

Micro Frontend-Based Development: Concepts, Motivations, Implementation Principles, and an Experience Report

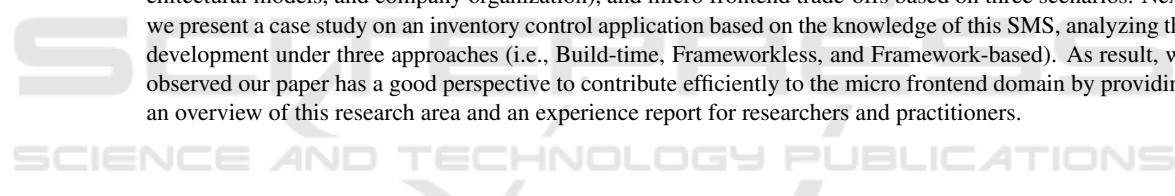
Fernando Rodrigues de Moraes^{1,2} ^a, Gabriel Nagasaki Campos¹,
Nathalia Rodrigues de Almeida¹ and Frank José Affonso¹ ^b

¹*Department of Statistics, Applied Mathematics and Computation, São Paulo State University – UNESP,
PO Box 178, Rio Claro, São Paulo, 13506-900, Brazil*

²*VR Software, 59 Narciso Gonçalves Street, Jardim Cidade Universitária, Limeira, São Paulo, 13484-646, Brazil*

Keywords: Micro Frontend Architecture, Development Approaches, Web Development.

Abstract: Micro frontend is an architectural style that enables us to build large software systems by combining independent micro applications, which can boost different aspects related to the development (e.g., innovation, continuous software delivery), besides increasing the flexibility and scalability of the final application itself. Although there are numerous benefits related to this architectural style, some companies are still hesitant to adopt development based on micro frontends because of a lack of knowledge about concepts, development approaches, architectural models, and organizational aspects of the company. This paper presents the results of a Systematic Mapping Study (SMS) on micro frontends based on 16 studies. The results were synthesized in an important overview that addressed concepts, aspects related to development (i.e., development approaches, architectural models, and company organization), and micro frontend trade-offs based on three scenarios. Next, we present a case study on an inventory control application based on the knowledge of this SMS, analyzing the development under three approaches (i.e., Build-time, Frameworkless, and Framework-based). As result, we observed our paper has a good perspective to contribute efficiently to the micro frontend domain by providing an overview of this research area and an experience report for researchers and practitioners.



1 INTRODUCTION

Designing the presentation layer of a contemporary Web application is a vital task for software development companies. Development teams have shown interest in exploring innovative approaches to create, deploy, and maintain applications efficiently, enabling companies to swiftly and effectively provide value to their customers. In this direction, micro frontends are increasing in popularity in small and large companies, because this technology enables to split off a monolithic frontend into independent and micro applications, developing the same frontend application by different teams (Bian et al., 2022; Mena et al., 2019; Mannisto et al., 2023; Peltonen et al., 2021).

Regarding development, the main purpose of micro frontends is to organize an application (i.e., Website or Web app) as a composition of features that are owned by independent teams, developing its fea-

tures like a full-stack specialist (i.e., from database to user interface) (Shakil and Zoitl, 2020). In parallel, microservice is an architectural style used to build distributed applications organized in a collection of services around business capabilities and independently deployable by automated deployment (Lewis and Fowler, 2014). Analyzing these two technologies, it can be said that micro frontend extends microservice because employs similar concepts. In short, we can segment a Web application into individual modules in a Micro Frontend Architecture (MFA), and implement them independently. The combination of these two technologies enables frontend teams a level of flexibility and quickness analogous to the benefits that microservices offer to backend teams (Mohammed et al., 2022; Lewis and Fowler, 2014; Geers, 2020; Jackson, 2019).

Although there is a relationship between the aforementioned technologies (microservice and MFA), we observed that MFA is still at a premature stage of development and requires deeper and more solid investigation. Despite the notorious volume of docu-

^a <https://orcid.org/0009-0004-8631-9385>

^b <https://orcid.org/0000-0002-5784-6248>

mentation available on the Web, this information still does not come from reliable sources, making companies not feel comfortable using it. This distrust comes from the lack of establishment of concepts, application scenarios, and a clear exposition of benefits that can be achieved. Furthermore, there is a certain contradiction between the concepts, definitions, and architectural models found in these sources of information. Thus, as happened with microservices, teams that were unprepared from a conceptual and technological point of view had severe difficulties in adopting the MFA (Soldani et al., 2018).

This paper presents a study on software development based on micro frontends, whose research interest was the formulation of concepts, integration approaches, management aspects (organizational and development), application scenarios, and micro frontend trade-offs. To establish a fair and broad overview of the aforementioned interests, we have adopted the Systematic Mapping Study (SMS) process proposed by Kitchenham and Charters (2007); Petersen et al. (2015). Next, based on the established panorama, we conducted a case study for a inventory control system, where the following aspects were explored: the organization of the system concerning the architectural model; the company's structural organization about the adoption of micro frontend architectural style; the development team's management; and the experience report through a case study developed based on three approaches. Thus, we identify the following contributions for our paper: (i) a solid panorama of the related to the development based on micro frontend; (ii) the main aspects of the management when designing micro frontend application(s) (MFAApp(s)); (iii) the case study as an experience report for the micro frontend development based on the integration approaches; and (iv) the micro frontend trade-offs.

The remainder of this paper is organized as follows. Section 2 introduces the main concepts related to development based on micro frontends and related work. The main findings of our SMS is reported in Section 3. Section 4 shows the key aspects of micro frontend development based on the integration approaches. Section 5 provides a discussion of results on the main finding. Finally, Section 6 summarizes our conclusions and perspectives for further research.

2 BACKGROUND AND RELATED WORK

In this section, we present the background that contributed to the development of our paper. Initially, concepts of microservices, micro frontends, and three

development approaches for micro frontends are presented. Next, related work on software development based on micro frontends is addressed.

Microservices and Micro Frontends. According to Lewis and Fowler (2014), the microservice architectural style is an approach to developing a single application based on a suite of small services, each one running in its process and communicating with lightweight mechanisms. These services are built around business capabilities and can be independently deployable by fully automated deployment. These features enable teams to be organized around an application, besides boosting continuous software delivery. In parallel, Jackson (2019) defined micro frontends as “an architectural style where independently deliverable frontend applications are composed into a greater whole”. According to this last author, the main benefits of micro frontends can be summarized in three features: (i) more cohesive and maintainable codebases; (ii) scalable organizations with decoupled and autonomous teams; and (iii) incremental development with continuous delivery of software. As observed, these features represent some of the same advantages that microservices can provide (Peltonen et al., 2021).

Development Approaches. The development of applications based on MFA can be conducted through three integration approaches (Jackson, 2019), namely: (i) Build-time; (ii) Frameworkless; and (iii) Framework-based. Before presenting these approaches, two considerations must be highlighted. First, in Frameworkless and Framework-based approaches, the term framework is used or referenced as an architectural framework for supporting the development of software based on micro frontends and not as a framework for Web development. Second, the Build-time approach differs from the others because it does not enable integration at runtime, meaning that micro frontends are packaged and coupled to the main application at compile time. Next, we addressed a description of each approach.

Build-time is the simplest approach to the architectural implementation of micro frontends because it enables the development of micro applications as packages. This approach does not depend on a framework or particular implementation because it imports libraries in a container application, which is responsible for calling each MFA. Although simply, this approach does not meet the main principles when compared to microservices features such as language agnosticism, fault isolation, and independent delivery of micro frontends container (Stefanovska and Trajkovik, 2022; Pavlenko et al., 2020; Lewis and Fowler, 2014; Jackson, 2019).

Frameworkless is a type of approach that does not require frameworks for software development. According to (Frameworkless, 2023), the lack of knowledge that developers have when adopting a framework and the technical debt that this can generate in a project has been one reason for adopting this approach type. Since MFA is technology agnostic, Mannisto et al. (2023) reported an experience based on a Frameworkless approach as a feasible alternative for small organizations, providing improvements in the development and deployment of MFAs. In this direction, we can highlight some elements that can support this approach, namely: (i) Webpack, a Web application packaging tool that can handle Javascript, HTML (HyperText Markup Language), CSS (Cascading Style Sheet), and images; (ii) Module Federation, a tool that enables the use of exposed external modules, allowing the applications' remote code to be consumed to build an integrated application at runtime; (iii) *iFrame*, an HTML element that enables the integration of other HTML elements (e.g., single pages, micro frontends); (iv) Web components, a technology used to create and reuse different Web components with functionality decoupled from the code; and (vi) Signal-based Web Components (Stefanovska and Trajkovik, 2022; Nishizu and Kamina, 2022; Pavlenko et al., 2020), a technology that uses reactive programming paradigm to facilitate the frontend construction with better code organization.

A **Framework-based** approach enables a pragmatic and successful implementation according to guidelines established within an architectural framework used to implement integration between MFAs, avoiding high effort demands (e.g., time) being used to design the final product, still providing built-in mechanisms to overcome anti-patterns solutions, and exposing APIs and plugins to help teams to achieve a highly maintainable micro frontend solution.

Drawing a parallel between the approaches presented in this section, the first two demands that teams implement their code to sustain an MFA, and the last one follows the conventions and restrictions of the frameworks. Although the multi-framework approach has been discouraged (Geers, 2020), we found architectural frameworks that enable the integration of MFAs built in different Web development frameworks (e.g., React¹), without high efforts with API design and micro frontend implementation, namely: (i) single-spa (Single-spa, 2016); (ii) qiankun (Qiankun, 2019); and (iii) Garfish (Garfish, 2021).

As related work, to the best of our knowledge, there is no paper about software development based on micro frontends that approaches the interests of in-

vestigation reported in Section 3. To ensure the originality of our paper, we conducted a literature investigation to establish a solid and complete definition of micro frontends, besides aspects related to the development and real needs to adopt this architectural style by the companies. The evidence revealed that the studies available in the literature addressed different aspects of the investigation, leaving gaps related to the micro frontend architectural style itself and aspects associated with software development.

3 MICRO FRONTENDS

As reported in Section 1, we have adopted the SMS technique to establish a fair and broad overview of software development based on micro frontends. To do so, we elaborated a research protocol based on guidelines proposed by Kitchenham and Charters (2007) and Petersen et al. (2015) that will not be presented in this paper for space reasons. Details of this protocol and the final list of selected studies are presented in full in Moraes et al. (2023). Next, we report the main findings of our investigation in Sections 3.1–3.3, which can be considered for the software design based on micro frontends.

3.1 What Is Micro Frontend?

An MFA is a microservices-inspired architectural style that implements an application as a collection of frontend applications decoupled that must be modeled around a business domain, enabling the building of micro applications as smaller, loosely coupled, and independently deployable components. In an MFA, each user interface (UI) portion can be treated as components or pages and can become a new MFApp, which is developed, tested, deployed independently, and containerized in a unique UI. In parallel, one can say that this modular organization enables the adoption of small autonomous teams with different technological expertise (i.e., own technology stacks and deployment pipelines), reduces the dependency between development teams, boosts the governance of such teams, reduces business complexity, has the potential to encourage innovation, and can optimize the maintenance activity of these applications in the future (Tilak et al., 2020; Noppadol and Limpiyakorn, 2021; Bühler et al., 2022; Pavlenko et al., 2020; Taibi and Mezzalira, 2022; Yang et al., 2019).

Besides the established definition, our research suggests to the stakeholders that the design of an application based on micro frontend must be guided by knowledge and decisions illustrated in Figure 1. In

¹<https://react.dev>

(A), we show the organization of an MFApp in three layers (i.e., frontend, backend, and database), where each column represents a development team working in full-stack mode (Oliveira et al., 2022; Yang et al., 2019). Next, we present possible layout decisions in (B), where the developers must define how the micro frontends will be organized. To do so, the developers can choose two layouts: (i) horizontal, which enables multiple micro frontends per page; and vertical, which enables one micro frontend per page. Regarding composition (C), an MFApp can be designed in three ways, namely: client-side, edge-side, and server-side. In the first, an application retrieves several micro frontends from a Content Delivery Network (CDN), typically using JavaScript or an HTML file as the entry point for each micro frontend. The second builds the view at the CDN level by fetching your micro frontends from the origin and then delivering the outcome to the client. The micro frontends are composed within a view in the third way, where they are cached at the CDN level and ultimately served to the client either at runtime or during compile time (Taibi and Mezzalira, 2022).

Finally, after defining the composition of the micro frontends, it is important to define two additional elements: routing and communication. Routing delineates the route from one view to another, dictating the traversal process. The communication outlines how micro frontends interact with each other, once data sharing is indispensable for both horizontal and vertical divisions (Pavlenko et al., 2020). Section 3.2 discusses routing and communication, as these aspects directly rely on the chosen approach and implementation strategies for this architectural style.

3.2 How Should the Development of MFApps Be Managed?

Before we introduce any appoint to this question, it is important to highlight some aspects related to development based on micro frontends, since the focus here is large-scope and distributed Web/Mobile applications. In this direction, aligning concepts and best practices between microservices and micro frontend reported in Section 2 enables us to establish a modular organization of development teams around the application's business capabilities. In short, such capabilities are smaller, specific features that can boost application scalability, optimize the reuse of features/components, and facilitate the reduction of development complexity. Moreover, the findings of our SMS have revealed some evidence regarding the management of software development based on micro frontends. As MFA is an architectural style that

comes from industrial environments, we can also mention the following benefits related to the decoupled development process, namely: (i) governance, which enables us to organize teams in autonomous vertical teams with capabilities in incremental updates; and (ii) agile behavior, which allows us to develop applications part of the MFA without causing negative impacts on other parts of the software, while optimizing the delivery process of a software product.

Based on the content exposed in this section and the evidence identified in our mapping studies, it is suggested the management of applications based on micro frontends can be guided based on two essential elements: company organization and architecture. Next, we addressed a description of each element.

Company Organization. As emphasized by Jackson (2019), it is essential to establish an organizational approach within the Software Development Life Cycle (SDLC) to achieve success with MFA. The main purpose of an organizational approach is to soften operational complexity through specific measures related to the following items: (i) *automation*, the development of decoupled applications requires automated processes to reduce complexity concerning the team setup. In parallel, such processes must also support the deployment of these micro applications, since the proposed architecture (MFA) increases infrastructure complexity in both: development and deployment (Taibi and Mezzalira, 2022); (ii) *process*, a micro frontend team must deal with the development, testing, and release of applications in short cycles of continuous delivery of software using a pipeline organized into a “production-like environment”. To do so, the software engineers must plan and organize an SDLC process to scale the teams concerning an architecture (Pavlenko et al., 2020; Taibi and Mezzalira, 2022); (iii) *practices*, an MFA requires decentralized processes and tools so that an organization (i.e. development teams) feels comfortable adjusting itself as an autonomous vertical team (Jackson, 2019); and (iv) *quality*, decentralized codebases and teams can represent a challenge for an organization that intends to maintain software quality. In general, it is harder to keep quality in multiple and decentralized processes in SDLC, even considering that micro applications and smaller teams cause less impact in a software ecosystem than in a monolith system.

Regarding operational complexity, we can define organizational processes to reduce possible problems and increase architectural reliability. In this direction, pipelines to deliver and integrate software can be a feasible alternative to overcome such complexity because they automate all the integration processes and software deployment to the codebases of each appli-

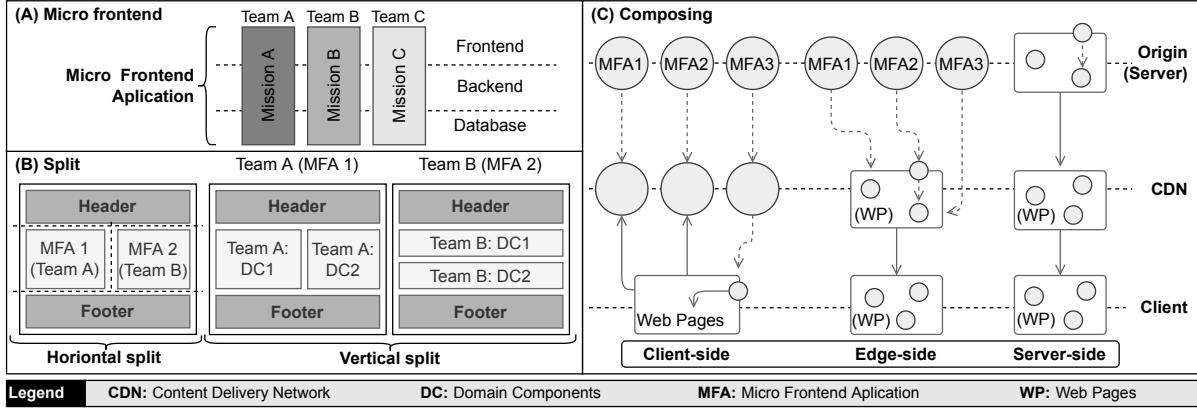


Figure 1: Definitions and decisions about MFApps.

cation (Taibi and Mezzalira, 2022). Furthermore, micro frontend-based development can be defined as the ideal scenario for setting up vertical teams that own different small frontend applications. This organizational model provides these teams greater freedom to develop tools and processes, alleviating architectural complexity and addressing concerns regarding decentralized development practices and team quality standards. Finally, it is recommended that the division of micro frontends be carried out around the business domain to keep each micro frontend in its specialized domain, besides ensuring its decoupling, increasing the quality level, and decentralized practices.

As observed in this section (**Architecture** Element), the adoption of micro frontend as an architectural style can be facilitated by considering trade-offs. The pros and cons analysis of any software project must be mindful of the inherent aspects of any architectural style. Therefore, based on the MFA presented in this paper (see Figure 1), it can be said industrial cases denote that a well-executed implementation of micro frontends, with clearly defined processes, contributes to a more efficient organizational scale in frontend applications (Yang et al., 2019).

Architecture. As we define the company's organizational structure, it is mandatory to establish a development approach. Choosing from different architectural implementations, including the innovative concept of micro frontends, enables us to make this decision by considering the features of each approach. The maturity evaluation of the development team and parameters related to the project's scalability must also guide this decision.

As reported in Section 2, the main choice that has to be defined in an MFA is an integration approach (see **Development approaches** item), namely: Build-time (BT), Frameworkless (FL), and Framework-based (FB). In this architecture type exists an entry-

point application, which can be called an application container so that it can render the initial elements. In short, these elements are used by the user's application to access other pages and elements that can be different micro frontends. Based on the distinct trade-offs associated with each approach, we recommend that each development team conduct a self-analysis to determine the most optimal choice for each project and the organizational context of each company. Table 1 provides a tabular representation of the main features identified in the BT, FL, and FB approaches.

Table 1: Comparison of development approaches.

Features	BT	FL	FB
Modeled around domain	✓	✓	✓
Decentralized governance	✓	✓	✓
Automation culture	✓	✓	✓
Incremental upgrades	✓	✓	✓
Independent deploy	✓	✓	✓
Failure isolation	✓	✓	✓
Architectural boundaries			✓

Analyzing the distribution presented in Table 1, the FB approach has shown full compliance with the set of features identified in our investigation (Moraes et al., 2023). The FL approach failed to address only one feature, while the Build-time (BT) approach did not cover three features (Jackson, 2019). As a result, this analysis can guide a team to make a more assertive decision about choosing a development approach, especially for teams that intend to adopt the use of frameworks (i.e., trade-offs for frameworks) to support the development of MFApps.

As reported in Section 3.1, an MFA can work as a component, element, or an entire page. Concerning the organization, the layout is a decision that the development teams must make before starting the development tasks (see Figure 1 – **Split** area). A micro frontend as an individual component or element can

not be a trivial choice, since it will usually be coupled on a page and will cause increased complexity in terms of maintainability. Therefore, defining the distributed impact on the team responsible for a micro frontend across different pages becomes challenging. A traditional solution that enables the use of components on different pages to share common elements, components, and even domains in a library can be a better strategy than using a coupled micro frontend on multiple pages. Therefore, a (micro) frontend will share a style guide that makes it visually consistent.

The split strategy is an important decision to create composable decoupled micro frontends, as described in Section 3.1. Although we have highlighted two strategies (i.e., horizontal split, and vertical split) in our definition, a third called hybrid split can be used too. In synthesis, the hybrid split enables the combination of both strategies (i.e., horizontal and vertical split). In this sense, we can highlight the composition of a micro frontend via hybrid split due to a micro frontend that needs a horizontal split on the page.

Regarding the development, Module Federation, Web Components, and iFrames have been the technological resources used to create an application container that will provide an entry point to render a user interface and routing strategy to other pages (Wang et al., 2020; Pölöskei and Bub, 2021; Stefanovska and Trajkovik, 2022). Since MFAApps need to communicate with each other, data-sharing approaches must be used in isolated or combined ways, namely (Nishizu and Kamina, 2022): (i) event-driven uses the publication and subscription of events to handle state between micro frontends; (ii) signals work with the emission of signals between micro frontends, using reactive programming principles (Nishizu and Kamina, 2022); (iii) global stores are libraries that use shared and global states for data management; (iv) query parameters and local storage enable to sharing of small data as query parameters in the micro frontend URL, as well as transport a larger data load shared by the browser's local storage; and (v) context (or technology's history) stores information about the stack of previously loaded interfaces, accessed by the container application and distributed among different micro frontends. Of the three integration approaches mentioned in this section (BT, FL, and FB), framework-based implements some data-sharing approaches by default, which can be an important decision-making factor when selecting a framework.

3.3 Do You Need Micro Frontends?

As mentioned in Sections 3.1 and 3.2, the development of applications based on the micro frontend

architectural style brings with it a series of implications. Initially, we identified in our investigation that the successful cases (i.e., developing new applications and/or modernizing the monolith to micro frontends) expressed having more solid knowledge about concepts of micro frontends with strong evidence about the pros and cons of this architectural style (Bian et al., 2022; Wang et al., 2020; Pölöskei and Bub, 2021; Mannisto et al., 2023; Shakil and Zoitl, 2020). These successful cases also reported having a solid understanding of the application being developed and/or modernized and its suitability in the organization of decision-making teams regarding the organization of development teams, layout division, and composition (Bühler et al., 2022).

Through our mapping process, we identified a set of advantages and disadvantages of the use of micro frontends in different organizations, which differ in terms of size and development domain. Lessons learned, benefits, challenges, pros, cons, advantages, disadvantages, trade-offs, limitations, and pitfalls were the most recurrent terms found in our mapping to express the opposing sides of the theme (i.e., micro frontend). In this sense, we grouped a set of features concerning the development teams and company organization that must be taken into consideration when adopting the micro frontend architectural style.

In the MFA, we can organize the development teams around the business domain, which must make local decisions about technology and design choices. Next, we addressed some benefits arising from this modular organization: dynamic teams concerning communication, ease of maintenance (small software units), fault isolation, incremental adoption, independent deployment, modular organization, reusability, scalability, and technology flexibility. Otherwise, the modular organization of the development teams can result in conflicting scenarios related to the company organization and the system, namely:

Scenario 1. Distributed teams are more complicated to manage, as well as being more complex to attribute responsibilities and coordinate activities. In this sense, implementation of cross-system concerns, communication between teams, and sharing of data are the most recurring challenging adversities concerning the company/team organization.

Scenario 2. The modular organization can be related to a series of challenges. For instance, organizing into smaller software units requires special care in managing dependencies between micro frontends. In parallel, communication between micro frontends can significantly increase the complexity of the application with the routing and orches-

tration of the micro frontends so that functionality can be executed. Another aspect inherent to communication, which may be combined with other factors (e.g., data sharing) is performance overhead. Sharing data between an application's micro frontends may require an excessive number of message exchanges, which will cause system overload.

Scenario 3. The development of a decoupled application may suffer arising from differences in the development environment since a micro frontend operates within a user interface context that inherits stylization and shared data (optional). This bottleneck implies the adoption of processes to ensure that the micro frontend developed maintains visual consistency across the entire frontend.

Although the evidence gathered in this section does not reveal a silver bullet solution as a tool for decision-making regarding the need to adopt the MFA, the issues addressed can serve as an important reference for the adoption of this architectural style by concepts presented in Section 3.1, by the suggestions on managing MFAApps reported in Section 3.2, and by the arguments about the real need for this architectural style discussed in this section.

4 CASE STUDY: AN EXPERIENCE REPORT

To evaluate the applicability, strengths, and weaknesses of our study, a case study was conducted. As subject application for our empirical analysis, we have selected an application addressed to the management of product inventory, which will be referenced from this point onwards as **MfaWebApp**². Next, a brief description of our subject application and the empirical strategies is presented.

Subject Application. MfaWebApp is a Web application that was developed to meet customers who need to control product inventory. We organized the interface of this application into three pages, one for each MFA, namely: (i) **Dashboard**, used to visualize graphs that enable the user to check the product inventory input and output data; (ii) **Products**, used for registering, listing, changing, and deleting products that the user intends to control the entry and exit of inventory in a store; (iii) **Inventory**, used to register the entry and exit of products from the store, the user can view the list of operations, such as an inventory extract. Figure 2 illustrates

²<https://github.com/fernandormoraes/mfa-cs>

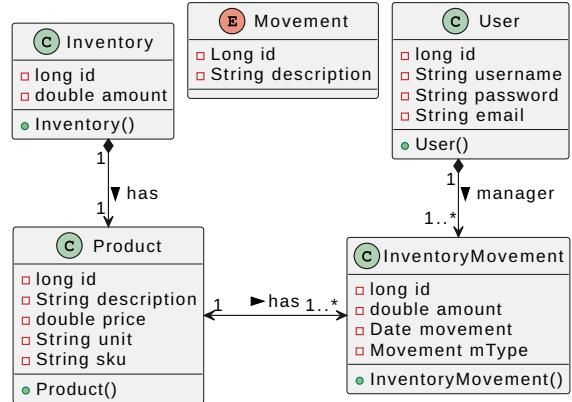


Figure 2: MfaWebApp's UML class diagram.

the complete UML model of our case study. Although our application does not have an extensive scope of functionality, we consider it a feasible system to show the development of MFAApps following the Build Time, Frameworkless, and Framework-based approaches (see Sections 2 and 3.2). Drawing a parallel between the model and the MFA aforementioned, the **Dashboard** MFA deals with the **InventoryMovement** class, the **Products** MFA deals with the **Product** class, and the **Inventory** MFA deals with the **Inventory** class.

Empirical Research Strategy. Figure 3 illustrates the implementation of the **MfaWebApp** application using the three development approaches: Build Time, Frameworkless, and Framework-based. Initially, we conducted technology-agnostic modeling of that application (i.e., **MfaWebApp**) so that MFAApps were identified (**Dashboard**, **Products**, **Inventory**). These applications follow the same organization (i.e., Frontend, Backend, and Database) as a MFAApp illustrated in Figure 1 (Area A).

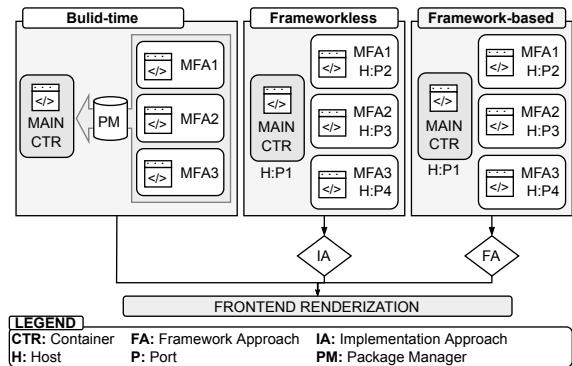


Figure 3: Organization of the MFAApp implementations.

Before starting the description of each approach, we referenced the MFAApps **Dashboard**, **Products**, **Inventory** as MFA1, MFA2, and

MFA3 in Figure 3, respectively. Moreover, MAIN (i.e., a container application) must be the first application to be developed for each approach, regardless of the development approach (i.e., Build-time, Frameworkless, Framework-based). This application is responsible for the initial rendering, acting as a communication bridge between the frontend and the elements for access to the application's other micro frontends. Next, we addressed a description of each approach.

In the ***Build-time approach***, the technology decisions do not represent a major impact on architectural development. In short, the development focuses on the process of dividing micro applications into different packages/libraries (MFA1, MFA2, and MFA3), which will be imported into the container application (MAIN) so that it can execute the application entry point. Moreover, It is worth noting that management of MFApps is carried out by the PACKAGE MANAGER element (e.g., Maven³). The selection of dependency management is at the discretion of each software engineer responsible for the application. Finally, it is worth highlighting that communication between the main application and the MFApps occurs by calling classes and methods exported from the micro applications. In this sense, it can be said that the application routing does not change, since the routes will direct to method calls or exported instances. Furthermore, data sharing between micro applications is done via parameter passing or global state sharing.

Similar to ***Build-time***, the ***Frameworkless approach*** also does not rely on technologies, since the development of MFApps is guided by the independence of the programming language or framework. In this sense, we can highlight the following implementation resources for composing MFApps: (i) *iFrames*, which can be included in any technology as an HTML element; (ii) *Web Components*, which represents a set of technologies based on component reuse decoupled from code; and (iii) *Module Federation*, despite being a library developed for Webpack with JavaScript, Module Federation loads a remote application in JavaScript by importing code, so it is also possible to implement it in any other language. We used Web Components and Module Federation in JavaScript micro applications to implement the **MfaWebApp** application. Therefore, our implementation in React had an adaptation in routing, using an interface for loading remotely hosted Web Components. To do so, the interface imports a Web Component and uses the browser's window object, passing the loaded component via a parameter. We use the passing of query parameters and the React's Context API to share data between micro frontends.

³<https://maven.apache.org>

Finally, the ***Framework-based approach*** requires the selection of frameworks for developing MFApps. According to our experiences and evidence collected in the gray literature Micro-frontend.dev (2023), we suggest five criteria for selecting frameworks in a software project, namely: learning curve, documentation quality, support quality, quality of the community related to the Framework/Library, and level of adherence concerning development approaches. Although such criteria are easy to interpret, the analyses are subjective to each individual. In this sense, we recommend adopting some metric that can guide decision-making. According to Micro-frontend.dev (2023), the decision matrix can be an alternative to assist any stakeholders in choosing a framework. Based on the aforementioned criteria, we made the following choices for the development **MfaWebApp** based on the ***Framework-based approach***. A JavaScript framework called Garfish (2021) to manage MFApps was selected. This framework enables employing the concepts of Module Federation were used to compile micro applications into a final application. The selected framework (Garfish) manages the decision regarding routing and data, therefore data-sharing routing implementations are abstracted, enabling greater focus on the development of micro applications.

5 DISCUSSION OF RESULTS

This section summarizes the main findings of our paper. As reported in Section 3, our investigation aimed to establish an important panorama on software development based on micro frontend, providing a more precise and complete definition of micro frontends (see Section 3.1), as well as guidelines (i.e., company and architecture) on the development of this type of software (see Section 3.2). Finally, Section 3.3, we described three scenarios that can help any stakeholder in decisions regarding the choice of this architectural style. Drawing a counterpoint between our paper and the study conducted by Nishizu and Kamina (2022), which presented an experience report for small organizations, our investigation suggests that the MFA can be applied to any type of software since the organization features and technological restrictions of the development team are met. Undoubtedly, it is simpler to manage problems/smaller teams; however, good practices and adequate infrastructure can boost team productivity, besides enabling continuous and valuable software delivery.

Concerning the case study reported in Section 4, the possibilities of technologies and approaches that can be used are notable since modern frontend de-

velopment relies heavily on frameworks and libraries. Furthermore, our case study also addressed operational complexity, since the entry point for adopting the MFA involves non-trivial decision-making regarding the use of approaches and implementation techniques. Despite the ease of implementation, the Build-time approach does not follow all the principles and advantages that MFA can provide. The Frameworkless approach can be seen as an attraction for mature teams in technical terms, since there are no mature and well-documented frameworks in the literature, the micro frontend management itself can make the architectural implementation more solid and robust with decisions to the approaches and technologies. However, the agility seen in modern software development encourages the use of frameworks for less complexity, especially when bringing new developers into teams, standardizing development processes and quickly delivering software to the user. Based on this scenario, it can be said that the Framework-based approach is still at an incipient stage with little content found in the developer community, besides the scarcity of documentation.

Drawing a trade-off of the MFA, different approaches can also present differences in results, mainly in operational complexity and development experience. Thus, it can be said that the MFA and the three approaches presented in this paper should not be considered as a “silver bullet” solution that can be used in all development scenarios, leaving the decision for the development team to best path towards the project to be developed. This evidence also reveals the complexity level of this architectural style concerning software development, especially considering scenarios where there is a predominance of inexperienced teams and a lack of development infrastructure since the involuntary application of anti-patterns in architectures that are not yet mature, can generate severe negative impacts on software projects.

6 CONCLUSIONS AND FUTURE WORK

This paper presented an overview of software development based on micro frontends, focusing on the development approaches Build-time, Frameworkless, and Framework-based. Based on the evidence of our study, it can be said that micro frontend-based development was inspired by the microservice architectural style, borrowing several consolidated concepts that can be applied to the development of micro applications such as decentralizing team governance, maintaining high availability, low coupling, and en-

couraging independent delivery of micro applications. Next, we report the main contributions of this paper: (i) a panorama for the development area by establishing concepts, motivations, architectural models, and principles that can be applied in the development of MFAs; (ii) a report for the researchers and practitioners by presenting an experience of this architectural style, which can be a reference important for the new implementations of software systems based on MFA. In this sense, it is worth highlighting the development approaches, company’s organizational structure, architectural model, and good engineering practices for companies used in our case study; and (iii) a case study for the Web development area by exploring the use of the micro frontend approaches to scalable Web application projects, permeating Web architecture concepts, and the use of new frameworks for the proposed architectures (see Figure 3).

As future work, we intend to conduct at least four activities: (i) conduction a more specific investigation about this research theme, considering newer technologies (e.g., frameworks), architectural styles, and software architectures; (ii) conduction of a study related to the decision-making on the architectural choice of project development using MFA; (iii) conduction of more case studies using different frameworks to evaluate the applicability and feasibility of the development of micro frontends based on different frameworks for the industry; and (iv) conduction of comparative benchmark between micro frontend and monolith approaches to evaluate the performance and impact of these architectural models when an application is developed. Therefore, based on the content presented in this paper, a positive research scenario can be established. Besides the effective contribution of our paper to the software engineering community, the industry can benefit from the analysis and experience reported as a trade-off for the MFA.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES).

REFERENCES

- Bian, Y., Ma, D., Zou, Q., and Yue, W. (2022). A multi-way access portal website construction scheme. In *The 5th International Conference on Artificial Intelligence and Big Data, ICAIBD 2022*, page 589 – 592. Institute of Electrical and Electronics Engineers Inc.

- Bühler, F., Barzen, J., Harzenetter, L., Leymann, F., and Wundrack, P. (2022). Combining the best of two worlds: Microservices and micro frontends as basis for a new plugin architecture. *Communications in Computer and Information Science*, 1603 CCIS:3 – 23.
- Frameworkless (2023). Frameworkless movement. on-line. <https://www.frameworklessmovement.org>, accessed on March 14, 2024.
- Garfish (2021). Garfish. on-line. <https://www.garfishjs.org>, accessed on March 14, 2024.
- Geers, M. (2020). *Micro frontends in action*. Manning Publications, New York, NY.
- Jackson, C. (2019). Micro frontends. on-line. <https://martinfowler.com/articles/micro-frontends.html>, accessed on March 14, 2024.
- Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report.
- Lewis, J. and Fowler, M. (2014). Microservices. on-line. <https://martinfowler.com/articles/microservices.html>, accessed on March 14, 2024.
- Mannisto, J., Tuovinen, A.-P., and Raatikainen, M. (2023). Experiences on a frameworkless micro-frontend architecture in a small organization. In *Proceedings - IEEE 20th International Conference on Software Architecture Companion, ICSA-C 2023*, page 61 – 67. Institute of Electrical and Electronics Engineers Inc.
- Mena, M., Corral, A., Iribarne, L., and Criado, J. (2019). A progressive web application based on microservices combining geospatial data and the internet of things. *IEEE Access*, 7:104577–104590.
- Micro-frontend.dev (2023). Micro-frontends and composable frontend architectures. on-line. <https://microfrontend.dev>, accessed on March 14, 2024.
- Mohammed, S., Fiaidhi, J., Sawyer, D., and Lamouchie, M. (2022). Developing a graphql soap conversational micro frontends for the problem oriented medical record (ql4pomr). In *ACM International Conference Proceeding Series*, page 52 – 60. Association for Computing Machinery.
- Moraes, F. R., Campos, G. N., de Almeida, N. R., and Affonso, F. J. (2023). Systematic mapping protocol – mapping study on. on-line. <https://drive.google.com/file/d/1CquhVgCo4c5uwBzWeBV2DvBIIM7-KZtn/view?usp=sharing>, accessed on March 14, 2024.
- Nishizu, Y. and Kamina, T. (2022). Implementing micro frontends using signal-based web components. *Journal of Information Processing*, 30:505 – 512.
- Noppadol, N. and Limpiyakorn, Y. (2021). Application of micro-frontends to legal search engine web development. *Lecture Notes in Electrical Engineering*, 782:165 – 173.
- Oliveira, D. S. M., Oliveira, F. C. M. B., Pernencar, C. A. C., de Moraes, B. S., Silva, J. W., Costa, A. R. B., Pereira, J. B. C., and Saboia, I. F. (2022). Licor: Beyond the design system. a proposal to empower teams to develop software in compliance with the principles of accessibility, usability, and privacy by design in the extreme contexts and challenging domains post-covid-19. *Communications in Computer and Information Science*, 1654 CCIS:139 – 147.
- Pavlenko, A., Askarbekuly, N., Megha, S., and Mazzara, M. (2020). Micro-frontends: Application of microservices to web front-ends. *Journal of Internet Services and Information Security*, 10(2):49 – 66.
- Peltonen, S., Mezzalira, L., and Taibi, D. (2021). Motivations, benefits, and issues for adopting micro-frontends: A multivocal literature review. *Information and Software Technology*, 136:106571.
- Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18.
- Pölöskei, I. and Bub, U. (2021). Enterprise-level migration to micro frontends in a multi-vendor environment. *Acta Polytechnica Hungarica*, 18(8):7 – 25.
- Qiankun (2019). Qiankun. on-line. <https://qiankun.umijs.org>, accessed on March 14, 2024.
- Shakil, M. and Zoitl, A. (2020). Towards a modular architecture for industrial hmis. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1267–1270.
- Single-spa (2016). Single-spa. on-line. <https://single-spa.js.org>, accessed on March 14, 2024.
- Soldani, J., Tamburri, D. A., and Van Den Heuvel, W.-J. (2018). The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232.
- Stefanovska, E. and Trajkovik, V. (2022). Evaluating micro frontend approaches for code reusability. *Communications in Computer and Information Science*, 1740 CCIS:93 – 106.
- Taibi, D. and Mezzalira, L. (2022). Micro-frontends: Principles, implementations, and pitfalls. *SIGSOFT Softw. Eng. Notes*, 47(4):25–29.
- Tilak, P. Y., Yadav, V., Dharmendra, S. D., and Bolloju, N. (2020). A platform for enhancing application developer productivity using microservices and micro-frontends. In *2020 IEEE-HYDCON*, pages 1–4.
- Wang, D., Yang, D., Zhou, H., Wang, Y., Hong, D., Dong, Q., and Song, S. (2020). A novel application of educational management information system based on micro frontends. *Procedia Computer Science*, 176:1567–1576.
- Yang, C., Liu, C., and Su, Z. (2019). Research and application of micro frontends. In *IOP Conference Series: Materials Science and Engineering*, volume 490. Institute of Physics Publishing.