ДИПЛОМЕН ПРОЕКТ

**Тема: РАЗРАБОТКА НА УЕБ ПРИЛОЖЕНИЕ .................................................................................................**

**Професия: код 481020 „Системен програмист“   
Специалност: код 4810201 „Системно програмиране“**

**Изготвил: Ръководител - консултант:**

**Мартин Мирославов Петров Румяна Ачанова**

**Ученик от XIIВ клас**

**Перник, 2025 г.**

СЪДЪРЖАНИЕ

[УВОД 2](#_Toc178688421)

[1.ПЪРВА ГЛАВА – ПРОУЧВАНЕ 5](#_Toc178688422)

[1.1. Проучване на пазара 5](#_Toc178688423)

[1.2. Преглед на технологии и езици за програмиране 5](#_Toc178688424)

[1.3. Избор на технологии и езици за програмиране 6](#_Toc178688425)

[2.ВТОРА ГЛАВА – ПРОЕКТИРАНЕ 7](#_Toc178688426)

[2.1. Описание и анализ на изискванията 7](#_Toc178688427)

[2.1.1.Описание и анализ на функционалните изисквания 7](#_Toc178688428)

[2.1.2. Потребителски истории (user stories) 8](#_Toc178688429)

[2.1.3. UML диаграми за случаи на употреба 9](#_Toc178688430)

[2.1.4. Анализ на нефункционалните изисквания 10](#_Toc178688431)

[2.2. Проектиране 12](#_Toc178688432)

[2.2.1. Проектиране на база данни 12](#_Toc178688433)

[2.2.2. Прототипи на потребителския интерфейс 13](#_Toc178688434)

[3. ТРЕТА ГЛАВА – РЕАЛИЗАЦИЯ 14](#_Toc178688435)

[3.1. Технология на създаване на уеб приложението 15](#_Toc178688436)

[3.2. Структура на базата от данни 15](#_Toc178688437)

[3.3. Реализация на уеб приложението 16](#_Toc178688438)

[3.3.1. Архитектура на приложението 16](#_Toc178688439)

[3.3.2. Контролери 16](#_Toc178688440)

[3.3.3. Сървиси 16](#_Toc178688441)

[3.3.4. Вю модели 16](#_Toc178688442)

[3.3.5. Изгледи 16](#_Toc178688443)

[3.3.6. Други технологии и функционалности 16](#_Toc178688444)

[ЗАКЛЮЧЕНИЕ 17](#_Toc178688445)

[ИЗПОЛЗВАНА ЛИТЕРАТУРА 18](#_Toc178688446)

[ПРИЛОЖЕНИЯ 19](#_Toc178688447)

**Генерирайте автоматично съдържанието!**

# УВОД

**Кратко въведение в темата -** В съвременния свят електронната търговия заема все по-важно място, като предлага удобство и бърз достъп до разнообразни продукти. Един от секторите, който се възползва от дигитализацията, е продажбата на музикални инструменти. Традиционните магазини често са ограничени от географското си местоположение и складовите си наличности, докато онлайн платформите позволяват на клиентите да избират измежду богато разнообразие от продукти, да сравняват цени и да получават детайлна информация за характеристиките на инструментите

**Актуалност на проблема -** защо изграждането на уеб приложение за ……………………………. е актуална и значима тема.

**Обект на дипломния проект -** уеб приложение за продажба на музикални инструменти.

**Предмет на дипломния проект -** проектиране и разработка на уеб приложение за продажба на музикални инструменти, базирано на ASP.Net Core MVC, на езика C#, което използва база от данни на MS SQL Server.

**Цел на дипломния проект** е да се проектира и разработи уеб приложение за музикални инструменти, в което да може да се публикува информация за …………………………… и да има функционалност за покупка на стоки и/или услуги, предлагани в сайта.

**Целевата аудитория** на уебмагазина за музикални инструменти обхваща както любители, така и професионални музиканти, които търсят качествени инструменти. Платформата е насочена както към начинаещи, които се нуждаят от достъпни продукти, така и към опитни музиканти, студия за звукозапис и преподаватели, които търсят надеждно оборудване. Освен това, уеб магазинът може да привлече музикални групи, оркестри и колекционери, за които са важни детайлните описания и автентичността на продуктите. Със своята функционалност и удобен интерфейс, платформата цели да улесни избора и покупката, като предоставя богата информация и сигурни методи на плащане.:

Приложението трябва да поддържа надеждна система за автентикация и авторизация на своите потребители, които трябва да бъдат разпределени минимум в три роли: нерегистрирани потребители (гости), регистрирани потребители (клиенти) и администратор. Администраторското меню трябва да съдържа секция „Статистически модул“. Нерегистрираните потребители да могат да се регистрират. Клиентите да могат да пускат поръчки. Всички потребители да могат да разглеждат съдържанието на сайта, да търсят продукти в търсачката и да ги филтрират.

Уеб приложението трябва да се базира на съвременни технологии, стандарти и модели, което гарантира лесна бъдеща поддръжка, ефективност и възможност за надграждане.

За постигане на целите на дипломния проект трябва да изпълним следните задачи:

* Проучване и анализ
* Проучване на конкурентните сайтове и услугите, които предлагат те.
* Проучване на известните технологии
* Избор на подходящи технологии.
* Проектиране
* Анализ на изискванията към уеб приложението.
* Създаване на потребителски истории
* Описание на случаи на употреба
* Проектиране на базата данни
* Създаване на прототипи на потребителския интерфейс на приложението
* Реализация
* Създаване на ASP.Net Core MVC проект
* Създаване на базата от данни
* Изграждане на навигационна система на сайта
* Имплементация на функционалностите на сайта

**Софтуер за изграждане на уеб приложението** - за изграждането на уеб приложението ще бъдат използвани: *Visual Studio Community 2022* и *MS SQL Server 2022*. Уеб приложението ще бъде базирано на *ASP.NET Core* технологията на Microsoft, трислойния модел, MVC модела и езика за програмиране *C#*. Ще бъдат използвани и други помощни софтуерни продукти, като *SQL Server Management Studio (SSMS),* както и други езици и технологии, като *HTML,* *CSS, Razor, Bootstrap, JavaScript* и др.

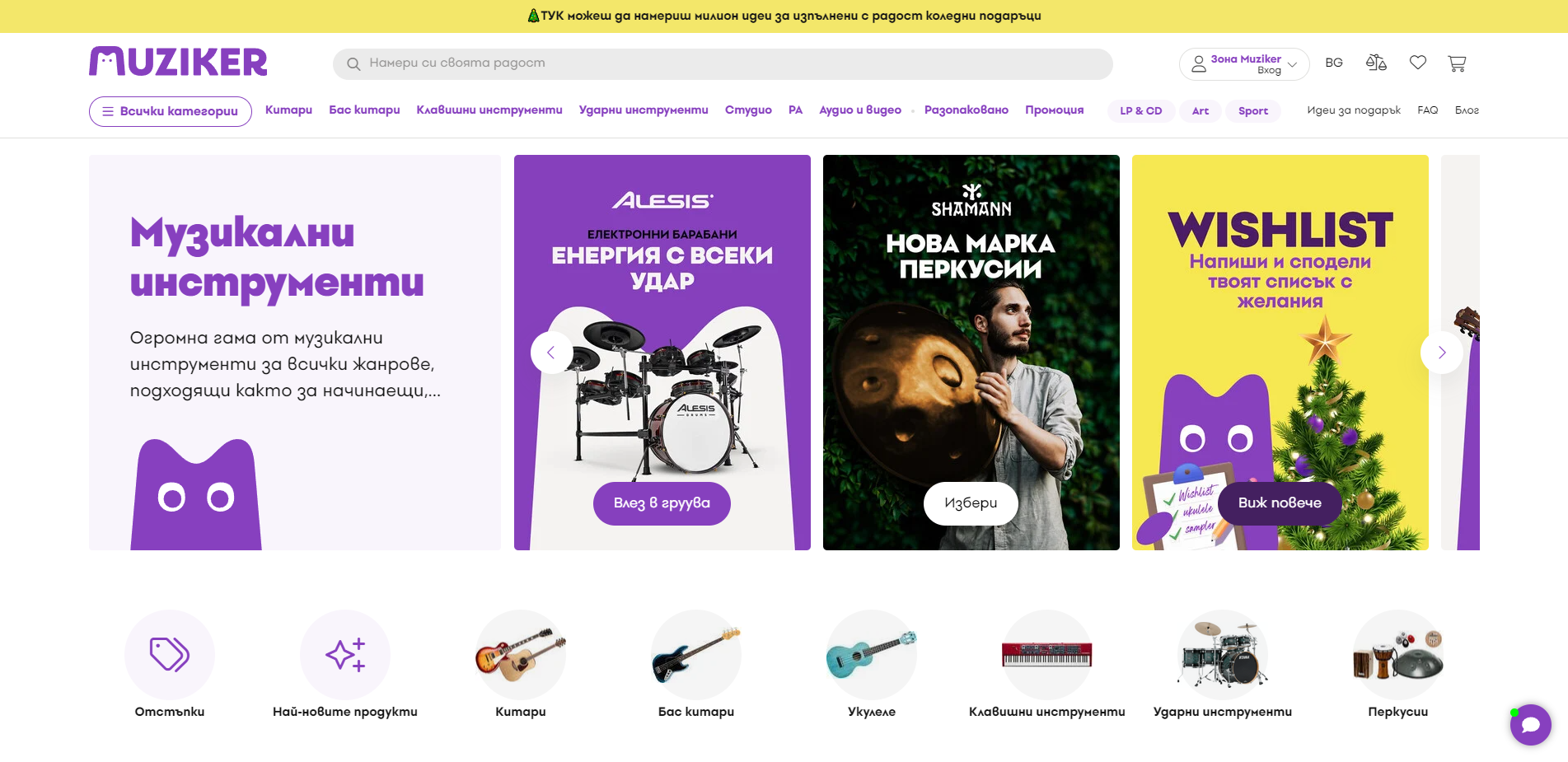
Описват се тези технологии, които наистина сте използвали в проекта!

# 1. ПЪРВА ГЛАВА – ПРОУЧВАНЕ

## 1.1. Проучване на пазара

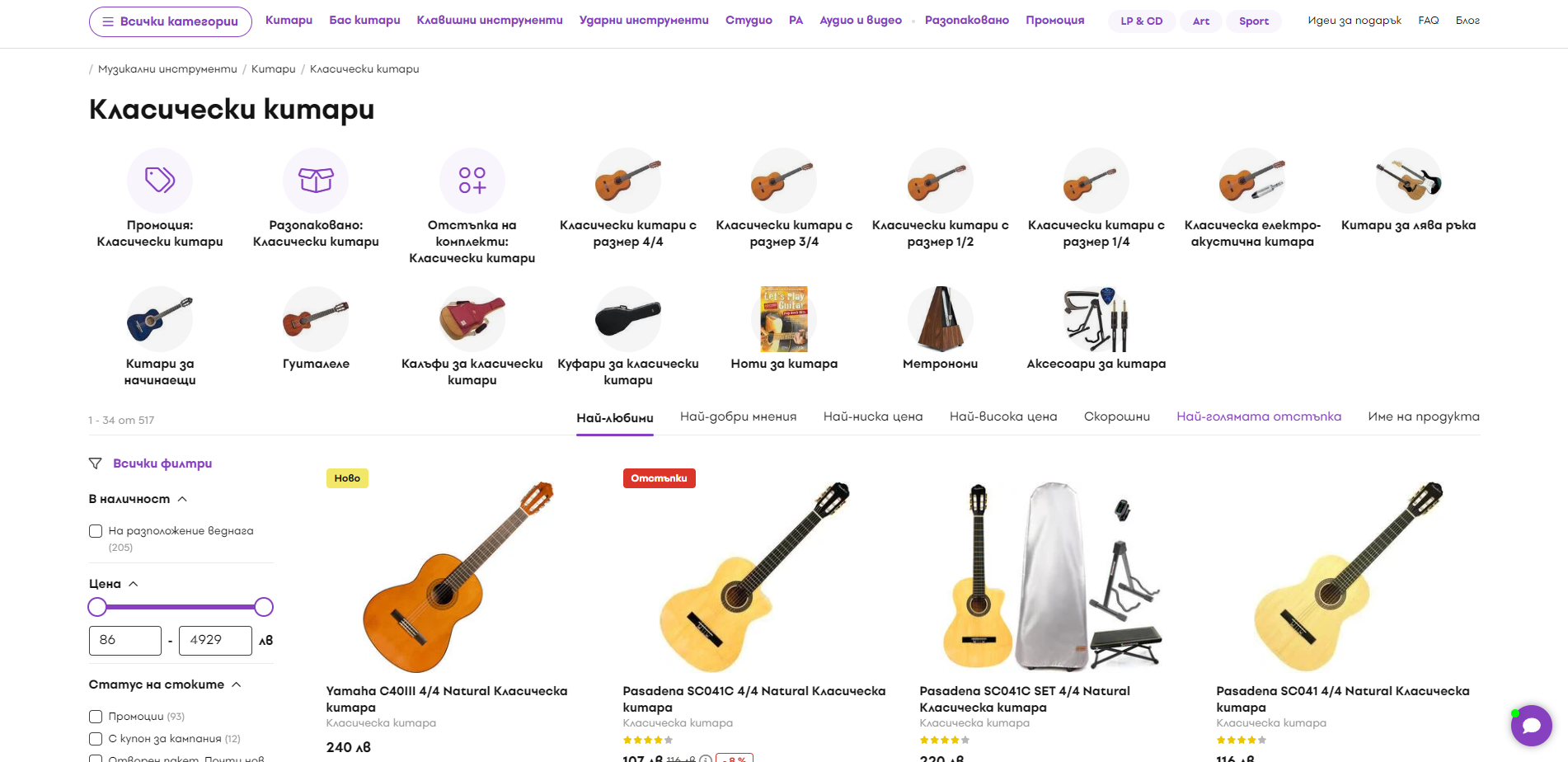
Реализацията на софтуера трябва да задоволява нуждите както на клиента , така и на собственика и разработчика. За да се избере най-добрия подход за реализацията, функционирането и поддръжката на продукта, трябва да се направи проучване на пазара.  
За направата на онлайн магазин за музикални инструменти са разгледани няколко вече съществуващи.

**1.https://www.muziker.bg/muzikalni-instrumenti**



Фигура 1.0.1

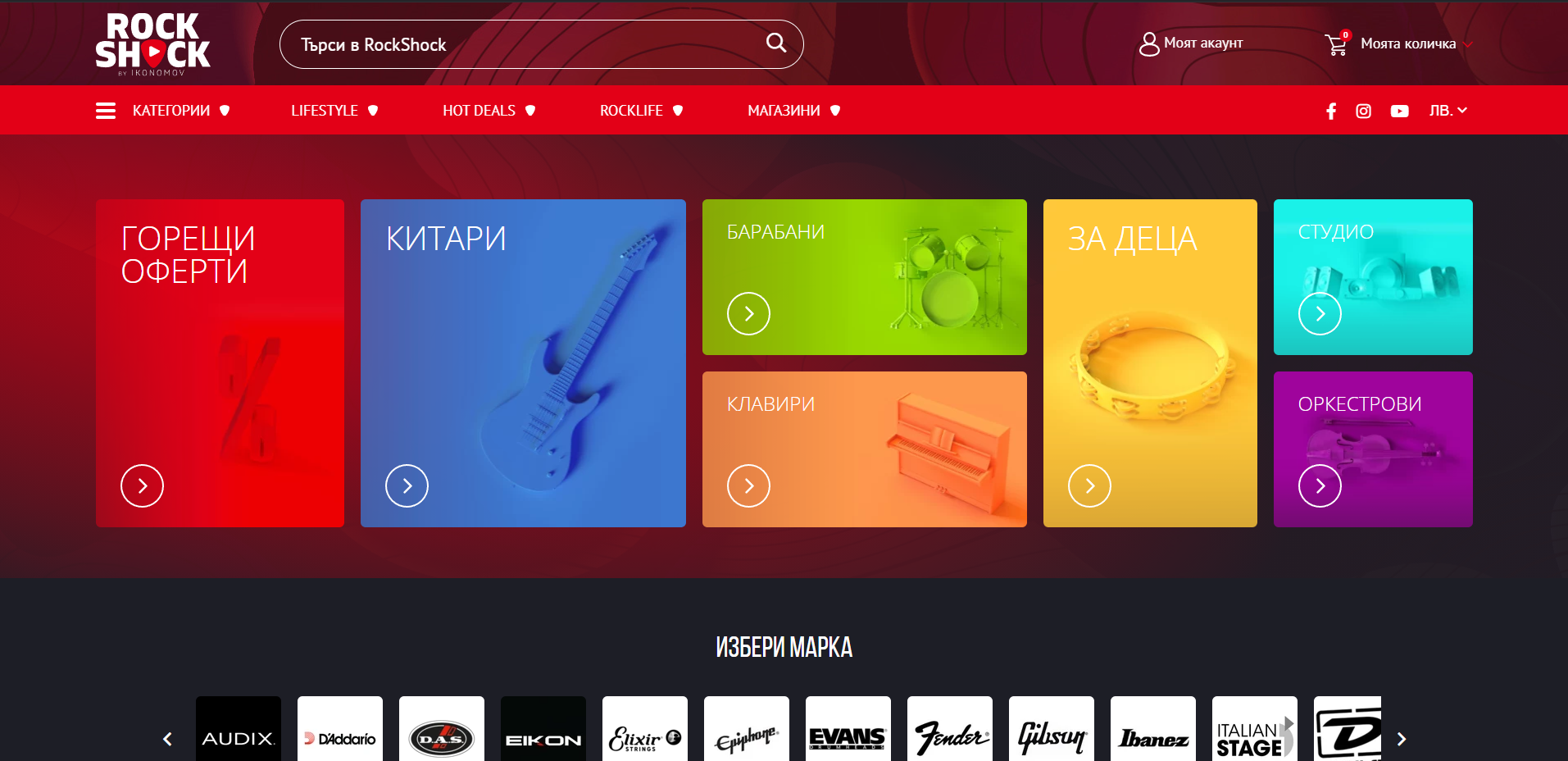
**Силни страни**-има няколко надписа уверяващи клиента, че са са на правилното място.Надолу в страницата има продукти на отстъпка, най-популярни и нови. Има поле за търсене на продукти на видимо място. Както фигура 2 показва може да избираме лесно подкатегориите в случай на нужда. Лесно видим е филтъра.Съществува функционалност за сравнение на продукти.



Фигура 1.0.2

**Слаби страни-**в началната страница елементите запродукти и категории са твърде големи, побирайки по-малко.

**2.** **https://rockshock.eu**

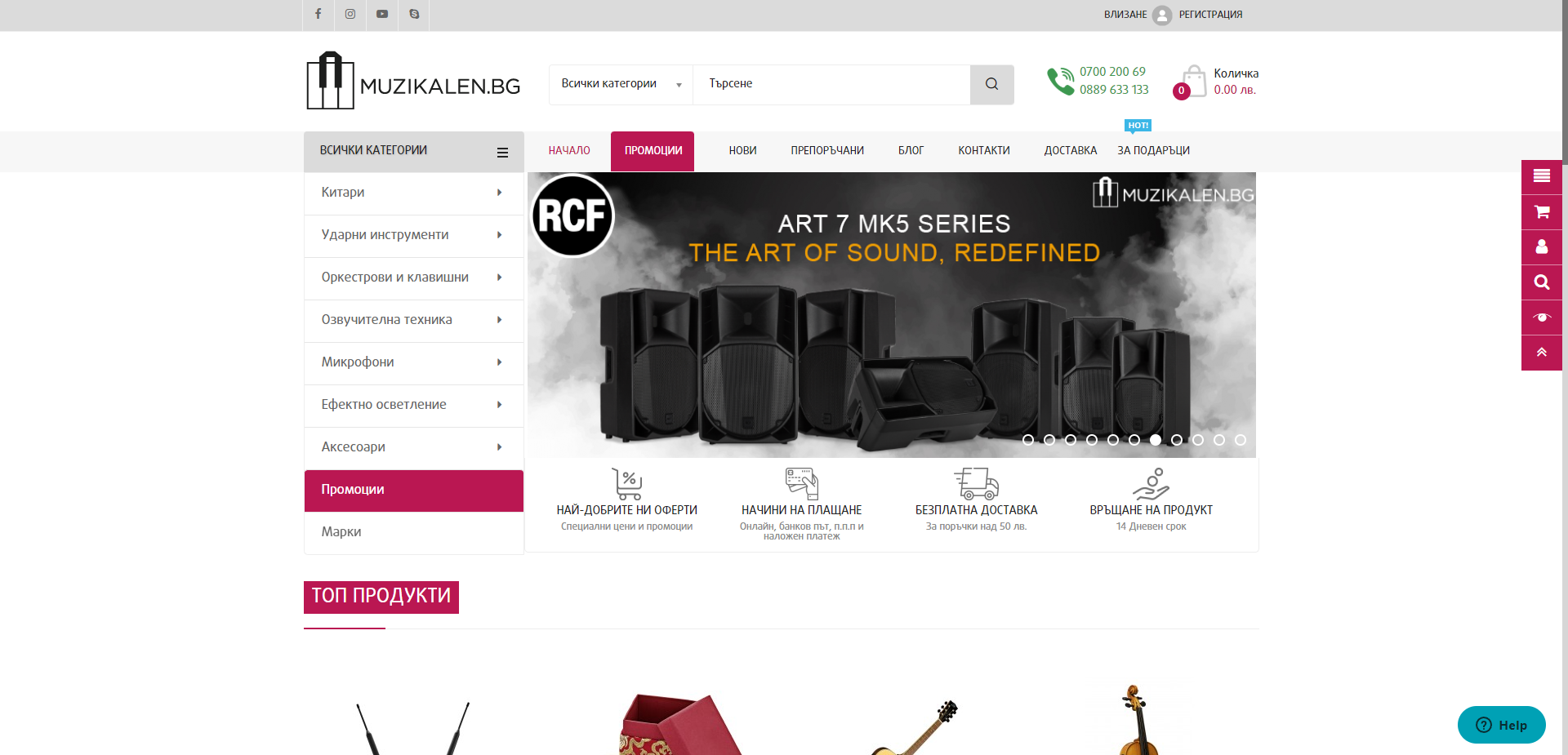


Фигура 1.0.3

**Силни страни-**темата на страницата е разноцветна, давайки по-добра ориентация.Зареждането на страниците в сайта става плавно, без да се презарежда горната част с навигационната лента и търсачката.

**Слаби страни-** броят на филтрите е малък.Съществува бутон за смяна на паричната единица, но той не работи. На места се ползва ненужно втори език.

**3.** [**https://www.muzikalen.bg/**](https://www.muzikalen.bg/)



Фигура 1.0.4

**Силни страни-**в началната страница е побрано почти всичко(контакти, категории, информация за доставка и начин на плащане и др.) без да е объркващо.

**Слаби страни-**Целият сайт е бял, липсват цветове.

Проучвайки конкутентните сайтове, установих, че повечето от тях имат твърде много категории на началната страница и клиентът трудно се ориентира. На други облика е твърде прост и не привлича вниманието с нищо.Проблем също е, че на някои продукти липсва описание или то е доста малко.

За да може да решим какви функционалности трябва да предлага продуктът ни, трябва да станем клиент и да се запитаме:“Какво искам от този сайт и какво ще го направи по-добър?“.Затова влязох в ролята на клиент в тези сайтове и анализирах дизайна, подредеността и облика им. По мои наблюдения няма един сайт в който всички тези неща да са изпълнени без забележки. Виждайки проблемите реших да осигуря тези функционалности на клиентите за по-добро пазаруване.

## 1.2. Преглед на технологии и езици за програмиране

При изграждането на съвременните приложения могат да се използват различни модели: трислоен, MVC, MVVM

**Трислоен модел–**състои се от презентационен слой(потребителски интерфейс), слой за визнес логика, служи за обработка на данните и слой за данни, който се състои от сървър база данни.[3]

Фигура 1.0.6 Трислоен модел

Вход

Обработка

БД

Обработка

Изход

Презентационен слой

Слой за услуги

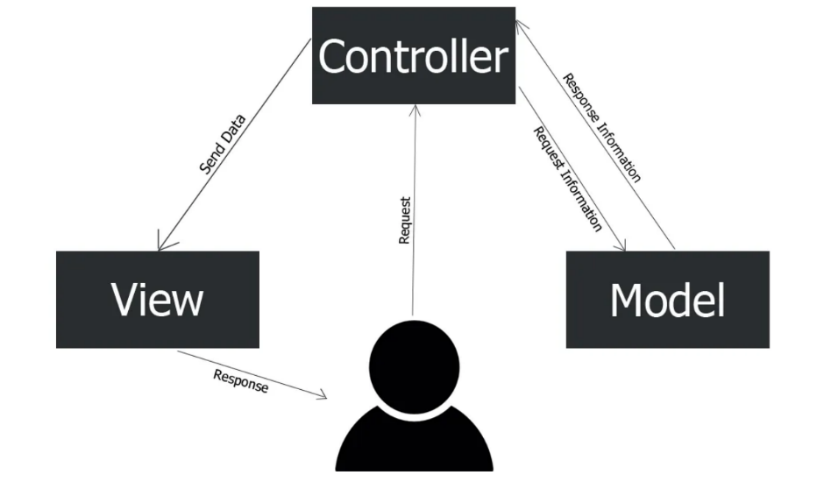
Слой за данни

**MVC шаблон –** Model-View-Controller е архитектурен шаблон, който подрежда папките и компонентите на приложението.[3]

*Моделът* - съдържа множество от класове, с които ще работим. Може да съдържа валидация и капсулация на данните. Работи с данните независими от потребителския интерфейс. Прилича на Data Access Layer от трислойния модел.

*Изгледът* – определя как ще се показва потребителския интерфейс на приложението. Изгледите са главни и частични.

*Контролерът* – главният MVC компонент, който е логиката на приложението.

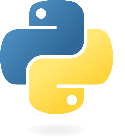


Фигура 1.7 MVC

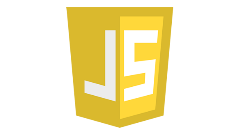
**MVVM (Model-View-ViewModel)** е архитектурен модел. Моделът и изгледа са както в MVC модела, но има нов компонент ViewModel, който работи с модела и помага да се запази изгледът отделен от модела, в същото време действа като контролер за улеснена комуникация между View и Model.[3]

В разработката на уеб приложения могат да се използват множество езици. Те се разделят на две главни категории Back-end и Front-end.Едни от най-популярните за Front-end са JavaScript, HTML/CSS , а за Back-end са Python, PHP, Java, C#.[5]

За разработката на сървърната част на приложението могат да се използват езиците:

* **Python** – популярността си дължи на своята универсалност и лесен за изучаване синтаксис. Той е широко използван в науката за данни, изкуствения интелект и уеб разработката;
* **Java** – продължава да бъде основен за разработка на мобилни приложения за Android. Това е езикът с една от най-големите общности от разработчици;
* **C#** – изключително популярен в разработката на игри и десктоп приложения, особено за платформата .NET. C# се отличава със своята мощност и гъвкавост.[6]
* **PHP** е бърз и гъвкав програмен език, на който се създават динамични уебсайтове – от блогове до онлайн магазини. Най-известната и използвана CMS платформа WordPress е написана на PHP.За разлика от останалите програмни езици PHP е специализиран език за уеб разработка и създаването на уебсайтове. [7]

**За стилистичната част(Front-end) се използват:**

* **JavaScript** –адаптивен скриптов език, който се ползва предимно за разработка на уеб интерфейси. Осигурява динамика и интерактивност на страницата.[8]
* **HTML/CSS** HyperText Markup Language е основния език за описание на уеб страници, а CSS (Cascading Style Sheets) е езикът за описание на стилове – определя как ще изглеждат елементите в страницата, работи с HTML.
* **Bootstrap –** най-популярната платформа с отворен код, която комбинира HTML, CSS и JavaScript код. Bootstrap се състои от вече готово написани класове. Има страхотна документация, лесно се персонализира и улеснява работата на разработчиците.
* **Razor –** поддържа C#. Това е много полезно в случай, че искаме да комбинираме C# и HTML код. Razor рендерира C# кода и го изкарва в браузъра като чист HTML

**Други технологии използвани в уеб приложение са:**

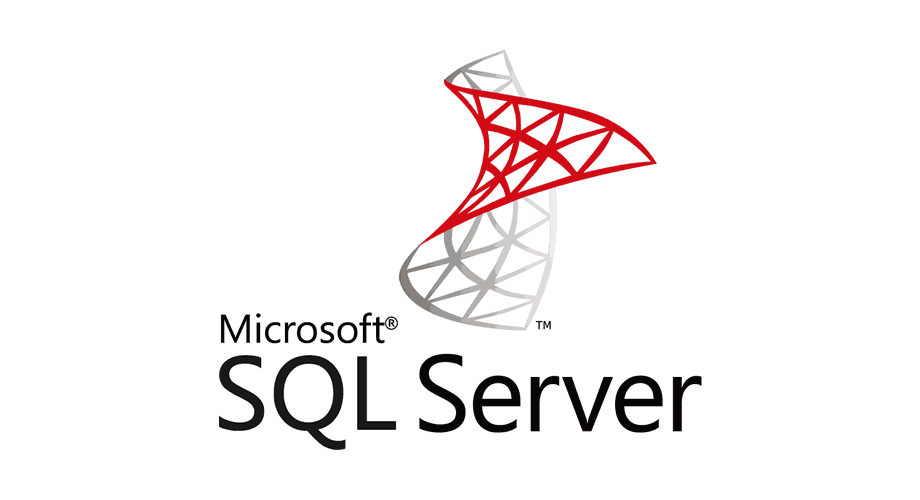
**ASP.NET Core-**framework, разработен от Microsoft за разработка на уеб приложения.Някои от предимствата е че поддържа различни платформи, и е с отворен код.

**Laravel-**е уеб фреймуърк на PHP за разработка на уеб приложения, който следва шаблона model-view-controller. PHP е сложен за поддържане и трябва постоянно да се следи какви нови технологии излизат.[10]

Производителността и сигурността не са толкова развити при Laravel, докато ASP.Net Core поддържа силна автентикация и авторизация, които го защитават от атаки като например Cross-site scripting (XSS).

Съвременните приложения използват бази данни за съхранение на информацията.Базите данни биват релационни и нерелационни.

**SQL –** Structured Query Language е език за работа с релационни бази от данни. С него правим заявки към бази данни, тъй като той е структурен език за заявки

**MS SQL Server** – система за управление на релационна база от данни, разработена от Microsoft

Тук се разглеждат популярни технологии и езици за програмиране, които се използват за разработка на уеб приложения. Това са технологии, които се използват от Front-End и Back-End разработчиците. Тук трябва да опишете основните им характеристики, предимства и недостатъци, приложение. Задължително трябва да разгледате: трислоен модел, MVC, MVVM, езици за програмиране /C#, PHP, Java и др/, бази данни /релационни и NoSQL , СУБД, MySQL, MS SQL Server/, ASP.NET, ASP.Net Core MVC, HTML, CSS, Razor, Bootstrap, JavaScript, среди за разработка като Visual Studio Community и др. Хубаво е да разгледате и други алтернативни технологии на посочените. Задължително трябва да бъдат разгледани технологиите, които ще използвате в проекта.

*Забележка: Важно е да цитирате източниците, които сте ползвали при описание на технологиите. Използваната литература да бъде цитирана в текста чрез [].*

Например, ако сте черпили информация от документацията на Microsoft за ASP.NET:

ASP.NET е уеб рамка с отворен код, създадена от Microsoft, за изграждане на модерни уеб приложения и услуги с .NET.

ASP.NET е междуплатформен и работи на Windows, Linux, macOS и Docker. [6]

## 1.3. Избор на технологии и езици за програмиране

За разработката на моето уеб приложение избрах технологията ASP.Net Core с езика C# и бази данни MS SQL Server, както и други езици и шаблони, които ще се използват за разработването на проекта, заради описаните по-долу предимства.

Предимствата на MVC са че компонентите са отделени и могат да се използват многократно и в други приложения.

ASP има доста предимства. NET платформата се поддържа от Microsoft, което е голям плюс, защото това е огромна, доказала се със своите продукти и услуги, компания.

NET Core е само част от NET платформата, която осигурява най-добрата среда за разработка и полезен дебъгер. Има страхотна документация и голяма общност, където хората споделят опита си и пробват нови проекти. Тази технология е с отворен код. NuGet – има над 166,000 библиотеки.

Чрез NET Core се създават десктоп приложения, уеб приложения, мобилни приложения, облачни услуги, игри, приложения за интернет на нещата. Голям плюс е че уеб приложенията разработени на тази платформа работят както на Windows, така и на macOS, и на Linux.[2]

Технологията позволява и работа с AZURE, облачна технология, за хостване на уеб приложението. ASP.Net Core се използва и за сложни проекти, което е още едно предимство, тъй като може да се направи сериозно приложение

**C# -** обектно-ориентиран език, който е много лесен за учене, има красив синтаксис, грешките са лесни за разбиране. C# е един от най популярните езици, като от 2019 година той става още по-използван и популярен, което го прави надежден и може да се разчита на този език дълго време. Има добре развити garbage collector, threading, LINQ и други. Можем да пишем C# код в браузъра – Blazor. C#, освен като ООП, може да се ползва като функционално програмиране или да се пише и Machine learning.[1]

Причината да избера HTML, CSS, JavaScript и Bootstrap е че те са най-популярните и универсални езици.

# 2. ВТОРА ГЛАВА – ПРОЕКТИРАНЕ

Описание и анализ на изискванията към програмния продукт; разглеждане на потребителски истории и случаи на употреба на продукта; прототип на потребителски интерфейс на приложението. Проектиране и описание на таблиците от БД и връзките между тях. Представяне на прототип/модел на диаграма на БД./8-15 стр./

Проектирането е важен етап от цикъла на софтуерната разработка. Необходимо е да се анализират изискванията към програмния продукт, да се проектира БД, да се създадат прототипи на потребителския интерфейс.

## 2.1. Описание и анализ на изискванията

### 2.1.1.Описание и анализ на функционалните изисквания

Функционалните изисквания определят какво трябва да прави системата. Те са свързани с видовете потребители и техните права.

Копирайте изискванията от заданието /преди описанието на БД/.

Приложението ще поддържа надеждна система за автентикация и авторизация на своите потребители, които ще бъдат разпределени в три роли: нерегистрирани потребители (гости), регистрирани потребители (клиенти) и администратор.

Всички потребители трябва да виждат меню със секции „Начало“, „Продукти“ и „Контакти“. Страницата „Продукти“ ще позволява разглеждане на продуктите в сайта. При разглеждането да има възможност за филтриране на информацията по цена и производител. Нерегистрираните потребители (гостите) трябва да могат да се регистрират в системата. Регистрираните потребители (клиенти) трябва да могат да се логват в системата чрез потребителско име и парола. Приложението ще позволява логнатите потребители да правят поръчка на продукти. Те ще могат да разглеждат своите собствени поръчки на продукти. При поръчка на продукти системата ще следи за изчерпване на наличните количества. Преди финализане на поръчката потребителят ще вижда общата дължима сума.

Приложението ще има един администратор с вградени в системата данни. Администраторът ще има възможност да добавя, редактира и изтрива данни за продуктите, предлагани в сайта. Той ще може да вижда направените поръчки на всички клиенти. Администраторът ще има възможност да вижда данните и да изтрива регистрирани потребители, които не са направили поръчки. Администраторското меню ще съдържа секция „Статистически модул“ със статистическа информация за брой регистриани потребители, брой продукти ..........

Допълнителни функционалности, които ще бъдат реализирани, са:

### 2.1.2. Потребителски истории (user stories)

Потребителските истории се използват при разработването на софтуер, като начин да се помогне на разработчиците да разберат желанията и нуждите на своите потребители.Потребителските истории са кратки и следват прост шаблон.

**Предимства:**

* Кратки
* Разбираеми за потребителите и разработчиците
* Не е нужна поддръжка
* Не налагат особени усилия

**Недостатъци**

* Може да са непълни
* Различна интерпретация

Потребителските истории имат за цел да представят гледната точка на потребителя, а не гледната точка на създателя (разработчика на програмата). Те могат да са от голяма полза при определяне на случаите на употреба.

**Шаблон за потребителски истории (кой-какво-защо):**

*„Като (потребител) искам да (действие/възможност), за да (цел).”*

*„Като (потребител) мога да (действие/възможност), за да (цел).”*

Този шаблон изобразява представата на потребителя, като идентифицира какво иска/може/ и защо го иска.

**Примери:**

**Гост**

* „Като гост искам да мога да разглеждам продуктите в сайта, за да мога да избера предпочитан продукт.“
* „Като гост искам да имам възможност за филтриране на продукти по даден критерий, с цел по-бързото им намиране.“
* „Като гост искам да мога да се регистрирам в системата, за да стана регистриран потребител.“

**Регистриран потребител – клиент**

* „Като клиент искам да мога да се логвам в системата с потребителския си профил, за да мога да поръчвам продукти.“
* „Като клиент искам да имам възможност да разглеждам продуктите, за да знам какво да си поръчам.“
* „Като клиент искам да имам възможност за филтриране на продукти по даден критерий, с цел по-бързото им намиране.“
* „Като клиент искам да мога да поръчвам продукти, с цел тяхното закупуване.“
* „Като клиент искам при поръчка на продукт да виждам дължимата сума, с цел да осигуря средства за плащането.“
* „Като клиент искам да имам възможност да виждам историята на всички мои поръчки, направени досега.“

**Администратор**

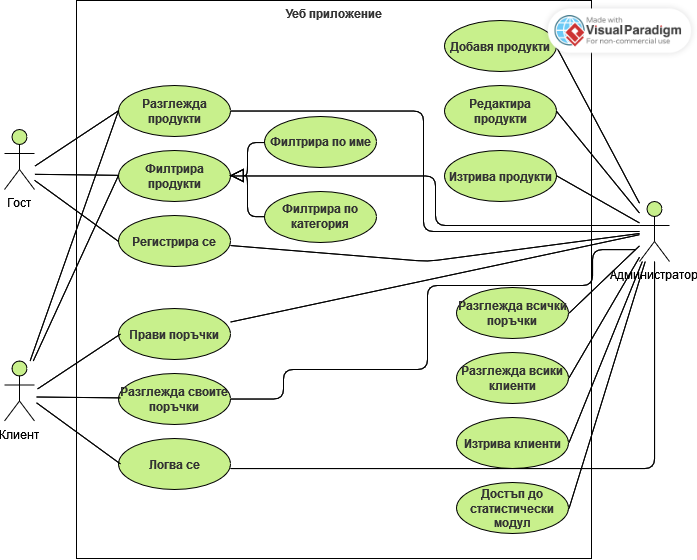
* „Като администратор искам да мога да се логвам в системата с администраторски профил, за да имам достъп до всички модули.“
* Като администратор искам да мога да добавям, редактирам и изтривам продукти, с цел закупуването им от клиентите.“
* „Като администратор искам да имам достъп до всички направени поръчки от клиенти, за да мога да виждам подробности за тях.“
* „Като администратор искам да виждам всички регистрирани потребители и тяхната информация.“
* „Като администратор искам да мога да изтривам регистрирани потребители.“
* „Като администратор искам да имам достъп до статистическия модул в системата, за да мога да виждам различна статистическа информация.“

Напишете потребителски истории, в зависимост от контекста на вашия дипломен проект. Опитайте се да напишете такива истории, които наистина са реализирани в сайта.

Написаните потребителски истории ще ви послужат за изготвянето на случаи на употреба за системата.

### 2.1.3. UML диаграми за случаи на употреба

Случаите на употреба показват какво могат да правят потребителите в различни роли. Един ясен начин на представянето им е чрез UML диаграми. Те осигуряват визуализация на случаите на употреба и действащите лица в системата.



***Фигура №*** *UML диаграма на случаи на употреба*

***Може да използвате free online tools за UML диаграмите***

### 2.1.4. Анализ на нефункционалните изисквания

Нефункционални изисквания налагат ограничения по отношение на дизайна, имплементацията, използвания софтуер, необходимия хардуер и други изисквания към системата, които не попадат към функционалните изисквания.

Например: изисквания за дизайн, ефективност, производителност, сигурност, качество на софтуера, бизнес правила и др.

Нефункционални изисквания към нашето приложение се отнасят до:

* + Сигурност
  + Потребителски интерфейс
  + Качество на софтуера
  + Архитектура на приложението
  + Технологии и софтуер за разработка

***Сигурност на използването на приложението***

Приложението трябва да поддържа надеждна система за автентикация и авторизация на своите потребители. Това означава, че в зависимост от ролята си потребителят има възможност да достъпва само определени страници, за другите ще му бъде отказан достъп.

***Изисквания към потребителския интерфейс***

Потребителският интерфейс трябва да бъде лек и интуитивен, с подходящо подбрани цветове и размер на компонентите. Това ще позволи ....... При въвеждане и редактиране на данни, трябва да има контрол на валидността им, за да не се допуска трафик на невалидни данни до сървъра. Дизайнът на страниците трябва да е адаптивен и да изглежда добре на различни устройства.

***Качество на софтуера***

Приложението трябва да бъде създадено чрез спазване на добрите практики за софтуерна разработка и писане на код. Това означава спазване на SOLID принципите при създаване на кода: класовете и методите да имат единствена отговорност, кодът да бъде „отворен“ за разширение и „затворен“ за модификация, интерфейсите да включват само най-необходимите методи, да има „инжектиране“ на външните за класа зависимости. Кодът да бъде добре форматиран и при необходимост - коментиран.

***Архитектура на приложението***

Уеб приложението ще бъде трислойно. Слоят за достъп до данните, бизнес слоят и презентационният слой ще бъдат обособени като отделни проекти.

***Изисквания към софтуера за разработка***

Приложението ще бъде създадено чрез средата за програмиране *Visual Studio Community 2022,* БД ще се управлява от *MS SQL Server 2022*, а за работа с нея ще се използва *SQL Server Management Studio (SSMS) 19.1*

***Изисквания към технологиите и езиците за разработка***

* Базов проект за ASP.NET Core включващ MVC - .Net 6.0
* Библиотеки, необходими за проекта:
* Microsoft.EntityFrameworkCore.SqlServer,
* Microsoft.AspNetCore.Identity.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.Tools
* Microsoft.EntityFrameworkCore.Proxies
* Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore
* Microsoft.VisualStudio.Web.CodeGeneration.Design - напишете и други, ако използвате.
* C#, SQL, HTML, CSS, Bootstrap, JavaScript и др.

## 2.2. Проектиране

### 2.2.1. Проектиране на база данни

Приложението за продажба на .... ще използва БД. Преди да започне разработката, е необходимо да изясним какви таблици ще има в БД и какви ще бъдат връзките между нях.

Тук трябва да опишете таблиците, какви полета ще съдържат, типове данни за всяко поле и евентуални ограничения.

Опишете релациите (връзките) между тях и как таблиците от вашата база от данни ще бъдат свързани с Identity System на ASP.Net Core.

Направете модел - прототип на диаграма на базата от данни.

Пример:

Базата от данни на приложението ще включва таблиците от системата за идентичност на ASP.NET Core и още .... таблици.

Таблицата с потребителите, идваща от системата за идентичност на ASP.Net Core, ще бъде разширена със следните колони /полета/:

* FirstName – тип string
* LastName – тип string
* Address – тип string

За всеки продукт ще се съхранява информация в таблица Products:

* Id – задължително поле от тип int, идентификационен номер. Ограничения: първичен ключ, уникални стойности
* Name - задължително поле от тип string за име на продукта
* Description – поле от тип string за описание на продукта
* Picture – поле от тип string за URL на снимката на продукта
* CategoryId - задължително поле от тип int. Ограничения: външен ключ към таблица Categories
* BrandId - задължително поле от тип int. Ограничения: външен ключ към таблица Brands
* Price - задължително поле от тип decimal за цена на продукта
* Qantity - задължително поле от тип int за наличност на продукта

За всяка категория ще се съхранява информация в таблица Categories:

* Id – задължително поле от тип int, Ограничения: първичен ключ, уникални стойности

• Name - име на категория - задължително поле от тип string

Опишете всички таблици на вашата система …..

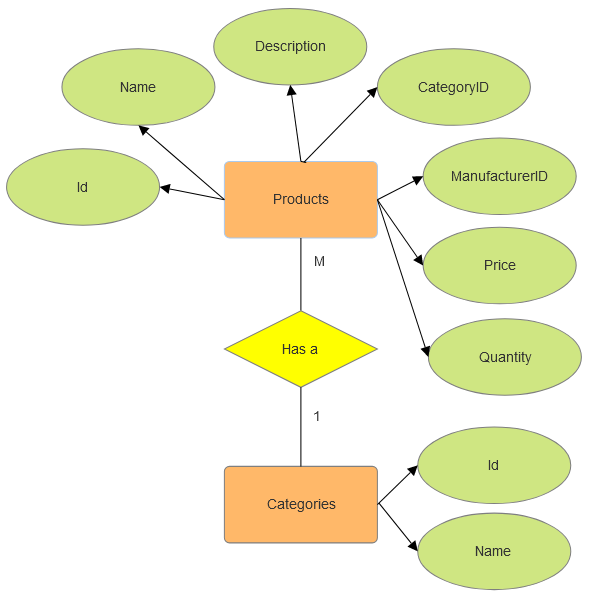
Връзките между таблиците са следните:

* Categories - Products – Едно към много, реализирана чрез полето Id на таблица Categories и полето CategoryId на таблицата Products

Опишете всички връзки между таблиците ….

Направете модел - прототип на диаграма на базата от данни.

Пример за прототип на диаграма на БД, която включва две таблици: Продукти и Категории



Фигура № Прототип на E/R диаграма на БД

### 2.2.2. Прототипи на потребителския интерфейс

Прототипите дават нагледна представа за визията на софтуерния продукт. Те позволяват на дизайнерите да покажат продукта си, което го прави по-лесен за разбиране. Прототипи могат да се създават по време на всеки етап от процеса по дизайн, за да помогнат да се демонстрират идеи, които трудно биха се изразили само с думи.

Тук направете ваши прототипи на страниците. Прототипът да бъде направен с помощта на програма, например Balsamiq Wireframes for Desktop: [https://balsamiq.com/wireframes/desktop/**#**](https://balsamiq.com/wireframes/desktop/)**.**

**Пример:**



***Фигура №***

***Направете прототипи на всички страници, от които ще се състои вашето приложение.***

# 3. ТРЕТА ГЛАВА – РЕАЛИЗАЦИЯ

Същинската част на дипломния проект, която е с най-голям обем. Да включва избор на подход на реализация, структура на базата от данни /чрез E/R диаграма/ и описание на полетата на всяка таблица, описание на архитектурата на проекта, контролери, изгледи, сървиси, **вю модели.**  Да се опишат реализираните функционалности, в това число чрез представяне и описване на екрани от приложението. /15-22 стр./

**Забележка: Това трябва да е най-дългата глава от вашия дипломен проект.**

В тази глава не се показва код, който е генериран автоматично от шаблона на ASP.Net Core.

Реализацията на приложението изисква да се направи избор между двата подхода на изграждане на уеб приложение с използване на БД: Code First и Database First. Опишете двата подхода и на кой от тях сте се спрели. Това е от съществено значение при описване на последователността на работа и необходимостта от описание на класовете за данни.

## 3.1. Технология на създаване на уеб приложението

За реализация на сайта ще се използва базов проект за ASP.NET Core на езика C#. Базовият проект има вграден MVC модел, който ще бъде разширен за да се изпълнят целите на уеб приложението.

**Стъпки за изграждане на уеб приложението**

Опишете стъпките /последователността/ за изграждане на приложението.

**По-важните стъпки при изграждане на уеб приложението са:**

* Създаване на базов проект от тип ................... Това ще бъде презентационният слой на приложението.
* Добавяне на два нови проекта от тип ...................... Проектът ..... представлява слой за достъп до БД, а проектът ....... – бизнес слой на приложението.
* За работа с потребители и роли се използва системата за идентичност на ASP.NET, като се прави scaffold /автоматично генериране на код/ и корекции на Register, Login, Logout. Автоматично е създаден администратор с вградени в приложението данни.
* Тъй като за реализация на приложението е избран подоходът Code First, следващата стъпка е създаването на класовете, описващи данните, в папка .... на проекта ...... При първоначалното създаване на БД, автоматично се записват данните за категориите и производителите в съответните таблици.
* Създаване на система за навигация в сайта.
* Имплементиране на CRUD операциите за продукт/услуга.
* Имплементиране на създаване на поръчка и преглед на поръчки. И т.н. Опишете какво друго сте направили.

## 3.2. Структура на базата от данни

За успешното създаване на базата от данни на приложението са създадени множество класове за описание на данните. Приложението използва следните класове:

Конкретни описания на пропъртитата(свойствата) за всеки клас, който съответства на таблица от БД, както и на класа ApplicationUser.cs: тип на пропърти(свойство), ограничения, атрибути. Опишете навигационните и свързващите пропъртита. Добре е тук или в отделно приложение да направите скрийншот на класовете.

Връзки между класовете (таблиците).

Как те се свързват с Identity System на ASP.NET Core.

**E/R диаграма на вече създадената база от данни.**

## 3.3. Реализация на уеб приложението

Описвате архитектурата на уеб приложението, контролери, сървиси, вю модели и изгледи.

### 3.3.1. Архитектура на приложението

Сложете скрийншот. Опишете архитектурата на уеб приложението – какви проекти, какви папки, общо какво има в тях. Напишете за трите проекта и че презентационният слой е изграден по модела MVC.

### 3.3.2. Контролери

Описвате контролерите и тяхната функционалност. За всеки контролер – какви атрибути сте сложили и защо; какви сървиси са инжектирани; какви екшъни са създадени и тяхната функционалност; как работи съответният екшън /get, post и др./

### 3.3.3. Сървиси

Опишете какви интерфейси сте създали, кои сървиси ги имплементират, какви методи има в тези сървиси и за какво служат тези методи. Може да покажете сигнатурата на методите и да напишете тяхното предназначение.

### 3.3.4. Вю модели

Описвате какви вю модели сте използвали.

### 3.3.5. Изгледи

Тук трябва да опишете и представите снимки на екрани/изгледи от работата на вашето приложение. Започнете с изгледа на началната страница на сайта си. Покажете снимки и обяснете какво може да прави нерегистрираният потребител, как може да се регистрира, какво вижда и може да прави като клиент на сайта. Аналогично покажете снимки и опишете функционалностите за администратора.

### 3.3.6. Други технологии и функционалности

Опишете какви други технологии сте използвали. **Акцентирайте върху функционалностите, с които сте обогатили своя проект, извън заданието.** Ако сте описали функционалностите в предишната точка – тук само ги избройте, ако не сте – тук покажете екраните и опишете по-подробно в какво се изразява съответната функционалност.

### 

# ЗАКЛЮЧЕНИЕ

Обобщава постигнатото в дипломния проект, представя възможностите за доразвиване и усъвършенстване, както и възможностите за приложение на създадения софтуер. /1-2 стр./

# ИЗПОЛЗВАНА ЛИТЕРАТУРА

**Включва цитираната и използвана в записката на дипломния проект литература.** Започва на отделна страница от основния текст. При имената на авторите първо се изписва фамилията. Всички описания в списъка с използваните източници трябва да са подредени по азбучен ред според фамилията на първия автор на всяка публикация.

Например:

1. //Колсиченко, Д. Адаптивен уеб дизайн с Bootstrap, Асеневци, 2017
2. //Наков, С. и колектив. Принципи на програмирането със C#. Фабер, 2018
3. //Наков, С. и колектив. Програмиране за .NET Framework. БАРС, София, 2006
4. //Свободно учебно съдържание по програмиране и ИТ за българските училища на Работна група „Образование по програмиране и ИТ“ – <https://github.com/BG-IT-Edu>
5. <https://survey.stackoverflow.co/2024/technology>
6. <https://note.bg/kakvi-vidove-ezici-za-programirane-sa-populyarni-prez-2023-g>
7. <https://blog.superhosting.bg/5-most-popular-programming-languages-in-web-development.html>
8. <https://www.techbulgaria.com/?p=1701>
9. <https://developer.mozilla.org/en-US/docs/Web/HTML>
10. <https://bg.wikipedia.org/wiki/Laravel>
11. //<https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet>
12. //<https://www.w3schools.com/js/default.asp>

# ПРИЛОЖЕНИЯ

Всяко на нова страница и да са номерирани /Приложение 1, Приложение 2 .../ Приложенията съдържат документация, която не е намерила място в текста поради ограниченията в обема й или за по-добра прегледност/подредба. В текста трябва да има препратка към всяко приложение.