DevShop - Dokumentation

ERSTELLT VON:
Benjamin M. Knoll

BETREUT VON:
Josef Strasser-Leitner, ba

TERMIN DER ABGABE:
06.02.2023

1. Eidesstattliche Erklärung

Ehrenwörtliche Erklärung

Hiermit bestätige ich, die vorliegende Projektarbeit namens „DevShop“ zur Gänze eigenständig und ausschließlich mit Hilfe der angegebenen Quellen verfasst zu haben. Die Arbeit hat einer anderen Prüfungsbehörde in gleicher oder ähnlicher Form noch nie vorgelegen.

....

Datum Unterschrift des Verfassers

2. Abstract

Der Vertrieb von Produkten im Onlinehandel ist heutzutage nicht mehr wegzudenken. Aufgrund meines Ursprungs als Web-Entwickler habe ich bereits zahlreiche Onlineshops mittels hauseigenem CMS der Firma aufgebaut. Jedoch war mir der programmier-Prozess im Hintergrund bislang unbekannt, da sich meine Tätigkeiten hauptsächlich auf das Frontend beschränkten. Das Projekt „DevShop“ diente somit als perfekte Gelegenheit, mein Wissen zu erweitern und einen Einblick in die Backend-Entwicklung eines Shops zu erhalten.

Inhaltsverzeichnis

3. Einführung

Ziel des Projektes ist es, Kenntnisse über den vollständigen Entwicklungsprozess eines Online-Shops zu erlangen. Primär liegt die Backend-Entwicklung im Fokus, welche sich auf den Authentifizierungsprozess sowie die Datenbankanbindung und Logik der einzelnen Funktionalitäten konzentriert.

Das Resultat aus dieser Entwicklung ist ein Demo-Shop, der die Möglichkeiten eines Onlinesystems mittels rechtegeschützter Eingabemasken im Backend und Ausgabe der angelegten Artikel im Frontend demonstriert. Sinn und Zweck der Ausgabe ist es, den Besucher über die gewünschten Artikel zu informieren, statt einen Bestellvorgang einzuleiten. Zudem stehen dem Anwender eine Suchfunktion und eine Auswahl zweier verschiedener Darstellungen der Artikel-Ausgabe zur Verfügung.

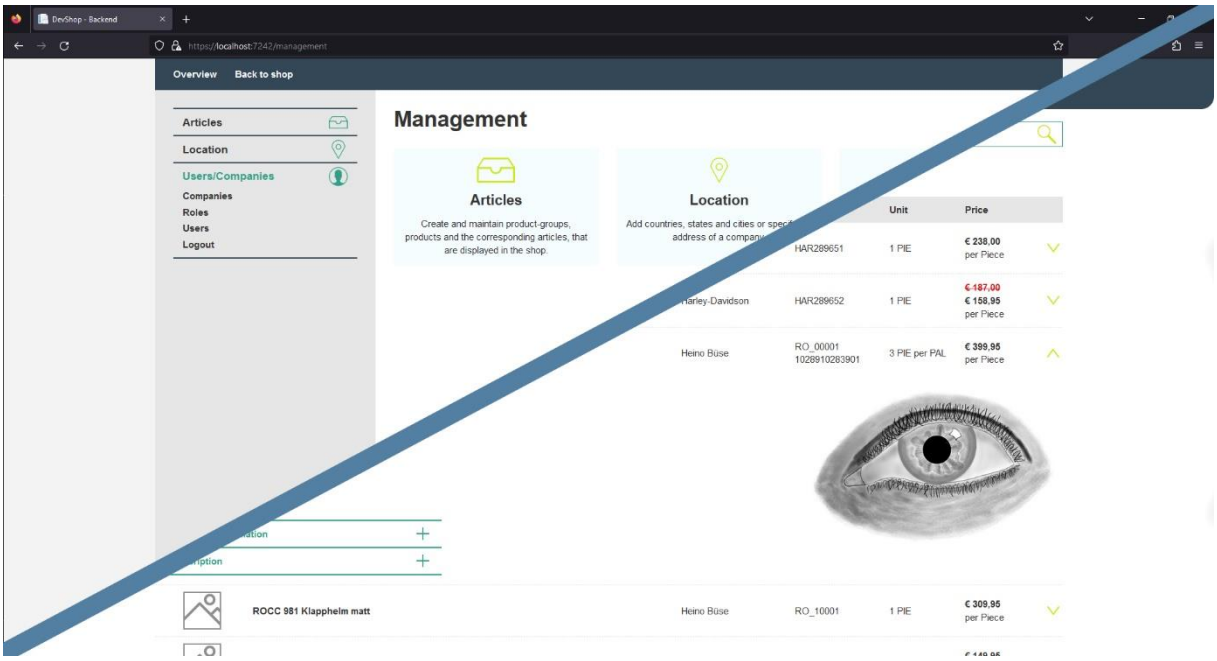

Abbildung 1: Teaser Verwaltung/Shop

4. Konzept

Für jede Software wird vor der Entwicklung zuerst ein Konzept erarbeitet, in dem alle Fragen hinsichtlich des Ausmaßes und Zieles des Projektes vorab geklärt, und die verwendeten Technologien bestimmt werden. Die sorgfältige Planung jedes Prozesses dient nicht nur dazu, spät im Projekt eintretende fehlende Ressourcen oder falsch umgesetzte Funktionen vorzubeugen, sondern gibt auch einen Ablauf der notwendigen Arbeitsschritte vor.

4.1. Datenbank

Eine der wichtigsten Säulen für Online-Shops ist die Datenbank. Der Entwurf der einzelnen Tabellen, ihre Abhängigkeiten zueinander sowie die benötigten Attribute und deren Datentypen erfolgt unter Berücksichtigung der Normalformen. Die Normalisierung von Datenbanken dient „bei relationalen Datenbanken zur Vermeidung von Redundanzen“ (Hosting-Technik, 2023).

Technologie
Für dieses Projekt wurde das relationale Datenbank-Managementsystem (DBMS) „SQL Server Management Studio 2018“ von Microsoft verwendet. Begründung der Wahl ist sowohl die bereits im Vorfeld vorhanden gewesene persönliche Erfahrung mit dem Programm als auch die Flexibilität des DBMS und die Möglichkeit, große Datenmengen schnell zu verarbeiten.

Verwendete Tabellen
Die ersten drei Tabellen sind den Standorten von Produkt-Herstellern gewidmet, und bestimmen Land, Bundesland und Stadt des Firmensitzes. Jede Stadt ist vom Bundesland abhängig, dieses wiederum vom jeweiligen Land.
Hersteller bzw. Firmen im Allgemeinen sind nur vom Land abhängig. Bundesland und Stadt dienen lediglich zur Bestimmung der Adresse, welche in einer eigenen Tabelle ausgelagert ist.

Da die Verwaltungsoberfläche hinter einem Login und Admin-Recht geschützt liegt, ist je eine Tabelle für Benutzer und verfügbare Rechte von Nöten. Jeder Benutzer kann nur genau ein Recht zugewiesen bekommen, weshalb diese Tabelle in einer 1:n Beziehung zur Rechte-Tabelle steht, und wird einer Firma bzw. einem Hersteller zugeordnet.

Grundidee des Shops ist es, aus verschiedenen Kategorien zu wählen und somit die dazugehörigen Artikel anzuzeigen. Hierfür benötigen die Kategorien eine eigene Tabelle. Diese können auch verschachtelt werden – sprich es können beliebig viele Unterkategorien mittels Attributes als Verweis auf den übergeordneten Datensatz erstellt werden.

Als Gruppierung zusammengehöriger Produkte dient die Tabelle der Produkt-Gruppen. Auch hier können den Gruppen wieder mithilfe eines Attributes in der Tabelle übergeordnete Einträge zugeordnet werden. Außerdem verweisen sie in einer 1:n Beziehung auf die Kategorien und sind abhängig vom Hersteller.

Folgend auf die Produkt-Gruppen gibt es die Tabelle der Produkte. Sie können als Überbegriff oder Zusammenfassung gleichartiger Artikel verstanden werden und verweisen mittels Fremdschlüssel auf die jeweilige Gruppe des jeweiligen Herstellers.

Letztlich ist eine Artikel-Tabelle von Nöten, welche auf ein Produkt verweist. Doch verschiedene Artikel haben möglicherweise verschiedene Einheiten, wie etwa Stück oder Liter. Um dies zu ermöglichen, werden die Einheiten in einer weiteren Tabelle ausgelagert. Weiters können Artikel zusätzliche Informationen erhalten, die zu Artikeln aus einem anderen Produkt variieren. Hierzu wird die Tabelle der Artikel-Header benötigt. Sie verweist über einen Fremdschlüssel auf ein Produkt und gibt an, welche Eigenschaften die Artikel im jeweiligen Produkt besitzen können.

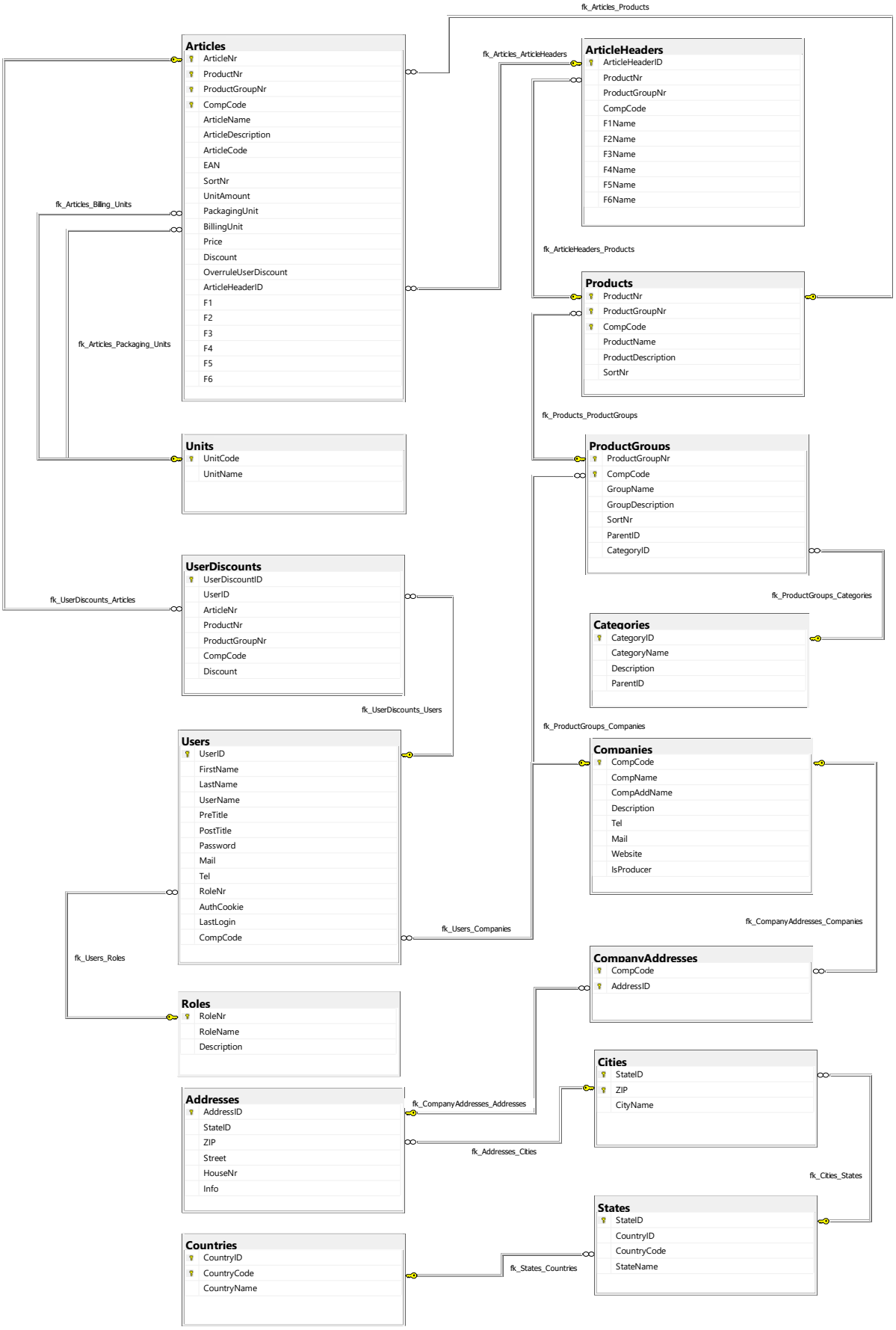

Abbildung 2: Datenbank Diagramm

4.2. Website

Grundsätzlich wird die Seite in zwei Bereiche unterteilt. Eine Verwaltungsoberfläche, die nur für Admins zugänglich ist und alle Eingabemasken beinhaltet, und der für jeden zugängliche eigentliche Shop mit Menü, Artikelausgabe und Detailansicht eines Artikels. Um sofort eine eindeutige Unterscheidung beider Bereiche zu ermöglichen, gibt es für das Frontend und das Backend je ein unterschiedliches Layout. Lediglich die Grundeinstellungen, wie verwendete Farben, Schriftarten/-größen und Abstandsdefinitionen, bleiben gleich.

Technologie
Als Editor für HTML, CSS und JS-Code kam aufgrund des extrem schnellen und flüssigen Workflows das Programm „Sublime Text 3“ zum Einsatz.

Alle auf der Website dargestellten Icons und Bilder wurden eigenständig mit dem Designer-Programm „Affinity Designer“ kreiert bzw. bearbeitet. Die vielseitigen Einsatzmöglichkeiten reichen von simpler Bildbearbeitung bis hin zur professionellen Illustration und machen das Programm somit zur geeigneten Wahl für das Designen von Seiten und Erstellen von Icons.

Programmiert wurde mit „Visual Studio 2022“ von Microsoft unter Verwendung der Programmiersprache C#. Bei der Auswahl des Frameworks fiel die Entscheidung auf Blazor Server, da die am Server vorgerenderten Komponenten für interaktiv bedienbare Elemente am Client sorgen.

Frontend
Vorgesehen ist ein Menü mit Option zum Login/Logout, ein aufklappbares Menü zur Auswahl der Kategorien und ein Footer. Auf der Einstiegsseite sollen eine gewisse Anzahl an zufälligen Artikeln platziert sein. Die Shop-Seite soll eine Suchfunktion und maximal 20 Artikel pro Seite mit der Möglichkeit zum Durchklicken auf vorige und nachführende Seiten beinhalten, um den Benutzer nicht mit Datenmengen zu überfluten und die Seitenlänge kurz und übersichtlich zu halten. Je nach Vorzügen des Anwenders ist eine Listendarstellung und eine Boxen-Ansicht der Artikelauflistung geplant. Bei Auswahl eines Artikels soll dieser auf einer Detail-Seite dargestellt werden.

Backend
Die Verwaltungsoberfläche ist mit einer Navigation zu den jeweiligen Seiten der Eingabemasken zu versehen. Jede Maske beinhaltet neben den einzelnen Feldern zur Eingabe der Werte eine Auswahl aller bereits angelegten Datensätze und Buttons mit den Funktionen zum Löschen oder Erstellen eines Eintrages.

5. Programm-Logik

Herzstück der Applikation ist die Logik der einzelnen Funktionalitäten sowie Verarbeitung und Austausch der Daten. Von notwendigen Einstellungen, die eine Verbindung zur Datenbank ermöglichen, über Hilfs-Klassen, um einen oft verwendeten Vorgang auszulagern, bis hin zu den eigentlichen Unterseiten der Website.

5.1. Einteilung des Programmcodes

Damit eine effiziente und erleichterte Arbeitsweise gewährleistet werden kann, werden alle Abschnitte des Programmcodes, beispielsweise Klassen und Views, in einer übersichtlichen Struktur unterteilt.

wwwroot
Dieses Verzeichnis dient als Wurzelverzeichnis für alle Dateien, die vom Browser verwendet werden bzw. auf der Website eingebunden sind. Nicht nur sind hier CSS und JS-Dateien enthalten, sondern auch jegliche grafischen Inhalte, welche auf der Seite Verwendung finden. Raufgeladene Bilder werden ebenfalls hier abgelegt.

Authentication
In diesem Ordner spiegeln sich alle für den Authentifizierungsprozess notwendigen Klassen und Methoden wider. Letztendlich liegen die dazugehörigen Views allerdings in einem anderen, extra dafür vorgesehenen Verzeichnis.

Data
Alles rund ums Thema Datenverarbeitung ist hier abgelegt – sei es die Verbindung zur Datenbank an sich, Modell-Klassen, oder Methoden zum Handhaben gewisser Daten.

Pages
Die eigentlichen Unterseiten der Website sind in diesem Verzeichnis vorzufinden. Die dazugehörigen Klassen dienen lediglich der Dynamisierung der Oberfläche durch interaktive Funktionalitäten und der Anforderung benötigter Daten. Es findet jedoch keine Verarbeitung der Daten mehr statt.

Shared
Hier hinterlegte Dateien sind für alle Views zugänglich. Layouts für das Frontend und das Backend sind hier gespeichert.

5.2. Datenbankanbindung

In der, wie in **4.2 Website** erläutert, gewählten Technologie Blazor Server wird eine Verbindung zur Datenbank mittels Entity Framework (EF) erstellt. EF ist ein von Microsoft entwickelter „ORM (Objektrelationaler Mapper), der den Zugriff auf die Datenbank über ein Objektmodell gewährleistet“ (Doberenz, Gewinnus, Kotz, & Saumweber, 2018, S. 728). Für jede Tabelle in der Datenbank wird daher je eine Modell-Klasse benötigt. Die darin enthaltenen Properties repräsentieren die einzelnen Attribute der Tabelle. Über eine Kontext-Klasse werden anschließend alle Datenbankzugriffe geregelt. Da der Kontext bereits allerlei Funktionen für das Zusammenspiel mit der Datenbank besitzt, ist es nicht mehr notwendig, SQL-Code selbst zu schreiben.

Prinzipiell ist zwischen zwei Herangehensweisen zu unterscheiden, die für die Erzeugung der Objektmodelle sorgen. Auf der einen Seite steht die sogenannte „CodeFirst“-Methode zur Verfügung, in der die Modelle und der Kontext händisch angelegt werden und die Datenbank anhand dieser Modelle über einen Befehl erstellt wird. „DatabaseFirst“ auf der anderen Seite ist das genaue Gegenstück. Nach händischer Erstellung der Datenbank werden mit einem Befehl die Modelle und die Kontext-Klasse automatisch generiert.
Für dieses Projekt kam letztere Methode zum Einsatz, da die Festlegung der Zusammenhänge der einzelnen Tabellen mittels SQL schneller und einfacher erfolgt.

Repositories (Repos)
Aufgrund der einzelnen Modell-Klassen wird pro Klasse ein sogenanntes Repository angelegt. Dieses beinhaltet unter anderem Methoden zum Selektieren, Speichern, Aktualisieren und Löschen von Datensätzen. Alle weiteren Funktionalitäten decken jegliche Datenverarbeitung des Modells ab, beispielsweise die Konvertierung eines Modell-Objektes in ein View-Modell, welches nur Daten beinhaltet, die in der View auch verwendet werden.

Unit-Of-Work (UOW)
Meistens werden in der View nicht nur Daten aus einem einzigen Modell, sondern aus mehreren verschiedenen Modellen benötigt. Hier kommt die Unit-Of-Work, oder kurz UOW, ins Spiel. Sie dient als Mittelmann zwischen den Repositories und den Views, indem sie jedes verfügbare Repo als Property beinhaltet. Somit muss in der View nur noch die UOW eingebunden werden, um Zugriff auf alle Repos zu erlangen.

Abbildung 3: Veranschaulichung UOW

5.3. Authentifizierung

Microsoft stellt einen fix fertigen Authentifizierungsprozess mit Login, Logout, Registrierung und noch mehr für die gewählte Technologie zur Verfügung. Diese Seiten müssen lediglich einmal generiert werden und sind dann sofort einsatzbereit.
In diesem Projekt wurde jedoch auf die automatisch generierbare Variante von Microsoft verzichtet. Der komplette Prozess vom Hashen des Passwortes bis hin zum Setzen eines Cookies beim Login wurde vollständig händisch selbst gemacht. Grund für diesen nicht ganz trivialen Umweg sind zum einen die, auch wenn bis zu einem gewissen Grad manipulierbaren, mit der Generierung erstellten Entitäten in der Datenbank, von denen weniger als die Hälfte genutzt werden würde, mitsamt vieler überflüssiger Attribute. Zum anderen war der eigene Lerneffekt ebenfalls ein ausschlaggebender Faktor.

Mit jeder Anfrage an die Website werden automatisch gewisse Authentifizierungs-Token zwischen der Web-Applikation und dem Browser ausgetauscht. Aufgrund dessen ist es sehr wichtig, einen sogenannten „Anti-Forgery Token“ in die Programmlogik einzubauen, da dieser den Benutzer vor XSRF-Attacken schützt, welche eine zuvor authentifizierte Sitzung des Anwenders für böswillige Aktionen ausnutzen. „Cross-site request forgery (also known as XSRF or CSRF) is an attack against web-hosted apps whereby a malicious web app can influence the interaction between a client browser and a web app that trusts that browser.“ (Hasan, Anderson, & Smith, 2023). Um dies zu ermöglichen, wird im Programm nach dem Rendern einer Seite besagter Token für die aktuelle Sitzung gesetzt.

Die eigentlichen Funktionalitäten zum Einloggen oder Registrieren eines Benutzers sind in einer eigenen Klasse ausgelagert. In den Views werden lediglich die eingegebenen Daten an öffentlich zugängliche Methoden jener Klasse übergeben. Dort folgt ein Aufruf privater Methoden, welche für die Überprüfung und Verarbeitung sorgen.

Hashen des Passwortes
Beim Erstellen eines neuen Users wird nach Validierung der angegebenen Werte das Passwort, welches zu diesem Zeitpunkt noch als Plain-Text vorliegt, vor der Speicherung in der Datenbank in einen Hash umgewandelt. Als Hash wird in der Kryptographie die Konvertierung eines gegebenen Textes, durch mathematische Berechnungen mithilfe eines Algorithmus, in eine nicht rückrechenbare Zeichenkette, mit fix vordefinierter Länge verstanden, die auf den Ursprungstext nicht zurückschließen lässt. Für dieses Projekt steht der Hash-Algorithmus namens „Rfc2898DeriveBytes“ im Einsatz. Weiters wird beim Verschlüsseln des Passwortes ein „Salt“ hinzugegeben, um für extra Sicherheit zu sorgen. Dieses wird vor Beginn des eigentlichen Passwortes beigefügt, wodurch sich für Cyberkriminelle nicht erschließen lässt, welcher Teil des Hashs zum Passwort, und welcher zum Salt gehört.

UserID	UserName	Password
2	bknoll	BVHAUBXKGIa70tAO5qcE1PR0m8Cw+h85JALK6+6lHN90Y/hfjamIoxLnDmiXRPLPp1fgcpZWlM737qgx5JD...
5	tvisit	U6xFO6cqpiH9tVYKLh2po34bmrRMguCXsVr1TuLxCizg6otye/0iJOd0il8QOmVjcyZQVDL0HfGxq+2WQMF5Sr...
6	hbue	Cro6f/oHPefgSxDIpXA4IvsN1FyXIF7RaFLBlxlTTQQ19JsByZuS5HMJFpp5r74nnQEpSk0zApIsiTpLn0VSwp11...

Abbildung 4: Passwort-Hashes

Login

Anmeldungen erfolgen entweder mittels E-Mail-Adresse oder Benutzername und Passwort des Anwenders. Da in der Datenbank nur der Hash des Passwortes hinterlegt ist, muss beim Login ebenfalls ein Hash erzeugt und mit dem Datensatz verglichen werden. Bei diesem Vorgang wird das Salt vom Hash weggerechnet. Möglich ist dies, da die Länge des Salt dem Entwickler im Programmcode bekannt ist.

Mittels setzen eines Authentifizierungs-Cookies, welcher im Browser für die aktuelle Sitzung gespeichert wird und nach spätestens fünf Stunden abläuft, kann die Applikation den angemeldeten Zustand erkennen, und auf die im Cookie gespeicherten Informationen – wie etwa den Benutzernamen oder das Recht des Benutzers – zugreifen. Somit können hinter einem Recht geschützte Funktionen oder Unterseiten aufgerufen werden, sofern der angemeldete User die nötigen Spezifikationen erfüllt.

Das Ausloggen erfolgt über ein Form-Element im HTML, welches eine Anfrage an die Logout-Seite schickt. Wichtig ist, der Form ein verstecktes Element mit dem Anti-Forgery Token als Übergabewert anzuhängen, da das Programm diesen Token zur Überprüfung auf die Echtheit des Clients benötigt bzw. um festzustellen, dass die Anfrage nicht durch Dritte getätigt wurde.

```
<form method="post" action="/logout">
    <input type="hidden" name="__RequestVerificationToken" value="@_tokenProvider.AntiForgeryToken">
    <input class="menuIcon iconLogoutCyan backTrans noLine vp0 hp0 pointer" type="submit" value="">
</form>
```

Abbildung 5: Logout-Form

5.4. Views

Views sind letztendlich die tatsächlichen Unterseiten der Website und setzen sich aus einem Layout, welches die Grundstruktur zusammengehöriger Views definiert, einer für den Anwender interaktiven Razor-Komponente, bestehend aus HTML-Code vermischt mit Razor-Syntax, und C#-Code im Hintergrund, um benötigte Daten zu selektieren und die Seite dynamisch zu machen, zusammen. Jede View erfordert zwingend mindestens eine Routing-Angabe, die vorgibt, wie die Unterseite aufgerufen werden kann. Es ist auch möglich, dieselbe Seite über verschiedene Verlinkungen zu erreichen. Als nützlich erweist sich das speziell dann, wenn die ausgegebenen Daten von gewissen Werten abhängig sind. In diesem Fall werden jene Schlüsselwerte als Parameter in der URL übergeben.

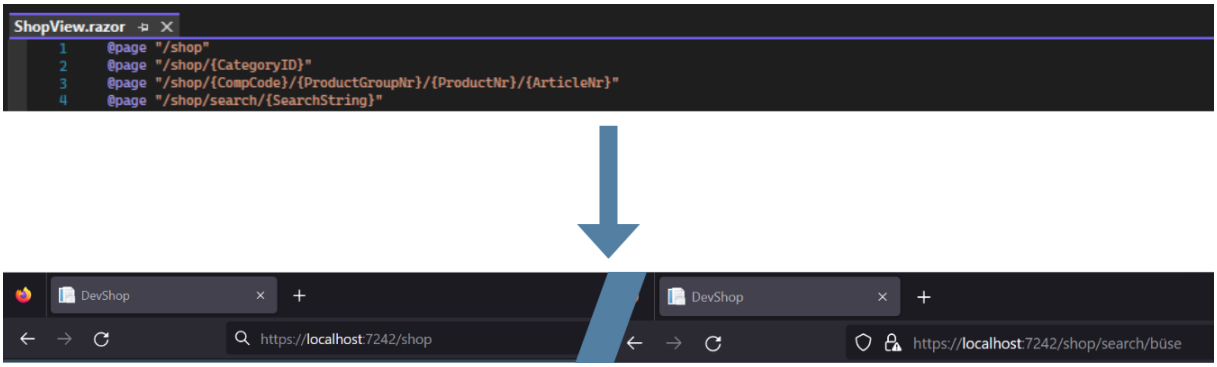

Abbildung 6: Routing

Layouts

Alle Unterseiten, die zum Shop gehören, sollen in derselben Struktur aufgebaut sein. Selbes gilt auch für jene Views, die der Verwaltungsoberfläche gewidmet sind, und aus diesem Grund anders formatiert werden sollen, als die zum Frontend gehörigen Seiten. So ist beispielsweise ganz oben auf der Website ein Menü, darunter die Ausgabe des Hauptinhalts, gefolgt von einem Fußbereich am Ende vorgesehen. Hierbei ändern sich das Menü und der Fußbereich nie und sollen auf jeder Unterseite gleich sein. Da kommen die Layouts ins Spiel. Sie fungieren als Templates, indem sie den allgemeinen HTML-Code beinhalten. An der Stelle im Code des Layouts, an jener der Inhalt der jeweiligen View angezeigt werden soll, wird dies mit einem einfachen Befehl „**@Body**“ ermöglicht.

Standardmäßig ist in den Einstellungen der Applikation ein bestimmtes Layout hinterlegt, welches automatisch für jede View verwendet wird. Ist jedoch ein anderes Layout gewünscht, kann dies mithilfe der Direktive „**@layout** *LayoutName*“ in der Razor-Komponente geändert werden.

Sowohl in der Formatierung für das Frontend als auch im Layout der Verwaltungsoberfläche des Backend wird an oberster Stelle jeder Unterseite ein Hauptmenü eingebunden. Der HTML-Code beider Menüs wird allerdings nicht direkt in die jeweiligen Layouts geschrieben, sondern

in eigenen Razor-Komponenten ausgelagert und an den gewünschten Stellen nur mehr eingebunden. Vorteil der Trennung ist einerseits die Wiederverwendbarkeit des Codes auch in anderen Dateien, ohne dass eine repetitive Programmierung erfolgen muss, und andererseits ermöglicht dies eine einfache und schnelle Wartung, sollte die Position des Menüs geändert werden.

CSS

Standardmäßig bietet Blazor die Möglichkeit, eine Bootstrap-Library für das Stylen der Website einzubinden. „DevShop“ verzichtet gänzlich auf externe Libraries und Styles, da diese extrem viele Dateien auf allen Seiten der Anwendung einbinden würden und somit den Daten-Traffic signifikant erhöhen, was in weiterer Folge eine verringerte Ladegeschwindigkeit bedeutet. Stattdessen werden auf der gesamten Website verteilt ausschließlich drei eigens codierte CSS-Dateien eingebunden, und das nur dort, wo sie auch zum Einsatz kommen. Grund hinter der Aufteilung in mehrere Dateien, statt eine Große vorliegen zu haben, ist, dass auf diese Weise nur die für die aufgerufene Unterseite benötigten CSS-Anweisungen geladen werden.

Daraus erschließt sich eine Datei für allgemeine Regeln, welche auf der gesamten Seite von Nöten ist, unter Anderem bestehend aus Definition der Schriftart- und Größe, Bestimmung von Abständen und Festlegung aller Farbcodes bzw. alles, was sowohl im Frontend als auch im Backend von Gebrauch ist. Eine Zweite beinhaltet nur Anweisungen, die mit dem eigentlichen Shop in Verbindung stehen, während die letzte CSS-Datei für das Stylen des Login-geschützten Verwaltungsbereiches zuständig ist. Selbstverständlich ist die gesamte Website responsive und demnach auf jedem Gerät – ob PC, Tablet oder Mobiltelefon – schön, übersichtlich und problemlos bedienbar.

JS

Blazor schickt Anfragen vom Client über eine Open-Source-Library namens „SignalR“, welche das Hinzufügen von Echtzeitwebfunktionen zu Apps vereinfacht, an den Server. Diese Funktionen „ermöglichen serverseitigen Code, Inhalte sofort an Clients zu übertragen“ (Gaster, et al., 2023), was mithilfe einer absolut notwendigen JS-Datei erfolgt, ohne die das Framework nicht funktioniert. Neben besagter JavaScript-Datei des Blazor-Frameworks ist noch eine Selbstprogrammierte in Verwendung, um das Öffnen und Schließen von Popups zu steuern.

6. Anwendung des UI

Neben einem klar strukturierten und übersichtlich eingeteilten Code für den Entwickler, ist eine saubere Bedienoberfläche mit simpler und gut nachvollziehbarer Anwendung für den Endbenutzer genauso von Bedeutung. Knackpunkt hierbei ist es, jegliche Funktionalitäten so einfach wie möglich zu gestalten, um Verwirrungen oder Schwierigkeiten in der Handhabung vorzubeugen.

6.1. Führung des Frontend

Einstiegsseite
Zu Beginn bekommt der Besucher eine Willkommensnachricht gefolgt von einer Suchleiste und zehn zufällig gewählten Artikeln zu Gesicht. Beim neu Laden der Seite werden die Artikel erneut nach einem Zufallsprinzip selektiert und ausgegeben. Auf diese Weise soll ein Vorgeschmack auf das bestehende Sortiment gegeben werden.

Nun bietet sich die Option, nach einem gewünschten Artikel mittels Namens, Artikelnummer, oder EAN-Code zu suchen. Weiters wird, falls vorhanden, die Beschreibung sowie Merkmale eines Artikels auf den eingegebenen Suchwert geprüft. Ein Klick auf das Lupe-Icon leitet den Anwender auf eine Ausgabeseite mit allen Datensätzen, die dem Suchkriterium entsprechen, weiter.

Eine weitere Möglichkeit ist, auf einen der zufälligen Artikel-Boxen zu klicken, um so direkt die Details des ausgewählten Eintrages begutachten zu können, oder ähnliche Artikel derselben Kategorie zu durchstöbern. Der auf der linken Seite fixierte Button öffnet ein Menü, welches eine Auswahl aller verfügbaren Kategorien beinhaltet.

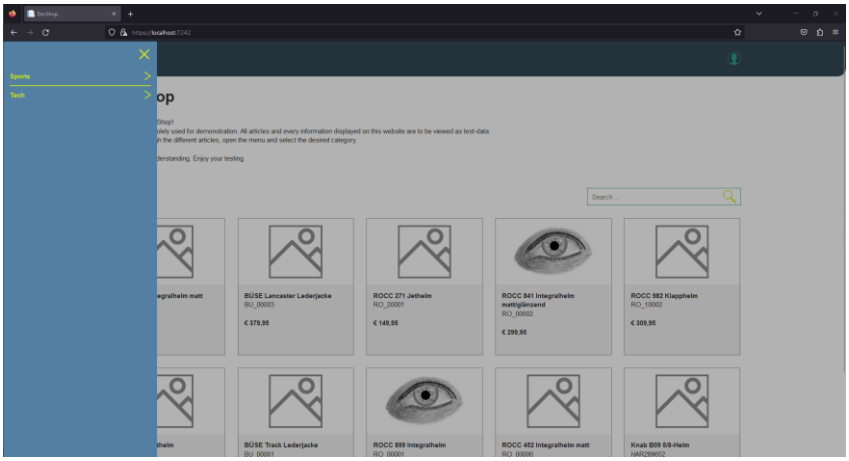

Abbildung 7: Shop – Einstiegsseite

Artikelausgabe

In der Ausgabe aller Artikel der gewählten Kategorie bzw. des gesuchten Wertes, kann zwischen zwei verschiedenen Darstellungen gewechselt werden – eine Boxenansicht und eine Listenansicht. Sobald die Ansicht geändert wird, wird ein Cookie im Browser gesetzt, welches sich die gewünschte Darstellung merkt, damit die Applikation diese nach Wechseln auf eine andere Unterseite wieder wählt.

Vorteil der Boxen ist, dass das Hauptaugenmerk auf den Bildern der jeweiligen Artikel liegt und somit auf ersten Blick sofort erkennbar ist, worum es sich handelt. Hinzu kommt, dass nur die wichtigsten Informationen präsent sind, detaillierte Beschreibungen sucht der Anwender in dieser Darstellung vergebens.

Im Kontrast dazu legt die Listenansicht Wert darauf, den Besucher schnell über alle Details zu informieren. Als besonders nützlich erweist sie sich, wenn für die vorhandenen Artikel noch keine Bilder hinterlegt sind, da diese im Gegensatz zur Boxen-Darstellung nur nebensächlich sind und klein dargestellt werden. Um die Seite kurz und übersichtlich zu halten, sind die Beschreibungen der Listen ausgeblendet und können mit Klick auf den Pfeil ganz rechts aufgeklappt werden.

Findet das Programm mehr als 20 Artikel, die ausgegeben werden sollen, erscheint eine Möglichkeit zum Wechseln auf die nächste bzw. vorige Seite der Ausgabe. Hiermit wird verhindert, dass der User mit Daten überflutet wird und die Seitenlänge kein Ende nimmt. Die Ladegeschwindigkeit wird dadurch ebenfalls optimiert. Klickt der Benutzer auf den Namen oder das Bild eines Artikels, wird er auf eine Detailseite geführt.

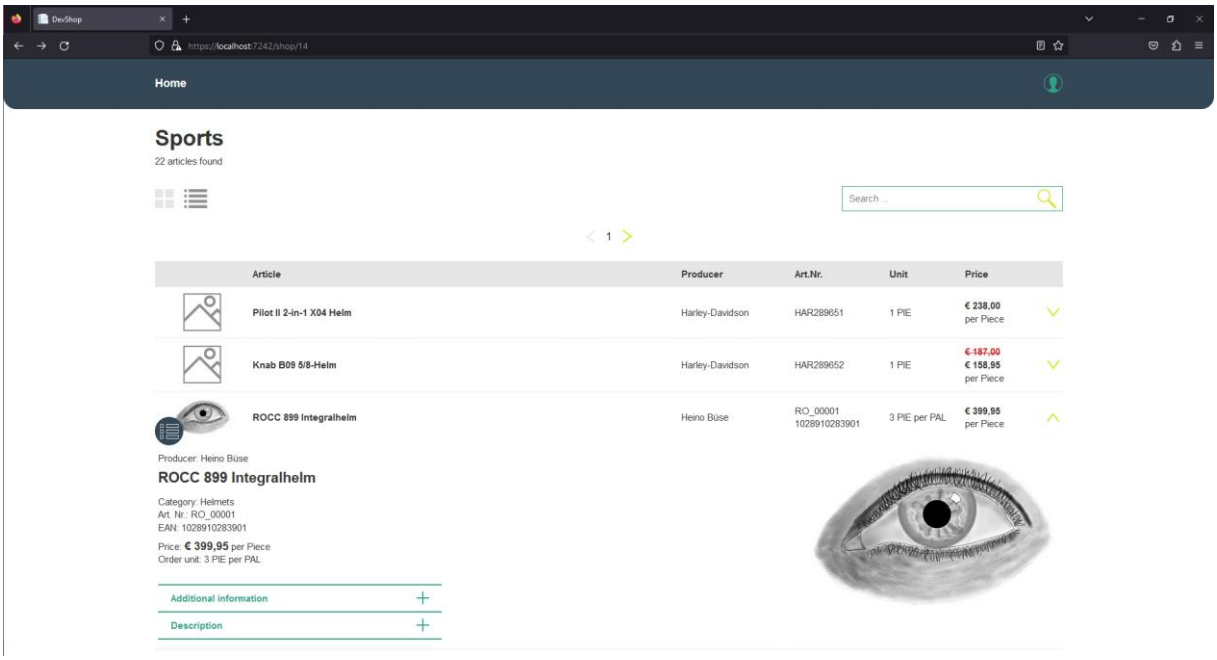

Abbildung 8: Shop – Listenansicht

Detailansicht

Die Detailansicht stellt alle Informationen zu einem gewählten Artikel inklusive großer Abbildung des Artikelbildes dar. Falls es gewünscht ist, das Bild genauer zu begutachten, kann dies mit einem Mausklick auf die Grafik erfolgen, wodurch sich diese in einem Popup in noch größerem Maß öffnet. Um bei Bedarf Platz zu sparen, befindet sich die Beschreibung in einem Dropdown und kann somit nach Belieben geöffnet oder geschlossen werden.

Unterhalb der Detailausgabe befinden sich alle weiteren Artikel, die zur selben Kategorie gehören bzw. die das Suchkriterium erfüllen.

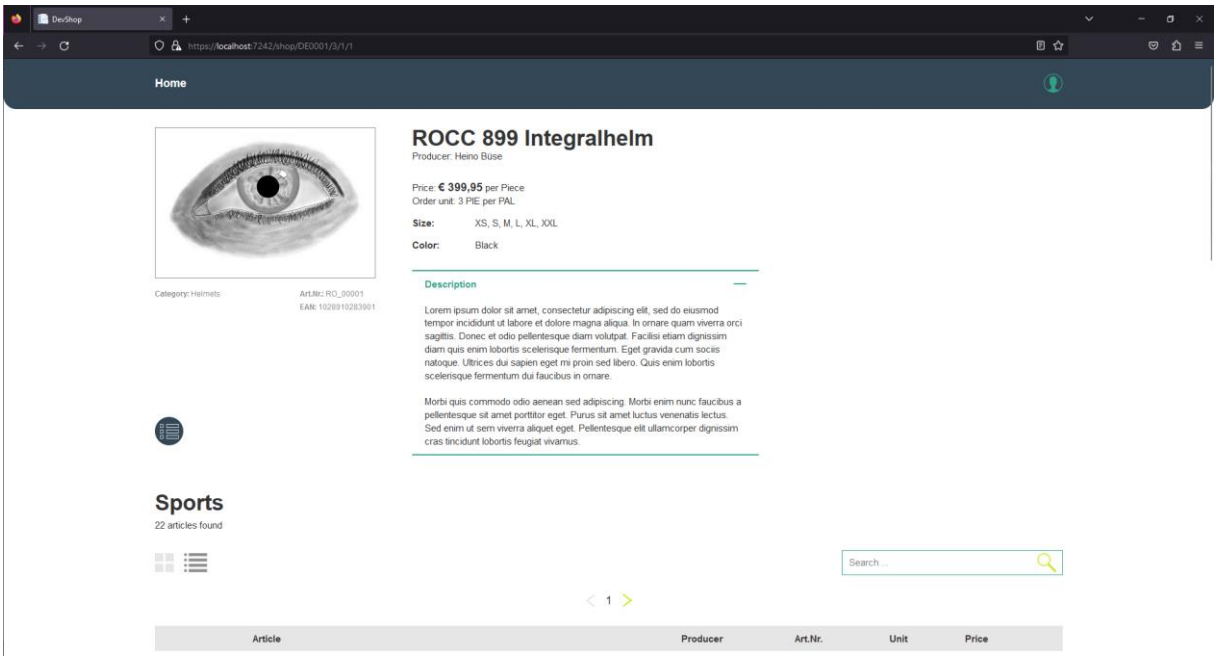

Abbildung 9: Shop - Detailansicht

6.2. Handhabung des Backend

Um in die Verwaltungsoberfläche der Website zu gelangen, muss sich der Anwender mit einem Benutzer anmelden, der über Admin-Rechte verfügt. Sollte für sich selbst noch kein Benutzer vorliegen, muss sich, falls vorhanden, an den Zuständigen der eigenen Firma oder dem Systemadministrator der Website gewendet werden, da aufgrund der erforderlichen Angabe mehrerer kritischer Faktoren eine eigenständige Registrierung nicht durchgeführt werden darf. Vorzufinden ist der Login im Menü auf der rechten Seite. Nach erfolgreicher Anmeldung erscheint im Menü ein zusätzlicher Punkt „Management“, welcher zum Backend führt.

Einstiegsseite

An der Einstiegsseite angekommen bekommt der Benutzer direkt eine grobe Übersicht mit Beschreibung zu den einzelnen Unterteilungen der Verwaltung. Die Eingabemasken sind gruppiert nach Benutzer-/Firmenverwaltung, Angabe der Standorte und Erstellung der Artikel.

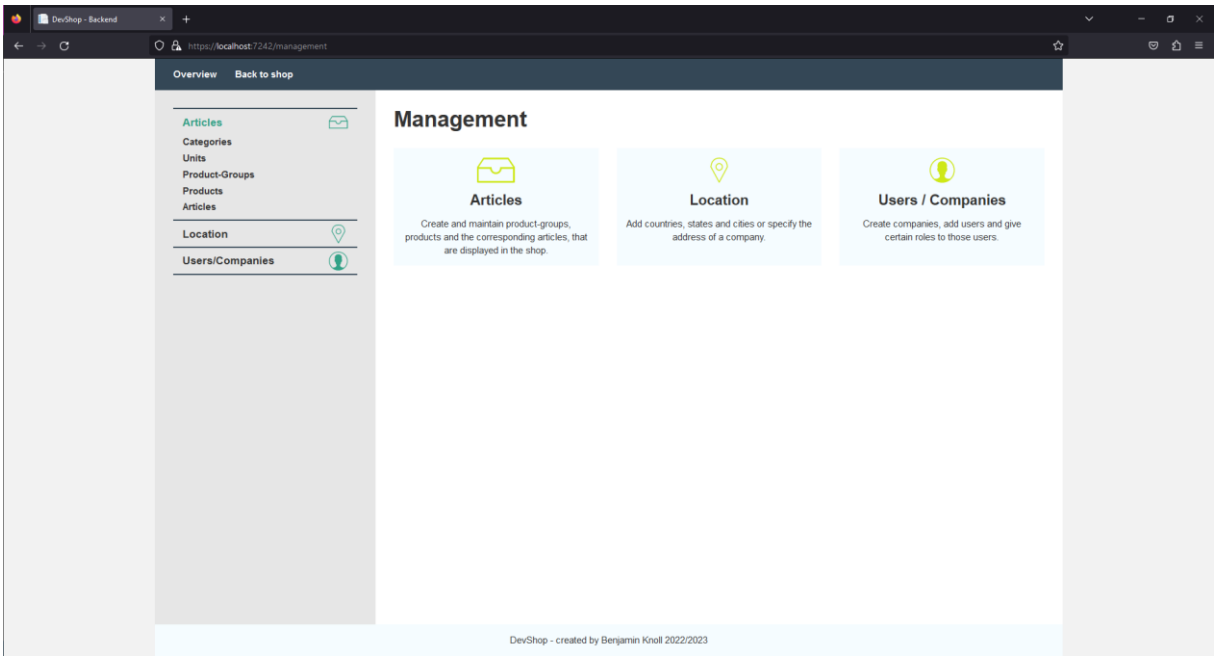

Abbildung 10: Verwaltung – Einstiegsseite

Standorte

Sinn hinter der Standort-Verwaltung ist nicht nur, dass dadurch das Land, in dem sich ein Hersteller befindet, festgelegt werden kann, sondern sie dient auch der Bestimmung der exakten Adresse eines Unternehmens oder eines Benutzers. Die Verwaltung teilt sich hierbei in drei Eingabemasken auf, jeweils zum Festlegen des Landes, Bundeslandes und der Stadt.

Benutzer/Firmen

Um einen Benutzer erstellen zu können, bedarf es einer Firma, der der User angehört. Ein Unternehmen kann dann erstellt werden, wenn bereits ein Land vorhanden ist. Der

einzigartige Firmen-Code setzt sich nämlich aus dem Länderkürzel und der Menge an bereits vorhandenen Unternehmen im ausgewählten Land zusammen.
Beispielsweise bekommt eine Firma, die in Österreich residiert, als Länderkürzel „AT“ und, angenommen sie sei die Vierte, die für dieses Land erstellt wird, die Nummer „0004“. Der zusammengesetzte Firmen-Code wäre in diesem Fall „AT0004“. Wird aber nun ein Unternehmen aus Deutschland angelegt, und es sind für dieses Land noch keine weiteren vorhanden, bekommt dieses den Code „DE0001“. Die Höhe der Nummer beginnt pro Land immer bei „0001“.
Ein weiterer wichtiger Punkt, der nicht übersehen werden darf, ist die Option, eine Firma als Hersteller zu markieren. Nur als Hersteller gekennzeichnete Unternehmen können Artikel vertreiben.

Da jedem Benutzer ein Recht zugewiesen werden muss, ist es unumgänglich, dieses im Vorhinein festzulegen. Dabei bestimmt die Nummer der Rolle gleichzeitig die Höhe des Rechtes. Je höher, desto mehr Zugriff hat der User auf das Backend bzw. auf Zusatzfunktionen. Nach Erstellung kann diese Nummer nicht mehr geändert werden.

Angelegte Benutzer dienen zum einen dazu, anderen Personen den Zugriff auf das Backend zu ermöglichen, und zum anderen kann jedem User ein individueller Rabatt auf alle Artikel versehen werden. Auch, wenn kein Zugriff auf die Verwaltung gewährleistet wird, hat es somit trotzdem Vorteile, einen Zugang zur Website zu besitzen.

Artikelverwaltung
Zuallererst müssen Kategorien festgelegt werden, denen bestimmte Artikel angehörig sind. Hierbei kann der Admin beliebig viele Unterkategorien erstellen, wodurch die Liste aller Bestehenden in einer Baumstruktur dargestellt wird. Zum Bearbeiten eines bereits existierenden Eintrages muss lediglich auf den Namen geklickt werden.

Folgend auf den Kategorien kommen die Einheiten. Ein Artikel kann beispielsweise in Stück, Liter oder Kilogramm angegeben werden. Nach Erstellung einer Einheit, ist es nicht mehr möglich, die abgekürzte Schreibweise dieser zu ändern.

Nun können Produkt-Gruppen erstellt werden. Wie der Name bereits vermuten lässt, dienen sie dazu, zusammengehörige oder ähnliche Produkte zu gruppieren. Auch hier ist es möglich, beliebig viele Untergruppen anzulegen, wobei aufgrund der Übersicht davon abzuraten ist, unnötige Verschachtelungen vorzunehmen. Besonderheit der für die Gruppen dargestellten Baumstruktur ist, dass diese nicht nur Produkt-Gruppen und Untergruppen enthält, sondern auch die dazugehörigen Produkte und Artikel gleich mit unterordnet. Dadurch wird ein schneller Wechsel zwischen den drei Bereichen gewährleistet, ohne jedes Mal im seitlichen Menü den gewünschten Punkt auswählen zu müssen.

Produkte werden auf dieselbe Art und Weise angelegt, wie Gruppen, mit dem Unterschied, dass jedem Produkt ein Artikel-Header zugewiesen werden muss. Dieser gibt nämlich individuelle Eigenschaften aller zum Produkt gehörigen Artikel an.

Nach diesem Schritt kann der Anwender die Artikel, die letztendlich im Frontend angezeigt werden, anlegen und sie mit einem Bild versehen. Nur dann, wenn ein bestehender Artikel bearbeitet wird, kann ein Bild hochgeladen werden, beim Erstellen eines neuen Datensatzes ist dies nicht möglich.

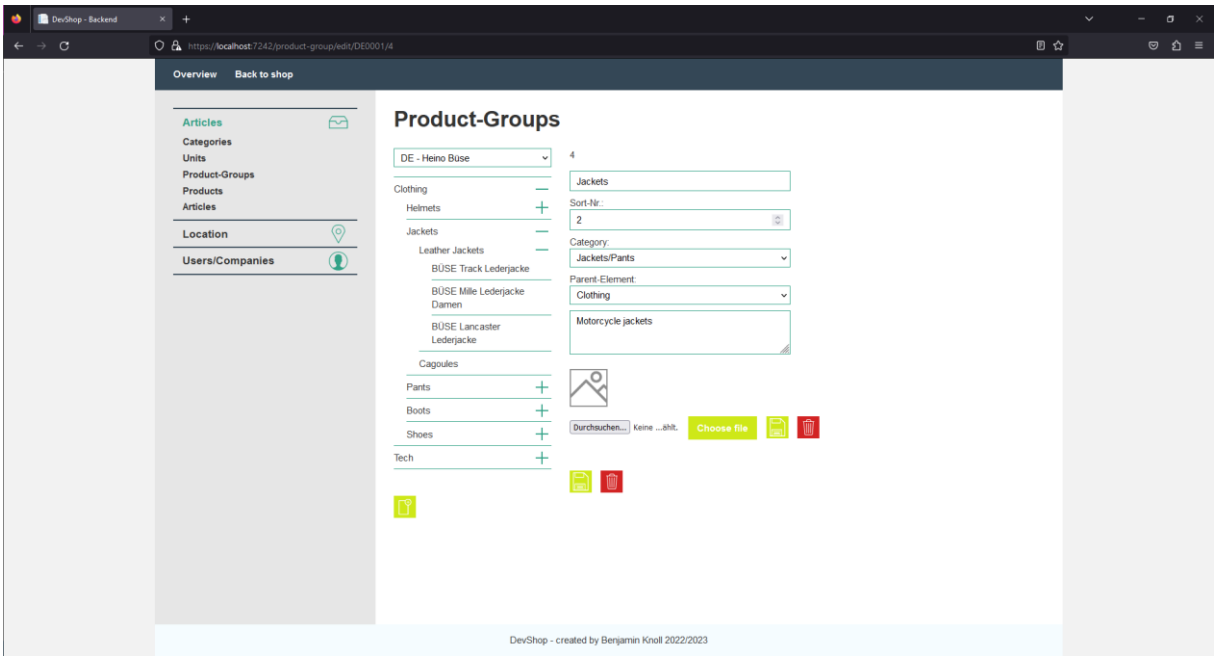

Abbildung 11: Verwaltung - Produkt-Gruppen

7. Fazit

Das Hauptziel des Projektes, sich mehr Wissen über die Backend-Entwicklung eines Shops anzueignen, ist zweifelsohne erfüllt. Neben dem bereits vorhanden gewesenem Wissen zu Datenbanken, HTML und CSS, aber auch den Programmierkenntnissen in C#, war der Lerneffekt sehr groß und deckt unter anderem die komplette Implementierung eines Authentifizierungsprozesses, sowie die Logik hinter üblichen Funktionen eines Shops, beispielsweise das Vor- und Zurückblättern auf mehrere Seiten, ab.

Jedoch sorgte der zu breit gewählte Umfang des Projektes für eine Entwicklungszeit von ca. 50 bis 60 Stunden. Somit mussten einige Features, wie die Einstellung individueller Benutzer-Rabatte oder eine detaillierte Ausgabe zu den Standorten der Hersteller inklusive Karte, bedauerlicherweise gestrichen werden.

„DevShop“ wird in Zukunft noch weiterentwickelt und optimiert. Auch alle entlassenen Funktionalitäten werden nachgepflegt. Bislang ist nicht in Aussicht, die Applikation für Kommerzielle Zwecke zu nützen, doch es ist nie gewiss, was die Zukunft alles mit sich bringt.

8. Literaturverzeichnis

Doberenz, W., Gewinnus, T., Kotz, J., & Saumweber, W. (2018). *Visual C# 2017 - Grundlagen, Profiwissen und Rezepte.* München: Carl Hanser Verlag.

Gaster, B., Buck, A., Latham, L., Pickett, W., Pine, D., Zhongke, Y., . . . Appel, R. (22. Januar 2023). *Übersicht über ASP.NET CoreSignalR*. Von https://learn.microsoft.com: https://learn.microsoft.com/de-de/aspnet/core/signalr/introduction?WT.mc_id=dotnet-35129-website&view=aspnetcore-6.0 abgerufen

Hasan, F., Anderson, R., & Smith, S. (21. Januar 2023). *Prevent Cross-Site Request Forgery (XSRF/CSRF) attacks in ASP.NET Core*. Von https://learn.microsoft.com: https://learn.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-6.0 abgerufen

Hosting-Technik. (12. Januar 2023). *Normalisierung von Datenbanken*. Von https://www.ionos.de: https://www.ionos.de/digitalguide/hosting/hosting-technik/normalisierung-von-datenbanken/ abgerufen

9. Abbildungsverzeichnis