Challenges

In this lesson, we'll discuss some challenges pertaining to frontend integration.

WE'LL COVER THE FOLLOWING

Frontend integration means that the frontend is composed of different systems. This causes some challenges.

UI infrastructure

For UI integration to work, some infrastructure has to be provided. That might be common CSS or JavaScript code, but it can also include server infrastructure for server-side transclusion.

There could be a need for **UI parts that do not belong to any specific microservices** – for example, the home page or a navigation bar. These have to be developed and maintained.

It is important to **not put too much into the UI infrastructure**. If the microservice relies too much on the generic UI infrastructure, it will depend heavily on it and that contradicts the goal of independent development.

UI integration leads to a certain degree of dependencies on the code level, which **leads to tight coupling**; therefore, the dependencies should be limited.

For example, it makes little sense to provide all the styling, UI frameworks, and UI code for all microservices in one single component.

This would be a problem in particular for migration to new technology stacks.

The stack is the same for all microservices, and it is, therefore, hard to migrate

stepwise to a new stack.

Uniform look and feel

For example, it is not easy to achieve a uniform look, feel, and design of the overall application.

A uniform look and feel is usually associated with the sharing of artifacts. Multiple microservices must share the CSS, fonts, or JavaScript code required to implement the design.

Interfaces in frontend integration

Transclusion generates a subtle type of interface definition. Normally, an interface is defined by data types and operations. This is obviously not the case for frontend integration. Still, there is a kind of interface definition.

For example, HTML code has to integrate itself into another page if it is displayed in another frontend. To do this, it may be necessary to have common CSS classes, and so all the frontends must possess the same CSS selectors. If JavaScript is used in the displayed HTML, the common JavaScript code and the JavaScript libraries used must be available in the other web pages.

Overall, these requirements form a kind of interface definition that ensures that HTML can actually be displayed. When only links or redirects are used, this challenge does not exist; only the URL must be known. The linked page can use completely different CSS and JavaScript.

Transclusion therefore couples the systems more strongly than links do.

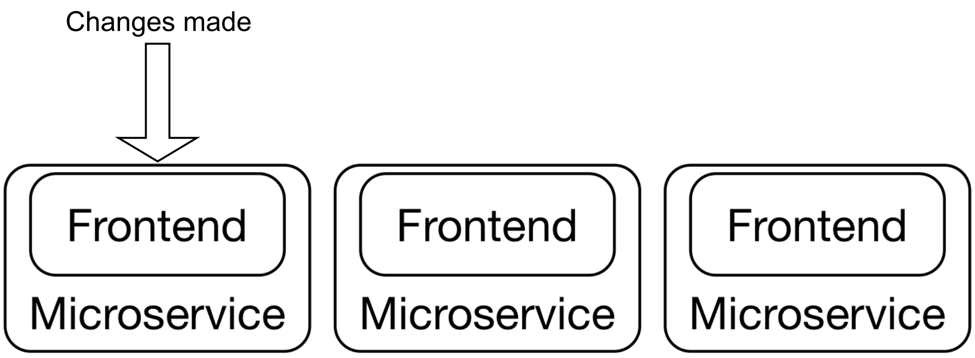
UI changes impact multiple modules

If the changes to a system typically only require changes to the UI, this can be more complicated with frontend modularization.

The code for the UI is distributed across the different frontends, so that a change means that all frontends have to be modified. Therefore, if the UI is modularization can increase the effort.

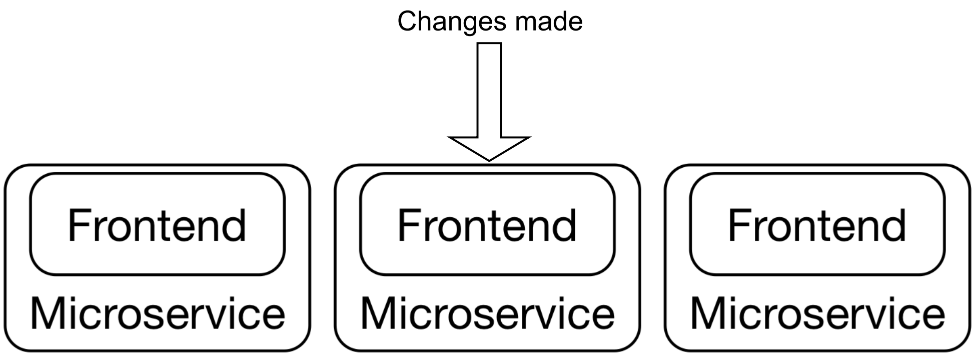
However, frontend modularization has an advantage, even in the case of such changes. The changes can be made step by step by changing only one frontend at a time, **thus minimizing the associated risk**.

In addition, changes that take place only in the UI should be far less frequent than domain-based changes that affect all layers. So, while it is harder to **change all of the UI**, those kinds of changes **should not be happening too often**. Making UI-based harder and making more frequent domain-based changes easier at the same time seems like a good trade-off.



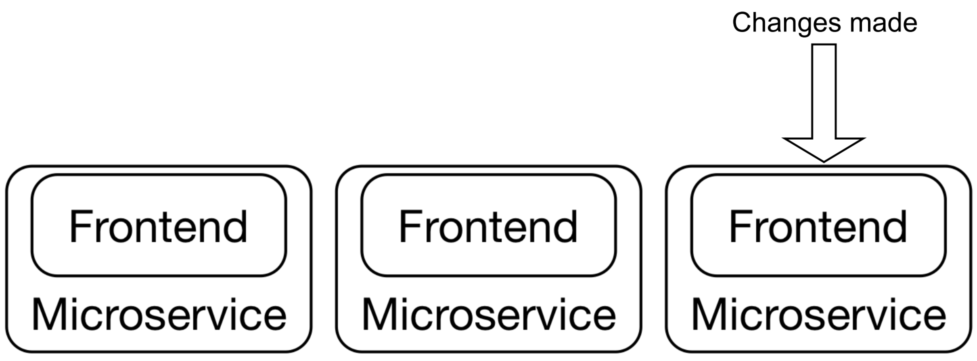
Changes can be made step-by-step which is safer.

1 of 4



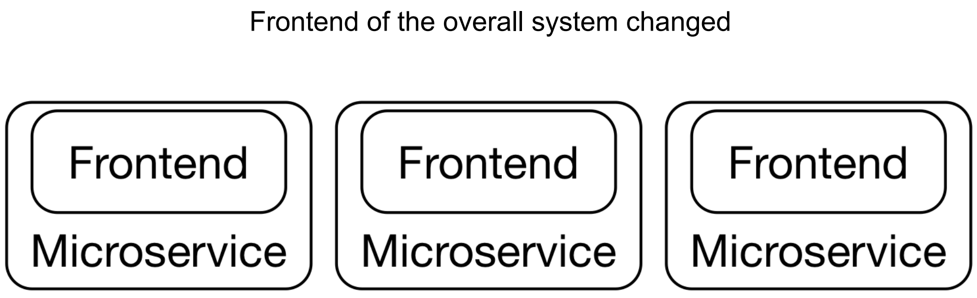
Changes can be made step-by-step which is safer.

2 of 4



Changes can be made step-by-step which is safer.

3 of 4



Changes can be made step-by-step which is safer.

4 of 4

QUIZ

Q

Suppose the management of a website have asked the developers to implement two changes:

1. Migrate to a microservices system with modularized UI.
2. A complete redesign of the UI.

Which change should the developers make first if they want to save effort?

COMPLETED 0%

1 of 1

In the next lesson, we’ll discuss the benefits of frontend integration.