Designing and Evaluating a Catalog of Micro Frontends Anti-patterns

Nabson Silva[1], **Tayana Conte**[1]

[1]Institute of Computing (IComp) – Federal University of Amazonas (UFAM)
Mailbox 6045 – 13083-970 – Manaus – AM – Brazil

{nabson.paiva,tayana}@icomp.ufam.edu.br

Abstract. *Micro frontend (MFE) is an architectural style that decomposes a monolithic frontend application into smaller, manageable, and independently deployable slices. Despite its increasing adoption, the field remains relatively underexplored, particularly in terms of identifying challenges and documenting best practices. Therefore, the goal of this Master's Thesis is to propose and evaluate an artifact that supports developers in implementing MFE architectures. We introduce a catalog of MFE anti-patterns that document common problems and practical solutions. After proposing the catalog's initial version, we conducted three empirical studies, a Personal Opinion Survey, a Controlled Experiment, and Multivocal Literature Review. After each study, we refined the catalog to produce a final version that helps developers identify, solve, and prevent problems when working with MFE architectures. The contributions of this Thesis include centralized documentation of common issues and solutions when developing MFE architectures, empirical evidence on how the catalog can be used, a web application that showcases the anti-patterns and promotes collaboration within industry practitioners, and the development of MFE teaching material that instructors can integrate into software architecture curricula. We believe that the results of this work have the potential to drive significant advances in both the practice and theory of MFE, helping shape future research and improve industry adoption.*

1. Introduction

Software architecture focuses on software design at the highest level of abstraction [Kazman et al. 2023]. At this level, architects are concerned not with specific classes or interfaces but how the system's components, modules, or layers are integrated [Valente 2020]. During the design phase, architects face critical decisions that shape the system, as these choices are often difficult to modify later. To support these decisions, architects can rely on architectural styles, which provide guidelines on organizing the system's modules [Medvidovic and Taylor 2010]. Examples of such architectural styles include Microservices and Micro Frontends.

As a monolithic application grows, it becomes challenging to scale due to limitations like technology constraints, the necessity for vertical scaling only, and the need to reboot the entire application with each deployment [Dragoni et al. 2017]. To address these issues, developers are adopting the Microservice (MS) architectural style to create autonomous, distributed, and loosely coupled services [Dmitry and Manfred 2014,

Lewis and Fowler 2014, Erl 2016]. This architecture enables teams to work independently, reducing the development time for new features. However, different teams often still need to share the same codebase for the presentation layer.

Micro Frontend (MFE) is an architectural style that extends the principles of MS to the frontend, breaking down a complex frontend application into smaller, manageable, and independently deployable slices [Mezzalira 2021a, Peltonen et al. 2021]. This approach facilitates independent testing, development, and deployment of front-end components, enabling teams to work autonomously and reducing the development time for new features [Geers 2020]. However, some issues faced when adopting MFEs are increased payload size, UX consistency, complex monitoring and debugging, state management, and duplicated code [Peltonen et al. 2021]. Many companies, such as SAP, Springer, Zalando, NewRelic, Ikea, Starbucks, and DAZN, have successfully implemented the MFE architecture [Taibi and Mezzalira 2022], showcasing its potential in diverse domains.

Despite its widespread adoption in the industry, the academic literature about guidelines and best practices when implementing MFE is relatively limited, especially when compared to the numerous experience reports and case studies documenting its implementation [Antunes et al. 2024, Capdepon et al. 2023, Kaushik et al. 2024, Perlin et al. 2023, Männistö et al. 2023, Pölöskei and Bub 2021, Moraes et al. 2024]. This disparity suggests a challenge in transferring knowledge from industry practice to academic research and underscores a gap in understanding how enterprises implement MFE in real-world architectures.

Over time, software architecture can deteriorate due to developers' insufficient understanding of the specific architectural style [Taibi et al. 2020]. This issue is particularly critical in MFE architectures, as they are inherently complex, and there is no well-defined method for evaluating or documenting both good and bad practices. Therefore, the objective of this master's thesis is to develop artifacts that can support developers in implementing and maintaining MFE architectures. We propose a catalog of MFE anti-patterns to preserve architectural integrity and assist developers in making well-informed decisions. Anti-patterns address emerging issues, common mistakes, poorly implemented solutions, misapplied best practices, and deviations from established process models [Brada and Picha 2019].

To evaluate the catalog, we first conducted a Personal Opinion Survey [Kitchenham et al. 2015] with practitioners to validate whether the identified problems are prevalent in MFE architectures and if the proposed solutions address them effectively. Based on their feedback, we improved the catalog. We then carried out a controlled quasi-experiment with undergraduate students to compare the use of the catalog to MFE guidelines provided on blogs and evaluate whether the catalog enhances students' perception of learning. Again, the catalog was refined based on feedback. Finally, we expanded the catalog by conducting a Multivocal Literature Review [Garousi et al. 2019] to add anti-patterns published by practitioners in blogs, sites, videos, etc. Our next steps involve empirically evaluating its use by practitioners in real-world architectures.

2. Research Objectives

The primary objective of this thesis is **to develop an artifact to support developers when implementing and maintaining MFEs architectures**. To satisfy the primary objective,

we devised the following specific objectives:

1. Build a comprehensive body of knowledge on the real-world challenges developers face when implementing Micro Frontend architectures.
2. Develop and iteratively improve an artifact to support developers when implementing and maintaining Micro Frontend architectures.
3. Examine how the artifact supports inexperienced developers in identifying and avoiding common pitfalls while maintaining Micro Frontend architectures.

3. Research Methodology

The research methodology employed in this thesis is an adapted version of the evidence-based methodology proposed by Mafra et al. [Mafra et al. 2006]. Figure 1 illustrates the steps of our adapted approach. The methodology consists of the following steps:

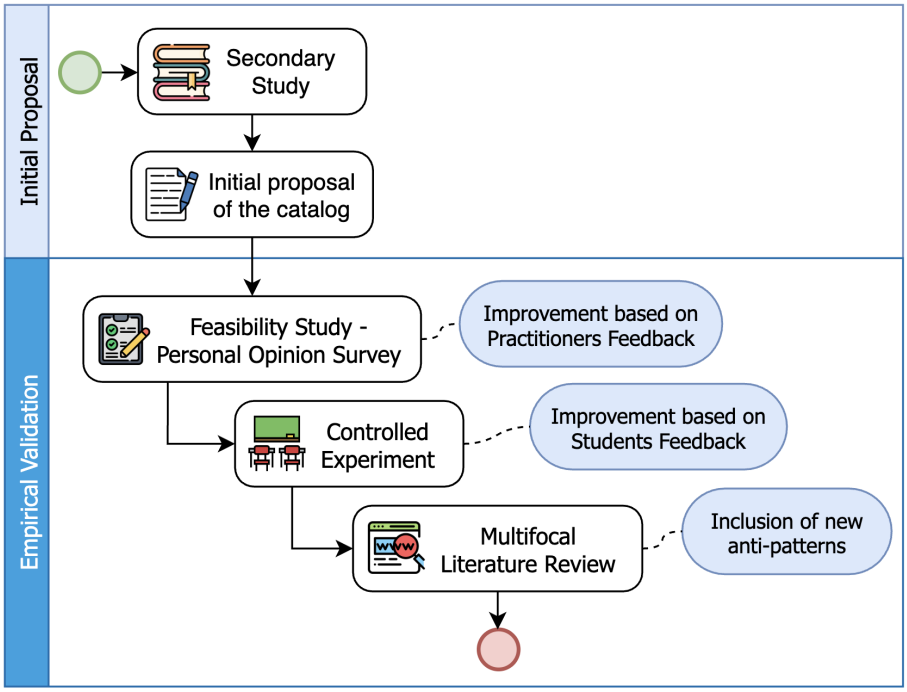

Figura 1. Diagram illustrating the research methodology based on Mafra et al. [Mafra et al. 2006]. Blue rounded items show what we made after each step.

1. **Secondary Study**: After conducting an ad hoc review, we identified a Multivocal Literature Review by Peltonen et al. [Peltonen et al. 2021] that maps existing knowledge on MFEs, highlighting their motivations, benefits, and challenges. The authors state that MFE anti-patterns have not yet been identified. To complement their review and verify whether any anti-patterns have been published since its completion, we conducted a Rapid Review [Cartaxo et al. 2018] and found no documented MFE anti-patterns.
2. **Initial proposal of the catalog**: To document the challenges faced when implementing and maintaining MFEs architectures, we propose a catalog of MFEs anti-patterns based on MSs anti-patterns. We propose 12 anti-patterns, each composed by name, category, problem, solution, and example.

3. **Feasibility study**: We conducted a Personal Opinion Survey [Kitchenham et al. 2015] with practitioners to assess and improve the anti-patterns and evaluate if they represent real problems on MFE architectures. We improved the catalog anti-patterns based on practitioners' feedback.
4. **Controlled experiment**: We conducted a controlled quasi-experiment [Wohlin et al. 2012] with Computer Science undergraduate students to compare the catalog to the guidelines proposed by practitioners on websites and blogs, and evaluate whether the catalog improves the perceived learning of students. We improved the catalog anti-patterns based on students' feedback.
5. **Multivocal Literature Review**: To uncover micro frontend (MFE) anti-patterns proposed by practitioners in informal sources, we conducted a Multivocal Literature Review [Garousi et al. 2019]. By incorporating anti-patterns shared by developers in online communities, blogs, and technical forums, we enriched and refined our catalog, producing a more comprehensive and practice-informed final version.

4. Final Catalog

After collecting feedback from participants and incorporating new anti-patterns from grey literature, we evolved our catalog and created it's final version with 27 anti-patterns. Drawing on the collaborative repository model proposed by Bogner et al.[Bogner et al. 2019], we developed an web application to showcase all anti-patterns.[1] The web application allows users to search for any word in the anti-patterns name, problem, solution, and example. Figure 2 presents the home screen of the web application and Table 1 presents the Global State Communication anti-pattern as an example.

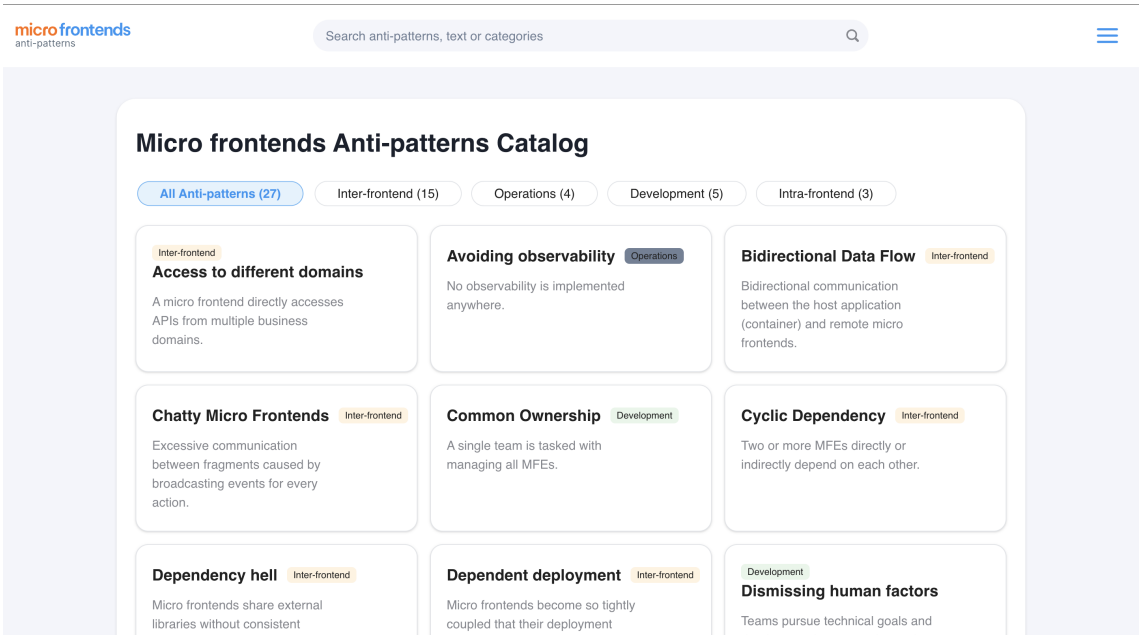

Figura 2. Main page of the catalog's web application.

[1] https://mfe-anti-patterns.online/micro-frontends-anti-patterns/#/catalog

Anti-pattern	**Global State Communication**
Also Known As	Relax, It's just code; Sharing state across Micro Frontends; Global State; Shared Global State
Category	Inter-frontend
Problem	Using shared states violates the principle of segregation, compromising the independence of each micro frontend and increasing coupling.
Symptoms and Consequences	Changes in the shared state require coordination across MFEs; otherwise, bugs may be introduced in MFEs that read the state, reducing independence.
Solution	Each micro frontend should maintain its own event store. To enable communication, use an event emitter-based approach instead of sharing state directly.
Resulting Context	Each MFE may keep a local state (when needed) and communicate by subscribing to events dispatched by other MFEs, rather than directly accessing their state.
Example	In a financial platform, `MFE-account` maintains the current account balance in a shared global state, enabling other MFEs to access it directly. When the team internationalizes the app and extends currency support, `MFE-account` changes the shared state structure to include both the numeric balance and its associated currency. Since other MFEs depend on this shared state without proper contracts, the change breaks their logic. Applying the solution, each MFE stops depending on shared state directly and instead subscribes to events or queries data through well-defined, versioned interfaces, so updates in one MFE do not unexpectedly break others.
Solution Pitfalls	When creating local states, be careful to avoid the *Distributed Data Inconsistency* anti-pattern.
Related Anti-patterns	Distributed Data Inconsistency
References	[Mezzalira 2020, Shinde 2022, Mezzalira 2021b, Mezzalira 2021c, Raimundo 2023]

Tabela 1. Global State Communication anti-pattern summary.

5. Contributions

This research has made several contributions to the MFEs field, spanning theoretical and practical aspects. Below, we categorize and describe the contributions of this thesis thus far:

- **Theoretical contributions**:
  1. A Catalog of Micro Frontends Anti-patterns, highlighting common problems and effective solutions, bridging knowledge from industry to academia.
  2. Empirical evidence on the use of the Micro Frontends anti-patterns catalog by students while learning Micro Frontends.
  3. Empirical evidence comparing the Micro Frontends anti-patterns catalog with Micro Frontends guidelines, focusing on students' ability to make informed decisions regarding the maintenance of a Micro Frontends architecture.

4. A Multivocal Literature Review that consolidates previously scattered practitioner knowledge on Micro Frontends, strengthening and enriching the theoretical and practical understanding of the field.

- **Practical contributions**:
  1. Development of a web application to present the Micro Frontends anti-patterns catalog, allowing developers to access it during the implementation and maintenance of Micro Frontends architectures while also enabling collaboration on the catalog.
  2. Development of Micro Frontends teaching materials that integrate theoretical and practical content, allowing instructors to incorporate this topic into Software Architecture courses seamlessly.
- **Published papers**:
  1. *A Catalog of Micro Frontends Anti-patterns* [Silva et al. 2025a]: Catalog's proposal and evaluation by practitioners through a Personal Opinion Survey. This paper was accepted for publication at the IEEE/ACM International Conference on Software Engineering (ICSE) 2025 – Research Track.
  2. *Evaluating Strategies for Teaching Micro Frontends: Do Anti-patterns Help?* [Silva et al. 2025b]: Experience report on teaching Micro Frontends to undergraduate Computer Science students, along with a controlled experiment comparing the catalog to guidelines provided by developers on the internet. This paper was accepted for publication at the XXXIX Brazilian Symposium on Software Engineering (SBES) 2025 – Education Track, and received a Distinguished Paper Award.
- **Planned papers**:
  1. *A Comprehensive Catalog of Micro Frontends Anti-patterns: A Multivocal Literature Review*: Report of the Multivocal Literature Review we conducted to expand the catalog by adding anti-patterns proposed by practitioners in Grey Literature sources. We intend to submit this paper to the Journal of Software and Systems (JSS).
  2. *Teaching Micro Frontends: Insights from Two Controlled Experiments on Guidelines and Anti-patterns*: Extended version of the paper published in SBES [Silva et al. 2025b] due to the invitation after receiving the Distinguished Paper Award. We will submit this paper to the Journal of Software Engineering Research and Development (JSERD).

6. Related work

Several papers present case studies or experience reports on implementing MFE architectures. Männistö et al. [Männistö et al. 2023] presented their experience through the migration of a monolithic frontend to the MFE architecture. Capdepon et al.[Capdepon et al. 2023] proposed an approach to migrate from monolithic mobile architecture to MFE. Moraes et al. [Moraes et al. 2024] provided an experience report demonstrating how the same application can be implemented using different MFE approaches. Perlin et al. [Perlin et al. 2023] presented a case study of how to implement a MFE application with Webpack. Kaushik et al. [Kaushik et al. 2024] proposed a framework for the design of web applications with MFEs and MS. Pavlenko et al. [Pavlenko et al. 2020] implemented MFEs case study to report and discuss the issues risen during development. All these papers can assist in making architectural decisions when implementing a MFE

architecture. However, they cannot be used to evaluate an architecture or serve as a guide for identifying hidden problems within it.

Although some practitioners have shared their experience on anti-patterns in MFE applications on blogs and keynotes [Mezzalira 2023, Shinde 2022, Rappl 2024], before this Master's Thesis, no scientific studies had been conducted to propose a catalog of MFE anti-patterns. We conducted a Rapid Review and found no studies proposing anti-patterns for MFEs. Mezzalira [Mezzalira 2021a] briefly mentions that sharing any state between micro-frontends is considered an anti-pattern but does not provide an in-depth definition or exploration of this or other anti-patterns. Peltonen et al.[Peltonen et al. 2021] conducted a Multivocal Literature Review and stated that researchers have not yet deeply investigated MFEs and patterns and anti-patterns have not been defined. On the educational side, to the best of our knowledge, no papers specifically discuss MFE education, and only a few report on experiences teaching MSs.

Given the shared characteristics of MS and MFE architectural styles, we also searched for papers proposing MS anti-pattern. Through a Systematic Literature Review (SLR), Cerny et al. [Cerny et al. 2023] crafted a catalog with 58 disjoint MS anti-patterns. Bogner et al. [Bogner et al. 2019] conducted a SLR to propose a taxonomy of 36 MS anti-patterns. Additionally, they developed a collaborative web application that allows users to explore and interact with their catalog. While both architectural styles share common anti-patterns, a dedicated catalog for MFEs is necessary to identify and address these issues specifically within the MFEs context.

7. Conclusion

By proposing and evaluating an MFE anti-pattern catalog, this dissertation advances Computer Science by turning fragmented practitioner experience around an emerging architectural style into structured, reusable, and empirically grounded knowledge. The catalog formalizes recurring failure modes by clearly defining problems, symptoms, consequences, and mitigation strategies, creating a shared conceptual basis for analyzing, comparing, and evolving MFE. Beyond documentation, this work provides empirical evidence of how these problems manifest in real systems and how practitioners use the catalog in day-to-day work, thereby strengthening the scientific understanding of MFE-specific constraints and informing future research on measurement, tool support, and automated detection. In doing so, the dissertation contributes both to theory, by characterizing phenomena and relationships in MFE architectures, and to practice, by enabling knowledge transfer that can improve architectural decision-making, education, and the quality of MFE-based systems at scale.

For future work, we plan to conduct a participatory case study to analyze how practitioners use the catalog and to provide empirical evidence of its impact on the overall quality of MFE architectures. We also aim to examine the effects of the proposed solutions, both positive and negative, investigating whether their application may introduce new anti-patterns. Regarding anti-pattern detection, we intend to formalize the definition of anti-patterns using a formal notation or specification language, enabling precise representation and systematic analysis. We plan to develop automated detection tools to help identify anti-patterns efficiently. Finally, we aim to explore how the catalog can serve as a foundational knowledge base for building AI-powered agents that assist practitioners in

making informed architectural decisions in micro frontends.

ACKNOWLEDGMENTS

We thank all participants of the studies. This paper was supported by the Brazilian Federal Agency for Support and Evaluation of Graduate Education — Brazil (AUXPE-CAPES-PROEX) — Finance Code 001. Additionally, this work was partially funded by the Amazonas State Research Support Foundation — FAPEAM — through the PDPG-CAPES project. We also would like to thank the financial support granted by CNPq 314797/2023-8; CNPq 443934/2023-1; and CNPq 445029/2024-2.

Referências

Antunes, F., Lima, M., Araújo, M., Taibi, D., and Kalinowski, M. (2024). Investigating benefits and limitations of migrating to a micro-frontends architecture. In *XXXVIII Simpósio Brasileiro de Engenharia de Software*, pages 103–113, Porto Alegre, RS, Brazil. SBC.

Bogner, J., Boceck, T., Popp, M., Tschechlov, D., Wagner, S., and Zimmermann, A. (2019). Towards a collaborative repository for the documentation of service-based antipatterns and bad smells. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 95–101. IEEE.

Brada, P. and Picha, P. (2019). Software process anti-patterns catalogue. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, pages 1–10.

Capdepon, Q., Hlad, N., Seriai, A.-D., and Derras, M. (2023). Migration process from monolithic to micro frontend architecture in mobile applications. In *Proceeding of the International Workshop on Smalltalk Technologies*.

Cartaxo, B., Pinto, G., and Soares, S. (2018). The role of rapid reviews in supporting decision-making in software engineering practice. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 24–34.

Cerny, T., Abdelfattah, A. S., Al Maruf, A., Janes, A., and Taibi, D. (2023). Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. *Journal of Systems and Software*, 206:111829.

Dmitry, N. and Manfred, S.-S. (2014). On micro-services architecture. *International Journal of Open Information Technologies*, 2(9):24–27.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2017). Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, pages 195–216.

Erl, T. (2016). *Service-oriented architecture: analysis and design for services and microservices*. Prentice Hall Press.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Information and software technology*, 106:101–121.

Geers, M. (2020). *Micro Frontends in Action*. Simon and Schuster.

Kaushik, N., Kumar, H., and Raj, V. (2024). Micro frontend based performance improvement and prediction for microservices using machine learning. *Journal of Grid Computing*, 22(2):1–26.

Kazman, R., Cai, Y., Godfrey, M. W., Pautasso, C., and Liu, A. (2023). A better way to teach software architecture. In *Software Architecture: Research Roadmaps from the Community*, pages 101–110. Springer.

Kitchenham, B. A., Budgen, D., and Brereton, P. (2015). *Evidence-based software engineering and systematic reviews*, volume 4. CRC press.

Lewis, J. and Fowler, M. (2014). Microservices a definition of this new architectural term.

Mafra, S. N., Barcelos, R. F., and Travassos, G. H. (2006). Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de software. In *Anais do XX Simpósio Brasileiro de Engenharia de Software*, pages 239–254. SBC.

Männistö, J., Tuovinen, A.-P., and Raatikainen, M. (2023). Experiences on a frameworkless micro-frontend architecture in a small organization. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pages 61–67. IEEE.

Medvidovic, N. and Taylor, R. N. (2010). Software architecture: foundations, theory, and practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 471–472.

Mezzalira, L. (2020). Micro frontends anti-patterns. InfoQ Conference Talk. Accessed: 2025-03-17.

Mezzalira, L. (2021a). *Building Micro-Frontends*. O'Reilly Media, Inc.

Mezzalira, L. (2021b). Chapter 4. discovering micro-frontend architectures. `https://www.oreilly.com/library/view/building-micro-frontends/9781492082989/ch04.html`. Book.

Mezzalira, L. (2021c). Techlead journal: #47 - micro-frontends and the socio-technical aspect. `https://techleadjournal.dev/page/16/`. Audio.

Mezzalira, L. (2023). Microfrontends anti-patterns: Seven years in the trenches.

Moraes, F., Campos, G., Almeida, N., and Affonso, F. (2024). Micro frontend-based development: Concepts, motivations, implementation principles, and an experience report. In *Proceedings of the 26th International Conference on Enterprise Information Systems*, volume 2, pages 175–184.

Pavlenko, A., Askarbekuly, N., Megha, S., and Mazzara, M. (2020). Micro-frontends: application of microservices to web front-ends. *J. Internet Serv. Inf. Secur.*, 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.

Perlin, R., Ebling, D., Maran, V., Descovi, G., and Machado, A. (2023). An approach to follow microservices principles in frontend. In *2023 IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*, pages 1–6. IEEE.

Pölöskei, I. and Bub, U. (2021). Enterprise-level migration to micro frontends in a multi-vendor environment. *Acta Polytechnica Hungarica*, 18(8):7–25.

Raimundo, J. L. P. (2023). Compositional qualities of microfrontends: The ldod archive. `https://fenix.tecnico.ulisboa.pt/downloadFile/281870113706102/49372-joao-raimundo.pdf`. Master's Thesis.

Rappl, F. (2024). Top 10 micro frontend anti-patterns. DEV Community. Accessed: 2025-03-17.

Shinde, S. (2022). 4 micro-frontend anti-patterns. `https://levelup.gitconnected.com/four-micro-frontend-anti-patterns-58aaa9fe19d5`. Blog.

Silva, N., Rodrigues, E., and Conte, T. (2025a). A Catalog of Micro Frontends Anti-patterns. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pages 616–616.

Silva, N., Rodrigues, E., and Conte, T. (2025b). Evaluating strategies for teaching micro frontends: Do anti-patterns help? In *Simpósio Brasileiro de Engenharia de Software (SBES)*, pages 522–532. SBC.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2020). Microservices anti-patterns: A taxonomy. *Microservices: Science and Engineering*, pages 111–128.

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

Valente, M. T. (2020). Engenharia de software moderna. *Princípios e Práticas para Desenvolvimento de Software com Produtividade*, 1(24).

Wohlin, C., Runeson, P., H¨ost, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.