Skateshop Dokumentation

Thema: Modul 326

Dokumentinformationen

Dateiname: Dokumentation.docx

Speicherdatum: 29.03.2022

Autoreninformationen

Autor: Ronnie Hartmann

E-Mail: ronnie.hartmann@noser.com

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc99444654)

[1.1 Sinn und Zweck 3](#_Toc99444655)

[2 Planung 4](#_Toc99444656)

[2.1 Projektbeschreibung 4](#_Toc99444657)

[2.2 Use Case Diagramm 4](#_Toc99444658)

[2.3 Use Case Description 5](#_Toc99444659)

[2.4 CRC Cards 8](#_Toc99444660)

[2.5 Domain Model 8](#_Toc99444661)

# Einleitung

## Sinn und Zweck

In diesem Projekt können wir selber wählen, was wir machen wollen. Es ist in zwei Teile unterteilt: Die Planung und die Realisierung. Das Projekt muss ein Design Pattern enthalten. Ich habe mich für das Composite Pattern entschieden.

# Planung

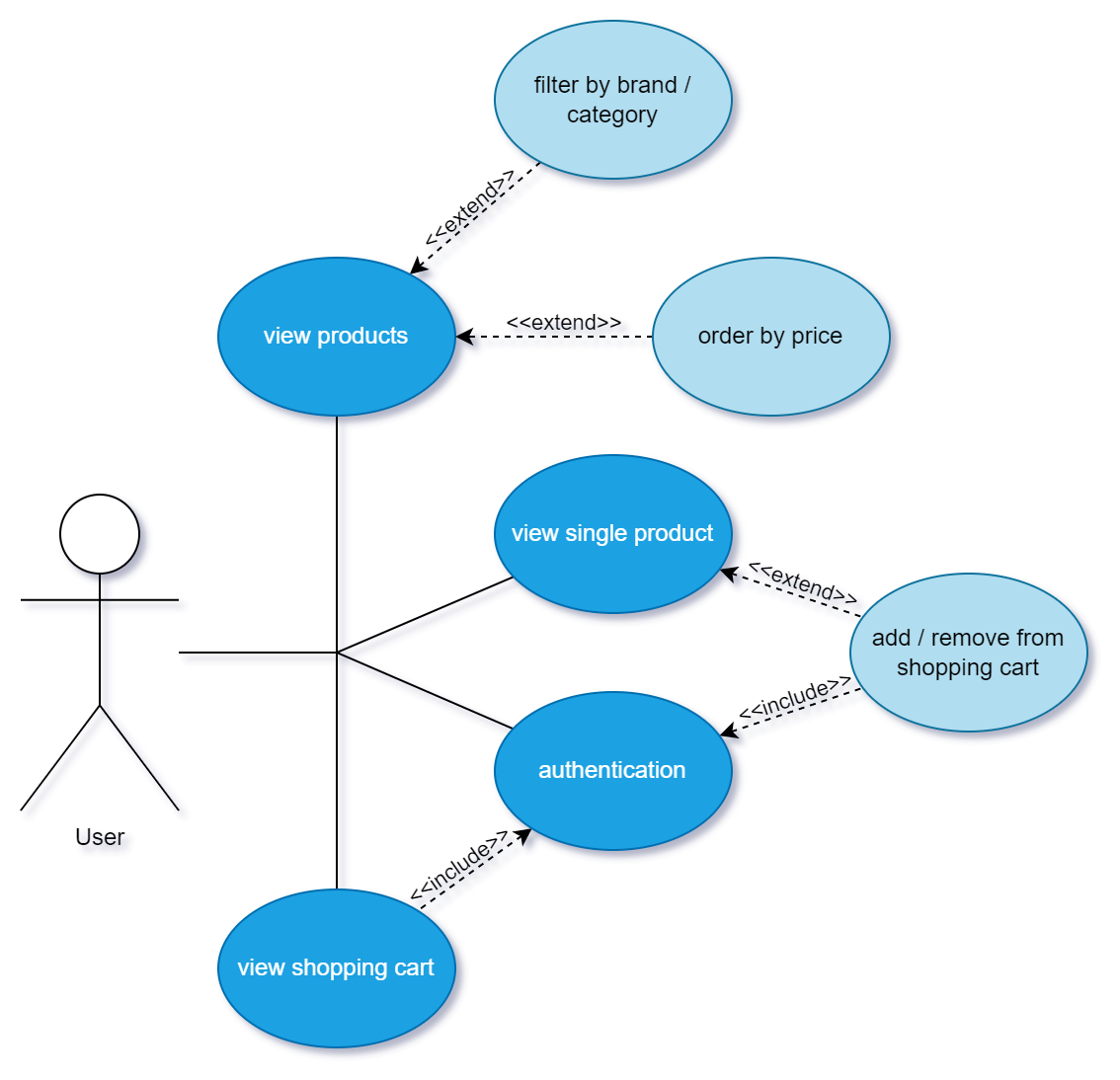
Zuerst habe ich mein Programm mit Diagrammen und Beschreibungen geplant. Erst danach beginne ich mit dem eigentlichen programmieren.

## Projektbeschreibung

Das Projekt ist ein Onlineshop, wo man Skateboard Teile kaufen kann. Es gibt die Kategorien Deck, Trucks, Wheels und Griptape. In den Kategorien gibt es wiederrum Marken. Auf der Startseite kann man die Produkte nach Kategorie und/oder Marke filtern. Dazu können die Produkte auch nach Preis sortiert werden. Die Produkte können in einen Warenkorb gelegt werden. Voraussetzung dafür ist, dass man eingeloggt ist.

## Use Case Diagramm

In diesem Use Case Diagramm sieht man alle Verwendungszwecke des Onlineshops. Da man für den Warenkorb eingeloggt sein muss, wird das Use Case «Authentication» included.



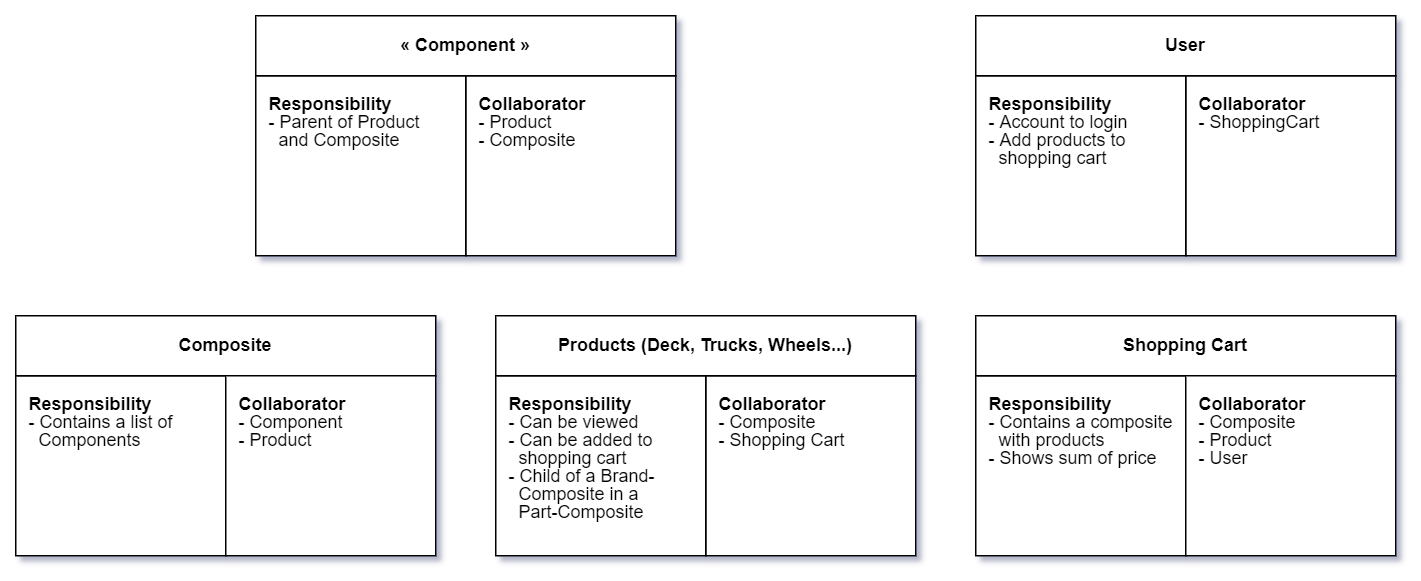
## Use Case Description

|  |  |  |
| --- | --- | --- |
| **Use Case #1** | **Show Products** | |
| Pre-Condition | - | |
| Description of Use Case in detail | **User**   1. Open /products endpoint with browser | **System**   1. Return list of products |
| Post-Condition | Show all products | |
| Exceptions (what can go wrong, how will the system respond?) | * No products available | |
| **Use Case #2** | **Change Order** | |
| Pre-Condition | Order by rating | |
| Description of Use Case in detail | **User**   1. Click on Order Dropdown 2. Select “Order by Price (Cheapest) | **System**   1. Return new list of products ordered by price (cheapest first) |
| Post-Condition | Show all products ordered by Price | |
| Exceptions (what can go wrong, how will the system respond?) | * No products available | |
| **Use Case #3** | **Filter by Skateboard Part (Category)** | |
| Pre-Condition | Showing all Products | |
| Description of Use Case in detail | **User**   1. Click on Category Filter Dropdown 2. Select Category “Decks” | **System**   1. Send back new product list only with decks |
| Post-Condition | Show all Skateboard Decks | |
| Exceptions (what can go wrong, how will the system respond?) | * No products available | |

|  |  |  |
| --- | --- | --- |
| **Use Case #4** | **Filter by Skateboard Brand** | |
| Pre-Condition | Showing all Products | |
| Description of Use Case in detail | **User**   1. Click on Brand Filter Dropdown 2. Select Brand “Polar” | **System**   1. Send back new product list only with “Polar” products |
| Post-Condition | Show all Products from the brand “Polar” | |
| Exceptions (what can go wrong, how will the system respond?) | * No products available | |
| **Use Case #5** | **View single Product** | |
| Pre-Condition | Showing all Products | |
| Description of Use Case in detail | **User**   1. Click on product | **System**   1. Show product details page |
| Post-Condition | Show single product and details | |
| Exceptions (what can go wrong, how will the system respond?) | * Product not found | |
| **Use Case #6** | **Add Product to Shopping Cart** | |
| Pre-Condition | Viewing single Product (User is logged in) | |
| Description of Use Case in detail | **User**   1. Click Button “Add to Shopping Cart” | **System**   1. Check if user is logged in 2. Add product to users shopping cart 3. Show Shopping Cart page |
| Post-Condition | Product added to shopping Cart | |
| Exceptions (what can go wrong, how will the system respond?) | * Product sold out * Product not found * User isn’t logged in | |

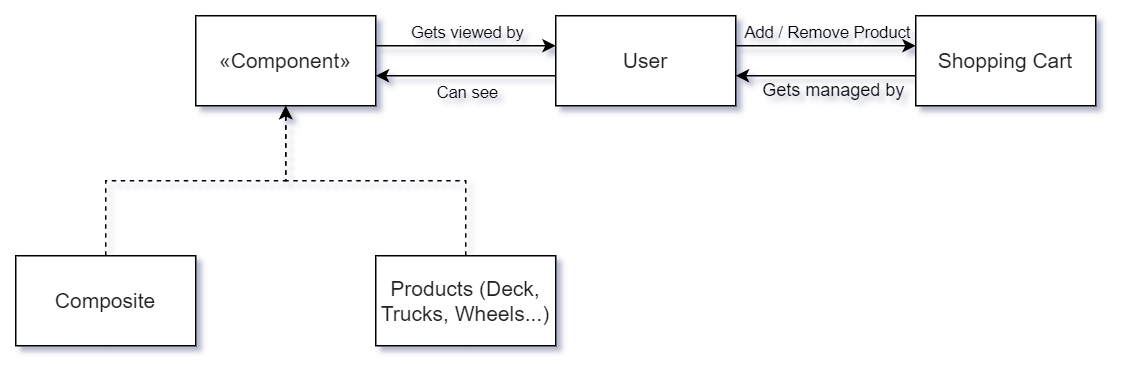
|  |  |  |
| --- | --- | --- |
| **Use Case #7** | **View Shopping Cart** | |
| Pre-Condition | Showing all Products | |
| Description of Use Case in detail | **User**   1. Click on Shopping Cart Button | **System**   1. Check if user is logged in 2. Show Shopping Cart page |
| Post-Condition | Show Shopping Cart and its products with a sum of the price. | |
| Exceptions (what can go wrong, how will the system respond?) | * Not logged in | |
| **Use Case #8** | **Login** | |
| Pre-Condition | Not logged in User | |
| Description of Use Case in detail | **User**   1. Click on Login Button 2. Enter username and password 3. Submit | **System**   1. Check if user isn’t logged in 2. Show Login form 3. Check if user existst & password is valid 4. Show Account page |
| Post-Condition | User is logged in and can see his shopping cart. | |
| Exceptions (what can go wrong, how will the system respond?) | * Invalid username / password * Already logged in | |
| **Use Case #9** | **Register** | |
| Pre-Condition | Not logged in User | |
| Description of Use Case in detail | **User**   1. Click on Register Button 2. Enter new username and password 3. Submit | **System**   1. Check if user isn’t logged in 2. Show Register form 3. Check if username doesn’t already exist 4. Check if password length > 5 5. Show Account page |
| Post-Condition | User is logged in and can see his shopping cart. | |
| Exceptions (what can go wrong, how will the system respond?) | * Already logged in * Username already exists * Password is too short | |
| **Use Case #10** | **Logout** | |
| Pre-Condition | Login (Authenticated) | |
| Description of Use Case in detail | **User**   1. Click Logout Button | **System**   1. Check if user is logged in 2. Log user out |
| Post-Condition | User is logged out and can’t see his shopping cart anymore. | |
| Exceptions (what can go wrong, how will the system respond?) | * User isn’t logged in | |

## CRC Cards



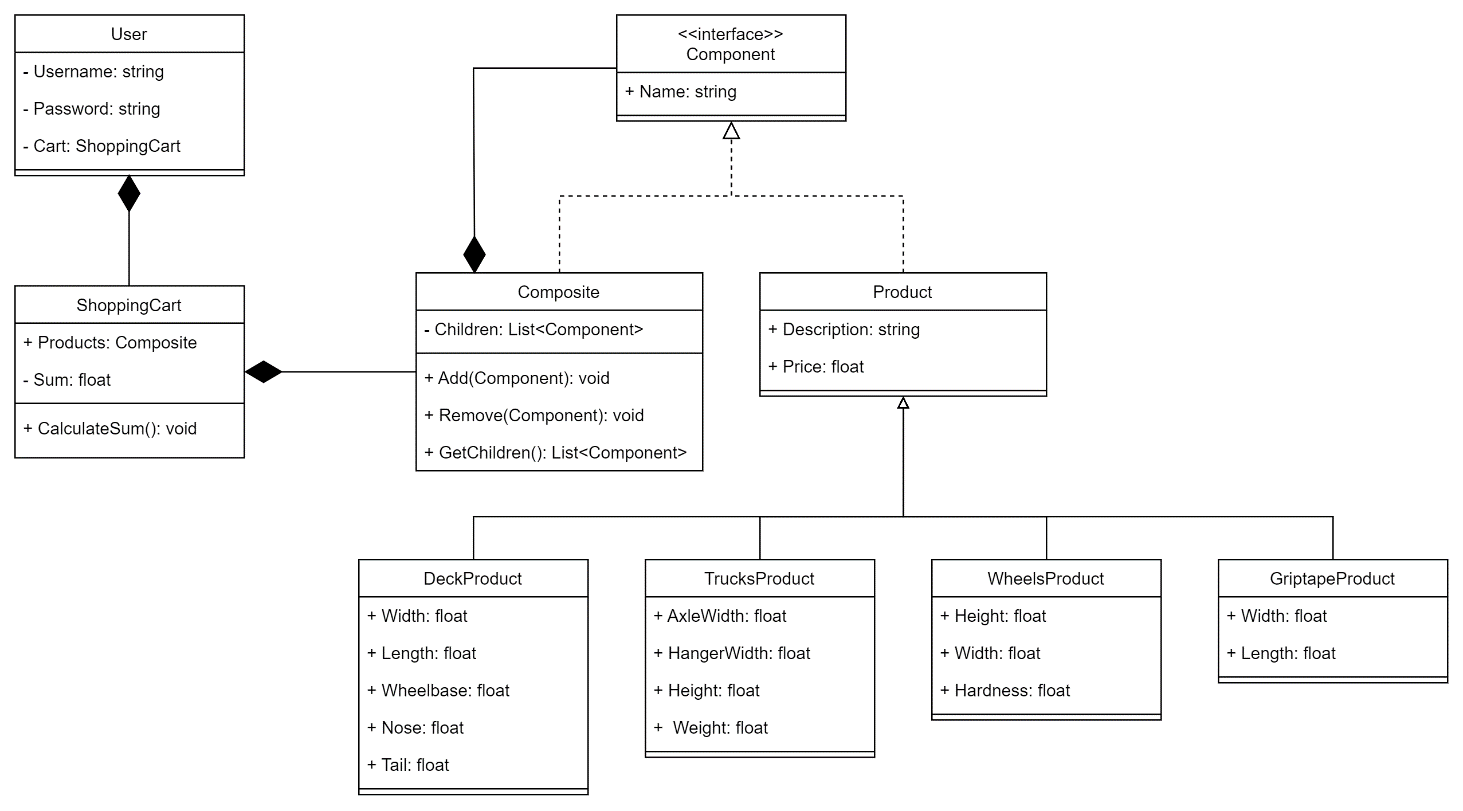
Die Product Klassen wurden in der «Products» Card zusammengefasst. Folgende Product Klassen wird es geben: DeckProduct, TrucksProduct, GriptapeProduct, WheelsProduct

## Domain Model



## Class Diagram

Dies ist mein Class Diagram für meinen Skate Shop.



# Implementation

Als Entwicklungsumgebung benutze ich Visual Studio 2019 mit einer MSSQL Datenbank. Ich erstelle ein ASP.NET Core MVC Projekt.

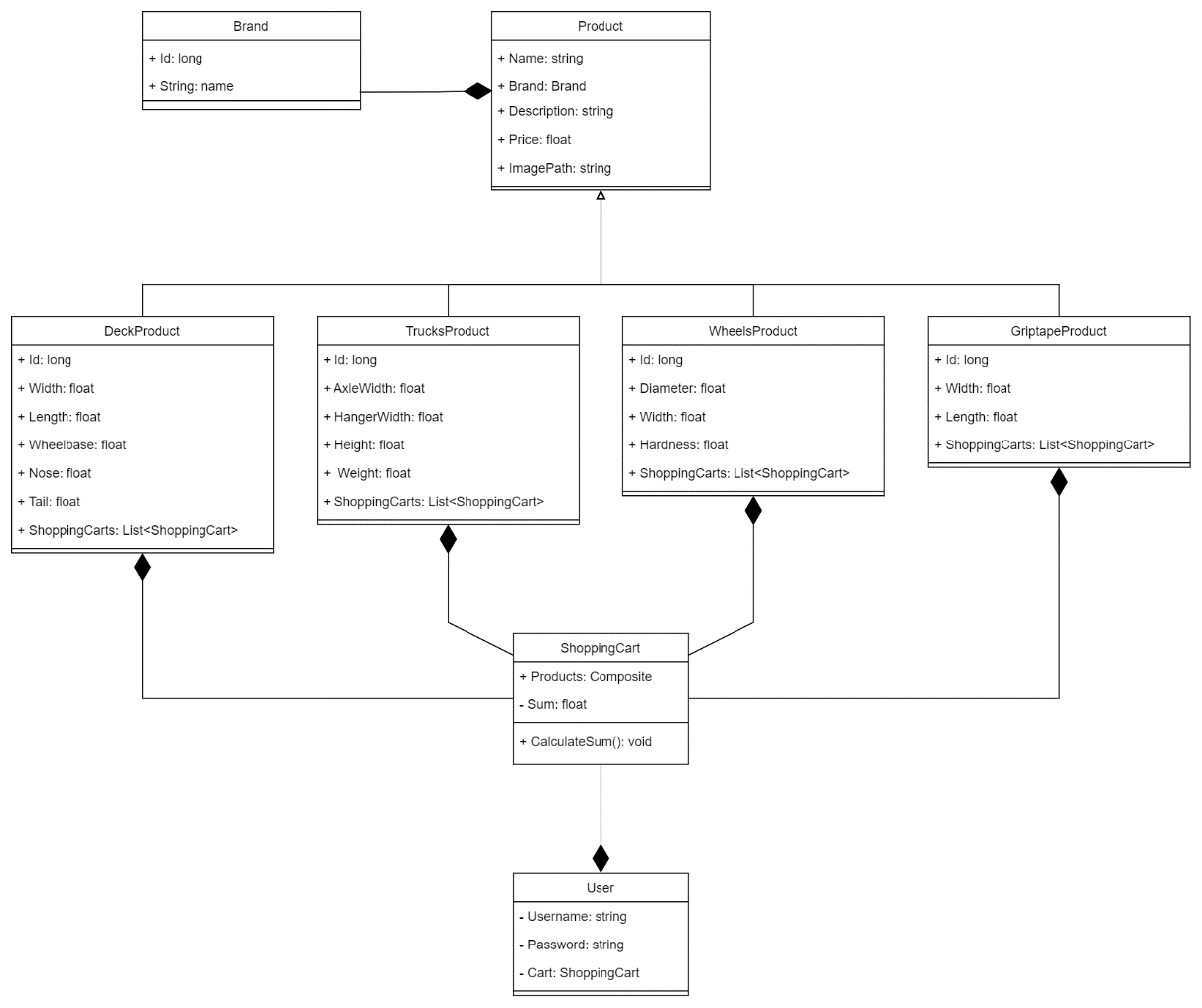
## Problem mit Composite Pattern

Als ich das Composite Pattern implementieren wollte sah ich, dass es dabei ein Problem gibt. In einem MVC Projekt arbeitet man mit Model-Klassen, welche gleichzeitig die Tabellen in der Datenbank sind. Da eine Tabelle nicht von einer anderen Tabelle erben kann, war es nicht möglich das Composite Pattern zu implementieren.

Ich habe beschlossen, stattdessen das Dependency Injection Design Pattern anzuwenden. Ausserdem ist MVC (Model-View-Controller) ebenfalls ein Pattern, welches ich anwende. Es unterteilt die Anwendung in Models (Tabellen / Komponenten), Views (Ansichten, das GUI) und Controller, welche die CRUD Operationen enthalten. Oftmals verschiebt man die Logik in sogenannte Services. In meinem Projekt verwende ich einen AuthService, welche die Autorisierung/Authentifizierung übernimmt. Ansonsten befindet sich die Logik bei mir in den Controllern.

## Neues Class Diagram

Zu meinen Planänderungen gehört auch ein neues Class Diagram. Hier ist die neue Version des Diagramms:



## Models

Wie man im neuen Class Diagram sehen kann, habe ich die Models «DeckProduct», «TrucksProduct», «WheelsProduct» und «GriptapeProduct» erstellt. Sie alle erben von der Klasse «Product», da sie viele gemeinsame Eigenschaften haben. Jedes Model benötigt eine Id, damit es in der Datenbank einen Primary Key hat und die Models leicht zu unterscheiden und zu finden sind.

## Ein Bild, das Text enthält. Automatisch generierte BeschreibungViews

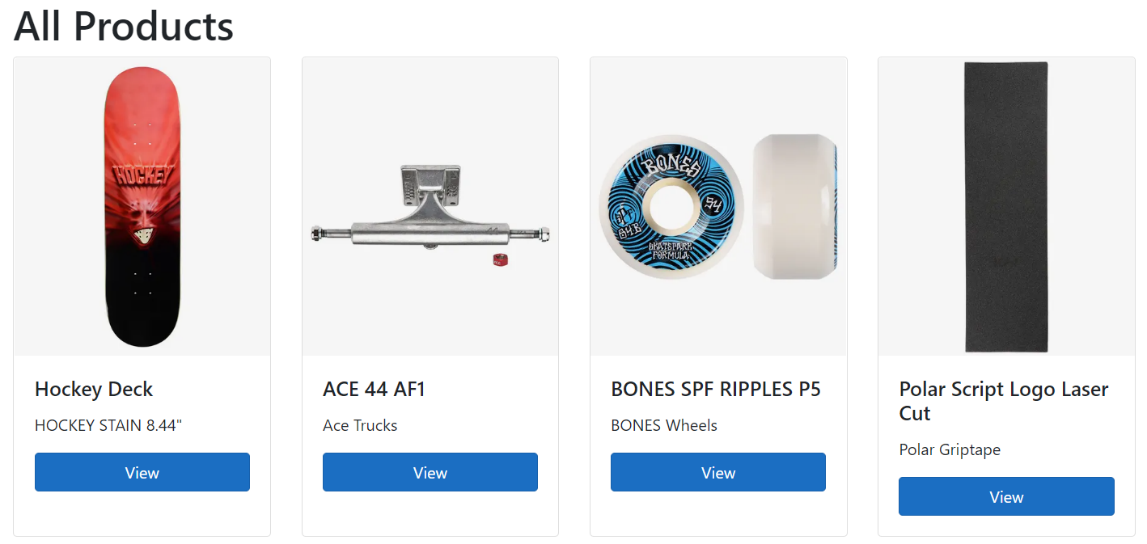
Mein Projekt enthält insgesamt 19 Views. Die Views im Share Ordner sind jedoch nur Teile von Views, wie zum Beispiel die Navbar oder der Footer.

Für die User gibt es noch Views zum Bearbeiten, Löschen und Ansehen der einzelnen Benutzer. Diese sind jedoch nur für den Admin verfügbar.

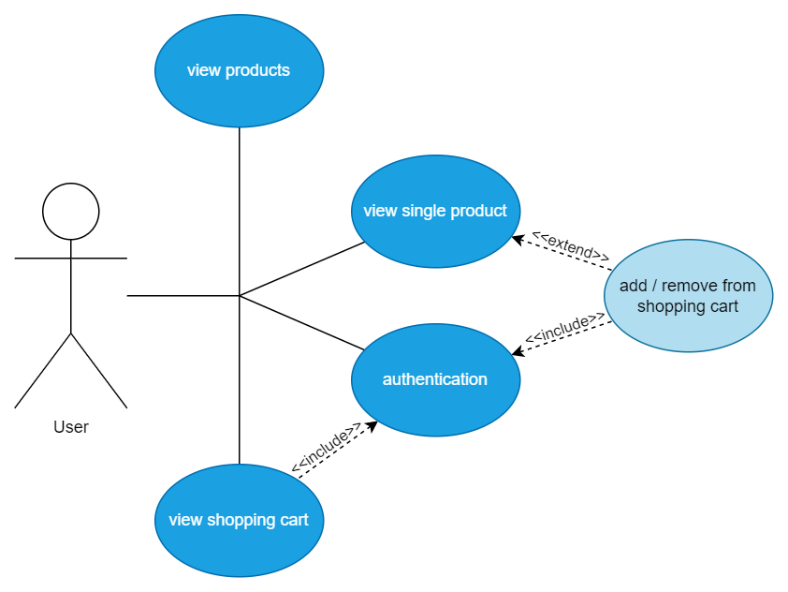
Für die Ansicht aller Produkte habe ich eine einzelne View erstellt, welche vom ProductsController alle Produkte bekommt uns anzeigt.

### Übersicht aller Produkte

Leider habe ich nicht herausgefunden, wie ich die Produkte filtern lassen kann. Ich habe etwa 2 Stunden herumprobiert und bin immer wieder auf Fehler gestossen. Deshalb kann nur alle Produkte auf einmal anzeigen lassen.



Ich musste dafür noch das Use-Case-Diagram überarbeiten. Das neue Use-Case-Diagram sieht so aus:



## Controller

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungOft erstellt man für jedes Model einen Controller. Ich habe jedoch die Endpoints und die Logik aller Produkt Arten im ProductsController zusammengefasst.

Ansonsten gibt es noch den HomeController, welcher die Startseite und Privacy anzeigen lässt, einen ShoppingCartController, welcher den Einkaufswagen verwaltet und einen UsersController, welcher die CRUD Operationen für das Einloggen, Ausloggen und Registrieren enthält.

Ausserdem gibt es dort noch Methoden, um die Benutzer zu verwalten, worauf man jedoch nur als Admin zugreifen kann.

# Testing

In diesem Abschnitt teste ich alle meine Use-Cases um zu überprüfen, dass sie fehlerfrei funktionieren.

|  |  |
| --- | --- |
| **Test ID** | #1 |
| **Test Name** | Produkte anzeigen |
| **Use Case** | #1 Show Products |
| **Description** | Alle Produkte auf der Seite «/products» anzeigen. |
| **Pre-Conditions** | Das Programm läuft, es gibt Einträge von Produkten in der Datenbank |
| **Expectation** | Die Produkte werden alle angezeigt, mit einem Bild und einem «View» Button |
| **Post-Condition** | Alle Produkte werden angezeigt, wie erwartet. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #2 |
| **Test Name** | Einzelnes Produkt anzeigen |
| **Use Case** | #5 View Single Product |
| **Description** | Wenn man auf den «View» Button klickt, sollten mehr Informationen über ein einzelnes Produkt angezeigt werden. |
| **Pre-Conditions** | Die Products Übersicht ist offen und es werden Produkte aufgelistet. |
| **Expectation** | Nachdem man auf den View Button klickt, sieht man je nach Produkt die dazugehörigen Informationen, wie die Beschreibung, den Preis etc. |
| **Post-Condition** | Die Informationen zum Produkt werden angezeigt. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #3 |
| **Test Name** | Produkt in den Warenkorb legen |
| **Use Case** | #6 Add Product to Shopping Cart |
| **Description** | Auf der einzelnen Produkt Ansicht gibt es den Button «Add to Shopping Cart», womit man Produkte in den Warenkorb legen kann. |
| **Pre-Conditions** | Ein einzelnes Produkt wird angezeigt und man ist eingeloggt. |
| **Expectation** | Nachdem man auf den Button klickt, wird das Produkt in den Warenkorb gelegt und er öffnet sich. |
| **Post-Condition** | Der Warenkorb wird geöffnet und das neue Produkt ist darin sichtbar. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #4 |
| **Test Name** | Produkt in den Warenkorb legen (Ohne Login) |
| **Use Case** | #6 Add Product to Shopping Cart |
| **Description** | Auf der einzelnen Produkt Ansicht gibt es den Button «Add to Shopping Cart», womit man Produkte in den Warenkorb legen kann. |
| **Pre-Conditions** | Ein einzelnes Produkt wird angezeigt doch man ist nicht eingeloggt. |
| **Expectation** | Wenn man nicht eingeloggt ist, sollte man auf die Login-Seite weitergeleitet werden. |
| **Post-Condition** | Man wird auf die Login-Seite weitergeleitet |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #5 |
| **Test Name** | Warenkorb anzeigen |
| **Use Case** | #7 View Shopping Cart |
| **Description** | In der Navbar sieht man, wenn man eingeloggt ist, einen Button, der einen zum Warenkorb weiterleitet. |
| **Pre-Conditions** | Man ist eingeloggt und klickt auf den Button. |
| **Expectation** | Nachdem man auf den Button klickt, wird der Warenkorb mit allen Produkten und der Summe der Preise angezeigt. |
| **Post-Condition** | Der Warenkorb wird geöffnet und alle hinzugefügten Produkte inklusive Summe sind vorhanden. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #6 |
| **Test Name** | Warenkorb anzeigen (ohne Login) |
| **Use Case** | #7 View Shopping Cart |
| **Description** | Ohne Login sollte der Shopping Cart nicht aufrufbar sein. |
| **Pre-Conditions** | Man ist nicht eingeloggt. |
| **Expectation** | Der «Shopping Cart» Button wird nicht angezeigt, da man nicht eingeloggt ist. Geht man trotzdem auf die URL «/shoppingcart» gibt es einen 404 zurück. |
| **Post-Condition** | Es gibt keinen Shopping Cart Button. Wie erwartet bekommt man eine NotFound Exception wenn man trotzdem auf die URL geht. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #7 |
| **Test Name** | Einloggen |
| **Use Case** | #8 Login |
| **Description** | Wenn man nicht eingeloggt ist, gibt es in der Navbar einen Login Button. Dort kann man den Username und das Passwort eingeben um sich einzuloggen. |
| **Pre-Conditions** | Man ist nicht eingeloggt und hat das Login Fenster offen. |
| **Expectation** | Gibt man korrekte Daten ein, wird man eingeloggt. |
| **Post-Condition** | Man wird eingeloggt uns sieht oben seinen Nutzernamen in der Navbar. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #8 |
| **Test Name** | Einloggen (falsche Daten) |
| **Use Case** | #8 Login |
| **Description** | Gibt man falsche Daten ein, funktioniert das Login nicht und man bekommt eine Meldung. |
| **Pre-Conditions** | Man ist nicht eingeloggt und hat das Login Fenster offen. |
| **Expectation** | Man gibt falsche Daten ein und bekommt eine Fehlermeldung. |
| **Post-Condition** | Es wird die Meldung «Wrong username or password» ausgegeben. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #9 |
| **Test Name** | Registrieren |
| **Use Case** | #9 Register |
| **Description** | Wenn man sich mit einem neuen Benutzernamen registriert wird ein Account erstellt und man ist eingeloggt. |
| **Pre-Conditions** | Man ist nicht eingeloggt und hat das Register Fenster offen. Ein neuer Benutzername mit einem Passwort wird eingegeben. |
| **Expectation** | Man wird eingeloggt und ein Benutzer wird erstellt. Dazu wird noch ein neuer Warenkorb für den User erstellt. |
| **Post-Condition** | Der Benutzer ist eingeloggt und hat einen Warenkorb |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #10 |
| **Test Name** | Registrieren (vorhandener Username) |
| **Use Case** | #9 Register |
| **Description** | Wenn man sich mit einem vorhandenen Benutzernamen registriert gibt es eine Fehlermeldung. |
| **Pre-Conditions** | Man ist nicht eingeloggt und hat das Register Fenster offen. Ein vorhandener Benutzername mit einem Passwort wird eingegeben. |
| **Expectation** | Es gibt eine Fehlermeldung und man wird nicht eingeloggt. |
| **Post-Condition** | Die Fehlermeldung «The username already exists» wird ausgegeben. |
| **Test Result** | Passed |

|  |  |
| --- | --- |
| **Test ID** | #11 |
| **Test Name** | Ausloggen |
| **Use Case** | #10 Logout |
| **Description** | Nach dem logout ist man wieder ausgeloggt und hat keinen Warenkorb mehr. |
| **Pre-Conditions** | Man ist eingeloggt und klickt auf den Logout Button |
| **Expectation** | Man wird ausgeloggt |
| **Post-Condition** | Man wird ausgeloggt |
| **Test Result** | Passed |