Integration in Assurance App

In this lesson, we'll discuss the details of how the application we saw in the last chapter was integrated.

We'll cover the following

- Why monolithic backend?
- Integration with redirects
- Integration with links

Why monolithic backend?

This application uses a monolithic backend and frontend microservices because the **microservices** lack logic and they **are not self-contained systems**.

However, this architecture can still make sense. The frontend, at least, consists of microservices, and so **independent development of the microservices is possible**.

Also, it is possible to **use different technologies in each frontend microservice**. With a large number of available UI frameworks and the high speed of innovation, that is a clear advantage.

Also, it is probably not possible to migrate the backend into microservices. Another team might be responsible for it. Therefore, if the scope of the project is just to improve the frontend, there is no way to change the architecture of the backend.

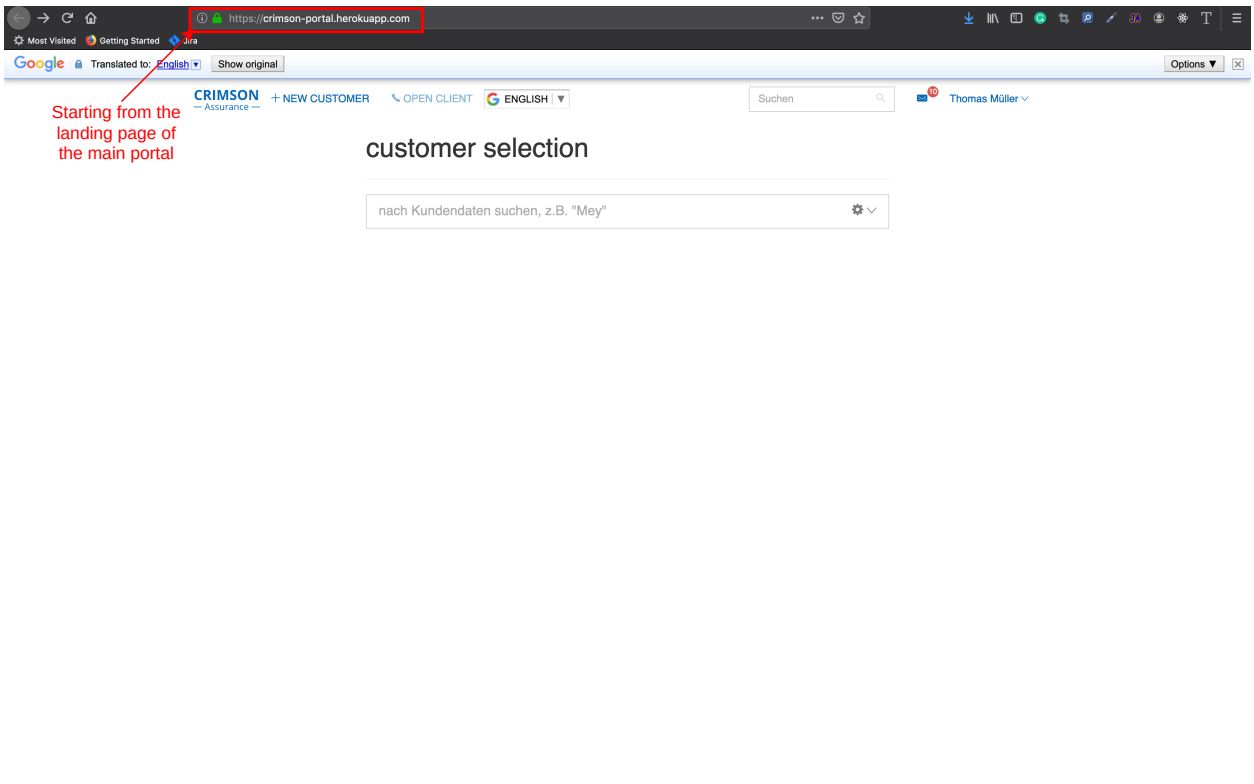
While SCSs are generally a great idea, this example shows one exception to the rule.

Integration with redirects

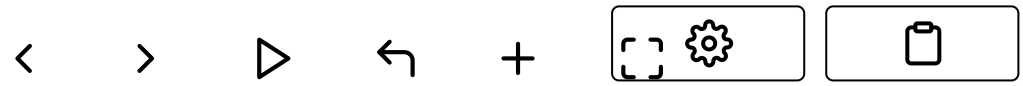
If a user in the *damage* application enters a claim for a car, the user is sent back to the overview of the car displayed on the portal. The transition from the *damage* application to the portal is implemented with a redirect. The *damage* application sends an HTTP redirect after reporting the claim, leading to the web page of the *main* application.

A redirect is a very simple integration. The *damage* application needs to know only the URL for the redirect. The portal could even pass this URL to the *damage* application to further decouple the two applications.

Such an integration is also used if a user registers with their Google account on a web page. After the user agrees to register on the Google web page, the Google page sends a redirect back to the initial web page.



Integration between the damage app and the main portal app with redirects



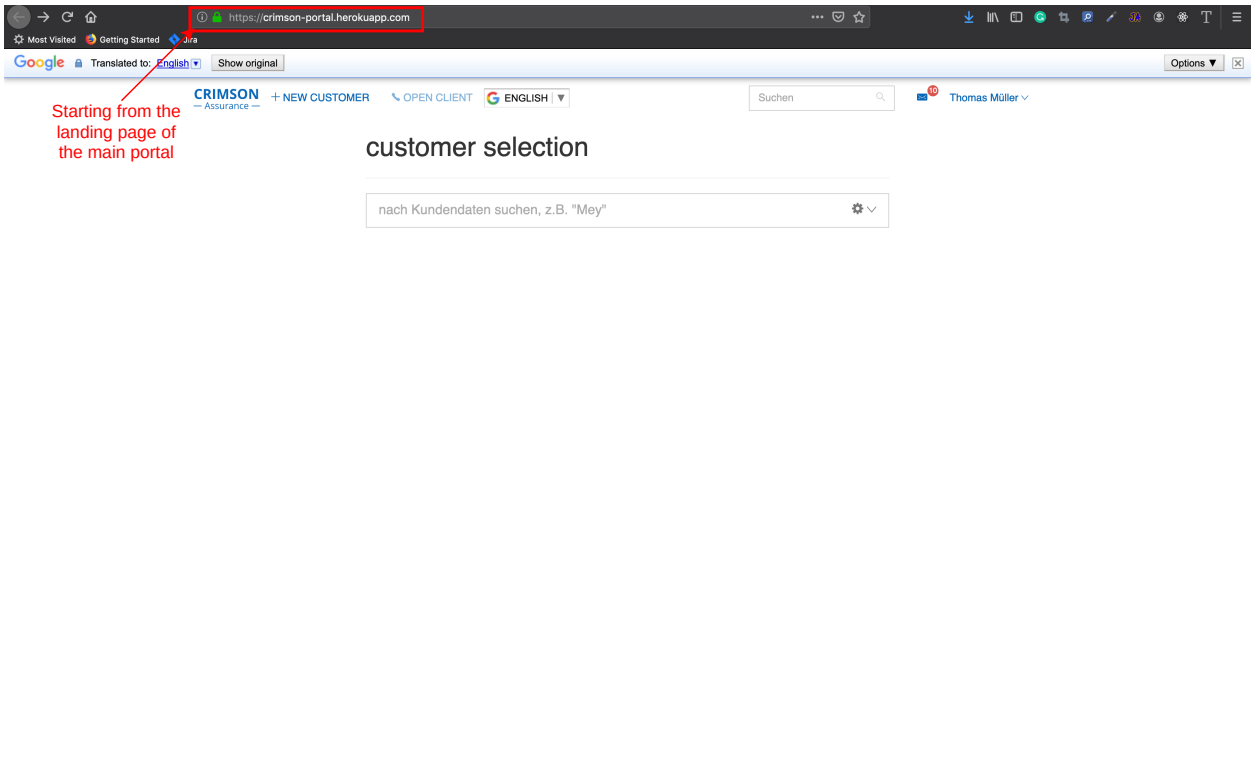
Integration with links

The integration of the applications is mainly done via links such as https://crimson-letter.herokuapp.com/template?contractId=996315077&partnerId=4711 (https://crimson-letter.herokuapp.com/template?contractId=996315077&partnerId=4711) to display a web page for writing a letter.

They contain all the essential information necessary for the web page to write the letter: **the contract number and the partner number**.

In this way, the *letter* application can retrieve the data from the backend simulator and render it in the web page. The coupling between the main application and the letter application is very loose; it's just a link with two parameters. The main application does not need to know what is behind the link. As a result, the *letter* application can change its UI at any time without any impact on the *portal* application.

However, all applications use a common database from the backend simulator and are consequently tightly coupled, because a change to the data would affect the backend and the respective microservice



Integration between the letter app and the main portal app with links

1 of 8

In the next lesson, we'll look at other ways that apps can be integrated on the client-side integration.

← Back

Overview

Next →

Other Integration Methods

Mark as Completed

Report an Issue