SPECIALIZED HIGH SCHOOL OF MATHEMATICS

"KONSTANTIN VELICHKOV" PAZARDZHIK

DIPLOMA THESIS

Topic: Development of a web application "ArtFusStudio"

Diploma project for the acquisition of a third degree of professional qualification – part of the theory of the profession

Diploma student: Stefan Georgiev Marinkov

Specialty: "Applied Programming" code 481030

Profession: "Application Programmer" code 4810301

Consultant:

Pazardzhik

2024

Content:

1. Introduction

The last decade has seen an increase in the relationship between business and new technologies. In the hectic everyday life, a person has less and less time to search and inspect items to buy, so he prefers the opportunity to search from home, which significantly reduces the time and energy needed to find the desired product. Therefore, more and more companies are turning to the digital space to offer their goods, because this way they reach a larger audience. Thus, the person from home can get acquainted with all the items that the person or company offers in no time.

The technologies that are used to create a connection with the company in the global network are constantly being improved. Phones have become an integral part of our daily lives and more and more people prefer them for their mobility to computers. The uniqueness in the ratio of its length, the height of the screen on different devices is becoming one of the leading criteria for websites.

2. Main part

The project is an interactive dynamic web application with a user-friendly interface that has many functionalities and provides buyers with detailed information about the items displayed on the site.

The site is divided into a public and a hidden part, depending on whether the account is owned by a user or an administrator. The public section contains several categories – "Help" (information), "Promotions", a section for saved items, a list for choosing a language in which to present the content and a shopping cart in which the user saves the items they want to buy at once. For people with more privileges, there is an "Administrator" category, which contains links to the panels for manipulating the main tables in the database - these are the opportunities to add product specifications, the products themselves and coupons with discounts, as well as to manipulate user accounts.

The web app requires users to register and log in to their account if they want to use the features to save items and make a purchase of multiple items at the same time. To purchase one product, the customer is not required to have a user account. The data required when creating this profile is e-mail, username and a secure password.

An N-layer architecture was used to build this application in combination with the MVC architecture, with MVC components located in each of its three main layers

As additional functionalities, the application will include the ability to filter and group products by a certain criterion or group of such, attach images and work with dates.

Finishing activities:

Writing quality program code, following good practices and principles of software engineering;

Providing responsive design for the website for an equally good experience across devices;
Selection of technologies and tools to ensure the successful implementation and future maintenance of the project.

The project aims to create an innovative web application that will not only satisfy the online needs of "ArtFusStudio", but will also offer effective and secure store management, providing a competitive advantage in the digital world.

2.1 Purpose and tasks of the thesis

The goal of this project is to participate in solving one of the main problems of the modern consumer, namely the lack of time. When everything is "*one click away*" and accessible from any type of device, the time required to search for the desired item decreases exponentially. The other main problem that this project undertakes to solve is the problem of choice - when the buyer cannot process too large a set of information, so ArtFusStudio offers a solution such as the option to filter all products offered on the store page and be able to see all those that have the specifications he is looking for.

Another problem that this application solves is the one related to the language barrier between different nationalities. Since websites are publicly available, this means that people from all over the world can use them and, like any project, in order to enjoy more and more users, it should be available in as many languages as possible, because not everyone speaks English.

2.2 Technologies used

ASP.NET Core - creation of the complete Back-end part, as well as the database of the Code-first application;

HTML5 - presentation of the objects from the database;

CSS3 - design of the web page (improvement of the user interface (UI));

JavaScript - improving user experience (UX);

Microsoft SQL - building a relational database (establishing the relationships between different tables through keys);

EF Core - .NET technology designed to work with the database using objects (models);

Bootstrap - using templates to make the site look equally good on devices with different resolutions;

Flexbox, Grid - CSS technologies that improve UI;

Ajax - Adds asynchrony to JavaScript code;

NMT - Neural machine translation, adds the ability to fully translate the content of the website.

2.3 Project implementation

2.3.1 Creating the project and user settings

The web application was created on ASP.NET Core 8 as ASP.NET Core Web App (Model-View-Controller) and then 3 projects of type Class Library were added to this Solution, with the corresponding links.

The project is implemented through the use of a multi-layer architecture, in combination with the MVC model, and each of its components is placed in the appropriate layer: ArtFusStudio (Front-end/Client Layer) contains the code that will be reproduced by the web browser and aims to reach the end user (the application interface). There are the views and their respective controllers, positioned in their respective zones, and they are of 3 types - for an administrator (**Admin**) for a normal user, (**Customer**) and for registration (**Identity**) as well as the database link string **appsettings.json**; ArtFusStudio.Models contains the models built using the object-oriented programming approach with the necessary validations for each of the properties. The Back-end layer (ArtFusStudio.DataAccess) contains migrations and codes that use the ORM technique to populate the tables. Separately, a fourth project (ArtFusStudio.Utility) has been added, where the constant values are recorded. This level of abstraction, which is obtained by breaking down into four parts, is used to reduce complexity, as well as facilitate the maintenance of the application. The references between the individual projects are shown in the diagram below, where A → B stands forthat A uses data from B:

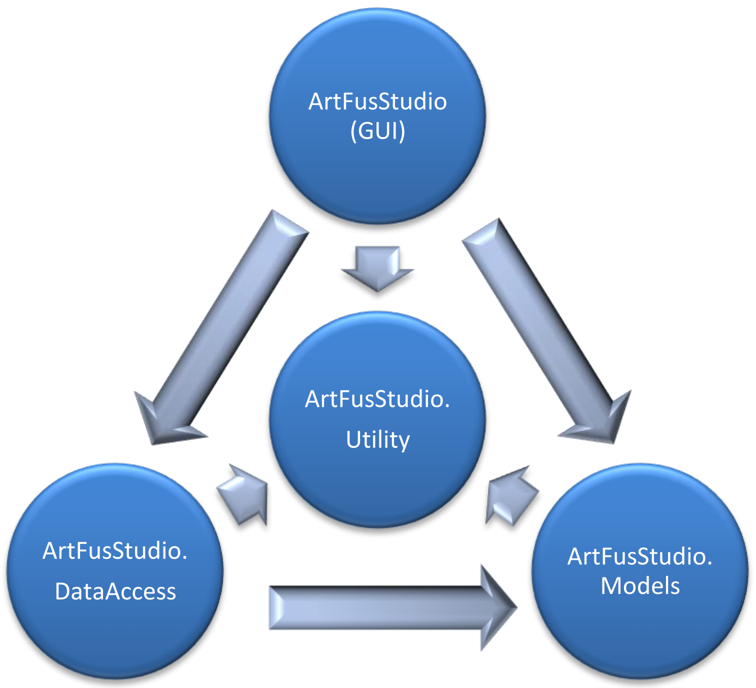


***Figure 1**: References between different layers*

For the implementation of the application, the Code-first technique was used, which allows us to create, manipulate and generally maintain the database through the .NET code. For the presentation and for the data layer, packages from the Microsoft.EntityFrameworkCore family are used (common are SqlServer and Tools, Relational and Sqlite are used in the visualization and Design files for the data) and Microsoft.AspNetCore.Identity (EntityFrameworkCore and Identity for views and Diagnostics. EntityFrameworkCore and EntityFrameworkCore for data manipulation), and only Microsoft.Extensions.Identity.Stores was used for the business layer.

Accounts from both groups are automatically added – client and administrator with login data username and password, respectively (user, User#1) and (admin, Admin#1), and to access the login form it is only necessary to click on some of the buttons that require the user to be logged in to their account in order to perform their intended action, such as the button for saved items (image of a heart, after "Help"), the "Add to cart" button, the shopping cart itself in the upper right corner, the default profile picture or on the text "Hello, sign in". To log out of the account, you need to click on the profile picture or on the text to the left of this photo, where it says "Hello <username>".

2.3.1.1 The attributes used in the supplementary table models are:

- KeyAttribute - indicates that the property is a primary key in the table;
- RequiredAttribute - specifies that the value of the field is mandatory;
- RangeAttribute - fixes an interval of allowed values for the property;
- MaxLengthAttribute - specifies the maximum allowed length to enter an array or string in a property;
- StringLengthAttribute - specifies the minimum and maximum number of characters that the field value can have;
- RangeAttribute - obliges the numeric value of the field to be in a specific range;
- RegularExpressionAttribute - specifies that the value of the field should match a specific pattern;
- DataTypeAttribute - specifies the form by which the dates are to be filled in (up to a day);
- DisplayFormatAttribute - specifies the format by which the dates should be displayed;
- ValidationAttribute - a base class used to create criteria for working with fields of type DateTime;
- ValidationResultAttribute - displays the result of the validation request;
- EndDateValidationAttribute - an attribute created that creates a restriction for the expiration date to be after the date of creation.

2.3.1.2 Attributes used by System.ComponentModel.DataAnnotations.Schema:

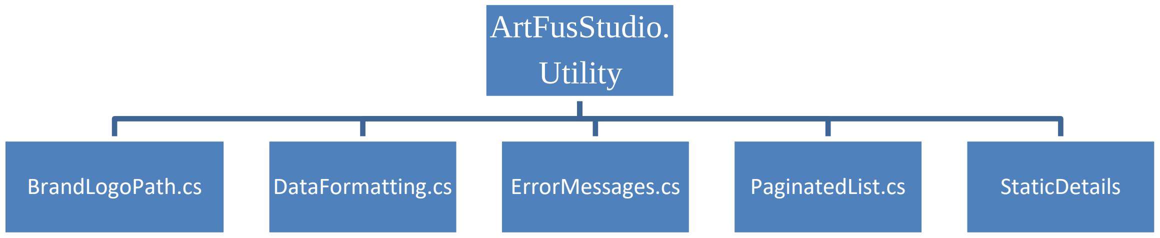
1) ForeignKeyAttribute – used to establish a relationship between different tables in the database

2.3.1.3 Used auxiliary classes in the models:

To facilitate the maintenance of the application, five help classes have been created in ArtFusStudio.Utility, where static data is saved. The ErrorMessages.cs contain the error messages to be used in the validation of model properties. In BrandLogoPath.cs there are SVG commands to create the logo of the respective brand in the form of vector graphics, because this way they will have a higher resolution than images with raster graphics (JPEG and PNG). This data is used when populating the database from ArtFusStudio.DataAccess\ApplicationDbContext.cs. The names of the constants are according to the naming rules in the ASP.NET. The DataFormating.cs contains a method to represent fractional numbers in abbreviated form (replacing " ," with " .", as well as reducing the characters after the decimal point of the fraction by deleting all zeros after the decimal point.

(Attached: *Appendix 1: Implementing the FormatDecimal Method in ArtFusStudio.Utility*)

PaginatedList.cs used not to present all the data at once on pages with N elements in each page. StaticDetails.cs is designed to preserve other static data in the future development of the project. The structure of this project is as follows:



2.3.2 Models

Figure 2: *The file system in ArtFusStudio.Utility*

The model files are located in the ArtFusStudio.Models project and are arranged in folders according to which other models they are associated with and which they inherit. The architecture of this project is detailed in **Appendix 2**.

(I attach *Appendix 2: Architecture of ArtFusStudio.Models)*

2.3.2.1 ApplicationUser Model

The **ApplicationUser** class inherits the **IdentityUser** class, which represents the model of the user data table, the **Users table**.

2.3.2.2 ImageExtension models

This model is designed to store the most commonly used image extensions. It has 2 properties:

1) *Id* (8 – 9 row) – this is a unique identifier for each mark that is added to the database. It is of type int (integer, not greater than 231) and has the attribute:
    - Key - indicates that this property is a primary key and is used to establish a relationship with the different tables in the database.
2) *Name* (11 – 13 lines) – this is the property through which the brand name is accessed. It is of type string and has the following attributes:
    - Required - specifies that a mandatory field must receive some value;
    - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 6 characters).

2.3.2.3 Модела UserProfilePics

This model is designed to store the most commonly used image extensions. It has 2 properties:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
    - Key attribute.
2) *UserId and ImageExtId (11 – 17 rows)* - foreign keys to associate the table with the table for the user and the most commonly used image extensions. No attributes.
3) *FileExtension* (line 19) - stores each user's profile picture extension.

2.3.2.4 Model Brand

The **Brand class** (Appendix: *Appendix 3: The Brand class*) represents the model of the item brand data table – the **Brand** table.

1) *Id* (10 – 11 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
    - Key attribute.
2) *Name (13 – 15 lines) – this is the property through which the brand name is accessed. It is of type string and has the following attributes:*
    - Required attribute;;
    - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

3) *pathD (17 – 18 row) – this is the property through which the brand logo is accessed, which can be called from the svg tag. It is of type string and has the attribute:*
    - Required – described in the Brand model**,** the *Name property*.

Through this model, a table is created to store the data for all models that can be found in the products on the site. It is used in the navigation bar by sending a query to the database to all records.

2.3.2.5 Model Category

The **Category class** (Appendix: *Appendix 4: The Category class*) represents the model of the table with the data of the main groups of items – the **Category table**. The properties in it are:

1) *Id* (8 – 9 row) - has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
    - Key attribute.
2) *Name* (11 – 12 row) - has the same type and purpose as the property of the same name, see the previous model, and has the same attribute:
    - Required attribute;
    - StringLength - specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 128 characters).

Through this model, a table is created to store the data for the main groups into which we can divide the products on the site.

2.3.2.6 Product Models

(Attached: *Appendix 4: The Product Class*)

This is one of the fundamental models in the project, which is intended to be used as a base class in all newly created models of a product category. It is also used to assign all items that are sold on the site without the need for table merging. It contains the properties that will be present in any product. They are:

1) The *Id property* (10 – 11 rows) - has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:

    - Key attribute

2) The Name property (row 13 – 15) - has the same type and purpose as the property of the same name, see the previous model, and has the following attributes for validating its value:
    - Required attribute;
    - StringLength - has the same length constraints to the field value as the above-mentioned attribute with the same name.
3) *Description* (17 – 19 row) - used to give a description of each product within the relevant limits. Therefore, it is of type string. They are:
    - AllowNull attribute - the product does not have to have a description;
    - StringLength - specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 512 characters).
4) *ProductURL* (line 21 – 23) - stores the path to the file with the product image, which is located in a subfolder that bears the name of the category from which the product is from, in the main directory (~/images/Products/<category name>/<image name>.png). It is similar to type string. It is desirable that the extension of this image is PNG, because it maintains a transparent background. This property has the following validation:
    - Required attribute;
    - StringLength - specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 64 characters).
5) *OldPrice* (25 – 27 row) – contains data about the previous value of the item. It must be of type decimal. This value has the following limitations:
    - AllowNull attribute;
    - Range - it turns out that the value of the field should be between 0.0 and 100000.0.
6) *CurrentPrice* (29 – 31 lines) – contains data about the current value of the item. It must be of type decimal. This value has the following limitations:
    - Required attribute;
    - Range - the value of the field must be in the same range as in the previous property.

7) ,7) *Linking the table to the category and brand tables using the ForeignKey attribute* (row 32 – 41): CategoryId and BrandId of type int.
8) *TotalVotes* (43 – 44 rows) - stores the total number of votes that the product has received from users. It must be of type int. By default it is set to 0, when an object is created and inherits this class it will have 0 ratings. This value has the following attribute:
   - AllowNull attribute;
9) *Score* (46 – 48 rows) – stores the sum of all votes that the product has received from users. It must be of type int. By default, the value is set to 0, when an object is created and inherits this class it will have a 0 score. This value has the following attributes:
   - AllowNull attribute;
   - ScoreValidation (row 62 – 75) – an attribute that inherits **the ValidationAttribute** and is used to create a new rule that states that the value of the field must be greater than or equal to the value of the *TotalVotes field* and less than or equal to the 5 * value of the *same field*.
10) *Rating (50 – 61 row)* - this field is used to calculate the rating of each item, as it cannot be assigned a value, it receives it according to the formula and is reformatted using the FormatDecimal method described above. $Round\ (\frac{\text{стойността на } Score}{\text{стойността на } TotalVotes},\ 2)$
11) *Quantity (79 – 81 lines)* - this property is used to record how many products are in stock. Its validation attributes are:
   - Required attribute;
   - Range - the value of the field should be in the range from 0 to 1000.

2.3.2.7 OperatingSystem models

Product has several classes that inherit it. One of them is the one for phones, but since the same phone model can have several different values for one property, the tables from the database are normalized to a normal form of Boyce-Code, which states that each of the properties that can have several values for the same object should be separate tables and the connections to them should be made through foreign keys:

1) *Id* (9 – 10 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.

2) *OSName* (12 – 14 lines) – stores the names of the operating systems that can be connected to a particular phone. Similarly, it is of type string It has the following attributes:
   - Required attribute;
   - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.8 OperatingSystemVersion models

The operating system must have a version in addition to a name. This table also has only 2 properties:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *OSVersion* (11 – 12 lines) – stores the possible values for the version of the operating system. It must be of type decimal, and has the same attribute:
   - Required attribute;
   - RegularExpression – sets a template for the value of the property: the decimal point to be " ." and after it there should be up to 2 digits, e.g.: 12.42
   - Range – it turns out that the value of the field should be from 0 to 99

2.3.2.9 Models OSNameAndVersion

This model is a union of the previous two, and is associated with them through their primary keys and the content to print all possible combinations of operating system with version. The properties it has are:

1) *Id* (9 – 10 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *OSNameId* (row 12 – 14) – establishes a connection with the table storing the names of the operating systems in order to retrieve its values.
   - ForeignKey attribute.

3) *OSVersionId* (row 16 – 18) – establishes a relationship with the table storing the versions of the operating systems in order to retrieve its values.
    - ForeignKey attribute.
4) *OSNamePlusVersion* (line 20 – 26) – used to return a concatenation between the operating system name and all versions that match it. It has no attributes and the value cannot be set externally, because there are no "*set*" access modifiers.

2.3.2.10 Camera Model

This model finds the possible values to be set as the number of cameras that a phone can have. The properties characteristic of the model are:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
    - Key attribute.
2) *CameraCount* (11 – 14 lines) – Contains the most commonly used values for the number of cameras that each phone can have. It must be of type byte. It has the following attributes:
    - Required attribute;
    - RegularExpression – sets a template for the value of the property: the decimal point should be "." and after it there should be up to 2 digits
    - Range – it turns out that the value of the field should be from 0 to 99

2.3.2.11 DisplayTechnology models

In this model are the possible values to be set as a screen technology that a phone can have. The features of this model are:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
    - Key attribute.
2) *Name* (11 – 13 rows) – has the same type and purpose as the property of the same name of the previous models, as well as has the same attribute:
    - Required attribute;

- StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.12 Memory Models

In this model are the possible values to be set as RAM that a phone can have. It has the following properties:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *RAM* (11 – 13 rows) – contains the most commonly used RAM capacity values in gigabytes, Therefore, its type must be of type int, as well as has the following attributes:
   - Required attribute;
   - Range – it turns out that the value of the field should be from 0 to 32. To find you

2.3.2.13 StorageCapacity model

This model contains the possible values to be set as the storage capacity that a phone can have. It has the following properties:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *CapacityGB* (11 – 13 rows) – The property is of type decimal and contains the most commonly used values for storage capacity in gigabytes, as well as has the following attributes:
   - Required attribute;
   - Range – specifies that the field value should be from 0 to 2048 (2TB).

2.3.2.14 USB Models

In this model are the usual charger models compatible with the phone. It has the following properties:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:

   - Key attribute.
2) *Type* (11 – 13 row) – saves the values of the most common charger models in phones. It has the following attributes:
   - Required attribute;
   - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.15 Phone Models

This model will be used to create one of the main items that will be sold in the store – namely the phone. Since it is from the main groups of items, it inherits the base class Product and takes its properties. In addition to them, six more properties are contained for the body of the class – the foreign keys of the tables, corresponding to models from number 7 to number 12, which establish a relationship of the type "many to one", i.e. many tables connected to one.

2.3.2.16 CaseMaterial models

In these models, we continue the practice of the previous ones and create a model for each of the properties of the class, aiming to describe the next group of products – the phone case. CaseMaterial stores the most commonly used values for the material from which the case is made. The properties it possesses are:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *Name* (11 – 13 row) – has the same type and purpose as the property of the same name of the previous models, as well as has the same attributes:
   - Required attribute;
   - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.17 CaseType models

Here the most commonly used types of cases are preserved, that is, whether it is closed, only the back or another. The properties of the class are:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *Name* (11 – 13 row) – has the same type and purpose as the property of the same name of the previous models, as well as has the same attributes:
   - Required attribute;
   - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.18 ProtectionLevel models

This is where the level of protection that the case offers is preserved. The properties are common for this type of model:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *Protection* (11 – 13 rows) – records the most commonly used levels of protection. It is similar to type string. Its attributes are:
   - Required attribute;
   - StringLength – specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

2.3.2.19 Model Case

This model will be used to create one of the main items that will be sold in the store – namely the case. Since it is from the main groups of items, it inherits the base class Product and takes its properties. In addition to them, three more properties are contained for the body of the class – the foreign keys of the tables, corresponding to models from number 14 to number 16, which establish a relationship of the type "many to one", i.e. many tables connected to one.

2.3.2.20 Charger Models

This model will be used to create one of the main items that will be sold in the store – namely the phone charger. Since it is from the main groups of items, it inherits the base class

Product and takes its properties. In addition to them, three more properties are contained for the body of the class – they are:

1) *OutputVoltage* (9 – 11 row) – The property is of type int and contains the value of the output voltage in volts. Its attributes are:
   - Required attribute;
   - Range – it turns out that the value of the field should be from 0 to 24.
2) *OutputCurrent* (13 – 15 row) – The property is of type int and contains the value of the output voltage in volts. Its attributes are:
   - Required attribute;
   - Range – it turns out that the value of the field should be from 0 to 24
3) *FastChargingSupport* (17 – 18 row) – The value of this property is Boolean, serving as an indicator of whether this charger supports fast charging. It has only one attribute:
   - AllowNull attribute.

With the creation of this model, they include the initialization and description of the products. Now the additional functionalities begin.

2.3.2.21 ShoppingCartItem model

After the user has identified the items he wants, it is the turn of the last part of the interaction, namely placing an order. This model will describe the standard object in the shopping cart with the necessary properties:

1) *Id* (9 – 10 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *ProductId and CartId* (12 – 19 row) – connect the product from the shopping cart with its respective item in the store, as well as with the cart in which it is located.
3) *Quantity* (21 – 23 row) – has the same type and purpose as the Product model property of the same name, as well as has the same attributes:
4) *Id* (9 – 10 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:

- Key attribute.

2.3.2.22 ShoppingCart Model

After the user saves the items he wants, it is the turn of the last part of the interaction, namely placing an order. First of all, the shopping cart should be created as a model where the products that the customer intends to buy will go. The properties in this model are:

1) *Id* (8 – 9 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *Id* (9 – 10 row) – has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
3) *UserId* (12 – 14 rows) – A foreign key is created to connect the user to the corresponding shopping cart. There are no attributes.
4) *Items* (15th row) - Create a list from ShoppingCartItem. No attributes.
5) *TotalPrice* (18 – 30 row) - used to calculate the total amount that the customer has to pay.
6) *TotalQuantity* (32 – 36 row) - used to show the total number of all items in the shopping cart.

2.3.2.23 Model Coupon

However, before finalizing the order, the user has the option to enter discount coupons to reduce the price. The properties of such a coupon are:

1) *Id* (8 – 9 row) - has the same type and purpose as the property of the same name of the previous model, as well as has the same attribute:
   - Key attribute.
2) *DiscountCode* (12 – 14 lines) - this is the code that the user writes in order to receive the corresponding discount on the desired item. Its type is string. The validation attributes here are:
   - Required attribute;

- StringLength - specifies that the number of characters of the field value should vary in a specific interval (in this case, between 2 and 32 characters).

3) *DiscountPercentage* (16 – 18 row) - Here is saved how many percent the discount of the item with the corresponding code is. The type of property is decimal. The validation attributes here are:
    - Required attribute;
    - Range - it turns out that the value of the field should be from 0.0 to 99.9
4) *ProductId* (20 – 23 row) - this is the foreign key that is used to take a product by identifier to whom this discount code relates. There are no attributes here.
5) *StartDate* (25 – 28 lines) - This is where the date from which this discount coupon is valid is stored. The type of the property is DateTime. The validation attributes here are:
    - Required attribute;
    - DataType(DataType.Date) - specifies the format in which the date should be when recording and visualizing. In this case, only the date is displayed, no hours;
    - DisplayFormat - specifies the format of the date presentation. In this case, it will appear in the form "*DD .MM. YYYY"*.
6) *EndDate* (25 – 28 lines) - This is where the date until which this discount coupon is valid is stored. The property type is DateTime. The validation attributes here are:
    - Required attribute;
    - DataType(DataType.Date) - specifies the format in which the date should be when recording and visualizing. In this case, only the date is displayed, no hours;
    - DisplayFormat - specifies the format of the date representation. In this case, it will come out in the form
      "*DD .MM. YYYY"*.
    - EndDateValidation – (Appendix: *Appendix 5: Implementing the Date Validation Attribute*) the manually created validation attribute that keeps track of the coupon's expiration date is after the date from which it is valid.

2.3.3 Tables

The tables are created and populated through migrations. Database schema migration is a controlled set of changes aimed at modifying the structure of objects within a relational database. It is used to transition the circuit from one state to another, making sure that the changes are gradual. In the context of this project, migrations are implemented using EF Core 8, which is an understanding of technologies that greatly facilitate working with data by increasing the level of abstraction I have accepted to work with it, that is, applications are created with less code. The very transition from models to tables is possible thanks to the paradigm of object-relational mapping (ORM) - a database of virtual objects is created, which can be used by the corresponding programming language, in this case C#.

In this project, the names of the tables correspond to the names of the models and the relationships between them, again, as in the models.

(Attached: *Appendix 6: The Connections between the Phone Table via Foreign Keys*)

(Attached: *Appendix 7: Detailed Description of the Phone Table*)

The style that is present in the two above-mentioned applications is the same for each table in the database schema (names, relationships, data types and constraints), and even in the models it is given by a detailed definition, because of the methods in the models.

2.3.3.1 The Brand Table

The first table that has been created is of the main patterns that will be found in the products. This is the Brand table, the structure of which is as follows:

Brand		
Name	**Type**	**Restrictions**

Id	INT	PRIMARY KEY, NOT NULL
Name	NVARCHAR(32)	NOT NULL
PathD	VARCHAR(MAX)	NOT NULL

Table 1. Field names and types from the *Brand table*

Description of the columns from *Table 1*:

- Id – unique identifier of the respective product;
- Name – the name of the brand;
- PathD – this is the svg path in the coordinate system where the logo of the respective brand will be drawn, because bootstrap cannot have all possible emblems, and some have reserved rights and limited access.

2.3.3.2 The Category Table

The next table that has been created is of the main groups in which the products will be distributed. This is the Category table, the structure of which looks like this:

Category		
Name	**Type**	**Restrictions**
Id	INT	PRIMARY KEY, NOT NULL
Name	NVARCHAR(32)	NOT NULL

Table 2. Field names and types from the *Category table*

Description of the columns from *Table 2*:

- Id – unique identifier of the respective product;
- Name – the name of the category.

2.3.3.3 The Product Table

The master table is the Product, which contains the properties and validation methods that each newly created product inherits from it. It is used to present all products to users (groups them). It increases the level of abstraction, with only one line of program code accessing the other tables. It looks like this:

Product		
Name	**Type**	**Restrictions**
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

Table 3. Field names and types from the *Product table*

Description of the columns from *Table 3*:

- Id – unique identifier of the respective product;
- Name – the name of the item;
- Description – description of the product;
- OldPrice – the old price of the product (added to compare with the current one and to highlight which products are discounted);
- CurrentPrice – current price of the product;
- TotalVotes – the number of ratings that users have given for the respective product;
- Score – the sum of all product ratings;
- CategoryId – the link to the Category table to add the product to its adjacent category;
- BrandId – Link to the Brand table to record the branding of the product.

Relationships of the *Product table*:

- A product can be of only one brand and be part of only one category. Here, the connection of tables is "one to one".

2.3.4 Authentication and authorization in the app

One of the fundamental things in information security is the CIA/PIN (Confidentiality, Integrity and Availability) triad. This model states that information must be protected from unauthorized access. The authenticity of the data, that is, not to alter it is a casual user, is essential and that it is accessed quickly, but only when the user needs it, and they must not reveal any information that is contrary to the laws on personal data protection.

The site is built according to this model, being divided into two roles with different privileges – "Customer", who can view the products that are sold, does not make purchases and use the shopping cart to save the items he wants for later, and "Admin", who can manipulate the data in the database through the graphical user interface of the web application. With this, the first major component of the triad (privacy) is fulfilled.

For the second (data integrity), when creating this application, it is divided into three zones: "Admin", "Customer" and one created automatically – "Identity", which contains the files necessary for authentication. These zones are used to restrict access to specific pages of the website, with only those users whose account is marked as "Admin" accessing in the "Admin" area.

The third main pillar is built with the creation of the models, and this division into very small tables with two rows ensures that the values of the desired property will be accessed much faster, because less data is traversed.

Generally speaking, with authentication and authorization, the authenticity and integrity of users' data and privacy are guaranteed by the site

2.3.5 Controllers

MVC controllers are responsible for responding to a query that a website user has made through the interface. Each query is associated with a specific controller. Example queries are for performing CRUD operations on a table from the database (creating, editing, deleting and displaying detailed data). Any public method that is added to the controller can be invoked by any user of the site by entering the correct URL in the address bar of the web browser. The method in the controller is used as an action. The action returns *an* Action Result in response to the request from the client.

The queries that are used in the project are GET and POST. The GET method is used to query data from a specific source and is used in the Index and Details methods, and the POST method is used to send data to the server through the Create, Edit and Delete methods.

In this project, the controller creates a new record in the relational database, if there is none, when the user wants to add an item to their shopping cart. It is also used for an administrator to be able to manipulate tables that are related to products or users, such as editing product data (name, description, price, etc.) or adding data to other tables (e.g. adding a new product brand, new category, etc.). Controllers are automatically generated by a template to perform CRUD operations on the specified tables (when created, the model corresponding to that table is specified to apply all the attributes to validate the value of the property.

The controllers are distributed among the respective zones (Admin and Customer) in the presentation layer (ArtFusStudio). In the one intended only for administrators, there are those who manipulate the tables describing the products or their properties, and in the one intended for each user, there are those who only present the data to customers and improve the UX.

2.3.5.1 A controller that is used to take data from other tables

DisplayLayoutController – is the controller that is used to access several different tables in a single view, because it can only be called up to one *@model* per file. Its methods are called in each method of a controller corresponding to the view in which we want to access the data from the particular table, just before it returns the value. Its methods contain the ViewData["<Class>"] property, which is activated when the controller action starts.

(Attached: *Appendix 9: Part of the DisplayLaoutController methods*)

2.3.5.2 A controller that is used to rearrange the data from a table

SortingController – is the controller that is used to sort the data in some of the tables with more data according to certain criteria, which is the value of a field inherited from the base model, even before the migration to a database. It is called in the method of the controller corresponding to a product, after all the tables to which this one is associated are accessed.

2.3.5.3 Controllers in area Admin

They are needed to manipulate the base through the user interface. The methods used in the controllers are the typical ones (Index, Create, Edit, Delete, DeleteConfirm, Details, etc.), which are generated automatically using the new functionalities added in EF Core 8 (Right click on the Controllers folder in the corresponding role → Add → Controller... → MVC Controller with views, using Entity Framework → select the desired model that we want to use to implement the ORM and create an identical table in the database → Select the DbContext class → Add and start the Scaffolding process (This is a working framework of ASP.NET MVC for automatically generating code that interacts with the database, i.e. the main CRUD operations)). The controllers that have common reactions are those associated with the following tables: Brand, Category, Camera, Memory, OperatingSystem, Version, StorageCapacity, USB, Phone, CaseMaterial, ProtectionLevel, Case, Charger and Coupon and this is the addition of access to the Brand, Category and Product tables by any method of these controllers because it improves the user experience. Additionally, only for controllers that have a connection to any of the products, the CategoryController, CamerasController, MemoriesController, OperatingSystemsController, OperatingSystemVersionsController, StorageCapacitiesController, USBsController, CaseMaterialsController, and ProtectionLevelController have deleted the Index() method because the retrieval of data from multiple tables is merged into a single view. Before returning the value in each of the methods of the mentioned controllers, the auxiliary controller is called as follows: DisplayLayoutController.AcceessAllTables(this, _context); The other controllers have the same add-ons, but in addition other necessary methods are called from this controller:

- With OSNameAndVersionsController, the AcceessAllTablesOS method is called in the Index method, thus displaying records of all existing operating systems, their versions and all possible combinations of the previous two;
- With the DisplayTechnologiesController, the AccessAllHardwareTables method is called in the Index method, displaying records of all existing hardware components (number of cameras, screen technology, RAM, storage memory, and USB port type);
- With CaseTypesController, the AccessAllAestheticTables method is called in the Index method, thus displaying records of all existing aesthetic elements for a single case (the material from which it is developed, the type of one and the level of protection it offers to the buyer).

In addition, a controller has been created to be used for user management, i.e. deleting them (ApplicationUsersController). Since users have the option to add a picture profile, a model controller has been created that saves the most common image extensions (ImageExtensionsController).

2.3.5.4 Controllers in the Customer area

In this area, the number of controllers is significantly smaller than those in the admin panel - only 4 are found here:

- HomeController – there is only one Index() method for the home page of the website;
- ProductsController – it is designed to display all items that are sold on the site, as well as to display detailed data for each of them. It contains options for filtering by brand by the user, as well as to rearrange all products according to a certain criterion, in this case by name, price, rating and total number of votes.
- ShoppingCartController – this controller contains the business logic behind the shopping cart. When a user wants to add an item to their cart, a GET query next to the table checks whether that user has an already created shopping cart or if a new one needs to be created. Then, when he wants to add an item to his cart, the same request checks whether this item is already there and if it is in the cart, then its number is increased by 1, instead of its name appearing twice. After the user has selected the items he wants, there are two options - to delete his cart or to purchase the items. Separately, it also offers the possibility for the user to delete only one item from his cart.
- The last controller is the one that allows users to add, change and delete their profile picture, as well as frames it to be in specific extensions (png, jpg, jpeg, tif, tiff, bmp). This is the UserProfilePicsController. It operates on the table that contains a record of all user photos and associates them with the accounts. The user is provided with information only about his photo. When someone wants to add a correct photo, a new record is created in the table, and the name of the photo becomes the unique identifier of the user, because this way there will be no files with the same names. It has a check: if the user already has their own profile picture and wants to replace it with another one, it does not create a new record, but simply modifies the existing one, as well as rooms in the profile picture and I replace it with a new one, whose name is the same, because the user ID does not change. Images can be accessed using streams.

2.3.6 Views

Views in ASP.NET Core represent the files with the extension "*cshtml*" that are part of the presentation layer. They are returned by the actions of the controller. Their main purpose is to present data from some data structure (collection) to the user. There are several types, in the context of the application, Razor Pages are used, which allow mixing HTML markup with C# code.

2.3.6.1 Template view

This is the view whose content is displayed for each web page. In this project, it is called "*_Layout.cshtml*" and is located in the Views/Shared folder. It contains references to all used CSS files, as well as JavaScript codes, because being in a shared view reduces code repetition. The HTML content of the file is the navigation bar, as well as the bar with the brands of the products that are offered on the site, as well as the background, which is common with all pages. In the top block are the site's logo, contact buttons, saved items of the user, promotions, links to the pages of the admin panel, the ability to log out or log in to an account, the integrated Google translator with which users choose the language in which to view the content and the shopping cart. The block below it contains a reference to the homepage, a link to the list of all goods offered, and the options for filtering by brand.

2.3.6.2 Views in area Admin

The views, just like controllers, are created using EF Core Scaffolding and are several files for each controller. The basic plan for the architecture of the view of a particular model is to have 5 files with the extension cshtml in the folder with the name corresponding to the controller: Create, Delete, Details, Edit and Index, and in some cases the index is deleted because the representation of data from tables with common features occurs in a single view (e.g. data from *OperatingSystem*, *OperatingSystemVersion* and *OSNameAndVersion* are displayed on a single page of the site; in the same way, hardware characteristics are grouped, as well as aesthetic and trademark characteristics with the categories to which we assign a particular product).

(Appendix: *Appendix 10: Display data from multiple tables at once*)

The structure of each Razor page with a similar name is identical:

- It is typical for "Index.cshtml" to present the data in the form of a table. At the beginning of the site we add the necessary libraries, as well as establishing the relationship with the table in the form of @model. @model directive allows access to the list of objects or to the object if it is not a type in the IEnumerable<T> collection that the controller passes to the view. The model is highly typed. In order for the website to look equally good on as many devices as possible, the table is inserted into a series of div tags with Bootstrap-specific class names for easier processing. However, links with an ASP action are added before the table (in the context of this file, this is the button to add a record to the table). Before the tables a lot of data, after it follows a form in which a search engine is included, where the administrator can filter all records by certain criteria, from any column. (I attach: *Appendix 11: Using the Search Engine for Phone*)
  The search engine is implemented via JavaScript, but in order for it to work, the table needs to be set to Id="myTable".
  (Attached: *Appendix 12: The implementation of search engines*).

  Then comes the actual part of presenting the data. First a thead is created, where the column headers are recorded, then comes the turn of tbody, which is the actual part here. There, with one foreach loop, the entire table is traversed and each record distributed by columns is displayed. On the far right after printing each row of data are the action buttons - to edit the record, to delete it and possibly if there are more fields and to display its details with details. For the stylization of these buttons, the icons from Bootstrap are used. They direct us to an action, the method of which is present in the control of the corresponding object or objects.

- In the groups of views in whose controllers the methods have been deleted, the views of the same name have also been deleted. The next view that is present in each folder in Views is "Create.cshtml". It is designed to create a record in the specified table from the database via an HTTP request from the client to the database. Again, in order for the interface to adapt to different resolutions, all the content is placed in a series of div tags and then in a form. This is where the input fields for the value of the new record are located, and below them is the validation that we have saved for the property in the model. Finally, there is a link to the previous page.

- During the creation of the object, some error may be made, so it is important to add a menu to edit the value of any of the properties. This is the function of "Edit.cshtml". The same techniques are applied there as in the previous view.
- After changing the content, there should be the possibility of a detailed overview of only this object if it has many properties and not all of them can be presented on one page. Therefore, the "Details.cshtml" view has been added. It shows the value of each of the properties of the particular product. Finally, a link has been added to the edit panel if we want to change any of the values.
- Finally, when the entity will no longer be present on the site, it should be deleted. However, the delete button can be pressed by mistake, so a new view has been added to confirm whether the administrator wants to delete the corresponding record from the database. This is the "Delete.cshtml" view, which contains the object data that the user wants to delete, as well as a confirmation button and a link to the previous open page.

2.3.6.3 Views in the Customer area

- There the views are from 4 folders. In the first is the view of the homepage, which is the first thing the user sees from this website. The three phones with the highest rating are shown there and the user is offered the option to add them to a cart or view them in more detail.
- The next folder (Products) shows all the items offered on the site are their general properties that inherit from the base class. There, the customer is offered the opportunity to add them to a cart, purchase them immediately or see more information about them.
- After adding them to their cart, it is their turn to be able to view what items are there. This is the essence of **CartView** view so that the user can operate with their cart from there.
- The purpose of the latest views is to provide the user with the opportunity to attach their profile picture, as well as edit it. He selects the photo from his local file system and it is saved in a folder in the root directory named his unique identifier.

2.3.6.4 Views in the Identity area

This is where the views required for user authentication are located, i.e. Create an account, log in to an existing account, or log out of your account. To register, the user needs to provide his/her e-mail, choose a username, a password that meets security considerations and a wish to attach his/her profile picture. It is possible that after the user has already logged in to his account, he will remain logged in to his account, when he next opens this site.

2.3.7 JavaScript (JS)

JavaScript is an extremely useful programming language because it can access HTML tags through their class or unique identifier and change their style. Its main application in web development is to speed up the website and improve the user experience by reducing the actions that the user has to do to find all the functionalities that the web application offers him. Use already on the homepage for rotating item effects as well as translator integration.

Its other application is in the home page are those small details that distinguish this site from the rest. These are the stars that show the rating of the respective product, as well as the effect that occurs when the user presses on the heart to save the product for later. Because these stars require about 200 CSS code, but with mathematical dependence and unnecessary repetition, JS was inserted, which in one loop generated the necessary code for styling the stars.

This programming language is used on the website because some processes are done in less time when JS code is compiled instead of C#. The most obvious example of this is the search engine in the administrator stations, which works with events and crawls each cell of the table and displays only those rows in which there is a match between the string of characters saved by the user and whether one of the cells occurs in the corresponding order.

Separately, in some views for printing all data from tables, the rows become too many and in addition to the search engine, pages have been added that can be changed both with buttons and through the keyboard arrows.

When using JavaScript, the condition is met that everything it is integrated into should load in no more than 2 seconds.

3. Conclusion

This project is a dynamic In-app in which users can browse the rich range of items divided into categories and use the search engine to filter and find the items they want with specific criteria. Thanks to the detailed database, customers can search for the item by any of its properties, such as price, or specific to the category, such as screen technology. With the integrated translator of Alphabet (Google), the goal is to significantly increase the audience of the site and people who do not understand Bulgarian to be able to use it without problems. This significantly increases the user experience, combined with the user interface.

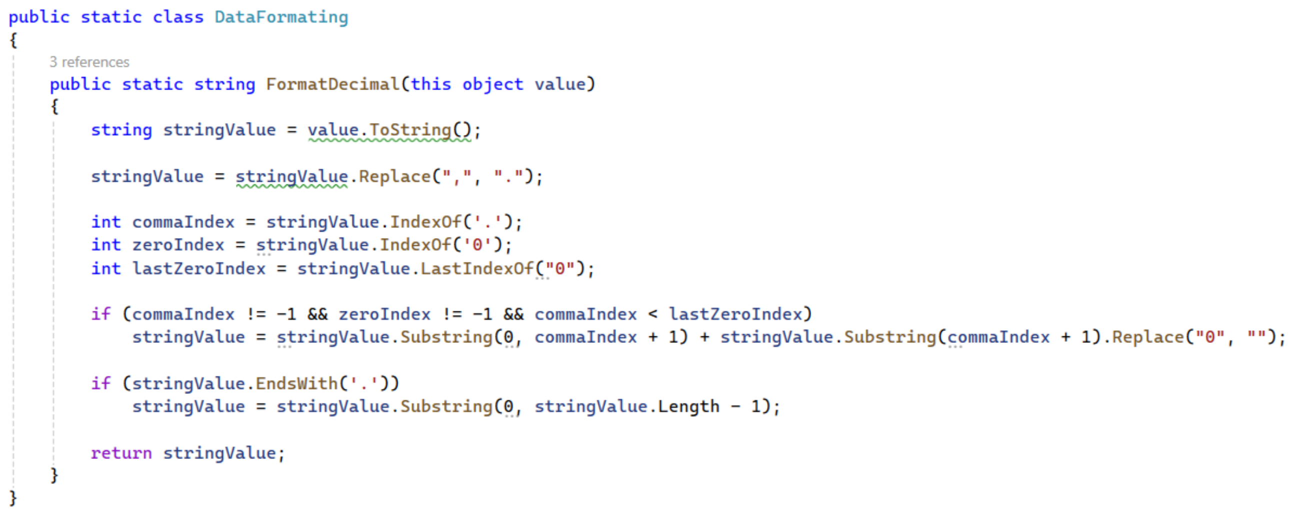
In terms of authentication, the project is divided into two roles - "Customer" and "Administrator", as the administrator has the right to manipulate any table from the database - to add, edit or delete a product or any of its properties, and the user has the right to change his profile picture, add items to the shopping cart and place orders.

Future plans for the development of the application:

- Adding more functionalities in the Identity zone;
- Adding the business logic for evaluating items
- Add the Business Logic Collection of Saved Products
- Adapting the website to more and more devices.

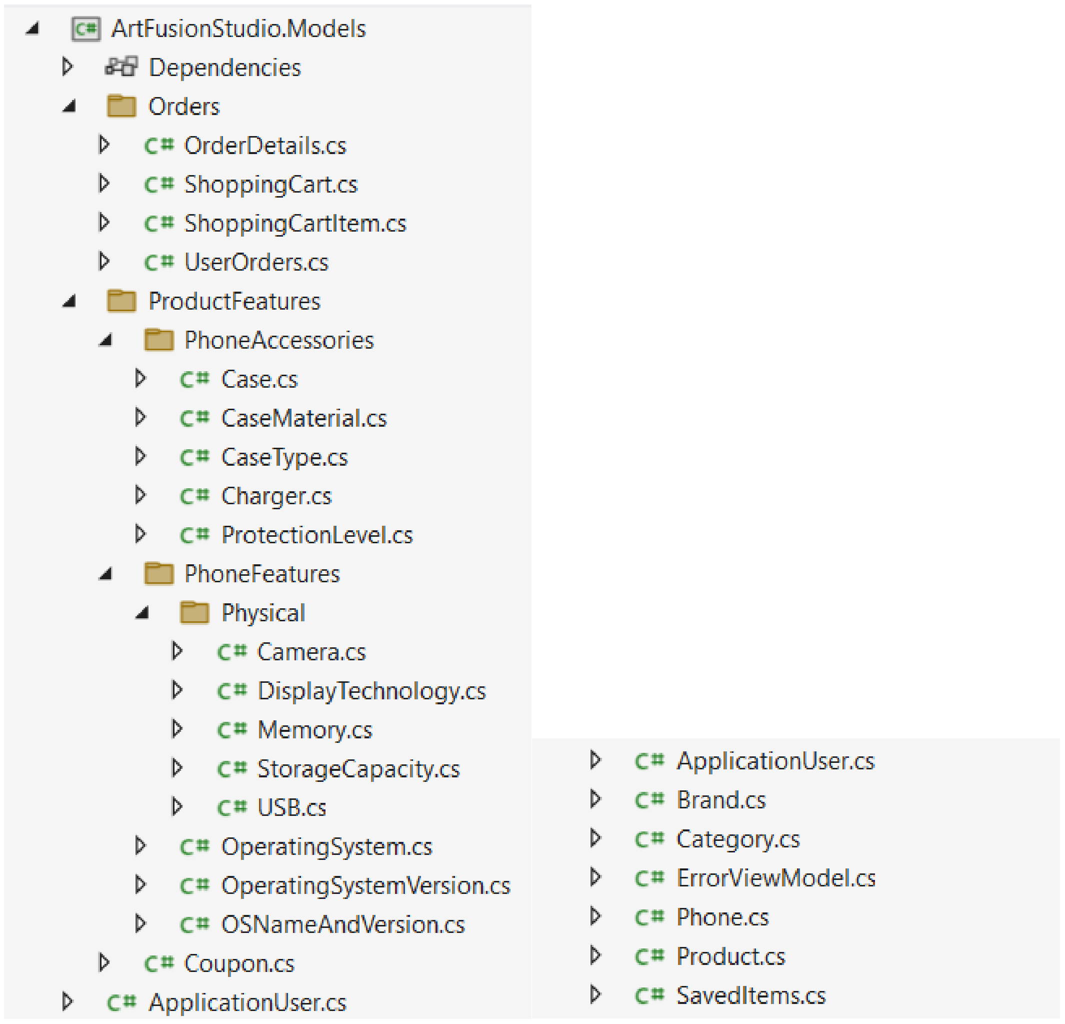
4. Applications

APPENDIX 1: Implementing the FormatDecimal Method in ArtFusStudio.Utility



```
public static class DataFormating
{
    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;
    }
}
```

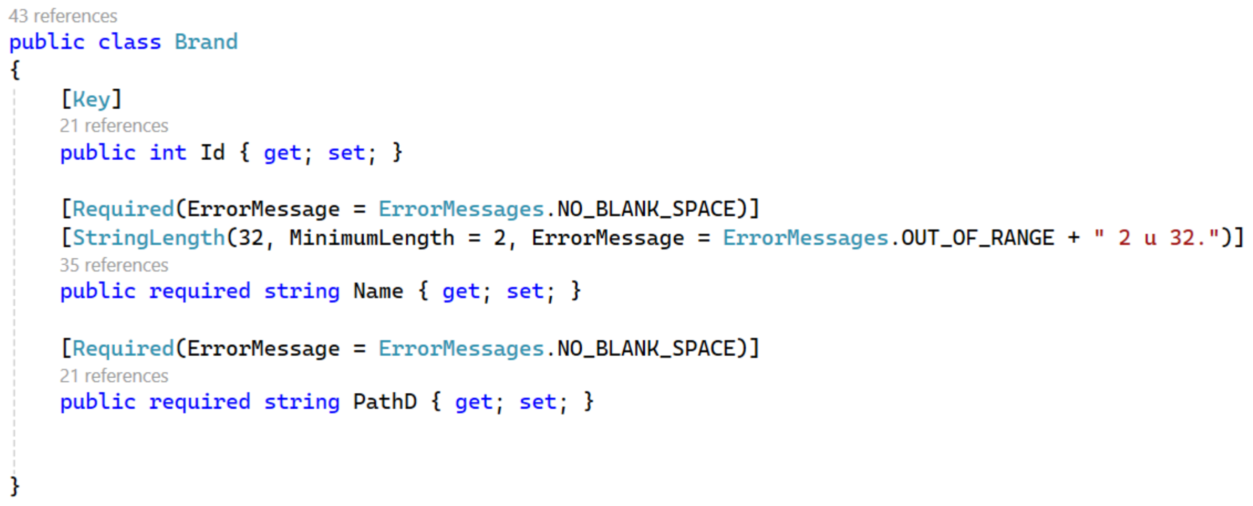
APPENDIX 2: Architecture of ArtFusStudio.Models



```
◢ ArtFusionStudio.Models
  ▷ Dependencies
  ◢ Orders
    ▷ OrderDetails.cs
    ▷ ShoppingCart.cs
    ▷ ShoppingCartItem.cs
    ▷ UserOrders.cs
  ◢ ProductFeatures
    ◢ PhoneAccessories
      ▷ Case.cs
      ▷ CaseMaterial.cs
      ▷ CaseType.cs
      ▷ Charger.cs
      ▷ ProtectionLevel.cs
    ◢ PhoneFeatures
      ◢ Physical
        ▷ Camera.cs
        ▷ DisplayTechnology.cs
        ▷ Memory.cs
        ▷ StorageCapacity.cs
        ▷ USB.cs
      ▷ OperatingSystem.cs
      ▷ OperatingSystemVersion.cs
      ▷ OSNameAndVersion.cs
    ▷ Coupon.cs
  ▷ ApplicationUser.cs
```

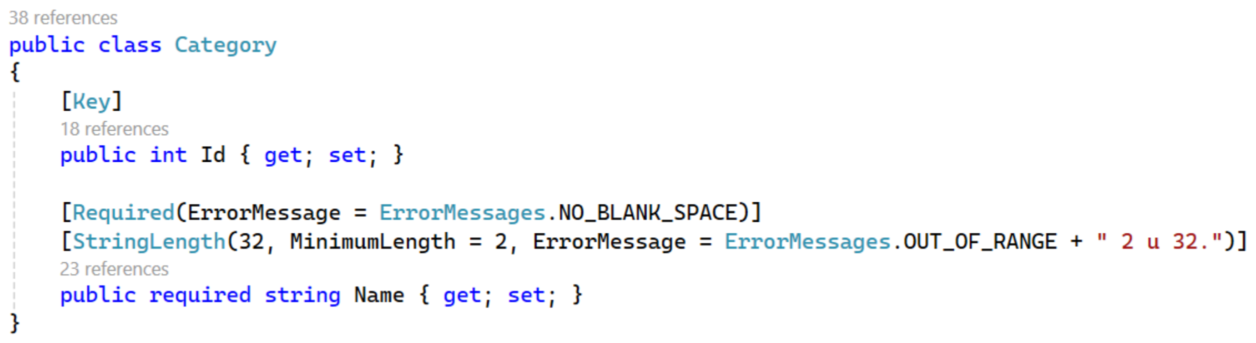
```
  ▷ ApplicationUser.cs
  ▷ Brand.cs
  ▷ Category.cs
  ▷ ErrorViewModel.cs
  ▷ Phone.cs
  ▷ Product.cs
  ▷ SavedItems.cs
```

ANNEX 3: The architecture of the 'Brands' model



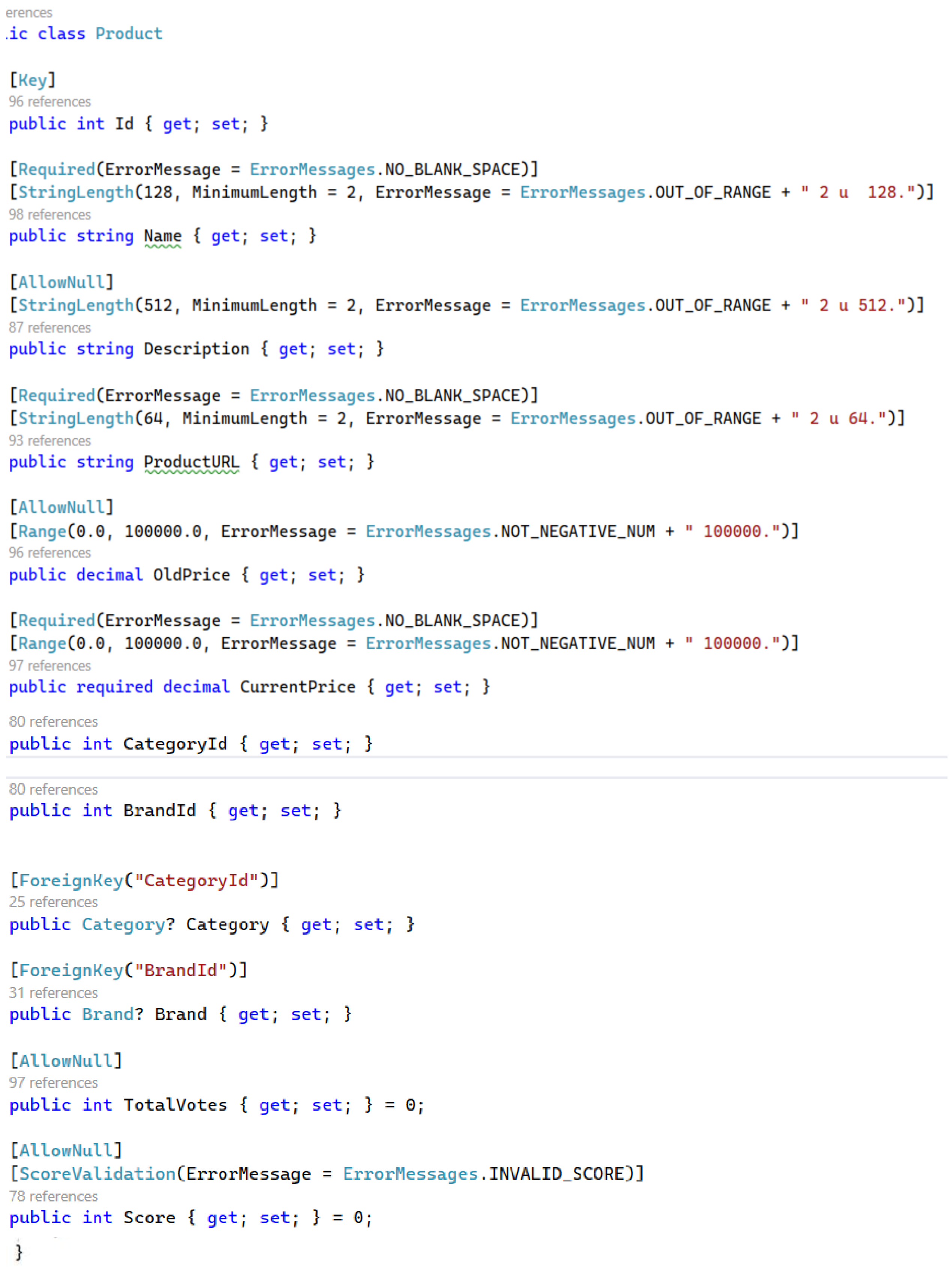
```
43 references
public class Brand
{
    [Key]
    21 references
    public int Id { get; set; }

    [Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
    [StringLength(32, MinimumLength = 2, ErrorMessage = ErrorMessages.OUT_OF_RANGE + " 2 u 32.")]
    35 references
    public required string Name { get; set; }

    [Required(ErrorMessage = ErrorMessages.NO_BLANK_SPACE)]
    21 references
    public required string PathD { get; set; }


}
```

ANNEX 4: The architecture of the 'Category' model



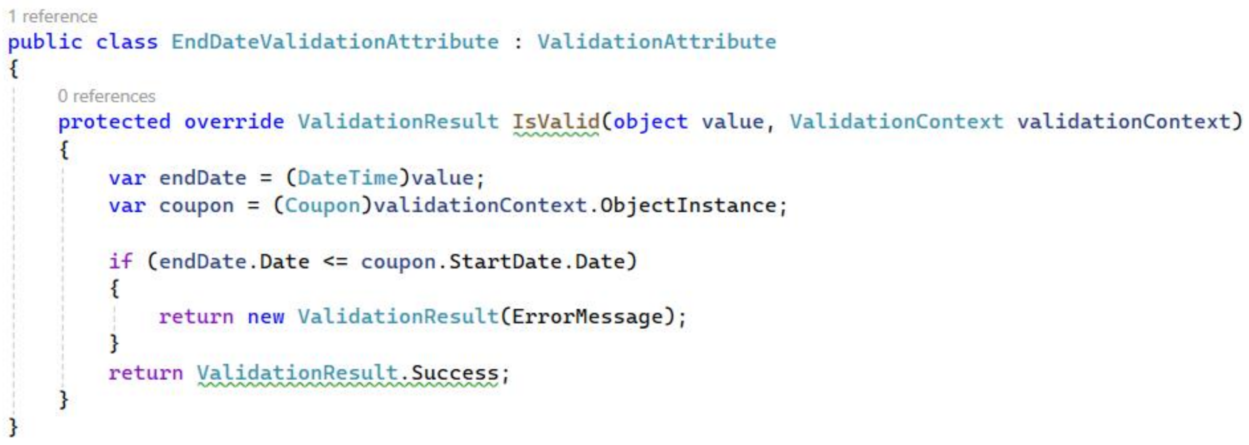
```
38 references
public class Category
{
    [Key]
    18 references
    public int Id { get; set; }

    [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; }
}
```

ANNEX 5: The architecture of the 'Product' model



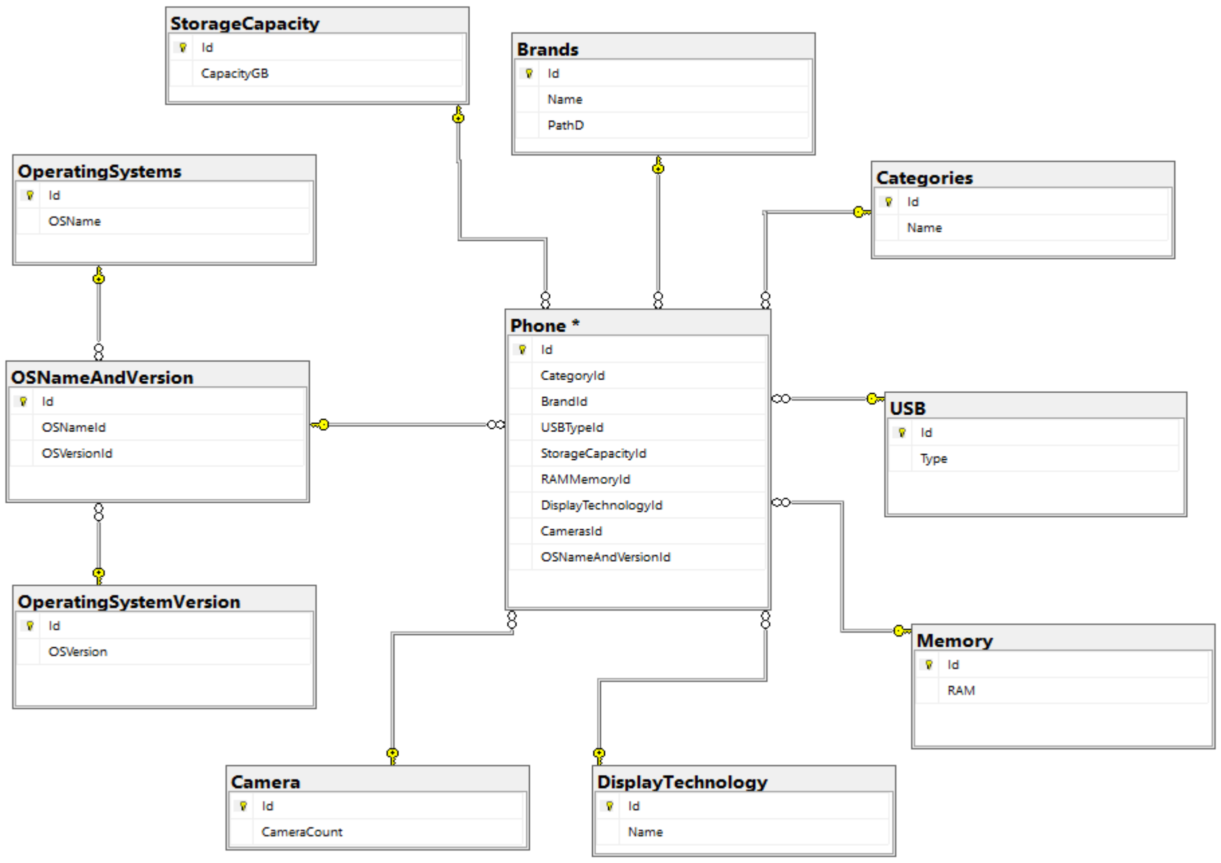
```
erences
.ic class Product

 [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;

 }
```

ANNEX 6: Implementation of the Data Validation Attribute

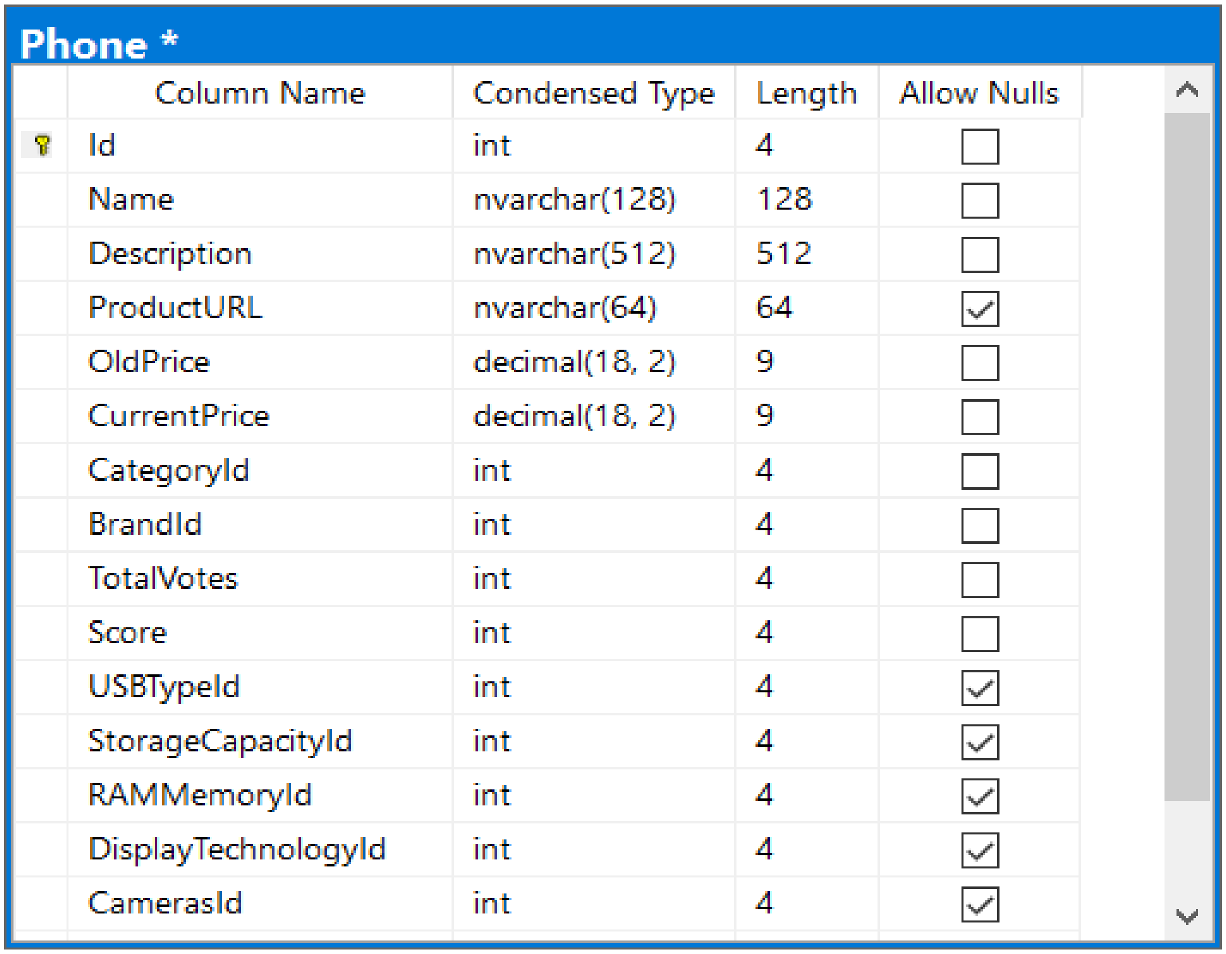


```
1 reference
public class EndDateValidationAttribute : ValidationAttribute
{
    0 references
    protected override ValidationResult IsValid(object value, ValidationContext validationContext)
    {
        var endDate = (DateTime)value;
        var coupon = (Coupon)validationContext.ObjectInstance;

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

ANNEX 7: Relationships between the Product table via foreign keys



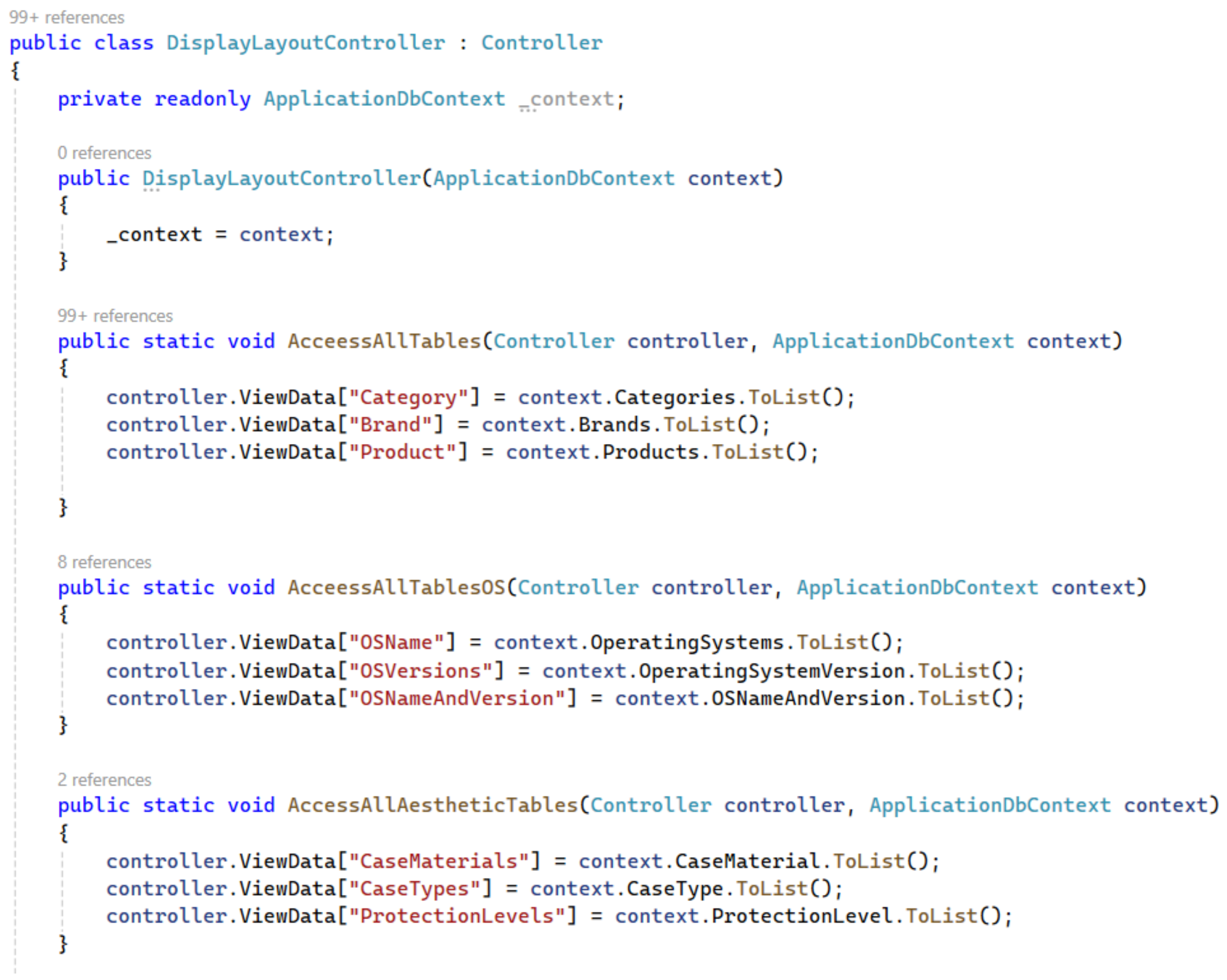
APPENDIX 8: Detailed Description of the Phone Table



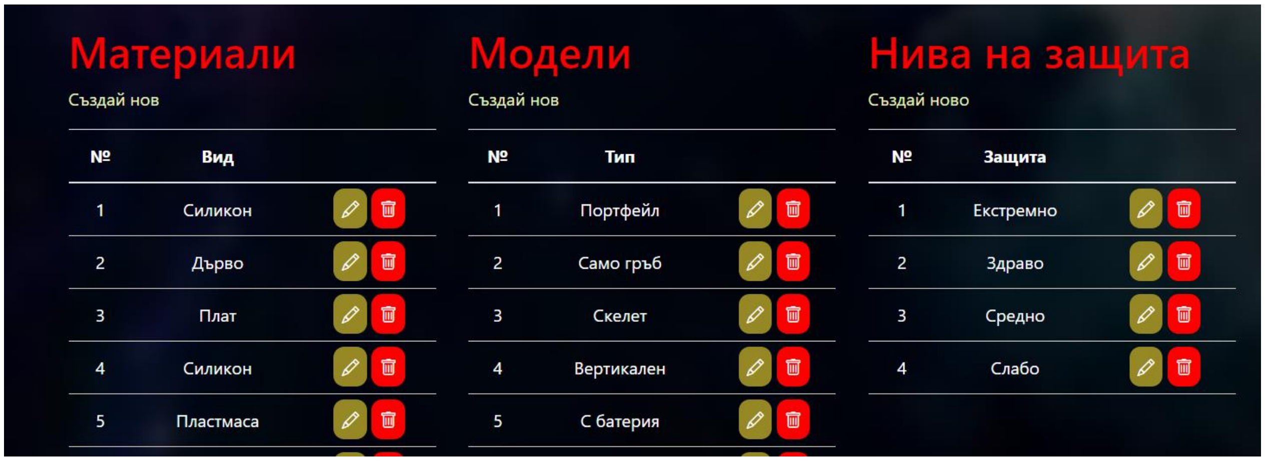
Phone *

	Column Name	Condensed Type	Length	Allow Nulls
🔑	Id	int	4	☐
	Name	nvarchar(128)	128	☐
	Description	nvarchar(512)	512	☐
	ProductURL	nvarchar(64)	64	✓
	OldPrice	decimal(18, 2)	9	☐
	CurrentPrice	decimal(18, 2)	9	☐
	CategoryId	int	4	☐
	BrandId	int	4	☐
	TotalVotes	int	4	☐
	Score	int	4	☐
	USBTypeId	int	4	✓
	StorageCapacityId	int	4	✓
	RAMMemoryId	int	4	✓
	DisplayTechnologyId	int	4	✓
	CamerasId	int	4	✓

APPENDIX 9: Part of the DisplayLayoutController methods


```
99+ references
public class DisplayLayoutController : Controller
{
    private readonly ApplicationDbContext _context;

    0 references
    public DisplayLayoutController(ApplicationDbContext context)
    {
        _context = context;
    }

    99+ references
    public static void AcceessAllTables(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 AcceessAllTablesOS(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();
    }
```

ANNEX 10: Display data from multiple tables at once



Материали

Създай нов

№	Вид	
1	Силикон	
2	Дърво	
3	Плат	
4	Силикон	
5	Пластмаса	

Модели

Създай нов

№	Тип	
1	Портфейл	
2	Само гръб	
3	Скелет	
4	Вертикален	
5	С батерия	

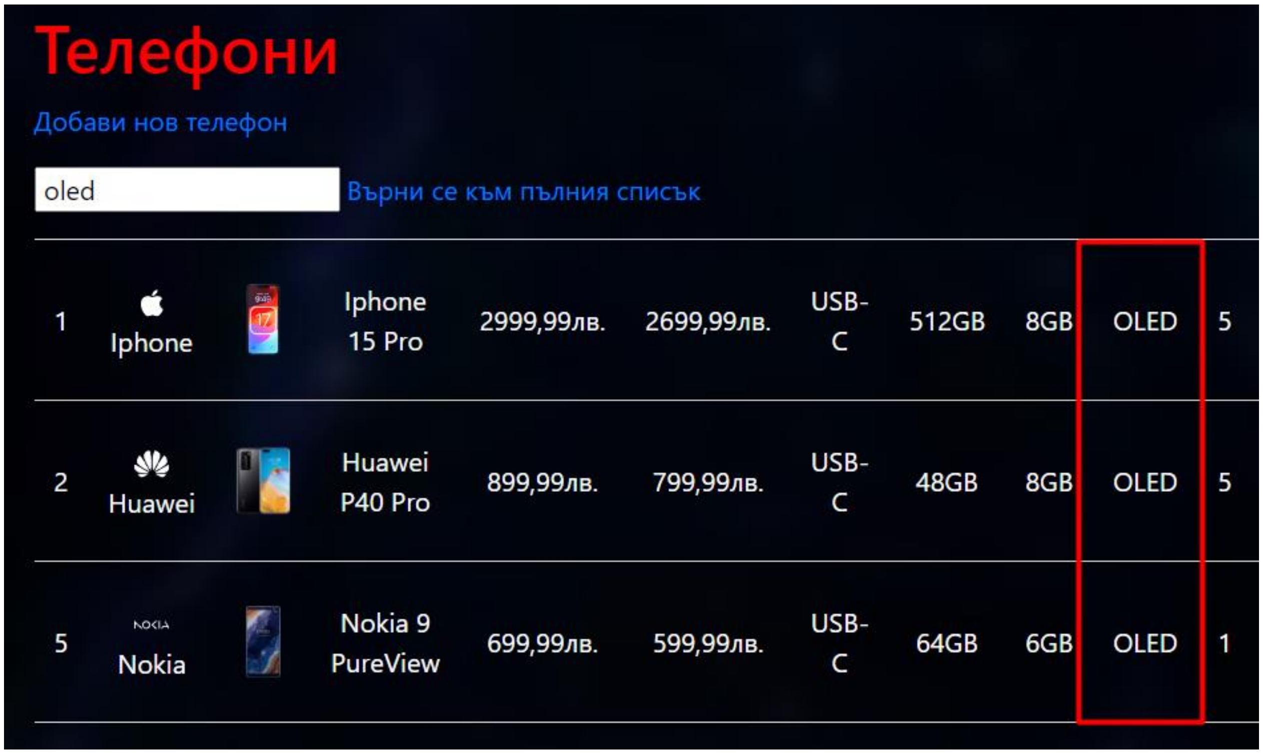
Нива на защита

Създай ново

№	Защита	
1	Екстремно	
2	Здраво	
3	Средно	
4	Слабо	

APPENDIX 11: Using the Phone Search Engine

The example shown shows how even in the absence of compatibility between uppercase and lowercase letters, the search engine returns the correct desired result, No matter which column the string we used as a filter is located from.



Телефони

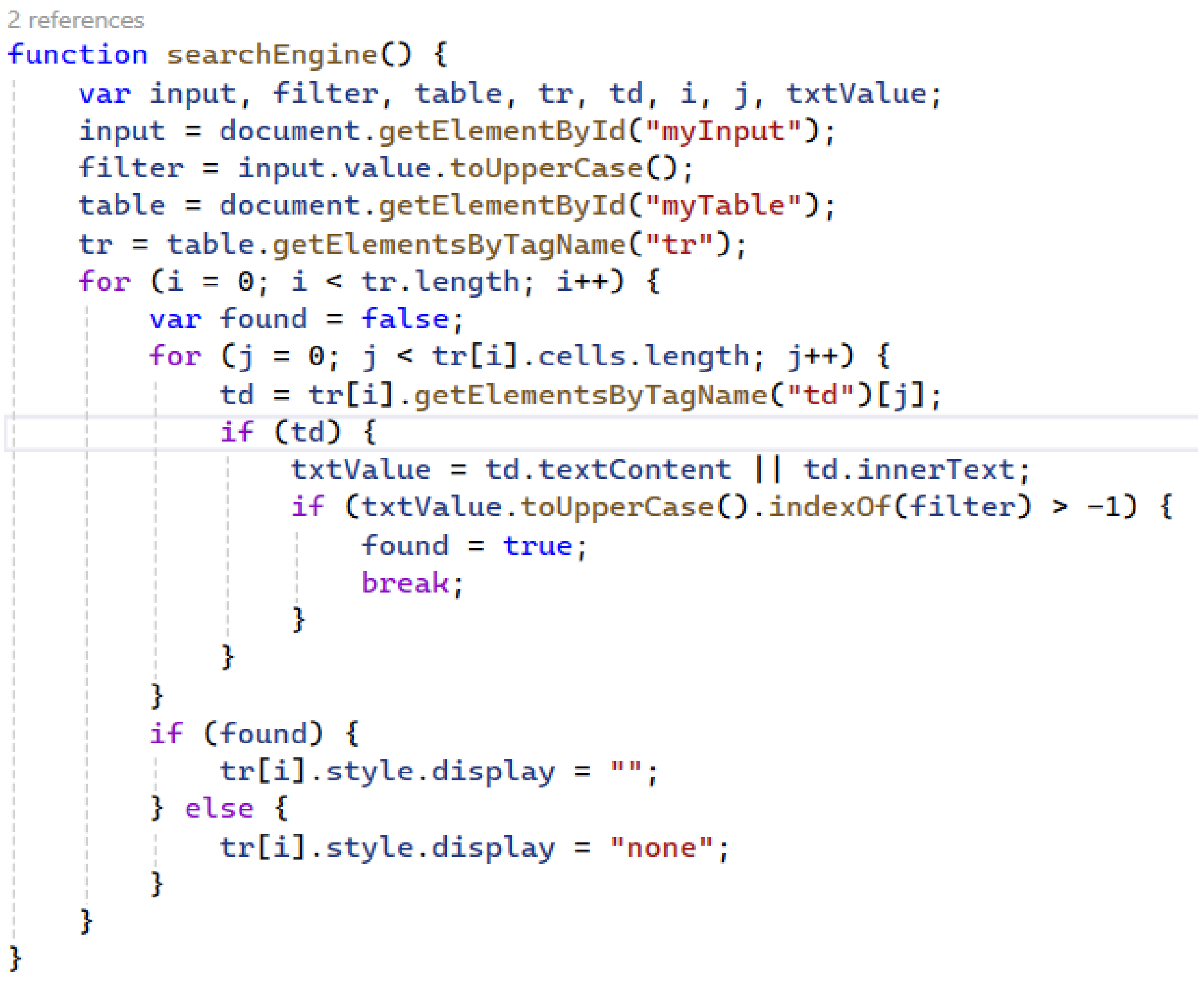
Добави нов телефон

oled

Върни се към пълния списък

1	Iphone		Iphone 15 Pro	2999,99лв.	2699,99лв.	USB-C	512GB	8GB	OLED	5
2	Huawei		Huawei P40 Pro	899,99лв.	799,99лв.	USB-C	48GB	8GB	OLED	5
5	Nokia		Nokia 9 PureView	699,99лв.	599,99лв.	USB-C	64GB	6GB	OLED	1

ANNEX 12: Implementing the search engine



```
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. References

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 (Search Engine), https://www.perplexity.ai/

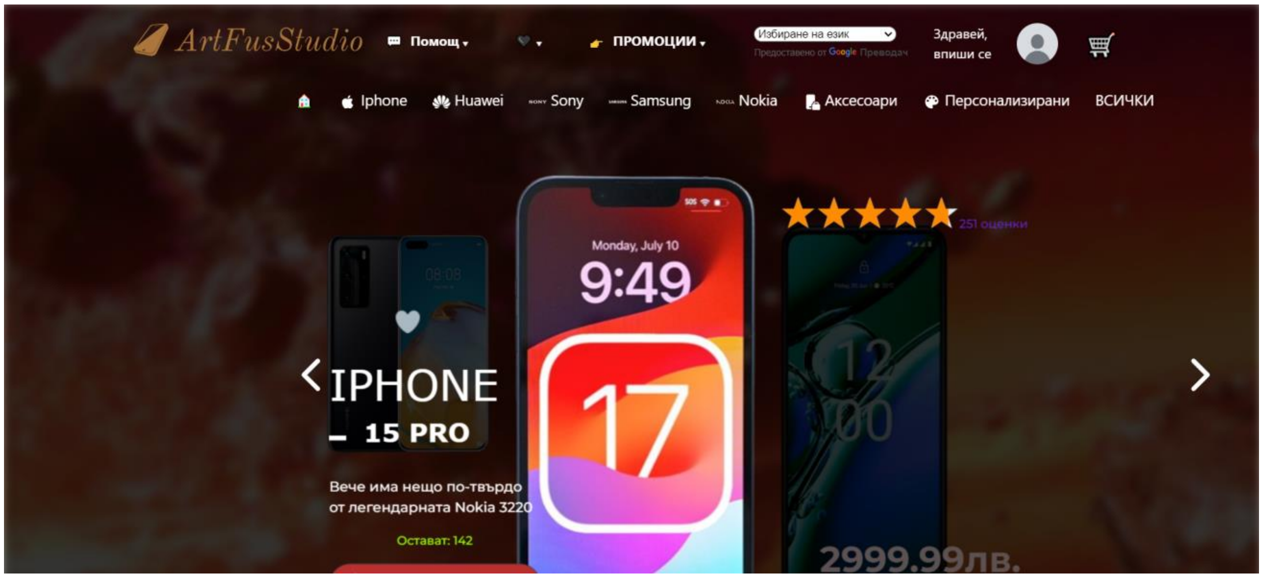
W3schools, https://www.w3schools.com/howto/

Startbootstrap, https://startbootstrap.com/

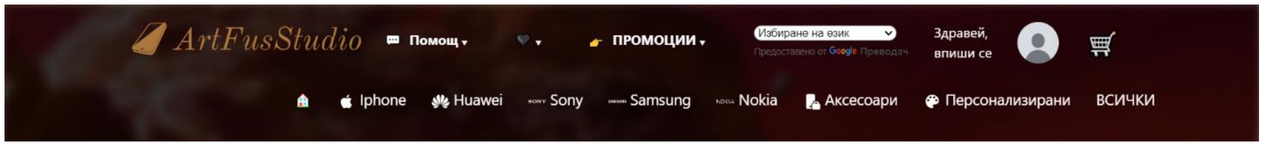
SVG icons, https://iconify.design/

User Guide

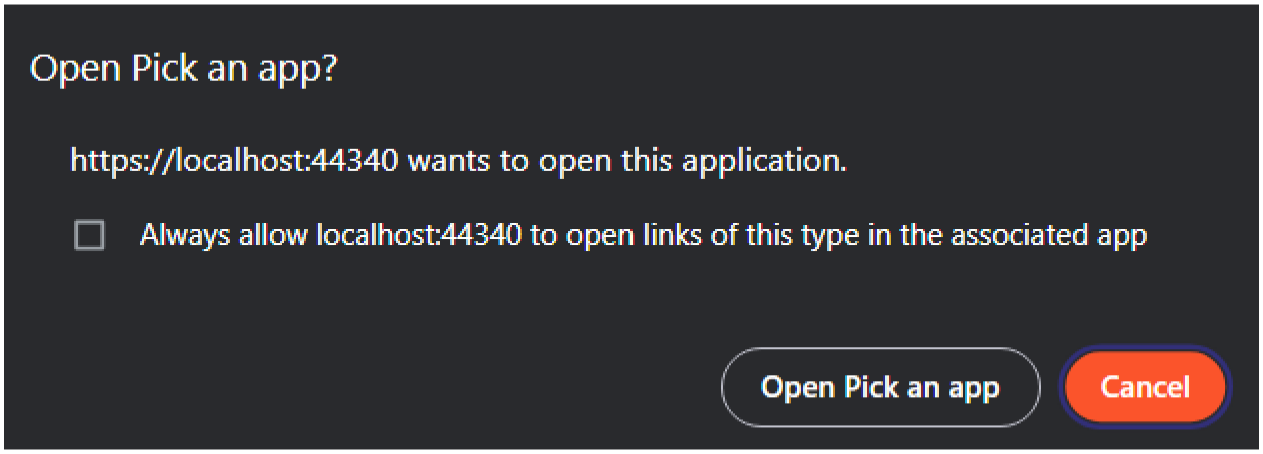
The website is interactive and has quite a lot of interaction between the user and the interface. Where possible, the option for the user to click on a corresponding button has been removed and replaced with drop-down lists when hovering over it. Separately, in order to shorten the time that the user will lose in the mouse to cover or press buttons, events have been added. The first thing that every user of the site will see is the homepage, so it should grab their attention. Until he logs into his account, a significant part of the functionalities remain hidden from him, and when he tries to use them, they redirect him to the authentication page. The first frame that users see after loading each code is:



The common view for each page is:



By clicking on the logo, which is located in the upper left corner, the user is taken back to the homepage. When the mouse arrow is placed on "Help", a drop-down list of links to the merchant behind the site (Help) provides a phone number and when you click on it, the following quality menu opens:

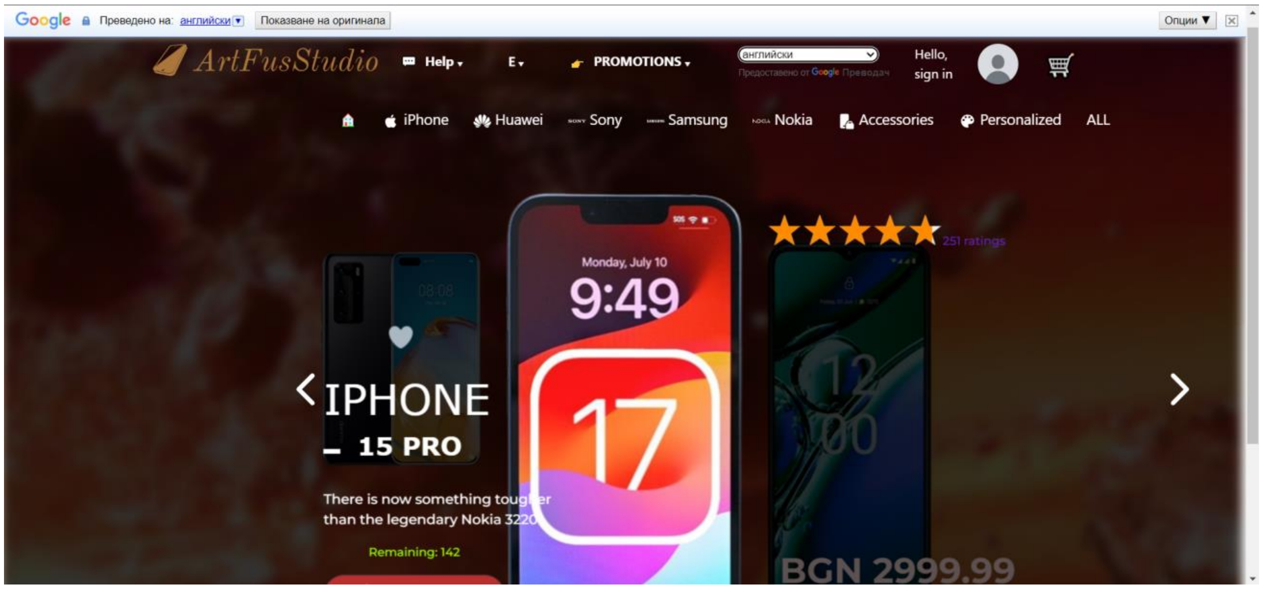

Open Pick an app?

https://localhost:44340 wants to open this application.

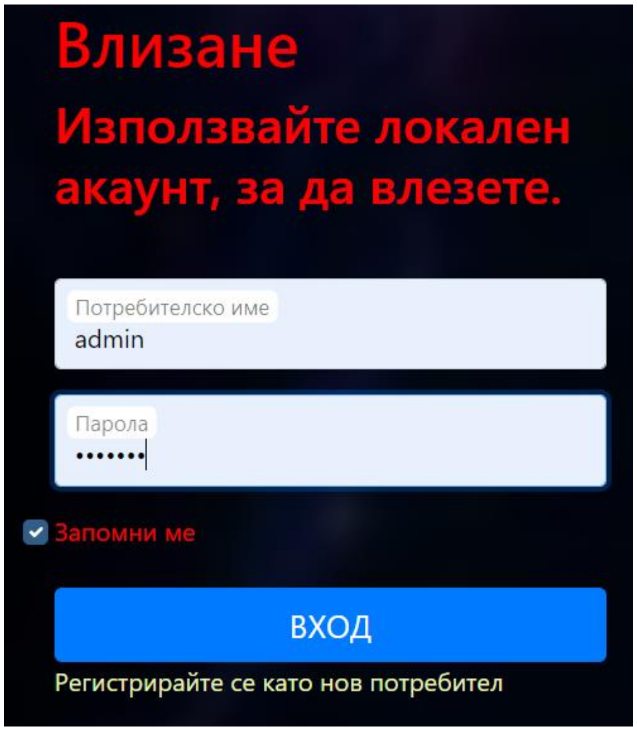
☐ Always allow localhost:44340 to open links of this type in the associated app

Open Pick an app | Cancel

The other information that the drop-down list provides is the person's email address, as well as the ability to chat). The links below don't lead anywhere because no real email is provided. The next drop-down list is the one with a heart icon, which should show the user the goods saved for later, but only when they are logged in to their user account. The next drop-down list is the one that shows the promotions. To say that an item is promotional, it must have its old price higher than its current price. It shows users only a few of the promotional items to avoid an overly large list. The next thing is key for any business that wants to have customers from all over the world – it's the ability for the customer to translate the website content into their native language. This is what the site would look like if English was chosen as such:



The next thing in the queue is the link to the user's authentication form and next to it is the correct photo that he would choose when creating his account. It looks like this:


Влизане

Използвайте локален акаунт, за да влезете.

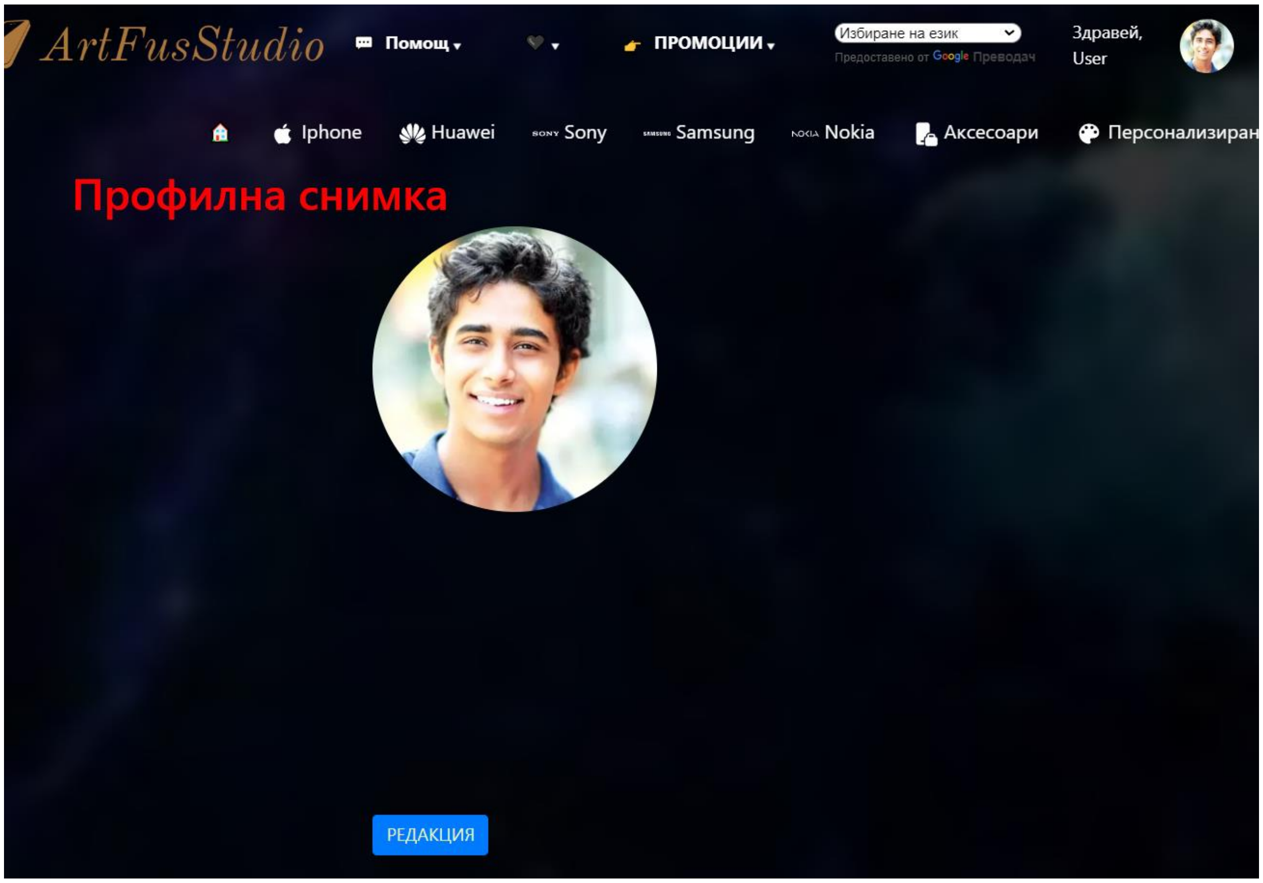
Потребителско име: admin

Парола: •••••••

✓ Запомни ме

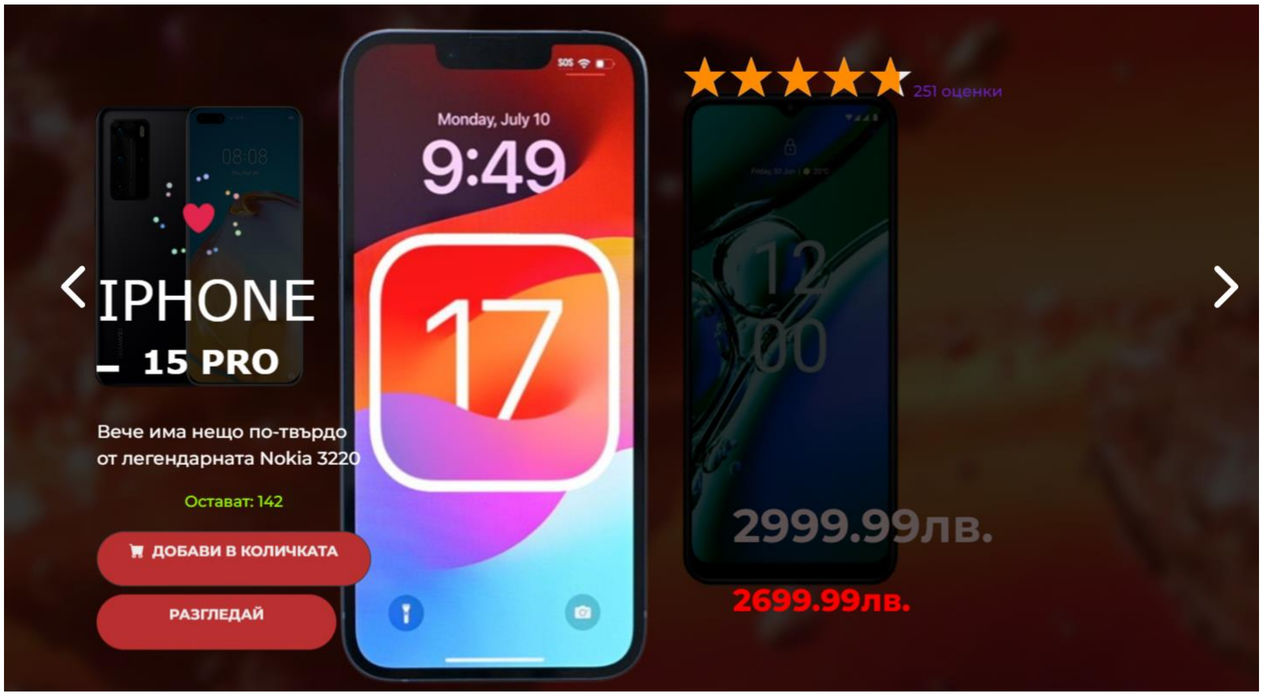
If the user does not have an account, they can register as a new user by clicking on the link with the text "Register as a new user".

When the user clicks on this picture, if the user is not logged in, they are taken to the login form, but if they are logged in, they are taken to a page that prompts them to log out of their account. Once he has authenticated, he can change his profile picture by uploading it from his local storage. To do this, he needs to click on the photo before the cart, then on the create button and then on the button for attaching an image. It must be in the most used extensions (PNG, JPG, JPEG), but there are others. After uploading his photo, he can change it by clicking on it and then on the "Retaltoray" button.

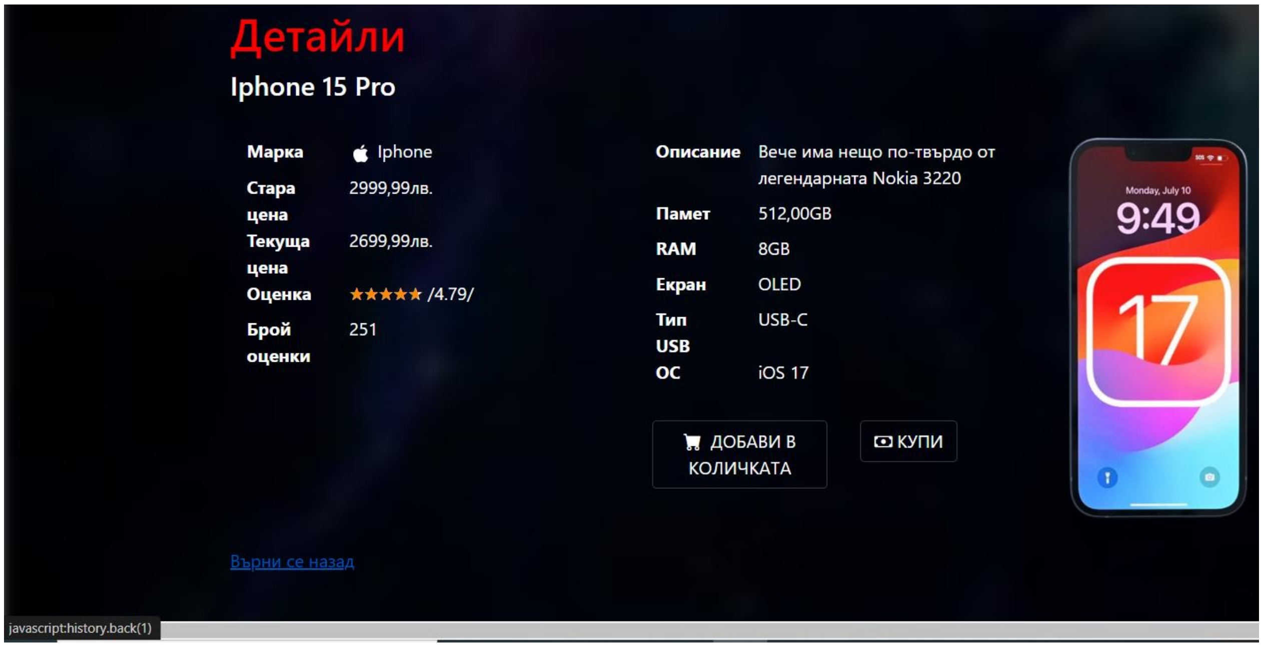


After attaching his new photo, he clicks on the save changes button and it is added and displayed next to his name. The photos are saved in the folder in the main directory, and their name is the user ID.

What is the essence of the homepage are the three phones with the highest rating. The information that is presented about them is their names, versions, rating, current price, old price if the item is promotional, description, availability if there are less than 150 items, the buttons for saving in cart and additional details, as well as the heart to save in "Favorites". Changing phones is carried out both through the user interface of the site, by pressing the arrows, and through events - when the user presses the left or right arrow on the keyboard, the phone in the middle is shifted in the appropriate direction and the one from the opposite side is in its place. Along with the phone, the background of the page also changes. It looks like this with the effect when the heart is pressed:



The "DISTRIBUTE" button takes us to a page with detailed information about this phone, which offers the option to add to cart or to purchase directly. The page also includes a link to go back to the previous open page using the browser history. It looks like this:

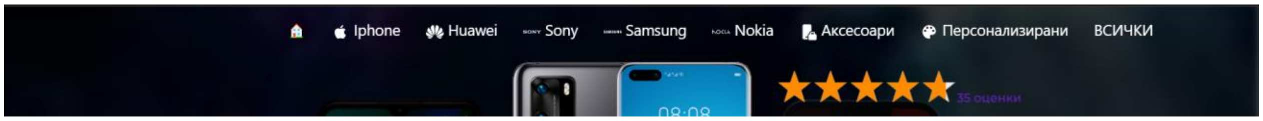


Детайли

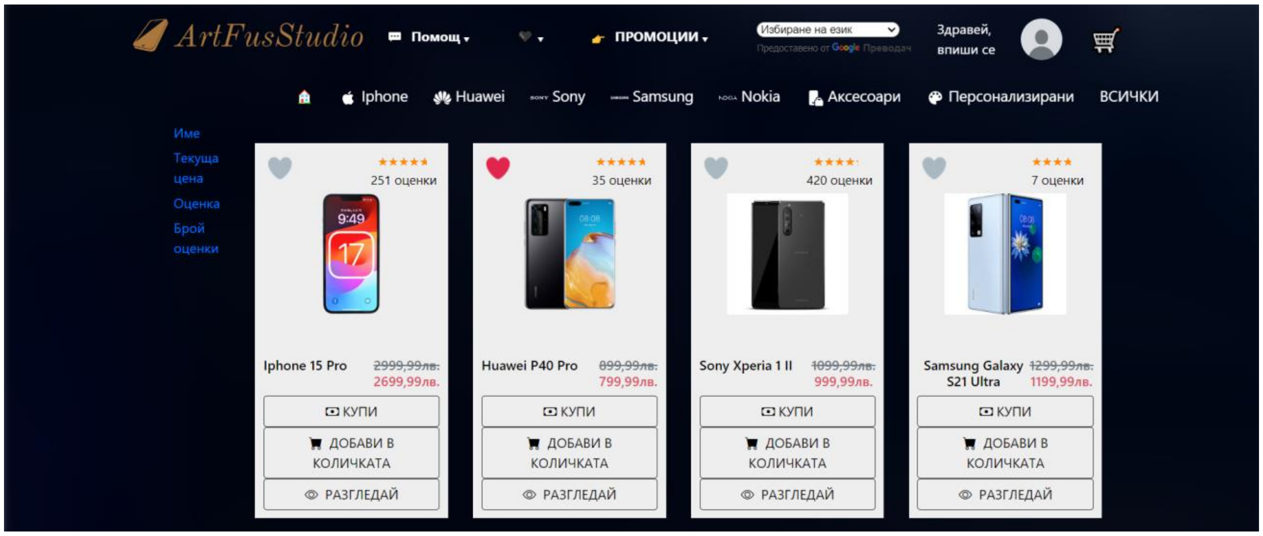
Iphone 15 Pro

Марка	Iphone	**Описание**	Вече има нещо по-твърдо от легендарната Nokia 3220
Стара цена	2999,99лв.	**Памет**	512,00GB
Текуща цена	2699,99лв.	**RAM**	8GB
Оценка	★★★★★ /4.79/	**Екран**	OLED
Брой оценки	251	**Тип USB**	USB-C
		ОС	iOS 17

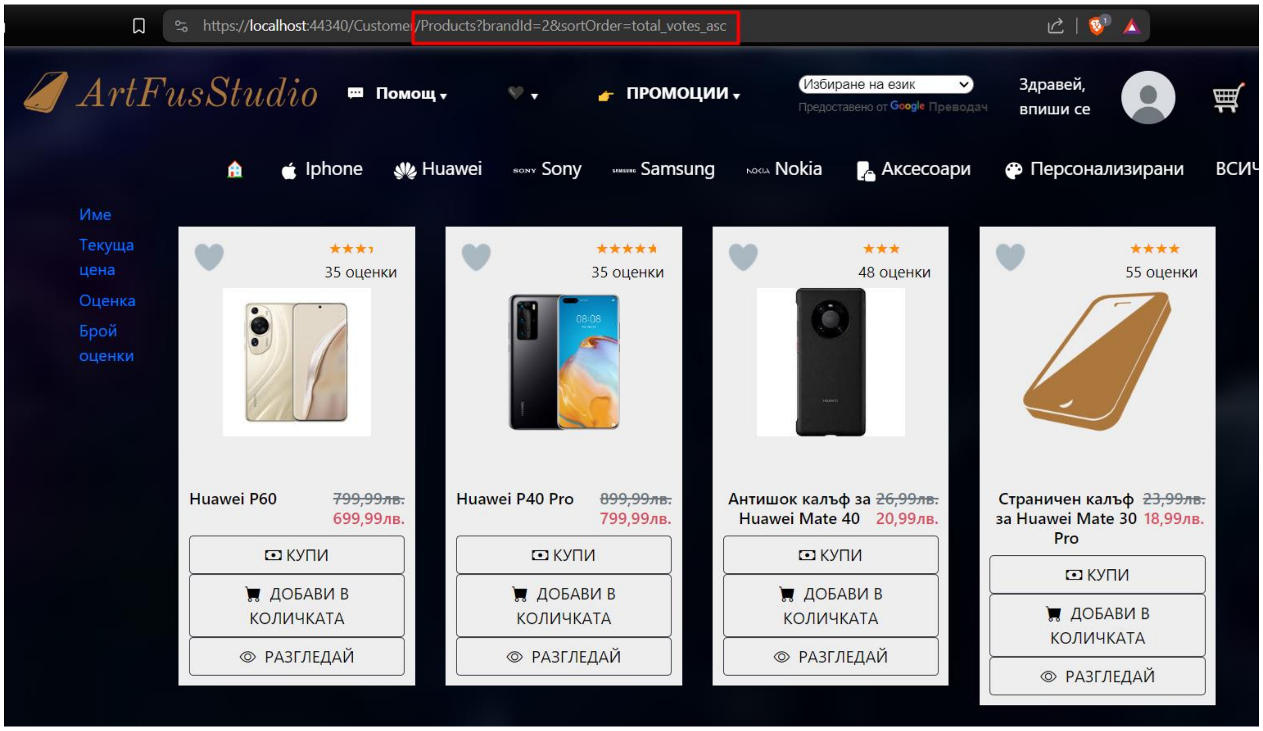
The links and information shown here are subject to the same checks as on the previous page. The additional features of the header, no longer mentioned at the beginning, are those related to the links from the second bar. The first thing that makes an impression is that this bar is attached to the very top of the screen and stays there no matter how far down the page the user has gone. Its main components are:



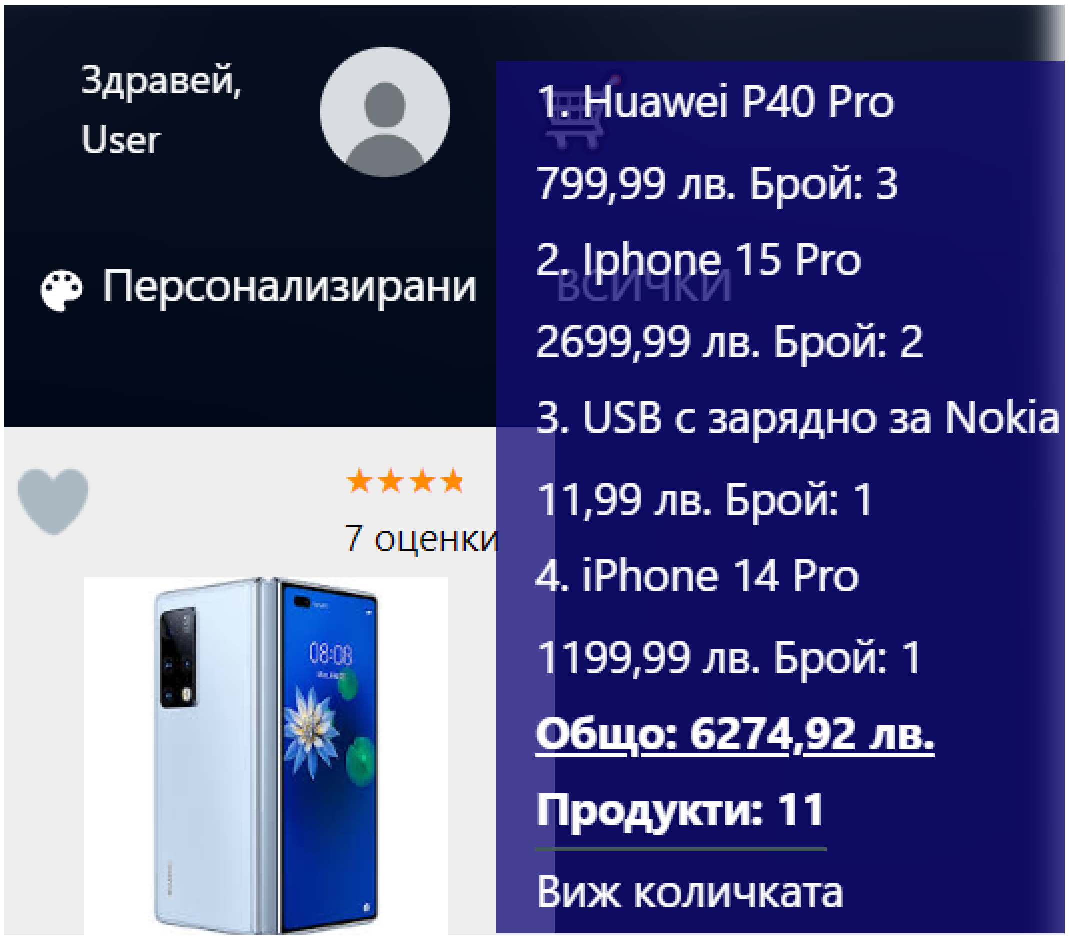
Link to your homepage Cottage icon, options to filter content and display a specific brand and display all items offered. It looks like this:



In addition to the content filtering capabilities, which are also used to show consumers what are the brands of the items they offer, it also has the ability to maintain the items according to a certain criterion - alphabetically, by price, by rating and by the number of ratings. By pressing the sorting buttons for the first time, the items are arranged in ascending order, and by pressing the same button again - in descending order. The site offers the ability to combine between filtering and sorting.

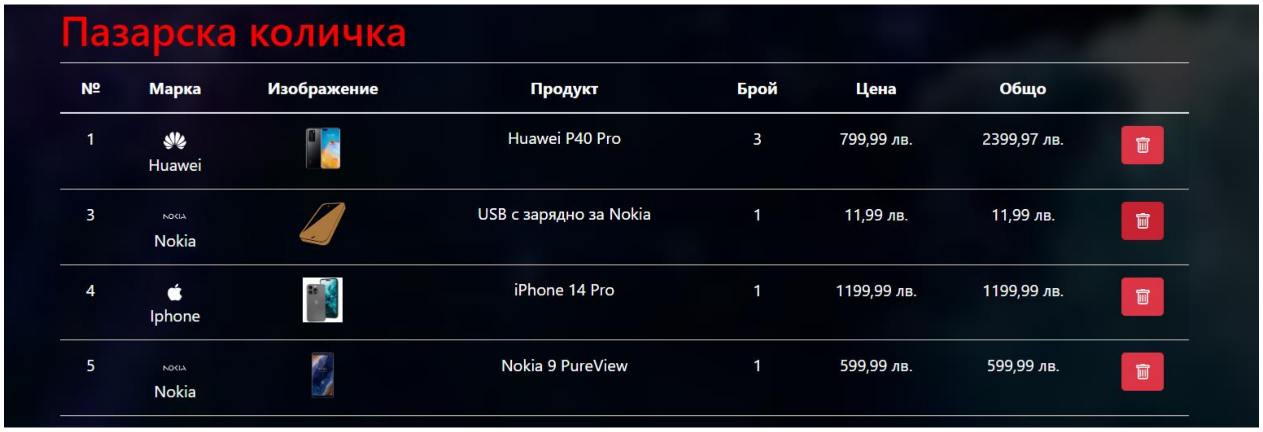


From the attached photo, you can see that only the items with the Huawei brand are shown, sorted by the number of ratings in ascending order. This happens in the form of a GET query to the Product table to retrieve all the items that meet the relevant requirements from the user. A lot of things that are shown in this picture is what products that don't have a photo look like. The logo of the site is displayed as an alternative one. Once the user has entered their username and password, they are now logged into their account and can perform more actions. This is what his shopping cart would look like after a few items are added. It shows no more than 4 items:



1. Huawei P40 Pro
799,99 лв. Брой: 3
2. Iphone 15 Pro
2699,99 лв. Брой: 2
3. USB с зарядно за Nokia
11,99 лв. Брой: 1
4. iPhone 14 Pro
1199,99 лв. Брой: 1
Общо: 6274,92 лв.
Продукти: 11

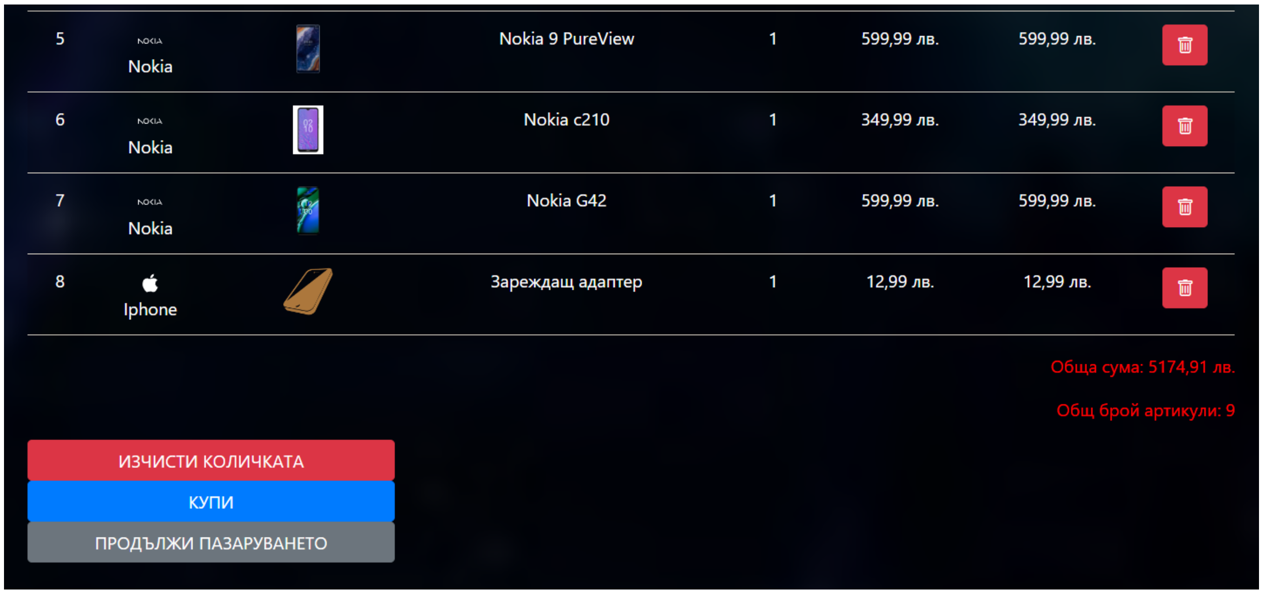
When the user clicks on the name of one of the selected items, he goes to the page with the detailed information about this item. From the photo shown, you can see that basic information is offered to the customer about how many items he has in his cart and how much is their total price. Because if the drop-down list shows all the items that have been added to the cart, it may become too long, so now you can notice from the photo a discrepancy between the total number of products and the sum of the quantities shown. When the "View cart" button is clicked, the page with the shopping cart of the respective user opens, which shows the necessary information about the products in it and provides the opportunity to remove a specific item:



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	Iphone		iPhone 14 Pro	1	1199,99 лв.	1199,99 лв.	
5	Nokia		Nokia 9 PureView	1	599,99 лв.	599,99 лв.	

In addition, it offers information about how much all items cost in total, how many they are, a button to empty the cart completely, purchase these items, as well as go back and continue shopping.



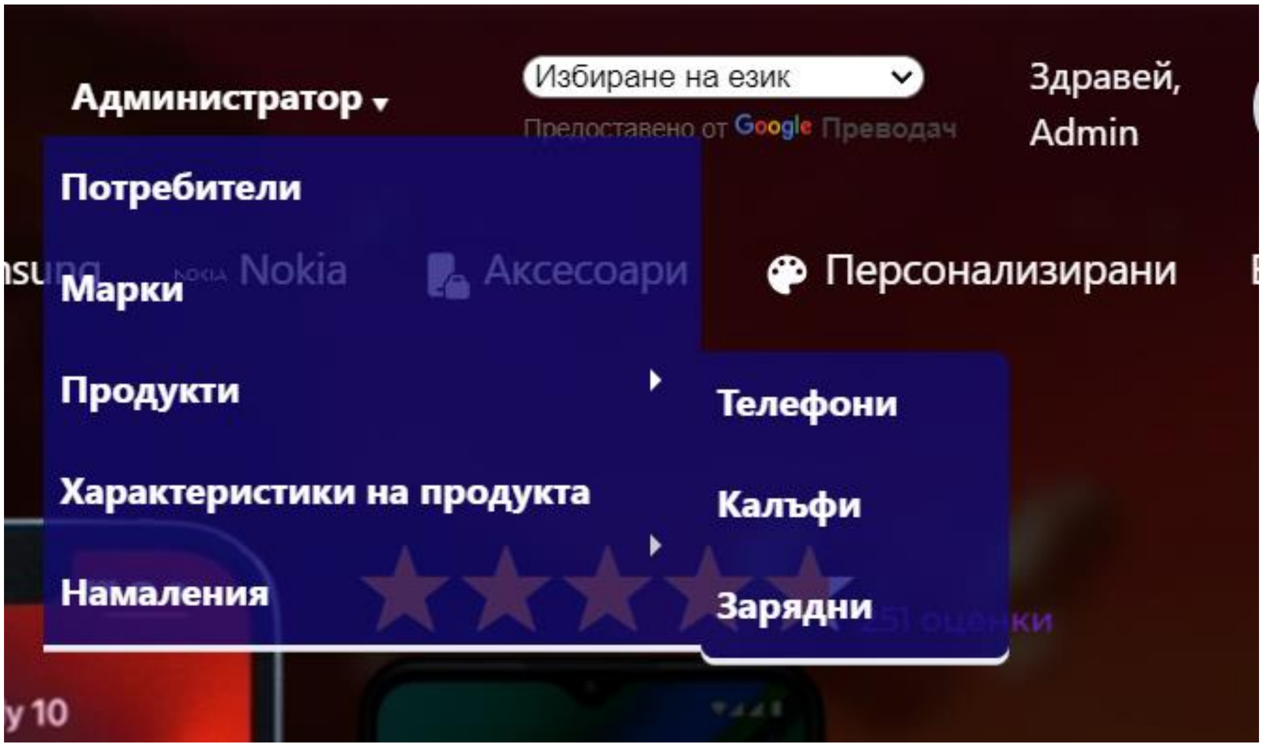
5	Nokia		Nokia 9 PureView	1	599,99 лв.	599,99 лв.	
6	Nokia		Nokia c210	1	349,99 лв.	349,99 лв.	
7	Nokia		Nokia G42	1	599,99 лв.	599,99 лв.	
8	Iphone		Зареждащ адаптер	1	12,99 лв.	12,99 лв.	

Обща сума: 5174,91 лв.

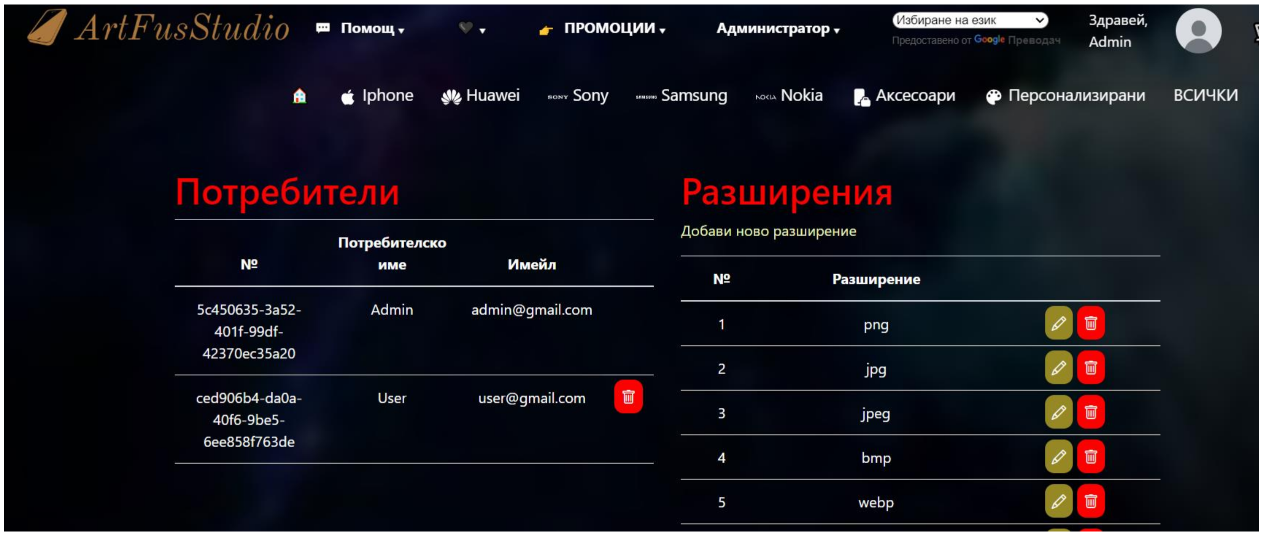
Общ брой артикули: 9

When the user clicks on the button to clear the cart or to make a purchase, the business logic from the control is performed and he is redirected again to the list of all products offered by the site.

This summarizes what the average consumer can do so far. In contrast, the admin account has access to more pages, basically the dashboard, which is accessed from the shared view, just before the translator.



The first link goes to the page from which the administrator has the right to delete registered users of the site, with the exception of other administrators.



Потребители

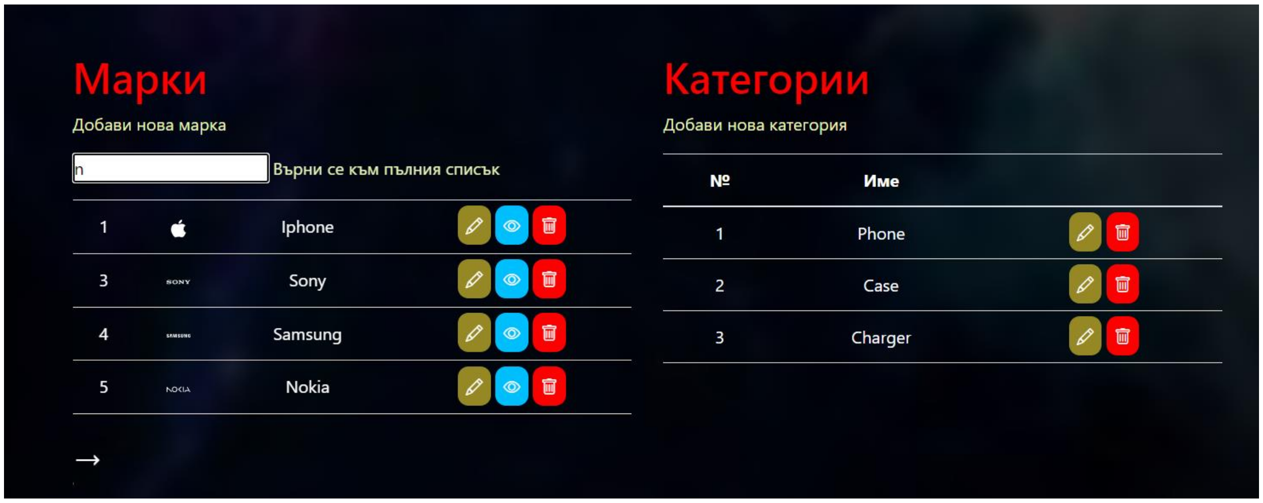
№	Потребителско име	Имейл	
5c450635-3a52-401f-99df-42370ec35a20	Admin	admin@gmail.com	
ced906b4-da0a-40f6-9be5-6ee858f763de	User	user@gmail.com	🗑

Разширения

Добави ново разширение

№	Разширение	
1	png	✎ 🗑
2	jpg	✎ 🗑
3	jpeg	✎ 🗑
4	bmp	✎ 🗑
5	webp	✎ 🗑

From the link named "Brands", the administrator is taken to a page where he can add new categories if new products will be offered on the site, as well as the corresponding brands of these categories. In this view, the search engine and the division of data into pages meet. They work with events, and for the search engine it is not necessary to press a button to confirm the input directly to offer the results, if any, and to divide the data into pages it is only necessary to press the arrow to the right of the keyboard to go to the next page, as well as the left arrow to go back. When the page is number 1, the lawyer on the left is removed to prevent errors. This page looks like this:



Марки

Добави нова марка

n Върни се към пълния списък

1		Iphone
3	SONY	Sony
4	SAMSUNG	Samsung
5	NOKIA	Nokia

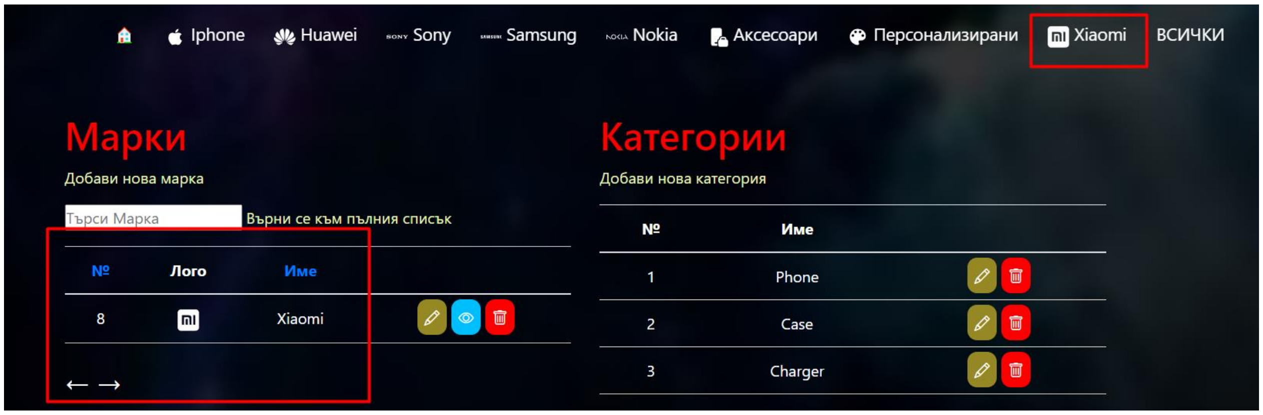
→

Категории

Добави нова категория

№	Име
1	Phone
2	Case
3	Charger

The special thing when the user adds a new brand is that it will be displayed in every view whose controller has the Mark Table Access Template Controller method, which is desirable to be present everywhere. The change would look like this:



Марки

Добави нова марка

Търси Марка Върни се към пълния списък

№	Лого	Име
8		Xiaomi

← →

Категории

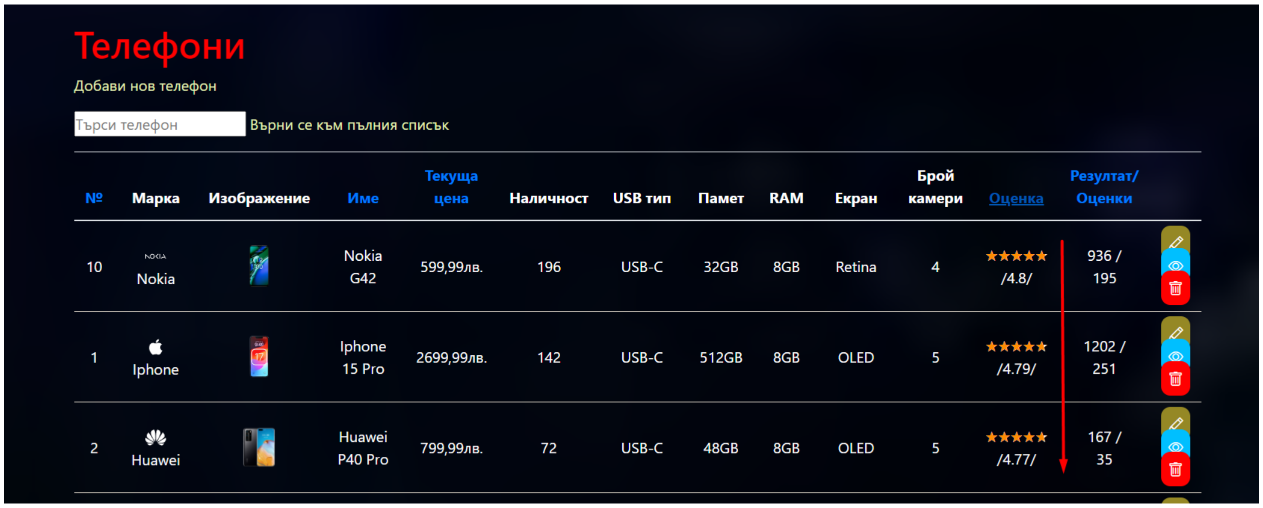
Добави нова категория

№	Име
1	Phone
2	Case
3	Charger

This is the reason why space is left and the entire bar is not filled - because this is a foundation that will be used to upgrade the project, no serious code rework will be required to be able to add new functionalities.

When the mouse cursor is hovered over the link named "Product Features", a new drop-down list of three categories opens. They combine several tables that have similar features - in "Aesthetic" are shown the characteristics of a phone case - the fabric from which it is made, the type of case and the level of protection it offers; "Software" contains data about the phone's operating system - the most commonly used names and versions of the operating system, as well as all possible variations, are recorded there; In third place are recorded the most commonly used hardware specifications of a phone - storage memory, RAM, USB type, screen technology, as well as the number of cameras it has. These tables have the ability to add a new value, edit and delete an existing one.

In the "Products" group are the products that are offered on the site, divided by their respective categories – phones, cases and chargers. Because all three categories inherit everything from the base "Product" model, the display of the data, the sorting and filtering criteria, and the manipulations of the tables are identical for them and for each new addition of a product category. The sorting that is offered is the same as the display of all products together. The design of the views is also identical. It looks like this, in addition to a demonstration of the ranking of phones by rating in descending order:



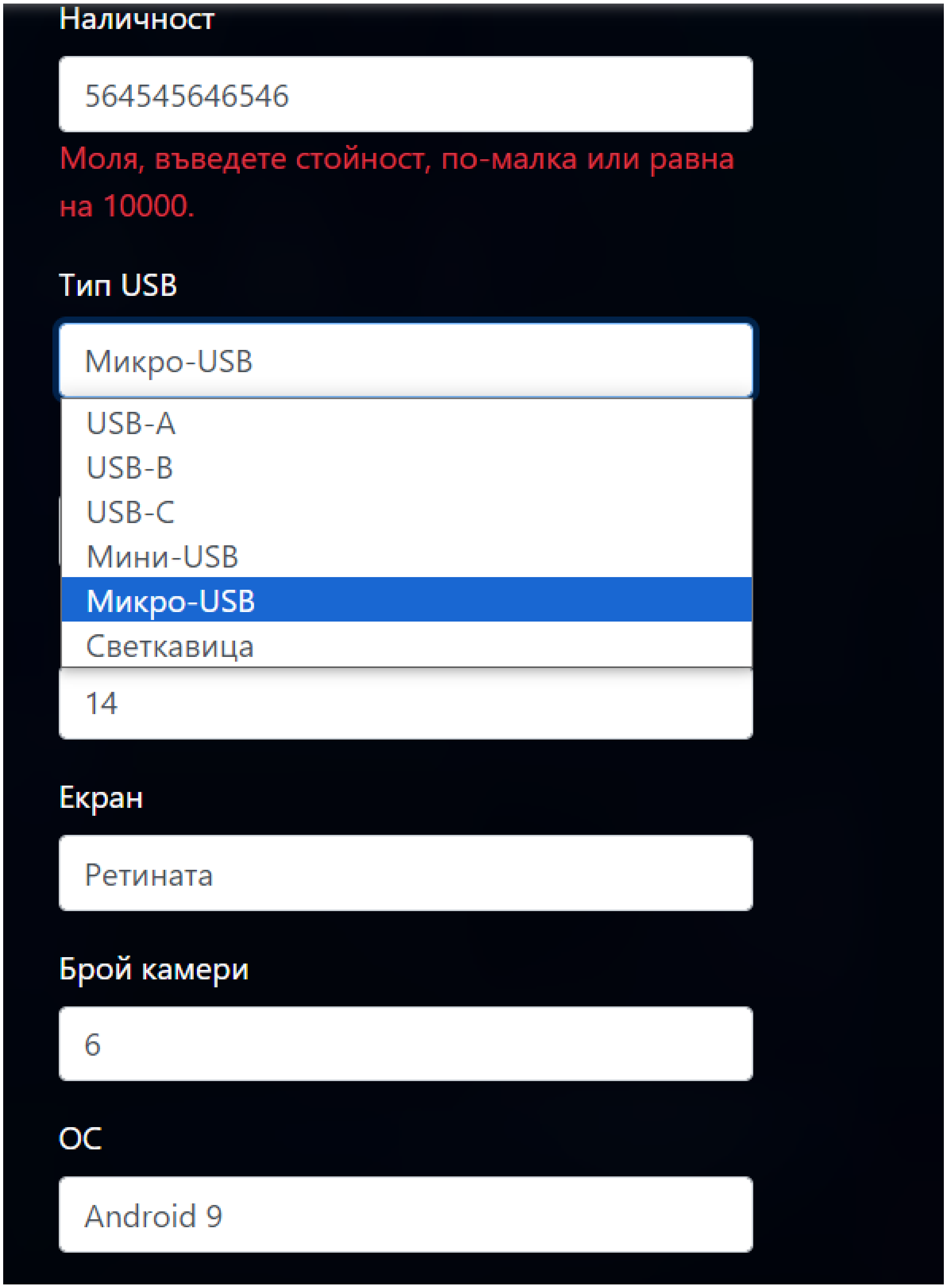
Телефони

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	Iphone		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

In the view of creating a new product, no matter what category it belongs to, the limitations used in the models have been added, but also others, typical for HTML inputs - the data entry fields are assigned a minimum and a maximum value, which even the administrator cannot exceed. This is to prevent a case like a negative cost. The reason why the database is divided into so many tables is that when we create or edit a product, we have a set of the most commonly used values, and so the values follow a single record format:



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

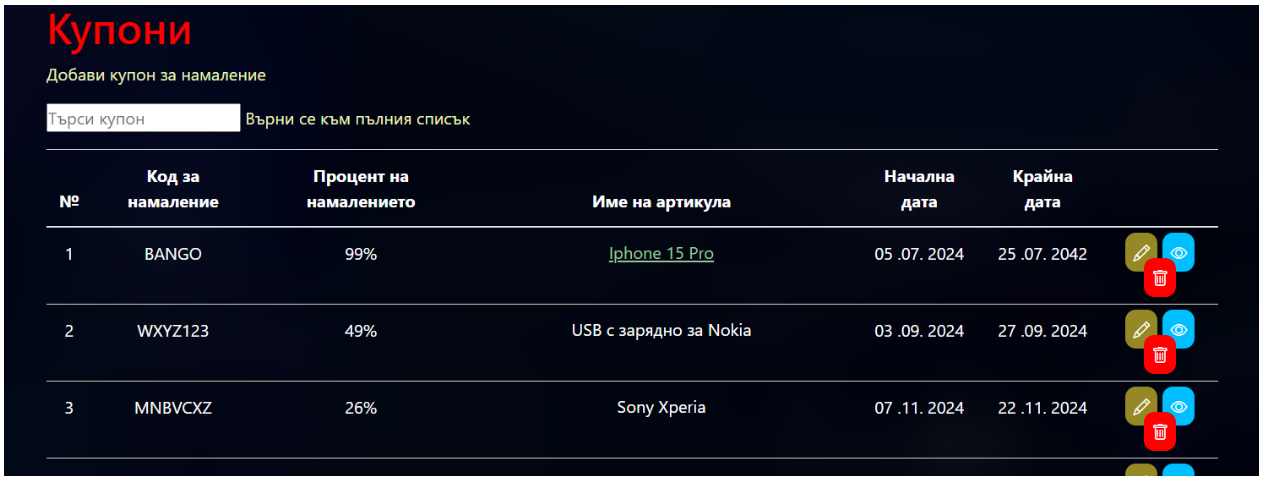
6

ОС

Android 9

The same is displayed when editing the product. In the detail and delete views, the data is displayed in the same style as the details in the user area shown at the beginning of the user guide.

As an additional functionality, discount coupons have been added, which lower the price of the item for which they are intended, with the specified number of percentages and have expiration dates. The moderator panel for these coupons looks like this:



Купони

Добави купон за намаление

Търси купон | 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. 2042
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

When the administrator clicks on the name of the item, it is redirected to the details page for that product. When creating them, restrictions are placed to prevent a case with a discount percentage of more than 100 or where the start date is after the expiration date of the promotion. The user can only use one discount coupon per purchase.