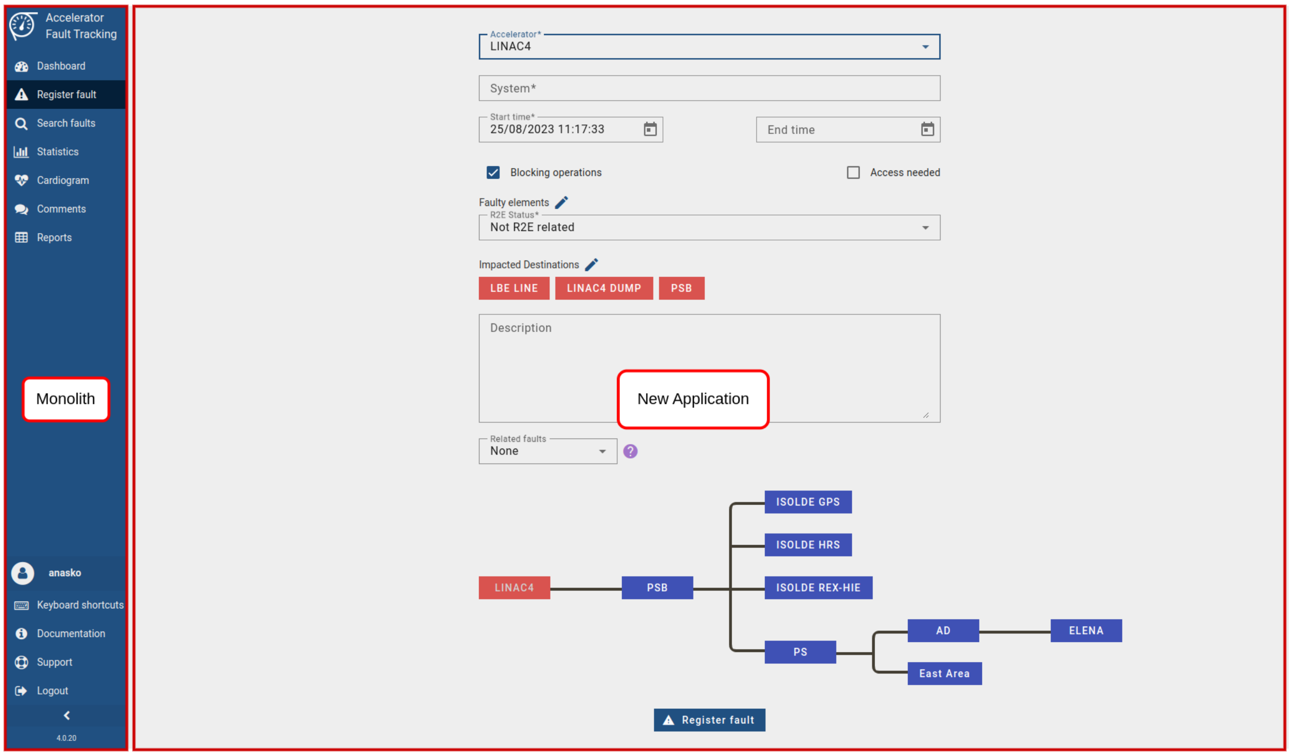
This decline in performance was mitigated by caching the static content of the applications. This method entails storing the static resources on the user's device via their web browser. Rather than loading the application on every visit, the cached version is presented to the user, resulting in significantly improved loading times (up to tenfold, depending upon the user's available bandwidth). Nevertheless, it requires meticulous configuration to guarantee that users consistently access the latest static content while still reaping the advantages of caching for enhanced performance.

Look and Feel

The aesthetics of the new applications differed notably when compared to the existing solutions (Fig. 4). The updated user interfaces provide a more modern appearance and introduce new interaction components. Therefore, it is important to acknowledge that in certain instances, this transition might lead to user frustration or disorientation and impact productivity. To minimize this concern, significant efforts were made to optimize the user experience and address any usability issues.

State Management

State management and communication between the new and legacy frontends posed significant challenges in both of the architectural approaches. The integration of the new SPA web component micro frontend into the existing workflow



Accelerator* LINAC4

System*

Start time* 25/08/2023 11:17:33 End time

☑ Blocking operations ☐ Access needed

Faulty elements

R2E Status* Not R2E related

Impacted Destinations

LBE LINE LINAC4 DUMP PSB

Description

Related faults None

Figure 3: Fault creation micro fronted web component integrated in the existing application.

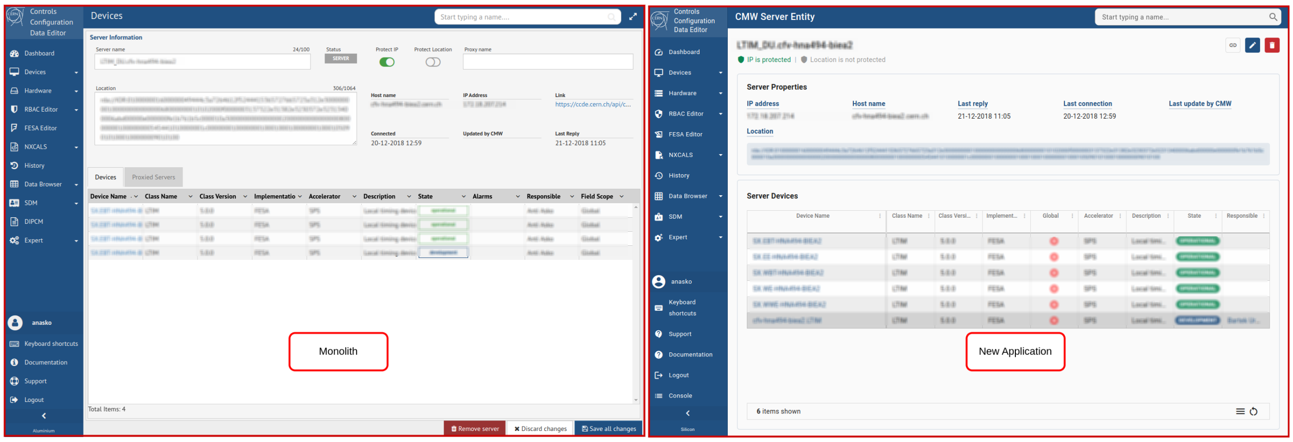


Figure 4: User interface differences between monolithic application and micro frontend on CMW module.

created several complications in terms of interactivity. For the distinct applications using unique URIs, when each application functions as a standalone solution, it limits interactivity between them. Navigating between the two applications relies solely on hyperlinks, which must be treated as external links, directing users to an entirely distinct application.

Addressing the complexity of communication within the SPA web components solution involved the implementation of explicit guidelines for data sharing, event propagation, and interaction between applications. Emphasis was placed on granting primary control to the old application, resulting in a streamlined and simplified approach for the new one. Conversely, with the distinct applications using unique URIs, it was opted to compartmentalize the new application to such an extent that interactivity between it and the legacy application was rendered unnecessary, apart from hyperlinks connecting the two.

Development and Testing

It is important to note that development and testing complexities saw a significant increase with both approaches. With ongoing development in both the legacy and new applications, the development team had to manage multiple frameworks and their differences. Additionally, cross-cutting concerns like authentication and security demanded attention on both fronts, occasionally resulting in duplicated efforts. Testing individual micro frontends and their interactions posed challenges, as did ensuring that changes don't break compatibility with other applications. While there isn't a definitive solution, efforts were made to increase vigilance and incorporate more standard best practices into the development and testing processes such as setting up automated build and continuous integration (CI) pipelines to ensure application builds and tests automatically with every change.

ANALYSIS OF APPROACHES

Upon the successful completion of the two proof of concepts, the conclusion was that the second approach, using distinct applications with unique URIs, aligns better with the accelerator's controls environment and development cycle. Digging deeper into a comparative analysis between monolithic development and the new approach, it was observed that while there may not be significant disparities between old and new applications in the short term, substantial differences become evident in the context of the cost and impact of renovations to adapt to new technologies.

To illustrate, the CCDE is an application developed over roughly 5 years. It was estimated that the renovation of the frontend, without incorporating any additional enhancements, would take around 3 years to complete. This monolithic approach would constrain development of any additional application features during the renovation, as well as locking in development resources over a long time period.

In a constantly evolving setting like accelerator controls, a long-term commitment of this nature, which could potentially encounter significant delays, might be viewed as a relatively unnecessary step in the user's eyes. As users typically prioritize the quick availability of new features, irrespective of the underlying architecture or technology.It is also essential to acknowledge that there is always a potential risk that the selected technology stack and frameworks could become outdated in the market, potentially stalling any on-going multi-year monolithic renovation efforts.

In the context of the micro frontends architecture within the same scenario, partitioning the application into smaller units allows for their prompt deployment as they reach completion relatively quickly. This approach enables the prioritization of renovations focused on delivering the most essential features, while also offering the flexibility to continue providing features in the old application whenever necessary.

With micro frontends, renovation becomes an integral aspect of the application evolution, and both users and developers can experience rapid results. Overall, this approach might slightly extend the total duration required for a complete renovation when compared to a monolithic application replacement. It can also potentially introduce new challenges related to the technologies employed. However, this can be mitigated by adopting newer tools in subsequent micro frontend developments. Nevertheless, the micro frontend architecture naturally introduces the challenge of maintaining multiple technology stacks, which should be kept to a minimum.

From a developers' viewpoint, micro frontends allow them to concentrate on their specific segment of the overall product. This focus enables them to deliver a higher level of testing and quality assurance, rather than grappling with an exhaustive understanding of the overall larger application domain.

In a micro frontend architecture, the deployment process becomes more complicated, as a result of the growing number of applications. At CERN, this was streamlined and automated through GitLab CI pipelines. While not currently used, if the need presents itself, the frontends can be delivered separately depending on which is affected by any newly introduced changes.

LESSONS LEARNED

Exploring micro frontends has provided numerous insights into the entire development cycle. It has prompted recognition of a multitude of potential challenges and a reevaluation of development procedures. It is important to note that all these lessons are context-specific and tailored to the present CERN accelerator controls environment.

Minimal Technical Diversity

One of the prominent arguments in favor of micro frontends, often cited, is the flexibility to use multiple frameworks. However, it was swiftly recognized that this approach presented more challenges than solutions. The divergence between different teams' preferences for various tools threatened to result in an incoherent and unmanageable development landscape. Consequently, a standardized set of tools and frameworks was established as the foundation for all newly developed applications. This decision promotes the cultivation of a shared knowledge base among all teams and promotes a more collaborative development environment overall.

Pragmatic Code Duplication

Inevitably, certain parts of the code are required by both the micro frontends and the legacy applications. The initial direction was to extract the common code into a set of smaller libraries to be shared between the applications. However, although this approach worked well with the micro frontends, it introduced unnecessary complexity and dependencies when applied to the old applications. As a result, the pragmatic decision was taken, to accept a certain level of code duplication between the legacy applications and the new micro frontends. Although this is not an ideal scenario, it proved to be the most cost effective and feasible solution, rather than attempting to force a single solution to fit all cases.

Well Defined Application Scope

It became apparent during the early stages of development, that defining the precise scope of each micro application was of paramount importance. Initally it was foreseen to break things down into numerous small fragments, but it soon became clear that in the long run, this approach would lead to a maintenance nightmare. Consequently, the micro applications were organized into manageable yet realistic chunks based on domain-level considerations. In this scheme, all code functionalities related to a specific domain are treated as an application, as opposed to dividing them into individual pages or even smaller units.

Lightweight Applications

Throughout this endeavor, the unpredictability inherent in the field of web development was once again confirmed. It became increasingly apparent that rapid technological evolution was the norm, with newer tools swiftly gaining prominence, requiring developers to eventually adapt. This realization underlines the importance of structuring the applications to be as ”lightweight” as possible, with the bulk of the business logic residing in a more stable architectural tier, i.e. the backend servers.

NEXT STEPS AND BEYOND

Many accelerator controls applications are currently built on outdated frameworks and tools, making any future development of them, at the very least, a challenging project and, in some cases, nearly impossible. With a year of hands-on experience in micro frontends, working on real production applications, there is now a clear intention to transition all controls web applications to this architecture.

A gradual migration process has already been initiated and elements of the monolithic applications are being effectively substituted with micro frontends. A substantial enhancement in productivity and reduced delivery times has been observed, which is positively impacting user satisfaction.

In the upcoming years, the goal is for micro frontends to evolve into the central solution, with the intention of migrating the majority of the applications to this architecture. It is clear that this is a gradual and long-term undertaking, but the advantages it brings will significantly improve the accelerator controls environment, enabling to deliver faster and more advanced web-based tools to the users.

SUMMARY

Micro frontends introduce a new standard in web application development. Even in its early stages, this architecture has already demonstrated its potential for the future development of large control applications. Based on open-source software and industry-standard tools, this new architecture ensures stability and longevity, provided that the guidelines and best practices described in this paper are followed.

The ongoing migration of monolithic CERN accelerator controls web applications towards micro frontends is progressing as planned. Based on experience gained over the last year, these developments are expected to reduce technical debt, prevent development stagnation in the applications, and accelerate delivery times. Though the early phases of this endeavor are still being navigated, the future looks promising.

REFERENCES

[1] L. Mezzalira, *Building Micro-Frontends*. O’Reilly Media, Inc., 2021.

[2] C. Roderick, L. Burdzanowski, D. Martin Anido, S. Pade, and P. Wilk, “Accelerator Fault Tracking at CERN”, in *Proc. ICALEPCS’17*, Barcelona, Spain, Oct. 2017, pp. 397–400. doi:10.18429/JACoW-ICALEPCS2017-TUPHA013

[3] J. Lauener and W. Sliwinski, “How to Design & Implement a Modern Communication Middleware Based on ZeroMQ”, in *Proc. ICALEPCS’17*, Barcelona, Spain, Oct. 2017, pp. 45–51. doi:10.18429/JACoW-ICALEPCS2017-MOBPL05

[4] L. Burdzanowski *et al.*, “CERN Controls Configuration Service - a Challenge in Usability”, in *Proc. ICALEPCS’17*, Barcelona, Spain, Oct. 2017, pp. 159–165. doi:10.18429/JACoW-ICALEPCS2017-TUBPL01