**Software design**

Student: Jorn Kersten

Student number: 483331

Date: 11-01-2023

Semester: S-DB-IPS3-S3-DB03

Table of contents

[User stories 3](#_Toc124344223)

[Software architecture 4](#_Toc124344224)

[Concept diagram 4](#_Toc124344225)

[C4 model 5](#_Toc124344226)

[System context diagram 5](#_Toc124344227)

[Container diagram 5](#_Toc124344228)

[Component diagram 6](#_Toc124344229)

[Asynchronous functionality 7](#_Toc124344230)

[Async in software 7](#_Toc124344231)

[Messaging and events 7](#_Toc124344232)

[What is messaging and events 7](#_Toc124344233)

[My use of messaging and events 8](#_Toc124344234)

# User stories

Before I started programming, I first put on paper what the user stories about my project were. Below you can see the user stories divided into functional and non-functional requirements.

**FR** Functional Requirement

**NF** Non-Functional requirement

**FR-01** As a user I want to be able to see if a products price has increased or decreased because I want to see price differences.

**FR-02** As a user I want to be able to be able to see one specific supermarket so I can see the prices of products of one specific supermarket.

**FR-03** As a user I want to be able to create an account so I’m able to keep track of products.

**FR-04** As a user I want to be able to login and see a list of my specified products, so I have a clear overview in products I want to see.

**FR-05** As a user I want to be able to create a shopping list with my purchased products on it so I’m able to see what I purchased.

**FR-06** As a user I want to be able to compare my shopping list to the one before so I can see where I’m spending more.

**FR-07** As a user I want to be able to see the difference in prices of products on my shopping list so I have an insight in where I end up spending more compared to different weeks.

**FR-08** As a user I want to be able to compare products from different supermarkets. That way I’m able to see where the products I want to purchase are the cheapest.

**FR-09** As a user I want to be able to see a clear indicator for price changes so I have a clear overview in what products got cheaper of more expensive.

**NF-01** As a user I want the website to be responsive on mobile, tablet and desktop so I can see the prices every ware I go.

**NF-02** As a admin I want the website to be maintainable so there is not a lot of maintenance needed for the website.

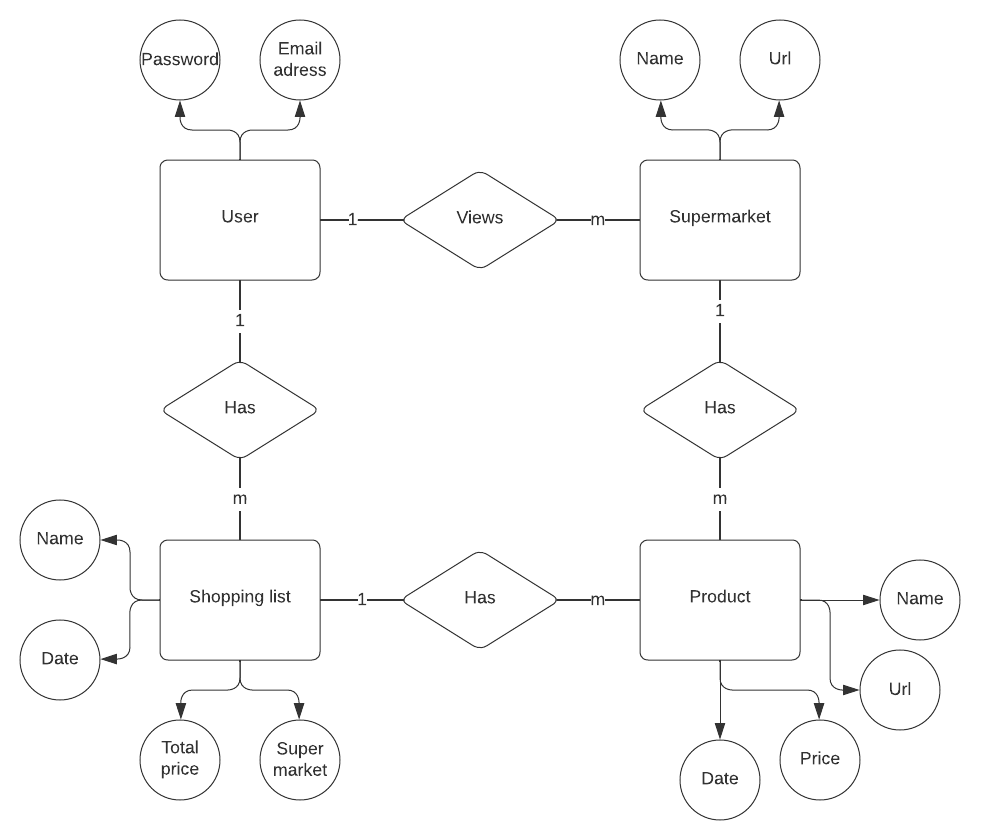
**NF-03** As a developer I want the website to be scalable so it’s easy to add more supermarkets.

**NF-04** As the product owner I want the website to load in 2 seconds so users don’t leave the website.

# Software architecture

## Concept diagram

To give myself a slightly better idea of how that interacts with the classes in the application, I first created a concept diagram. From this I could more easily get a picture for myself of how everything communicates what I want to make and how I want to make it. The concept diagram can be seen in the image below.



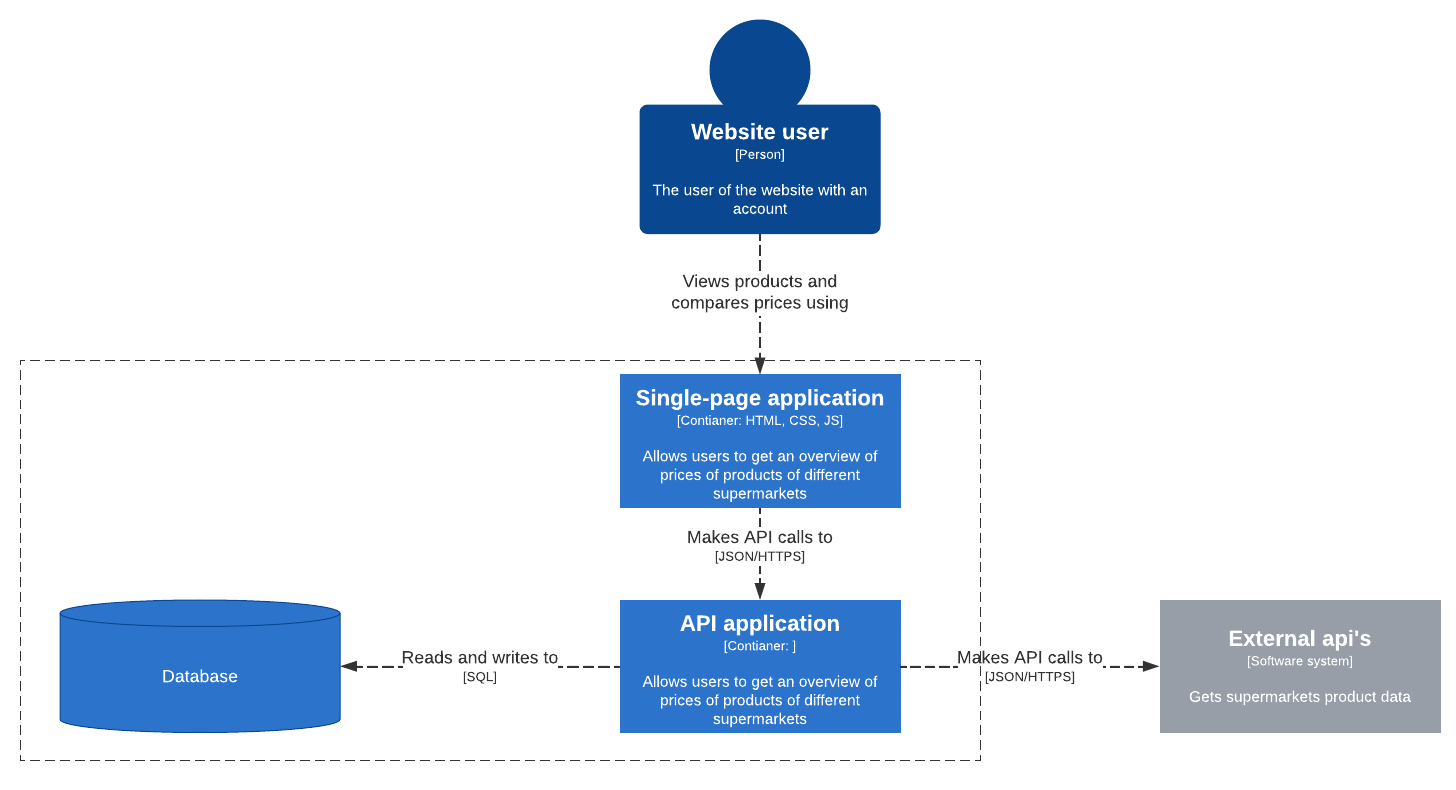
## C4 model

Besides a concept diagram, I also created a C4 model to get an idea of how the application communicates. This also gives you a better idea of how the components in the application communicate more with each other, which can be seen in the three images below.

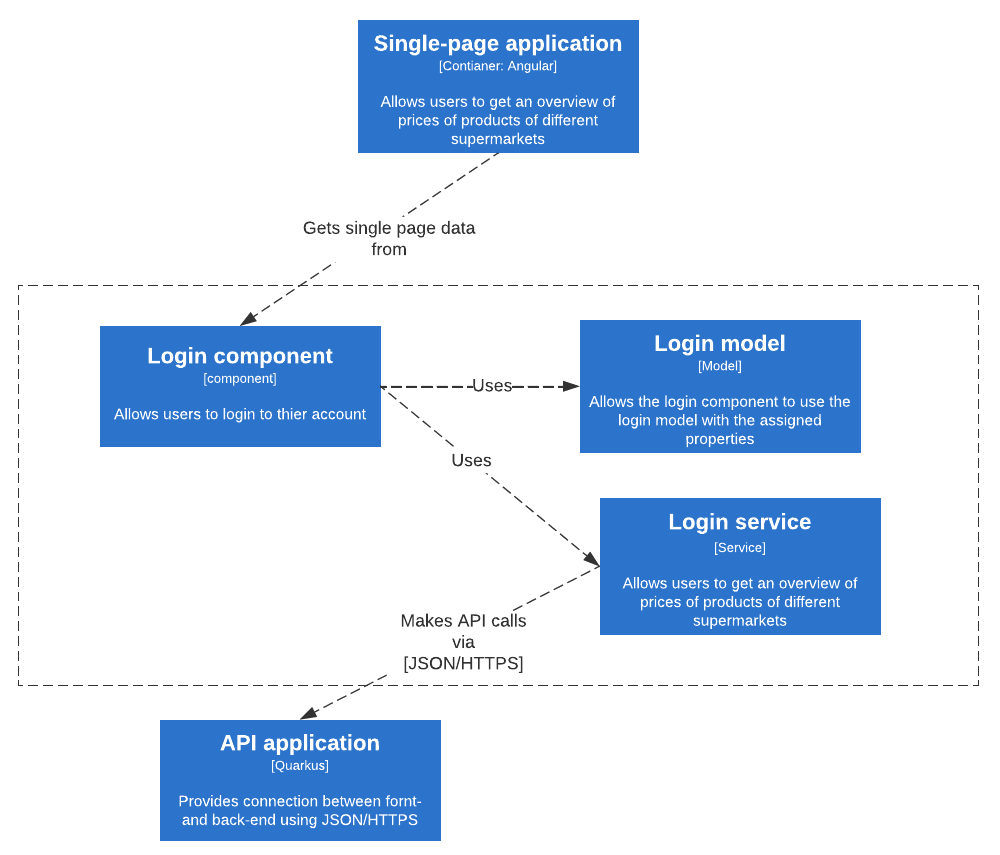
### System context diagram



### Container diagram



### Component diagram



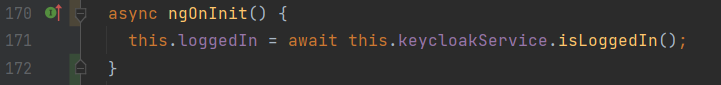
# Asynchronous functionality

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

An async function is a function that returns a promise, which represents the eventual outcome of the function. The function can continue to execute and return control to the program while other code is executing or while the function is waiting for a response from an external resource. The async keyword allows you to define an asynchronous function and is used before the function declaration.

The await keyword is used within an async function to "wait" for a promise to resolve without blocking the execution of the rest of the program. It is used before a promise-based expression. When await is used in an async function, any code waits for the promise to resolve and then resumes the execution of the function. The resolved value of the promise is then returned. 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 to catch any errors that may occur.

In the image below you see a code snippet. Here the program checks if the user is logged in, the program has to receive this data from an external source which may take a bit longer to return data from. That is why it’s put into a await.

async and await can be used for many situations, for example: retrieving data from an external source and retrieving data from a database. In addition, it is easier to catch problems if something goes wrong then.

## Messaging and events

### What is messaging and events

In software development, messaging and events are two related but distinct concepts that are often used to facilitate communication between different components of a system.

Messaging refers to the process of sending and receiving messages between different parts of a system. A message typically contains some data and meta-data such as the sender, the recipient and a topic or routing key that allows the message to be sent to the correct endpoint. Once a message is sent, it is typically processed asynchronously and in a non-blocking way, which means that the sender is not blocked while waiting for the recipient to process the message.

Events, on the other hand, are a way of notifying other parts of the system that something has happened. In contrast to messaging, where a message is sent from one component to another, events are typically broadcast to multiple listeners. Events can be used to notify interested parties that some state has changed or that some action has occurred. Events are commonly used in event-driven architectures, which are becoming increasingly popular as a way of designing scalable, high-performance systems. In event-driven architecture, events are used to trigger a change in a system's state, and then other components are notified of the change via a messaging channel.

### My use of messaging and events

In my project, I often used messaging and events. In Angular, the .subscribe method is used for it. This method is called when a response is returned from an await. Two things can then have happened, a success or an error. If there is a success, this is usually displayed with a message, also if there is an error, a message is usually displayed.

In the case of my application, I used .subscribe when retrieving products. When data is returned, the data is put into a list and then shown to the user. If something goes wrong, it is caught in the error. A message is thereby displayed to the user that something has gone wrong.

Afbeelding met tekst

Automatisch gegenereerde beschrijving