Micro-Frontend Architectures for Modern User Interfaces in Enterprise Systems: A Systematic Mapping Study

Vítor Marco[a], Kleinner Farias[a,*], Carlos Carbonera[a]

[a]*PPGCA, University of Vale do Rio dos Sinos (Unisinos), Av. Unisinos, 950, Sao Leopoldo, RS, Brazil*

Abstract

Micro-frontend architecture has been increasingly adopted to support the development of modular and scalable user interfaces in enterprise systems. Various technologies, design patterns, and deployment strategies have emerged to meet this demand in academia and industry. However, a comprehensive classification of existing approaches to designing, implementing, and evaluating micro-frontends remains absent. As a result, a thorough understanding of the current state of the art, including the technological stack, architectural practices, and motivations for adoption, remains limited and inconclusive. This article, therefore, presents a classification of studies on the technologies, design patterns, and industrial practices used in micro-frontend architectures for modern user interfaces in enterprise systems. Following well-established guidelines, we conducted a systematic mapping study to address ten research questions. After applying a rigorous selection process to an initial pool of 5,434 candidate studies, we selected and analyzed 29 primary studies. The results indicate that (1) a significant portion of the studies focuses on JavaScript-based stacks, with ReactJS and Web Components being the most common technologies; (2) micro–frontend adoption is driven primarily by modularization, component isolation, and flexibility; (3) most studies do not specify evaluation methods or design patterns, indicating a gap in empirical evalaution and architectural standardization. This study identifies key challenges and offers directions for future research, including the need to investigate the effects of micro-frontend architecture on user experience, team collaboration, and cognitive load in large-scale systems.

Keywords: Software architecture, front-end, micro–frontend, Enterprise Resource Planning
PACS: 0000, 1111
2000 MSC: 0000, 1111

1. Introduction

In recent years, the development of large-scale, user-facing enterprise systems has driven the adoption of modular and scalable front-end solutions [1, 2]. Among the architectural strategies that support this paradigm, micro-frontend architectures have gained significant traction [3, 4, 5]. By decomposing complex user interfaces into independently developed and deployed components, this approach enables greater agility, maintainability, and scalability [6]. Both academia and industry have responded by proposing various supporting technologies, design patterns, and deployment practices. As the architectural complexity of enterprise software continues to grow, micro-frontend solutions offer a promising avenue for managing interface evolution while maintaining cohesion across distributed development teams.

Despite the increasing popularity of micro-frontend architectures, the field lacks a consolidated understanding of how these systems are designed, implemented, and evaluated. Existing studies focus on isolated technical implementations or narrow use cases, resulting in fragmented knowledge [3, 4, 7, 5]. There is no unified classification of the architectural patterns, technology stacks, or evaluation strategies in use. Furthermore, limited empirical validation and the absence of standardized methodologies hamper efforts to assess the impact of micro-frontend architectures on user experience and team productivity. Consequently, key questions remain unanswered regarding best practices, effective technology integration, and the strategic motivations driving adoption in enterprise contexts.

To address this gap, we conducted a systematic mapping study following established methodological guidelines [8, 9, 10, 11] already adopted widely in recent literature reviews [12, 13, 14, 15, 16]. Our goal was to identify and classify the technologies, design patterns, and industrial practices associated with micro-frontend architectures for modern user interfaces. The study was guided by ten research questions. We began with an initial corpus of 5,434 candidate studies retrieved from ten reputable digital libraries. After applying exclusion and inclusion criteria through a multi-phase filtering process, we selected 29 primary studies for in-depth analysis. Each study was coded and categorized based on its treatment of front-end architecture, technology usage, deployment strategies, and reported outcomes.

Our results reveal three main findings. First, most studies employ JavaScript-based stacks, particularly ReactJS and Web

*Corresponding author
Email address: kleinnerfarias@unisinos.br (Kleinner Farias)

Components, which dominate the current landscape of micro-frontend implementations. Second, the primary motivations for adopting this architectural style include modularization, component isolation, and the flexibility to support independent development workflows. Third, there is a pervasive lack of detail regarding evaluation strategies and design pattern specifications. This absence indicates a critical gap in empirical assessment and architectural standardization, which may hinder replication and scalability in industry settings. These results underscore the need for further investigation into usability, cognitive load, and collaborative practices in micro-frontend systems.

This study contributes to the software engineering literature in several ways. It systematically classifies current technologies and patterns, outlines the prevalent motivations for adoption, and highlights gaps in empirical validation. Unlike prior work, which often focuses narrowly on individual tools or case studies, this research provides a comprehensive synthesis of the field. Practitioners gain insights into emerging architectural trends, while researchers have a foundation for developing new methods, frameworks, and empirical studies. Furthermore, this work has implications for human-computer interaction, particularly in designing interfaces that minimize cognitive effort and support user adaptability in complex software environments.

The rest of this article is organized as follows. Section 2 reviews related work. Section 3 outlines the methodology used in our study. Section 4 describes the selection process. Section 5 presents the results of the study. Section 6 presents implications and challenges for future research. Section 7 presents some concluding remarks.

2. Related work

This section presents an analysis of related studies addressing micro-frontend architecture and literature reviews. First, we examine carefully four related studies (Section 2.1). After, we contrast these studies with the proposed work to highlight some commonalities and differences (Section 2.2).

2.1. Analysis of related work

Taibi et al. (2022) [3] conducted a systematic mapping study to evaluate different approaches to implementing micro-frontends, including web components and single-page applications (SPAs). Their research outlines the pros and cons of each, with a particular focus on security, browser compatibility, and management complexity. The study does not sufficiently address how these approaches impact user satisfaction, particularly regarding the fluidity of interaction and adaptability to frequent interface changes–key aspects for positive user experiences in highly modular systems.

Peltonen et al. (2021) [17] details the motivations, benefits, and technical challenges of adopting the micro-frontend design pattern. The authors affirm that a flexible and scalable architecture is essential to meet the evolving needs of modern systems. Key benefits include team autonomy, increased modularity, and component reuse. So, these technical advantages streamline development processes. This study does not fully address how such architectural choices impact end users, particularly regarding usability and cognitive load in complex user interfaces.

Pavlenko et al. (2020) [7] investigated micro-frontend approaches in the context of microservices, focusing on architectural solutions to complexity and scalability challenges. The study highlights improvements in cross-team collaboration and modular design that benefit the development process. However, while successful implementations are presented, there is limited discussion of how these modular architectures improve or hinder user experience, such as navigation and system responsiveness, from the user's perspective.

Prajwal et al. (2019) [4] conducted a review of micro–frontend implementations in web applications. Their analysis centered on scalability, decoupling, component reuse, and team independence. They identified challenges such as increased management complexity, security concerns, and browser compatibility. Although these are critical technical aspects, the study lacks an in-depth examination of how micro-frontend modularity affects user interaction, specifically regarding interface consistency and cognitive strain when switching between independent components.

Harms et al. (2017) [5] proposed guidelines for selecting frontend design patterns in microservices-based applications. They explored management complexity and team cohesion criteria and provided examples of frontend architectural patterns such as component-based and layered architectures. This study provides valuable insights into architectural decision-making. However, it does not evaluate how these decisions affect user engagement, particularly in intuitive interface flow and the cognitive load on users navigating between components developed by different teams.

2.2. Comparative analysis of related work

The comparative analysis was performed based on comparison criteria, a method already demonstrated in previous studies to be effective in pinpointing commonalities and differences between related studies [1, 18]. Four comparison criteria were used to analyze the related works: the main purpose of each study, the research method adopted, the architecture investigated, and the technologies explored. Table 1 summarizes the principal features of each work.

Some studies, such as those by Taibi et al.[4] and Pavlenko et al. [7], also adopted systematic mapping as their primary research method, aligning them methodologically with our work. These studies provide general overviews of micro-frontend architectures, frameworks, and implementation styles, offering foundational knowledge for the field.

Despite these contributions, significant gaps remain. Most prior studies limit their scope to high-level descriptions or technology stacks without systematically examining design patterns, industrial practices, or evaluation methods. For instance, Harms et al.[17] focus on implementation guidelines and organizational motivations but do not integrate empirical

Table 1: Comparative analysis of related works

Study	Purpose	Research method	Architecture	Technologies
Our study	Analysis of benefits of micro-frontends adoption, applied technologies and industry context	Systematic mapping study	Micro–frontend	ReactJs, Angular, Vue, single-spa, NextJS, Ionic and Apache Cordova
Taibi et al. (2022) [3].	Overview of micro–frontends, how to implement and how to select the best approach in development	Systematic mapping study	Micro-frontend in Application Shell, iFrames, Edge Side Includes, Web Components and Server-Side	Angular, SAP Luigi, Stencil, Tailor, Open Components, Ara and Piro
Prajwal et al. (2021) [4].	Overview of micro–frontend architecture	systematic mapping study	Micro–frontend	React, Angular and Vue
Peltonen et al. (2021) [17].	Understanding the motivations and benefits for companies when adopting micro–frontend	Survey	micro–frontend	–
Pavlenko et al. (2020) [7].	Micro–frontends in single-page applications	Systematic mapping study	Micro–frontend in single-page applications	React, Angular, Web Components
Harms et al. (2017) [5].	Guide to selecting micro–frontend architectures	Mapping study	Micro–frontend	–

findings with architectural trends or technological choices. Furthermore, some works [3, 4, 5, 17, 7] overlook how technologies are applied in industrial contexts or how they relate to modularity, component isolation, or team interaction. Unlike these, our study systematically maps the landscape by examining ten research questions, capturing academic and industry perspectives on micro-frontend adoption in enterprise systems.

Our work addresses these omissions by (i) classifying applied technologies, (ii) identifying design patterns and industrial domains, and (iii) analyzing motivations and evaluation practices. These dimensions were previously unexplored in an integrated fashion. Moreover, our study uniquely investigates how micro-frontends are perceived regarding benefits and challenges, offering actionable insights for researchers and practitioners. By aligning our analysis with current HCI concerns, such as cognitive load and user interaction, we seek to extend the scope beyond technical implementation, thus contributing to a more holistic understanding of micro-frontend architectures.

3. Methodology

This section outlines our research protocol. Section 3.1 defines the objective and research questions addressed in our study. Section 3.2 shows our search strategy used to identify a sample of representative works. Section 3.3 specifies the inclusion and exclusion criteria for filtering potentially relevant studies. Our research protocol is based on well-established guidelines widely adopted in the literature [10, 11, 19] and follows methodologies of previously published studies to ensure scientific validity and reliability [12, 13, 14, 15, 16, 20].

3.1. Objective and research questions

This study pursues two primary objectives. First, it aims to classify and synthesize existing research on micro–frontend architecture, focusing on the technologies, frameworks, architectural types, and design patterns employed in modern user interface development (GQ, FQ1–FQ3). Second, it seeks to examine how micro–frontends are evaluated and perceived by end-users, particularly in terms of usability, user experience, and interaction quality, while also identifying motivations for adoption, system domains, and architectural characteristics (FQ4–FQ7) (Section 5). In addition, our study explores publication trends and the evolution of interest in the field (SQ1–SQ2), offering a structured overview of current practices and outlining critical research challenges at the intersection of software engineering and human-computer interaction (Section 6).

To explore the different facets of our objectives, we have developed ten research questions (RQs). These RQs are grouped into three distinct categories: *General Question* (GQ), *Focused Questions* (FQ), and *Statistical Questions* (SQ). Table 2 summarises the research questions investigated.

General Question (QG). The primary objective of this research is to comprehensively explore the contemporary technologies and methodologies applied in the context of front-end development. This inquiry aims to provide a detailed overview of the current landscape of user interface development, encompassing critical technological advancements, tools, and development guidelines that shape user experiences. By understanding these aspects, we can better assess their usability and user satisfaction implications in modern applications.

Focused Questions (FQ). To guide our investigation, we have developed seven focused questions that explore key aspects of front-end application development. These questions analyze essential criteria such as architectural types, platforms, design patterns, and micro-frontend evaluations while examining the benefits of selected technologies. Our main objective is to gain valuable insight into best practices and challenges in front-end development. Providing developers

and the academic community with important knowledge and strategies for improving user interaction and experience in real-world applications.

Statistical questions (SQ). To evaluate the quantitative aspects of our study, we have defined two statistical questions to investigate the geographical distribution of studies in this field (SQ1) and to evaluate the trends of annual publication (SQ2). By analyzing these patterns, we gain insight into the geographical reach of front-end development research and the temporal evolution of related studies. This data-driven perspective can provide a powerful understanding of the growth and relevance of this field in the context of human-computer interaction.

Table 2: Research Questions

Type	Question
GQ	What technologies are currently employed to develop modern user interfaces using micro-front-end architecture?
FQ1	What architecture types are used in front-end development?
FQ2	What platforms are most commonly used in front-end development?
FQ3	What design patterns are commonly employed in the development and deployment of micro-frontends?
FQ4	How are the micro-frontends evaluated?
FQ5	How are the benefits of micro-frontends perceived by users?
FQ6	What are the motivations for selecting micro-frontends?
FQ7	What types of software systems are being developed in the industry using micro-frontend architecture, and what are their key characteristics?
SQ1	Where have the studies been published?
SQ2	How did the studies evolve in recent years??

Legend: General Questions (GQ), Focused Questions (FQ), and Statistical Questions (SQ)

3.2. Search strategy

Search string. To define the search string, terms related to the user interface, software architecture, and modern applications were used as keywords: front-end, architecture, and enterprise applications. Table 3 presents these main terms and their most relevant synonyms. The search string was constructed by combining these terms and synonyms, using logical Boolean operators such as disjunction and conjunction to ensure a comprehensive search string. The resulting search string is: ((“front end” OR “user interface” OR “graphical user interface” OR UI) AND (“software design” OR pattern OR “software architecture”) AND (“corporate systems” OR “enterprise applications” OR “enterprise resource planning” OR “enterprise asset management” OR “human capital management”)).

List of search engines. Table 4 details the search string applied to retrieve eligible articles in the most important digital engines very well known by the academy and the industry: ACM Digital Library, IEEE Xplore Digital, Science Direct, Scopus, and Google Scholar. These engines were chosen for two reasons. Initially, these platforms provide comprehensive access to high-quality, peer-reviewed research in software engineering, human-computer interaction, and emerging technologies. This ensures that the studies are credible and rigorous. The databases such as Scopus and ScienceDirect offer extensive interdisciplinary coverage, allowing us to identify relevant studies that intersect with other fields such as usability and cognitive science. In addition, Google Scholar serves as a broader search tool, aggregating content from various academic sources, which helps to capture grey literature and emerging trends that are not yet fully indexed by specialized databases. This combination of sources ensures a well-rounded and comprehensive systematic mapping study for our study. Finally, these search engines were selected because they have been evaluated and validated in published literature mapping studies as reliable and comprehensive sources for finding relevant research [12, 13, 14, 15, 16, 20, 21].

Table 3: Search string terms

Term	Expression
Front-End	(“micro front-end” OR “micro frontend” OR “microfrontend” OR interface OR “graphical user interface” OR UI)
Architecture	(“software design” OR pattern OR “software architecture”)
Enterprise Applications	(“corporate systems” OR “enterprise platforms” OR “enterprise resource planning” OR “enterprise asset management” OR “human capital management”)

Table 4: Repositories used

Repositories	Address
ACM Digital Library	`https://dl.acm.org/`
IEEE Xplore Digital	`https://ieeexplore.ieee.org/`
Science Direct	`https://www.sciencedirect.com/`
Scopus	`https://www.scopus.com/`
Google Scholar	`https://scholar.google.com/`

Some customizations were necessary to adapt the application of the Search Strings to the specific requirements and functionalities of the previously reported search engines. Table 5 shows the customizations used in our study.

3.3. Inclusion and exclusion criteria

This section defines the inclusion and exclusion criteria to retrieve eligible studies from the electronic databases. The inclusion and exclusion criteria helped to refine the pool of potentially relevant studies [11]. Inclusion criteria (IC) ensure that only papers that respond to pre-defined conditions - such as relevance to the research topic, methodological rigour, or publication aspects are assessed.

Exclusion criteria (EC) help eliminate studies that do not align with the review’s scope or quality standards, such as outdated research or studies lacking sufficient empirical

Table 5: Customization for each repository

Repository	Customization
ACM Digital Library	Advanced filter "Custom range"–"From Jan 2000 and "to August 2024" and filtering by "Content Type – Research Article" and "Publications – Media Formats – PDF."
IEEE Xplore Digital	Advanced Filter "Publication Year"–"Specify Year Range 2000–2024"
Science Direct	Advanced Filter "year(s)"–"2000–2024"
Scopus	Regular Filter
Google Scholar	Regular Filter

evidence. Together, these criteria provide a structured approach to selecting studies, ensuring consistency, objectivity, and the reliability of the findings derived from the systematic mapping study. Such criteria have been validated in previous systematic mapping studies, and their usefulness and reliability are already well established [12, 13, 14, 15, 16, 20, 21].

We defined four IC to add studies to our sample. The IC inserted studies that:

- **IC1:** Articles, books, or book chapters that focus on exploring micro–frontend architecture, software engineering, and user experience topics.
- **IC2:** Publications written in English.
- **IC3:** Works published in journals, books, conferences, and workshops.
- **IC4:** Publications from January 2000 to August 2024.

Additionally, we applied six EC to refine the selection process by eliminating studies that:

- **EC1:** The title, abstract, or any part of the content is lexically associated with the search string terms but does not relate to the research topic.
- **EC2:** The study was not published in English.
- **EC3:** The authors excluded definitions of patents, gray literature (non-peer-reviewed material), secondary studies, and descriptions of presentations.
- **EC4:** The abstract did not address the context of micro–frontend architectures, software engineering, and user experience.
- **EC5:** Duplicate studies.
- **EC6:** Studies that do not address issues arising from the defined research questions with the interplay between micro–frontend architectures, software engineering, and user experience questions.

4. Study Filtering

This section explains the study filtering process, providing a comprehensive overview of the method used to select candidate studies. Seven steps were defined, and inclusion (IC) and exclusion (EC) criteria were applied. Figure 1 illustrates the results of each step. The steps involved in the filtering of studies are detailed:

- **Step 1: Initial search.** The objective of this step is to collect the initial search results from the search engines using the search string defined in Section 3.2, without applying any filters. A total of 5,434 candidate studies were retrieved from five digital libraries.
- **Step 2: Impurities removal (EC1 and EC2).** This step removes irrelevant results from the initial search. We applied exclusion criteria EC1 and EC2 to eliminate calls for journal special issues, conference studiess, talk descriptions, research reports, patent specifications, and non-peer-reviewed studies. As a result, we removed 2,787 studies (51.29%), leaving 2,647 candidate studies (48.71%) for the next step.
- **Step 3: Filter by title (EC3).** This step filters studies based on their title. Using EC3, we removed studies unrelated to the research questions of this SMS. As a result, 2,536 studies (95.2%) were excluded, leaving 111 potentially relevant studies for the next step.
- **Step 4: Filter by abstract (EC4).** Using EC4, we filtered articles by carefully reading their abstracts to remove those not closely aligned with our research questions. In total, 55 articles (49.54%) were excluded, leaving 56 for the next phase.
- **Step 5: Combination.** This step consolidates the results from the previous stage into a single directory in preparation for the next phase. No exclusion criteria were applied during this process.
- **Step 6: Duplicated removal (EC5).** By applying EC5, we identified and removed duplicate studies during this step, resulting in 52 studies.
- **Step 7: Filtering by full text (EC6).** This step aimed to filter studies based on their full texts. We applied EC6 to 52 potentially relevant articles, removing 25 unrelated to our SMS. This resulted in a final set of 29 studies for the next phase.
- **Step 8: Selection of a representative studies.** This step serves as an additional reading and discussion phase to ensure that the selection process is completed without retaining multiple versions (different studies) of the same study. When reviewing the 29 remaining studies from the previous step, we increased the comprehensiveness of the selection process by combining the list of studies obtained from the search string with a snowballing strategy. However, this effort did not result in additional

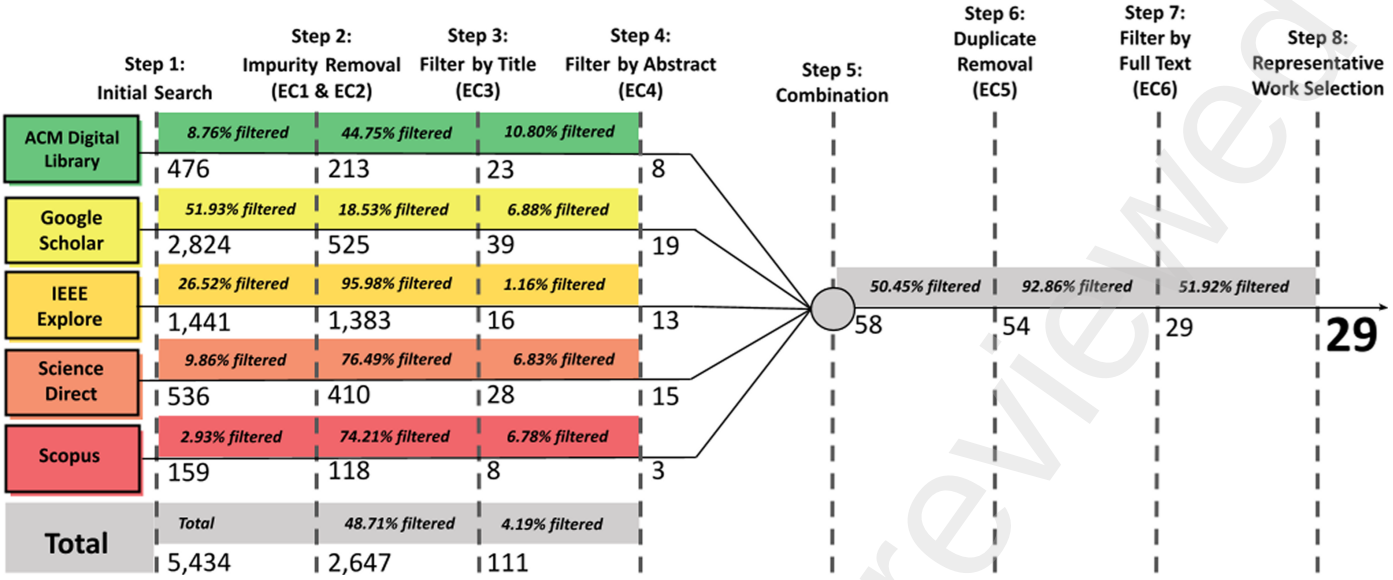

Figure 1: Steps of the study filtering process.

studies, resulting in a final set of 29 primary studies, detailed in Table 6.

5. Results

This section presents the systematic mapping results according to the formulated research questions.

5.1. GQ: What technologies are employed to develop modern user interfaces using micro-frontend architecture?

Table 7 introduces the technologies adopted by the primary studies. These results reveal two interesting findings. First, JavaScript is the most widely used technology in micro-frontend development, appearing in 22 of the 29 primary studies (75.86%). This result aligns with JavaScript's central role in building dynamic user interfaces. While our results combine technologies by language or tooling level, many of these JavaScript references relate to modern frameworks (e.g., Angular and VueJS) or libraries (e.g., ReactJS), each supporting a component-based approach crucial for micro-frontends. Both promote modularization and reuse, reinforcing JavaScript's dominance in modular front-end architectures.

The second most frequent technology is Webpack, a module bundler adopted in 13 studies (44.83%). Webpack's ability to split code and load modules asynchronously makes it particularly well-suited for micro-frontend applications, which benefit from independent deployment and runtime composition. An ever-present co-occurrence of Webpack and JavaScript suggests that developers favor toolchains that offer seamless integration and performance optimization. HTML (41.38%) and CSS (37.93%) were also often used, reflecting their indispensable roles in content structure and styling. These results reinforce the relevance of the classic Web stack in the context of micro-frontend solutions. Technologies such as Redux (6.90%), TypeScript (6.90%), LitElement (6.90%), and Styled-Components (3.45%) were far less prevalent. Their limited presence raises questions about the extent to which advanced state management, type safety, and component styling play a primary role in micro-frontend implementations.

Finally, technologies used in micro-frontend development exhibit rapid and short innovation cycles, driven by the constant demand for technical optimization and improved user experience. Front-end applications must meet performance and search engine indexing requirements, which impose strict constraints on tooling and architecture. Few studies evaluated server-side rendering (SSR) frameworks like Next.js and Nuxt.js or bundling optimization tools like PurgeCSS [50]. This gap suggests limited attention to performance-oriented technologies in current micro-frontend literature despite their relevance for scalability and discoverability.

Discovering patterns of technology combinations. We analyze the co-occurrence of technologies across studies (e.g., JavaScript + Webpack + React) to reveal de facto standard stacks or emerging combinations in micro-frontend development. For this, Figure 2 shows the top 15 most frequent technology combinations used in primary studies on micro-frontend architectures. The two most common pairs are CSS + HTML and JavaScript + Webpack, each appearing in 11 out of 29 studies (37.93%). These findings confirm the foundational role of these technologies in modern user interface development. HTML and CSS form the structural and stylistic backbone of front-end design, while JavaScript and Webpack enable interactive behavior and modular bundling. The combination of HTML + JavaScript (8 studies) and CSS + JavaScript (7 studies) further reinforces the dominant presence of these core web technologies.

The data also reveals meaningful insights into emerging combinations. For example, HTML + Webpack and CSS + Webpack appear in 4 studies, suggesting that bundling

Table 6: List of the primary studies.

ID	Author	Year	Type	Repository	Venue
A1	Männistö, Jouni and Tuovinen, Antti-Pekka and Raatikainen, Mikko [6]	2023	Conference	IEEE	International Conference on Software Architecture Companion
A2	Tilak, P Yedhu and Yadav, Vaibhav and Dharmendra, Shah Dhruv and Bolloju, Narasimha [22]	2020	Conference	IEEE	HYDerabad CONference
A3	Abdullah, Hanin M. and Zeki, Ahmed M. [23]	2014	Conference	IEEE	International Conference on Advanced Computer Science Applications and Technologies
A4	Mena, Manel and Corral, Antonio and Iribarne, Luis and Criado, Javier [24]	2019	Journal	IEEE	IEEE Access
A5	Micko, Kristian and Beca, Michal and Papcun, Peter and Zolotova, Iveta [25]	2023	Conference	IEEE	SAMI
A6	Antunes, Hiuram and da Fonseca, Inácio de Sousa Adelino [26]	2021	Conference	IEEE	Iberian Conference on Information Systems and Technologies
A7	Rafael, Christopher and Adikusumo, Philipus Wijaya and Sanjaya, Ngurah Agus Bangkit and Anggreainy, Maria Susan [27]	2021	Conference	IEEE	International Conference on New Media Studies (CONMEDIA)
A8	Andrade, Cassiano and Alves, Paulo and Fernandes, José Eduardo and Coutinho, Flávio [28]	2020	Conference	IEEE	Iberian Conference on Information Systems and Technologies (CISTI)
A9	Dwivedi, Palak and Kshamta and Joshi, Abhishek [29]	2022	Journal	IEEE	International Conference on Cyber Resilience (ICCR)
A10	Olaf Zimmermann and Christoph Miksovic and Jochen M. Küster [30]	2012	Journal	Science Direct	Journal of Systems and Software
A11	Eike Schäffer and Andreas Mayr and Jonathan Fuchs and Martin Sjarov and Johannes Vorndran and Jörg Franke [31]	2019	Journal	Science Direct	Procedia CIRP
A12	Shen Shen and Cheng-Zhi Qin and Liang-Jun Zhu and A-Xing Zhu [32]	2023	Journal	Science Direct	Journal of Environmental Management
A13	Harry Sneed and Chris Verhoef [33]	2019	Journal	Science Direct	Journal of Systems and Software
A14	Xing-Yu Su et al. [34]	2014	Journal	Science Direct	Procedia Computer Science
A15	Américo Rio and Fernando Brito e Abreu [35]	2023	Journal	Science Direct	Journal of Systems and Software
A16	Kroiß, Manuel [36]	2021	Journal	Google Scholar	Institut für Information Systems Engineering
A17	Adolphs, Christoph and Schubert, Petra [37]	2008	Conference	Google Scholar	BLED 2008 Proceedings
A18	Ziani, Djamal [38]	2014	Journal	Google Scholar	International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)
A19	Font Escribano, Jaume and Reynoso Vasquez, Vilma Karina [39]	2017	Journal	Google Scholar	MASTEAM
A20	Linkola, Lasse [40]	2021	Journal	Google Scholar	Master's Programme in Information Technology
A21	Koppala, Jarno [41]	2018	Journal	Google Scholar	Master's Degree Programme in Information Technology
A22	Gunawan, J and Kosala, RR [42]	2020	Conference	Google Scholar	IOP Conference Series: Earth and Environmental Science
A23	da Silva, Ricardo Alexandre Pinto [43]	2021	Journal	Google Scholar	ProQuest Dissertations
Theses					
A24	BP, I Wayan Krishna Dharma and Anggraini, Dina [44]	2022	Journal	Google Scholar	International Research Journal of Advanced Engineering and Science
A25	Hiltunen, Mira [45]	2018	Journal	Google Scholar	Information and Communication Technology Degree Programme
A26	Simões, Bruno et al. [46]	2023	Conference	ACM	International Conference on 3D Web Technology
A27	Hidayat, Dwiky Chandra and Atmaja, I Ketut Jaya and Sarasvananda, Ida Bagus Gde [47]	2024	Journal	Google Scholar	Jurnal Galaksi
A28	Wang, Daojiang et al. [48]	2020	Journal	Google Scholar	Procedia Computer Science
A29	Antunes, Fabio et al. [49]	2024	Journal	Google Scholar	International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)

tools are tightly integrated with presentation layers in current development. Frameworks and libraries like Bootstrap + Webpack and Bootstrap + JavaScript, both with three occurrences, highlight a recurring stack for responsive, component-based UI construction. The combination Redux + Webpack and JavaScript + Redux, appearing in two studies each, indicates early but growing interest in state management tools paired with modular builds.

These results can offer practical guidance for researchers and developers. Practitioners can benchmark their toolchains against the most common stacks. At the same time, researchers can identify underexplored combinations, such as Typescript + Webpack or Tailwind CSS with LitElement, as potential areas for further study. Finally, the frequent co-occurrence of JavaScript, HTML, CSS, and Webpack in micro-frontend studies confirms their role as a standardized foundation for modular front-end development. Less frequent combinations, such as TypeScript with Webpack or Tailwind CSS with LitElement, suggest opportunities for further empirical exploration and architectural improvement.

Exploring pairwise co-occurrence intensity. Figure 3 shows a heatmap, highlighting the pairwise co-occurrence intensity (i.e., strong and weak correlations) between all technology pairs. The pairwise co-occurrence intensity analysis provides a deeper understanding of how technologies are commonly combined in modern user interfaces using micro-frontend architecture. Strong associations, such as JavaScript + Webpack, JavaScript + HTML, and HTML + CSS, indicate pivotal stacks widely adopted for their compatibility and ability to support modular and maintainable front-end designs. We believe these frequent pairings confirm the stability and maturity of core web technologies in the micro-frontend context. Meanwhile, technologies such as Tailwind CSS, Redux, and SASS, which show sparse co-occurrence, indicate a niche presence or emerging role within the architecture landscape. These findings reinforce our results (displayed in Figure 2).

From a scientific contribution perspective, this co-occurrence intensity analysis advances the research community's ability to classify and pinpoint technology usage in micro-frontend architectures. It contributes to developing a knowledge base that distinguishes tightly integrated technology stacks from underexplored combinations, enabling a structured classification scheme. For practitioners, the visualization offers a practical reference for choosing compatible tools and aligning architectural choices with system requirements and user experience goals.

This finding can support data-driven decision-making and promote the identification of reusable development patterns when software developers and architects design and develop enterprise systems. Moreover, the ability to pinpoint rare combinations helps researchers define new research directions around unexplored technology synergies, particularly those with potential benefits for superior performance, cognitive load reduction, or cross-team integration.

5.2. FQ1: What are the commonly used architecture types in front-end development?

Table 8 presents the commonly used architecture types in the primary studies. The micro-frontend architecture is the most frequently employed approach in modern front-end development, appearing in 41.38% (12/29) of the analyzed works. This result suggests a clear preference for modular and distributed architectures, likely driven by the need for scalability and independent team workflows. Adopting micro-frontends aligns with industry practices prioritizing

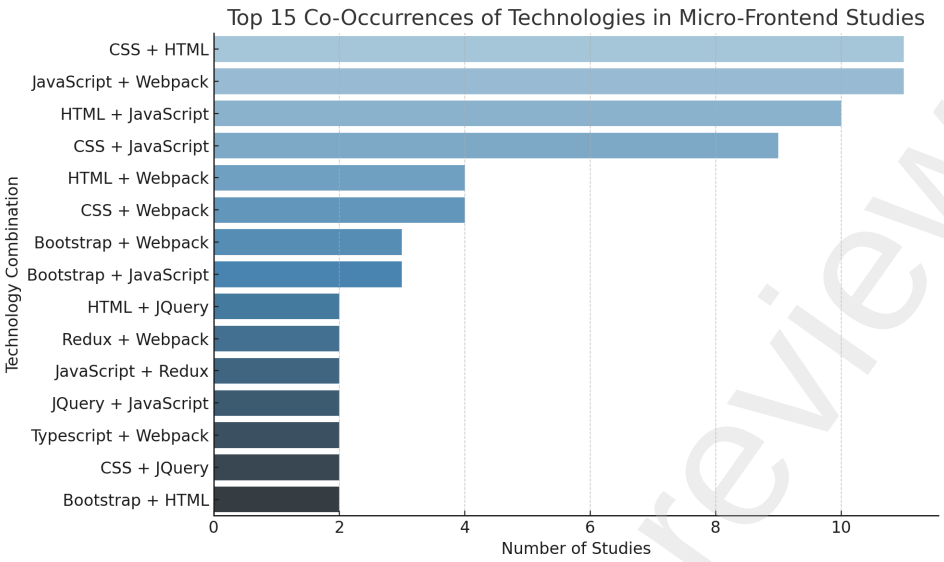

Figure 2: The top 15 most frequent technology combinations.

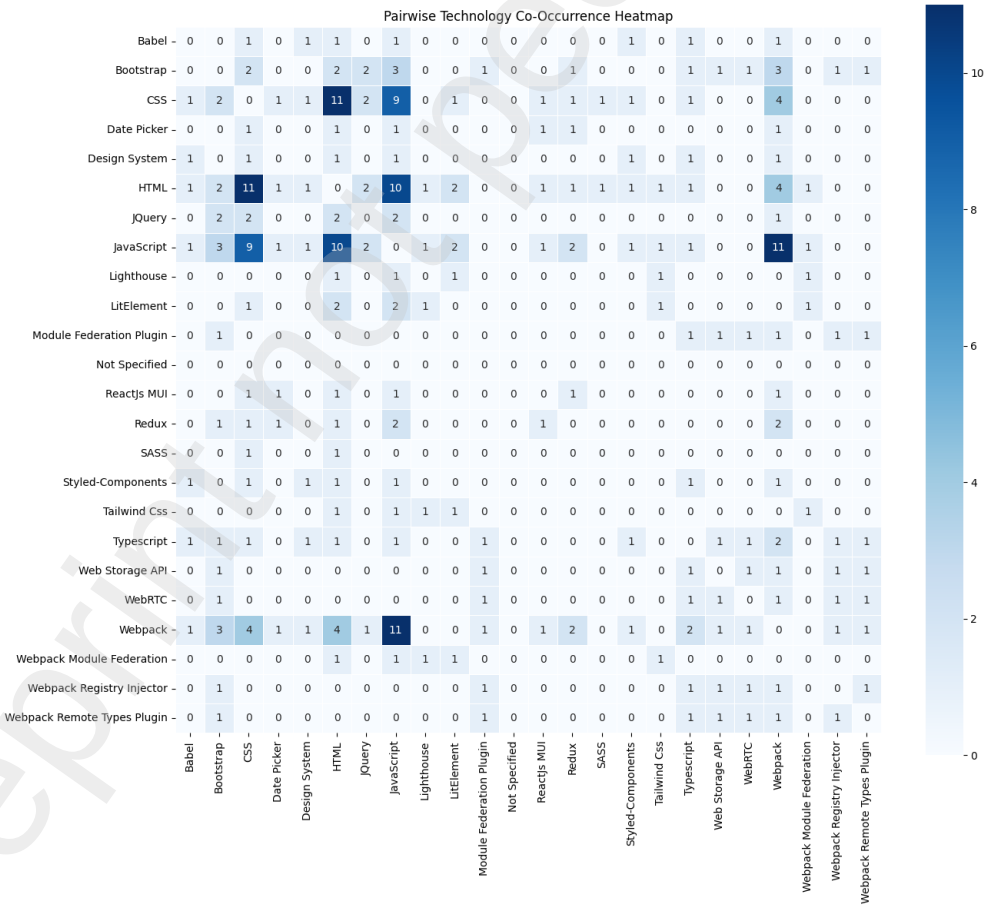

Figure 3: Pairwise co-occurrence intensity between all technology pairs.

Table 7: Current technologies for developing front-end applications (GQ).

Results	Quantity	Percentile	Studies
Javascript	22/29	75.86%	[A1],[A2],[A3],[A4],[A5], [A6],[A7],[A8],[A9],[A12], [A14],[A20],[A21],[A23], [A24],[A25],[A27],[A28], [A29]
Webpack	13/29	44.83%	[A2],[A4],[A6],[A7],[A8], [A9],[A16],[A20],[A21], [A23],[A24],[A25],[A26]
HTML	12/29	41.38%	[A1],[A3],[A5],[A7],[A9], [A12],[A14],[A15],[A19], [A20],[A24],[A27]
CSS	11/29	37.93%	[A1],[A3],[A5],[A7],[A9], [A12],[A14],[A15],[A19], [A20],[A24]
Bootstrap	3/29	10.34%	[A5],[A7],[A21],[A26]
Redux	2/29	6.90%	[A9],[A21]
JQuery	2/29	6.90%	[A5],[A7]
LitElement	2/29	6.90%	[A1],[A27]
Date Picker	1/29	3.45%	[A9]
ReactJs MUI	1/29	3.45%	[A9]
SASS	1/29	3.45%	[A19]
Styled-Components	1/29	3.45%	[A20]
Typescript	2/29	6.90%	[A20],[A26]
Babel	1/29	3.45%	[A20]
Design System	1/29	3.45%	[A20]
Tailwind Css	1/29	3.45%	[A27]
Webpack Module Federation	1/29	3.45%	[A27]
Lighthouse	1/29	3.45%	[A27]
WebRTC	1/29	3.45%	[A26]
Web storage API	1/29	3.45%	[A26]
Webpack registry injector	1/29	3.45%	[A26]
Module federationPlugin	1/29	3.45%	[A26]
Webpack remote types plugin	1/29	3.45%	[A26]
Not specified	5/29	17.24%	[A10],[A13],[A17],[A18], [A22]

maintainability, incremental updates, and integration with heterogeneous technology stacks. In contrast, the Model-View-Controller (MVC) architecture was found in 24.14% (7/29) of the studies, suggesting continued relevance in structured, component-based applications, particularly where separation of concerns remains central to system design.

Single-page application (SPA) architectures accounted for 17.24% (5/29) of the studies, reinforcing their popularity for delivering seamless user experiences through dynamic content loading and reduced server-side dependencies. However, it is noteworthy that 17.24% of the studies did not specify an architecture, which may reflect the use of traditional monolithic patterns or a lack of formal architectural documentation. This ambiguity can threaten long-term system maintainability, especially without experienced developers or structured team workflows. Without deliberate architectural choices, such as micro-frontends, projects may suffer from early degradation of design quality and technical debt accumulation.

Table 8: Architectures for developing front-end applications (FQ1).

Results	Quantity	Percentile	Studies
Micro-frontend	12/29	41.38%	[A1],[A2],[A4], [A11],[A16],[A20], [A23],[A24],[A26], [A27],[A28],[A29]
MVC	7/29	24.14%	[A7],[A14],[A15], [A25],[A18],[A19], [A22]
SPA	5/29	17.24%	[A8],[A5],[A6], [A9],[A21]
Not specified	5/29	17.24%	[A3],[A10],[A12], [A13],[A17]

5.3. FQ2: What frameworks are employed for developing micro-frontends?

Table 9 presents the results concerning frameworks or libraries employed for developing micro-frontends. Our results show a clear preference for ReactJS, which appears in 41.38% of the studies. React's component-based architecture supports reusable and responsive UI elements, making it well-suited for micro-frontend design. The same proportion of studies did not specify any platform, indicating a documentation gap or reliance on default configurations. This absence of specification hinders reproducibility and suggests that some projects may use traditional monolithic approaches or proprietary frameworks not disclosed in academic reporting.

Angular ranks second, cited in 24.14% of the studies. Its built-in dependency injection and modular architecture align with micro-frontend principles, enabling maintainable and scalable codebases. Although present in fewer studies (10.34%), Vue represents a lightweight alternative that balances flexibility and simplicity. The relatively low frequency of Single-SPA (6.90%), a framework designed specifically for micro-frontend orchestration, suggests that integration across libraries remains underexplored in academic literature despite its potential to reduce inter-framework complexity.

The data suggest combining multiple frameworks and libraries improves architectural efficiency in large-scale systems. Modular solutions support maintainability and scalability and influence key HCI metrics such as user satisfaction, accessibility, and performance. The technical debt incurred from relying on a single technology (particularly one with a limited talent pool) can result in architectural inconsistencies and degraded UX over time. These findings highlight the need for further empirical studies measuring the direct impact of platform choice on usability and system performance in micro-frontend-based applications.

Table 9: List of platform used (FQ2).

Results	Quantity	Percentile	Studies
ReactJS	12/29	41.38%	[A2],[A4],[A6], [A9],[A20],[A7], [A24],[A25],[A21], [A26],[A27],[A28]
Angular	7/29	24.14%	[A4],[A7],[A8], [A16],[A27],[A28], [A29]
Vue	3/29	10.34%	[A4],[A7],[A28]
Single-SPA	2/29	6.90%	[A20],[A23]
Apache Cordova	1/29	3.45%	[A19]
Ionic	1/29	3.45%	[A19]
NextJS	1/29	3.45%	[A24]
Not specified	12/29	41.38%	[A1],[A3],[A5], [A10],[A11],[A12], [A13],[A14],[A15], [A17],[A18],[A22]

5.4. FQ3: What are the design patterns used to develop and deploy micro-frontends?

Table 10 presents an overview of the results regarding the design strategies adopted. Our results reveal a notable lack of consensus regarding design patterns (or strategies) for micro-frontend implementation. In 51.72% of the studies (15/29), no design pattern was specified, indicating either insufficient reporting practices or a general absence of formal architectural planning. Among the studies that did identify patterns, Web components were the most frequently employed, accounting for 27.59% (8/29). These components enable the encapsulation of functionality and layout, aligning with micro-frontend goals of independence and reusability. Other patterns, such as component-based composition, modular micro-frontend, and iframe composition, appeared in only 6.90% (2/29) of the studies, suggesting limited adoption or lack of uniform terminology.

The scarcity of reported patterns suggests a gap in formalizing best practices within the field. This fragmentation impairs replicability and hinders the ability to evaluate architectural decisions empirically. As micro-frontend architecture gains traction, the absence of standardized design strategies may lead to increased technical debt or architectural degradation. The underreporting of patterns also points to a reliance on implicit knowledge or practitioner experience rather than explicit documentation, limiting academic contributions to the topic.

Technically, developers favor decomposing applications into components to enhance maintainability. However, the results show that few studies leverage well-established design patterns. Advanced constructs such as compound components are absent, which facilitate efficient property handling and promote flexible component composition. This oversight highlights a need for more precise guidance on structuring micro-frontend codebases. A comprehensive framework for pattern adoption and documentation could support academic rigor and practical implementation.

Table 10: List of design patterns to build micro–frontends (FQ3).

Architecture	Quantity	Percentile	Studies
Web Component	8/29	27.59%	[A1],[A3],[A5],[A7], [A9],[A12],[A16],[A27]
Component-Based	2/29	6.90%	[A4],[A20]
Composition	2/29	6.90%	[A11],[A23]
Modular micro–frontend	2/29	6.90%	[A20],[A23]
Iframe Composition	2/29	6.90%	[A23],[A28]
Fragmented	1/29	3.45%	[A11]
No-visual-representation	1/29	3.45%	[A11]
Unified SPA	1/29	3.45%	[A16]
Separate Runtime	1/29	3.45%	[A23]
Modular functional Components	1/29	3.45%	[A26]
Not specified	15/29	51.72%	[A2],[A6],[A8],[A10], [A13],[A14],[A15],[A17], [A18],[A19],[A21],[A22], [A24],[A25],[A29]

5.5. FQ4: How are the micro–frontends evaluated?

The results presented in Table 11 reveal a critical gap in evaluating micro-frontends. Of the 29 primary studies analyzed, 72.41% (21 studies) did not specify any evaluation method. This lack of formal assessment undermines efforts to systematically compare implementations or validate architectural choices. The few studies that did report evaluations adopted diverse and fragmented approaches, including ATAM-based analysis, data visualization, load testing, qualitative evaluation, and performance metrics such as time to interact and build time. The absence of standard practices reflects the field's novelty and the need for methodological rigor in assessing micro-frontend systems.

The scarcity of defined evaluation strategies signals a compelling opportunity for future research. There is a need to identify appropriate metrics and propose replicable evaluation frameworks that reflect the unique characteristics of micro-frontend architectures, such as modularity, independent deployment, and integration complexity. Establishing

consistent practices would help developers understand trade-offs, improve system performance, and support user-centered design. Ultimately, advancing evaluation methods in this domain could lead to higher software quality and more effective user experiences in modular web applications.

Table 11: micro–frontend evaluations (FQ4).

Results	Quantity	Percentile	Studies
ATAM-based	1/29	3.45%	[A1]
Data Visualization	1/29	3.45%	[A2]
Quality of Experience	1/29	3.45%	[A16]
Time-Instant	1/29	3.45%	[A16]
Testing in Test Environment	1/29	3.45%	[A20]
Cycle of Testing and Issue Fixing	1/29	3.45%	[A20]
Static Code Analysis	1/29	3.45%	[A20]
ISO-20510	1/29	3.45%	[A23]
Load Test	1/29	3.45%	[A24]
Score Time to Interactive	1/29	3.45%	[A27]
Build Time	1/29	3.45%	[A27]
Qualitative Evaluation	1/29	3.45%	[A29]
Not specified	21/29	72.41%	[A3],[A4],[A5],[A6],[A7], [A8],[A9],[A10],[A11], [A12],[A13],[A14],[A15], [A17],[A18],[A19],[A21], [A22],[A25],[A26],[A28]

5.6. FQ5: How are the benefits of micro–frontends perceived by users?

The results in Table 12 indicate that the benefits of micro-frontends remain underreported in the current literature. A substantial proportion of studies (62.07%, 18/29) did not specify any benefits, reflecting a lack of systematic investigation into end-user impacts. Among the studies that did report benefits, flexibility was the most cited (27.59%), followed by maintainability and scalability (each with 24.14%). These findings suggest that while some advantages of micro-frontends are recognized, such as modular architecture and ease of scaling, a comprehensive understanding of their value remains limited.

The limited focus on end-user benefits restricts the ability to generalize or quantify how micro-frontends enhance user experience. Nevertheless, the identified advantages—particularly flexibility—are consistent with architectural principles that support smoother integration of legacy systems and phased technology migrations. This modularity allows for controlled deployment and minimizes disruption to users. Without adequate empirical validation or user-centered evaluation, however, these perceived benefits remain assumptions rather than evidence-based claims. Future research should address this gap by designing studies that explicitly evaluate the user impact of micro-frontend architectures.

Table 12: List of perceived benefits of using micro–frontends (FQ5).

Results	Quantity	Percentile	Studies
Maintenance	7/29	24.14%	[A1],[A11],[A16], [A20],[A24],[A26], [A27]
Scalability	7/29	24.14%	[A1],[A11],[A16], [A20],[A23],[A26], [A27],[A28],[A29]
Flexibility	8/29	27.59%	[A1],[A11],[A20], [A23],[A26],[A27], [A28],[A29]
Configurability	2/29	6.90%	[A1],[A11]
Cost Reduction	1/29	3.45%	[A1]
Implementation	1/29	3.45%	[A11]
Versioning Independent	1/29	3.45%	[A20]
Performance	1/29	3.45%	[A29]
Not specified	18/29	62.07%	[A2],[A3],[A4],[A5], [A6],[A7],[A8],[A9], [A10],[A12],[A14], [A15],[A17],[A18], [A19],[A21],[A22], [A25]

5.7. FQ6: What are the motivations for selecting micro–frontends?

The findings related to the motivations for selecting micro–frontends suggest a limited and uneven understanding within the primary studies. As shown in Table 13, 62.07% (18/29) of the studies did not report any specific motivation. This omission may reflect either the implicit adoption of micro–frontends without formal rationale or the absence of rigorous documentation practices in software projects. Such a gap reveals the need for empirical investigations to systematically identify the factors driving the architectural decision to adopt micro–frontends, especially in projects with varying levels of complexity and team maturity.

Among the studies that did specify motivations, modularity, and isolation were the most frequently cited reasons, appearing in 20.69% (6/29) of cases. Flexibility followed closely at 17.24% (5/29), while maintainability accounted for 13.79% (4/29). These motivations align with the architectural benefits of micro–frontend approaches, such as managing distributed teams, enabling independent deployments, and supporting heterogeneous technology stacks. These factors indicate that the use of micro–frontends is often driven by the need to decompose complex interfaces into smaller, manageable units, thereby increasing scalability and adaptability in modern software systems.

Table 13: Motivations for micro–frontend selection (FQ6).

Results	Quantity	Percentile	Studies
Modularity / isolation	6/29	20.69%	[A1],[A2],[A11],[A16], [A20],[A23]
Flexibility	5/29	17.24%	[A1],[A11],[A20], [A23],[A26]
Maintainability	4/29	13.79%	[A1],[A16],[A20], [A26],[A27]
Independent development	3/29	10.34%	[A1],[A20],[A23]
Scalability	3/29	10.34%	[A16],[A20],[A23], [A26],[A27]
Independent release	3/29	10.34%	[A20],[A26],[A27]
Structure-induced problems	2/29	6.90%	[A1],[A4]
Reduced costs	2/29	6.90%	[A1],[A26]
Short release cycles	2/29	6.90%	[A1],[A26]
Technical Upgrades	2/29	6.90%	[A28],[A29]
Performance	1/29	3.45%	[A16]
Deployment	1/29	3.45%	[A29]
Not specified	18/29	62.07%	[A2],[A3],[A5],[A6], [A7],[A8],[A9],[A10], [A12],[A13],[A14], [A15],[A17],[A18], [A19],[A21],[A22], [A25]

5.8. *FQ7: What types of software systems are being developed in the industry using micro–frontend architecture, and what are their key characteristics?*

Analyzing the domains in which micro-frontend architecture has been applied reveals a concentration in complex, large-scale systems. As shown in Table 14, enterprise resource planning (ERP) systems appear most frequently (10.34%), indicating the architecture's suitability for modular and scalable enterprise solutions. Additional applications include educational platforms, government information systems, manufacturing systems, and spatial data management. These domains share a common demand for high modularity, independent development teams, and frequent deployment cycles, i.e., characteristics that align with the advantages provided by the micro-frontend architecture.

Notably, 58.62% of the studies reviewed did not report the specific domain of application, which limits the ability to generalize findings across industries. However, among the specified systems, the prevalence of ERP and industrial platforms confirms a trend in the general research question: micro-frontends are often employed alongside modern JavaScript frameworks (e.g., ReactJS) and architecture styles that prioritize modularity and maintainability. These characteristics are critical in domains requiring long-term evolution, where teams must manage multiple components independently without compromising system integrity.

These findings reinforce the need for domain-aware architectural decisions. While micro-frontends offer flexibility and modularity, their benefits are maximized in contexts that demand scalability and autonomous development. Therefore, FQ7 illustrates the current landscape of industry adoption and supports practitioners in evaluating whether their project's characteristics align with the architecture's strengths. This prevents inappropriate applications and contributes to longer-lasting and more maintainable front-end ecosystems.

Table 14: Domain and Characteristics of software systems in industry with micro–frontend.

Results	Quantity	Percentile	Studies
ERP - Real Estate Management	1/29	3.45%	[A1]
Team Management - Performance	1/29	3.45%	[A2]
Spatial Management - IoT Data	1/29	3.45%	[A4]
ERP	2/29	6.90%	[A16], [A20]
Engineering - Industrial Automation	1/29	3.45%	[A11]
Government - Management Information	1/29	3.45%	[A24]
Manufacturing – Factory Operations	1/29	3.45%	[A26]
Educational - Learning support	1/29	3.45%	[A27]
EMIS - Undergraduate Education	1/29	3.45%	[A28]
Geophysics - Oil & Gas	1/29	3.45%	[A29]
Academic Work	1/29	3.45%	[A23]
Not Include	17/29	58.62%	[A3],[A5],[A6], [A7],[A8],[A9], [A10],[A12],[A13], [A14],[A15],[A17], [A18],[A19],[A21], [A22],[A25],

5.9. *SQ1: Where have the studies been published?*

The geographical distribution of studies, with 58.62% originating from European countries (Figure 4), suggests that Europe is leading the exploration of micro–frontends in modern software engineering. Although other countries contribute, their presence is comparatively smaller. This regional concentration reflects current research efforts but points to an opportunity for global expansion. To more inclusive and symbiotic human-technology interactions [51],

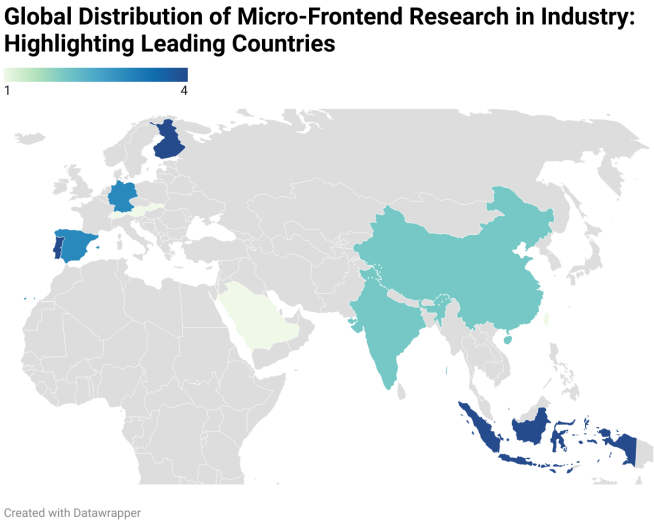

Figure 4: Global Distribution of Micro-Frontend Research in Industry

future research must extend beyond Europe, incorporating diverse perspectives and technologies.

This concentration of studies can be directly linked to the challenge of human-technology symbiosis [51], where the integration of human values with technological advances needs to be demonstrated. As technology, such as micro–frontends, evolves to exhibit characteristics similar to human behaviour, such as adaptability and problem-solving, there is a growing need to optimise people and technology. By enabling modular user interfaces, micro–frontends can play a critical role in creating ecosystems that are intuitive and responsive to human aspects while respecting individual values such as accessibility and adaptability.

For example, extending the research to underrepresented regions could reveal new ways micro-frontends support the human-technology symbiosis in different cultural and technological contexts. This approach could further enhance the user experience by fostering intelligent environments where technology not only supports human behaviour and perspectives. The research also suggests that the human-technology symbiosis will require adaptable solutions that can be smoothly integrated into everyday life, making user interaction seamless and minimizing cognitive load.

5.10. SQ2: How did the studies evolve in recent years?

Figure 5 shows clear trends in the publication of front-end development studies over the period evaluated. In 2012, these studies accounted for only 4% of the total. In 2014, it increased to 11.11%, detailing an interest in front-end technologies. However, this momentum slowed down in 2017, where the percentage remained at 4%. In 2018, research activity increased again, reaching 7.41%, followed by a notable jump in 2020, when 11.11% of studies were published. The most significant increase occurred in 2021, when 22.22% of studies focused on front-end development, marking a peak in the trend. These trends suggest a significant increase in front-end development research over the last twelve years. By August 2024, 3.70% of the evaluated studies will have explored front-end development topics.

The rise in research activity reflects the software industry's growing emphasis on developing efficient front-end architectures. This shift highlights the need for scalable, maintainable, and modular front-end solutions as developers and researchers increasingly prioritize performance and flexibility in modern web applications. The growth in research supports the broader trend toward more dynamic, user-centered front-end solutions that align with the industry's evolving demands. As shown in Figure 5, it illustrates this upward trajectory, emphasizing the heightened attention given to front-end development in recent years.

6. Implications and Challenges for Future Research

Figure 6 presents a bubble chart that organizes the primary studies across three dimensions: architecture used (d1), publication year (d2), and number of studies (d3). Each bubble represents a combination of these values, allowing for clear visualization of the relationships between the technologies used to create user interfaces (d1), the year of publication (d2), and the number of studies (d3). This chart illustrates how our primary studies evolved, showing their distribution across different years.

Implications for human-technology symbiosis. micro–frontend architecture can promote a vision where humans live and work harmoniously with technology. It advances the user experience by promoting modular, scalable, and flexible user interfaces, which can seamlessly adapt to user needs across diverse platforms. This superior modularity can promote human-technology symbiosis by enabling technology to respond to user behavior more intuitively, mirroring human cognitive processes such as learning and reasoning. In addition, as smart environments become more interconnected and autonomous, mainly with the widespread adoption of smart wearable devices, micro–frontends can improve human environment interactions for two reasons.

First, it can help users engage with entire ecosystems more cohesively and responsively, not just isolated technologies. Second, the widespread adoption of smart wearable devices allows for collecting personalized user data, enabling enhanced user experiences by leveraging data closely linked to individual behaviors and preferences rather than relying on generic, pre-formatted information. Investigating how micro–frontend architecture can be optimized to create intuitive, adaptive, and seamless interactions between humans and their technological surroundings can contribute to a future where technology enhances human life without overwhelming or fragmenting user experiences. Finally, this research can help define best practices for creating intelligent environments that respect human values while promoting harmonious integration with emerging technologies.

Implications for human environment interactions. micro–frontend architecture offers notable benefits for addressing the human environment interaction challenge [51], particularly

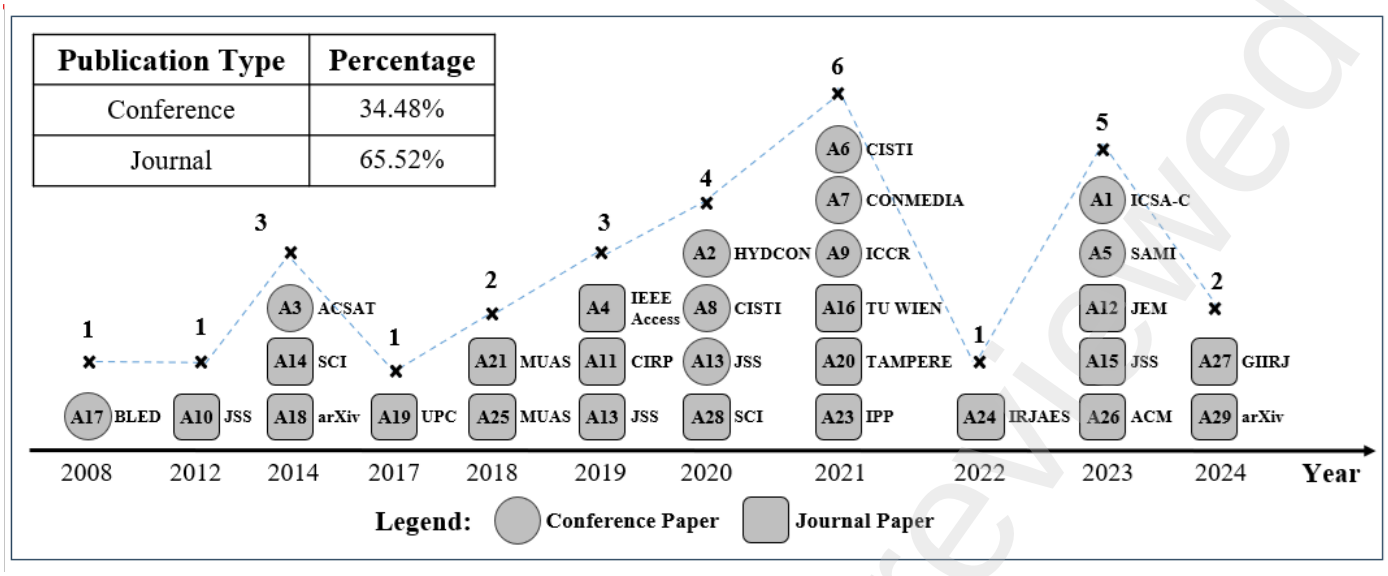

Figure 5: Distribution of primary studies over the years (based on [16])

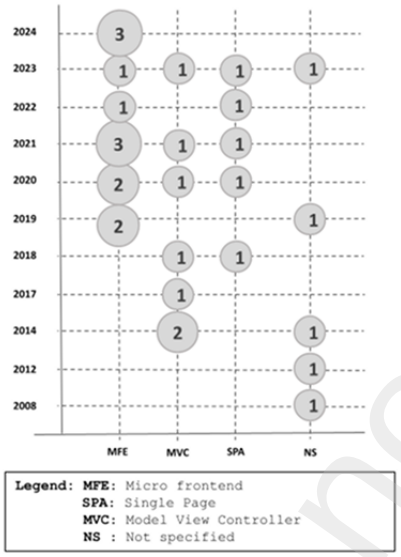

Figure 6: Bubble chart details the relationship between the three variables. The X-axis shows the architecture used. The Y-axis represents the year. The number of primary studies according to the X- and Y-axis criteria applied to determine the bubble size.

in smart and interactive ecosystems. By breaking down monolithic front-end systems into smaller, independent components, micro–frontends may create flexible, modular interfaces that can adapt seamlessly to evolving environments.

Two research questions that the scientific community could investigate are: (1) How can these smaller components enhance user experiences in intelligent ecosystems? micro–frontends support non-invasive, intuitive interactions by enabling personalized user journeys across devices, reducing cognitive load, and minimizing confusion; and (2) Could the ability to blend physical and digital interactions more smoothly improve how people engage with their surroundings? Additionally, micro–frontends allow for real-time adjustments, ensuring intelligent environments remain responsive and accountable to users. This architectural approach scales up existing HCI methods to meet the demands of complex and multimodal interactions. This can improve the flow of human environment interactions.

In addition, with the rise of intelligence-as-a-service design principles [51], innovations such as micro-frontends can help support more dynamic, adaptive and context-aware user experiences. These principles allow intelligent services to be seamlessly integrated into front-end architectures. This can enable systems to respond to user needs in real time by leveraging cloud–based AI and data analytics. As a result, micro-frontends can continuously adapt interfaces to individual preferences and environmental conditions, making interactions more intuitive and efficient. This evolution enhances the symbiosis between humans and technology, fostering smarter, more responsive ecosystems.

7. Conclusion and Future Work

This study systematically mapped technologies, design patterns, and industrial practices related to micro-frontend architectures in enterprise systems. We analyzed 29 primary studies and addressed ten research questions, offering a structured overview of the field. The investigation focused on identifying the technologies employed (GQ, FQ1–FQ3), evaluating the methods used to assess micro-frontends (FQ4), and understanding their perceived benefits, adoption motivations, and application domains (FQ5–FQ7). Additionally, we examined where and how the research has evolved (SQ1–SQ2).

Our results show that JavaScript-based technologies dominate the landscape, with ReactJS and Web Components being the most frequently employed tools. Design patterns

and architectural choices are often insufficiently documented, indicating a need for standardization. The adoption of micro-frontends is mainly driven by goals of modularization, flexibility, and team autonomy. However, few studies rigorously evaluate user experience or explore the intersection of micro-frontend architectures with Human-Computer Interaction (HCI) principles, particularly in cognitive load, usability, and adaptive user interfaces.

These findings expose key research gaps and signal opportunities for future exploration. There is no empirical evidence concerning how micro-frontends affect user interaction quality, especially in dynamic and large-scale environments. Future work should extend the current mapping by incorporating additional digital libraries and expanding the scope to include longitudinal case studies of industrial adoption. Developing standardized evaluation frameworks to measure cognitive effort, usability, and adaptability in micro-frontend implementations is essential.

Our study contributes by consolidating existing knowledge and setting a foundation for future research. It offers a reproducible protocol for subsequent reviews, supports researchers in identifying underexplored themes, and aids practitioners in making informed architectural decisions. We position this work as the first step toward a broader research agenda that connects software engineering with the grand challenges of HCI, promoting the development of modular, usable, and cognitively sustainable systems. Finally, the gaps outlined in this study call for interdisciplinary collaborations. Bridging micro-frontend architecture with HCI research can inform the design of systems that align more closely with human capabilities and expectations, contributing to more inclusive, adaptive, and efficient software ecosystems.

8. Acknowledgement

This work was supported in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), under Grant No. 314248/2021-8, and Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), under Grant No. 21/2551-0002051-2.

References

[1] M. Rubert, K. Farias, On the effects of continuous delivery on code quality: A case study in industry, Computer Standards & Interfaces 81 (2022) 103588.

[2] A. Oliveira, V. Bischoff, L. J. Gonçales, K. Farias, M. Segalotto, Brcode: An interpretive model-driven engineering approach for enterprise applications, Computers in Industry 96 (2018) 86–97.

[3] D. Taibi, L. Mezzalira, Micro-frontends: Principles, implementations, and pitfalls, ACM SIGSOFT Software Engineering Notes 47 (4) (2022) 25–29.

[4] Y. Prajwal, J. V. Parekh, R. Shettar, A brief review of micro-frontends, United International Journal for Research and Technology 2 (8) (2021).

[5] H. Harms, C. Rogowski, L. Lo Iacono, Guidelines for adopting frontend architectures and patterns in microservices-based systems, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 902–907.

[6] J. Männistö, A.-P. Tuovinen, M. Raatikainen, Experiences on a frameworkless micro-frontend architecture in a small organization, in: 2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C), 2023, pp. 61–67. doi:10.1109/ICSA-C57050.2023.00025.

[7] A. Pavlenko, N. Askarbekuly, S. Megha, M. Mazzara, Micro-frontends: application of microservices to web front-ends., Journal of Internet Services and Information Security. 10 (2) (2020).

[8] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, et al., The prisma 2020 statement: an updated guideline for reporting systematic reviews, International journal of surgery 88 (2021) 105906.

[9] I. D. Cooper, What is a "mapping study?", Journal of Medical Library Association 104 (1) (2016) 76–78. doi:10.3163/1536-5050.104.1.013.

[10] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and software technology 64 (2015) 1–18.

[11] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, et al., Experimentation in software engineering, Vol. 236, Springer, 2012.

[12] L. d. S. Paula, L. Pfeiffer Salomão Dias, R. Francisco, J. L. V. Barbosa, Analysing iot data for anxiety and stress monitoring: A systematic mapping study and taxonomy, International Journal of Human–Computer Interaction 40 (5) (2024) 1174–1194.

[13] S. O. Dorneles, R. Francisco, D. N. F. Barbosa, J. L. V. Barbosa, Context awareness in recognition of affective states: A systematic mapping of the literature, International Journal of Human–Computer Interaction 39 (8) (2023) 1563–1581.

[14] J. P. Menzen, K. Farias, V. Bischoff, Using biometric data in software engineering: a systematic mapping study, Behaviour & Information Technology 40 (9) (2021) 880–902.

[15] V. Bischoff, K. Farias, J. P. Menzen, G. Pessin, Technological support for detection and prediction of plant diseases: A systematic mapping study, Computers and Electronics in Agriculture 181 (2021) 105922.

[16] C. Eduardo Carbonera, K. Farias, V. Bischoff, Software development effort estimation: A systematic mapping study, IET Software 14 (4) (2020) 328–344.

[17] S. Peltonen, L. Mezzalira, D. Taibi, Motivations, benefits, and issues for adopting micro-frontends: A multivocal literature review., Information and Software Technology. 136 (106571) (2021).

[18] H. Cabane, K. Farias, On the impact of event-driven architecture on performance: An exploratory study, Future Generation Computer Systems 153 (2024) 52–69.

[19] K. Petersen, R. Feldt, S. Mujtaba, et al., Systematic mapping studies in software engineering, in: International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008, pp. 1–10. doi:10.14236/ewic/EASE2008.8.

[20] V. Bischoff, K. Farias, L. J. Gonçales, J. L. V. Barbosa, Integration of feature models: A systematic mapping study, Information and Software Technology 105 (2019) 209–225.

[21] L. J. Gonçales, K. Farias, T. C. D. Oliveira, M. Scholl, Comparison of software design models: an extended systematic mapping study, ACM Computing Surveys (CSUR) 52 (3) (2019) 1–41.

[22] P. Y. Tilak, V. Yadav, S. D. Dharmendra, N. Bolloju, A platform for enhancing application developer productivity using microservices and micro-frontends, in: 2020 IEEE-HYDCON, 2020, pp. 1–4. doi:10.1109/HYDCON48903.2020.9242913.

[23] H. M. Abdullah, A. M. Zeki, Frontend and backend web technologies in social networking sites: Facebook as an example, in: 2014 3rd International Conference on Advanced Computer Science Applications and Technologies, 2014, pp. 85–89. doi:10.1109/ACSAT.2014.22.

[24] M. Mena, A. Corral, L. Iribarne, J. Criado, A progressive web application based on microservices combining geospatial data and the internet of things, IEEE Access 7 (2019) 104577–104590. doi:10.1109/ACCESS.2019.2932196.

[25] K. Micko, M. Beca, P. Papcun, I. Zolotova, Responsive web applications enables automatization of karaoke song playlist, in: 2023 IEEE 21st World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2023, pp. 000133–000138. doi:10.1109/SAMI58000.2023.10044533.

[26] H. Antunes, I. d. S. A. da Fonseca, Advanced web methodology

for flexible web development, in: 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), 2021, pp. 1–4. doi:10.23919/CISTI52073.2021.9476295.

[27] C. Rafael, P. W. Adikusumo, N. A. B. Sanjaya, M. S. Anggreainy, Application of software engineering in intermediate and higher education through web apps development, in: 2021 6th International Conference on New Media Studies (CONMEDIA), 2021, pp. 135–139. doi:10.1109/CONMEDIA53104.2021.9617201.

[28] C. Andrade, P. Alves, J. E. Fernandes, F. Coutinho, A web platform to support mentoring programs in higher education, in: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), 2020, pp. 1–6. doi:10.23919/CISTI49556.2020.9140982.

[29] P. Dwivedi, Kshamta, A. Joshi, Reactjs for trading applications, in: 2022 International Conference on Cyber Resilience (ICCR), 2022, pp. 01–07. doi:10.1109/ICCR56254.2022.9995932.

[30] O. Zimmermann, C. Miksovic, J. M. Küster, Reference architecture, metamodel, and modeling principles for architectural knowledge management in information technology services, Journal of Systems and Software 85 (9) (2012) 2014–2033, selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011). doi:https://doi.org/10.1016/j.jss.2012.05.003.

[31] E. Schäffer, A. Mayr, J. Fuchs, M. Sjarov, J. Vorndran, J. Franke, Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions, Procedia CIRP 86 (2019) 86–91. doi:https://doi.org/10.1016/j.procir.2020.01.017.

[32] S. Shen, C.-Z. Qin, L.-J. Zhu, A.-X. Zhu, From scenario to roadmap: Design and evaluation of a web-based participatory watershed planning system for optimizing multistage implementation plans of management practices under stepwise investment, Journal of Environmental Management 342 (2023) 118280. doi:https://doi.org/10.1016/j.jenvman.2023.118280.

[33] H. Sneed, C. Verhoef, Re-implementing a legacy system, Journal of Systems and Software 155 (2019) 162–184. doi:https://doi.org/10.1016/j.jss.2019.05.012.

[34] X.-Y. Su, F.-C. Lin, L. Chen, K.-C. Huang, C.-W. Lu, C.-Y. Chen, T.-W. Ho, F. Lai, A service oriented tele-health promotion information system with mobile application, Procedia Computer Science 37 (2014) 274–281, the 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2014)/ The 4th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH 2014)/ Affiliated Workshops. doi:https://doi.org/10.1016/j.procs.2014.08.041.

[35] A. Rio, F. Brito e Abreu, Php code smells in web apps: Evolution, survival and anomalies, Journal of Systems and Software 200 (2023) 111644. doi:https://doi.org/10.1016/j.jss.2023.111644.

[36] M. Kroiß, From backend to frontend-case study on adopting micro frontends from a single page erp application monolith, Ph.D. thesis, Wien (2021).

[37] C. Adolphs, P. Schubert, Persobox: A personalization engine between erp system and web frontend, BLED 2008 Proceedings (2008) 7.

[38] D. Ziani, Configuration in erp saas multi-tenancy, arXiv preprint arXiv:1405.0650 (2014).

[39] J. Font Escribano, V. K. Reynoso Vasquez, Erp implementation for an administrative agency as a corporative frontend and an e-commerce smartphone app, Master's thesis, Universitat Politècnica de Catalunya (2017).

[40] L. Linkola, Design and implementation of modular frontend architecture on existing application, Master's thesis, Tampere University (2021).

[41] J. Koppala, Erp solution with reactjs (2018).

[42] J. Gunawan, R. Kosala, Genie enterprise resource planning for small medium enterprises implementing single page web application, in: IOP Conference Series: Earth and Environmental Science, Vol. 426, IOP Publishing, 2020, p. 012170.

[43] R. A. P. da Silva, A micro frontends solution–analyzing quality attributes, Ph.D. thesis, Instituto Politecnico do Porto (Portugal) (2021).

[44] I. W. K. D. BP, D. Anggraini, a development of modern web application frontend structures using micro frontends, International Research Journal of Advanced Engineering and Science 7 (1) (2022) 149–155.

[45] M. Hiltunen, Creating multiplatform experiences with progressive web apps (2018).

[46] B. Simões, M. Del Puy Carretero, J. Martinez Santiago, S. Muñoz Segovia, N. Alcain, Twinark: A unified framework for digital twins based on micro-frontends, micro-services, and web 3d, in: Proceedings of the 28th International ACM Conference on 3D Web Technology, Web3D '23, Association for Computing Machinery, New York, NY, USA, 2023. doi:10.1145/3611314.3615915.

[47] D. C. Hidayat, I. K. J. Atmaja, I. B. G. Sarasvananda, Analysis and comparison of micro frontend and monolithic architecture for web applications, Jurnal Galaksi 1 (2) (2024) 92–100. doi:10.70103/galaksi.v1i2.19.

[48] D. Wang, D. Yang, H. Zhou, Y. Wang, D. Hong, Q. Dong, S. Song, A novel application of educational management information system based on micro frontends, Procedia Computer Science 176 (2020) 1567–1576.

[49] F. Antunes, M. J. D. Lima, M. A. P. AraÃšjo, D. Taibi, M. Kalinowski, Investigating benefits and limitations of migrating to a micro-frontends architecture, arXiv preprint arXiv:2407.15829 (2024).

[50] J. Vepsäläinen, A. Hellas, P. Vuorimaa, Overview of web application performance optimization techniques, arXiv preprint arXiv:2412.07892 (2024).

[51] C. Stephanidis, G. Salvendy, M. Antona, J. Y. Chen, J. Dong, V. G. Duffy, X. Fang, C. Fidopiastis, G. Fragomeni, L. P. Fu, et al., Seven hci grand challenges, International Journal of Human–Computer Interaction 35 (14) (2019) 1229–1269.