



ПРОФИЛИРАНА МАТЕМАТИЧЕСКА ГИМНАЗИЯ
„КОНСТАНТИН ВЕЛИЧКОВ“ ГР.ПАЗАРДЖИК

ДИПЛОМНА РАБОТА

Тема: Разработка на уеб приложение „ArtFusStudio“

**Дипломен проект за придобиване на трета степен на професионална
квалификация – част от теорията на професията**

Дипломант: Стефан Георгиев Маринков

Специалност: „Приложно програмиране“ код 481030

Професия: „Приложен програмист“ код 4810301

Консултант:

Пазарджик

2024г.

Съдържание:

1.	Увод	4
2.	Основна част	5
2.1	Цел и задачи на дипломната работа.....	6
2.2	Използвани технологии	6
2.3	Реализация на проекта	7
2.3.1	Създаване на проекта и потребителски настройки	7
2.3.2	Модели (Models).....	11
2.3.3	Таблицы	24
2.3.4	Удостоверяване и упълномощаване в приложението.....	27
2.3.5	Контролери (Controllers).....	28
2.3.6	Изгледи (Views)	32
2.3.7	JavaScript (JS).....	35
3.	Заключение.....	36
4.	Приложения	38
	ПРИЛОЖЕНИЕ 1: Имплементирането на метода FormatDecimal в ArtFusStudio.Utility	38
	ПРИЛОЖЕНИЕ 2: Архитектура на ArtFusStudio.Models.....	39
	ПРИЛОЖЕНИЕ 3: Архитектурата на модела „Brands“	40
	ПРИЛОЖЕНИЕ 4: Архитектурата на модела „Category“.....	41
	ПРИЛОЖЕНИЕ 5: Архитектурата на модела „Product“	42
	ПРИЛОЖЕНИЕ 6: Имплементация на атрибута за валидиране на данни.....	43
	ПРИЛОЖЕНИЕ 7: Връзките между таблицата Product чрез външни ключове	44
	ПРИЛОЖЕНИЕ 8: Подробно описание на таблицата Phone	45

ПРИЛОЖЕНИЕ 9: Част от методите на <code>DisplayLayoutController</code>	46
ПРИЛОЖЕНИЕ 10: Показване на данни от няколко таблици наведнъж	47
ПРИЛОЖЕНИЕ 11: Използване на търсачката за Phone.....	48
ПРИЛОЖЕНИЕ 12: Имплементиране на търсачката	49
5. Използвана литература	50
Наръчник за потребителя.....	51

1. Увод

През последното десетилетие се наблюдава нарастване на връзката между бизнеса и новите технологии. В забързаното ежедневие човек има все по малко време, което може да отдели за търсене и оглед на артикули, които да закупи, затова предпочита възможността да търси от вкъщи, което значително съкращава времето и енергията, необходими за намиране на желаната стока. Затова все повече компании се насочват към дигиталното пространство където да предлагат техните стоки, понеже така достигат до по-голяма аудитория. Така човекът от вкъщи може да се запознае с всички предмети които съответното лице или фирма предлага за отрицателно време.

Технологиите, които се използват за създаване на връзка с компанията в глобалната мрежа, се усъвършенстват постоянно. Телефоните станаха неизменна част от ежедневието ни и все повече хора предпочитат тях заради мобилността им пред компютрите. Уникалността в отношението на дължината ѝ височината на екрана на различните устройства става един от водещите критерии за уеб сайтовете.

2. Основна част

Проектът представлява интерактивно динамично уеб приложение с приятен за потребителя интерфейс, който притежава множество функционалности и предоставя на купувачите подробна информация за предметите, който се показват в сайта.

Сайтът е разделен на публична и скрита част, според това дали даденият акаунт е на потребител или на администратор. Публичния дял съдържа няколко категории – „Помощ” (информация), „Промоции“, секция за запазени артикули, списък за избиране на език, на който да се представи съдържанието и пазарска количка, в която потребителят запазва предметите, които иска да закупи наведнъж. За хората с повече привилегии има категория „Администратор“, която съдържа връзки към панелите за манипулиране на основните таблици в базата данни - това са възможностите за добавяне на спецификации на продуктите, самите продукти и купони с намаления, както и за манипулиране на потребителските акаунти.

Уеб приложението изисква от потребителите да се регистрират и да влязат в акаунта си ако искат да използват функциите за запазване на предмети и извършване на покупка на няколко артикула едновременно. За покупка на една стока не се изисква клиентът да има потребителски акаунт. Данните, които се изискват при създаването на този профил, са електронна поща, потребителско име и сигурна парола.

За изграждането на това приложение е използвана N-слойна архитектура в съчетание с MVC архитектурата като във всеки един от трите ѝ основни слоя се намират компонентите на MVC

Като допълнителни функционалности, приложението ще включва възможности за филтриране и групиране на продуктите по определен критерий или група от такива, прикрепяне на изображения и работа с дати.

Завършващи дейности:

Писане на качествен програмен код, следейки добри практики и принципи на софтуерното инженерство;

Осигуряване на адаптивен дизайн за уеб сайта за еднакво добро изживяване на различни устройства;

Избор на технологии и инструменти, които да гарантират успешната реализация и бъдеща поддръжка на проекта.

Проектът цели създаването на иновативно уеб приложение, което не само ще задоволи онлайн потребностите на "ArtFusStudio", но и ще предложи ефективен и сигурен мениджмънт на магазина, осигурявайки конкурентно предимство в дигиталния свят.

2.1 Цел и задачи на дипломната работа

Целта на този проект е да участва в разрешаването на един от основните проблеми на съвременния консуматор, а именно недостигащото време. Когато всичко е *„на един клик разстояние“* и достъпно от всякакъв вид устройство, времето необходимо за търсенето и желания артикул намалява експоненциално. Другият основен проблем, който този проект се заема да реши е проблемът с избора - когато купувачът не може да обработи прекалено големия набор от информация, затова ArtFusStudio предлага решение като опцията за филтриране на всички предлагани в страницата на магазина продукти и да може да види всички които имат търсените от него спецификации.

Друг проблем, който това приложение решава, е този свързан с езиковата бариера между различните националности. Понеже уеб сайтовете са публично достъпни това означава че хора от цял свят могат да ги използват и като всеки един проект, за да се радва на все повече потребители, трябва да е достъпен на колкото се може повече езици, понеже не всеки говори английски.

2.2 Използвани технологии

ASP.NET Core - създаването на цялостната Back-end част, както и базата данни на Code-first приложението;

HTML5 - представяне на обектите от базата данни;

CSS3 - дизайн на уеб страницата (подобряване на потребителски интерфейс (UI));

JavaScript - подобряване на потребителското преживяване (UX);

Microsoft SQL - изграждането на релационна база данни (установяване на връзките между различните таблици чрез ключове);

EF Core - .NET технология, предназначена за работа с базата от данни посредством обекти (модели);

Bootstrap - използване на шаблони за да може сайтът да изглежда еднакво добре на устройства с различна разделителна способност;

Flexbox, Grid - CSS технологии, подобряващи UI;

Ajax - Добавя асинхронност към JavaScript кода;

NMT - невронен машинен превод, добавя възможността за цялостно превеждане на съдържанието на уеб сайта.

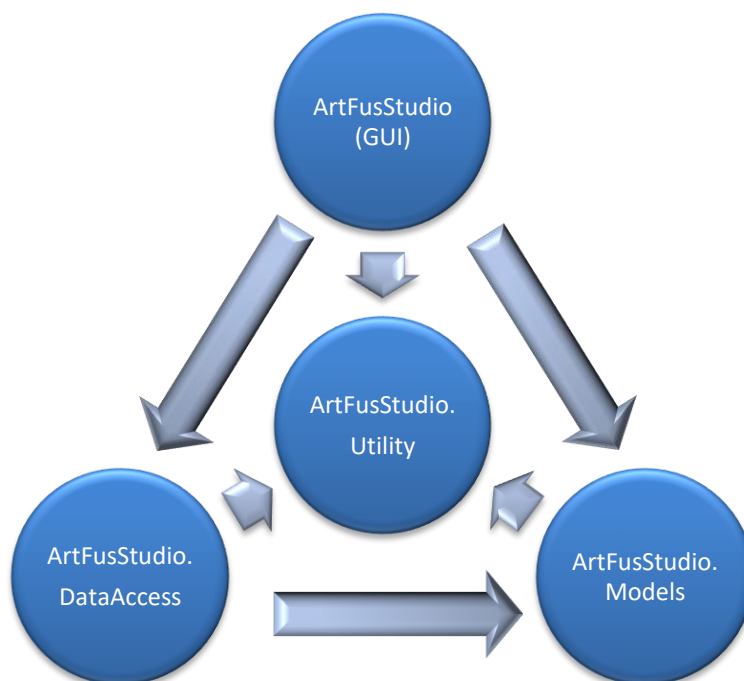
2.3 Реализация на проекта

2.3.1 Създаване на проекта и потребителски настройки

Уеб приложението е създадено на ASP.NET Core 8 като ASP.NET Core Web App (Model-View-Controller) и след това в този Solution са добавени 3 проекта от тип Class Library, със съответните връзки.

Проектът е реализиран, чрез използването на многослойна архитектура, в комбинация с MVC модела, като всеки от компонентите му е поставен в подходящия слой: ArtFusStudio (Front-end/Клиентски слой) съдържа кода, който ще се възпроизведе от уеб браузъра и има за цел да достигне до крайния потребител (интерфейсът на приложението). Там се намират изгледите и съответните им контролери, позиционирани в съответните им зони, като те биват 3 вида - за администратор (**Admin**), за нормален потребител (**Customer**) и за вписване (**Identity**), както и низът за връзка с базата данни **appsettings.json**; ArtFusStudio.Models съдържа моделите, изградени чрез подхода на обектно-ориентираното

програмиране с необходимите валидации за всяко от свойствата. В Back-end слоя (ArtFusStudio.DataAccess) се намират миграциите и кодовете, използващи ORM техниката за попълване на таблиците. Отделно е добавен и четвърти проект (ArtFusStudio.Utility), където са записани константните величини. Това ниво на абстракция което се получава с разбиването на четири части се използва за намаляване на сложността, както и да улесни поддръжката на приложението. Референциите между отделните проекти са показани в диаграмата по долу, където $A \rightarrow B$ означава, че A използва данни от B:



Фигура 1: Референциите между различните слоеве

За осъществяването на приложението е използвана Code-first техниката, която ни позволява да създаваме, манипулираме и като цяло поддържаме базата данни, чрез .NET кода. За презентационния и за слоя за данни са използвани пакети от семейството на Microsoft.EntityFrameworkCore (общи са SqlServer и Tools, Relational и Sqlite се използват при файловете за визуализация и Design при данните) и Microsoft.AspNetCore.Identity (EntityFrameworkCore и Identity за изгледите и Diagnostics. EntityFrameworkCore и Identity за манипулацията на данните), а за бизнес слоя е използван само Microsoft.Extensions.Identity.Stores.

Автоматично са добавени акаунти от двете групи – клиент и администратор с входни данни потребителско име и парола, съответно (user, User#1) и (admin, Admin#1), като за достъп до формата за вписване е необходимо само да се натисне върху някои от бутоните, които изискват потребителят да бъде влязъл в акаунта си, за да извършат действието си по предназначение, като бутонът за запазени артикули (изображение на сърце, след „Помощ“), бутонът „Добави в количката“, самата пазарска количка в горния десен ъгъл, профилната снимка по подразбиране или върху текста „Здравей, впиши се“. За излизане от акаунта е необходимо да се натисне върху профилната снимка или върху текста отляво на тази снимка където пише „Здравейте <потребителско име>“.

2.3.1.1 Използваните атрибути в моделите на допълнителните таблици са:

- KeyAttribute - показва че свойството е първичен ключ в таблицата;
- RequiredAttribute - указва, че стойността на полето е задължителна;
- RangeAttribute - фиксира интервал на позволените стойности за свойството;
- MaxLengthAttribute - указва максималната позволена дължина за въвеждане на масив или низ в свойство;
- StringLengthAttribute - задава минимален и максимален брой символи, които стойността на полето може да притежава;
- RangeAttribute - задължава числовата стойност на полето да бъде в конкретен диапазон;
- RegularExpressionAttribute - указва, че стойността на полето трябва да съответства на конкретен шаблон;
- DataTypeAttribute - указва формата, по който да се попълнят датите (до ден);
- DisplayFormatAttribute - указва формата, по който да се визуализират датите ;
- ValidationAttribute - базов клас, използван за създаването на критерии за работа с полетата от тип DateTime;
- ValidationResultAttribute - показва резултата от заявката за валидиране;
- EndDateValidationAttribute - създаден атрибут, който създава ограничение датата на изтичане да е след датата на създаване.

2.3.1.2 Използвани атрибути от System.ComponentModel.DataAnnotations.Schema:

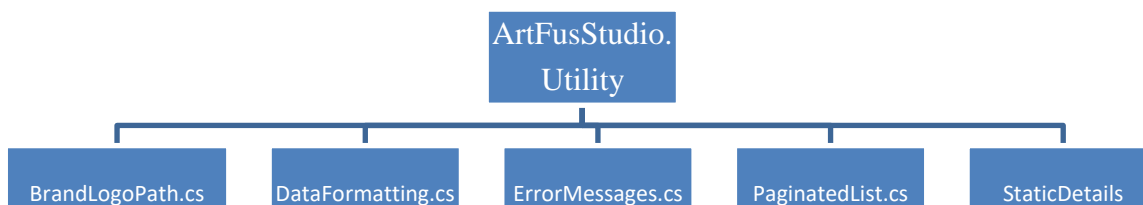
- 1) ForeignKeyAttribute – използва се за установяване на връзка между различни таблици от базата

2.3.1.3 Използвани помощни класове в моделите:

За да се улесни поддръжката на приложението, в ArtFusStudio.Utility са създадени пет помощни класа, където са запазени статичните данни. В ErrorMessage.cs се намират съобщенията за грешка, които да се използват при валидацията на свойствата на моделите. В BrandLogoPath.cs се намира SVG командите, които да създадат логото на съответната марка под формата на векторна графика, защото така ще имат по-висока разделителна способност от изображенията с растрена графика (JPEG и PNG). Тези данни се използват при попълването на базата данни от ArtFusStudio.DataAccess\ApplicationDbContext.cs. Имената на константите са съгласно правилата за именуване в ASP.NET. В DataFormatting.cs се намира метод който да представи дробните числа в съкратен вид (заменя „ „ с „ „, както и намалява символите след десетичната запетая на дробта, като изтрива всички нули след десетичната запетая. Използва се при представянето на данните, като звездите с оценка на съответен продукт.

(Прилагам: Приложение 1: Имплементирането на метода FormatDecimal в ArtFusStudio.Utility)

PaginatedList.cs се използва да не представят всичките данни наведнъж да са на страници с по N елемента във всяка страница. StaticDetails.cs е предназначен да запазва други статични данни в бъдещото развитие на проекта. Структурата на този проект е следната:



Фигура 2: Файловата система в ArtFusStudio.Utility

2.3.2 Модели (Models)

Файловете на моделите се намират в проекта ArtFusStudio.Models и са подредени в папки, според това с кои други модели са свързани и кои наследяват. Архитектурата на този проект е разгърната подробно в **Приложение 2**.

(Прилагам *Приложение 2: Архитектура на ArtFusStudio.Models*)

2.3.2.1 Модела ApplicationUser

Класът **ApplicationUser** наследява класът **IdentityUser**, който представя моделът на таблицата с данните на потребителите – таблицата **Users**.

2.3.2.2 Модела ImageExtension

Този модел е предназначен да съхранява най често използваните разширения на изображение. Има 2 свойства:

- 1) *Id* (8 – 9 ред) – това е уникален идентификатор за всяка марка която се добавя в базата от данни. То е от тип `int` (цяло число, не по голямо 2^{31}) и има атрибута:
 - *Key* - указва, че е това свойство е първичен ключ и се използва за установяване на връзка с различните таблици от базата.
- 2) *Name* (11 – 13 ред) – това е свойството, чрез което се достъпва името на марката. То е от тип `string` и има атрибутите:
 - *Required* - указва че задължително поле то трябва да получи някаква стойност;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 6 символа).

2.3.2.3 Модела UserProfilePics

Този модел е предназначен да съхранява най често използваните разширения на изображение. Има 2 свойства:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - *Key* атрибута.

- 2) *UserId* и *ImageExtId* (11 – 17 ред) - външни ключове за свързване на таблицата с таблицата за потребителя и за най-често използваните разширения при изображенията. Няма атрибут.
- 3) *FileExtension* (19 ред) - съхранява разширението на профилната снимка на всеки потребител.

2.3.2.4 Модела Brand

Класът **Brand** (Прилагам: *Приложение 3: Класът Brand*) представя моделът на таблицата с данните на марките на артикулите – таблицата **Brand**. Свойствата в него са:

- 1) *Id* (10 – 11 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Name* (13 – 15 ред) – това е свойството, чрез което се достъпва името на марката. То е от *min string* и има атрибутите:
 - Required атрибута;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 32 символа).
- 3) *pathD* (17 – 18 ред) – това е свойството, чрез което се достъпва логото на марката, което да се извика от *svg* тага. То е от *min string* и има атрибута:
 - Required – описано е в модела **Brand**, свойството *Name*.

Чрез този модел се създава таблица която да се съхраняват данните за всички модели които могат да се срещнат при продуктите в сайта. Тя се използва в лентата за навигация като се изпраща заявка до базата данни да всички записи.

2.3.2.5 Модела Category

Класът **Category** (Прилагам: *Приложение 4: Класът Category*) представя моделът на таблицата с данните на основните групи артикули – таблицата **Category**. Свойствата в него са:

- 1) *Id* (8 – 9 ред) - има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.

2) *Name* (11 – 12 ред) - има същият тип и предназначение като едноименното свойство виж предишния модел, както и има същия атрибут:

- Required атрибута;
- StringLength - указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 128 символа).

Чрез този модел се създава таблица която да се съхраняват данните за основните групи в които може да разделим продуктите в сайта.

2.3.2.6 Модела Product

(Прилагам: Приложение 4: Класът Product)

Това е един от фундаменталните модели в проекта, който е предназначен да се използва като базов клас във всички новосъздадени модели на категория от продукти. Използва се и за отстъпване на всички предмети, което се продават в сайта, без да е необходимо обединение на таблици. В него се намират свойствата които ще присъстват във всеки един продукт. Те са:

- 1) Свойството *Id* (10 – 11 ред) - има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута
- 2) Свойството *Name* (13 – 15 ред) - има същият тип и предназначение като едноименното свойство виж предишния модел, както и разполага със следните атрибути за валидиране на стойността му:
 - Required атрибута;
 - StringLength - има същите ограничения за дължина да стойността на полето като гореспоменатия атрибут със същото име.
- 3) *Description* (17 – 19 ред) - използва се за да се даде описание на всеки един продукт в рамките на съответните ограничения. Следователно е от тип string. Те са:
 - AllowNull атрибута - не е задължително продуктът да има описание;

- `StringLength` - указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 512 символа).
- 4) *ProductURL* (21 – 23 ред) - съхранява пътя до файла с изображението на продукта, което се намира в подпапка, която носи името на категорията от която е продуктът, в главната директория (`~/images/Products/<име на категорията>/<име на изображението>.png`). Аналогично е от тип `string`. Желателно е разширението на това изображение да бъде `PNG`, защото поддържа прозрачен фон. Това свойство разполага със следната валидация:
- `Required` атрибута;
 - `StringLength` - указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 64 символа).
- 5) *OldPrice* (25 – 27 ред) – съдържа данни за предишната стойност на артикула. Тя трябва да бъде от тип `decimal`. Тази стойност притежава следните ограничения:
- `AllowNull` атрибута;
 - `Range` - оказва че стойността на полето трябва да бъде между 0.0 и 100000.0.
- 6) *CurrentPrice* (29 – 31 ред) – съдържа данни за настоящата стойност на артикула. Тя трябва да бъде от тип `decimal`. Тази стойност притежава следните ограничения:
- `Required` атрибута;
 - `Range` - стойността на полето трябва да бъде във същия диапазон, както при предишното свойство.
- 7) ,7) *Свързване на таблицата с таблиците за категория и марка посредством ForeignKey атрибута* (32 – 41 ред): `CategoryId` и `BrandId` от тип `int`. С установяването на първите връзки, базата данни вече става релационна.
- 8) *TotalVotes* (43 – 44 ред) - съхранява общия брой гласове, които продуктът е получил от потребителите. Тя трябва да бъде от тип `int`. По подразбиране е зададена стойност 0, когато обект се създаде и наследи този клас той ще има 0 оценки. Тази стойност притежава следния атрибут:
- `AllowNull` атрибута;
- 9) *Score* (46 – 48 ред) – съхранява сумата от всички гласове, които продуктът е получил от потребителите. Тя трябва да бъде от тип `int`. По подразбиране е

зададена стойност 0, когато обект се създаде и наследи този клас той ще има 0 резултат. Тази стойност притежава следните атрибути:

- AllowNull атрибута;
- ScoreValidation (62 – 75 ред) – атрибут, който наследява **ValidationAttribute** и се използва за създаването на ново правило което гласи че стойността на полето трябва да е по-голяма или равна на стойността на полето *TotalVotes* и по-малка или равна на 5 * стойността на *същото поле*.

10) *Rating* (50 – 61 ред) - това поле се използва за изчисляването на оценката на всеки артикул като на него не може да се зададе стойност, той я получава по формулата $Round(\frac{\text{стойността на } Score}{\text{стойността на } TotalVotes}, 2)$ и се преформатира, чрез гореописания метод `FormatDecimal`.

11) *Quantity* (79 – 81 ред) - това свойство се използва за да запише колко продукта има в наличност. Атрибутите му за валидация са:

- Required атрибута;
- Range - стойността на полето трябва да бъде в диапазона от 0 до 1000.

2.3.2.7 Модела OperatingSystem

Product имат няколко класа, които го наследява. Един от тях е този за телефоните, но понеже един и същи модел телефон може да има няколко различни стойности за едно свойство таблиците от базата са нормализирани до нормална форма на Бойс-Код, което гласи че всяко от свойствата, което може да има няколко стойности за един и същи обект да бъдат отделни таблици и връзките към тях да се осъществяват чрез външни ключове :

1) *Id* (9 – 10 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:

- Key атрибута.

2) *OSName* (12 – 14 ред) – съхранява имената на операционните системи които могат да се свържат с определен телефон. Аналогично е от тип string Разполага със следните атрибути:

- Required атрибута;

- `StringLength` – указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 32 символа).

2.3.2.8 Модела `OperatingSystemVersion`

Операционната система освен име трябва да има и версия. Тази таблица има също само 2 свойства:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - `Key` атрибута.
- 2) *OSVersion* (11 – 12 ред) – съхранява евентуалните стойности за версия на операционна система. То трябва да бъде от тип `decimal`, както и има същия атрибут:
 - `Required` атрибута;
 - `RegularExpression` – задава шаблон за стойността на свойството: десетичната запетая да бъде „.“ и след нея да има до 2 цифри, напр.: 12.42
 - `Range` – оказва че стойността на полето трябва да бъде от 0 до 99

2.3.2.9 Модела `OSNameAndVersion`

Този модел представлява обединение на предишните два, като е свързан с тях посредством първичните им ключове и съдържанието което да отпечатва всички възможни комбинации от операционна система с версия. Свойствата, с които разполага са:

- 1) *Id* (9 – 10 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - `Key` атрибута.
- 2) *OSNameId* (12 – 14 ред) – установява връзка с таблицата съхраняваща имената на операционните системи с цел извличане на стойностите ѝ. Разполага единствено с един атрибут:
 - `ForeignKey` атрибута.
- 3) *OSVersionId* (16 – 18 ред) – установява връзка с таблицата съхраняваща версиите на операционните системи с цел извличане на стойностите ѝ. Разполага единствено с един атрибут:

- ForeignKey атрибута.
- 4) *OSNamePlusVersion* (20 – 26 ред) – използва се за да върне конкатенация между името на операционната система и всички версии които и съответстват. Няма никакви атрибути и стойността и не може да се задава отвън, понеже модификатори за достъп „set“ липсва.

2.3.2.10 Модела Camera

Този модел се намират възможните стойности, които да се зададат като брой камери, които един телефон може да има. Свойствата, характерни за модела са:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *CameraCount* (11 – 14 ред) – Съдържа най-често, използваните стойности за брой камери които всеки телефон може да има. То трябва да бъде е от тип byte. Има следните атрибути:
 - Required атрибута;
 - RegularExpression – задава шаблон за стойността на свойството: десетичната запетая да бъде „.“ И след нея да има до 2 цифри
 - Range – оказва че стойността на полето трябва да бъде от 0 до 99

2.3.2.11 Модела DisplayTechnology

В този модел се намират възможните стойности, които да се зададат като технология за екрана, които един телефон може да има. Характеристиките на този модел са:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Name* (11 – 13 ред) – има същият тип и предназначение, като едноименното свойство на предишните модели, както и има същия атрибут:
 - Required атрибута;

- `StringLength` – указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 32 символа).

2.3.2.12 Модела Memory

В този модел се намират възможните стойности, които да се зададат като RAM памет, които един телефон може да има. Притежава свойствата:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - `Key` атрибута.
- 2) *RAM* (11 – 13 ред) – съдържа най-често използваните стойности за капацитет RAM в гигабайтове, Следователно типът му трябва да бъде `int`, както и има следните атрибути:
 - `Required` атрибута;
 - `Range` – оказва че стойността на полето трябва да бъде от 0 до 32. За да те открие

2.3.2.13 Модела StorageCapacity

Този модел съдържа възможните стойности, които да се зададат като капацитет на хранилището, които един телефон може да има. Разполага със свойствата:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - `Key` атрибута.
- 2) *CapacityGB* (11 – 13 ред) – Свойството е от тип `decimal` и съдържа най-често използваните стойности за капацитет на хранилището в гигабайти, както и има следните атрибути:
 - `Required` атрибута;
 - `Range` – указва че стойността на полето трябва да бъде от 0 до 2048 (2ТБ).

2.3.2.14 Модела USB

В този модел се намират обичайните модели зарядно, съвместими с телефона. Има следните свойства:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Type* (11 – 13 ред) – запазва стойностите на най-често срещаните модели зарядно при телефоните. Разполага с атрибутите:
 - Required атрибута;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 32 символа).

2.3.2.15 Модела Phone

Този модел ще се използва за създаването на един от основните артикули, които ще се продават в магазина – а именно телефона. Понеже е от основните групи артикули, той наследява базовия клас *Product* и взема неговите свойства. Освен тях за тялото на класа се съдържат и още шест свойства – външните ключове на таблиците, съответстващи на модели от номер 7 до номер 12, които установяват то връзка от тип „много към едно“, т.е. много таблици свързани към една.

2.3.2.16 Модела CaseMaterial

В тези модели продължаваме практиката от предишните и създаваме модел за всяко от свойствата на класа, целящ да опише следващата група продукти – калъфа за телефон. *CaseMaterial* запазва най-често използваните стойности за материал, от който да е изработен калъфът. Свойствата които притежава са:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Name* (11 – 13 ред) – има същият тип и предназначение, като едноименното свойство на предишните модели, както и има същите атрибути:
 - Required атрибута;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 32 символа).

2.3.2.17 Модела CaseType

Тук се запазват най-често използваните видове калъфи, тоест да дали е затворен, само гръб или друг. Свойствата н класа са:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Name* (11 – 13 ред) – има същият тип и предназначение, като едноименното свойство на предишните модели, както и има същите атрибути:
 - Required атрибута;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 32 символа).

2.3.2.18 Модела ProtectionLevel

Тук се запазват нивото на защита, което калъфът предлага. Свойствата са обичайните за този тип модели:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Protection* (11 – 13 ред) – записва най често използваните нива на защита. Аналогично е от тип string. Атрибутите му са:
 - Required атрибута;
 - *StringLength* – указва че броя на символите на стойността на полето трябва да варира в конкретен и интервал (в случая между 2 и 32 символа).

2.3.2.19 Модела Case

Този модел ще се използва за създаването на един от основните артикули, които ще се продават в магазина – а именно калъфът. Понеже е от основните групи артикули, той наследява базовия клас *Product* и взема неговите свойства. Освен тях за тялото на класа се съдържат и още три свойства – външните ключове на таблиците, съответстващи на модели от номер 14 до номер 16, които установяват то връзка от тип „много към едно“, т.е. много таблици свързани към една.

2.3.2.20 Модела Charger

Този модел ще се използва за създаването на един от основните артикули, които ще се продават в магазина – а именно зарядното за телефон. Понеже е от основните групи артикули, той наследява базовия клас Product и взема неговите свойства. Освен тях за тялото на класа се съдържат и още три свойства – те са:

- 1) *OutputVoltage* (9 – 11 ред) – Свойството е от тип `int` и съдържа стойността на изходното напрежение във волтове. Атрибутите му са:
 - `Required` атрибута;
 - `Range` – оказва че стойността на полето трябва да бъде от 0 до 24.
- 2) *OutputCurrent* (13 – 15 ред) – Свойството е от тип `int` и съдържа стойността на изходното напрежение във волтове. Атрибутите му са:
 - `Required` атрибута;
 - `Range` – оказва че стойността на полето трябва да бъде от 0 до 24
- 3) *FastChargingSupport* (17 – 18 ред) – Стойността на това свойство е булева, като служи като индикатор дали това зарядно поддържа бързо зареждане. Разполага само с един атрибут:
 - `AllowNull` атрибута.

Със създаването на този модел те включва инициализацията и описването на продуктите. Сега започват допълнителните функционалности.

2.3.2.21 Модела ShoppingCartItem

След като потребителят си набележи желаните от него артикули идва ред на последната част от взаимодействието, а именно осъществяването на поръчка. Този модел ще описва стандартният обект в пазарската количка с необходимите свойства:

- 1) *Id* (9 – 10 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - `Key` атрибута.
- 2) *ProductId* и *CartId* (12 – 19 ред) – свързват продукта от пазарската количка със съответният му артикул в магазина, както и с количката, в която се намира.

- 3) *Quantity* (21 – 23 ред) – има същият тип и предназначение, като едноименното свойство на модела *Product*, както и има същите атрибути:
- 4) *Id* (9 – 10 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.

2.3.2.22 Модела *ShoppingCart*

След като потребителят си запазва желаните от него артикули идва ред на последната част от взаимодействието, а именно осъществяването на поръчка. На първо място трябва да се създаде пазарската количка като модел, където да отиват продуктите, които клиентът възнамерява да закупи. Свойствата в този модел са:

- 1) *Id* (8 – 9 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 2) *Id* (9 – 10 ред) – има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:
 - Key атрибута.
- 3) *UserId* (12 – 14 ред) – Създава се външен ключ за свързване на потребителя със съответната пазарска количка. Няма атрибути.
- 4) *Items* (15 ред) - Създаване на списък от *ShoppingCartItem*. Няма атрибути.
- 5) *TotalPrice* (18 – 30 ред) - използва се да изчисли общата сума която клиентът трябва да заплати.
- 6) *TotalQuantity* (32 – 36 ред) - използва се да покаже общия брой на всички артикули в пазарската количка.

2.3.2.23 Модела *Coupon*

Преди обаче да се финализира поръчката потребителят има опцията да въведе купони за отстъпка които да намалят цената. Свойствата на един такъв купон са:

- 1) *Id* (8 – 9 ред) - има същият тип и предназначение, като едноименното свойство на предишния модел, както и има същия атрибут:

- Key атрибута.
- 2) *DiscountCode* (12 – 14 ред) - това е кодът който потребителят изписва за да получи съответната отстъпка на желаните от него артикул. Типът му е string. Атрибутите за валидация тук са:
- Required атрибута;
 - StringLength - указва че броя на символите на стойността на полето трябва да варира в конкретен интервал (в случая между 2 и 32 символа).
- 3) *DiscountPercentage* (16 – 18 ред) - Тук се запазва колко процента е отстъпката на артикула със съответния код. Типът на свойството е decimal. Атрибутите за валидация тук са:
- Required атрибута;
 - Range - оказва че стойността на полето трябва да бъде от 0.0 до 99.9
- 4) *ProductId* (20 – 23 ред) - това е външният ключ, който се използва да вземе продукт по идентификатор, за когото да се отнася този код за отстъпка. Тук няма никакви атрибути.
- 5) *StartDate* (25 – 28 ред) - Тук се съхранява датата от която е валидна този талон за намаление. Типът на свойството е DateTime. Атрибутите за валидация тук са:
- Required атрибута;
 - DataType(DataType.Date) - указва формата в който да бъде датата при записването и визуализацията. В този случай се показва само датата, без часове;
 - DisplayFormat - задава форматът на представяне на дата. В този случай тя ще излиза във формата „ДД.ММ. ГГГГ“.
- 6) *EndDate* (25 – 28 ред) - Тук се съхранява датата до която е валидна този талон за намаление. Типът на свойството е DateTime. Атрибутите за валидация тук са:
- Required атрибута;
 - DataType(DataType.Date) - указва формата в който да бъде датата при записването и визуализацията. В този случай се показва само датата, без часове;
 - DisplayFormat - задава форматът на представяне на дата. В този случай тя ще излиза във формата

„ДД.ММ. ГГГГ“.

- EndDateValidation – (Прилагам: Приложение 5: Имплементация на атрибута за валидиране на дати) ръчно създадената атрибут за валидиране, който следи датата на изтичане на купона да бъде след датата, от която е валиден.

2.3.3 Таблици

Таблиците са създадени и попълнени чрез миграции. Миграцията на схема на бази данни представлява контролирани набори от промени, целящи да модифицират структурата на обектите в рамките на релационната база данни. Тя се използва за преход на схемата от едно състояние в друго, Като се гледа промените да са постепенни. Необходимо е да се създаде миграция когато се прави промяна в някои от моделите или искаме да попълним някоя от таблиците ѝ. В контекста на този проект миграциите са реализирани посредством EF Core 8, която представлява разбрах от технологии, които улесняват значително работата с данни, като повишават нивото на абстракция приех работа с тях, тоест създават се приложения с по-малко код. Самият преход от модели към таблици е възможен благодарение на парадигмата за обектно-релационното картографиране (ORM) - създава се база данни с виртуални обекти, която може да се използва от съответния език за програмиране, в случая C#.

В този проект имената на таблиците съответстват на имената на моделите и връзките между тях, отново, се както във моделите.

(Прилагам: Приложение 6: Връзките между таблицата Phone чрез външни ключове)

(Прилагам: Приложение 7: Подробно описание на таблицата Phone)

Стилът, който присъства в двете гореспоменати приложения, е същия за всяка таблица в схемата на базата (имена, връзки, типове данни и ограничения), като даже в моделите е дадена по подробна дефиниция, заради методите в моделите.

2.3.3.1 Таблицата Brand

Първата таблица, която е създадена е на основните модели, които ще се срещат при продуктите. Това е таблицата Brand, чиято структура е следната:

Brand		
Име	Тип	Ограничения
Id	INT	PRIMARY KEY, NOT NULL
Name	NVARCHAR(32)	NOT NULL
PathD	VARCHAR(MAX)	NOT NULL

Таблица 1. Наименования и типове на полетата от таблицата *Brand*

Описание на колоните от Таблица 1:

- Id – уникален идентификатор на съответния продукт;
- Name – името на марката;
- PathD – това е svg пътът в координатната система, където ще се нарисува логото на съответната марка, понеже bootstrap не може да разполага с всички възможни емблеми, а някои са със запазени права и ограничен достъп.

2.3.3.2 Таблицата Category

Следващата таблица, която е създадена е на основните групи, в които ще се разпределят продуктите. Това е таблицата Category, чиято структура изглежда по този начин:

Category		
Име	Тип	Ограничения
Id	INT	PRIMARY KEY, NOT NULL

Name	NVARCHAR(32)	NOT NULL
------	--------------	----------

Таблица 2. Наименования и типове на полетата от таблицата *Category*

Описание на колоните от Таблица 2:

- Id – уникален идентификатор на съответния продукт;
- Name – името на категорията.

2.3.3.3 Таблицата Product

Главната таблица е Product, която съдържа свойствата и методите за валидация, които всеки новосъздаден продукт наследява от нея. Използва се да представи всички продукти на потребителите (групира ги). С нея се повишава нивото на абстракция, като само с един ред програмен код се достъпват останалите таблици. Тя изглежда по следния начин:

Product		
Име	Тип	Ограничения
Id	INT	PRIMARY KEY, NOT NULL
Name	NVARCHAR(128)	NOT NULL
Description	NVARCHAR(512)	NOT NULL
ProductUrl	NVARCHAR(64)	NOT NULL
OldPrice	DECIMAL(18,2)	NOT NULL
CurrentPrice	DECIMAL(18,2)	NOT NULL
TotalVotes	INT	NOT NULL
Score	INT	NOT NULL
CategoryId	INT	FOREIGN KEY, NOT NULL
BrandId	INT	FOREIGN KEY, NOT NULL

Таблица 3. Наименования и типове на полетата от таблицата *Product*

Описание на колоните от Таблица 3:

- Id – уникален идентификатор на съответния продукт;
- Name – името на артикула;
- Description – описание на продукта;

- OldPrice – старата цена на продукта (добавена за да се прави съпоставка с текущата и да се изтъква кои продукти са с намаление);
- CurrentPrice – настояща цена на продукта;
- TotalVotes – броя на оценките, които потребителите са дали за съответния продукт;
- Score – сбора от всички оценки на продукта;
- CategoryId – връзката към таблицата Category за прибавяне на продукта към прилежащата му категория;
- BrandId – връзка към таблицата Brand, за да запише марката на продукта.

Връзки (relationships) на таблицата *Product*:

- Един продукт може да бъде само от една марка и да е част от само една категория. Тук свързването на таблици е „едно към едно“.

2.3.4 Удостоверяване и упълномощаване в приложението

Едно от фундаменталните неща в информационната сигурност е CIA/ПИН (поверителност, интегритет и наличност) триадата. Този модел гласи, че информацията трябва да бъдат защитена от неоторизиран достъп. Автентичността на данните, тоест да не се променят той е случаен потребител, е от съществено значение и те да се достъпват бързо, но само, когато потребителят се нуждае от тях, като те не трябва да разкриват никаква информация, която е в разрез със законите за защита на личните данни.

Сайтът е изграден съгласно този модел, като е разделен на две роли с различни привилегии – „Customer“, който може да разглежда продуктите които се продават, не извършва покупки и да използва пазарската количка за запазване на желаните от него артикули за по късно и „Admin“, който може да манипулира данните в базата през графичният потребителски интерфейс на уеб приложението. С това първият основен компонент от триадата (поверителност) е изпълнен.

За втория (интегритет на данните) още при създаването на това приложение, то бива разделено на три зони (Areas): „Admin“, „Customer“ и една създадена автоматично – „Identity“, която съдържа необходимите за удостоверяването файлове. Тези зони се

използват, за да се ограничава достъпа до конкретни страници на уебсайта, като в зоната „Admin“ на достъп само тези потребители, чийто акаунт е маркиран като „Администраторски“.

Третият основен стълб е изградена още със създаването на моделите, като това разделение на много малки таблици с по два реда гарантира че стойностите на желаното свойство ще бъдат достъпни много по бързо, понеже се обхождат по-малко данни.

Най-общо казано, с удостоверяването и упълномощяването автентичността и целостта на данните и поверителността на потребителите е гарантирана от страна на сайта

2.3.5 Контролери (Controllers)

MVC контролерите са отговорни за отговора на заявка, която потребителя на даден уебсайт е направил, посредством интерфейса. Всяка заявка се свързва с определен контролер. Примерни заявки са за извършване на CRUD операциите върху дадена таблица от базата от данни (създаване, редактиране, изтриване и показване на подробни данни). Всеки публичен метод, който се добавя в контролера, може да се извика от всеки потребител на сайта с въвеждане на правилния URL в адресната лента на уеб браузера. Методът в контролера се използва като действие (Action). Действието връща *резултат от действие* (Action Result) в отговор на заявката от клиента (Client).

Заявките, които са използвани в проекта са GET и POST. GET методът се използва да заяви данни от конкретен източник и се използват в Index и Details методите, а POST методът се използва да се изпратят данни до сървъра чрез Create, Edit и Delete методите.

В този проект контролерът създава нов запис в релационната база от данни, ако няма такъв, когато потребителят иска да добави предмет в пазарската си количка. Също се използва администратор да може да манипулира таблиците, които са свързани с продуктите или потребителите, като да редактира данните за продуктите (име, описание, цена и др.) или да добавя данни в други таблици (напр. добавяне на нова марка продукт, нова категория и др.). Контролерите са автоматично генерирани от шаблон да изпълняват CRUD операции върху указаните таблици (при създаването се посочва модела, съответстващ на тази таблица, за да приложи и всичките атрибути за валидиране на стойността на свойството.

Контролерите са разпределени из между съответните зони (Admin и Customer) в презентационния слой (ArtFusStudio). В тази, предназначена само за администратори се намират тези, които манипулират таблиците, описващи продуктите или техните свойства, а в тази, предназначена за всеки потребител, се намират тези, които само презентират данните на клиентите и подобряват UX.

2.3.5.1 Контролер, който се използва за взема на данни от други таблици

DisplayLayoutController – е контролера, който се използва, за да се достъпват няколко различни таблици в един изглед, понеже може да се вика само до един *@model* на файл. Методите му се викат във всеки метод на контролер, съответстващ на изгледа, в който искаме да достъпим данните от конкретната таблица, точно преди да върне стойността. Методите му съдържат ViewData[“<Class>”] свойството, което се активира когато действието на контролера започне.

(Прилагам: Приложение 9: Част от методите на DisplayLaoutController)

2.3.5.2 Контролер, който се използва за пренареждане на данните от таблица

SortingController – е контролера, който се използва, за да сортира данните в някои от таблиците с повече данни по определени критерии, който е стойността на поле, наследено от базовия модел, още преди мигрирането към база данни. Той се вика в метода на контролера, съответстващ на продукт, след като се достъпят всички таблици, с които е свързана тази.

2.3.5.3 Контролери в area Admin

Те са необходими за манипулиране на базата през потребителския интерфейс. Методите използвани в контролерите са типичните (Index, Create, Edit, Delete, DeleteConfirm, Details и др.), които се генерират автоматично, с помощта на новите, добавени в EF Core 8 функционалности (Десен бутон на папката Controllers в съответната поля → Add → Controller... → MVC Controller with views, using Entity Framework → избира се желания модел, който искаме да използваме за осъществяване на ORM и създаването на идентична на него таблица в БД → Избиране на DbContext клас → Add и започва процеса по Scaffolding (Това е работна рамка на ASP.NET MVC за автоматично генериране на код, който взаимодейства с базата данни, т.е. основните CRUD операции)). Контролерите които имат общи реакции са тези, свързани със следните таблици: Brand, Category, Camera,

Memory, OperatingSystem, OperatingSystemVersion, StorageCapacity, USB, Phone, CaseMaterial, ProtectionLevel, Case, Charger и Coupon и това е добавянето на достъп до таблиците Brand, Category и Product от всеки метод на тези контролери, защото това подобрява потребителското преживяване. Допълнително само за контролерите, които имат връзка с някой от продуктите, при CategoryController, CamerasController, MemoriesController, OperatingSystemsController, OperatingSystemVersionsController, StorageCapacitiesController, USBsController, CaseMaterialsController и ProtectionLevelController е изтрит Index() метода, защото извличането на данни от няколко таблици е обединено в един изглед. Преди връщането на стойността във всеки от методите на споменатите контролери е повикан помощния контролер по следния начин: DisplayLayoutController.AcceessAllTables(this, _context); При другите контролери се срещат същите добавки, но като допълнение са повикани и други необходими методи от този контролер:

- При OSNameAndVersionsController е повикани AcceessAllTablesOS метода в Index метода, като така се показват записи на всички съществуващи операционни системи, техните версии и всички възможни комбинации от предходните две;
- При DisplayTechnologiesController е повикани AccessAllHardwareTables метода в Index метода, като така се показват записи на всички съществуващи хардуерни компоненти (брой камери, технология на екрана, RAM памет, памет на хранилището и тип на USB порта);
- При CaseTypesController е повикани AccessAllAestheticTables метода в Index метода, като така се показват записи на всички съществуващи естетически елементи за един калъф (материалът от който е разработен, типът такава и нивото на защита което предлага на купувача).

Допълнително е създаден и контролер, който да се използва за менажирането на потребителите, т.е. изтриване то им (ApplicationUsersController). Понеже потребителите имат възможността да си добавя профила снимка е създаден контролер и на модела, който запазва най-често срещаните разширения за графични изображения (ImageExtensionsController).

2.3.5.4 Контролери в area Customer

Тази област броят на контролерите е значително по малък от тези в администраторския панел - тук се срещат само 4:

- HomeController – има само един метод Index() за заглавната страница на уеб сайтът;
- ProductsController – предназначен е за показване на всички артикули които са продават в сайта, както и за показване на подробни данни за всеки от тях. Той съдържа опции за филтриране по марка от страна на потребителя, както и да пренареди всички продукти по определен критерий, в случая по име, цена, рейтинг и общ брой гласове.
- ShoppingCartController – този контролер съдържа бизнес логиката зад пазарската количка. Когато потребител иска да добави предмет в своята количка, с GET заявка до таблицата се проверява дали този потребител има вече създадена пазарска количка или трябва да се създаде нова. След това когато той иска да добави артикул в количката си, със същата заявка се проверява дали този артикул вече присъства там и ако се намира в количката то тогава неговата бройка се увеличава с 1, вместо да му излиза името по два пъти. След като потребителят е избрал желаните от него предмети има две опции - да си изтрие количката или да закупи артикулите. Отделно той предлага и възможността потребителят да изтрие само един артикул от количката си.
- Последният контролер е този, който позволява на потребителите да добавят, променят и изтриват своята профилна снимка, както и поставя рамка тя да бъде във конкретни разширения (png, jpg, jpeg, tif, tiff, bmp). Това е UserProfilePicsController. Той оперира върху таблицата, която съдържа запишете на всички потребителски снимки и ги свързва със акаунтите. На потребителя предоставя информация само за неговата снимка. Когато някой иска да си добави правилна снимка се създава нов запис в таблицата, а името на снимката става уникалния идентификатор на потребителя, понеже така няма да има файлове с еднакви имена. Той има проверка ако потребителят вече има своя профилна снимка и иска да я замени с друга то не създава нов запис а просто модифицира съществуващия както и стаи в профилната снимка и аз заменя с нова, чието име е същото, защото идентификаторът на

потребителя не се променя. Достъпването на изображения става с помощта на потоци.

2.3.6 Изгледи (Views)

Изгледите в ASP.NET Core представляват файловете със разширение „*cshtml*“, които са част от презентационния слой. Те се връщат от действията на контролера. Главната им цел е да представят данни от някаква структура от данни (колекция) на потребителя. Биват няколко вида, в контекста на приложението са използвани Razor Pages, които позволяват смесването на HTML маркиране със C# код.

2.3.6.1 Шаблонен изглед

Това е този изглед, чието съдържание се показва за всяка уеб страница. В този проект той се нарича „*_Layout.cshtml*“ и се намира в папката за споделени изгледи (Views/Shared). В него се намират референции към всички използвани CSS файлове, както и JavaScript кодове, защото като се намират в споделен изглед се намалява повтарянето на код. HTML съдържанието на файла е навигационната лента, както и лентата с марките на продуктите, които се предлагат в сайта, както и фонът, който е общ с всички страници. В най-горния блок се намират логото на сайта, бутоните за връзка, запазените артикули на потребителя, промоциите, връзките към страниците на администраторския панел, възможността за излизане или влизане в акаунт, интегрираният преводач на Google с който потребителите избират езика, на който да виждат съдържанието и пазарската количка. Блокът под нея съдържа препратка към началната страница, препратка към списъка със всички предлагани стоки и опциите за филтриране по марка.

2.3.6.2 Изгледи в area Admin

Изгледите, също като контролерите са създадени с помощта на EF Core Scaffolding и биват няколко файла за всеки контролер. Основният план за архитектура на изгледа на конкретен модел е в папката, с име съответстващо на контролера да имат 5 файла с разширение *cshtml*: Create, Delete, Details, Edit и Index, като при някои индекси е изтрит, понеже представянето на данни от таблици с общи черти се случва в един изглед (напр. данните от *OperatingSystem*, *OperatingSystemVersion* и *OSNameAndVersion* се показват в една страница на сайта; по същия начин са групирани хардуерните характеристики, както и

естетическите и търговските марки с категориите, към които начисляваме конкретен продукт).

(Прилагам: *Приложение 10: Показване на данни от няколко таблици наведнъж*)

Структурата на всяка една Razor страница със сходно име е идентична:

- За „Index.cshtml“ е характерно данните да се презентират под формата на таблица. Началото на сайта добавяме необходимите библиотеки, както и установяване на връзката с таблицата под формата на @model. @model директива позволява достъп до списъка с обекти или до обекта, ако той не е тип в IEnumerable<T> колекция, който контролера предава на изгледа. Model е силно типизиран. За да може уебсайтът да изглежда еднакво добре на колкото се може повече устройства таблицата е вкарана в серия от div тагове с имена на класовете, характерни за Bootstrap за по-лесната им обработка. Преди таблицата обаче се добавят връзки с ASP действие (в контекста на този файл това е бутонът за добавяне на запис към таблицата). Преди таблиците много данни, след нея следва форма, в която е включена търсачка, където администраторът може да филтрира всички записи по определен критерий, от която и да е колона. (Прилагам: *Приложение 11: Използване на търсачката за Phone*)

Търсачката е имплементирана чрез JavaScript, но за да работи трябва на таблицата да се зададе Id="myTable".

(Прилагам: *Приложение 12: Имплементирането на търсачките*).

След това идва същинската част представяне на данните. Първо се създава thead, където се записват заглавията на колоните, след това идва ред на tbody, което е същинската част тук. Там с един foreach цикъл се обхожда цялата таблица и се извежда всеки запис, разпределен по колони. Най-вдясно след отпечатването на всеки ред данни се намират бутоните за действия - за редактиране на записа, за изтриването му и евентуално ако има повече полета и за показване на детайлите му с подробности. За стилизацията на тези бутони се използват иконите от Bootstrap. Те ни насочват към някое действие, методът на който присъства в контрола на съответният за това действие обект или обекти.

- В групите от изгледите, в чиито контролери са изтрети методите, са изтрети и едноименните изгледи. Следващия изглед, който присъства във всяка папка във Views е „Create.cshtml“. Той е предназначен за създаване на запис в указаната таблица от базата чрез HTTP заявка от клиента до базата данни. Отново, за да може интерфейса да се адаптира към различни резолюции цялото съдържание е поставено в серия от div тагове и след това във форма. Там се намират полетата за вход на стойност на новия запис, както и под тях е валидацията, която сме записали за свойството в модела. Накрая се намира връзка към предишната страница.
- По време на създаването на обекта може бъде допусната някаква грешка, затова е важно да се добави меню за редактиране на стойността на някое от свойствата. Това е функцията на „Edit.cshtml“. Там са приложени същите похвати, както в предишния изглед.
- След промяна на съдържанието трябва да има възможността за подробен преглед само на този обект, ако той има много свойства и не всички могат да се представят на една страница. Затова е добавен изгледа „Details.cshtml“. Той показва стойността на всяко от свойствата на конкретния продукт. Накрая е добавена връзка към панела за редактиране, ако искаме да променим някоя от стойностите.
- Накрая, когато обекта няма да присъства повече в сайта, трябва да бъде изтрит. Но бутонът за изтриване може да бъде натиснат по грешка, затова е добавен нов изглед за потвърждаване, дали администраторът желае да изтрие съответния запис от базата. Това е изгледа „Delete.cshtml“, който съдържа данните на обекта, които потребителя иска да изтрие, както и бутон за потвърждаване и връзка към предишната отворена страница.

2.3.6.3 Изгледи в area Customer

- Там изгледите са от 4 папки. В първата се намира изгледа към началната страница, която е първото нещо, което потребителят вижда от този уебсайт. Там са показани трите телефона с най-висока оценка и на потребителят се предлага опцията да ги добави в количка или да ги разгледа по-подробно.
- Следващата папка (Products) са показани всички предлагани в сайта артикули са общите им свойства, които наследяват от базовия клас. Там на клиентът се

предлага възможността да ги добави в количка, да ги закупи веднага или да види повече информация за тях.

- След като ги добави в количката си, идва ред да може да прегледа какви предмети има там. Това е и същността на **CartView** изгледа от там потребителят да може да оперира със своята количка.
- Предназначението на последните изгледи е предоставяне на потребителя възможността да може да прикачи своя профилна снимка, както и да я редактира. Той избира снимката от локалната си файлова система и тя се запазва в папка в главната директория със име неговия уникален идентификатор.

2.3.6.4 Изгледи в area Identity

Там се намират изгледите, необходими за удостоверяването на потребителите, т.е. Създаване на акаунт, влизане в съществуващ акаунт или излизането от профила си. За регистриране е необходимо потребителят да предостави своята електронна поща, да си избере потребителско име, парола, отговаряща на съображенията за сигурност и пожелание да прикачи профилната си снимка. Предлага се възможността след като потребителят вече си влезе в профила да избере опцията, когато следващия път отвори този сайт, той ще остане влязъл в акаунта си.

2.3.7 JavaScript (JS)

JavaScript е изключително полезен език за програмиране, понеже той може да достъпва HTML тагове, чрез техния клас или уникален идентификатор и да променя стила им. Основното му приложение в уеб разработката е забързването на уебсайта и подобряването на потребителското преживяване, като намалява действията които потребителят трябва да прави, за да намери всички функционалности, които уеб приложението му предлага. Използване още в началната страница за ефектите на въртящите се артикули, както и за интегрирането на преводача.

Другото му приложение е в началната страница са онези малки детайли, които отличават този сайт от останалите. Това са звездите, които показват рейтинга на съответния продукт, както и ефектът, който се получава, когато потребителят натисне на сърцето, за да запази продукта за по-късно. Понеже за тези звезди се изискват около 200 код на CSS, но с

математическа зависимост и ненужно повтаряне, е вмъкнат JS, който с един цикъл генерира необходимия код за стилизация на звездите.

Този език за програмиране е използван в уеб сайта, защото някои процеси се извършват за по-малко време, когато се компилира JS код, вместо C#. Най-явният пример за това е търсачката в администраторските станции, която работи със събития и обхожда всяка една клетка от таблицата и извежда само тези редове, в които има съвпадение между записания от потребителя низ от символи и стой настана някоя от клетките, в съответния ред.

Отделно в някои изгледи за отпечатване на всички данни от таблици, редовете станат прекалено много и освен търсачката са добавени и страници, които могат да се сменят, както с бутони, така и чрез стрелките на клавиатурата.

При използването на JavaScript е изпълнено условието, че всичко в което е интегриран, трябва да зарежда за не повече от 2 секунди.

3. Заключение

Този проект представлява динамично в приложение в което потребителите могат да разгледат богатият набор от артикули разделени по категории и да използват търсачката за филтриране и да намерят желаните от тях артикули с конкретен критерии. Благодарение на

подробната база от данни, клиентите могат да търсят артикула по някое от неговите свойства, като цена или специфични до категорията, като технология на екрана. С интегрираният преводач на Alphabet (Google) се цели аудиторията на сайта да се увеличи значително и хора които не разбират български пак да могат да го използват безпроблемно. Това значително увеличава потребителското преживяване, в съчетание с потребителския интерфейс.

Откъм удостоверяване проектът е разделен на две роли – „Customer“ и „Administrator“, Като администраторът има правото да манипулира всяка таблица от базата с данни - да добавя, редактира или изтрива продукт или някое от неговите свойства, а потребителят и в правото да си промени профилната снимка, да добавя артикули в пазарската количка и да извършва поръчки.

Бъдещи планове за развитието на приложението:

- Добавяне на повече функционалности в Identity зоната;
- Добавяне на бизнес логиката за оценяване на артикулите
- Добавяне на бизнес логиката колекция със запазени продукти
- Адаптиране на уеб сайта към все повече устройства.

4. Приложения

ПРИЛОЖЕНИЕ 1: Имплементирането на метода FormatDecimal в ArtFusStudio.Utility

```
public static class DataFormatting
{
    3 references
    public static string FormatDecimal(this object value)
    {
        string stringValue = value.ToString();

        stringValue = stringValue.Replace(",", ".");

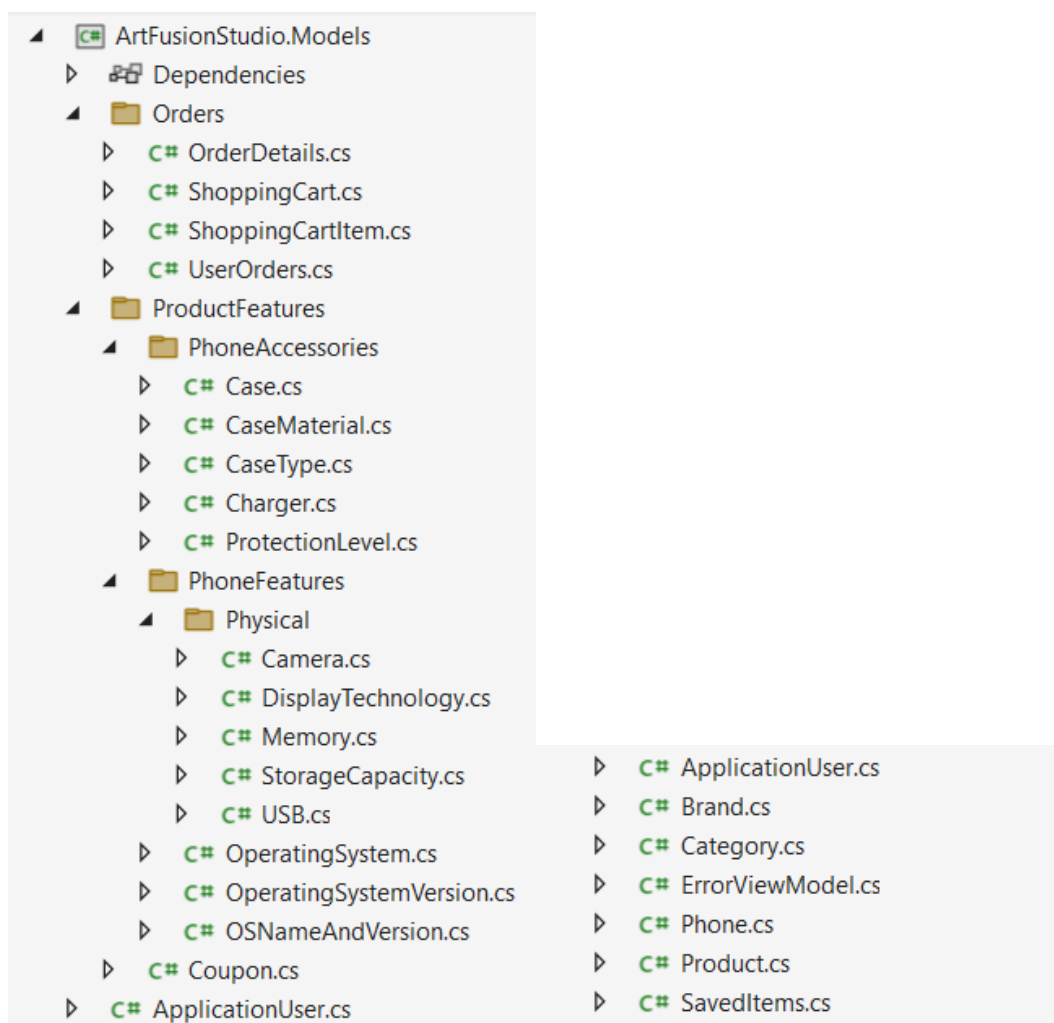
        int commaIndex = stringValue.IndexOf('.');
        int zeroIndex = stringValue.IndexOf('0');
        int lastZeroIndex = stringValue.LastIndexOf("0");

        if (commaIndex != -1 && zeroIndex != -1 && commaIndex < lastZeroIndex)
            stringValue = stringValue.Substring(0, commaIndex + 1) + stringValue.Substring(commaIndex + 1).Replace("0", "");

        if (stringValue.EndsWith('.'))
            stringValue = stringValue.Substring(0, stringValue.Length - 1);

        return stringValue;
    }
}
```

ПРИЛОЖЕНИЕ 2: Архитектура на ArtFusStudio.Models



ПРИЛОЖЕНИЕ 3: Архитектурата на модела „Brands“

43 references

```
public class Brand
```

```
{
```

```
    [Key]
```

21 references

```
    public int Id { get; set; }
```

```
    [Required(ErrorMessage = ErrorMessage.NO_BLANK_SPACE)]
```

```
    [StringLength(32, MinimumLength = 2, ErrorMessage = ErrorMessage.OUT_OF_RANGE + " 2 u 32.")]
```

35 references

```
    public required string Name { get; set; }
```

```
    [Required(ErrorMessage = ErrorMessage.NO_BLANK_SPACE)]
```

21 references

```
    public required string PathD { get; set; }
```

```
}
```


ПРИЛОЖЕНИЕ 4: Архитектурата на модела „Category“

38 references

```
public class Category
```

```
{
```

```
    [Key]
```

18 references

```
    public int Id { get; set; }
```

```
    [Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
```

```
    [StringLength(32, MinimumLength = 2, ErrorMessage = ErrorMessages.OUT_OF_RANGE + " 2 u 32.")]
```

23 references

```
    public required string Name { get; set; }
```

```
}
```

ПРИЛОЖЕНИЕ 5: Архитектурата на модела „Product“

erences

```
ic class Product
```

```
[Key]
```

96 references

```
public int Id { get; set; }
```

```
[Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
```

```
[StringLength(128, MinimumLength = 2, ErrorMessage = ErrorMessages.OUT_OF_RANGE + " 2 u 128.")]
```

98 references

```
public string Name { get; set; }
```

```
[AllowNull]
```

```
[StringLength(512, MinimumLength = 2, ErrorMessage = ErrorMessages.OUT_OF_RANGE + " 2 u 512.")]
```

87 references

```
public string Description { get; set; }
```

```
[Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
```

```
[StringLength(64, MinimumLength = 2, ErrorMessage = ErrorMessages.OUT_OF_RANGE + " 2 u 64.")]
```

93 references

```
public string ProductURL { get; set; }
```

```
[AllowNull]
```

```
[Range(0.0, 100000.0, ErrorMessage = ErrorMessages.NOT_NEGATIVE_NUM + " 100000.")]
```

96 references

```
public decimal OldPrice { get; set; }
```

```
[Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
```

```
[Range(0.0, 100000.0, ErrorMessage = ErrorMessages.NOT_NEGATIVE_NUM + " 100000.")]
```

97 references

```
public required decimal CurrentPrice { get; set; }
```

80 references

```
public int CategoryId { get; set; }
```

80 references

```
public int BrandId { get; set; }
```

```
[ForeignKey("CategoryId")]
```

25 references

```
public Category? Category { get; set; }
```

```
[ForeignKey("BrandId")]
```

31 references

```
public Brand? Brand { get; set; }
```

```
[AllowNull]
```

97 references

```
public int TotalVotes { get; set; } = 0;
```

```
[AllowNull]
```

```
[ScoreValidation(ErrorMessage = ErrorMessages.INVALID_SCORE)]
```

78 references

```
public int Score { get; set; } = 0;
```

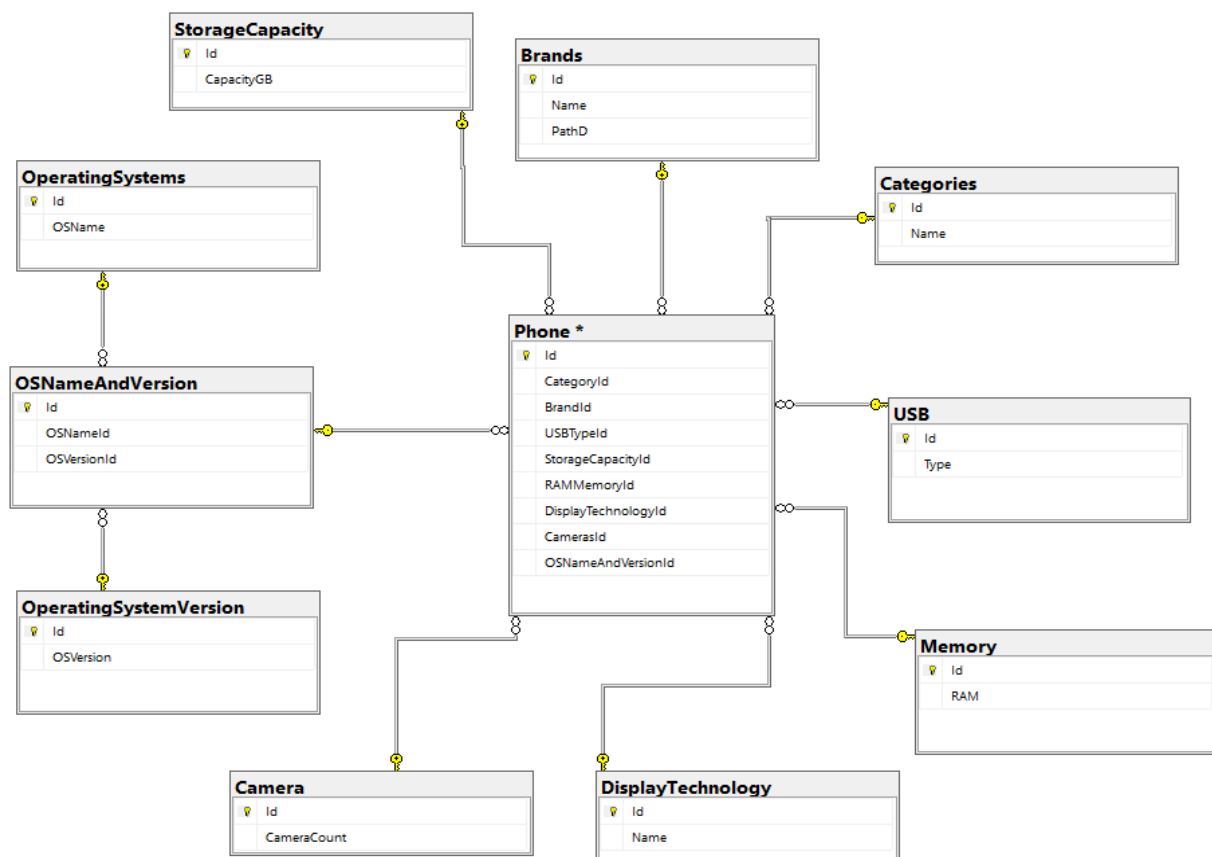
```
}
```

ПРИЛОЖЕНИЕ 6: Имплементация на атрибута за валидиране на данни


```
1 reference
public class EndDateValidationAttribute : ValidationAttribute
{
    0 references
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        var endDate = (DateTime)value;
        var coupon = (Coupon)validationContext.ObjectInstance;

        if (endDate.Date <= coupon.StartDate.Date)
        {
            return new ValidationResult(ErrorMessage);
        }
        return ValidationResult.Success;
    }
}
```

ПРИЛОЖЕНИЕ 7: Връзките между таблицата Product чрез външни ключове



ПРИЛОЖЕНИЕ 8: Подробно описание на таблицата Phone

Phone *				
	Column Name	Condensed Type	Length	Allow Nulls
	Id	int	4	<input type="checkbox"/>
	Name	nvarchar(128)	128	<input type="checkbox"/>
	Description	nvarchar(512)	512	<input type="checkbox"/>
	ProductURL	nvarchar(64)	64	<input checked="" type="checkbox"/>
	OldPrice	decimal(18, 2)	9	<input type="checkbox"/>
	CurrentPrice	decimal(18, 2)	9	<input type="checkbox"/>
	CategoryId	int	4	<input type="checkbox"/>
	BrandId	int	4	<input type="checkbox"/>
	TotalVotes	int	4	<input type="checkbox"/>
	Score	int	4	<input type="checkbox"/>
	USBTypeId	int	4	<input checked="" type="checkbox"/>
	StorageCapacityId	int	4	<input checked="" type="checkbox"/>
	RAMMemoryId	int	4	<input checked="" type="checkbox"/>
	DisplayTechnologyId	int	4	<input checked="" type="checkbox"/>
	CameraId	int	4	<input checked="" type="checkbox"/>

ПРИЛОЖЕНИЕ 9: Част от методите на DisplayLayoutController

99+ references

```
public class DisplayLayoutController : Controller
{
```

```
    private readonly ApplicationDbContext _context;
```

0 references

```
    public DisplayLayoutController(ApplicationDbContext context)
    {
        _context = context;
    }
```

99+ references

```
    public static void AccessAllTables(Controller controller, ApplicationDbContext context)
    {
        controller.ViewData["Category"] = context.Categories.ToList();
        controller.ViewData["Brand"] = context.Brands.ToList();
        controller.ViewData["Product"] = context.Products.ToList();
    }
```





























8 references

```
    public static void AccessAllTablesOS(Controller controller, ApplicationDbContext context)
    {
        controller.ViewData["OSName"] = context.OperatingSystems.ToList();
        controller.ViewData["OSVersions"] = context.OperatingSystemVersion.ToList();
        controller.ViewData["OSNameAndVersion"] = context.OSNameAndVersion.ToList();
    }
```

2 references

```
    public static void AccessAllAestheticTables(Controller controller, ApplicationDbContext context)
    {
        controller.ViewData["CaseMaterials"] = context.CaseMaterial.ToList();
        controller.ViewData["CaseTypes"] = context.CaseType.ToList();
        controller.ViewData["ProtectionLevels"] = context.ProtectionLevel.ToList();
    }
```

ПРИЛОЖЕНИЕ 10: Показване на данни от няколко таблици наведнъж

Материали			Модели			Нива на защита		
Създай нов			Създай нов			Създай ново		
№	Вид		№	Тип		№	Защита	
1	Силикон	 	1	Портфейл	 	1	Екстремно	 
2	Дърво	 	2	Само гръб	 	2	Здраво	 
3	Плат	 	3	Скелет	 	3	Средно	 
4	Силикон	 	4	Вертикален	 	4	Слабо	 
5	Пластмаса	 	5	С батерия	 			

ПРИЛОЖЕНИЕ 11: Използване на търсачката за Phone

Показаният пример се вижда, как дори при липса на съвместимост между главни и малки букви, търсачката връща правилните желани резултат, Без значение от коя колона се намира низа, когото сме използвали като филтър.



The screenshot shows a search results page for mobile phones. The title 'Телефони' is in red. Below it is a search bar containing the text 'oled' and a link 'Върни се към пълния списък'. The results are displayed in a table with three rows. A red rectangular box highlights the 'OLED' column across all three rows.

Телефони										
Добави нов телефон										
oled			Върни се към пълния списък							
1			Iphone 15 Pro	2999,99лв.	2699,99лв.	USB-C	512GB	8GB	OLED	5
2			Huawei P40 Pro	899,99лв.	799,99лв.	USB-C	48GB	8GB	OLED	5
5			Nokia 9 PureView	699,99лв.	599,99лв.	USB-C	64GB	6GB	OLED	1

ПРИЛОЖЕНИЕ 12: Имплементиране на търсачката

2 references

```
function searchEngine() {
    var input, filter, table, tr, td, i, j, txtValue;
    input = document.getElementById("myInput");
    filter = input.value.toUpperCase();
    table = document.getElementById("myTable");
    tr = table.getElementsByTagName("tr");
    for (i = 0; i < tr.length; i++) {
        var found = false;
        for (j = 0; j < tr[i].cells.length; j++) {
            td = tr[i].getElementsByTagName("td")[j];
            if (td) {
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1) {
                    found = true;
                    break;
                }
            }
        }
        if (found) {
            tr[i].style.display = "";
        } else {
            tr[i].style.display = "none";
        }
    }
}
```

5. Използвана литература

Microsoft, <https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/>

YouTube, <https://www.youtube.com/watch?v=AopeJjkcRvU&t=19015s>

YouTube, <https://www.youtube.com/@ASPNETMVCCORE>

GitHub, <https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-mvc/intro.md>

Perplexity (Търсачка), <https://www.perplexity.ai/>

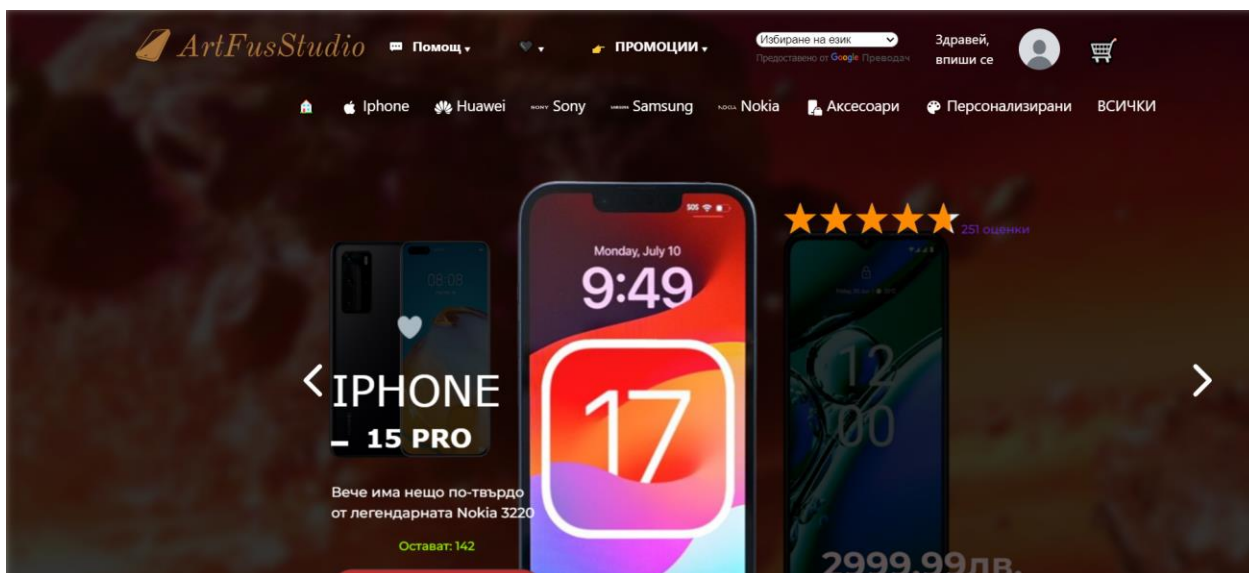
W3schools, <https://www.w3schools.com/howto/>

Startbootstrap, <https://startbootstrap.com/>

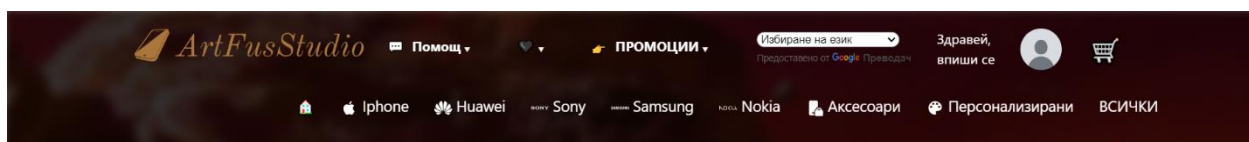
SVG икони, <https://iconify.design/>

Наръчник за потребителя

Уебсайтът е интерактивен и има доста голямо взаимодействие между потребителя и интерфейса. Където е възможно, опцията потребителят да натиска върху съответен бутон е премахната и заменена с падащи списъци когато си сложи курсора върху него. Отделно за да се съкрати времето, което потребителят ще изгуби в на мишката за покриване на бутони или натискането на такива, са добавени и събития. Първото нещо, което всеки потребител на сайта ще види, е началната страница, затова тя трябва да грабне вниманието му. Докато той не влезе в акаунта си, една немалка част от функционалностите остават скрити за него и когато се опитва да ги използва, те го препращат към страницата за удостоверяване. Първият кадър, който потребителите виждат, след като зареди всеки код е:

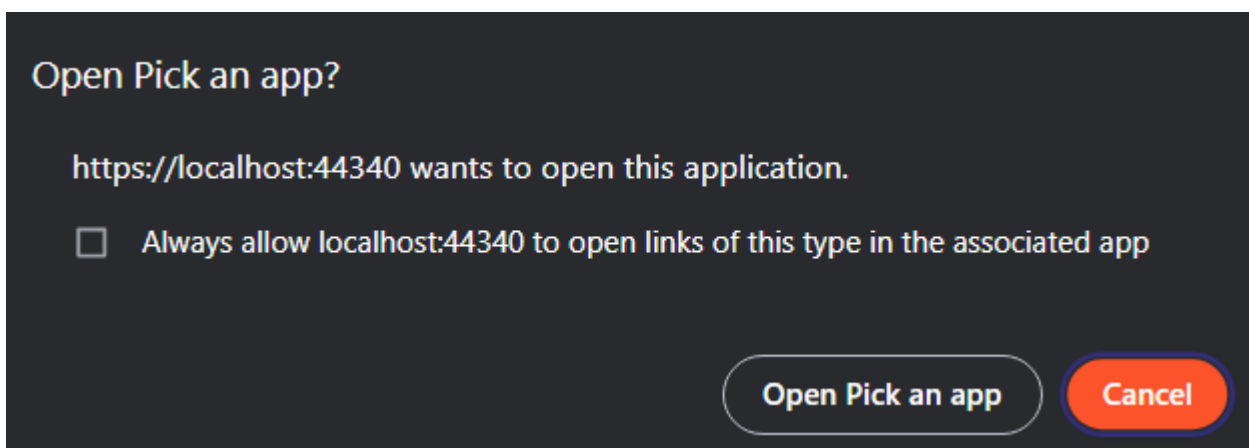


Общият за всяка страница изглед е:

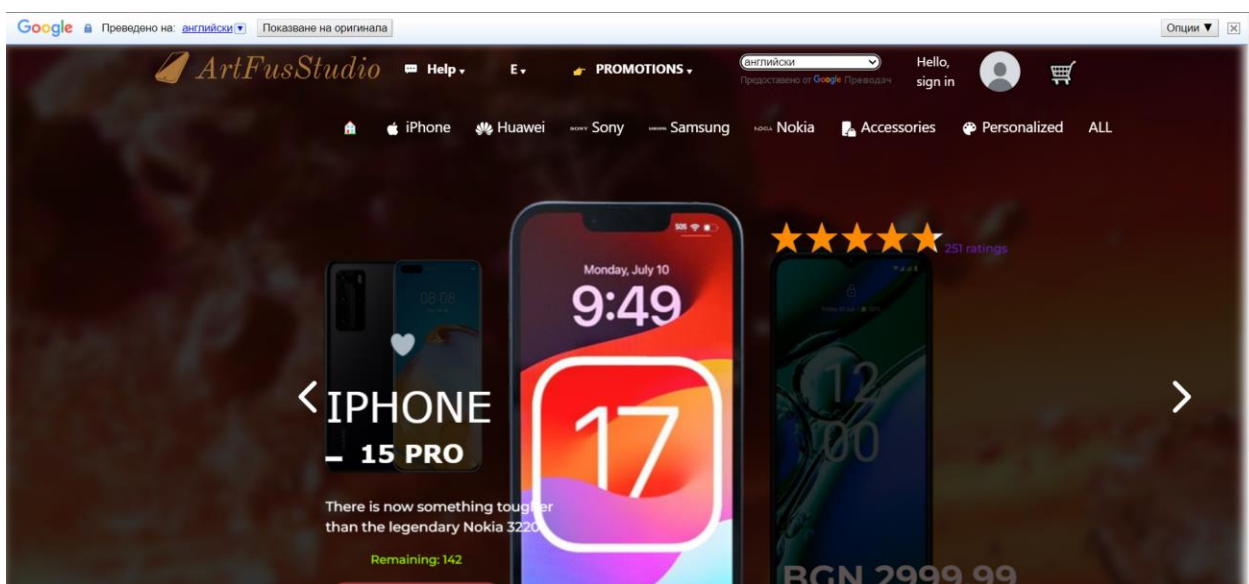


С натискането върху логото, което се намира в горния ляв ъгъл, потребителят се връща обратно в началната страница. Когато стрелката на мишката се сложи върху „Помощ“ излиза падащ списък с връзки към търговското лице, който стои зад сайта (предоставя

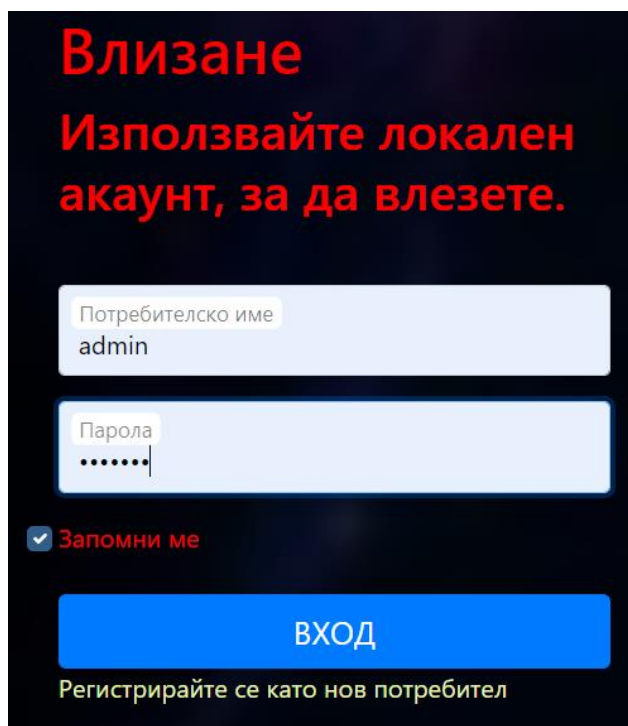
телефонен номер и като се натисне върху него се отваря следното из качество меню:



другата информация, която падащият списък предоставя, е адресът на електронната поща на лицето, както и възможността за чат). По-долните връзки не отвеждат към никъде, защото не е предоставен истински имейл. Следващият падащ списък е този, с икона на сърце, който трябва да покаже на потребителя запазените за по-късно от него стоки, но само, когато е влязъл в своя потребителски акаунт. Следващият падащ списък е този който показва промоциите. За да се каже че един артикул е промоционален, то трябва старата му цена да бъде по-висока от текущата му Той показва на потребителите само няколко от промоционалните артикули, за да се избегне прекалено голям списък. Следващото нещо е ключово за всеки бизнес, който иска да има клиенти от целия свят - това е възможността клиентът да преведе съдържанието на уебсайта на своя роден език. Така би изглеждал сайта, ако английският бъде избран като такъв:



Следващото нещо в редицата е връзката към формата за удостоверяване на потребителя и до нея се намира правилната снимка, която той би си избрал при създаването на акаунта си. Тя изглежда по този начин:



Влизане

Използвайте локален акаунт, за да влезете.

Потребителско име
admin

Парола
.....

☒ Запомни ме

ВХОД

Регистрирайте се като нов потребител

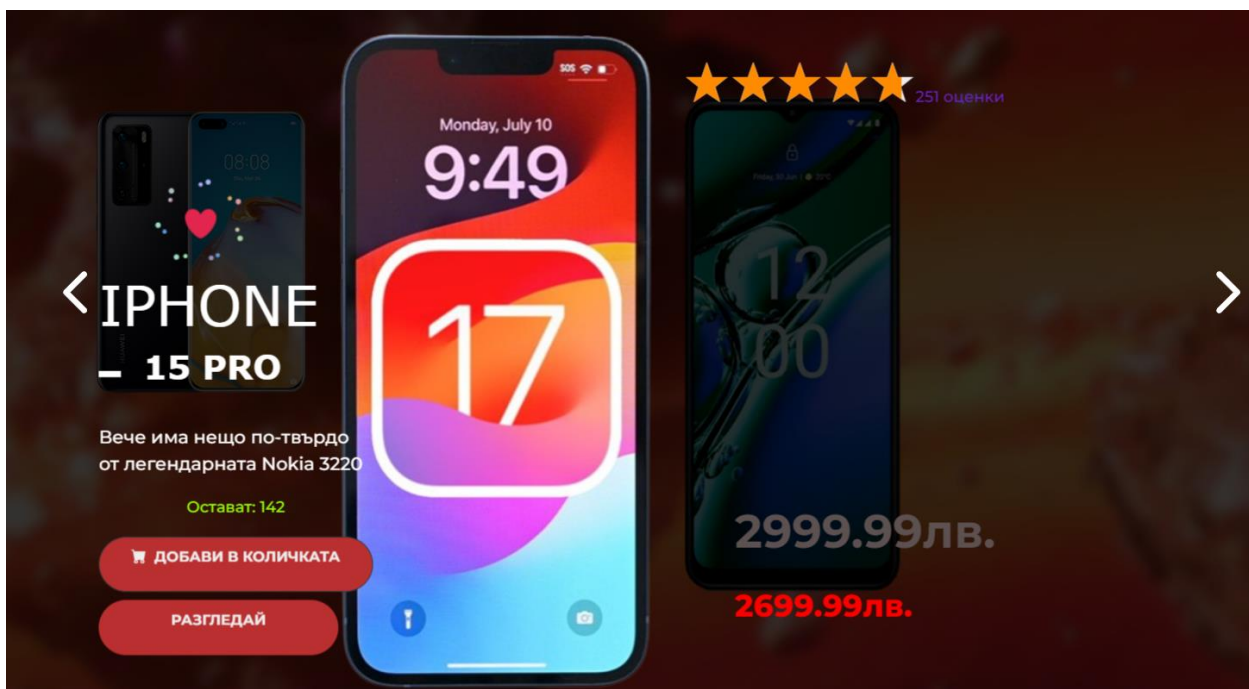
Ако потребителят няма акаунт може да се регистрира като нов потребител, като натисне върху връзката с текст „Регистрирайте се като нов потребител“.

Когато натисне върху тази снимка, ако потребителят не е влязъл в акаунта си, той бива отвеждане към формата за влизане, но ако е влязъл в профила си, той бива отвеждан към страница, която му предлага да излезе от акаунта си. След като се е удостоверил той може да промени своята профилна снимка, като прикачи от локалното си хранилище. За целта трябва да натисне върху снимката преди количката, след това на бутона за създаване и после на този за прикачване на изображение. То трябва да бъде в най-използваните разширения (PNG, JPG, JPEG), но може и други. След като е качил снимката си може да я промени като натисне върху нея и после върху бутона „Редалторай“.



След като прикачи новата си снимка натиска върху бутона за запазване на промените и тя се добавя и показва до името му. Снимките се запазват в папката в главната директория, като името им е потребителския идентификатор.

Това което е същността на началната страница са трите телефона с най-висока оценка. Информацията която е представена за тях е техните имена, версии, оценка, текуща цена, стара цена, ако артикулът е промоционален, описанието, наличност, ако остават под 150 артикула, бутоните за запазване в количка и допълнителни детайли, както и сърцето за запазване в „Любими“. Сменянето на телефоните се осъществява, както през потребителския интерфейс на сайта, чрез натискане на стрелките, така и чрез събития - когато потребителят натисне лявата или дясната стрелка на клавиатурата телефонът в средата се измества в съответната посока и този от отсрещната застава на негово място. Заедно с телефона се сменя и фона на страницата. Тя изглежда по този начин с ефекта, когато сърцето бива натиснато:



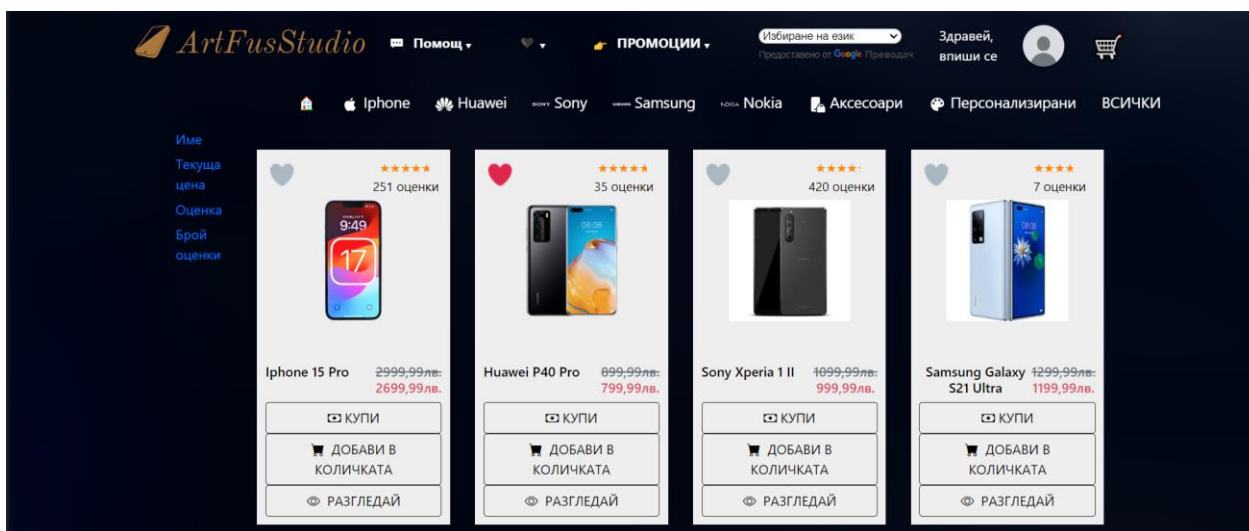
Бутонът „РАЗГЛЕДАЙ“ ни отвежда към страница с подробна информация за този телефон, в която се предлага опцията за добавяне в количка или за директно закупуване. Страницата също включва и връзка за връщане назад към предишната отворена страница, използвайки историята на браузера. Изглежда по следния начин:



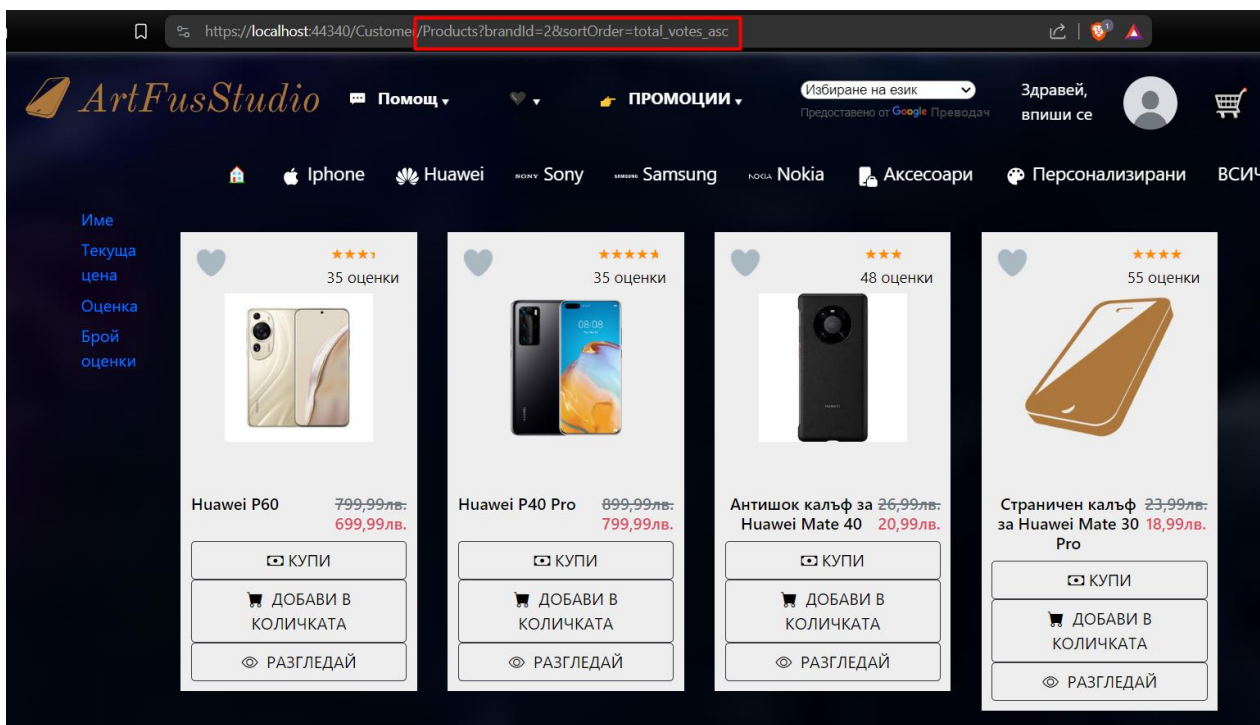
Връзките и информацията, показани тук, са подложени на същите проверки, както в предишната страница. Допълнителните функции на заглавката, вече не бяха споменати в началото са тези, свързани с връзките от втората лента. Първото нещо, което прави впечатление е, че тази лента се закача за най-горната част на екрана и стои там, колкото и надолу в страницата да е отишъл потребителя. Основните и компоненти са:



връзка към началната страница си икона на къщичка, опциите за филтриране на съдържанието и показване на определена марка и показване на всички предлагани артикули. Тя изглежда по следния начин:



Освен възможностите за филтрация на съдържанието, които също се използват и за показване на потребителите кои са марките на артикулите които са предлагат, тя притежава и възможностите за поддържането на артикулите по определен критерий - по азбучен ред, по цена, по рейтинг и по брой оценки. Като се натисне първия път върху бутоните за сортиране, предметите са подреджват по възходящ ред, а като се натисне отново върху същия бутон - по низходящ. Сайтът предлага възможността за комбиниране между филтриране и сортиране.











От прикачената снимка се вижда че са показани само артикулите с марка „Huawei“, подредени по брой оценки във възходящ ред. Това се случва под формата на GET заявка до таблицата Product за извличане на всички артикули, отговарящи на съответните изисквания от страна на потребителя. Многото нещо което се показва в тази снимка е как изглеждат продуктите които нямат снимка. Показва се логото на сайта, като алтернативна такава. След като потребителят е въвел своето потребителско име и парола, той вече е влязъл в акаунта си и може да извършва повече действия. Ето как би изглеждала пазарската му количка след като са добавени няколко артикула. В нея се показват не повече от 4 артикула:

в нея и предоставя възможността за премахването на конкретен артикул:

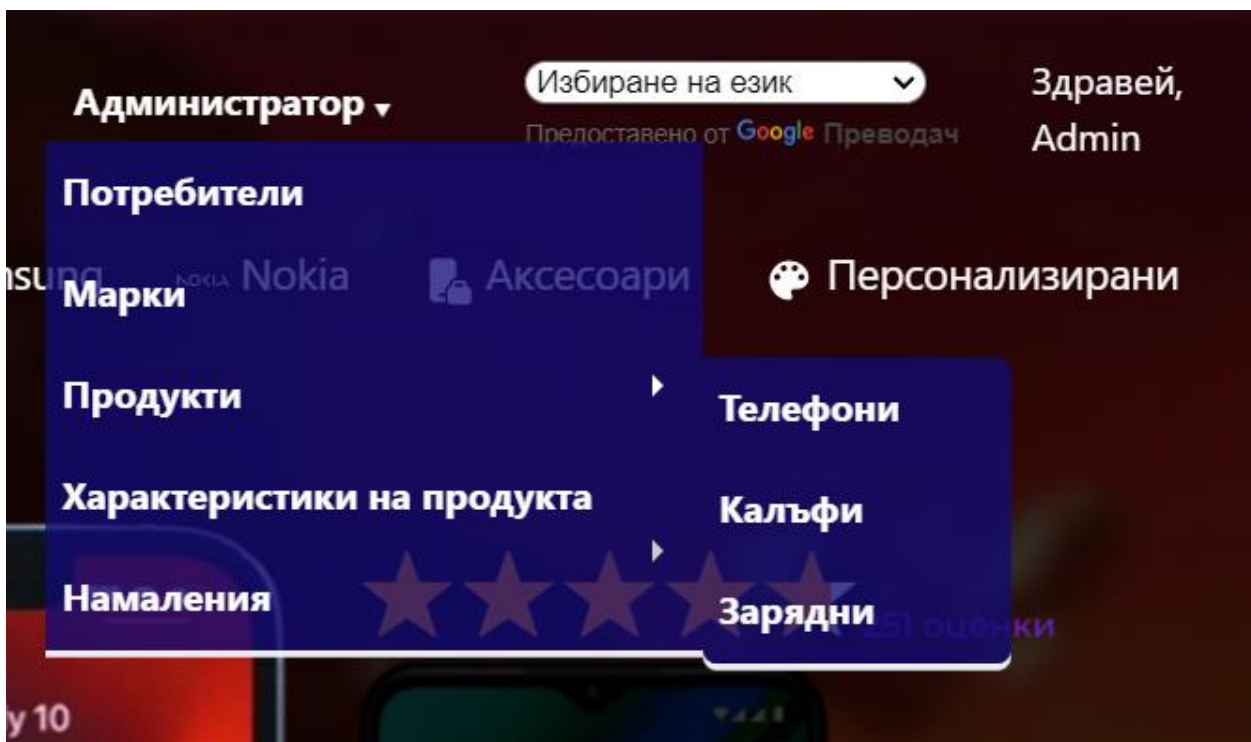
Пазарска количка						
№	Марка	Изображение	Продукт	Брой	Цена	Общо
1	Huawei		Huawei P40 Pro	3	799,99 лв.	2399,97 лв.
3	Nokia		USB с зарядно за Nokia	1	11,99 лв.	11,99 лв.
4	Apple		iPhone 14 Pro	1	1199,99 лв.	1199,99 лв.
5	Nokia		Nokia 9 PureView	1	599,99 лв.	599,99 лв.

Като допълнение предлага информация за това колко струват всички артикули общо, колко е броят им, бутон за пълно изпразване на количката, покупка на тези предмети, както и връщане назад и продължаване на пазаруването.

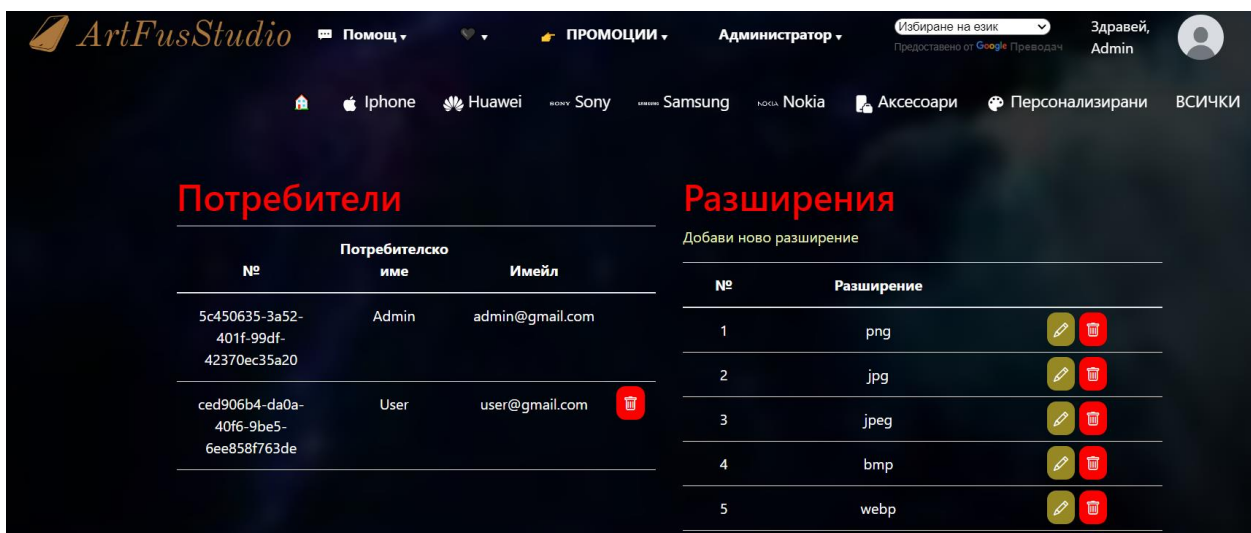
5	Nokia		Nokia 9 PureView	1	599,99 лв.	599,99 лв.	
6	Nokia		Nokia c210	1	349,99 лв.	349,99 лв.	
7	Nokia		Nokia G42	1	599,99 лв.	599,99 лв.	
8	Apple		Зареждащ адаптер	1	12,99 лв.	12,99 лв.	
							Обща сума: 5174,91 лв.
							Общ брой артикули: 9
ИЗЧИСТИ КОЛИЧКАТА							
КУПИ							
ПРОДЪЛЖИ ПАЗАРУВАНЕТО							

Когато потребителят натисне върху бутонът за изчистване на количката или за извършване на покупка се извършва бизнеслогиката от контрола и той се пренасочва отново към списъка с всички предлагани от сайта продукти.

Това обобщава какви са възможностите на обикновения потребител до момента. За разлика от него администраторският акаунт има достъп до повече страници, най-общо до таблото за управление, което се достъпва от споделения изглед, точно преди преводача.

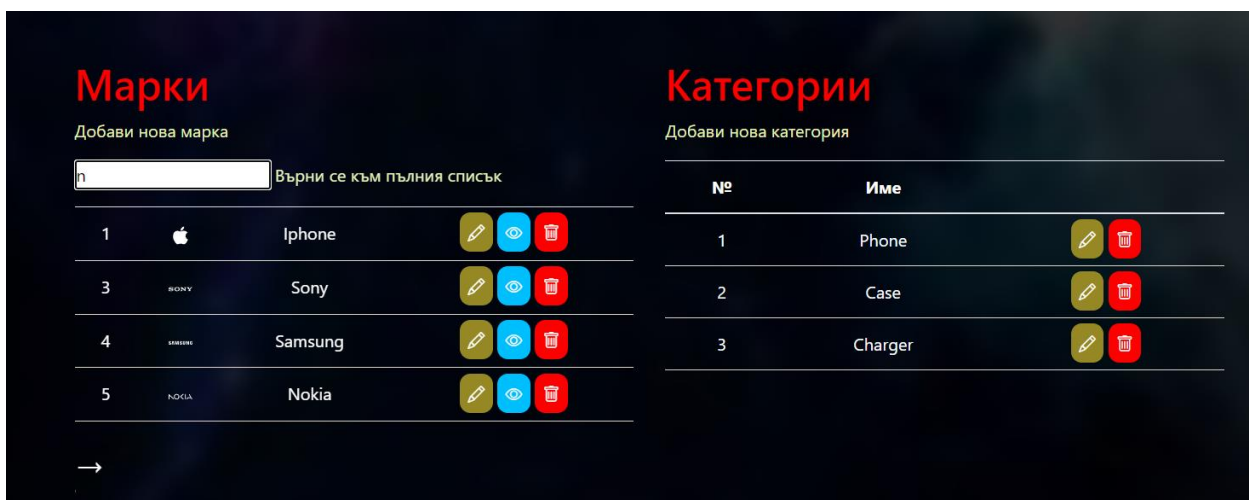


С първата връзка се отива към страницата, от която администраторът има правото да изтрива регистрираните потребители на сайта, с изключение на другите администратори.

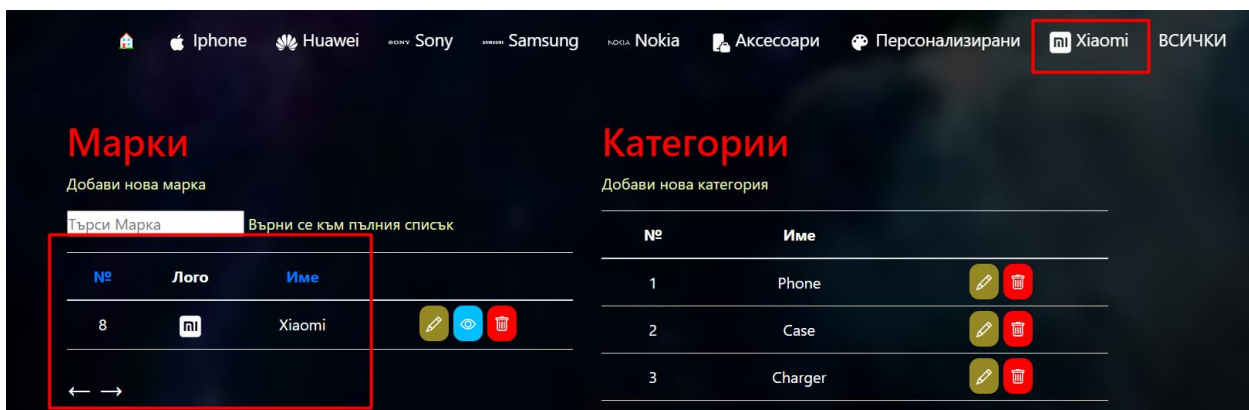


От връзката с име „Марки“ администраторът се отвежда до страница, където може да добавя нови категории, ако в сайта ще започнат да се предлагат нови продукти, както и съответните марки на тези категории. В този изглед се срещат търсачката, както и разделянето на данните на страници. Те работят със събития, като за търсачката не е необходимо да се натисне бутон за потвърждаване на входа директно предлагане резултатите ако има такива, а за

разделянето на данните на страници е необходимо само да се натисне стрелката надясно от клавиатурата за отиване към следващата страница, както и стрелката наляво за връщане назад. Когато страницата е с номер 1, този адвката на ляво е премахната за да се предотвратят грешки. Тази страница изглежда по следният начин:



Особено когато потребителят добави нова марка е че тя ще се показва във всеки изглед, чийто контролер има методът шаблонния контролер за достъпване на таблицата с марки, който е желателно да присъства навсякъде. Промяната би изглеждала по този начин:



Това е причината поради която е оставено място и не е запълнена цялата лента - защото това е основа която ще се използва за надграждане на проекта няма да се изисква сериозна преработка на кода за да може да се добавят нови функционалности.

Когато курсорът на мишката се постави върху препратката, именувана „Характеристики на продукта“, се отваря нов падащ списък с три категории. В тях са обединени няколко таблици, които имат сходни черти - в „Естетически“ са показани характеристиките за един

калъф на телефон – материята, от която е изработен, вида на калъфа и нивото на защита, което предлага; в „Софтуер“ се намират данните за операционната система на телефона - там са записани най-често използваните имена и версии на операционна система, както и всички възможни вариации; На третото място са записани най-често използваните хардуерни спецификации на един телефон - памет на хранилището, RAM, тип на USB, екран на технология, както и броят камери, с които разполага. Тези таблици разполагат с възможността за добавяне на нова стойност, редактиране и изтриване на съществуваща такава.

В групата „Продукти“ се намират продуктите, които се предлагат в сайта, разделени по съответните им категории – на телефони, калъфи и зарядни. Понеже и трите категории наследяват всичко от базовия модел „Product“, показването на данните, критериите за сортиране и филтриране и манипулациите на таблиците са идентични за тях и за всяка нова добавяне на категория продукти. Сортирането, което се предлага, е същото, който се среща и показването на всички продукти заедно. Дизайнът на изгледите също е идентичен. Той изглежда по този начин, в допълнение демонстрация на подреждането на телефоните по рейтинг в низходящ ред:

Телефони

Добави нов телефон

Търси телефон Върни се към пълния списък

№	Марка	Изображение	Име	Текуща цена	Наличност	USB тип	Памет	RAM	Екран	Брой камери	Оценка	Резултат/Оценки	
10	Nokia		Nokia G42	599,99лв.	196	USB-C	32GB	8GB	Retina	4	★★★★★ /4.8/	936 / 195	  
1	Apple		iPhone 15 Pro	2699,99лв.	142	USB-C	512GB	8GB	OLED	5	★★★★★ /4.79/	1202 / 251	  
2	Huawei		Huawei P40 Pro	799,99лв.	72	USB-C	48GB	8GB	OLED	5	★★★★★ /4.77/	167 / 35	  

В изгледа създаването на нов продукт, от която и категория да бъде той, са добавени ограничения използвани в моделите, но и други, типични за входовете в HTML - на полетата за въвеждане на данни са им зададени минимална и максимална стойност, които дори администраторът не може да надвиши. Това е за предотвратяване на случай като отрицателна цена. Причината за разделението на базата данни на толкова много таблици е

че при създаването или редактирането на даден продукт разполагаме с набор от най-често използваните стойности и така стойностите следват един формат на запис:

Наличност

564545646546

Моля, въведете стойност, по-малка или равна на 10000.

Тип USB

Микро-USB

USB-A

USB-B

USB-C

Мини-USB

Микро-USB

Светкавица

14

Екран

Ретината

Брой камери

6

ОС

Android 9










Същото се показва и при редактирането на продукта. В изгледите за детайли и изтриване данните са показани в същия стил, както при детайлите в зоната на потребителя, показани в началото на пътеводителя за потребителя.

Като допълнителна функционалност са добавени купоните за намаления, които свалят цената на артикула, за когото са предназначени, с указания брой проценти и имат дати на валидност. Модераторския панел за тези талони изглежда по следния начин:

Купони

Добави купон за намаление

[Върни се към пълния списък](#)

№	Код за намаление	Процент на намалението	Име на артикула	Начална дата	Крайна дата	
1	BANGO	99%	Iphone 15 Pro	05 .07. 2024	25 .07. 2024	  
2	WXYZ123	49%	USB с зарядно за Nokia	03 .09. 2024	27 .09. 2024	  
3	MNBVCXZ	26%	Sony Xperia	07 .11. 2024	22 .11. 2024	  

Когато администраторът натисне върху името на артикула, той бива препращан към страницата с детайлите за този продукт. При създаването им са поставени ограничения, които предотвратяват случай с процент на намаление над 100 или където началната дата е след датата на изтичане на промоцията. Потребителят може да използва само по един талон за отстъпка на покупка.

