EXPERT GUIDE

Accelerate your business with Micro-Frontends

A guide on why to split up your frontend (and when you don't want to)

Index

Albert Brand

Lead consultant at Xebia

Albert Brand works at Xebia as a Lead Consultant. He is specialised in web technology and software architecture, working for diverse customers for the last 20+ years. He is a devoted professional who has extensive experience with full stack technologies, such as cloud computing, user interface frameworks, continuous integration tools, a range of database products, and micro-frontends of course. Using frontend libraries such as React, React Native or Angular and backend technologies such as Spring, NodeJS or Firebase, he is able to create well-working software from scratch and can bring existing legacy stacks to a new level. His role as a lead consultant is to give customers new capabilities to keep ahead of the competition, as well as to increase software quality aspects in general. He wants to make any team he is part of successful and uses his extensive experience with agile methodologies to make this happen.

Micro-frontends:

Taking a Microservices approach to your frontend.

A micro-frontend is not an app or a tool. Instead, it's a term that refers to dividing your frontend into smaller pieces, enabling you to develop and deploy them independently.

Businesses increasingly split up large monolithic applications into smaller modular ones as it offers multiple benefits, like improved scalability, fewer bottlenecks in the development process, and it makes it easier to test. However, a large number of businesses think renewing their frontend from time to time, with or without a third party, is sufficient. But, is it? If you are serious about taking the next step and delivering high-quality services, micro-frontends is definitely something to consider.

Think about your backend. How businesses used to build the backend turned out to be insufficient in practice, so they started using Microservices. If you are serious about your frontend, at some point, you will also apply Microservices there -provided your organization is a good fit, which we will discuss a little later.

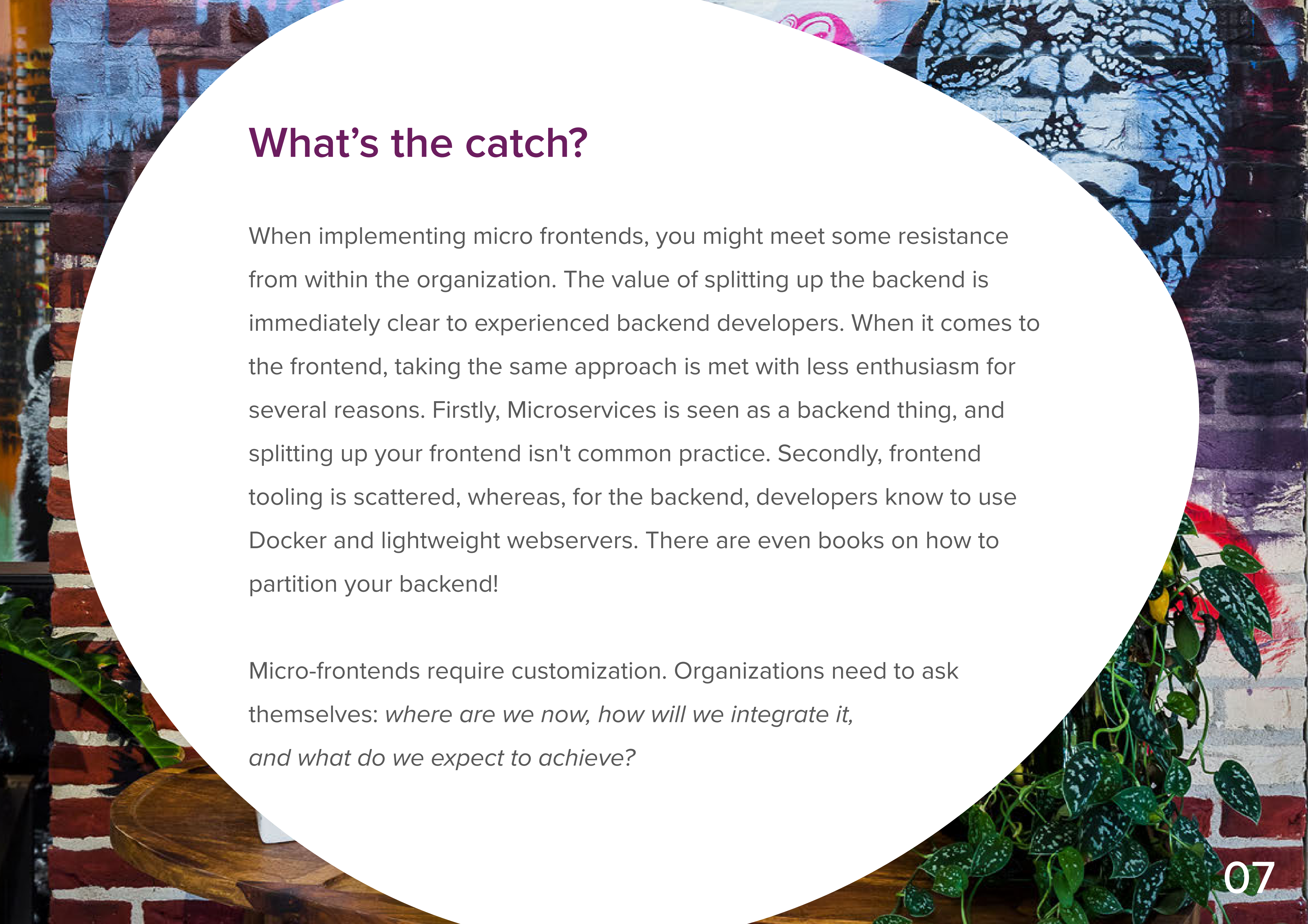
The Benefits

Your frontend is where the interaction with your end-users takes place. Customers want something from you, and you want to serve them well. You can wait for them to walk away or actively explore what struggles they are experiencing—one of the most common complaints is that a website isn't well-organized or doesn't function properly. When applications are stuck together, the frontend experience is usually far from optimal. You can overcome this by giving your teams more responsibility in a specific domain. This works best in a product-related setting, as any improvement will directly reflect in your sales figures.

“The ultimate goal is to build more autonomous teams that take responsibility for the domain they produce and raise its quality, ultimately leading to more sales.”

On that note, constant optimization and monitoring are recommended in any case, and micro-frontends provide you with the agility and flexibility to do that. Also, instantly developing frontend features enables you to quickly put new business ideas into practice, thus accelerating the business.

For instance, if your mobile app is updated with a new feature weekly, there's a greater chance of customers remaining loyal to your service than if you update every three months. But, if there are bugs in your app and it takes months to get them out of the system, that's a real problem. That can cause customers to walk away, and sometimes they will not let you know. You will have to distill that from your numbers. Customers decide to walk away for the tiniest of things, so it's important to serve them as best you can.



What’s the catch?

When implementing micro frontends, you might meet some resistance from within the organization. The value of splitting up the backend is immediately clear to experienced backend developers. When it comes to the frontend, taking the same approach is met with less enthusiasm for several reasons. Firstly, Microservices is seen as a backend thing, and splitting up your frontend isn't common practice. Secondly, frontend tooling is scattered, whereas, for the backend, developers know to use Docker and lightweight webservers. There are even books on how to partition your backend!

Micro-frontends require customization. Organizations need to ask themselves: *where are we now, how will we integrate it, and what do we expect to achieve?*

So, when should you implement micro-frontends?

The prime reason to use micro frontends is to mitigate problems with delivering new frontend due to capacity issues or other bottlenecks. If you are releasing a new version of your monolithic application every two weeks without concerns, you may not need it. The cost to implement micro-frontends is high, so a little acceleration is not worth it. But if there is a serious bottleneck, it is unmistakably worth exploring.

Checklist -

The ideal conditions to implement micro-frontends:

- If the organization understands the need to divide an application into multiple autonomous domains
- If there is currently a lot of legacy frontend code that needs to continue to work well collectively
- If teams can accept duplicated code in multiple domains
- If teams are already capable of replacing application components without downtime
- If much time is lost on updating the frontend

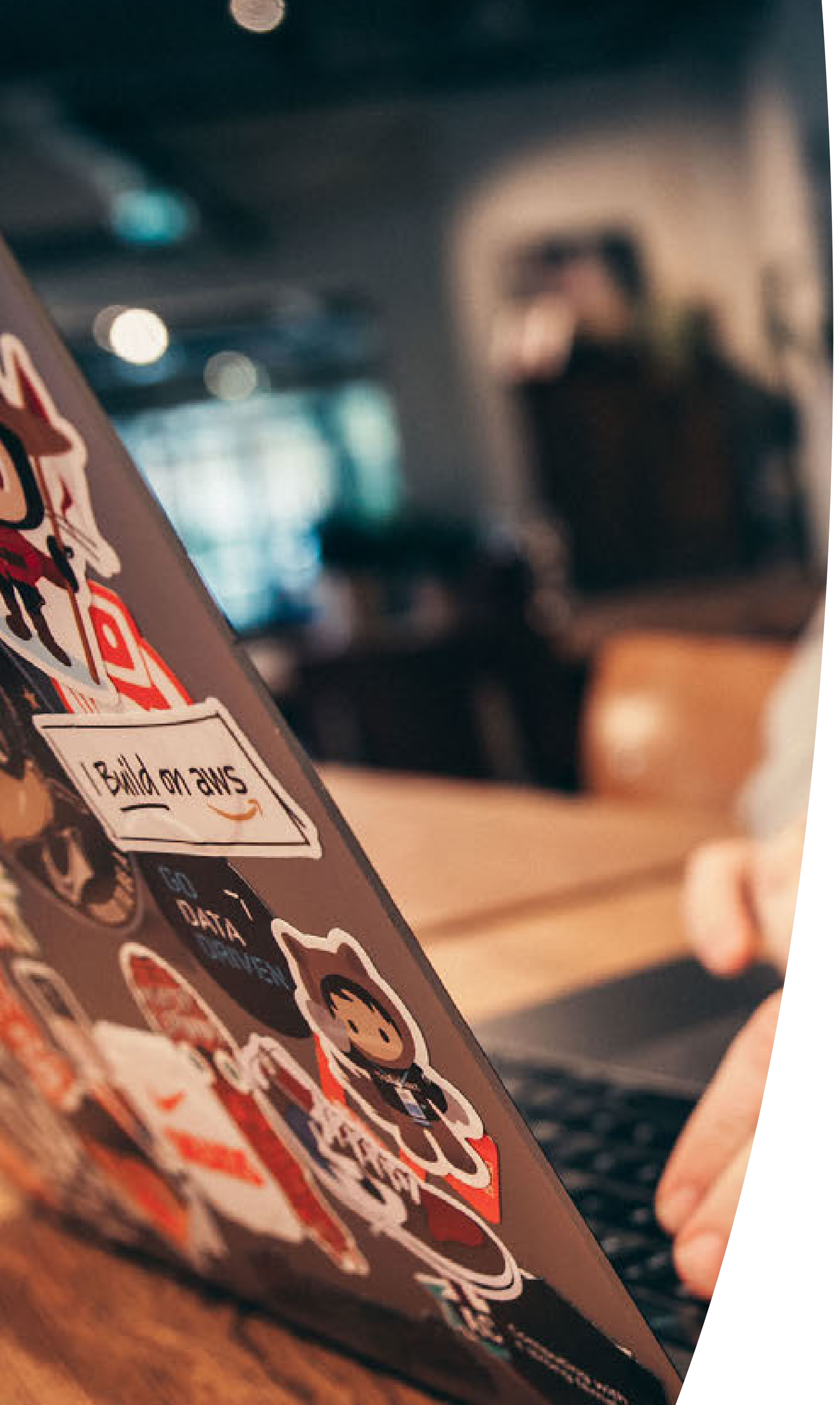
Less ideal:

A homogeneous organization that is hesitant to use new technology.

Adopting micro-frontends comes with changes an organization needs to be ready for—for instance, the restructuring of teams. If teams are functioning well and development occurs with little conflict, is it worth it? Moving from a comfortable yet traditional, layered approach to a vertical division of domains could set you up for a backlash.

Before diving in head-first, consider how your organization will respond to change and if there's a pressing issue to justify it. Finally, assess your organization's frontend knowledge. If this isn't sufficient and you are using an off-the-shelf product, we don't recommend implementing micro-frontends.

If you are using backend Microservices and are past the teething troubles, now is a good time to apply it to your frontend.



What type of applications benefits most from micro-frontends?

Micro-frontends apply to every kind of application with a frontend. The web is very much suited for it, though, as the dynamic nature of loading the frontend is the norm, tied into the platform.

It's also applicable to mobile or desktop applications. However, you will have to find ways to bring the different micro-frontend codes together in an earlier stage.

The absolute sweet spot:

large organizations with multiple teams working on one web application.

Spoiler

Accelerating your business with micro-frontends is not that easy

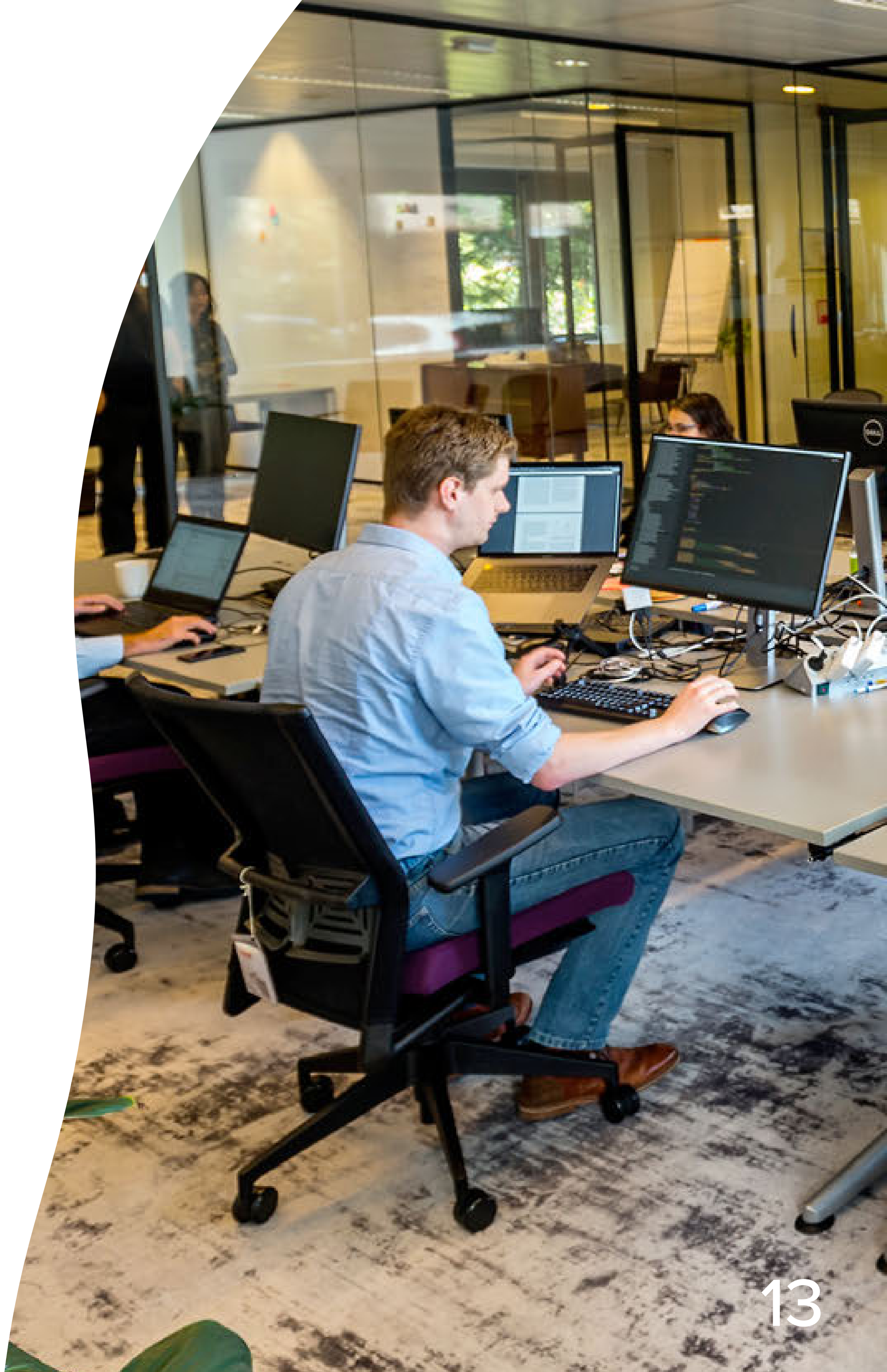
An often-heard complaint is that development is slowing down the business. This is usually due to the increased complexity of applications. One way to reduce this complexity is by dividing and conquering. This starts with: where do I put the right slice? It takes time to master this, and you will not get it right the first time!

The organization must be willing to learn, to make mistakes, and come back from them. Fortunately, we have a lot of experience in this field, and we can help you.

“You have to reorganize the application to reduce complexity and create simple systems that are easy to maintain and modify.”

Even if you reduce your systems' size, you will still have to deal with complex subdomains. To manage them properly and ensure they are valuable, you will need to put the effort in. If you find development is slowing down at any point, you can keep slicing, which will allow you to accelerate in the short-term. However, it's important to ask yourself if this is the right thing to do, as you might be creating more waste (in the long-term).

Waste is often the result of poor collaboration between multiple teams, causing them to wait for each other: the more teams, the more communication lines, the more waste. By giving teams more focus and reducing the amount of communication between them, you can improve this. It sounds crazy, but you want less dependency, less bureaucracy, less waiting time, and more autonomy.



The Up-Side

Reducing complexity and waste allows teams to work more autonomously and efficiently. They no longer have to wait for other teams, which drastically reduces time-to-market. The organization becomes agile and can respond to changing circumstances.

Working with mini-applications can really accelerate your business. Especially if your micro-frontend is completely isolated, you can deliver a super optimized frontend that works much faster than a huge CMS that is not optimized. The knife cuts both ways - making it well worth the consideration.

Next steps to find out more!

Some good resources to read about the technology behind micro frontends:

- https://micro-frontends.org/ -
  somewhat outdated information, but the concepts and considerations are clearly explained.

- https://martinfowler.com/articles/micro-frontends.html -
  Not written by Martin himself, but this also explains what tools are available and the intended benefits.

If you want to explore whether your organization is ready for micro-frontends, we are happy to help!

Joseph Gouriye

Business Manager,
Xebia Software Development

PHONE: +31 6 24 17 38 48
EMAIL: jgouriye@xebia.com

Xebia