SOFTWARE DESIGN

S3-INDIVUDUAL

Welles,Nick N.H.M.

2022 - 2023

Inhoudsopgave

[User stories 2](#_Toc121823617)

[Concept diagram 3](#_Toc121823618)

[Asynchronous functionality 3](#_Toc121823619)

[Async in software 3](#_Toc121823620)

[Messaging and events 4](#_Toc121823621)

[What is messaging and events 4](#_Toc121823622)

[Messaging and events in my project. 5](#_Toc121823623)

[Software architecture 6](#_Toc121823624)

[System context diagram 6](#_Toc121823625)

[Container diagram (Web shop system) 6](#_Toc121823626)

[Component diagram (API) 7](#_Toc121823627)

[Component diagram (Single page) 8](#_Toc121823628)

# User stories

Here are a number of user stories that have been formulated to help me verify the functionality of the Instruweb application. The user stories are in a table which are numbered by functional- and non-functional requirements.

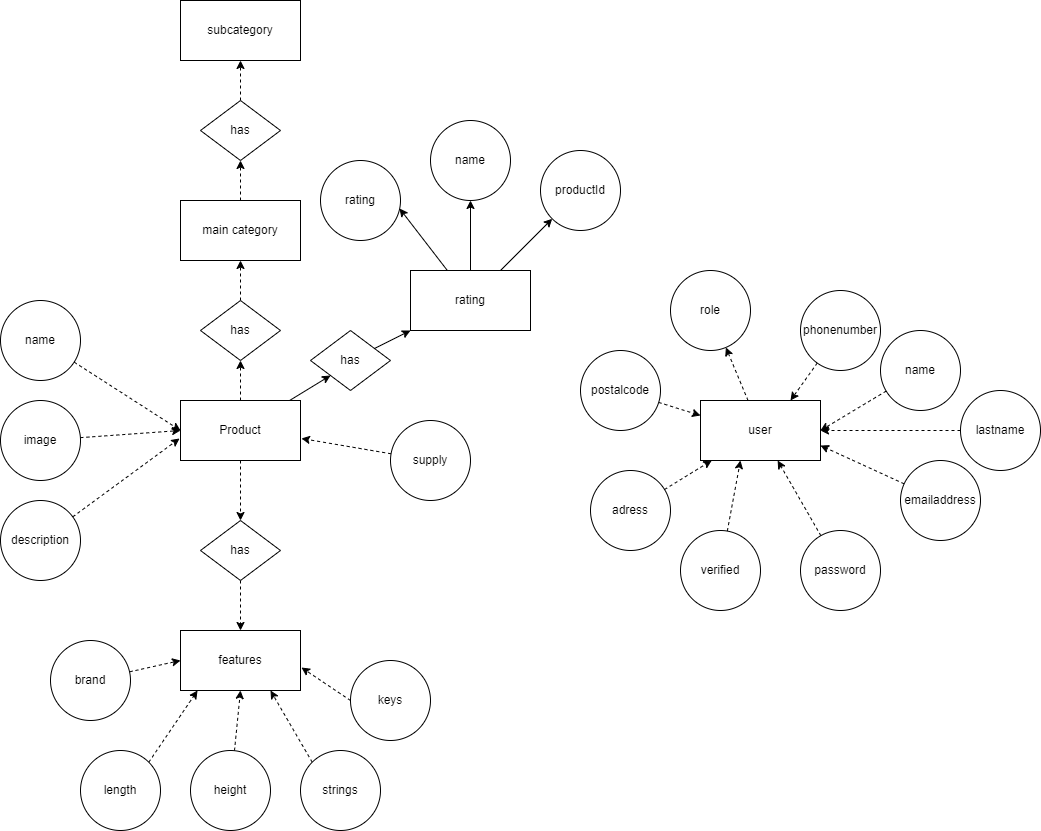
**FR** stands for functional requirement.  
**NF** stands for non-functional requirement.

|  |  |
| --- | --- |
|  | Functional |
| FR-01 | As a user I want to be able to select categories so I can find a potential desired product(s). |
| FR-02 | As a user I want to be able to add one or multiple items to my shopping cart so I can order my desired products. |
| FR-03 | As a user, I want to be able to easily remove one or more products from my shopping cart. |
| FR-04 | As a user, I want to be able to create an account so that my data is saved for a subsequent order. |
| FR-05 | As an administrator, I want to be able to easily create user and administrator accounts so that selecting roles remains under administrators. |
| FR-06 | As an administrator, I want to be able to modify products in the web shop so that the information and price remain up-to-date. |
| FR-07 | As a user, I want to be able to check out the products in my cart so that the order is initiated. |
| FR-08 | As a user, I want to be able to update my account information so that my information can be kept up-to-date. |
| FR-09 | As a user, I want to be able to search for products all over the site so that I can find my desired product. |
| FR-10 | As a user, I want to be able to filter by certain characteristics in the selected category so that I can more easily find the product I am looking for. |
| FR-11 | As a user, I want to see related products on the product page so that I can order something with them if necessary. |
| FR-12 | As a developer, I want a product to be able to contain multiple categories so that searching for particular products is simplified. |
| FR-13 | As a user, I want to be able to log in so I can update my account information. |
| FR-14 | As a user, I want to be able to register so I can update my account information. |

|  |  |
| --- | --- |
|  | Non-functional |
| NF-01 | As a user I want to be able to view the website on every device whenever I want so I can purchase products everywhere. |
| NF-02 | As a software developer I want the application to be maintainable and scalable so I can add functionality to it in the future. |
| NF-03 | As a software developer I want the front-end and backend communicating via an API. |
| NF-04 | As a user I want the website to load within 1 seconds so I won’t visit another website. |

# Concept diagram

To make it easier on myself, I took a chart from semester 2 into this. The concept diagram contains simple entities which I have all given a property. For me, this clarifies the picture of how I can possibly build another diagram (think ERD, ERM etc) to then make my project more complete.



In a web shop the most important entities are the user and the product. The product contains all the information to make it easy to search one or select one via the categories the product has. The user can register for an account so they don’t have to fill in their billing information again since their information will be saved.

Since this is a concept diagram it has never really been completely finished.

# Asynchronous functionality

Many everyday things you already do asynchronously. For example, with the washing machine: You're not going to wait 4 hours for the washing machine to finish and you can hang up your laundry, are you? No, in the meantime you will do other things while the washing machine continues to run. That's asynchronous communication.

## Async in software

Using an async function in your application has all sorts of advantages. Suppose you are expecting data from a data source, your application can run other code while this function is fetching the data. Async functions can contain zero or more await expressions. Await expressions make promise-returning functions behave as though they're synchronous by suspending execution until the returned promise is fulfilled or rejected. The resolved value of the promise is treated as the return value of the await expression. Use of async and await enables the use of ordinary try / catch blocks around asynchronous code.

In my own project, I also used async functions. When waiting for the SSO service. See the image below.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

And in the Angular initializer function:



I also use an async function when retrieving products:  
This is the product service in the front-end

Afbeelding met tekst

Automatisch gegenereerde beschrijving

And the function in the Angular product component:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

## Messaging and events

### What is messaging and events

In software development, messaging and events refer to the communication and interaction between different software components or systems. Messaging involves the exchange of structured data between components, often using a messaging protocol or messaging service. This allows the components to communicate and exchange information in a decoupled and asynchronous manner. Events, on the other hand, refer to specific occurrences or actions that happen within a system, such as a user clicking on a button or a file being saved. These events can trigger a response or action from other components within the system, allowing for more complex and dynamic behaviour.

Source: <https://en.wikipedia.org/wiki/Event-driven_messaging>

### Messaging and events in my project.

The async section in this document also contains a code snippet with a .subscribe function. So this function is subscribed to another function from another component at all times. Once it passes new data then the rest of the function can be executed, or an error can be displayed. If you can imagine yourself subscribing to a newsletter, every time there is a new newsletter, they will send it to your home (the method inside subscribe gets called).

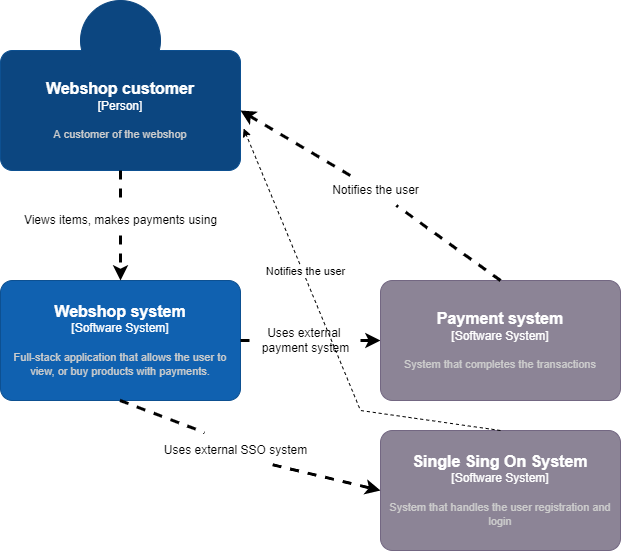
In the picture below is an example of the waiting of data we request from the user service. First from keycloak (SSO service) we retrieve the profile of the user, and then we are going to see if it already exists in our database. So for this we retrieve the user with the username we get from keycloak. If it exists, the console logs that the user already exists, if not, another function adds it to the database.  
if res (the result of the subscribe) contains an error, or it is empty, the subscribe function passes to the error method below, where also the same thing happens, but an error message is returned.

Afbeelding met tekst

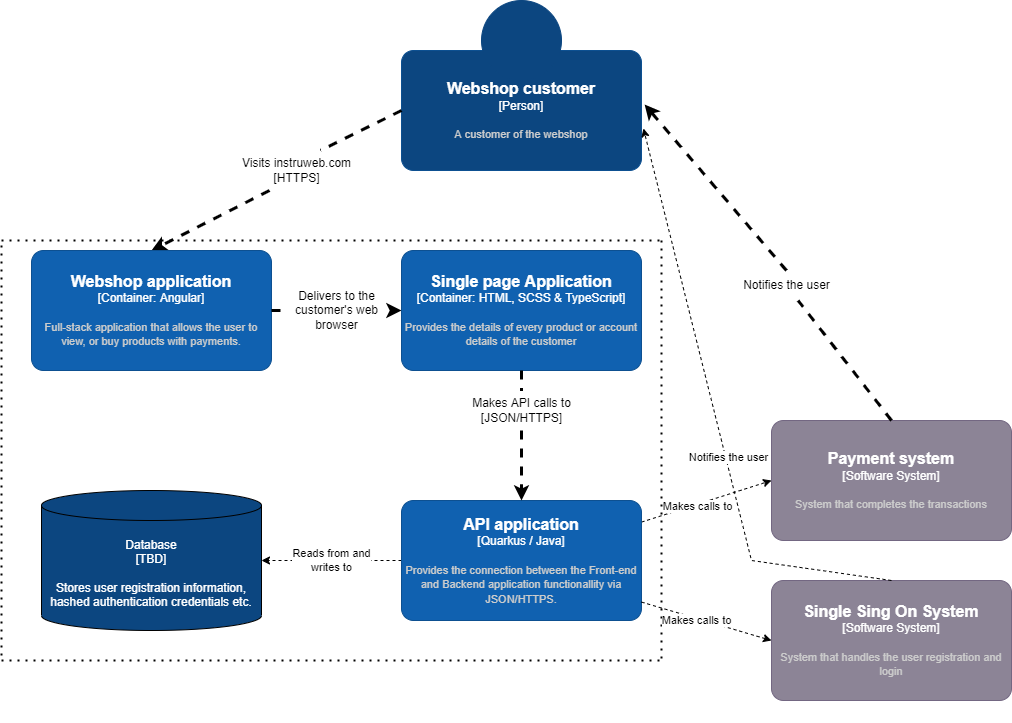
Automatisch gegenereerde beschrijving

# Software architecture

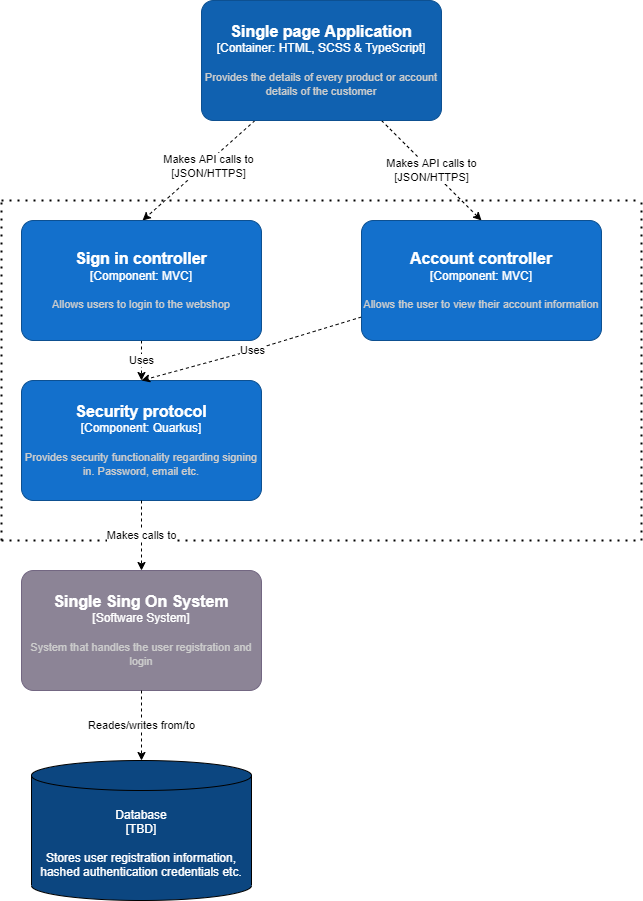
## System context diagram



## Container diagram (Web shop system)



## Component diagram (API)



## Component diagram (Single page)

