

# Micro Frontends

A new migration process for monolithic web applications (FR2BCO04)

A. Asko\*, S. Deghaye, E. Galatas, A. Kustra, C. Roderick, B. Urbaniec



On behalf of BE-CSS, CERN

# Introduction

---

Currently, the CERN accelerators controls domain relies on more than **30 monolithic web applications**

- **Diverse** technology stacks  
*(VueJS, AngularJS, Angular)*
- Obsolete or **unsupported** technologies  
*(e.g. AngularJS EoL was on January 2022 & Vue 2 is on December 2023)*
- **Frozen** development due to extensive migrations plans
- **Challenging recruitment** of newcomers

***Can we transform the process to deliver solutions without long development cycles and repeated migrations?***

# Architecture

---

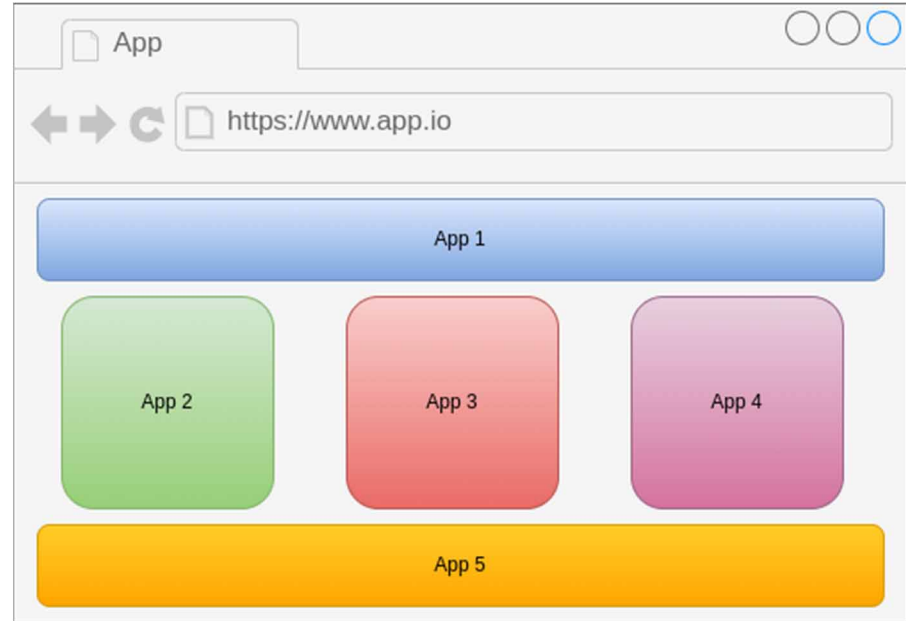
A new architecture known as **Micro Frontends** was introduced to tackle the aforementioned issues

- **Independent**, applications that can be **composed** into a greater one
- **Incremental** upgrade and **eradication** of legacy solutions
- **Simpler** applications with **smaller** codebases
- **Autonomous** teams with different methodologies/technologies

*We evaluated 2 different micro frontend architectures in existing applications*

# Micro frontend Architecture 1 - Single Page Application

- **Encapsulation** of specific functionality and **reusability**
- Impose **isolation** to prevent unintended conflicts with other components
- **Interoperability** allowing **integration** of multiple libraries and frameworks



# Example - Single Page Application

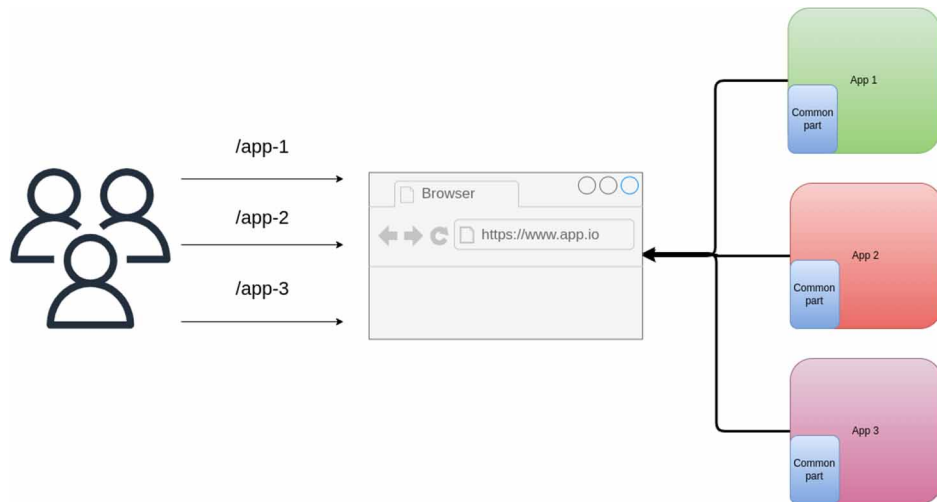
- Your micro frontends are **within** a Single Page Application
- Integration & **communication** between the different applications
- **Urgent** development of small features

The screenshot displays the Accelerator Fault Tracking (AFT) system interface. On the left is a dark blue sidebar menu with the following items: Dashboard, Register fault, Search faults, Statistics, Cardigram, Comments, Reports, a user profile for 'anako', Keyboard shortcuts, Documentation, Support, and Logout. A red box highlights the 'Monolith' label in the sidebar. The main content area shows a form for 'Accelerator LINAC4' with fields for 'System\*', 'Start time\*' (25/08/2023 11:17:33), and 'End time'. There are checkboxes for 'Blocking operations' (checked) and 'Access needed'. Below this is a 'Faulty elements' section with a dropdown menu set to 'Not RZE related'. Underneath are 'Impacted Destinations' with buttons for 'LBE LINE', 'LINAC4 DUMP', and 'PSB'. A 'Description' field contains the text 'New Application', which is also highlighted with a red box. At the bottom, there is a 'Related faults' dropdown set to 'None' and a network diagram showing components: LINAC4, PSB, ISOLDE GPS, ISOLDE HRS, ISOLDE REX-HIE, PS, AD, East Area, and ELENA. A 'Register fault' button is at the bottom right.

*Micro frontend SPA-based upgrade of CERN's Accelerator Fault Tracking (AFT) system*

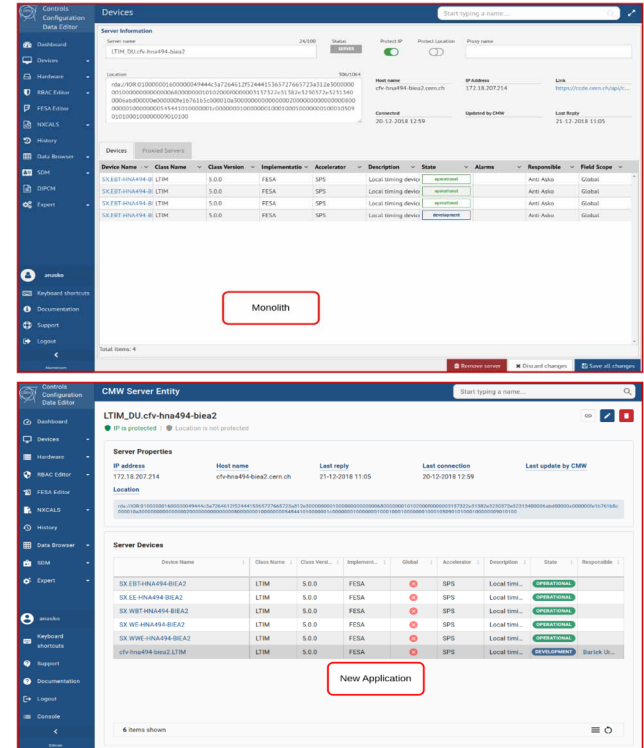
# Micro frontend Architecture 2 - Different URIs

- **Independent** and **isolated** development of each application
- Provides **granular** deployment
- **Scalability** with **dynamic** loading of each application
- **Sharing** of common functionalities across applications



# Example - Different URIs

- Modules can live as **standalone** applications
- **Separation** of different applications and teams
- **Independent** deployment and potential lifecycle



Micro frontend URI-based upgrade of CERN's Controls Configuration Data Editor

# Challenges - Size

---

Application size nearly **doubled** due to the multiple frameworks under the same application

- **Caching** of the static content on the client's side

*After first load, display of the micro frontends is instant*

- **Compression** of the static content during transmission

*Faster loading times depending on the network bandwidth (5.2 MB -> 1.3 MB)*

*24 seconds -> 1.5 seconds*

***Improvements are visible on both Monolith and Micro frontends***



# Challenges - Look & Feel

- Aim to **minimize** the differences as much as possible
- Provide **enhancements** in the usability and ergonomics

Before

After

Micro frontend URI-based upgrade of CERN's Controls Configuration Data Editor

# Challenges - State management

---

**Communication** between the applications becomes more challenging due to unawareness of each other

- Create rules early and use **standard** solutions

*Use standard HTML events (onclick, oninput etc.)*

- **Minimise** communication as much as possible

*Hyperlinks communication*

***State management is a complicated matter and there is no perfect solution***

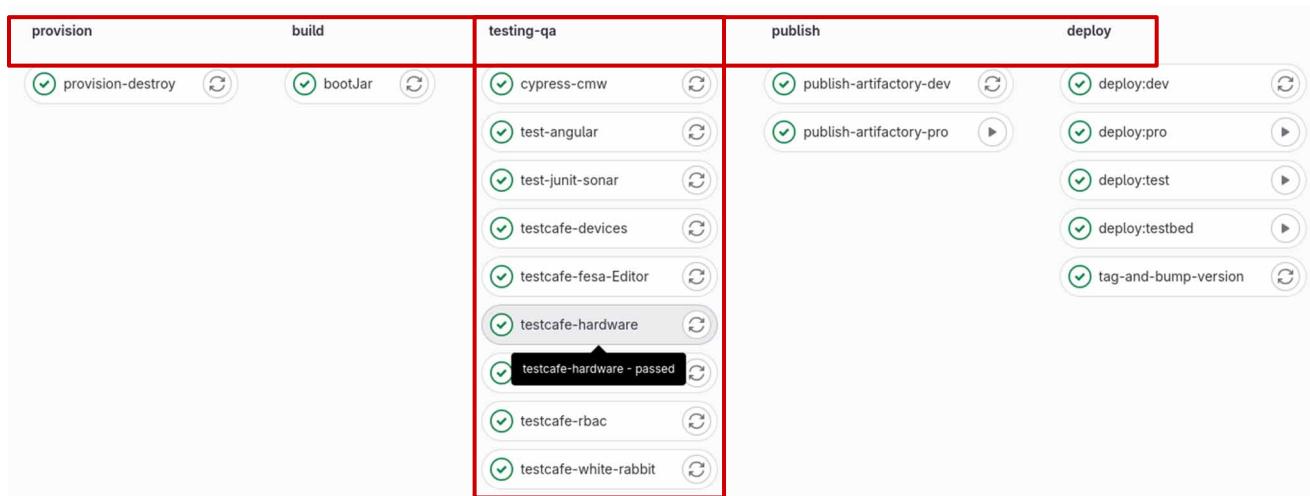
# Challenges - Development & Testing

- **Standardizing** testing processes throughout the full stack

*80% unit/component testing and 20% integration/e2e testing*

- CI/CD throughout the whole development **lifecycle**

*Complete automation of the process, minimizing human interaction*



# Conclusions

---

After evaluating both approaches for the past year, we concluded that the **URI-based solution** is more appropriate for our needs

Lessons learned:

- Establish a **standardized** set of **tools/frameworks** and **processes**
- Some code **duplication** is **inevitable** between the old and new applications
- A well-defined application **scope** should be **defined** early on
- Aim for a **lightweight** solutions if possible with the bulk of the **business logic** in the **backend**

*Micro frontends help to embrace  
inevitable changes in the web development ecosystem*

# Technology stack

---

## Frontend:

- Angular a **mature, feature-rich** framework
- Angular Material a **powerful UI** component library



## Backend:

- Spring boot **ecosystem** a Java based, **enterprise** framework
- Gradle an advance **build automation tool**

