



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

ДИПЛОМЕН ПРОЕКТ

**Тема: Разработка на уеб приложение
„Онлайн ресторант“**

Ученик: Моника Тихомирова Блажева

Специалност: Системно програмиране

Професия: Системен програмист

Научен ръководител: Иван Илиев



СЪДЪРЖАНИЕ

1. Увод.....	4
1.1. Цел на проекта.....	5
2. Архитектура на приложението.....	6
2.1. Монолитна архитектура.....	6
2.2. All-in-one MVC.....	8
3. Използвани технологии.....	9
3.1. Microsoft Development Stack.....	9
3.2. .NET 6.....	9
3.3. Visual Studio 2022 Community Edition.....	10
3.4. C#.....	11
3.5. ASP.NET Core Web App (Model-View-Controller).....	12
3.6. Microsoft SQL Server.....	13
3.7. SQL Server Management Studio.....	14
3.8. Entity Framework Core.....	14
3.9. Identity (on ASP.NET Core).....	15
3.10. .Net Mail.....	16
3.11. JSON.....	16
3.12. Ajax.....	17
3.13. Braintree.....	17
3.14. Braintree Sandbox.....	18
3.15. Nicepage.....	19
4. База данни.....	20
4.1. ApplicationDbContext.....	22
4.2. ContextSeed.....	22
4.3. Моделът ApplicationUser.....	23
4.4. Моделът Product.....	24
4.5. Моделът Sushi.....	24
4.6. Моделът Combo.....	25
4.7. Моделът Dessert.....	26
4.8. Моделът Beverage.....	26
4.9. Моделът CartItem.....	26
4.10. Моделът Order.....	27
4.11. Моделът OrdersItem.....	28
5. Контролери.....	30
5.1. Sushis, Combos, Desserts и Beverages контролери.....	30
5.2. ProductsMenu контролер.....	30
5.3. Favourites контролер.....	30
5.4. ShoppingCart контролер.....	32



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

5.5. Orders контролер.....	34
5.6. ToDoList контролер.....	36
5.7. UsersRoles контролер.....	36
6. Помощни класове.....	37
6.1. HttpContext.....	37
6.2. EmailSender.....	37
6.3. CartService.....	38
6.4. Braintree.....	40
7. Уеб сайт.....	42
7.1. Начална страница.....	42
7.2. Меню с продукти.....	43
7.3. Регистрация.....	44
7.4. Логин.....	45
7.5. Детайли на потребителския акаунт.....	45
7.6. Пазарска количка.....	49
7.7. Осъществяване на поръчка.....	49
7.8. Детайли за поръчка.....	50
7.9. To-do лист.....	51
7.10. CRUD на Sushis, Combos, Desserts и Beverages.....	52
7.11. Мениджър на потребителските роли.....	54
8. Заключение.....	55
9. Използвана литература.....	56



1. Увод

Технологиите са незаменима част от съвременния свят и включват широк спектър от научни и инженерни постижения, които подобряват начина, по който функционираме и взаимодействаме със света около нас. Една от най-забележителните технологии, която промени живота на милиони хора по света, е интернетът. Освен бързо и лесно споделяне на всяка къмпания информация, комуникация с хора от различни краища на света, интернетът също така стимулира онлайн търговията и предоставя възможност за нейното развитие. Потребителите могат да закупуват онлайн стоки и услуги от всяка вид в комфорта на своя дом или офис.

Уеб приложенията са достъпни от всяко устройство с достъп до интернет, без да е необходима инсталация или специфична операционна система. Те работят върху сървъри и се достъпват чрез браузъри, което означава, че не е нужен мощен компютър или устройство, за да се използват. Това прави уеб приложенията много удобни за голям брой потребители. Също така могат да бъдат актуализирани, без да е необходимо инсталиране на нова версия. Потребителите могат да се възползват от новите функции веднага, без инсталација на нова версия. Поради тези причини, много фирми и организации използват уеб приложения, за да постигнат по-голяма ефективност и да подобрят комуникацията и взаимодействието със своите клиенти.

Напредъкът в бързината и множеството от функционалностите на уеб приложенията е несравним. Тези две тенденции са силно свързани и взаимно подпомагащи се, тъй като по-бързите приложения могат да използват по-голям брой функционалности, а по-голямото количество функционалности може да бъде използвано единствено върху по-бързи приложения.

В днешно време има множество платформи за доставка на храна, като например Uber Eats, Glovo, Bolt и Foodpanda. Те предоставят възможност на потребителите онлайн да изберат и поръчат храна от голям брой ресторани и менюта и да я получат в удобно за тях време и място. Също толкова важни и полезни са и за самите ресторантъри и ръководители, тъй като им позволява да разширят бизнеса си и да достигнат до по-голям брой клиенти. В основата на тези платформи са именно мобилните приложения и уеб сайтове.

1.1. Цел на проекта

Идеята, която стои зад настоящия проект, е да се създаде една платформа за доставка на храна, наподобяваща на по-горе споменатите, а такава, която да пренесе бизнеса, свързан с ръководството на ресторант, онлайн. Благодарение на уеб приложението, изцяло проектирано според услугите, които предлага ресторантът, може да се избегне нуждата от наемане/закупуване на помещение за



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

настанияване на клиенти и персонал за тяхното обслужване. Замяната на физическия ресторант с уеб приложение и обслужващия персонал с доставчици е иновативен и съвременен начин да се улесни стартирането на нов бизнес. Това може да бъде особено полезно за малки предприятия, които нямат достатъчно ресурси, за да отворят традиционен ресторант.

В този ред на мисли основната цел на проекта е да се улесни възможно най-много взаимодействието между собственика, служителите и клиентите. Осъществяването на тази цел ще бъде постигнато чрез асинхронна обработка на данни, за да може приложението да извършва различни задачи в едно и също време, без да блокира основния процес на изпълнение. Всеки потребител ще може да си направи акаунт и в зависимост от ролята на потребителя, уеб приложението ще предоставя достъп до различни функционалности:

- Без регистрация клиентът ще има възможност само да разглежда сайта, а след създаване на потребителски профил ще може да добавя продукти от менюто в кошницата си, да ги поръча и плати онлайн. В профила си ще може да разгледа списъка си с “любими продукти”, както и настоящите и предишните си поръчки.
- Служителите, отговарящи за изпълнението на изпратените от клиентите поръчки, ще имат възможност да следят поръчките, които трябват да бъдат пригответи за доставка, както и да ги отбелязват като готови след предаването им за доставка.
- Роля “Админ” ще притежават управителите на ресторанта. Чрез нея те ще могат да регулират продуктите, предлагани в менюто на сайта – да добавят нови, да променят и изтриват останалите.
- За собственика или главния ръководител на бизнеса е предвидена специална роля “Супер Админ”, чрез която ще се осъществява управлението на ролите на потребителите.



2. Архитектура на приложението

"Ако мислите, че добрата архитектура е скъпа, опитайте лошата архитектура." – Брайън Фут и Джоузеф Йодерити.

Създаването на софтуер изисква планиране. Без план дори най-травиалният проект може бързо да се превърне в непоправима безредица. Именно поради тази причина архитектурата на приложението е от изключителна важност. Подходящата софтуерна архитектура позволява да се постигне по-високо качество и по-добра ефективност и да се избегнат бъдещи усложнения. Няма универсална архитектура, която да може да се приложи за всеки проект. Например, едно по-комплексно приложение би изисквало по-сложен концептуален модел, отколкото едно не толкова сложно приложение. Добре проектираният софтуер може да бъде модифициран много по-лесно, отколкото този с лоша архитектура. Така, когато бъде пуснат и след това има нужда от допълнителни функции, процесът на интегриране на новите функционалности ще бъде възможно най-безболезнен.

2.1. Монолитна архитектура

Монолитната архитектура е стандартен модел за софтуерно приложение, при която то е изградено като цялостна единица и е самостоятелно и независимо от други приложения (fig. 1). Тя е удобна за по-малки или стартиращи проекти, поради по-лесното управление на кода и по-бързото внедряване. Предвид големината на настоящия проект е удачно да бъде изграден като монолитно приложение. При евентуално разрастване на потребността и нужда от увеличаване на функционалностите и услугите, може да се премине към микросървисна архитектура. При нея приложението е разделено на различни „услуги“, свързани помежду си. „Услугите“ могат да бъдат имплементирани чрез различни програмни езици, бази данни, хардуерна и софтуерна среда, в зависимост от това какво пасва най-добре за отделния проект. Услугите са малки по размер, разработват се автономно, разпространяват се независимо една от друга, дори на отделни физически или логически сървъри и се пускат чрез автоматизиран процес (fig. 2).

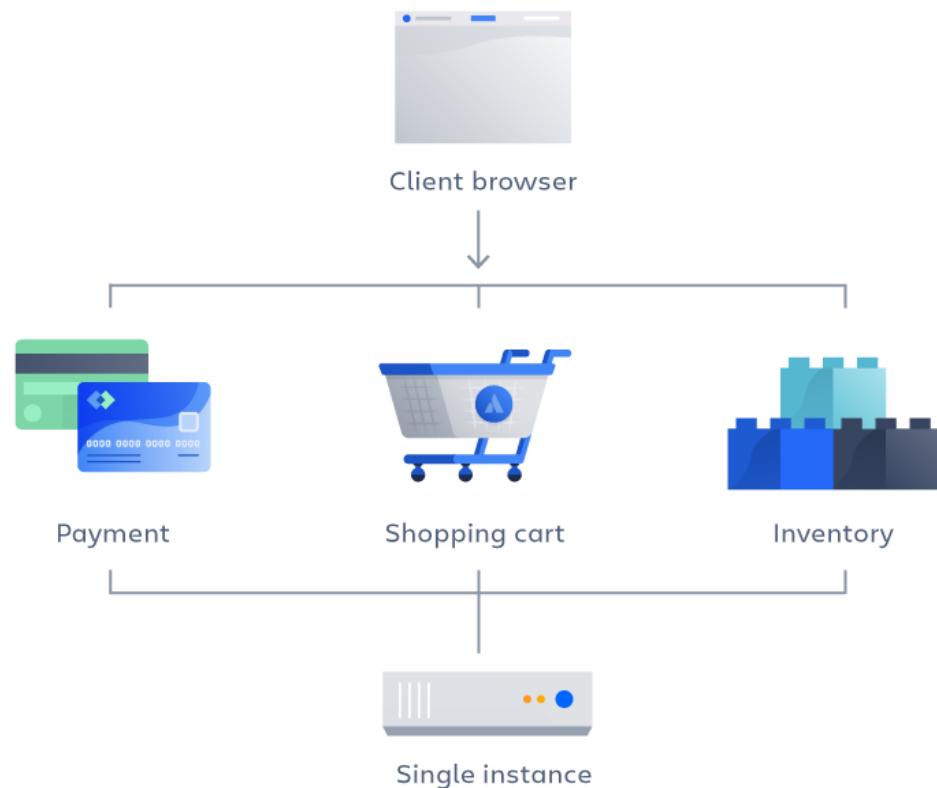
Въпреки неуспоримите си плусове, микросървисите могат да увеличат както сложността, така и времето за разработка на проекта, поради което решението уеб приложението да бъде разработено като монолитно, остава по-правилното към момента.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

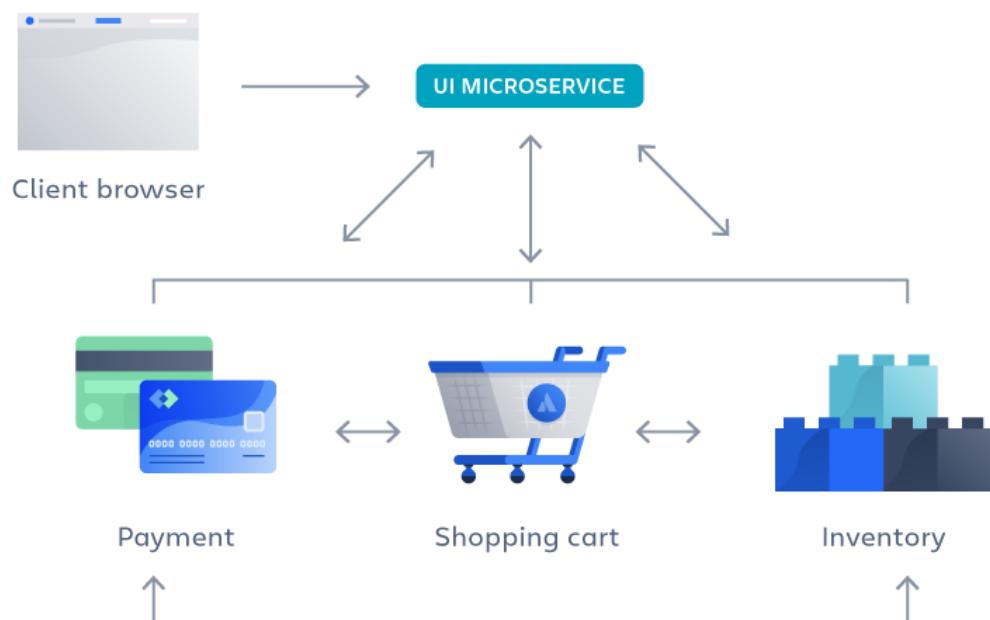
4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

Monolithic architecture



фиг. 1: Монолитна архитектура

Microservice architecture



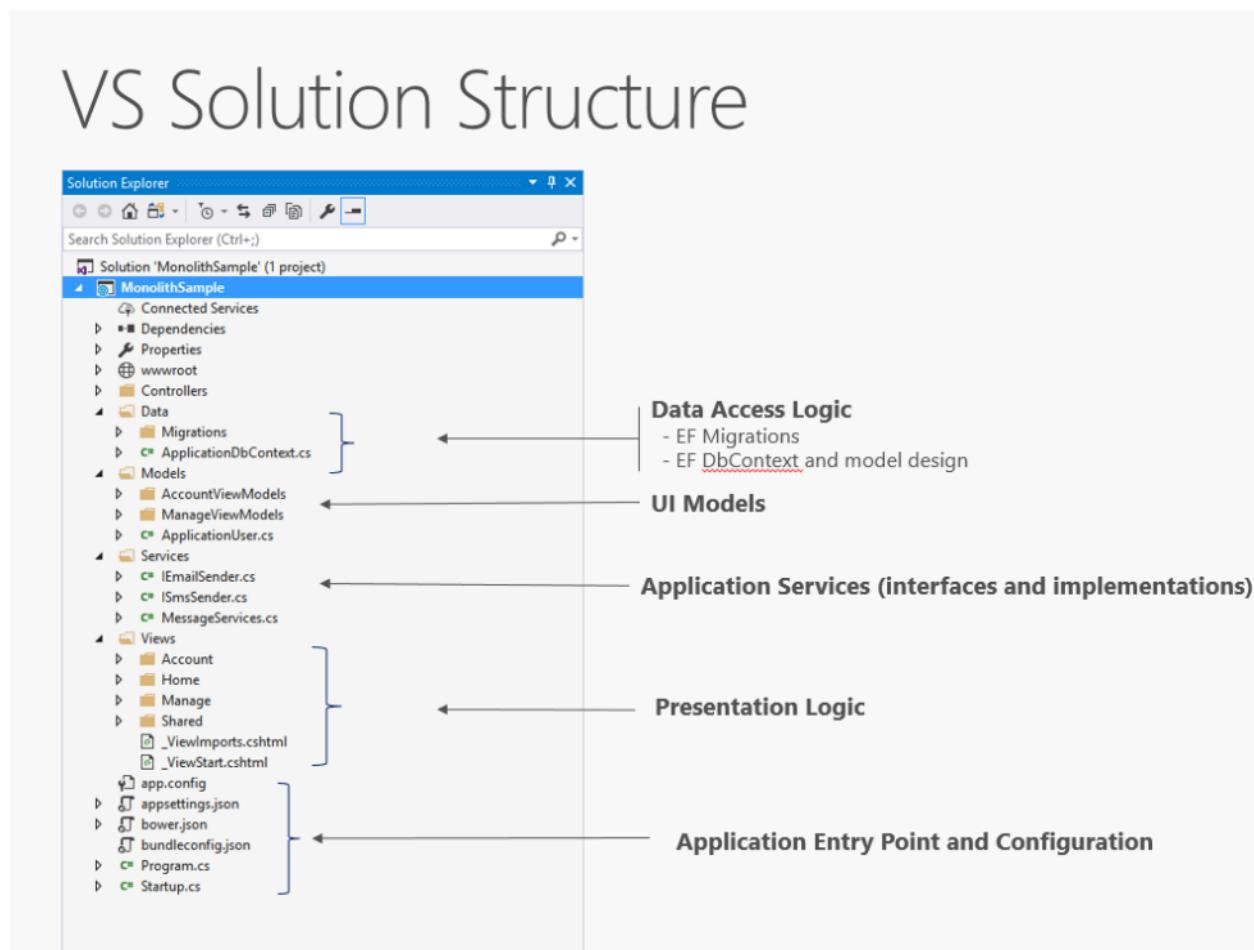
фиг. 2: Микросървисна архитектура



2.2. All-in-one MVC

При single-project сценария разпределянето на кода се постига чрез употребата на папки. По подразбиране MVC шаблонът включва отделни папки, отговарящи за Моделите, Изгледите и Контролерите, както и допълнителни папки "Data" и "Services". При тази подредба Изгледите трябва да бъдат ограничени до възможно най-голяма степен в папката Views, data access логиката да бъде в класовете, съхранявани в папката Data, а бизнес логиката трябва да се намира в класове в папката "Models" (фиг. 3).

Макар и опростено, монолитното single-project решение има някои недостатъци. С увеличаването на размера и сложността на проекта, броят на файловете и папките също ще продължи да расте, което може да доведе до така нареченият "спагети код". Под "спагети код" се разбира код, който е заплетен, особено такъв, в който е трудно да се проследи логиката на програмата. Както вече бе споменато, настоящият проект не е толкова голям и комплексен, че да се достигне до този проблем.



фиг. 3: Логическо разпределение на класовете по папки спрямо MVC шаблона



3. Използвани технологии

3.1. Microsoft Development Stack

Microsoft Development Stack е първият избор на голям брой компании, желаещи да разработят висококачествен софтуер, който да отговаря на изискванията както на клиентите, така и на предприемачите. MDS е съставен от пълен набор от инструменти и ресурси, които взаимодействват помежду си, за да предоставят стабилна и сигурна инфраструктура на бъдещите приложения. Това включва Microsoft SQL Server за управление на бази от данни, C#, F# и VB за програмиране на back-end, Blazor и Razor Pages за front-end, Visual Studio и Visual Studio Code за удобна и ефективна среда за разработка, както и Azure за хостинг и управление на завършеното приложение. Това означава, че всички компоненти, които се използват по време на изграждането на проекта, са създадени да работят един с друг. Например интеграцията на MSSQL Server в приложението е много по-лесна в сравнение с MySQL.

Може би най-голямото предимство на Microsoft Development Stack е свързано със сигурността и поддръжката на инструментите, езиците и приложението им. При внедряване на технологии, които не са на Microsoft, може да се срещнат проблеми с поддръжката и стабилността на отделните компоненти. Microsoft обаче е сериозна компания, която е изключително отговорна по отношение на качеството на продуктите, които предоставя, поддръжката на по-старите версии, пускането на редовни и стабилни актуализации и гарантирането на сигурност. И все пак, ако има проблем с даден компонент, Microsoft знае и работи върху възможно най-скорошното му разрешаване.

Единствен минус на MDS би бил фактът, че немалка част от услугите са платени, ако продуктът се използва за по-сериозни и сложни проекти, и че не всички технологии са open-source. В последните години обаче Microsoft предоставиха голям брой open-source технологии като .NET Core, Azure DevOps и бесплатни версии, предвидени за малки или пробни проекти, на платените продукти, като Visual Studio Community Edition.

3.2. .NET 6

.NET е бесплатна и с отворен код среда (open-source framework) на Microsoft за създаване на cross-platform софтуер на Windows, macOS и Linux. Това означава, че програмистите могат да разработват софтуер за различни операционни системи, като използват един и същи код, който може да бъде компилиран на различни операционни системи. На него може да се разработват всякакъв тип приложения – десктоп, уеб, мобилни, както и игри и IoT приложения (*фиг. 4*).

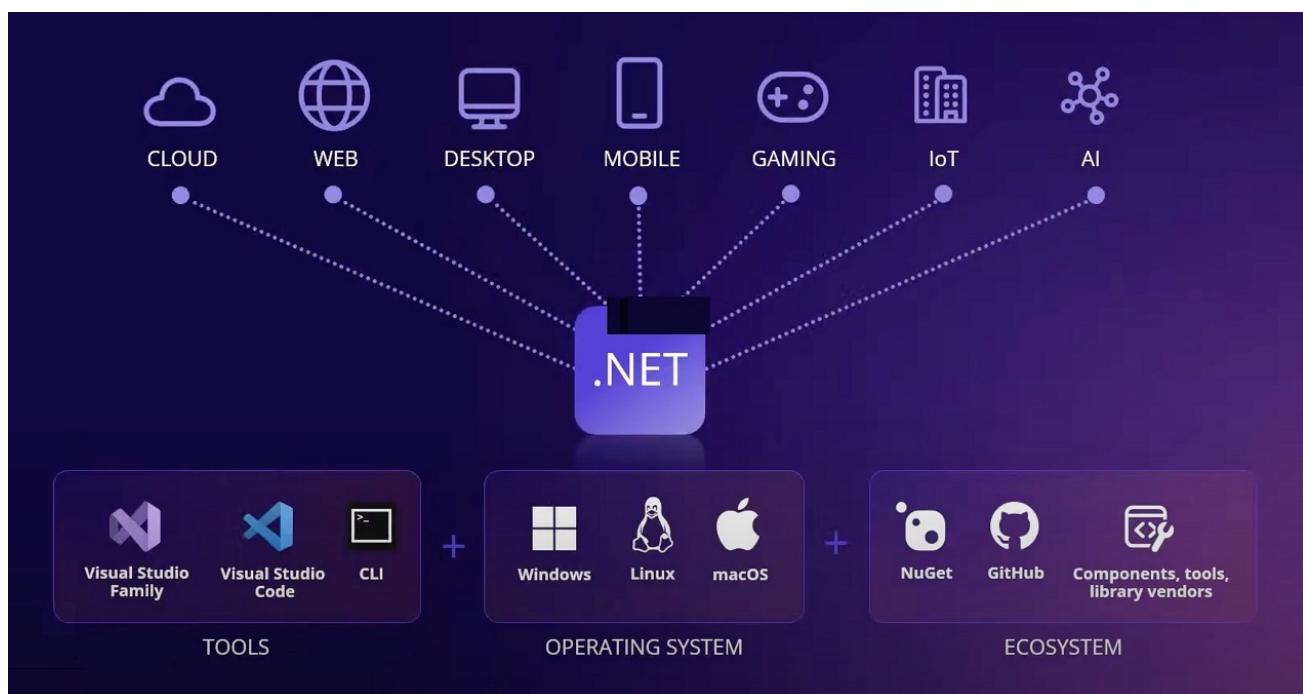


МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

Други предимства, които ще допринесат на работния процес и качеството на софтуера са:

- .NET е основан на обектно-ориентирано програмиране, което позволява на програмистите да създават повторно използваем и лесно поддържаем код.
- Поддържа множество езици за програмиране, включително C#, VB.NET, F#, IronPython, IronRuby и други. Това позволява на програмистите да изберат език, който най-добре отговаря на техните нужди и да го използват за разработката на софтуер.
- Лесна интеграция с други технологии, като например бази от данни, уеб услуги и други.
- Безопасност – .NET предлага вградена защита на приложението, която помага да се избегнат атаки и злоупотреби с приложението.



фиг. 4: Обобщение на .NET

3.3. Visual Studio 2022 Community Edition

Visual Studio 2022 Community е бесплатната версия на средата за разработка (Integrated Development Environment или IDE) Visual Studio, която може да се изтегли и инсталира от официалния сайт на Microsoft. Тази версия е подходяща точно за проекти като настоящия, защото е предназначена за индивидуални разработчици, студенти и малки екипи, които търсят среда за разработка на софтуер, която да им помогне да създават бързо и ефективно качествен код. Поддържа голям брой езици за програмиране, като C#, Visual Basic, F#, C++ и други, и има вградена поддръжка на множество технологии и платформи,



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

включително .NET, Xamarin и Unity. Visual Studio 2022 Community включва всички основни функции за създаване на софтуер, като например:

- редактор на код с функционалности като синтаксис highlighting, Intellisense и други;
- инструменти за дебъгване;
- вграден поддръжка на Git и други системи за контрол на версии;
- интеграция с Azure и други облачни услуги;
- множество плъгини и допълнения, като Nuget Packages, които разширяват функционалността на средата за разработка.

Все пак, при нужда от допълнителни функции, Microsoft предлага платени версии на Visual Studio с по-нататъшни функционалности и възможности, като Visual Studio Professional и Visual Studio Enterprise (*фиг. 5*).

Supported Features	Visual Studio Community	Visual Studio Professional	Visual Studio Enterprise
	Free download	Buy	Buy
⊕ Supported Usage Scenarios	●●●○	●●●●	●●●●
Development Platform Support ²	●●●●	●●●●	●●●●
⊕ Integrated Development Environment	●●●○	●●●○	●●●●
⊕ Advanced Debugging and Diagnostics	●●○○	●●○○	●●●●
⊕ Testing Tools	●○○○	●○○○	●●●●
⊕ Cross-platform Development	●●○○	●●○○	●●●●
⊕ Collaboration Tools and Features	●●●●	●●●●	●●●●

●●●● ●●●○ ●●○○ ●○○○

All features available Most features available Some features available Few features available

фиг. 5: Различни версии на Visual Studio

3.4. C#

C# е обектно-ориентиран език за програмиране, разработен от Microsoft. Той е един от основните езици за програмиране в .NET Core платформата, която ще се използва за разработката на учеб приложението. C# е проектиран да бъде безопасен, стабилен и лесен за използване, като същевременно предоставя широки възможности за разработка на приложения. Близък до езиците C++ и Java, но е по-modерен и синтактично по-лесен за употреба. C# използва CLR (Common Language Runtime), докато Java използва JRE (Java Runtime Environment). Друго важно подобреие към езика е така нареченият Garbage Collector, който позволява

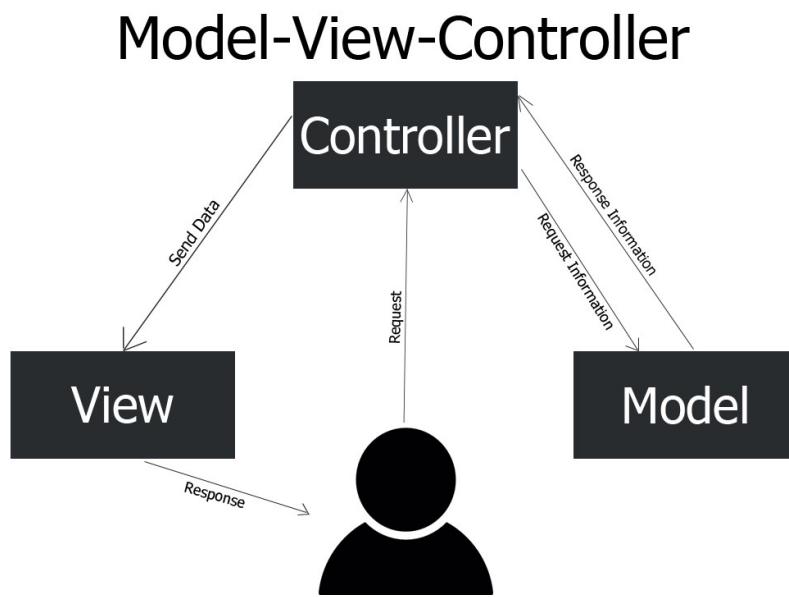


автоматично да се освобождава паметта, когато животът на дадена променлива изтече. Това има следните предимства:

- значително улеснява програмиста, за разлика от езици като C++, където той е длъжен да се грижи за освобождаването на ресурси;
- позволява по-ефективно разпределение и работа с heap паметта;
- освобождава паметта, заделена за обекти, които вече не се използват;
- гарантира, че даден обект не може неволно да пренаписва паметта, заделена за друг обект.

3.5. ASP.NET Core Web App (Model-View-Controller)

ASP.NET Core Web App MVC е гъвкав фреймуърк за уеб разработка, базиран на архитектурния шаблон Model-View-Controller (MVC), при който уеб приложението се разделя на три основни компонента: Модел, Изглед и Контролер. Моделите са класове, отговорни за обработката на данните и осигуряването на бизнес логиката на уеб приложението. Изгледите са компоненти, които се използват за представяне на информацията на потребителите (фиг. 6). Те са написани на често на HTML, CSS и JavaScript код, който се изпраща към браузър, като получават информацията, която трябва да се покаже на потребителите от моделите. Контролерите са компоненти, които са отговорни за обработката на заявките на потребителите и взаимодействието между Моделите и Изгледите.



фиг. 6: Логическата връзка между трите основни компонента на MVC

С ASP.NET Core Web App, всеки компонент на архитектурния шаблон MVC е реализиран чрез специални класове и интерфейси. Например, Контролерите се изграждат на базата на класовете ControllerBase и Controller, а Изгледите се

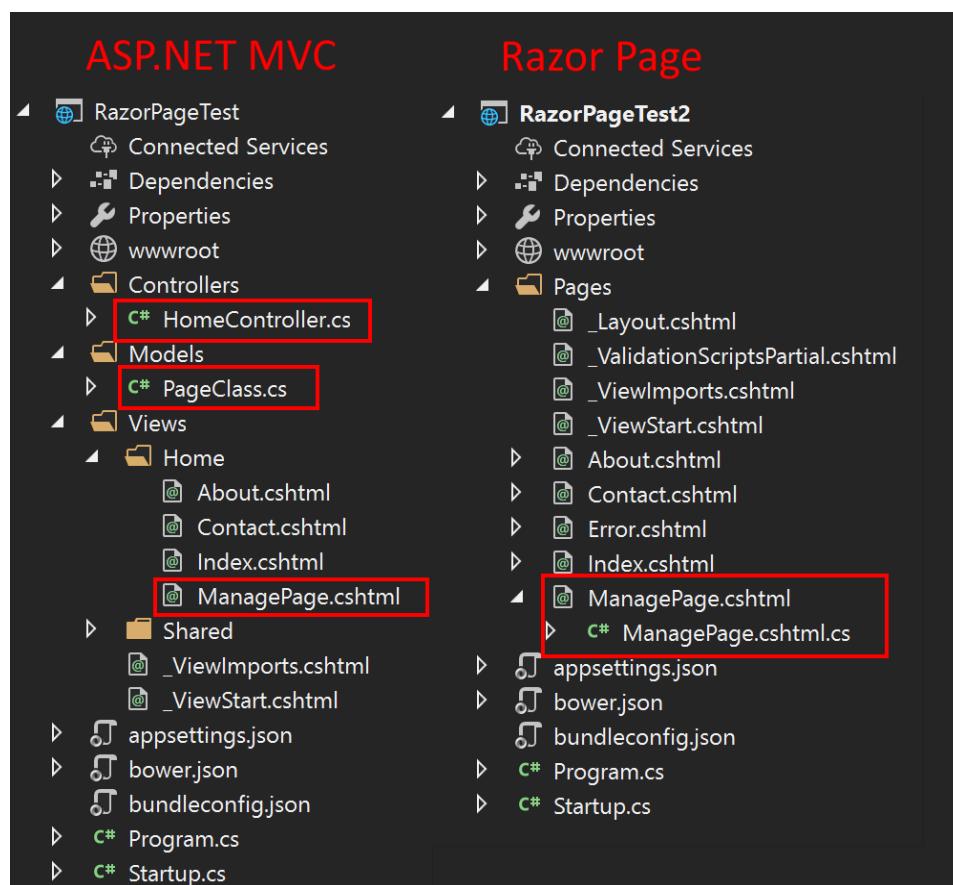


МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

изграждат чрез използването на Razor view engine. Благодарение на това, проектът ще бъде както с лесно управляема логика, така с ефективна и стабилна структура.

Razor Page наподобява на ASP.NET MVC view-component-ите, синтаксисът и функционалността му са същите като на MVC. Основната разлика между Razor pages и MVC е, че кода на Action-а и ViewModel-а е обединен в една самостоятелна Razor страница, а при MVC се пише поотделно (фиг. 7). Razor Pages е най-подходящо за page-focused проекти (проекти с голям брой страници и ViewModel-и). Тъй като настоящото приложение няма да използва прекалено много Изгледи, ASP.NET Core MVC е подходящ избор. Също така в MVC приложенията на .NET Core вече може безпроблемно да се комбинира употреба на Razor страници с традиционните Модели, Контролери и Изгледи. Така към вече познатите папки за логическо разпределение на кода Models, Controllers и Views се появява и папка Pages. Това ще бъде приложено на практика при употребата на Razor Class библиотека Identity.



фиг. 7: Разликите между Razor Page и ASP.NET MVC

3.6. Microsoft SQL Server

Microsoft SQL Server е релационна база данни, която е разработена от Microsoft. Тя е един от най-известните и използвани релационни управляващи системи за бази данни на пазара. Позволява на разработчиците да създават и



управляват големи количества данни, които могат да бъдат използвани за анализи, бизнес интелигентност, управление на транзакции, уеб приложения и други. SQL Server е съвместим с множество програмни езици и технологии, включително C# и .NET, което го прави лесен за използване и интеграция в настоящето уеб приложение. Освен това, той може да бъде използван както в локални, така и в облачни среди, като Microsoft Azure. Предвижда се проектът да използва именно локална база данни.

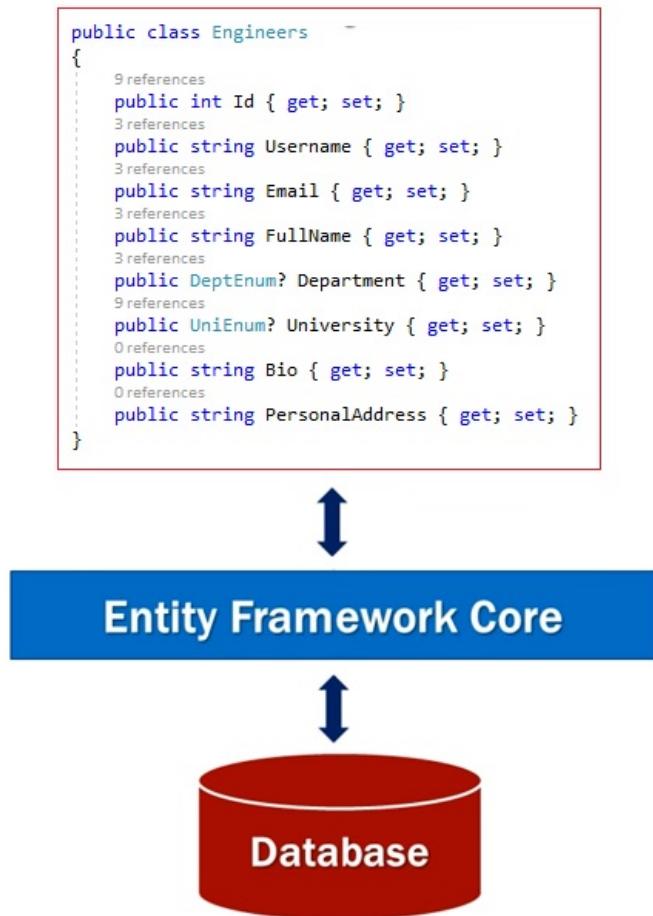
3.7. SQL Server Management Studio

SQL Server Management Studio (SSMS) е софтуер, необходим за управлението на базата данни на SQL Server. Това се осъществява чрез използването на графичен интерфейс. SSMS се използва за конфигуриране, управление и администриране на всички компоненти в SQL Server. То съчетава широка група от графични инструменти с голям брой текстови редактори, осигуряващи на разработчиците и администраторите всички нива на достъп до сървъра. Водещ елемент в SSMS е Object Explorer, който позволява на потребителя да търси, избира и да работи с всеки от обектите на сървъра. Подобно на Visual Studio има бесплатна и платена версия в зависимост от нуждите на потребителя и размера на проекта. В контекста на този проект бесплатната версия е напълно достатъчна. Сред функциите на SSMS са:

- създаване и управление на бази данни и таблици;
- изпълнение на SQL заявки;
- изпълнение на задачи като резервно копиране и възстановяване на бази данни;
- интегриране с други инструменти на Microsoft като Visual Studio и Excel.

3.8. Entity Framework Core

Entity Framework Core е ORM технология на Microsoft, която ще улесни използването на релационни бази данни, защото с нейна помощ създаването на базите данни и необходимите заявки ще се извършва автоматично, без да се нарушава обектно-ориентираната парадигма. EF Core работи като абстрактен слой над базата данни, който позволява на разработчиците да работят с нея чрез обекти и класове в .NET, вместо да използват SQL заявки директно (*фиг. 8*). Класовете могат да имат свойства, които съответстват на колони в таблиците, както и методи, които могат да извършват операции върху тези данни. Данните се извличат под формата на LINQ заявки. Всичко това ще улесни много работата върху проекта, защото ще позволи фокусът да е върху бизнес логиката на приложението, вместо да се отделя време за работа от по-ниско ниво с базата данни.



фиг. 8: Връзката, която осъществява EF Core между класовете и БД

3.9. Identity (on ASP.NET Core)

ASP.NET Core Identity е библиотека, която предоставя система за управление на потребителски акаунти в уеб приложения, базирани на ASP.NET Core, която е различна от Microsoft Identity Platform. Тя се основава на Entity Framework Core и използва по подразбиране релационната база данни Microsoft SQL Server за съхранение на потребителските данни. Identity предоставя функционалности за автентикация, авторизация и управление на потребителски акаунти и роли, като например регистрация, потвърждение на електронната поща, възстановяване на забравена парола.

Осъществяването на едни от основните цели на проекта ще бъдат осъществени именно чрез изброените функционалности. С помощта на Identity, създаването на сигурна и лесна за използване потребителска система ще бъде възможно най-безпроблемно и бързо, защото ще се избегне работата с ниско ниво детайли като криптиране на пароли и управление на сесии.



Identity също така предоставя вградена поддръжка за външни доставчици на идентичност, като например Google, Facebook и Microsoft. Това означава, че потребителите могат да използват своите акаунти в социалните мрежи, за да се автентицират в уеб приложението, което е удобна функционалност, която ще е хубаво да се добави в бъдеще.

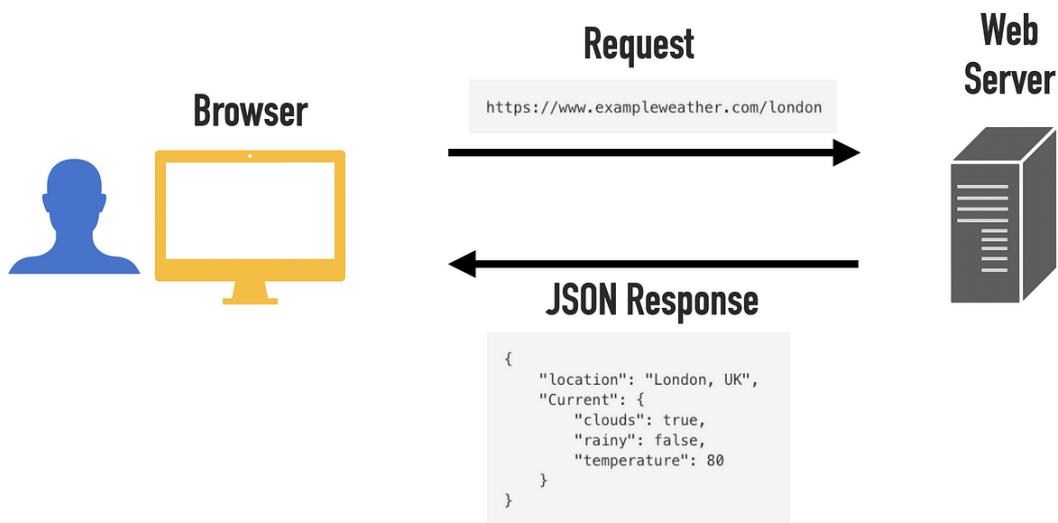
3.10. .Net Mail

.Net Mail е библиотека, част от .NET Framework и .NET Core, която предоставя класове за изпращане на електронна поща. Тя ще позволи на уеб приложението да изпраща електронни съобщения към потребителите му по електронната поща. Библиотеката предлага класове за създаване на SMTP сесия за изпращане на имейли, както и за обработка на получените съобщения. Ключовите класове в тази библиотека са MailMessage, SmtpClient и SmtpDeliveryMethod, функционалностите на които ще бъдат разгледани по-надолу в документацията.

3.11. JSON

JSON (JavaScript Object Notation) е опростен формат за обмен на данни, удобен за работа както за хората, така и за компютрите. Той е базиран на едно подмножество на езика за програмиране JavaScript, Standard ECMA-262 3rd Edition – от декември 1999. Използва се в много приложения за изпращане и получаване на данни между клиенти и сървъри. Има много добра поддръжка от много езици за програмиране, включително C#. JSON се състои от две основни структури: обекти и масиви. Обектите са списък от имена на полета и техните стойности, като стойностите могат да бъдат от различен тип, като числа, низове, булеви стойности, масиви или други обекти. Масивите са списък от стойности, като те също могат да бъдат от различен тип (*фиг. 9*).

Едно от основните предимства на JSON е неговият чист и лек формат. Той е добра алтернатива на XML, който е с по-сложен и тежък формат. Освен това, JSON поддържа гъвкавост на различни типове данни, което и го прави по-предпочитания формат за настоящето уеб приложение



фиг. 9: JSON обмен на данни

3.12. Ajax

AJAX (Asynchronous JavaScript and XML) е технология за уеб разработка, която позволява на уеб страниците да изпращат и получават данни от сървъра, без да бъде необходимо да се презарежда цялата страница. AJAX използва JavaScript и HTTP заявки, за да изпраща и получава данни асинхронно, и динамично обновява съдържанието на страницата. Предвид целите на проекта технологията ще бъде изключително полезна, защото ще подобри значително потребителския опит и ще ускори производителността на уеб приложението.

3.13. Braintree

Braintree е една от водещите платежни платформи, която предлага услуги за онлайн плащания. Тя е създадена, за да улесни интеграцията на различни методи на плащане в уеб приложения, като кредитни и дебитни карти, PayPal, Venmo и други (фиг. 10). Braintree е наличен за 45 страни, включително България, и обменя над 130 вида валути. Силно наподобява на друга известна платформа – Stripe. Stripe таксува 2.9% от транзакцията плюс 30 цента на превод, докато Braintree таксува 2.59% плюс 49 цента на превод, което е едно от предимствата, поради което е избран за настоящия проект, заедно с това, че Braintree е известен като по-удобен за по-малки приложения.

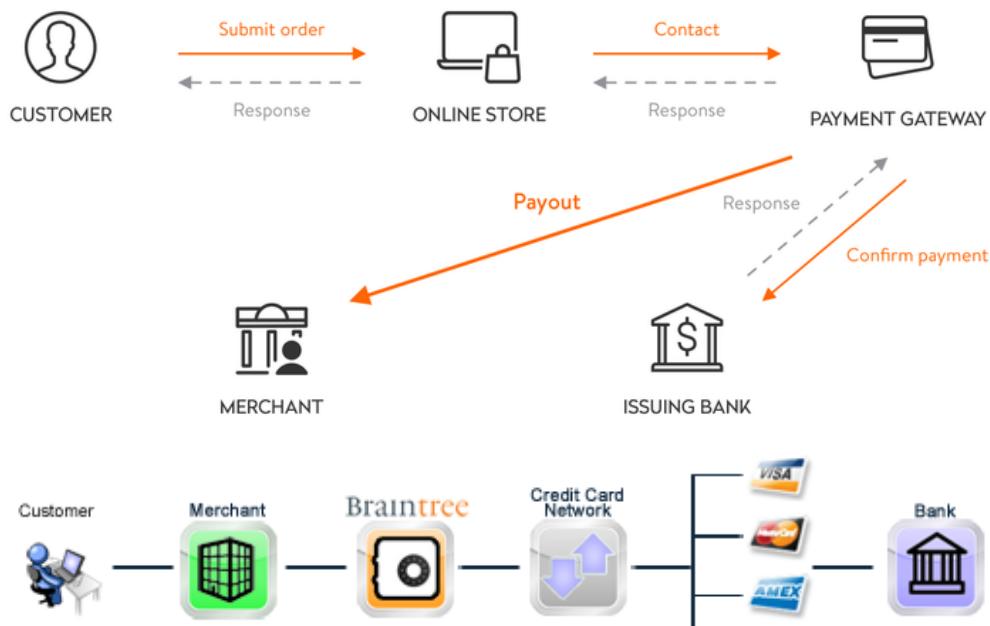
В ASP.NET може да се използва Braintree SDK за .NET, който включва библиотеки за C# и VB.NET. След като се инсталира SDK, ще може да се генерират



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

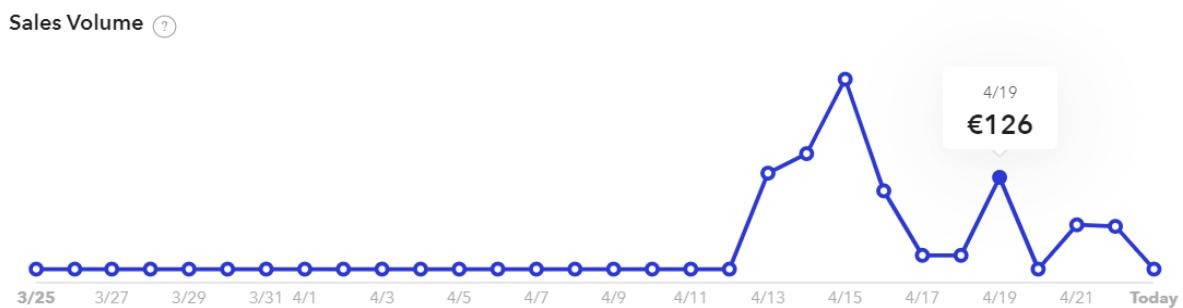
токени за плащане, да се създават транзакции и да се управляват плащанията в настоящия проект.



фиг. 10: Процесът, който осъществяват платежните платформи

3.14. Braintree Sandbox

Braintree Sandbox е среда за тестване на интеграцията с Braintree, която предоставя възможност за създаване на тестови транзакции и използване на тестови кредитни карти, като Mastercard, Visa, Discover, American Express и други (фиг. 11). Тя е безопасна и безплатна, идентична е на продукционната среда на Braintree, като включва всички функционалности, налични и в продукционната среда. Braintree Sandbox е полезна, за да се гарантира, че интеграцията работи коректно преди да се премине към продукционната среда. Това ще помогне да се избегнат грешки и нежелани последици, като губа на данни или финансови загуби, които биха могли да възникнат при неправилна интеграция.

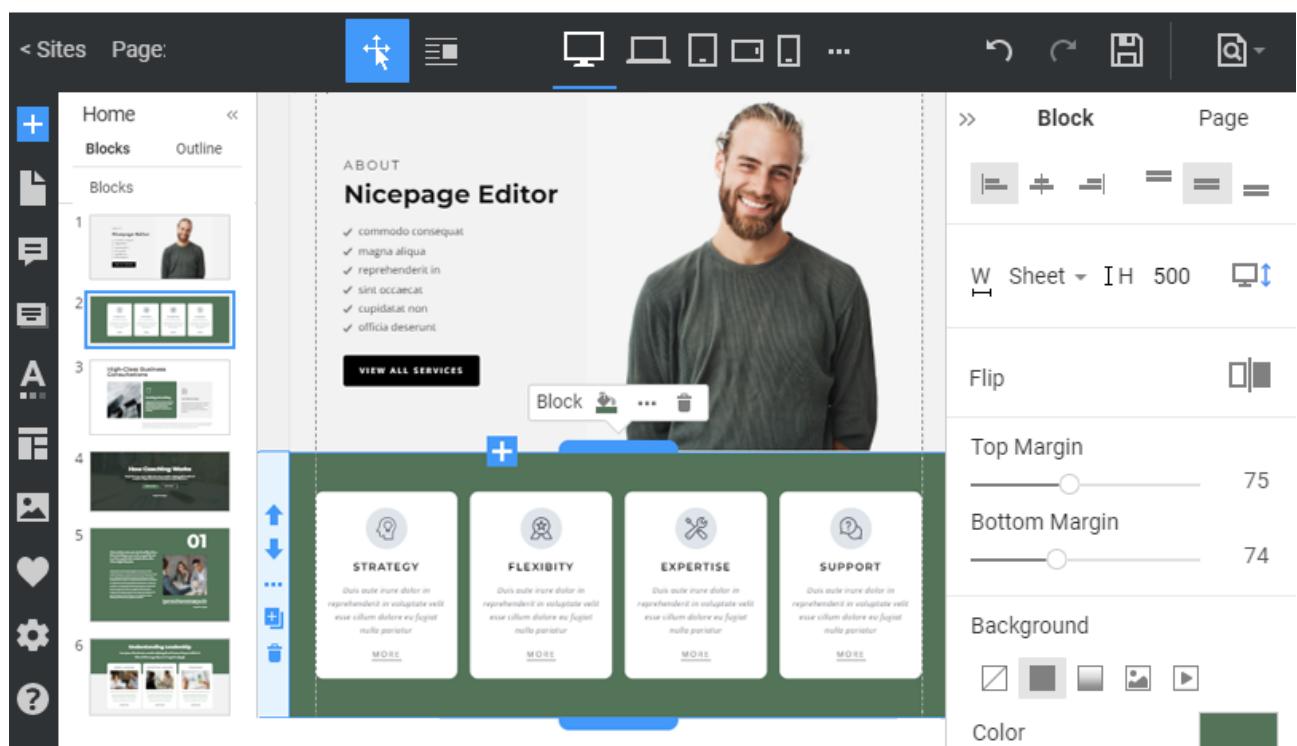


фиг. 11: Получените от тестовите транзакции суми в тестовата среда



3.15. Nicepage

Nicepage е уеб дизайн инструмент и платформа, предназначен за уеб дизайнери, малки и средни предприятия, и всеки, който желае да създаде качествени уеб страници без необходимостта от сложно програмиране. Това е WYSIWYG (What You See Is What You Get) редактор, който предоставя лесна и интуитивна визуална среда за създаване на уеб страници. С помощта на Nicepage, потребителите могат да избират от голям брой предварително направени дизайнерски блокове и шаблони, които могат да се персонализират и променят според техните нужди. Инструментът предлага различни функционалности като редакция на текст, вмъкване на изображения, създаване на менюта, добавяне на мултимедийно съдържание и други (фиг. 12). Nicepage поддържа адаптивен дизайн, който позволява създаването на уеб страници, които се приспособяват към различни устройства и размери на екраните.

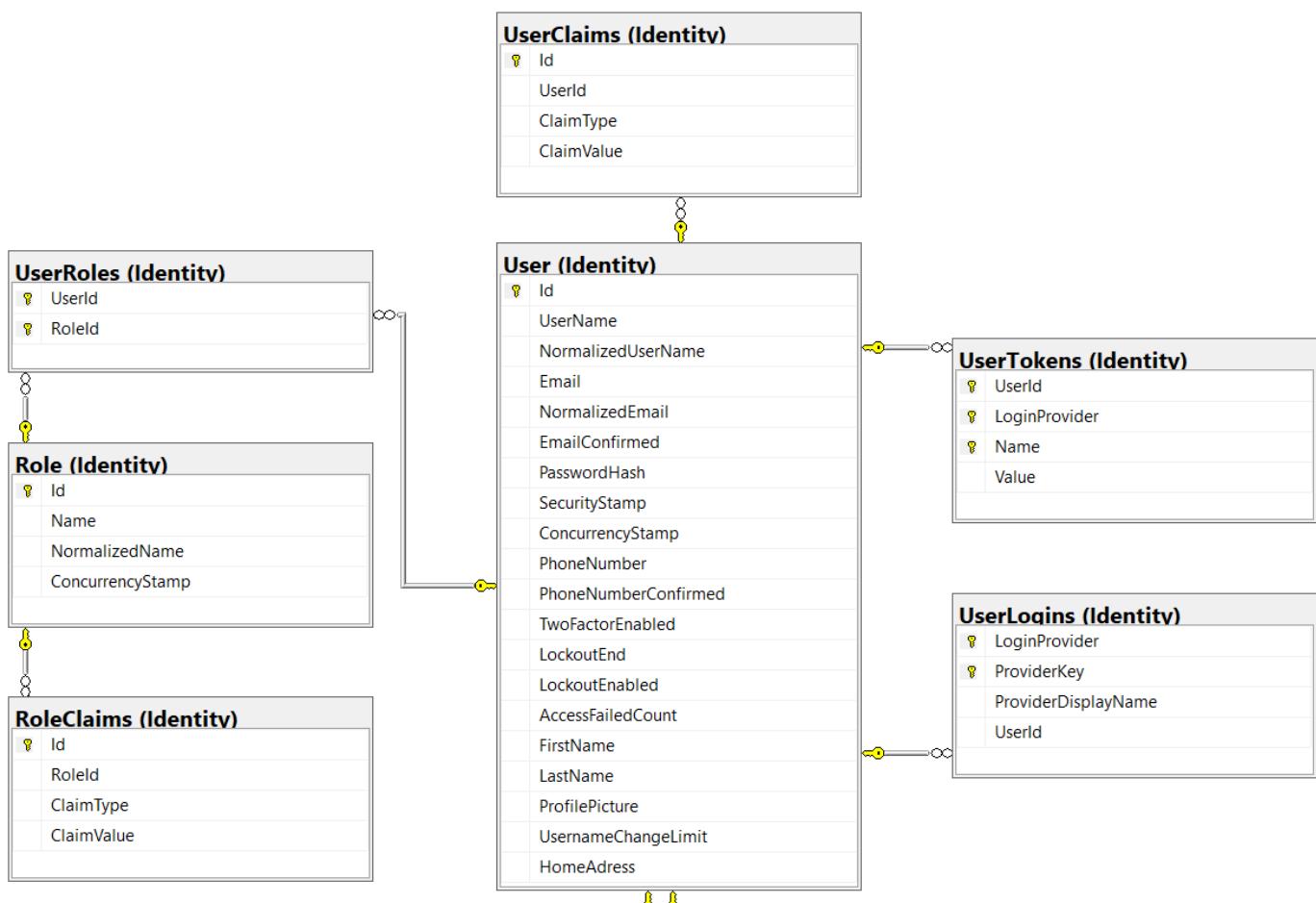


фиг. 12: Средата за разработка на уеб страници Nicepage



4. База данни

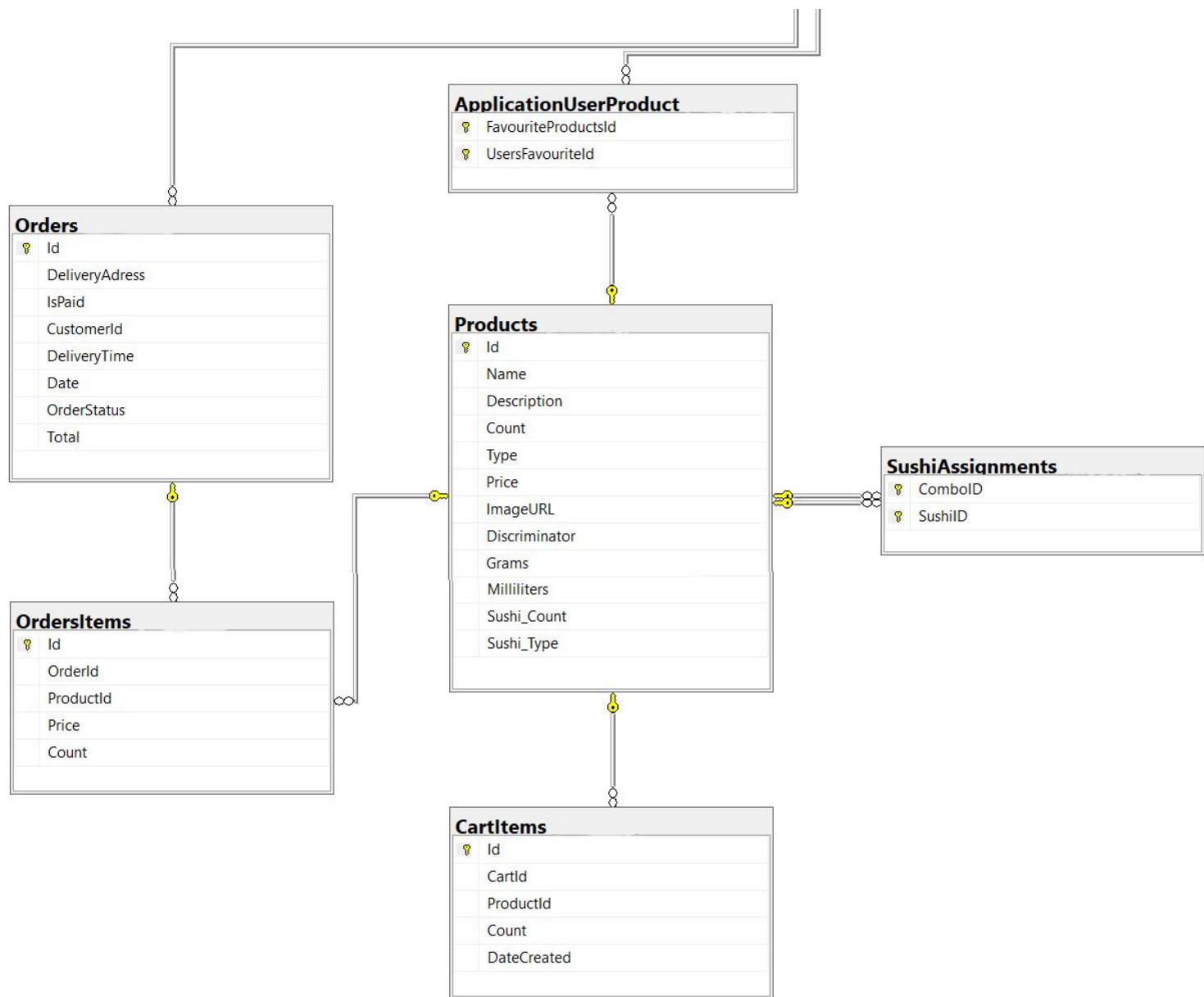
Както вече беше уточнено, настоящият проект използва релационната база данни Microsoft SQL Server за съхранение на данните. За да се улесни работата с нея, проектът използва Entity Framework Core заедно с подхода Code First, където първо се създават моделите на данните като класове, а след това автоматично се генерират съответстващи на таблиците в базата данни. С помощта на атрибути и индекси се задават допълнителни настройки и изисквания за моделите. Схемата на базата данни след автоматичното ѝ генериране изглежда така (fig. 13):





МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com



фиг. 13: Връзките между таблиците в базата данни



4.1. ApplicationDbContext

Класът ApplicationDbContext е клас за контекст на базата данни, който наследява класа IdentityDbContext<ApplicationUser>. Този клас се използва за управление на връзката с базата данни и заявките към нея. Конструкторът на класа приема параметър от тип DbContextOptions<ApplicationContext>, който се използва за конфигуриране на контекста на базата данни. Класът съдържа няколко DbSet свойства, които представят таблиците в базата данни. В този случай има DbSet свойства за Product, Sushi, Beverage, Dessert, Combo, SushiAssignmentViewModel, CartItem, Order и OrdersItem.

4.2. ContextSeed

Класът ContextSeed представлява статичен клас, който се използва за създаване на началните данни в базата данни (seeding) при стартиране на приложението и съдържа два метода – SeedRolesAsync и SeedSuperAdminAsync (*фиг. 14*).

```
1 reference
public static async Task SeedRolesAsync(UserManager< ApplicationUser > userManager, RoleManager< IdentityRole > roleManager)
{
    //Seed Roles
    await roleManager.CreateAsync(new IdentityRole(Enum.Roles.SuperAdmin.ToString()));
    await roleManager.CreateAsync(new IdentityRole(Enum.Roles.Admin.ToString()));
    await roleManager.CreateAsync(new IdentityRole(Enum.Roles.Moderator.ToString()));
    await roleManager.CreateAsync(new IdentityRole(Enum.Roles.Basic.ToString()));
}

1 reference
public static async Task SeedSuperAdminAsync(UserManager< ApplicationUser > userManager, RoleManager< IdentityRole > roleManager)
{
    //Seed Default User
    var defaultUser = new ApplicationUser
    {
        UserName = "superadmin",
        Email = "superadmin@gmail.com",
        FirstName = "Monika",
        LastName = "Blazheva",
        EmailConfirmed = true,
        PhoneNumberConfirmed = true
    };
    if (userManager.Users.All(u => u.Id != defaultUser.Id))
    {
        var user = await userManager.FindByEmailAsync(defaultUser.Email);
        if (user == null)
        {
            await userManager.CreateAsync(defaultUser, "123Pa$$word.");
            await userManager.AddToRoleAsync(defaultUser, Enum.Roles.Basic.ToString());
            await userManager.AddToRoleAsync(defaultUser, Enum.Roles.Moderator.ToString());
            await userManager.AddToRoleAsync(defaultUser, Enum.Roles.Admin.ToString());
            await userManager.AddToRoleAsync(defaultUser, Enum.Roles.SuperAdmin.ToString());
        }
    }
}
```

фиг. 14: Класът ContextSeed



Методът SeedRolesAsync служи за създаване на ролите в приложението и приема два параметъра – userManager и roleManager, които са отговорни за управлението на потребителите и ролите в приложението. Чрез метода се seed-ват четири роли: "SuperAdmin", "Admin", "Moderator" и "Basic", които си взимат от енумерацията Roles.

Методът SeedSuperAdminAsync се използва за създаването на Супер Администратора в приложението и приема два параметъра – userManager и roleManager. В метода се задават стойности за потребителското име, имейл адрес, име, фамилия, телефон и дали имейлът му е потвърден. Ако супер администраторът все още не е създаден, се извършва създаването му. Задават му се и четирите роли, създадени преди това с метода SeedRolesAsync.

4.3. Моделът ApplicationUser

Класът ApplicationUser е потребителският клас, който наследява вградения клас IdentityUser от ASP.NET Identity Framework. Базовият клас IdentityUser предоставя свойства за безопасно съхранение на информация за потребителя, като имейл, парола, потребителско име и телефон. Добавени са допълнителни свойства към класа ApplicationUser, за съхранението на допълнителна информация, специфична за проекта, като профилна снимка и други (фиг. 15). Класът притежава следните свойства:

- FirstName – първото име на потребителя.
- LastName – фамилията на потребителя.
- UsernameChangeLimit = 10; – броя пъти, които остават на потребителя да си сменя потребителското име, максималния брой пъти е 10.
- ProfilePicture – профилната снимка на потребителя под формата на масив от байтове.
- CompletedOrders – колекция от поръчките на потребителя.
- FavouriteProducts – колекция от любимите продукти на потребителя

```
99+ references
public class ApplicationUser : IdentityUser
{
    8 references
    public string FirstName { get; set; }
    8 references
    public string LastName { get; set; }
    3 references
    public int UsernameChangeLimit { get; set; } = 10;
    4 references
    public byte[] ProfilePicture { get; set; }
    1 reference
    public ICollection<Order> CompletedOrders { get; set; }

    6 references
    public ICollection<Product> FavouriteProducts { get; set; }
}
```

фиг. 15: Свойствата на класа ApplicationUser



4.4. Моделът Product

Класът Product е абстрактен клас, което означава, че не може да бъде инстанциран. Той служи като базов клас за класовете Sushi, Combo, Dessert и Beverage. Класът съдържа следните свойства, които определят характеристиките на обектите от тип Product (*фиг. 16*):

- Id – първичен ключ.
- Name – името на продукта.
- Description – описанието на продукта.
- Price – цената на продукта, има атрибут [Range], който задава диапазона на валидни стойности да е от 0 до 1000. Също така има атрибут [DataType], който указва типа на данните, с който ще бъде свързано свойството, а [Column] указва типа на в базата данни, на който съответства това свойство.
- ImageURL – снимка продукта под формата на масив от байтове.

```
13 references
public abstract class Product
{
    [Key]
93 references
    public int Id { get; set; }

    [Required]
60 references
    public string Name { get; set; }

    [Required]
42 references
    public string Description { get; set; }

    [Range(0, 1000, ErrorMessage = "Price must be between 0 and 1000.")]
    [DataType(DataType.Currency)]
    [Column(TypeName = "decimal(18, 2)")]
52 references
    public decimal Price { get; set; }

    47 references
    public byte[] ImageURL { get; set; }

    0 references
    public ICollection<ApplicationUser> UsersFavourite { get; set; }
}
```

фиг. 16: Свойствата на класа Product

4.5. Моделът Sushi

Класът Sushi е наследник на класа Product, което означава, че всички свойства, дефинирани в класа Product, са достъпни и за класа Sushi. В допълнение към наследените свойства, класът Sushi има следните свойства (*фиг. 17*):

- Count – броя сушита, има атрибут [Range], който задава валидния диапазон да е от 1 до 100.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

- Type – типа на сушито. Това свойство се дефинира с енумерацията SushiType. Енумерацията SushiType определя пет възможни стойности: Uramaki, Maki, Sashimi, Nigiri и Temaki.
- SushiAssignments – колекция от обекти от типа SushiAssignmentViewModel, служи за свързване на Sushi с класа Combo.

```
22 references
public class Sushi : Product
{
    [Range(1, 100, ErrorMessage = "Count must be between 1 and 100.")]
    12 references
    public int Count { get; set; }

    [Required]
    12 references
    public SushiType Type { get; set; }
    1 reference
    public ICollection<SushiAssignmentViewModel> SushiAssignments { get; set; }
}
3 references
public enum SushiType { Uramaki, Maki, Sashimi, Nigiri, Temaki }
```

фиг. 17: Свойствата на класа Sushi

4.6. Моделът Combo

Класът Combo (произнася се комбо, съкратено от комбинация) е наследник на класа Product, което означава, че всички свойства, дефинирани в класа Product, са достъпни и за класа Combo. Комботата и сушитата имат връзка много към много, защото всяко комбо е съставено от няколко типа сушита, а едно суши може да участва в различни комбота. В допълнение към наследените свойства, класът има следните свойства (фиг. 18):

- Count – броя сушитата в комбото, има атрибут [Range], който задава валидния диапазон да е от 0 до 1000.
- SushiAssignments – колекция от обекти от типа SushiAssignmentViewModel, осъществява връзката много към много.

```
24 references
public class Combo : Product
{
    [Range(0, 1000, ErrorMessage = "Count must be between 0 and 1000.")]
    12 references
    public int Count { get; set; }
    14 references
    public ICollection<SushiAssignmentViewModel> SushiAssignments { get; set; }
}
```

фиг. 18: Свойствата на класа Combo



4.7. Моделът **Dessert**

Класът Dessert също е наследник на класа Product, което означава, че всички свойства, дефинирани в класа Product, са достъпни и за класа Dessert, който в допълнение има единствено свойството Grams, което пази грамажа на десерта (*фиг. 19*).

```
19 references
public class Dessert : Product
{
    [Range(1, 3000, ErrorMessage = "Milliliters must be between 1 and 3000.")]
    12 references
    public int Grams { get; set; }
}
```

фиг. 19: Свойствата на класа Dessert

4.8. Моделът **Beverage**

Класът Beverage е последният наследник на класа Product и всички свойства, дефинирани в класа Product, са достъпни и за класа Beverage. В допълнение има единствено свойството Grams, което пази грамажа на десерта (*фиг. 20*).

```
19 references
public class Dessert : Product
{
    [Range(1, 3000, ErrorMessage = "Milliliters must be between 1 and 3000.")]
    12 references
    public int Grams { get; set; }
}
```

фиг. 20: Свойствата на класа Dessert

4.9. Моделът **CartItem**

Класът CartItem е клас, описващ предмет за нечия пазарска количка. Нека да обясним по-детайлно какви свойства има един такъв предмет (*фиг. 21*):

- Id – първичен ключ.
- CartId – идентификатор, посочващ на кой потребител е конкретният предмет.
- ProductId – външен ключ, сочещ към типа продукт, който се съдържа в този конкретен предмет.
- Product – това е препратка към продукта, свързан с този предмет, използва се за достъп до всички данни, свързани с продукта, като име, описание, цена и други.
- Count – броя на продуктите.
- DateCreated – датата, на която предмета е създаден и добавен в количката.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

- SubTotal() – това е метод, който връща общата стойност на предмета, като умножи цената на продукта по точния броя на продуктите в тази него. Не е колона в таблицата.

```
5 references
public class CartItem
{
    [Key]
    13 references
    public int Id { get; set; }

    [Required]
    11 references
    public string CartId { get; set; }

    [Required]
    [ForeignKey("Product")]
    2 references
    public int ProductId { get; set; }

    [Required]
    10 references
    public int Count { get; set; }

    [Required]
    1 reference
    public System.DateTime DateCreated { get; set; }

    [Required]
    17 references
    public Product Product { get; set; }
    3 references
    public decimal SubTotal()
    {
        return Product.Price * Count;
    }
}
```

фиг. 21: Свойствата на класа CartItem

4.10. Моделът Order

Класът Order представлява поръчка в един онлайн магазин или друг подобен тип приложения. Следващите редове обясняват по-подробно всяко от свойствата на този клас (фиг. 22):

- Id – първичен ключ.
- DeliveryAddress – адресът за доставка на поръчката.
- DeliveryTime – времето, в което се очаква да бъде доставена поръчката.
- Date – това е датата, на която поръчката е направена.
- Customer – препратка към потребителя, който е направил поръчката. Този обект може да бъде използван за достъп до всички данни, свързани с потребителя, като име, телефон му и имейл.
- IsPaid – булева стойност, която показва дали поръчката е заплатена или не.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

- OrderStatus – свойството е дефинирано с енумерацията OrdersStatus, която показва статуса на поръчката, като например "в процес на изпълнение", "потвърдена", "доставена" и други.
- Total – общата сума на поръчката.
- OrdersItems – колекция от обекти от тип OrdersItem, който съдържат информация за всеки един продукт, който е част от тази поръчка.

```
22 references
public class Order
{
    [Key]
    14 references
    public int Id { get; set; }

    [Required]
    6 references
    public string DeliveryAddress { get; set; }

    [Required]
    6 references
    public string DeliveryTime { get; set; }

    2 references
    public DateTime? Date { get; set; }

    15 references
    public ApplicationUser? Customer { get; set; }

    1 reference
    public bool? IsPaid { get; set; }

    5 references
    public OrdersStatus OrderStatus { get; set; }

    3 references
    public decimal Total { get; set; }

    5 references
    public ICollection<OrdersItem> OrdersItems { get; set; }
}
```

фиг. 22: Свойствата на класа Order

4.11. Моделът OrdersItem

Класът OrdersItem представлява един елемент от конкретна поръчка, който съдържа информация за определен продукт, заедно с цената му и количеството, което е закупено. Класът съдържа следните свойства (фиг. 23):

- Id – първичен ключ.
- Order – препратка към обект от тип Order, който представлява поръчката, към която този елемент принадлежи.
- Product – препратка към обект от тип Product, който представлява продукта, който е закупен.
- Count – количеството, закупено от този продукт.
- Price – цената на елемента (цената на продукта умножена по закупеното количество).



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

```
4 references
public class OrdersItem
{
    [Key]
    0 references
    public int Id { get; set; }

    [Required]
    1 reference
    public Order Order { get; set; }

    [Required]
    9 references
    public Product Product { get; set; }

    2 references
    public decimal Price { get; set; }

    3 references
    public int Count { get; set; }
}
```

фиг. 23: Свойствата на класа OrdersItem



5. Контролери

Контролерите са един от основните компоненти в шаблона MVC. Ще бъдат използвани за обработка на заявките на потребителите към уеб приложението и за връщането на подходящ отговор, най-често под формата на изглед, в други случаи JSON обект или редирект към друга страница.

5.1. Sushis, Combos, Desserts и Beverages контролери

Четирите контролера са идентични и обработват заявките, свързани с CRUD операциите, които може да извършва Админът. Притежават следните методи, които са достъпни само за потребители с роля “Супер Админ” или “Админ”:

- Index (GET) – извлича всички обекти от базата данни и ги предава на изгледа за визуализация;
- Details (GET) – извлича един конкретен обект от базата данни по Id и го предава на изгледа за визуализация;
- Create (GET) – създава нов обект и го предава на изгледа за създаване;
- Create (POST) – записва новия обект в базата данни, ако моделът на обекта е валиден;
- Edit (GET) извлича един конкретен обект от базата данни по Id и го предава на изгледа за редактиране;
- Edit (POST) – записва промените по обекта в базата данни, ако моделът на обекта е валиден;
- Delete (GET) – извлича един конкретен обект от базата данни по Id и го предава на изгледа за изтриване;
- Delete (POST) – изтрива конкретния обект от базата данни, ако съществува.

5.2. ProductsMenu контролер

Контролерът ProductsMenu отговаря за менюто на ресторана и има един метод с име Index (GET), който използва ApplicationDbContext, за да извлече списъци с продукти от базата данни "суши", "комбо", "десерти" и "напитки". Списъците се добавят към обект от тип ProductsMenuViewModel. Този обект се изпраща към Index изгледа, който го показва в уеб браузъра. Контролерът е достъпен за всеки, включително хората без регистрация.

5.3. Favourites контролер

Контролерът Favourites има два метода – Index и AddRemoveFavourite (фиг. 24). Контролерът е достъпен само за потребители с регистрация.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемшир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

Методът Index (GET) се използва за извлечане на любимите продукти на потребителя от базата данни и извеждането им в изгледа на приложението, като първо се намира потребителят, който е логнат в момента, чрез използването на UserManager.

```
[HttpPost]
0 references
public async Task<IActionResult> AddRemoveFavourite(int id)
{
    ClaimsPrincipal currentUser = this.User;
    var currentUserID = currentUser.FindFirst(ClaimTypes.NameIdentifier).Value;
    var user = await _userManager.Users.Include(u => u.FavouriteProducts).
        FirstOrDefaultAsync(u => u.Id == currentUserID);

    Product favouriteProduct = user.FavouriteProducts.FirstOrDefault(p => p.Id == id);
    if(favouriteProduct == null)
    {
        favouriteProduct = await _context.Products.FindAsync(id);
        if (favouriteProduct != null)
        {
            user.FavouriteProducts.Add(favouriteProduct);
            await _userManager.UpdateAsync(user);
            IsFav = true;
        }
    }
    else
    {
        user.FavouriteProducts.Remove(favouriteProduct);
        await _userManager.UpdateAsync(user);

        IsFav = false;
    }

    return Json(new { success = IsFav });
}
```

фиг. 24: Методът AddRemoveFavourite

Методът AddRemoveFavourite (POST) се използва за добавяне и премахване на продукт от списъка с любими продукти на потребител. Първо намира потребителят, който е логнат в момента, чрез UserManager, след това търси продуктът, който потребителят желае да добави или премахне от любими, проверявайки дали вече е добавен към списъка. Ако не е, се извлича от базата данни и се добавя към списъка. Ако вече е добавен, се премахва. Накрая се записва информацията в базата данни и се връща резултат под формата на JSON обект, който указва дали операцията е била успешна или не.



5.4. ShoppingCart контролер

Контролерът ShoppingCart е достъпен само за потребители с регистрация. Съдържа следните пет метода, чрез които обработва заявките, свързани с функционалностите на пазарската количка, с помощта на инстанция на помощния клас CartService (Dependency injection):

- Index (GET) – показва списъка с продукти в количката, подробностите към тях и общата им сума чрез ShoppingCartViewModel, който притежава две свойства: CartItems – представлява списък от продуктите, добавени в количката на потребителя, като за тях се извиква метода GetCartItems() на CartService инстанцията; Cart Total – съхранява общата сума на продуктите в количката на потребителя, която се изчислява чрез метода GetTotalToString() на CartService инстанцията.
- AddToCart (POST) (*фиг. 25*) – служи за добавяне на продукт към количката за пазаруване. Приема Id–то на предмета, който трябва да бъде добавен (CartItem) и го добавя чрез метода AddToCart() на CartService инстанцията.
- RemoveFromCart (POST) (*фиг. 26*) – премахва продукт от количката на потребителя по Id чрез метода RemoveFromCart() на CartService инстанцията. След това методът връща резултат във формат JSON, съдържащ съобщение за потвърждение, новата обща сума на количката, новия брой на продуктите и идификационния номер на елемента, който е бил премахнат. Този резултат се използва, за да се актуализира интерфейса на потребителя, без да се презарежда отново цялата страница.
- IncreaseCartItemCount (POST) (*фиг. 27*) – увеличава броя на конкретния продукт, намерен по даденото Id, с едно чрез метода IncreaseByOneCartItemCount() на CartService инстанцията. След това методът връща резултат във формат JSON, съдържащ актуализираното количество на продукта и новата обща сума на количката.
- DecreaseCartItemCount (POST) – намаля броя на конкретния продукт, намерен по даденото Id, с едно чрез метода DecreaseByOneCartItemCount() на CartService инстанцията. След това методът връща резултат във формат JSON, съдържащ актуализираното количество на продукта и новата обща сума на количката.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

```
public async Task<IActionResult> AddToCart(int? id)
{
    if (id == null || _context.Products == null)
    {
        return NotFound();
    }

    var product = await _context.Products.FirstOrDefaultAsync(m => m.Id == id);
    if (product == null)
    {
        return NotFound();
    }
    // Add it to the shopping cart
    var cart = ShoppingCart;

    cart.AddToCart(product);
    // Go back to the main store page for more shopping
    return RedirectToAction(nameof(Index));
}
```

фиг. 25: Методът AddToCart

```
public ActionResult RemoveFromCart(int cartItemId)
{
    var cart = ShoppingCart;
    var productName = _context.CartItems.Include(p => p.Product)
        .FirstOrDefault(p => p.Id == cartItemId).Product.Name;

    cart.RemoveFromCart(cartItemId);

    var results = new ShoppingCartRemoveViewModel
    {
        Message = WebUtility.HtmlEncode(productName) +
            " has been removed from your shopping cart.",
        CartTotal = cart.GetTotalToString(),
        CartCount = cart.GetCount(),
        DeleteId = cartItemId
    };
    return Json(results);
}
```

фиг. 26: Методът RemoveFromCart



```
public ActionResult IncreaseCartItemCount(int cartItemId)
{
    var cart = shoppingCart;
    int itemCount = cart.IncreaseByOneCartItemCount(cartItemId);
    var results = new ShoppingCartUpdateItemCountViewModel
    {
        ItemCount = itemCount,
        CartTotal = cart.GetTotalToString(),
        CartCount = cart.GetCount()
    };
    return Json(results);
}
```

фиг. 27: Методът IncreaseCartItemCount (DecreaseCartItem е аналогичен)

5.5. Orders контролер

Контролерът Orders е достъпен само за потребители с регистрация. Работи с помощта на сервисните класове CartService и UserManager и интерфейса IBrainTree (Dependency injection) и има следните методи:

- Index (GET) – връща списък с всички поръчки на логнатия в момента потребител.
- Details (GET) – връща подробната информация за конкретна поръчка по Id.
- Delete (POST) – изтрива конкретен обект по Id от базата данни, ако съществува.
- Payment (GET) (*фиг. 28*) – от инстанция на Gateway от Braintree генерира токен, който ще бъде използван от клиента за идентификация на плащането, и го предава на изгледа Payment, който съдържа форма за плащане.
- Payment (POST) (*фиг. 28*) е по-комплексен метод, който приема като параметри обект от тип IFormCollection, съдържащ токена за плащане, и обект от тип Order. Първо създава поръчка с въведените от потребителя в обекта адрес и час за доставка. След това прави заявка за Braintree транзакция, която заявка има следните свойства: общата сума, която трябва да бъде преведена (сумата на всички продукти от кошницата); уникалния токен на клиента за идентификация на плащането; идентификационния номер на поръчката от базата данни. Ако транзакцията е успешна, поръчката се отбележва като платена в базата данни, статусът ѝ се променя на “Потвърдена” и се извиква метода на помощния клас CartService CreateOrder(). Този метод попълва списъка от продукти на поръчката и изпразва кошницата на потребителя. Ако транзакцията обаче е неуспешна, статуса на поръчката се променя на “Отказана” и уеб страницата се презарежда за повторен опит.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

```
public IActionResult Payment()
{
    var gateway = _brainTree.GetGateway();
    var clientToken = gateway.ClientToken.Generate();

    ViewBag.ClientToken = clientToken;
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Payment(IFormCollection formCollection,
                                            [Bind("Id,DeliveryAdress,DeliveryTime")] Order order)
{
    if (ModelState.IsValid)
    {
        order.Customer = this.GetCustomer();
        order.Date = DateTime.Now;

        _context.Orders.Add(order);
        await _context.SaveChangesAsync();

        var total = shoppingCart.GetTotal();
        string nonceFromTheClient = formCollection["payment_method_nonce"];
        var request = new TransactionRequest
        {
            Amount = total,
            PaymentMethodNonce = nonceFromTheClient,
            OrderId = order.Id.ToString(),
            Options = new TransactionOptionsRequest
            {
                SubmitForSettlement = true
            }
        };

        var gateway = _brainTree.GetGateway();
        Result<Transaction> result = gateway.Transaction.Sale(request);

        if (result.Target.ProcessorResponseText == "Approved")
        {
            order.IsPaid = true;
            order.OrderStatus = OrdersStatus.Confirmed;

            shoppingCart.CreateOrder(order);

            _context.Update(order);
            await _context.SaveChangesAsync();

            return RedirectToAction("Details", new { id = order.Id });
        }
        else
        {
            order.OrderStatus = OrdersStatus.Cancelled;
            TempData["Unsuccessful"] = "Transaction was unsuccessful. Please try again";
            return RedirectToAction("Payment", new { orderId = order.Id });
        }
    }
    return NotFound();
}
```

фиг. 28 GET и POST методите Payment



5.6. ToDoList контролер

Контролерът ToDoList е предназначен за употреба от служителите на онлайн ресторанта, тоест потребителите с роля “Moderator”. Той притежава метод Index, който извлича всички поръчки със статус “Потвърдена” от базата данни и ги предава на изгледа за визуализация, и метод UpdateStatus (*фиг. 29*), който служи за промяната на статуса на поръчката от “Потвърдена” на “Изпратена”. UpdateStatus се извиква от изгледа чрез AJAX заявка, когато служителя означи поръчката, като предадена за доставка. По този начин потребителите с направена поръчка могат да следят статуса ѝ и в кой етап на изпълнение е тяхната поръчка.

```
0 references
public async Task<IActionResult> UpdateStatus(int? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var orderFromDb = await _context.Orders.FirstOrDefaultAsync(o => o.Id == id);
    if(orderFromDb == null)
    {
        return NotFound();
    }

    orderFromDb.OrderStatus = Enums.OrdersStatus.Shipped;
    _context.Update(orderFromDb);
    await _context.SaveChangesAsync();
    return Json(new { updatedStatusId = id });
}
```

фиг. 29: Методът UpdateStatus

5.7. UsersRoles контролер

Контролерът UsersRoles се използва за управление на ролите на потребителите в уеб приложението и е достъпен единствено от Супер Админа. Методът Index връща списък с всички потребител и техните роли. Manage() методът се използва за добавяне и премахване на ролите на потребител.



6. Помощни класове

6.1. HttpContext

HttpContext е клас в ASP.NET, който съдържа информация за текущия HTTP заявка и отговор. Може да бъде достъпен в различни части от ASP.NET приложението, като контролери, средни слоеве и изгледи, което го прави изключително полезен за манипулиране на HTTP заявките и отговорите в приложението. Някои от основните свойства на HttpContext включват Request, Response, Session и User. Това, което ще бъде използвано в настоящия проект е:

- Session – съхранява данните между заявките за конкретен потребител.
- User – съдържа информацията за текущия потребител, който изпълнява заявката, като име, идентификационен номер и роли.

6.2. EmailSender

Класът EmailSender имплементира интерфейса IEmailSender. Съдържа метода SendEmailAsync, който се използва за изпращане на електронни писма. В тялото му се създава нов обект от класа SmtpClient, който служи за изпращане на SMTP заявки към посочения хост и порт. Това е част от .NET.Mail библиотеката, спомената в използваните технологии. Задават се различни настройки за клиента, като порта, хоста, SSL връзката, начина на доставка и удостоверяването на идентичността. В конкретния случай за настоящото приложение се използва Gmail SMTP сървър, който изпраща електронното писмо, използвайки аргументите, подадени на метода, като адреса на получателя (email), предмета на писмото (subject), HTML съобщението (htmlMessage). Адрес на изпращача е специално създадения за проекта имейл lotusushi.restaurant@gmail.com (фиг. 30).

```
4 references
public Task SendEmailAsync(string email, string subject, string htmlMessage)
{
    SmtpClient client = new SmtpClient
    {
        Port = 587,
        Host = "smtp.gmail.com",
        EnableSsl = true,
        DeliveryMethod = SmtpDeliveryMethod.Network,
        UseDefaultCredentials = false,
        Credentials = new NetworkCredential("lotusushi.restaurant@gmail.com", "XXXXXXXXXXXXXX")
    };

    return client.SendMailAsync("lotusushi.restaurant@gmail.com", email, subject, htmlMessage);
}
```

фиг. 30: Класът EmailSender



6.3. CartService

Класът CartService съдържа логиката на функционалностите, които притежава пазарската количка на потребителя. Той три променливи – `_context` и `httpContextAccessor`, служещи за Dependency Injection на `ApplicationContext` и `IHttpContextAccessor`, и `ShoppingCartId`, чиято стойност се задава при инициализацията на обекта, като извиква метода `GetCartId()`. При успешно влизане в профила, потребителя получава своята кошница чрез метода `MigrateCart()`, който приема като параметър потребителско име. Освен този метод, класът има и следните методи:

- `AddToCart` – приема обект от тип `Product` като параметър и го добавя към кошницата на потребителя (*фиг. 31*).
- `RemoveFromCart` – приема `Id`-то на продукта, който потребителят е пожелал да изтрие, и го премахва от кошницата на потребителя (*фиг. 32*).
- `IncreaseByOneCartItemCount` – приема `Id`-то на продукта, чието количество потребителят е пожелал да увеличи с една бройка.
- `DecreaseByOneCartItemCount` – приема `Id`-то на продукта, чието количество потребителят е пожелал да намали с една бройка.
- `EmptyCart` – премахва всички артикули от кошницата на потребителя.
- `GetCartItems` – връща списък от `CartItems`, които могат да се използват за изписване или обработка.
- `GetCount` – връща общия брой на всички продукти, които потребителят има в кошницата си.
- `GetTotal` – изчислява общата стойност на всички артикули в кошницата.
- `GetTotalToString` – изписва като символен низ изчислената от `GetTotal` обща сума.
- `CreateOrder` – преобразува продуктите от кошницата на потребителя в елементи за поръчка.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

1 reference

```
public void AddToCart(Product product)
{
    var cartItem = _context.CartItems.Include(c =>c.Product)
        .SingleOrDefault(c => c.CartId == ShoppingCartId && c.ProductId == product.Id);

    if (cartItem == null)
    {
        // Create a new cart item if no cart item exists
        cartItem = new CartItem
        {
            Product = product,
            ProductId = product.Id,
            CartId = ShoppingCartId,
            Count = 1,
            DateCreated = DateTime.Now
        };
        _context.CartItems.Add(cartItem);
        _context.SaveChanges();
    }
    else
    {
        cartItem.Count++;
    }
    // Save changes
    _context.SaveChanges();
}
```

фиг. 31: Методът AddToCart

1 reference

```
public void RemoveFromCart(int id)
{
    // Get the cart
    var cartItem = _context.CartItems.Include(c=>c.Product).Single(
        c => c.CartId == ShoppingCartId
        && c.Id == id);

    if (cartItem != null)
    {
        _context.CartItems.Remove(cartItem);
        _context.SaveChanges();
    }
}
```

фиг. 32: Методът RemoveFromCart



6.4. Braintree

6.4.1. BraintreeSettings

Класът BrainTreeSettings има четири свойства, които се използват за конфигуриране на необходимите настройките за успешното изпълнение на платежните транзакции в BrainTree (*фиг. 34*):

- Environment – съхранява информация за средата, в която трябва да се изпълняват плащанията (например, "production" за продуктова среда или "sandbox" за тестова среда, в настоящия проект ще се си използва "sandbox").
- MerchantId – съхранява уникалния идентификационен номер на търговеца в BrainTree.
- PublicKey – съхранява публичният ключ, който се използва за автентикация на плащанията.
- PrivateKey – съхранява частният ключ, който се използва за криптиране на данните на плащането.

```
3 references
public class BrainTreeSettings
{
    1 reference
    public string Environment { get; set; }

    1 reference
    public string MerchantId { get; set; }

    1 reference
    public string PublicKey { get; set; }

    1 reference
    public string PrivateKey { get; set; }
}
```

фиг. 34: Свойствата на класа BraintreeSettings

6.4.2. Braintree

Клас Braintree (*фиг. 35*) се използва за създаване и използване на връзка с платформата за онлайн плащания BrainTree. Има две свойства:

- Options – от тип "BrainTreeSettings" и съхранява настройките за конфигурация на връзката с BrainTree.
- "BraintreeGateway" – от тип "IBraintreeGateway" и съхранява връзката с BrainTree, която ще бъде създадена при първото извикване на метода "GetGateway()".



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

Методът "CreateGetway()" създава нова връзка за извършване на плащания с BrainTree, като използва настройките, зададени в свойството "Options".

Методът "GetGateway()" връща вече създадената връзка с BrainTree, ако съществува. Ако все още няма създадена връзка, методът я създава, като използва метода "CreateGetway()", и я връща за ползване.

```
2 references
public class BrainTree : IBrainTree
{
    5 references
    public BrainTreeSettings Options { get; set; }
    3 references
    private IBraintreeGateway BraintreeGateway { get; set; }

    0 references
    public BrainTree(IOptions<BrainTreeSettings> options)
    {
        Options = options.Value;
    }
    2 references
    public IBraintreeGateway CreateGetway()
    {
        return new BraintreeGateway(Options.Environment,
            Options.MerchantId, Options.PublicKey, Options.PrivateKey);
    }

    3 references
    public IBraintreeGateway GetGateway()
    {
        if(BraintreeGateway == null)
        {
            BraintreeGateway = CreateGetway();
        }
        return BraintreeGateway;
    }
}
```

фиг. 35: Класът BraintreeSettings



7. Уеб сайт

7.1. Начална страница

Началната страница (фиг. 36) предоставя основна информация за ресторантa. Навигационният бар се променя спрямо потребителя, който е влязъл и неговата роля.

LotusSushi Home Menu Privacy

Register Login

Lotus Sushi Restaurant

Delivering happiness to your home

Menu



How We Work

01

Goods order

You can order any product if you have an account

02

Payment Confirmation

You can pay using credit card

03

Delivery

Free delivery

04

Confirmation

We will let you know what the status of your order is

Facts & Questions

Here are some frequently asked questions



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

Facts & Questions

Here are some frequently asked questions

How to pay for an order?	+
How to see the details of my order?	+
Where is my order going to be delivered?	+
Do you deliver orders outside the town?	+

No, due to the fact that our restaurant is located in Plovdiv, we deliver only in Plovdiv district.

фиг. 36: Начална страница

7.2. Меню с продукти

Страницата предоставя списък от всички налични продукти, които предлага ресторантата (фиг. 37). Всеки продукт си има бутонче-сърце, чрез което го отбелязва като любим, и бутон-количка, чрез което го добавя в количката. При натискате на името му се отваря нова страница с детайлите към продукта.

Menu

 SUSHI SALMON & AVOCADO ROLL 14,95 лв.	 SUSHI SHRIMP TEMPURA ROLL 16,95 лв.	 SUSHI RAINBOW ROLL 18,95 лв.
---	---	--

фиг. 37: Страница с меню на ресторантата



7.3. Регистрация

Страницата за регистрация съдържа форма, чрез която се осъществява създаването на потребителски профил (*фиг. 38*). За създаването на акаунт е необходимо да се въведе име и фамилия, имейл, телефонен номер и парола. Паролата трябва да бъде поне 6 символа дълга, да е съставена от символите на английската азбука и да съдържа поне една малка и една главна буква, както и един небуквен символ (като !). Потребителското име за уеб приложението се създава автоматично, като се взима за потребителското име на имейл. След това може да бъде заменено с друго, стига да е уникално, тоест да не е заето от друг човек. Поради тази причина с един имейл може да се създаде само един профил. За да бъде профила активен за употреба, имейл адресът трябва да бъде потвърден чрез натискането на автоматично изпратен от уеб приложението линк в пощата на същия този имейл.

Register

Create a new account.

Make your account now!

First Name

Having an account gives you the opportunity to order products, have personal shopping cart, pay using credit card and track your order status.

Last Name

Email

Password

Confirm password

Phone number

[Register](#)

фиг. 38: Страницата за регистрация



7.4. Логин

Страницата за влизане във вече създаден и потвърден акаунт съдържа форма, в която трябва да се въведе валиден имейл/потребителско име със съответстваща парола (*фиг. 39*). Ако опита за влизане е неуспешен, потребителят може или да направи повторен опит за влизане в профила си, или в случай на забравена парола, да натисне линка за смяна на забравената парола.

LotusSushi Home Menu Privacy

Register Login

Log in

Use a local account to log in.

Email / Username

youremail@gmail.com

Use another service to log in.

There are no external authentication services configured. See [this article](#) for details on setting up this ASP.NET application to support logging in via external services.

Password

Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

фиг. 39: Страницата за влизане в потребителския акаунт

7.5. Детайли на потребителския акаунт

Всеки потребител има собствена страница с детайлите относно акаунта му (*фиг. 40*). Тя съдържа навигационно меню, което води към следните страници:

- Поръчки (*фиг. 41*) – където потребителят може да види списък с направените от него поръчки;
- Любими продукти (*фиг. 42*) – където потребителят може да види списък с любимите му продукти;
- Персонализиране на профила (*фиг. 38*) – където потребителят може да види и промени данните относно личния си акаунт;
- Имейл (*фиг. 43*) – където потребителят може да замени и/или потвърди имейла си;
- Парола (*фиг. 44*) – където потребителят може промени паролата си;
- Изтриване на акаунта (*фиг. 45*) – където потребителя може да изтрие всички данни относно акаунта си, тоест да изтрие профила си.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾

Hello monikadev87! Logout Cart (0)

Manage your account

Change your account settings

Orders

Favourites

Profile

Email

Password

Personal data

Profile

First Name

Monika

Profile Picture



Last Name

Dev

Username

monikadev87

Phone number

0893487484

Save

Choose File flora.jpg

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾

Hello monikadev87! Logout Cart (2)

Your orders

Date	Order's status	Products	Total	
16.4.2023 г. 11:40:51	Shipped	Shrimp Tempura Roll 1 Rainbow roll 1	35,90	Details
19.4.2023 г. 8:08:42	Shipped	Shrimp Tempura Roll 2 Mochi 1 Umeshu 1	46,40	Details
22.4.2023 г. 17:47:16	Shipped	Shrimp Tempura Roll 3 Mochi 1	58,80	Details
27.4.2023 г. 10:38:13	Confirmed	Tokyo star 1 Sake 2	54,85	Details
27.4.2023 г. 11:38:07	Confirmed	Cherry blossom lava cake 1 Sesame tuna roll 1	24,90	Details

фиг. 41: Страница с поръчките на потребителя



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾



Hello monikadev87! Logout Cart (0)

Favourites



SUSHI
RAINBOW ROLL
18,95 лв.



SUSHI
PARADISE COMBO
30,99 лв.



SUSHI
UMESHU
4,55 лв.

фиг. 42: Страница с любимите продукти на потребителя

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾



Hello monikadev87! Logout

Manage your account

Change your account settings

Orders

Favourites

Profile

Email

Password

Personal data

Manage Email

Email

monikadev875@gmail.com



New email

monikadev875@gmail.com

Change email

фиг. 43: Страница с любимите продукти на потребителя



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾



Hello monikadev87! Logout

Manage your account

Change your account settings

Orders

Favourites

Profile

Email

Password

Personal data

Change password

Current password

New password

Confirm new password

Update password

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾



Hello monikadev87! Logout

Manage your account

Change your account settings

Orders

Favourites

Profile

Email

Password

Personal data

Personal Data

Your account contains personal data that you have given us. This page allows you to download or delete that data.

Deleting this data will permanently remove your account, and this cannot be recovered.

Download

Delete

фиг. 44: Страница за смяна на паролата



7.6. Пазарска количка

Страницата съдържа таблица с продуктите, добавени в количката (*фиг. 46*). Всеки продукт си има количество, което може да се регулира с помощта на + и – бутони, и бутона за премахване. Количката автоматично пресмята общата сума на продуктите. При натискането на бутона “Checkout” се зарежда страницата за осъществяване на поръчка.



Review your cart:

[Back to Menu](#)

Name	Price (each)	Quantity	
 Tokyo star	47,95 лв.	- 1 +	Remove from cart
 Sake	3,45 лв.	- 1 +	Remove from cart
 Cherry blossom lava cake	9,95 лв.	- 1 +	Remove from cart
Total:			51,40 лв.

[Checkout](#)

Item was removed

фиг. 46: Пазарска количка на потребителя

7.7. Осъществяване на поръчка

Страницата за осъществяване на поръчка (*фиг. 47*) съдържа форма, в която потребителят трябва да попълни желания адрес и час на доставка на поръчката. След това е нужно да избере метод за плащане, като например с карта. Ако плащането е успешно (*фиг. 48*) поръчката получава статус “Потвърдена” и се зарежда страницата с детайлите за нея



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

LotusSushi Home Menu Privacy Management ▾ To-Do List ▾



Hello monikadev87! Logout Cart (3)

Make an order

We are going to use your profile name, email and phone number for delivery information. You only need to enter the delivery address and the time of delivery

DeliveryAddress

Momina sulza 17

DeliveryTime

17:30

Make a transaction for your delivery. You can use credit card or paypal

Pay with card

Card Number
4111 1111 1111 1111

Expiration Date (MM/YY)
02 / 24

[Choose another way to pay](#)

Order and pay

фиг. 47: Страницата за осъществяване на поръчка

Make a transaction for your delivery. You can use credit card or paypal

Paying with Card

Ending in 1111 Visa

[Choose another way to pay](#)

Order and pay

фиг. 48: Успешно плащане

7.8. Детайли за поръчка

Страницата съдържа подробната информация относно конкретната поръчка (фиг. 49), която включва: адрес и час на доставка, името, имейла и телефонния номер на потребителя, който я е поръчал, и списъка с продуктите.



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com



Order's details

First name: Monika
Last name: Dev
Email: monikadev875@gmail.com
Phone number: 0893487484
Delivery address: Momina sulza 17
Time of delivery: 17:30



	Name	Price (each)	Quantity	Sub total
	Tokyo star	47,95 лв.	1	47,95
	Sake	3,45 лв.	2	6,90
Total:				54,85 лв.

фиг. 49: Детайли на поръчката

7.9. To-do лист

Страницата съдържа списък с всички поръчки, направени от клиентите (фиг. 50), които предстоят да бъдат пригответи. При означаване на поръчката като готова, тя се изтрива от списъка и статуса ѝ се променя на “Изпратена”.



Orders to do

When the order is give to the delivery guy mark it as "Shipped" by clicking on it once. Then it will fade out.

Nikola Vapcarov 27 for 21:00 : Sesame tuna roll - 1 pcs. | Paradise Combo - 1 pcs. |

Momina sulza 17 for 17:30 : Tokyo star - 1 pcs. | Sake - 2 pcs. |

Скопие 45 for 15:45 : Cherry blossom lava cake - 1 pcs. | Sesame tuna roll - 1 pcs. |

фиг. 50: To-do лист

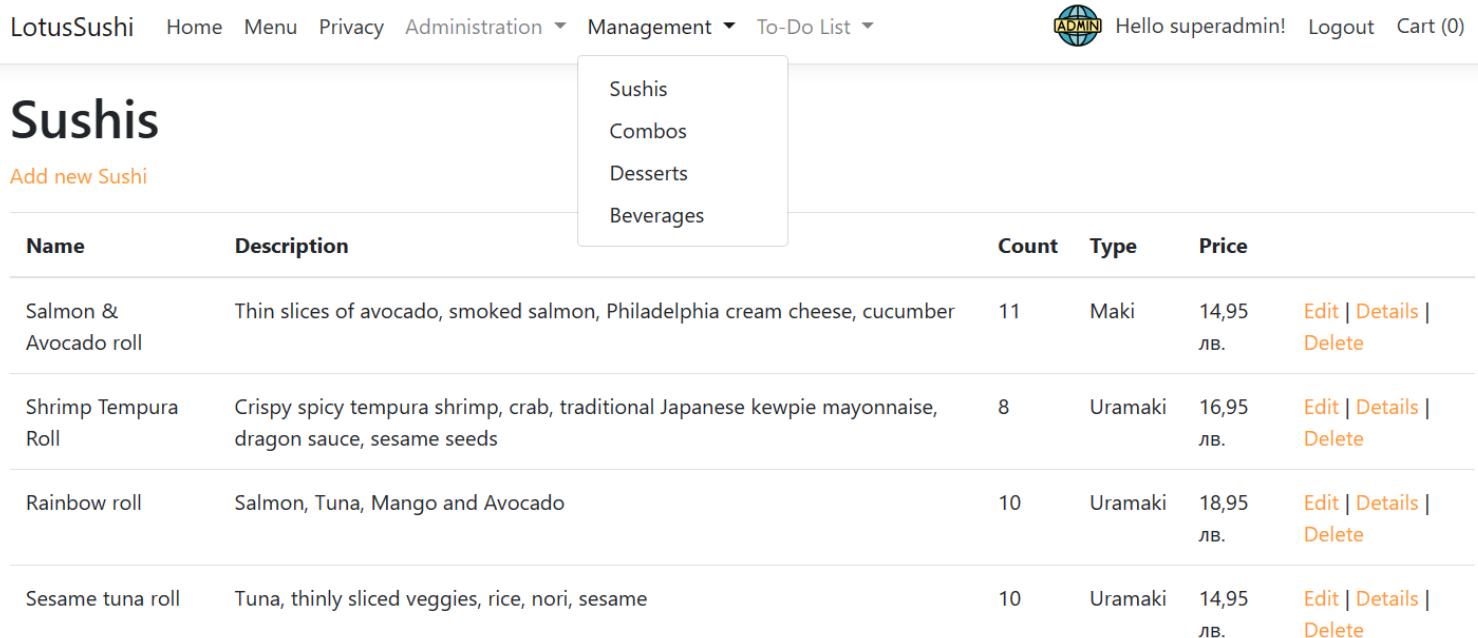


МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

7.10. CRUD на Sushis, Combos, Desserts и Beverages

Всеки тип продукт има страница за списък от всички продукти (фиг. 51), страница с форма за добавяне на нов продукт (фиг. 52), страница с форма за промяна на конкретен продукт (фиг. 53), страница с детайлите на продукт (фиг. 54) и страница с форма за изтриване на даден продукт (фиг. 55).



Name	Description	Count	Type	Price	
Salmon & Avocado roll	Thin slices of avocado, smoked salmon, Philadelphia cream cheese, cucumber	11	Maki	14,95 лв.	Edit Details Delete
Shrimp Tempura Roll	Crispy spicy tempura shrimp, crab, traditional Japanese kewpie mayonnaise, dragon sauce, sesame seeds	8	Uramaki	16,95 лв.	Edit Details Delete
Rainbow roll	Salmon, Tuna, Mango and Avocado	10	Uramaki	18,95 лв.	Edit Details Delete
Sesame tuna roll	Tuna, thinly sliced veggies, rice, nori, sesame	10	Uramaki	14,95 лв.	Edit Details Delete

фиг. 51: Списък от всички продукти

Create

Dessert

Name

Description

Grams

Price

[Back to List](#)

фиг. 52: Добавяне на нов продукт



МАТЕМАТИЧЕСКА ГИМНАЗИЯ „АКАДЕМИК КИРИЛ ПОПОВ“

4001 Пловдив, ул. „Чемпир“ № 11, e-mail: omg@omg-bg.com, www.omg-bg.com

LotusSushi Home Menu Privacy Administration ▾ Management ▾ To-Do List ▾

Edit

Combo

Name	ImageURL
<input type="text" value="Salmon Combo"/>	
Description	<input type="text" value="Smoked salmon, Philadelphia cre"/>
Count	<input type="text" value="20"/> <input type="button" value="Choose File"/> No file chosen
Price	<input type="text" value="20,00"/>
<input checked="" type="checkbox"/> 15 Salmon & Avocado roll <input type="checkbox"/> 16 Shrimp Tempura Roll <input checked="" type="checkbox"/> 17 Rainbow roll <input type="checkbox"/> 28 Sesame tuna roll	
<input type="button" value="Save"/>	

фиг. 53: Промяна на продукт

Details

Beverage

Name	Okinawa's secret
Description	The secret to longevity - aloe vera juice
Milliliters	250
Type	NonAlcoholic
Price	5,95 лв.

[Edit](#) | [Back to List](#)

фиг. 54: Детайли на продукт



Delete

Are you sure you want to delete this?

Sushi

Name	Salmon & Avocado roll
Description	Thin slices of avocado, smoked salmon, Philadelphia cream cheese, cucumber
Count	11
Type	Maki

[Delete](#)

| [Back to List](#)

фиг. 55: Изтриване на продукт

7.11. Мениджър на потребителските роли

Страницата съдържа списък с всички потребители и техните роли (фиг. 56).

За всеки потребител има бутон Manage, чрез който се достъпва помощна страница за добавяне/премахване на роля на конкретния потребител с checkbox-ове.

Users				
First Name	Last Name	Email	Roles	Action
Monika	Snimki	monikasnimki@gmail.com	Basic	Manage Roles
Monika	Dev	monikadev875@gmail.com	Admin , Basic	Manage Roles
test	testov	sushilotus.restaurant.web@gmail.com	Basic	Manage Roles
Monika	Blazheva	monitob875@gmail.com	Basic	Manage Roles
Radoslav	Dimitrov	silverking@mail.bg	Basic , Moderator	Manage Roles
test	testov	testvanetesting@gmail.com	Basic	Manage Roles
Monika	Blazheva	monitob875@gmail.com	Admin , Basic , Moderator	Manage Roles
Sofia	Izidova	sofiaiz@mail.bg	Basic	Manage Roles

фиг. 56: Списък с всички потребители и техните роли



8. Заключение

Идеята на проекта да се създаде уеб приложение, което да предоставя ръководство на ресторант онлайн е вече превърната в реалност. То предлага услуги, специално разработени за ръководството, служителите и клиентите на ресторант, и позволява на предприемачите да стартират своя собствен бизнес без необходимостта от физическо пространство за ресторанта и наемане на персонал за обслужването на клиентите. Ресторантът ще използва уеб приложението, за да приема поръчки и да изпраща храната на клиентите с помощта на доставчици, да управлява изготвянето на поръчаната храна и да управлява ролите на служителите. Идеи за бъдещо развитие на проекта включват:

- Добавяне на възможност за плащане на поръчка чрез PayPal;
- Подобряване на графичния дизайн на приложението;
- Добавяне на филтрация на продуктите в менюто.



9. Използвана литература

Уеб сайтове:

- <https://learn.microsoft.com/en-us/dotnet/>

Книги:

- “Architect Modern Web Applications with ASP.NET Core and Azure”, Steve “ardalis” Smith - Software Architect and Trainer;
- “Porting Existing ASP.NET Apps to .NET”, Steve “ardalis” Smith, Software Architect and Trainer;
- “Основи на програмирането със C#”, Наков и колектив, 2017;
- “Въведение в програмирането със C#”, Наков, Колев и колектив, 2011;
- “Програмиране за .NET Framework”, (том 1 и том 2), Наков и колектив.