Thematic Analysis

Anti Pattern	Question	Participant	Answers	Quotations	2nd authors' themes	1st authors' themes	Final theme
Cyclic Dependency	If you disagree, please provide a description of what is not clear	P4	The Problem is correctly explained but not as much exemplified. The example only states a dependency between A and B, but not another that would close the cycle between B and A.	The example only states a dependency between A and B, but not another that would close the cycle between B and A.	Improvements to examples	Improvements to examples	Improvements to examples
	If you disagree, please provide a description of what is not clear	P11	Alternative solutions can be considered for this problem, such as creating a single interface to access the functionalities shared between the MFEs.	creating a single interface to access the functionalities shared between the MFEs	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
	In case you have a different suggestion on the problem solution, please provide a description	P5	I agree with the solution; however, microfrontends inherently have the characteristic of being more oriented toward an organizational context (not always, but often), unlike microservices, which are more focused on solving software issues such as cost and scalability. The solution of composing microfrontends based on domain analysis can be valid in an organizational context where the responsible parties are close (or preferably the same team). In very large companies, due to the distances between teams, duplication may be a worse solution in terms of software but better in organizational terms. In highly coupled systems, creating intermediary layers can be a solution to avoid merging the MFEs, which could eventually turn these MFEs into a monolith due to the system's high coupling characteristic. So, in general, the presented solution makes technical sense, but the business context might require a solution that is not "optimal."	composing microfrontends based on domain analysis can be valid in an organizational context where the responsible parties are close	How to use the catalog	Improvements to solutions	Improvements to solutions
				creating intermediary layers can be a solution to avoid merging the MFEs	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				eventually turn these MFEs into a monolith due to the system's high coupling characteristic	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
		P7	Employing an event-driven architecture can address the issue of high coupling between MFEs without necessarily merging them into a single MFE	event-driven architecture can address the issue of high coupling between MFEs	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
		P18	maybe, instead of changing informations directly between the calculation fragment and component that shows a calculate value, its possible to use a global state or global event that "consume" this information without dependency each other	use a global state or global event that "consume" this information without dependency each other	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
		P20	It does address the problem correctly although it can have others solutions, for example a shared lib between the MFEs that can specifically handle this situation, a "bridge module" to manage dependent states/values, etc.	shared lib between the MFEs that can specifically handle this situation	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				a "bridge module" to manage dependent states/values	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
Knot Micro Frontend	If you disagree, please provide a description of what is not clear	P2	I'm a bit confused if the problem is communicating among multiple MFEs or just stablishing a communication contract. If the problem is with stablishing clear communication contracts, than the number of MFEs involved is not important to the problem definition. It's similar to implementing APIs in Microservices where you should keep a clear contract between them.	confused if the problem is communicating among multiple MFEs or just stablishing a communication contract	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
				If the problem is with stablishing clear communication contracts, than the number of MFEs involved is not important to the problem definition	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	It's related to the first question. I think the solution could state what is a good communication pattern vs a bad one. In the mentioned example, the MFE payments should be aligned with the domain level of Product, agnostic to wether it is digital or physical. Stablishing communication patterns/contracts aligned to the domain level seems a good proposal in my opinion.	solution could state what is a good communication pattern vs a bad one	Improvements to solutions	Improvements to solutions	Improvements to solutions
				Stablishing communication patterns/contracts aligned to the domain level	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
	In case you have a different suggestion on the problem solution, please provide a description	P13	I see that there are room to explain how this interface can be prepared to accommodate future changes	explain how this interface can be prepared to accommodate future changes	Improvements to solutions	Improvements to solutions	Improvements to solutions
Hub-like Dependency	If you disagree, please provide a description of what is not clear	P5	Here I have a few points: any screen, whether it is a microfrontend or not, should be resilient and have good error handling. The fallback solution will depend on many factors, such as if the error is being caused by one of the MFEs that makes the entire screen stop functioning and if it is a critical MFE that must be present on the screen. So in general, although the path presented for solving the problems makes sense, this type of issue is not specific to MFEs but rather to "hub screens" in any type of context.	any screen, whether it is a microfrontend or not, should be resilient and have good error handling	Improvements to solutions	Improvements to solutions	Improvements to solutions
				The fallback solution will depend on many factors	Improvements to solutions	Improvements to solutions	Improvements to solutions
				this type of issue is not specific to MFEs but rather to "hub screens"	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	"Avoiding screens that serve as a starting point for other functionalities is recommended" how can this be avoided? If the MFE is seen from the same perspective of a component composition, it is inevitable to have aggregators. What can be done to avoid aggregators? (and can they be avoided at all?)	how can this be avoided?	Improvements to solutions	Improvements to solutions	Improvements to solutions
				it is inevitable to have aggregators	Improvements to solutions	Improvements to solutions	Improvements to solutions
		P20	Is understandable to avoid this type of screens due to his heavy use of other MFEs, having multiple dependencies and allowing a screen to be heavier by consuming external resources to mount the screen. But not use it goes against the main idea of the MFE to be contextually segregated, that when a module is updated, there has to be no worries about the liveness of other modules that are mounted together with the module updated on that screen. Is expected due to the idea of the MFE that nothing is going to be affected and that the main change must affect and be tested in it's own context, event that is consumed in a starting-point screen.	But not use it goes against the main idea of the MFE to be contextually segregated	Improvements to solutions	Improvements to solutions	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P4	The solution solves only a part of the problem. While proper error-handling is a must in any given environment, if we are talking about front-end, it's possible for each MFE to have it's own error handling functions, that should: A - Disable the feature whenever needed; B - Warn the user of any inconveniences. A main screen would simply import everything that is already done and aggregate it on the same screen. It is also very important to make this "hub screen" a very simple one, without tasks that can compromise all the other MFEs. For instance, let's use the same example: A main banking screen that has charts, lists and balances. If this screen implements its own data fetch function that fails and renders the screen useless, then it is a problem. But if the crypto chart fails by itself, and displays its own error message while all the other ones are still functional, then its not much of an issue.	error-handling is a must in any given environment,	Improvements to solutions	Improvements to solutions	Improvements to solutions
				A main screen would simply import everything that is already done and aggregate it on the same screen.	Improvements to problem definitions	Proposal of new solutions	Proposal of new solutions
				very important to make this "hub screen" a very simple one	Improvements to solutions	Improvements to solutions	Improvements to solutions
				A main banking screen that has charts, lists and balances. If this screen implements its own data fetch function that fails and renders the screen useless, then it is a problem.	Improvements to examples	Improvements to examples	Improvements to examples
				But if the crypto chart fails by itself, and displays its own error message while all the other ones are still functional, then its not much of an issue.	Improvements to examples	Proposal of new solutions	Improvements to examples
		P8	The fallback should haven't call the issued fragments	The fallback should haven't call the issued fragments	Improvements to solutions	Improvements to solutions	Improvements to solutions
		P20	I agree that the solution is addressed to the problem but i don't think it's the best solution. Maybe segregate even more the contexts of each module so that one don't effect the other so drastically even that they share some data and states. The interdependencies must be reviewed so that they don't affect modules that didn't even was changed.	segregate even more the contexts of each module so that one don't effect the other so drastically even that they share some data and states	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
Nano Frontend	In case you have a different suggestion on the problem solution, please provide a description	P5	Just like in the first question, the solution here will depend entirely on the organizational context, but it makes sense.	the solution here will depend entirely on the organizational context	Improvements to solutions	Improvements to solutions	Improvements to solutions
		P10	I agree, but the solution can be a interface or abstraction on main domain like a super domain and the little fragments can be used like a template	I agree, but the solution can be a interface or abstraction on main domain like a super domain and the little fragments can be used like a template	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
Mega Frontend	In case you have a different suggestion on the problem solution, please provide a description	P4	The solution indeed solves the problem. But we could also look from another perspective and try to prevent the problem. In my opinion, the main issue that makes monoliths come into existence, is a lack of communication between the product team and the development team. It should be well discussed between the two teams to define when two or more features are different products.	lack of communication between the product team and the development team. It should be well discussed between the two teams to define when two or more features are different products	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions

Micro Frontends Greedy	If you disagree, please provide a description of what is not clear	P8	I can't separate this anti-pattern from nano or mega frontends	I can't separate this anti-pattern from nano or mega frontends	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P2	I don't disagree but I'd be bold and state micro frontends always are born from refactor and never from feature.	I'd be bold and state micro frontends always are born from refactor and never from feature.	Improvements to solutions	Improvements to solutions	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	1) In this case, i'd focus on the business context. Instead of conducting a comprehensive review of all existing MFEs to determine the best fit for the new functionality, it's more effective to evaluate the business context of the feature in collaboration with the relevant teams. By doing so, we can ensure that the feature is integrated into an MFE managed by the appropriate team, avoiding boundary issues and ensuring cohesive development. 2) In addition to evaluating the business context and team boundaries, it is important to consider how MFEs are named and categorized. Proper naming and handling of MFEs based on broader business domains rather than specific features can significantly simplify the decision-making process when integrating new functionalities That means, naming MFEs based on broader business domains (like "payment mfe," "user mfw," or "security mfe") as opposed to specific features (like "login mfe," "OTP mfe," etc.) can help maintain a more coherent and manageable architecture, in a way that avoids the greediness	Instead of conducting a comprehensive review of all existing MFEs to determine the best fit for the new functionality, it's more e ective to evaluate the business context of the feature in collaboration with the relevant teams.	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				Proper naming and handling of MFEs based on broader business domains rather than specific features can significantly simplify the decision-making process when integrating new functionalities	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				naming MFEs based on broader business domains (like "payment mfe," "user mfw," or "security mfe") as opposed to specific features (like "login mfe," "OTP mfe," etc.) can help maintain a more coherent and manageable architecture, in a way that avoids the greediness	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
		P13	It would be beneficial to have the domain and all responsibilities of each MFE documented and summarized. This way, when making decisions like this, the information can support whether or not to create a new MFE.	have the domain and all responsibilities of each MFE documented and summarized. This way, when making decisions like this, the information can support whether or not to create a new MFE.	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
No CI/CD	In case you have a different suggestion on the problem solution, please provide a description	P2	CI/CD should be included as definition of done of a micro frontend.	CI/CD should be included as definition of done of a micro frontend.	Improvements to solutions	Improvements to solutions	Improvements to solutions
No Versioning	In case you have a different suggestion on the problem solution, please provide a description	P5	Here, I see the following problem: versioning for micro frontends does not work as well as it does for libraries because the final build is usually unique. If we consider the Knot Micro Frontend anti-pattern, where there are versions where two screens, A and B, are present, and screen B changes its communication interface, screen A could not continue to depend on the previous version of screen B because there would be "two" screen Bs in the final build. For "smaller" dependencies, like an isolated component or a shared function, versioning might work well, but in this case, versioning might only partially solve the problem.	versioning for micro frontends does not work as well as it does for libraries because the final build is usually unique	Improvements to solutions	Improvements to solutions	Improvements to solutions
				For "smaller" dependencies, like an isolated component or a shared function, versioning might work well	Improvements to solutions	Improvements to solutions	Improvements to solutions
		P12	"It is essential to adopt the Semantic Versioning", não é essencial considerando que o versionamento semantico não é a única opção viável. Em projetos que possuem release diariamente, talvez o ideal seja usar de fato o calendar versioning.	considerando que o versionamento semantico não é a única opção viável. Em projetos que possuem release diariamente, talvez o ideal seja usar de fato o calendar versioning	Proposal of new solutions	Improvements to solutions	Improvements to solutions
Lack of Skeleton	If you disagree, please provide a description of what is not clear	P2	The solution seems to be restrict in terms of technology decisions. One of the big advantages of Micro Frontends is being able to be tech agnostic, and have different MFEs in dffierent technologies to better suit its needs. Is this still a problem if such freedom is intended? How to provide boiler plates agnostic of frameworks?	Is this still a problem if such freedom is intended?	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
				How to provide boiler plates agnostic of frameworks?	Improvements to solutions	Improvements to solutions	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P15	I add that keeping the skeleton updated is also very important	keeping the skeleton updated is also very important	Improvements to solutions	Improvements to solutions	Improvements to solutions
Common Ownership	If you disagree, please provide a description of what is not clear	P15	I don't believe software should be modularized due to team size. Naturally, larger software involves more people, but I believe small teams can also benefit from software modularization, such as separation of layers and responsibilities, observability and maintainability	Naturally, larger software involves more people, but I believe small teams can also benefit from software modularization, such as separation of layers and responsibilities, observability and maintainability	Improvements to problem definitions	Improvements to problem definitions	Improvements to problem definitions
	If you disagree, please provide a description of what is not clear	P15	Yes and No, teams with knowledge in more than one context tend to do better when solving unusual problems. In my opinion, I don't believe that the responsibility of a team that works with micro frontends should be restricted to its own context	I don't believe that the responsibility of a team that works with micro frontends should be restricted to its own context	Proposal of new solutions	Improvements to solutions	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	If the problem of a single team managing all (or a lot of) MFEs persists even with context definition, defining shared components and libraries can make boundary definition easier. Shared components reduce duplication and standardize functionalities, simplifying boundary definition. And by standradizing functionalities, it could be possible to get rid of some MFEs after some refactoring.	defining shared components and libraries can make boundary definition easier.	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				Shared components reduce duplication and standardize functionalities, simplifying boundary definition.	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
				by standradizing functionalities, it could be possible to get rid of some MFEs after some refactoring.	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
Golden Hammer	If you disagree, please provide a description of what is not clear	P11	Manter padrões de desenvolvimento comuns na organização, no meu ponto de vista é benéfica: 1- Tecnologias podem ser reutilizáveis em vários projetos. 2 - Melhoram as experiências de engenharia de software. 3 - Fornecem transparência ao design das aplicações. 4 - Acelera o processo de desenvolvimento.	Manter padrões de desenvolvimento comuns na organização, no meu ponto de vista é benéfica: 1- Tecnologias podem ser reutilizáveis em vários projetos. 2 - Melhoram as experiências de engenharia de software. 3 - Fornecem transparência ao design das aplicações. 4 - Acelera o processo de desenvolvimento.	Proposal of new solutions	Improvements to solutions	Improvements to solutions
	In case you have a different suggestion on the problem solution, please provide a description	P7	It is hard to define the right technology on stone at one point in time and having it stay the most adequate during the product evolution. To address the specific needs of each MFE, adopting a hybrid technology approach supported by a common facade, such as a Backend for Frontend (BFF) layer, and using feature flags to manage gradual migration and experimentation might be a better approach (and would also save a lot of time on decision-making processes for big companies). This allows each MFE to utilize the most suitable technology, while maintaining overall architectural coherence. Feature flags enable testing and gradual rollout and routing of new technologies without disrupting the entire system	adopting a hybrid technology approach supported by a common facade, such as a Backend for Frontend (BFF) layer, and using feature flags to manage gradual migration and experimentation might be a better approach	Proposal of new solutions	Proposal of new solutions	Proposal of new solutions
		P2	Selecting the wrong type of micro frontends based on the user needs. There are mainly two ways to integrate micro frontends: buildtime and runtime. The decision on which to adhere reflects deeply in the teams' and users' needs more than the technical pros and cons each of them offer. For example. in my experience each team dealt with its own MFE in a scilo, with the integrations happening only at the API level. Depending on other teams for frontend development in any way would only slow them down, so the runtime integration was the chosen integration strategy. If build time was chosen, they would still depend on each other for deployment even though the integration of the screens were minimum. Same thing could happen otherwise, when there is heavy integration that is safe to do it in buildtime. Choosing runtime integration strategy could generate multiple bugs and instability in different environments.	There are mainly two ways to integrate micro frontends: buildtime and runtime. The decision on which to adhere reflects deeply in the teams' and users' needs more than the technical pros and cons each of them o er	New anti-patterns	New anti-patterns	New anti-patterns
				in my experience each team dealt with its own MFE in a scilo, with the integrations happening only at the API level. Depending on other teams for frontend development in any way would only slow them down, so the runtime integration was the chosen integration strategy. If build time was chosen, they would still depend on each other for deployment even though the integration of the screens were minimum	New anti-patterns	New anti-patterns	New anti-patterns
		P3	Inconsistent User Experience Fragmented State Management Complex Inter-MFE Communication Overhead of Independent Deployments Security and Authentication Challenges	Inconsistent User Experience Fragmented State Management Complex Inter-MFE Communication Overhead of Independent Deployments Security and Authentication Challenges	New anti-patterns	New anti-patterns	New anti-patterns

Final Considerations	If you do agree, please provide a description of the problem.	P5	- Performance Bottlenecks: addressing solutions for dynamic loading can be a good topic. - Security Risks: especially in the context where micro frontends use different technologies, which means each one can introduce different types of risk.	Performance Bottlenecks: addressing solutions for dynamic loading can be a good topic	New anti-patterns	New anti-patterns	New anti-patterns
				Security Risks: especially in the context where micro frontends use different technologies, which means each one can introduce different types of risk.	New anti-patterns	New anti-patterns	New anti-patterns
		P6	Vários times pequenos cuidando de muitos mfes por vez. Imaginando um time que tenha 3 devs e mantém 5 mfes	Vários times pequenos cuidando de muitos mfes por vez. Imaginando um time que tenha 3 devs e mantém 5 mfes	New anti-patterns	New anti-patterns	New anti-patterns
		P7	1) Poor state management: Data persistence on MFEs when each frontend manages the state independently. For example, when a user navigates back to a page with different data or state, inconsistent or unexpected behavior can occur, leading to a poor user experience. This issue arises because each micro frontend typically manages its state independently,	Poor state management: Data persistence on MFEs when each frontend manages the state independently.	New anti-patterns	New anti-patterns	New anti-patterns
				For example, when a user navigates back to a page with dfferent data or state, inconsistent or unexpected behavior can occur, leading to a poor user experience. This issue arises because each micro frontend typically manages its state independently	New anti-patterns	New anti-patterns	New anti-patterns
		P10	One of the patters is related to include use more than one technology to extract more from SEO or something else, that could cause a more complex architecture. I think if you include observability overwhelmed, or patterns in observability that would be nice.	that could cause a more complex architecture	New anti-patterns	Improvements to solutions	Improvements to solutions
				I think if you include observability overwhelmed, or patterns in observability that would be nice	New anti-patterns	New anti-patterns	New anti-patterns
	How do you think this catalog would help improve the quality of micro frontend architecture in your work?	P1	This catalog can act as a checklist to ensure good practices and avoid anti-patterns in the micro frontend context	This catalog can act as a checklist to ensure good practices and avoid anti-patterns in the micro frontend context	How to use the catalog	How to use the catalog	How to use the catalog
		P2	It will serve as a guide for some DOs and DONTs that are missing in the Micro Frontends world. The term and technology is fairly new and such patterns are not well stablished in the software community yet. This is great work!	It will serve as a guide for some DOs and DONTs that are missing in the Micro Frontends world	How to use the catalog	How to use the catalog	How to use the catalog
				This is great work	Commendations for the catalog	Commendations for the catalog	Commendations for the catalog
				The term and technology is fairly new and such patterns are not well stablished in the software community yet	Commendations for the catalog	Commendations for the catalog	Commendations for the catalog
		P4	It's a well detailed "bible", containing the most essentials "don'ts" of working with micro frontends. It is a nice guide for any new or experienced developer and would definitely read it and pass it along if it is ever published.	containing the most essentials "don'ts" of working with micro frontends	Commendations for the catalog	How to use the catalog	How to use the catalog
				It is a nice guide for any new or experienced developer	Commendations for the catalog	How to use the catalog	How to use the catalog
		P5	This catalog highlights several real-world problems encountered in the day-to-day work with micro frontends. Many proposed solutions are sensible in various contexts, and even those that seem less practical from my perspective serve as valuable discussion points. These discussions can help us develop effective solutions to the identified issues.	This catalog highlights several real-world problems encountered in the day-to-day work with micro frontends	Commendations for the catalog	Commendations for the catalog	Commendations for the catalog
				Many proposed solutions are sensible in various contexts, and even those that seem less practical from my perspective serve as valuable discussion points	Commendations for the catalog	How to use the catalog	Commendations for the catalog
				These discussions can help us develop e ective solutions to the identified issues	How to use the catalog	How to use the catalog	How to use the catalog
		P6	Ajudando a perceber alguns anti padrões que por conta do dia-a-dia de trabalho se tornam "padrões" na empresa, dando uma visão geral do problema e como podemos solucionar e evitar que eles se propaguem ainda mais.	Ajudando a perceber alguns anti padrões que por conta do dia-a-dia de trabalho se tornam "padrões" na empresa	How to use the catalog	How to use the catalog	How to use the catalog
				dando uma visão geral do problema e como podemos solucionar e evitar que eles se propaguem ainda mais	How to use the catalog	How to use the catalog	How to use the catalog
		P7	As a valuable resource for training new team members and onboarding them to micro frontend projects, also to keep up to date with latest patterns and to encounter possible problems as the architecture evolves	training new team members and onboarding them to micro frontend projects	How to use the catalog	How to use the catalog	How to use the catalog
				keep up to date with latest patterns and to encounter possible problems as the architecture evolves	How to use the catalog	How to use the catalog	How to use the catalog
		P8	Creating moments with teams to discuss this catalog and sharing this knowledge to improve the frontend architecture.	discuss this catalog and sharing this knowledge to improve the frontend architecture	How to use the catalog	How to use the catalog	How to use the catalog
		P9	Ajuda a pensar nas decisões de arquitetura e na decisão de usar a arquitetura de micro frontends ou não. Vi muitos problemas que já presenciei no dia a dia, porém não era um problema identificado e por isso era apenas ignorado. Acredito que possa ajudar a identificar problemas em andamento e trabalhar na solução.	pensar nas decisões de arquitetura e na decisão de usar a arquitetura de micro frontends ou não	How to use the catalog	How to use the catalog	How to use the catalog
				Vi muitos problemas que já presenciei no dia a dia, porém não era um problema identificado e por isso era apenas ignorado	How to use the catalog	How to use the catalog	How to use the catalog
				Acredito que possa ajudar a identificar problemas em andamento e trabalhar na solução	How to use the catalog	How to use the catalog	How to use the catalog
		P10	That will be a guide to think about our MFE and communication to back-end to.	guide to think about our MFE and communication to back-end to.	How to use the catalog	How to use the catalog	How to use the catalog
		P13	It makes a perfect checklist to use when designing a new MFE project or even to review an already existing one	checklist to use when designing a new MFE project	How to use the catalog	How to use the catalog	How to use the catalog
				review an already existing one	How to use the catalog	How to use the catalog	How to use the catalog
		P14	By listing directly many issues we may find while developing micro frontend architecture, it helps to avoid such mistakes	By listing directly many issues we may find while developing micro frontend architecture, it helps to avoid such mistakes	How to use the catalog	How to use the catalog	How to use the catalog
		P15	Help mainly with mega frontends and single technology in all micro frontends	Help mainly with mega frontends and single technology in all micro frontends	How to use the catalog	How to use the catalog	How to use the catalog
		P17	I think this anti-pattern catalog is very useful for sharing information about micro frontend, both for developers who do not have experience with micro frontend and for developers with experience. Helps make decisions about when to adopt this pattern or not, and when to break into a new MFE.	sharing information about micro frontend	How to use the catalog	How to use the catalog	How to use the catalog
				both for developers who do not have experience with micro frontend and for developers with experience	How to use the catalog	How to use the catalog	How to use the catalog
		P18	I think that a catalog for micro frontend architecture can significantly enhance the quality of development, a lot of points that were addressed in this form creates a lack of efficiency and scalability in all development and maintaining process, furthermore, have a catalog like a guide before thinking in MFE and during the development is a very useful tool for a dev team	significantly enhance the quality of development	Commendations for the catalog	How to use the catalog	Commendations for the catalog
				have a catalog like a guide before thinking in MFE	Commendations for the catalog	How to use the catalog	How to use the catalog
		P20	It will help by making me more self-aware of the possible breaches that the use of MFE can make. Also, the last few questions were about the use (or not) of a MFE and where it is necessary and where it isn't, a very important point to be raised due to the "hype" that the use of a technology can have, and is not always the use case of the architecture/solution that is being made.	making me more self-aware of the possible breaches that the use of MFE can make	How to use the catalog	How to use the catalog	How to use the catalog
				the last few questions were about the use (or not) of a MFE and where it is necessary and where it isn't, a very important point to be raised due to the "hype" that the use of a technology can have	How to use the catalog	How to use the catalog	How to use the catalog
	Do you have any suggestions for improving the anti-patterns catalog?	P4	Don't know if this is an issue with Forms, but a "wall of text" is never too friendly, specially on a bright computer screen. I would use some flow charts to exemplify most of the anti-patterns and make it more readable. Or event some prototyped examples, since most of the examples uses a lot of terms such as "screens". Would be nice to see a picture of it instead of plain description.	I would use some flow charts to exemplify most of the anti-patterns and make it more readable	Improvements to the catalog	Improvements to the catalog	Improvements to the catalog
				Would be nice to see a picture of it instead of plain description	Improvements to the catalog	Improvements to the catalog	Improvements to the catalog
		P7	Addressing common challenges faced by development teams according to their experience in real world scenarios and by technology	Addressing common challenges faced by development teams according to their experience in real world scenarios and by technology	Improvements to the catalog	New anti-patterns	Improvements to the catalog
		P8	This catalog need to be shared in a website after this study have finished like refactoring.guru	shared in a website after this study have finished like refactoring.guru	Improvements to the catalog	Improvements to the catalog	Improvements to the catalog
		P9	Não, achei ótima a separação por categorias.	ótima a separação por categorias	Commendations for the catalog	Commendations for the catalog	Commendations for the catalog
		P10	Like I said, observability patters is a good start	observability patterns is a good start	New anti-patterns	New anti-patterns	New anti-patterns
		P15	I believe that the problem of managing dependencies between modules would be a good anti-pattern to solve.	managing dependencies between modules would be a good anti-pattern to solve	New anti-patterns	New anti-patterns	New anti-patterns

		P17	Maybe if there were images for some anti-pattern it would be interesting. Showing the MFEs, the communication between them, etc.	images for some anti-pattern it would be interesting	Improvements to the catalog	Improvements to the catalog	Improvements to the catalog
		P18	maybe add some diagrams and images help to understanding some examples	add some diagrams and images help to understanding some examples	Improvements to the catalog	Improvements to the catalog	Improvements to the catalog