Chapter 1

Front-End

This chapter discusses the front-end. We start by presenting the technology stack in section 1. After that we describe the most important control flows in the front-end in the section Design.

1 Technology Stack

2 Design

In this section we will discuss the control flow in the front end upon certain actions of the user. To avoid a lengthy report we will only discuss the flow of successful cases.

AJAX calls with shadows!

2.1 Log In

As mentioned earlier users have to log in to a social network before being allowed to stalk somebody on that network. Since we store the ID of the user and we have not implemented an api call that adds for example the LinkedIn identification to a user that we have already stored with its Facebook identification.

Figure 1.1 presents the flow of control when a users logs in.

The `loggedIn`, and its not shown equivalent `loggedOut`, event ensure that the `searchController` knows that a user is logged in. The `Facebook Service`, `ngFacebook`, is discussed **??** on page ??.

Logging out of social network is comparable to logging into it. Figure 1.1 uses Facebook, the application works in exactly the same way for e.g. LinkedIn.

2.2 Start Stalking

Figure 1.2 shows what happens in the front end when the indicates that he has logged in into all social networks he wishes to use.

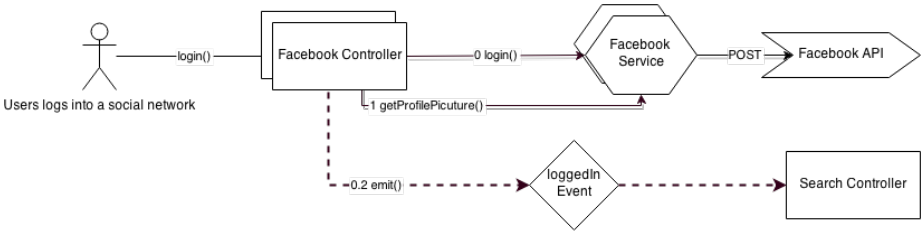

Figure 1.1: A schematic overview of the control flow when a user logs in.

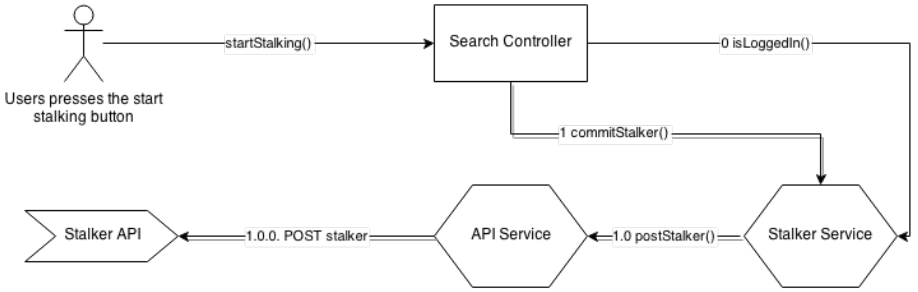

Figure 1.2: A schematic overview of the control flow when a presses the 'start stalking' button.

If the POST to the `Stalker API` fails no error is reported to the user, since it does not matter to the user if his information is added to our database.

2.3 Stalk

Figure 1.3 presents the flow of control when a user searches for somebody on the social networks that he has logged in to.

The `Stalker Service` contains all information on the stalker. The `API Service` contains all calls to our own API, we have chosen for a separate service so that we only have to store the address of the API in one place. The Search Service broadcasts the search to the different social media controllers, in Figure 1.3 Facebook is shown as an example. Each social network controller does a search call by calling some function in the matching service that interacts with the actual API of that social network.

We had to use a broadcast instead of an emit as in section 2.1 since we had to down in the scope hierarchy instead of up.

We added the start stalking button to force our users to first login to all social networks they wanted to use before actually stalking somebody. This was one of the concessions we had to make due to time constraints, since this button ensures that we can post complete stalkers, instead of having to update a stalker every time he adds a social network.

Add setMostRecentSearch() to the figure.

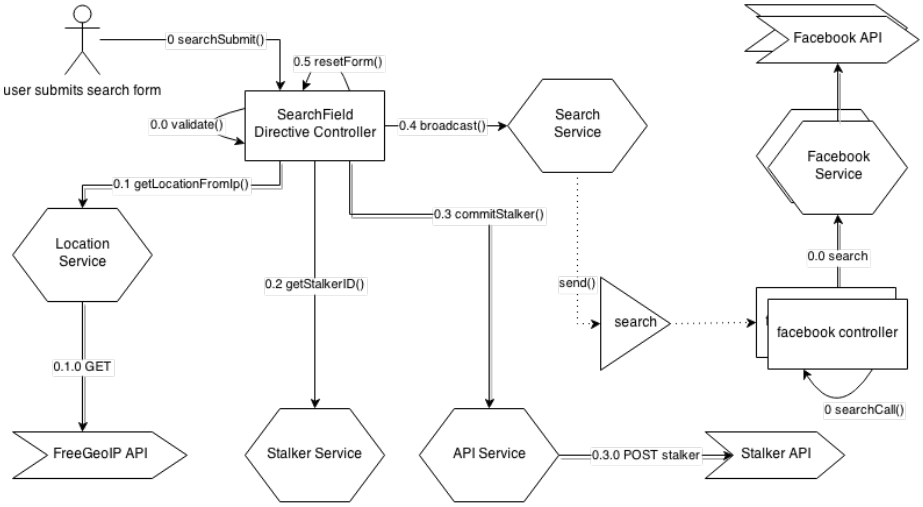

Figure 1.3: A schematic overview of the control flow when a user stalks somebody.

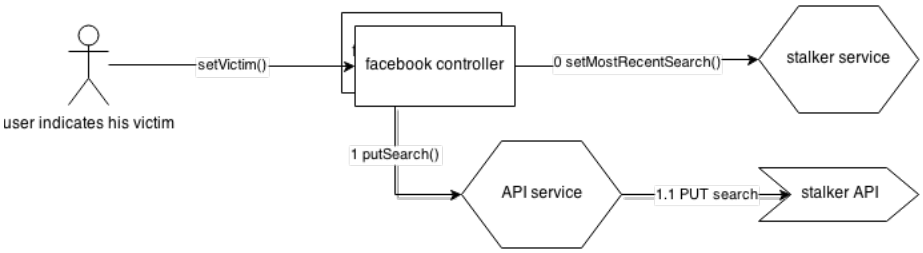

Figure 1.4: A schematic overview of the control flow when a user selects somebody as his victim.

2.4 Select Victim

Figure 1.4 shows the control flow in the front end when a user indicates that one of the found results was his victim. Even tough it does not matter to the user if the addition of his victim to our database has been reported we report success, and failure, since the user has no feedback on his actions otherwise. Once again we have chosen Facebook as an example, but the flow of control would be the same with a different social network.

2.5 View Statistics

The control flow when a user views statistics is presented in Figure 1.5.

Begeidend verhaaltje schrijven

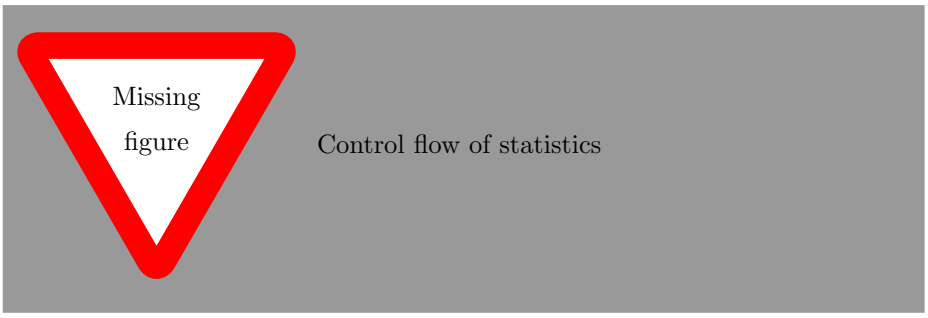

Figure 1.5: A schematic overview of the control flow of requesting the statistics and showing them to the user.