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

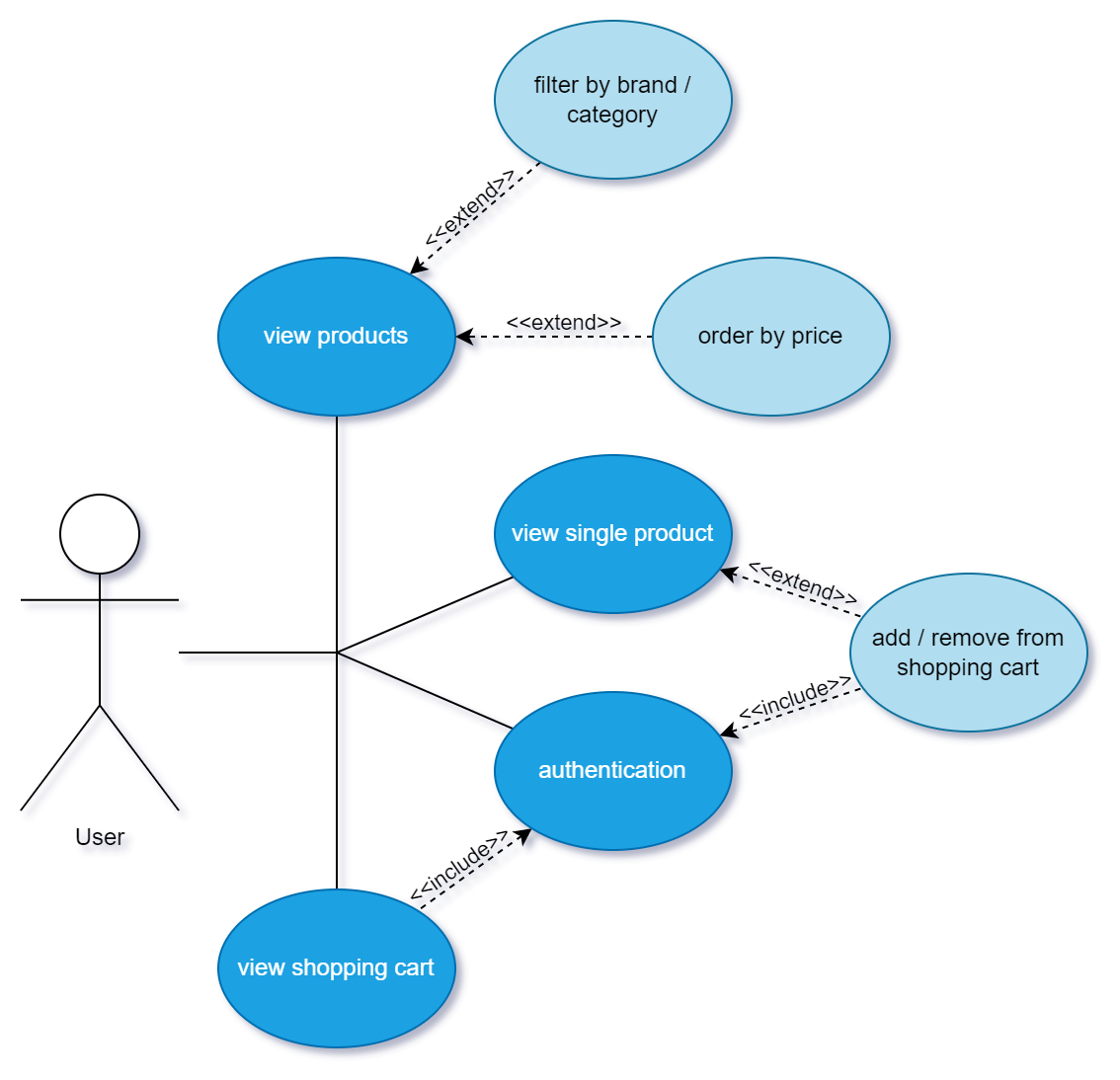
Zuerst habe ich mein Programm mit Diagrammen und Beschreibungen geplant. Erst danach beginne ich mit dem eigentlichen programmieren.

## 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.



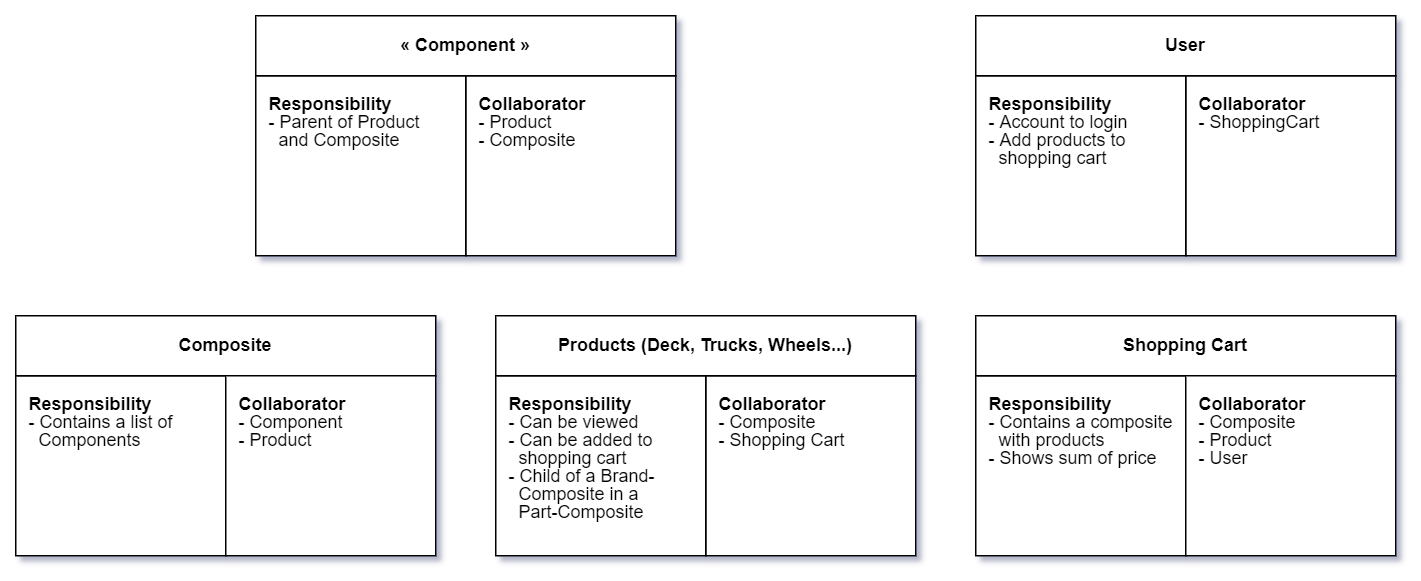
## Use Case Description

|  |  |  |
| --- | --- | --- |
| **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 ordered by rating (Default) |
| Post-Condition | Show all products ordered by rating | |
| 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 | |

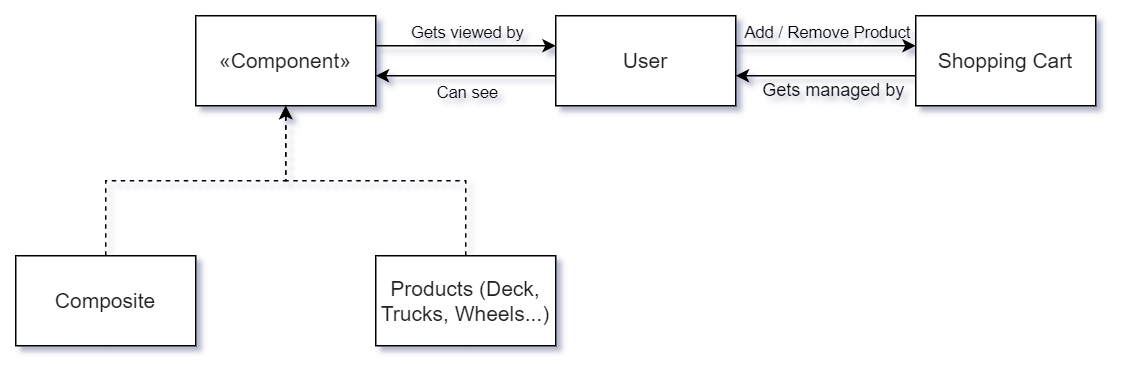
|  |  |  |
| --- | --- | --- |
| **Use Case #7** | **View Shopping Cart** | |
| Pre-Condition | Showing all Products | |
| Description of Use Case in detail | **User**   1. Click on Shopping Cart Button | **System**   1. Check if user is logged in 2. Show Shopping Cart page |
| Post-Condition | Show 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 #9** | **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

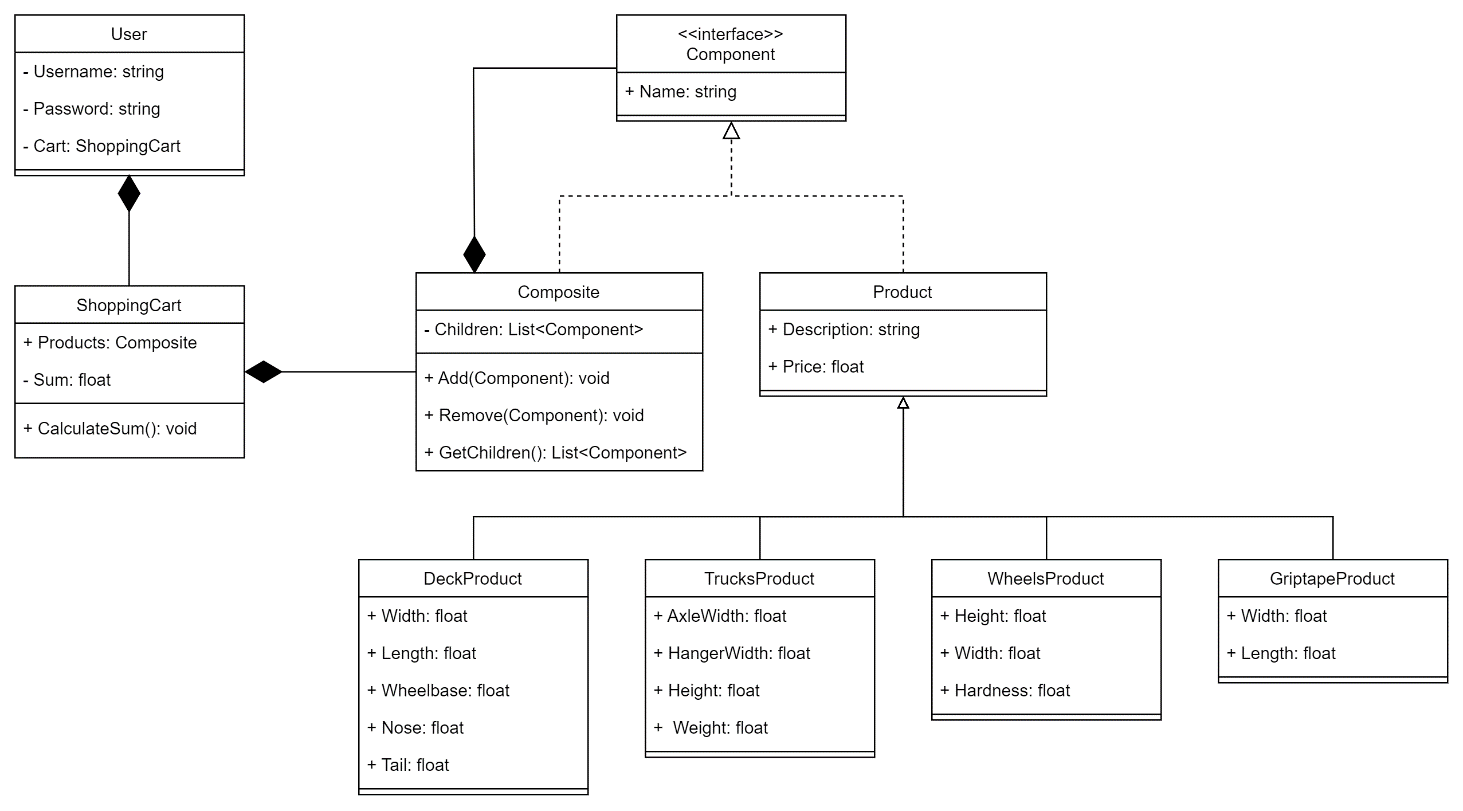


Die Product Klassen wurden in der «Products» Card zusammengefasst. Folgende Product Klassen wird es geben: DeckProduct, TrucksProduct, GriptapeProduct, WheelsProduct

## Domain Model



## Class Diagram



# 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.