****

**ДИПЛОМЕН ПРОЕКТ**

**Тема: „Резервации“**

**Разработил: Сезгин Айдън Салим ……………**

*(име, презиме, фамилия) (подпис)*

**професия код 481030 „Приложен програмист“специалност код 4810301 „Приложно програмиране“**

**Ръководител-консултант: инж. Николай Христов ………………**

*(име, фамилия) (подпис)*

**2022/2023 г.**

**Разработил: Иван Петров Христов ……………**

*(име, презиме, фамилия) (подпис)*

**професия код 481030 „Приложен програмист“специалност код 4810301 „Приложно програмиране“**

**Ръководител-консултант: инж. Георги Кожухаров ………………**

*(име, фамилия) (подпис)*

**2022/2023 г.**

**ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО МЕХАНОТЕХНИКА, ЕЛЕКТРОНИКА, ТЕЛЕКОМУНИКАЦИИ И ТРАНСПОРТ „ХРИСТО БОТЕВ“ – ШУМЕН**

**Съдържание**

**1.    Въведение**

**2. Използвани технологии**

**3.    Описание на приложението**

**3.1. Кратко описание**

**3.2. Описание и анализ на известни решения**

**3.3. Основни етапи в реализирането на проекта**

**3.4. Основни функции**

**3.5. Основни модули и връзки между тях**

**3.6. Изисквания за инсталация**

**4.    Архитектура на проекта. Реализация.**

**4.1. Бази данни**

**4.2. Потребителски интерфейс**

**4.3. Слой за услуги /контролери, екшъни, сървиси/**

**5.   Заключение /бъдещо развитие на проекта/**

**6.   Списък на използвана литература**

**7.    Приложения**

**7.1. Наръчник за работа на потребителя**

**7.2. Програмен код**

**1.Въведение:**

Целта на уеб приложението е резервиране на места в ресторант. Собственикът на ресторанта има право да публикува ресторанти в уеб сайта, да ги изтрива и да ги редактира. Публикуваните ресторанти са достъпни за всички. При споделянето на ресторант, потребителят попълва нужните полета. Тези полета са име на ресторант, адрес, град, цена и снимка. Потребителят има право да направи резервация, който става чрез бутона „Резервирай“.

**2.Използвани технологии:**

HTML - основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в интернет, а неговите стандарти се определят от международния консорциум W3C. Текущата версия на стандарта е HTML 5.0 (от 28 октомври 2014 г.), а предходната стабилна версия е HTML 4.1.

Описанието на документа става чрез специални елементи, наречени HTML елементи или техните маркери, които се състоят от тагове и съответстващите етикети (HTML tags) и ъгловискоби (като например елемента <html>). HTML елементите са основната градивна единица на кода, който изграждат уеб страниците. Чрез тях се форматира, графично оформя текста и неговите отделните части в рамките на една уеб страница, като например заглавия, цитати, текстови раздели, хипертекстови препратки и т.н. Най-често HTML елементите са групирани по двойки <h1> и </h1>.

В повечето случаи HTML кодът е написан в текстови редактори, с файлов формат .html, .htm, dhtml и се качва и хоства на сървъри, които са онлайн в интернет или са част от www мрежата. Тези .html файлове съдържат програмно на таговете на HTML и текстово съдържание със маркери и коментари – също инструкции за браузъра, за това какъв точно тип е .html страницата, а също за това как да се показва текстът, особено що се отнася до езиковите характеристики. За да се илюстрира как се включва текст в HTML код: <маркер> Някакъв текст. </край на маркера>. уеб браузърите са програмирани от своя страна така, в повечето случаи, макар че някои браузъри могат да имат съответно проблеми на версията, за да могат да прочетат HTML документите и да ги покажат на екрана като уеб страници. Браузърите не показват самите HTML тагове, освен ако не се отиде в менюто за да се направи това, така че те „интерпретират“ (тоест парсват) съдържанието на страницата като код и текст за да могат след работа на процесора да покажат желаното уеб-съдържание.

Основното предимство на HTML е, че уеб-страниците, които са го включват в кода си, могат да се разглеждат чрез показването им от браузъра на екрана на повечето устройства. Уебстраницата може да има дизайн, който дори изглежда с добър дизайн с помощта на CSS или „правилно оформен“ (например с помощта на C#), както върху монитора на персоналния компютър, но също и върху миниатюрния дисплей на пейджър или дисплея на мобилен телефон.

HTML може да прикрепя скриптове писани на езици като JavaScript, който е помощен за HTML, и това променя поведението на дадена уеб страница. Cascading Style Sheets (CSS) се използват, като това се прави за да се определя изгледа и оформлението на текста и други включени в страницата изображения и илюстриращи материали. World Wide Web Consortium (W3C) поддържа както HTML, така и CSS, и насърчава използването на CSS в HTML страниците още от 1997. Това допринася за разделяне съдържанието и структурата на уеб страниците от тяхното визуално представяне.

**CSS -** проектиран да позволява разделянето на съдържание и представяне , включително оформление , цветове и шрифтове .  Това разделяне може да подобри достъпността на съдържанието ; осигуряват повече гъвкавост и контрол при определяне на характеристиките на представяне; позволяват на множество уеб страници да споделят форматиране чрез указване на съответния CSS в отделен .css файл, което намалява сложността и повторенията в структурното съдържание; и активирайте .css файла да бъде кеширан , за да подобрите скоростта на зареждане на страницата между страниците, които споделят файла и неговото форматиране.

Разделянето на форматирането и съдържанието също така прави възможно представянето на една и съща страница за маркиране в различни стилове за различни методи за изобразяване, като например на екрана, при печат, чрез глас (чрез базиран на реч браузър или екранен четец) и базиран на Брайл тактилни устройства. CSS също има правила за алтернативно форматиране, ако съдържанието е достъпно от мобилно устройство .

*Каскадното* име идва от определената приоритетна схема, за да се определи кое правило за стил се прилага, ако повече от едно правило съответства на определен елемент. Тази каскадна приоритетна схема е предвидима.

CSS спецификациите се поддържат от World Wide Web Consortium (W3C). Типът интернет медия ( тип MIME ) text/cssе регистриран за използване с CSS от RFC 2318 (март 1998 г.). W3C управлява безплатна услуга за валидиране на CSS за CSS документи.

В допълнение към HTML, други езици за маркиране поддържат използването на CSS, включително XHTML , обикновен XML , SVG и XUL .

**Подход Code First** – Ние пишем кода, а базата данни се създава автоматично.

Подходът "Code First" (Код-Първо) е един от подходите за създаване на бази данни в програмния език C#. Той се основава на това да се пише код, който да дефинира модела на базата данни и след това да се генерира съответната база данни въз основа на този модел.

В подхода "Code First" се използва подходящ ORM (Object-Relational Mapping) фреймуърк, който да позволи на програмиста да дефинира модела на базата данни с помощта на класове и свойства в C#. Когато моделът е дефиниран, ORM фреймуъркът може да генерира съответната база данни въз основа на този модел. Това позволява на програмистите да се фокусират върху дизайна на модела на базата данни, без да се налага да пишат SQL заявки за създаване на базата данни.

При подхода "Code First" се използват анотации в C#, за да се дефинират свойствата на класовете, които съответстват на колоните в базата данни. С помощта на тези анотации програмистите могат да зададат различни атрибути, като например дали едно свойство е задължително или определен тип на данните, които да се съхраняват в тази колона.

Подходът "Code First" има няколко предимства, като например възможността да се използват обектно-ориентирани концепции при създаването на базата данни, както и възможността за автоматично генериране на базата данни въз основа на дефинирания модел. Това прави процеса на създаване на база данни по-лесен и по-бърз. Освен това, "Code First" подходът позволява да се избегне писането на SQL заявки за създаване на базата данни и да се работи с по-познатия за програмистите C# код.

**C#** - [език за програмиранe](https://en.wikipedia.org/wiki/Programming_language)на високо ниво с общо предназначение, поддържащ множество парадигми . C# обхваща статично типизиране, силно типизиране , лексикално обхват , императивно , декларативно , функционално , общо , обектно-ориентирано ( базирано на клас ) и компонентно-ориентирано програмиране.

Езикът за програмиране C# е проектиран от Anders Hejlsberg от Microsoft през 2000 г. и по-късно е одобрен като международен стандарт от Ecma (ECMA-334) през 2002 г. и ISO / IEC (ISO/IEC 23270) през 2003 г. Microsoft въведе C# заедно с .NET Framework и Visual Studio , като и двете бяха със затворен код . По това време Microsoft нямаше продукти с отворен код. Четири години по-късно, през 2004 г., стартира безплатен проект с отворен код, наречен [Mono , който предоставя](https://en.wikipedia.org/wiki/Mono_(software))[междуплатформен](https://en.wikipedia.org/wiki/Cross-platform)компилатор и среда за изпълнение за езика за програмиране C#. Десетилетие по-късно Microsoft пусна Visual Studio Code (редактор на код), Roslyn (компилатор) и унифицираната .NET платформа (софтуерна рамка), всички от които поддържат C# и са безплатни, с отворен код и междуплатформени. Mono също се присъедини към Microsoft, но не беше обединен с .NET.

Към ноември 2022 г. най-новата стабилна версия на езика е C# 11.0, която беше пусната през 2022 г. в .NET 6.0.

**Bootstrap** -  безплатна [CSS рамка](https://en.wikipedia.org/wiki/CSS_framework)[с отворен код,](https://en.wikipedia.org/wiki/Free_and_open-source) насочена към отзивчиво, [първо ориентирано към мобилни устройства](https://en.wikipedia.org/wiki/Responsive_web_design" \l "Mobile_first,_unobtrusive_JavaScript,_and_progressive_enhancement" \o "Адаптивен уеб дизайн)предно уеб развитие . Той съдържа HTML , CSS и (по избор) базирани на JavaScript шаблони за дизайн за типография , формуляри , бутони , навигация и други компоненти на интерфейса.

Към декември 2022 г. Bootstrap е 14-ият проект с най-много звезди (4-та библиотека с най-звезди) в GitHub с над 161 000 звезди. Според W3Techs, Bootstrap се използва от 19,2% от всички уебсайтове.

**ASP .NET CORE** - безплатна [уеб рамка](https://en.wikipedia.org/wiki/Web_framework" \o "Web framework)с отворен код и наследник на ASP.NET , разработена от Microsoft. Това е модулна рамка, която работи както на пълната .NET Framework , на Windows , така и на [кросплатформената](https://en.wikipedia.org/wiki/Cross-platform" \o "Cross-platform).NET . Въпреки това ASP.NET Core версия 3 работи само на .NET Core, премахвайки поддръжката на .NET Framework.

Рамката е напълно пренаписана, която обединява отделните преди това ASP.NET MVC и ASP.NET Web API в единен програмен модел .

Въпреки че е нова рамка, изградена върху нов уеб стек, тя има висока степен на концептуална съвместимост с ASP.NET. Рамката ASP.NET Core поддържа успоредно създаване на версии, така че различните приложения, разработени на една машина, да могат да се насочват към различни версии на ASP.NET Core. Това не е възможно с предишни версии на ASP.NET.

Blazor е скорошен (незадължителен) компонент за поддръжка на WebAssembly и от версия 6.0 той преустанови поддръжката за някои стари уеб браузъри. Докато текущият Microsoft Edge работи, наследената му [версия , т.e. „](https://en.wikipedia.org/wiki/Legacy_system" \o "Наследена система)Microsoft Edge Legacy“ и Internet Explorer 11 се премахват, когато използвате Blazor.

**MVC -** софтуерен архитектурен модел, който обикновено се използва за разработване на потребителски интерфейси , които разделят свързаната програмна логика на три взаимосвързани елемента. Това се прави, за да се отделят вътрешните представяния на информация от начините, по които информацията се представя и приема от потребителя.

Традиционно използван за настолни графични потребителски интерфейси (GUI), този модел стана популярен за проектиране на уеб приложения . Популярните езици за програмиране имат MVC рамки, които улесняват прилагането на модела.

**Entity Framework Core -** рамка с отворен код обектно-релационно картографиране (ORM) за ADO.NET . Първоначално беше доставен като неразделна част от .NET Framework , но започвайки с Entity Framework версия 6.0, той се доставя отделно от .NET Framework.

Entity Framework 6.4 беше най-новата версия на класическата рамка. Въпреки че Entity Framework 6 все още се поддържа, той вече не се разработва и ще получава само корекции за проблеми със сигурността.

Нова рамка, известна като Entity Framework Core (EF Core), беше въведена през 2016 г. с подобен, но не пълен паритет на функциите. Номерирането на версиите на тази рамка е рестартирано от 1.0 и най-новата версия на EF Core е 6.0.

**Razor Wiew Engine -** синтаксис за програмиране на ASP.NET , използван за създаване на динамични уеб страници с езиците за програмиране C# или VB.NET . Razor беше в процес на разработка през юни 2010 г. и беше пуснат за Microsoft Visual Studio 2010 през януари 2011 г. Razor е машина за изглед с прост синтаксис и беше пусната като част от MVC 3 и набора от инструменти WebMatrix .

Razor стана компонент на AspNetWebStack и след това стана част от ASP.NET Core .

**Sql Server Manageament Studio (SSMS) -** софтуерно приложение, разработено от Microsoft , което се използва за конфигуриране, управление и администриране на всички компоненти в Microsoft SQL Server . Стартиран за първи път с Microsoft SQL Server 2005, той е наследник на **Enterprise Manager** в SQL 2000 или по-рано. Инструментът включва както редактори на скриптове, така и графични инструменти, които работят с обекти и характеристики на сървъра. [[3]](https://en.wikipedia.org/wiki/SQL_Server_Management_Studio#cite_note-3)

Централна характеристика на SSMS е Object Explorer, който позволява на потребителя да преглежда, избира и действа върху който и да е от обектите в сървъра. Той също така доставя отделно издание Express, което може да бъде изтеглено безплатно; но последните версии на SSMS са напълно способни да се свързват и управляват всеки екземпляр на SQL Server Express. Microsoft също така включи обратна съвместимост за по-стари версии на SQL Server, като по този начин позволи на по-нова версия на SSMS да се свързва с по-стари версии на екземпляри на SQL Server. Освен това идва с Microsoft SQL Server Express 2012 или потребителите могат да го изтеглят отделно.

**Visual Studio 2022 –** кода е написан във Visual Studio.

**GitHub -** GitHub е уеб базирана хостинг платформа за съхранение и управление на програмен код, която позволява на разработчиците да сътрудничат по проекти и да ги контролират с помощта на система за контрол на версиите (VCS).

В основата на GitHub лежи Git - децентрализирана система за контрол на версиите, която позволява на разработчиците да работят с множество различни версии на своите проекти и да комбинират и контролират техните промени на едно място.

GitHub предоставя на потребителите удобна, лесна за използване и мощна платформа, която им позволява да:

Създават, съхраняват и управляват проекти

Разглеждат, коментират и преглеждат проекти на други потребители

Сътрудничат с други разработчици по проекти

Придружават собствените си проекти с документация, бъг репорти и функционални изисквания

Използват различни инструменти за сътрудничество и управление на проекти, като задачи, проблеми, издания и др.

GitHub е много популярна платформа сред разработчиците по цял свят и е използвана за съхранение и управление на някои от най-известните проекти, включително Linux Kernel, Ruby on Rails и jQuery.

**Turtoise Git** - TortoiseGit е софтуер, който предоставя графичен потребителски интерфейс (GUI) за работа с Git, система за контрол на версиите. Git е популярна система за контрол на версиите, която позволява на потребителите да проследят и контролират измененията в програмен код и други файлове. TortoiseGit е наличен за операционни системи Windows и е разработен като допълнение към Windows Explorer.

TortoiseGit предоставя графичен потребителски интерфейс, който позволява на потребителите да използват Git без да са запознати с командния ред. С помощта на TortoiseGit, потребителите могат да използват Git, за да проследят и контролират версиите на своите файлове, да правят изменения, да добавят нови файлове към хранилището на Git, да правят отклонения от последната версия и много други.

TortoiseGit предлага много функции, които могат да помогнат на потребителите да работят с Git по-лесно. Например, TortoiseGit предоставя възможности за преглед на историята на измененията, за възстановяване на предишни версии, за сливане на различни версии, за добавяне на нови файлове и много други. Всички тези функции могат да бъдат извършени чрез контекстното меню в Windows Explorer.

В допълнение към основните функции за работа с Git, TortoiseGit предлага и допълнителни инструменти за управление на кода. Например, той може да се интегрира с различни онлайн хранилища като GitHub и Bitbucket. TortoiseGit предоставя и възможности за анотиране на кода, така че потребителите да могат да видят, кой е направил изменения в определен файл или линия от код.

В заключение, TortoiseGit е мощен и лесен за използване инструмент за работа с Git, който предоставя много функции и инструменти за управление на кода. Той е наличен безплатно за изтегляне от официалния уебсайт и може да бъде използван от начинаещи и опитни програмисти.

**3. Описание на приложението**

**3.1. Кратко описание**

Уеб приложението има за цел да позволи на потребителите да добавят ресторанти и да правят резервации за тях. Това приложение е базирано на модел-изглед-контролер (MVC) архитектура и използва следните технологии: HTML, CSS, C#, .NET Core.

Потребителите могат да създават профил в приложението, където могат да добавят нови ресторанти и да въведат информация за тях, като име, адрес, снимка и др. Създаването на профил може да бъде защитено с парола или други методи за автентикация.

След като потребителите добавят ресторанти, те могат да разглеждат менютата. Менютата могат да включват информация за ястия, напитки, цени и др. Потребителите могат да правят резервации за тези ресторанти.

Администраторите могат да преглеждат всички резервации, съобщения, отзиви и да извършват CRUD операции върху ресторантите.

За да бъде възможно изграждането на това уеб приложение, се използва .NET Core фреймуърк, който осигурява основни функционалности за уеб приложения, като HTTP обработка, маршрутизация, автентикация, и други. Също така, приложението използва CSS и HTML за стилизиране и изграждане на интерфейса, а C# езикът се използва за програмиране на backend логиката и базата данни.

**3.2. Описание и анализ на известни решения**

Този сайт е достъпен за всички, но наличните сайтове за резервиране на места в интернет не са така. Този сайт позволява добавяне на ресторанти и резервиране на места.

Съществуват много решения за уеб приложения, които позволяват на потребителите да добавят ресторанти и да правят резервации, като разглеждат менютата на ресторантите.

OpenTable - OpenTable е едно от най-известните решения за онлайн резервации в ресторанти. Това приложение позволява на потребителите да търсят ресторанти по градове и региони и да правят резервации чрез интернет. OpenTable има широка гама от ресторанти и предоставя детайлна информация за всяко място, включително менюта и отзиви на клиенти. Тъй като OpenTable има голяма клиентска база, това прави приложението привлекателно за ресторантите да бъдат включени в тази мрежа. Един от недостатъците на OpenTable е, че по-малките ресторанти могат да намерят по-трудно място в мрежата.

Resy - Resy е още едно решение за онлайн резервации в ресторанти, като основното му предимство е възможността да правите резервации в топ ресторанти в света. Resy има технологичен подход, който включва защитен софтуер и функции като динамично справяне с натоварването, което го прави ефективно решение за по-големи ресторанти със сложни системи за резервации. Един от недостатъците на Resy е, че по-малките ресторанти не могат да се включат в мрежата.

Yelp - Yelp е популярна платформа за отзиви и резервации в ресторанти, която предлага цялостна информация за ресторантите, включително местоположение, отзиви на клиенти и менюта. Потребителите могат да правят резервации чрез Yelp и да изпращат съобщения до ресторантите. Yelp има голяма клиентска база и се използва често в САЩ, като една от главните най-големи платформи за резервации в ресторанти. Недостатъците на Yelp включват това, че качеството на отзивите може да варира и ресторантите могат да бъдат обект на лоши ревюта.

OpenRest - OpenRest е решение за онлайн резервации и поръчки за храна. OpenRest предлага на потребителите възможността да разглеждат менютата и да правят резервации чрез уебсайт или мобилно приложение. Ресторантите могат да използват OpenRest, за да управляват поръчки и да следят инвентара си. Един от недостатъците на OpenRest е, че ресторантите трябва да се регистрират и да плащат за да използват платформата.

Tablein - Tablein е решение за онлайн резервации в ресторанти, като предлага интуитивен интерфейс, който позволява на потребителите да разглеждат менюта и да правят резервации. Tablein има интегрирани функционалности за управление на резервациите, включително автоматично потвърждение и напомняния за резервации. Един от недостатъците на Tablein е, че приложението не е толкова популярно, колкото други решения като OpenTable и Yelp.

Общият анализ на тези решения показва, че всяко решение има своите предимства и недостатъци. OpenTable е едно от най-популярните решения и има голяма клиентска база, но може да бъде трудно за по-малките ресторанти да се включат. Resy е подходящо за по-големите ресторанти със сложни системи за резервации, но не е подходящо за по-малките ресторанти. Yelp е популярен в САЩ, но качеството на отзивите може да варира. OpenRest е подходящо за управление на поръчки и резервации, но ресторантите трябва да плащат за да използват платформата. Tablein е интуитивно и има интегрирани функционалности за управление на резервации, но не е толкова популярно като други решения.

**3.3. Основни етапи в реализирането на проекта**

1. Идеен проект (Януари, 2023)

2. Работа по дизайна (Февруари, 2023)

3. Проектиране на Базата данни (Февруари, 2023)

4. Свързване на Уеб приложението с База Данни (Март, 2023)

5. Тестване (Март, 2023)

6. Публикуване в интернет (Май, 2023)

**3.4. Основни функции**

Сайтът съдържа следните публични бутони, намиращи се в **навигацията**:

- Ресторанти – в тази секция се намират публикуваните ресторанти. За тези ресторанти има информация за тях като име на ресторанта, цената на резервация, снимка и други.

- Контакти – съдържа формуляр за ипращане на съобщения.

- За нас – съдържа информация за администратора.

- Вход – изполва се за логването в сайта.

- Регистрация – използва се за създаване на акаунт. След създаването на акаунта бутоните вход и регистрация стават невидими за потребителите.

След регистрирането или логването в сайта се появават бутони за изход и бутон чрез който се достъпва собствената информация на потребителя.Тази информация може да бъде променена.

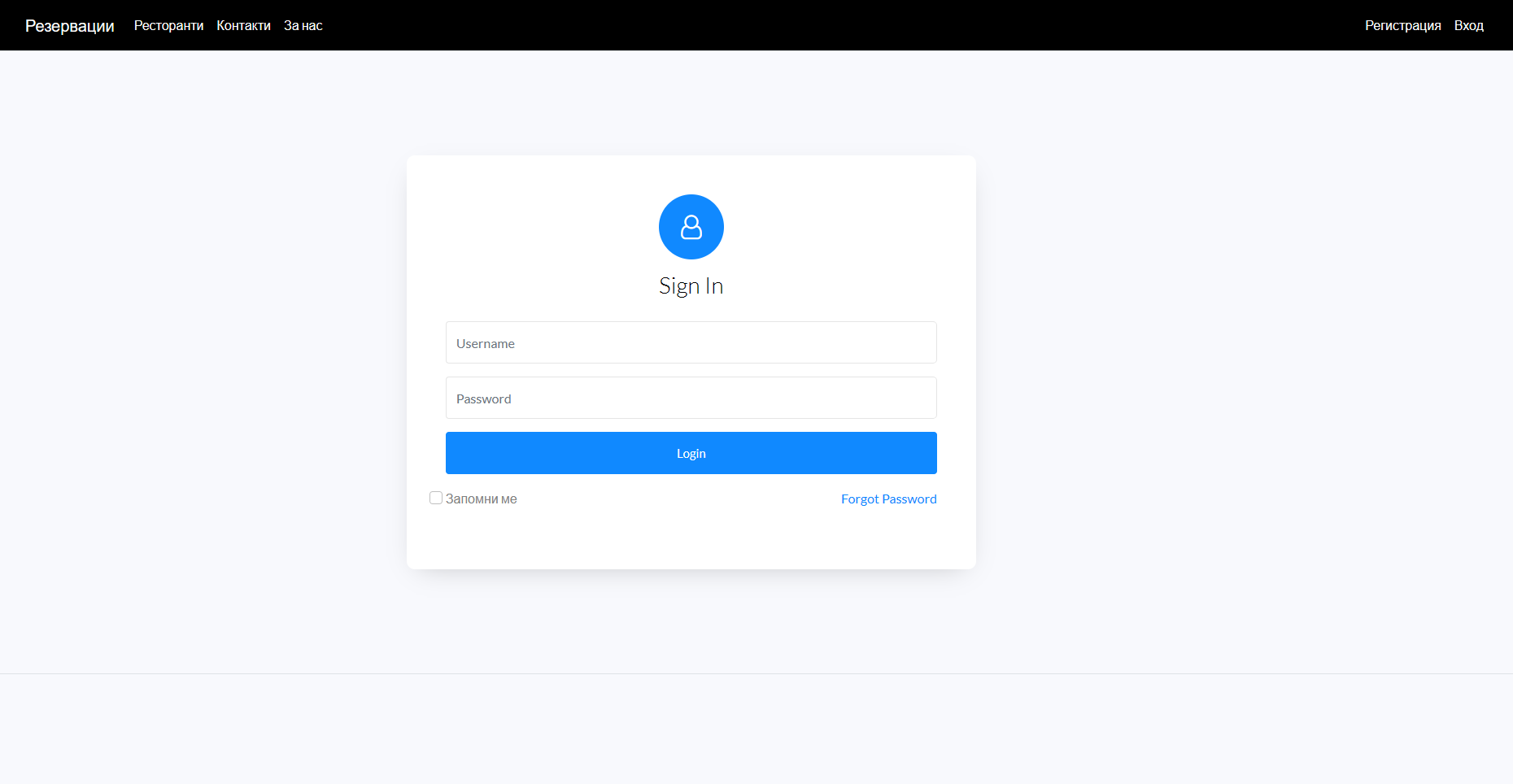
**Футърът** се намира единствено и само в началната страница.

**Добавянето на ресторант** става от бутона „Добави ресторант“. При добавянето на ресторант се попълвар следните полета:

* Име на ресторанта, цена на резервация, адрес, град и снимка.

**Резервация –** ставаследвлизането в детайлите. Избира се бутонът „Резервирай“.

**Вход**



В системата се влиза от бутона „Вход“, който се намира горе в най-дясната част на уеб приложението.

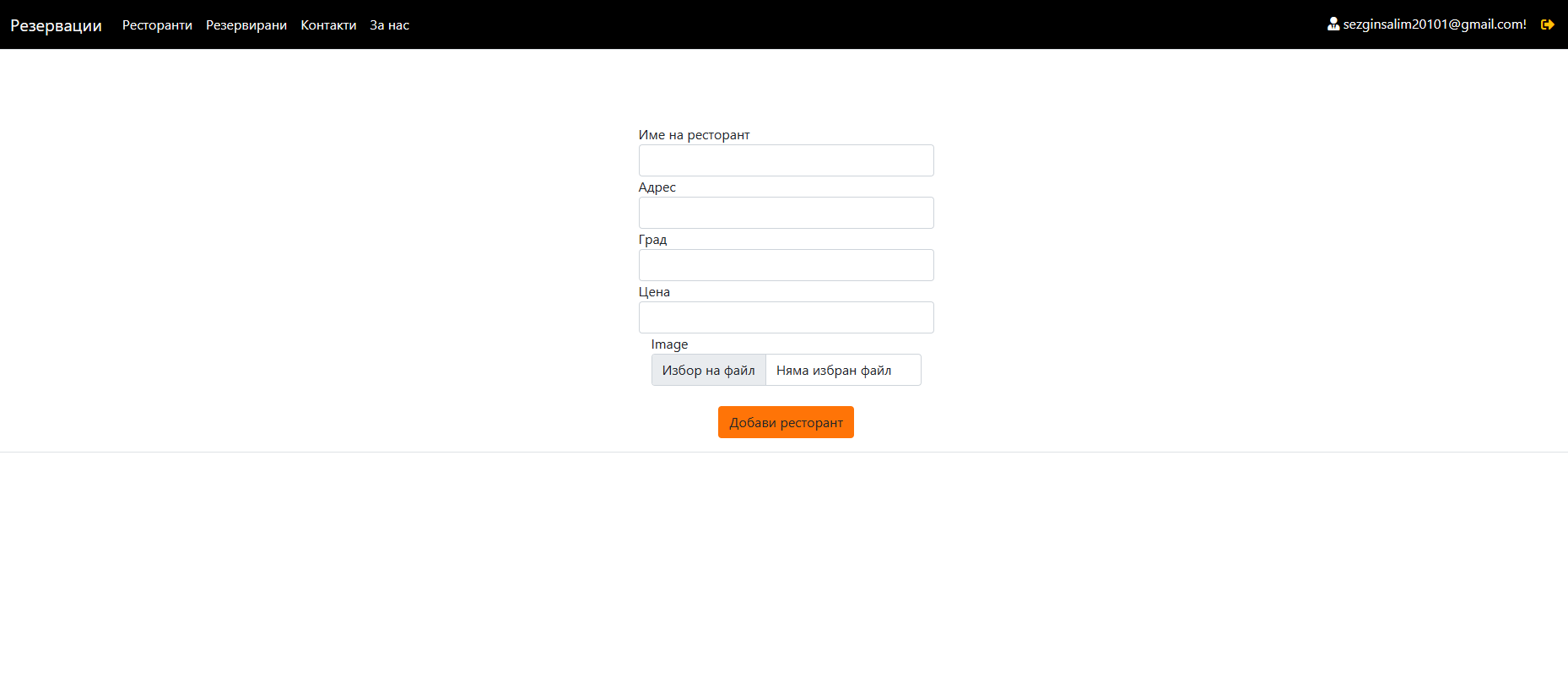
Задължителните полета за попълване при логването са потребителско име и парола.

Входът се осъществява само и единствено ако потребителят е направил регистрация.

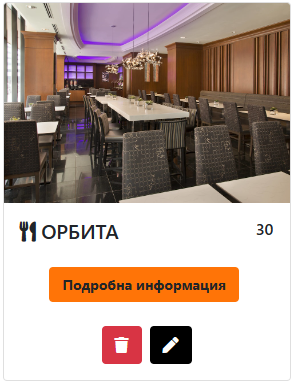
**Регистрация**

Бутонът за регистрация се намира от лявата страна на бутона за логване.

При регистрация се попълват следните полета:

* - имейл
* - парола
* - повтори парола
* **Добавяне на ресторант**
* ****
* При добавянето на ресторанти в уеб приложението се попълват следните задължителни полета:
* - Име на ресторанта – стрингова променлива;
* - Адрес – стрингова променлива;
* - Град – стрингова променлива;
* - Цена – дробно число;
* **-** Снимка – png, jpg, jpeg.

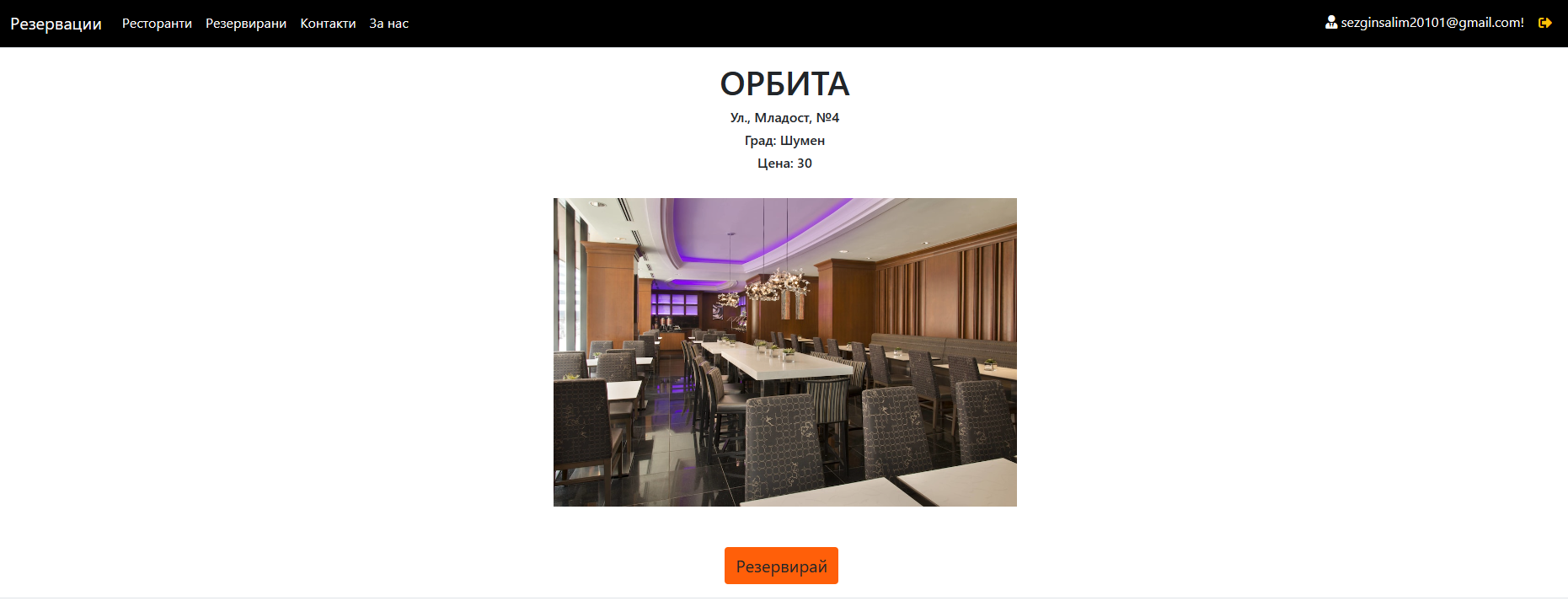
**Визуализация на ресторантите**

****

Собственикът и администратора имат право да редактират и изтриват ресторанти.

При визуализацията на ресторантите излиза една глана снимка, заглавие,цена и бутон за детайлите.

**Резервиране**

****

Резервирането на ресторант е много елементарно. Това става от бутона „Резервирай“.

**3.5. Основни модули и връзки между тях**

Приложението е програмиран чрез платформата .NET и MVC, който включва 3 слоя:

* Модели, Вюта и контролери.

***Потребителски интерфейс*** - това е модулът, който осигурява интерфейса на приложението, който потребителите виждат и използват. Той може да се състои от различни елементи, като например HTML, CSS, JavaScript, изображения, видео и други.

***Управление на потребители*** - този модул се грижи за регистрацията и управлението на потребителите на приложението, като например потребителските профили, автентикацията и авторизацията.

***База данни*** - това е модулът, който се грижи за съхраняването на данните, необходими за работата на приложението, като например данни за потребителите, продуктите, поръчките и други.

***Бизнес логика*** - този модул се грижи за обработката на данните и логиката на приложението, като например обработката на поръчки и плащания, управлението на наличността на продукти, автоматично генериране на фактури и други.

***Административен модул*** - това е модулът, който позволява на администраторите да управляват и контролират приложението, като например добавят или премахват потребители, управляват наличността на продукти, следят статистики за приложението и други.

Връзките между модулите са много важни, тъй като те позволяват на приложението да функционира като цяло. Например, потребителският интерфейс използва бизнес логиката за показване на информацията за поръчките на потребителите, докато базата данни съхранява данните за поръчките. Бизнес логиката използва данните от базата данни, за да обработи поръчките. Административният модул използва бизнес логиката и базата данни, за да управлява наличността на продукти и да добавя или премахва ресторанти.

Обикновено модулите в едно уеб приложение са свързани чрез API (Application Programming Interface), което позволява на тях да комуникират помежду си. Например, когато потребител прави поръчка, потребителският интерфейс изпраща заявка към бизнес логиката чрез API, която обработва поръчката и я записва в базата данни. След това, ако поръчката е успешна, бизнес логиката може да изпрати уведомление за потвърждение на потребителя през потребителския интерфейс.

Когато се проектира уеб приложение, е важно да мислим за взаимодействието между модулите и да ги проектирате така, че да работят ефективно заедно. Това ще помогне на приложението да бъде по-ефективно и да предоставя по-добър потребителски опит.

**3.6 Изисквания за инсталация**

Приложението е публикувано в ГитХъб. Може да се изтегли от там. След изтеглянето му се стартира с Visual Studio. Ако има актуални миграции в проекта, направо се стартира. Но ако няма трябва да се направят миграции.

Когато е публикувано в интернет достатъчно е само да се влезе във връзката.

**4.Архитектура на проекта. Реализация.**

**4.1. Бази данни**

За съхранение на данните, свързани с ресторантите, менюто и резервациите, използвам база данни на SQL Server. Моделът на базата данни е разработен с помощта на подхода Code First на Entity Framework Core. Използвам релационна база данни с три основни таблица:

Restaurants - тук се съхраняват данните за ресторантите, включително името на ресторанта, адресът му, градът, телефонния номер и др.

MenuItems - тук се съхраняват данните за ястията и напитките в менюто, включително името на ястието, описанието му и цената му.

Reservations - тук се съхраняват данните за резервациите, включително името на клиента, датата и часът на резервацията, броя на гостите и др.

**Таблици в приложението**

Restaurants – тази е главната таблица, в която се записват данните на ресторантите. Свързана е с таблиците Image и Reservations(за бъдещо подобрение).

Images – Запазва името и разширението на снимката. А самата снимка се записва в папка img, която се намира в wwwroot.

Reservations – таблица за бъдещо подобрение, в която ще се записват данните на резервиращия и данните зарезервацията.

AspNetRoles – таблицата, в която се записват ролите на потребителите.

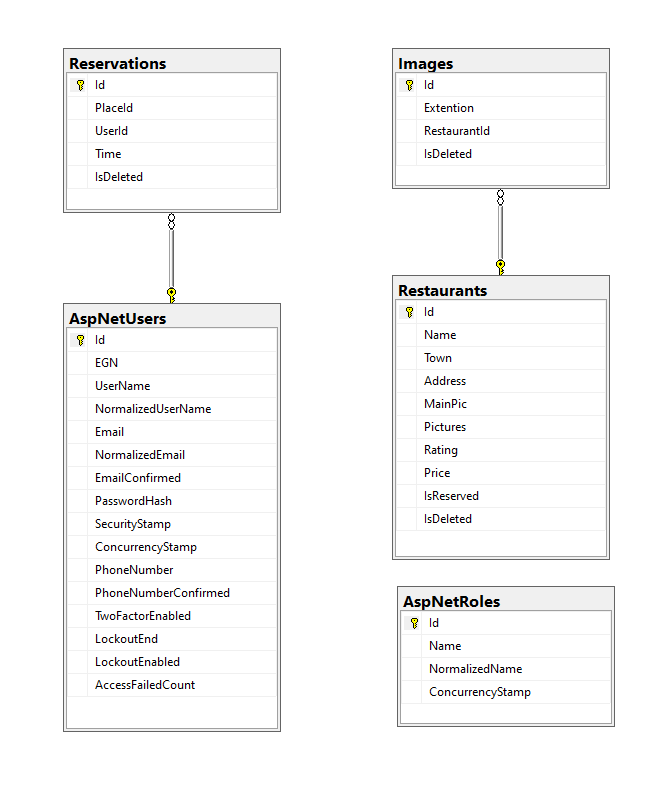
AspNetUsers – в тази таблица се записват данните на потребителите, които правят регистрация в уеб приложението. Тези данни са име, пароле, имейл и много други.

Таблиците в базата данни на уеб приложение са мястото, където се съхраняват данните, необходими за функционирането на приложението. Те представляват организирани данни, подредени в редове и колони, които могат да бъдат лесно търсени и манипулирани.

Таблиците са създадени за съхранение на конкретни видове информация, като например потребители, продукти, поръчки, резервации и др. Всяка таблица има свои полета (колони), които съдържат конкретни типове данни, като текст, числа, дати и други. В таблиците може да има и връзки между тях, като например връзки между таблицата с потребителите и таблицата с поръчките, за да може приложението да знае кой потребител е направил коя поръчка.

Базите данни и таблиците имат важна роля за съхраняването на информацията в уеб приложението и за управлението на данните от страна на приложението. Освен това, те позволяват на приложението да извлича информацията, която е необходима за да показва на потребителите, като например списък с продукти, и да изпълнява различни действия, като добавяне на нови записи или промяна на съществуващи.

**Снимка на базата данни с наличните таблици**

****

**Миграции**

Създаването на миграции и бази данни е важна част от разработката на уеб приложения, които използват релационни бази данни. В тази задача използваме SQL Server база данни и Entity Framework Core Code First подхода за създаване на моделите на данните и базата данни.

Начална подготовка

Първоначално е нужно да имаме инсталиран Microsoft SQL Server и Visual Studio 2022. След това ще инсталираме Entity Framework Core чрез NuGet пакетите.

Създаване на моделите на данните

Създаваме моделите на данните, които ще използваме в приложението. Моделите трябва да отговарят на нуждите на приложението и да съдържат необходимата информация за ресторантите, менюто и резервациите. Примерно, за ресторантите можем да имаме модел с име, адрес, град, телефонен номер, снимка и други полета. За менюто можем да имаме модел с име на ястието, описание, цена и снимка, а за резервациите - име на клиента, дата и час на резервацията, брой гости и др.

Създаване на контекста на базата данни

След създаването на моделите на данните, трябва да създадем контекст на базата данни, който ще използваме в приложението. Контекстът се грижи за свързването с базата данни и за извличането и записването на данни. Той също така може да създаде базата данни, ако все още не съществува.

Генериране на миграции

След като сме създали моделите и контекста на базата данни, можем да създадем миграции. Миграциите са скриптове, които автоматично променят базата данни, за да отразят промените в моделите на данните. Миграциите създават таблиците, колоните и връзките в базата данни.

Когато създадем миграциите, трябва да ги приложим върху базата данни. За да направим това, можем да използваме инструмента за управление на миграции на Entity Framework Core, който е вграден в .NET Core.

За да приложим миграциите, можем да използваме командата "Update-Database" в Package Manager Console. Преди да я използваме, трябва да уверим, че сме настроили правилната връзка с базата данни във файлът appsettings.json и че имаме инсталиран SQL Server и SQL Server Management Studio на компютъра си.

Командата "Update-Database" ще приложи всички миграции, които все още не са били приложени върху базата данни. Ако вече сме приложили миграциите върху базата данни, но сме направили някои промени в модела на данните и искаме да ги отразим в базата данни, можем да създадем нова миграция, която да съдържа промените и след това да я приложим с командата "Update-Database".

Приложаването на миграции може да отнеме малко време, в зависимост от големината на базата данни и броя на миграциите, които трябва да бъдат приложени. В крайна сметка, обаче, ще имаме създадена база данни, която отговаря на нашия модел на данните и която можем да използваме в нашето уеб приложение.

**Релационни връзки**

Релационните връзки са важна част от създаването на бази данни, която позволява свързването на данни от различни таблици, което може да бъде много полезно за съхраняване и обработка на големи обеми от информация. В тази статия ще разгледаме какво са релационните връзки, какви видове съществуват и какви са техните основни характеристики.

*Какво са релационните връзки?*

Релационните връзки са свързващи връзки между две или повече таблици в релационна база данни. Те определят начина, по който записите от една таблица са свързани с записите от друга таблица. Тези връзки могат да бъдат едно към едно, едно към много или много към много.

*Едно към едно връзки*

В едно към едно връзки, всеки запис в едната таблица е свързан с точно един запис в другата таблица и обратно. Този вид връзка е често използван в ситуации, когато има точно една връзка между две таблица, например между таблицата за потребители и таблицата за профилите на потребителите.

*Едно към много връзки*

В едно към много връзки, всеки запис в едната таблица е свързан с много записи в другата таблица, но всеки запис в другата таблица е свързан само с един запис в първата таблица. Този вид връзка е често използван в ситуации, когато има много връзки между две таблица, например между таблицата за ресторанти и таблицата за ястията в менюто на ресторанта.

*Много към много връзки*

В много към много връзки, всеки запис в едната таблица е свързан с много записи в другата таблица и обратно. Този вид връзка е често използван в ситуации, когато има много връзки между две таблица.

**Връзки между таблиците**

*Restaurants и Images – Едно към много.*

От таблицата за ресторантите към таблицата за снимките връзката е от типа много към много. А от снимките към ресторантите е едно към едно. И резултатът става едно към много 1:М.

*AppUser –* клас, който отговаря за потребителите и наследява класа IdentityUser. При създаването на миграция тази таблица не се появява в базата, а добавя нова колона в таблицата за потребителите, а именно към AspNetUsers.

**Как работят релационните връзки в SQL?**

Релационните връзки в SQL се създават между таблиците в базата данни. Те могат да бъдат от тип "един към един" (one-to-one), "един към много" (one-to-many) и "много към много" (many-to-many).

**4.2. Потребителски интерфейс**

Потребителският интерфейс на приложението е разработен с помощта на HTML, CSS и Razor View Engine. Използваме модел-изглед-контролер (Model-View-Controller, MVC) архитектурата за управление на приложението. За дизайна на уеб страниците използваме Bootstrap CSS библиотеката**.**

Архитектурата на проекта е базирана на модел-изглед-контролер (Model-View-Controller, MVC) архитектурния шаблон. В проекта използваме .NET Core, Razor View Engine, и Entity Framework Core за свързване към базата данни. За дизайна на уеб страниците използваме Bootstrap CSS библиотеката за по-лесно стилизиране и управление на съдържанието на страниците.

Приложението включва три основни функционалности: добавяне на ресторанти, правене на резервации и показване на менюто. За съхранение на данните използваме SQL Server база данни, като моделът на базата данни е разработен с помощта на подхода Code First на Entity Framework Core. Има три основни таблица: Restaurants, Images и Reservations.

В проекта има навигация, която позволява на потребителите да се движат между различните страници на приложението. Футърът също е включен и съдържа полезна информация за контакт и други.

С помощта на тези технологии и архитектурен шаблон, приложението позволява на потребителите да добавят ресторанти, да правят резервации и да разглеждат менюта на различни ресторанти, като всички данни се съхраняват в базата данни.

**Основни функционалности**

**Основните функционалности на уеб приложението са:**

Добавяне на нов ресторант с информация за име, адрес, град, снимка и описание.

Преглед на списъка с ресторантите, включително тяхната снимка, име и адрес.

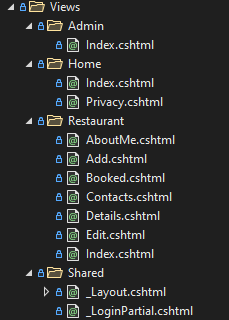
Преглед на менюто на даден ресторант, включително цените на ястията и напитките.

Правене на резервация за даден ресторант.

Преглед на списъка с всички резервации за даден ресторант

Изтриване на ресторант или резервация.

**View-та в проекта**

****

***Admin –*** достъпен е само за администратора.

***Home –*** автоматично генерирани при създаването на проекта.

***Restaurant***

***-*** *Add –* за добавяне на ресторант в приложението.

- *Booked –* за резервации

- *Contacts –* съдържа формуляр за изпращане на съобщения

- *Details –* изгледът за визуализация на детайлите на ресторанта.

- *Edit –* за редактиране на ресторанти.

- *Index –* за визуализация на наличните ресторанти в сайта.

***Shared –*** Папка, която също се генерира при създаването на мвс проекти, съдържаща следните изгледи:

- *Layout –* съдържа общите части на сайта като навигация и футър.

- *\_LoginPartial –* изглед отговарящ за бутоните вход и изход от навигацията.

**4.3 Слой за услуги**

Контролери, екшъни и сервиси са основни елементи в архитектурата на приложенията, използващи Model-View-Controller (MVC) патерн. Те предоставят логика за управление на заявките на потребителите и обработка на данните, въз основа на моделите и изгледите. В тази секция ще разгледаме тези елементи и техните функции подробно.

*Контролери (Controllers)*

Контролерите са класове, които обработват заявките на потребителите и съдържат логика за управление на тях. Те използват HTTP методи (GET, POST, PUT, DELETE и др.) за да обработят заявките и да връщат подходящ отговор. Контролерите в MVC патерна се използват за обработка на HTTP заявки и взаимодействие с моделите и изгледите на приложението.

Контролерите са основната част от архитектурата на MVC и са отговорни за обработката на заявките от клиента. Те обработват заявките и връщат отговорите към клиента.

За да създадем контролер в ASP.NET Core MVC, трябва да създадем нов клас, който да наследява ControllerBase класа. В този клас трябва да създадем методи, наречени екшъни (actions), които да изпълнят функционалността на контролера.

Когато клиентът направи заявка към уеб приложението, ASP.NET Core MVC маршрутизира заявката към определен контролер и екшън, като използва маршрутизационната система.

*Екшъни (Actions)*

Екшъните (actions) в контролерите са методите, които изпълняват дадена функционалност при извикването им от клиента. Те могат да връщат резултат, който да бъде изпратен обратно към клиента.

В един контролер може да има множество екшъни. За да може ASP.NET Core MVC да маршрутизира заявката към правилния екшън, трябва да се зададе правилен маршрут.

*Сървиси (Services):*

Сървисите са класове, които предоставят функционалност, която може да бъде използвана от контролерите и други части на приложението. Те предоставят отделни функционалности, които не са пряко свързани с логиката на контролерите и моделите.

Предимствата на използването на сървиси в ASP.NET Core MVC са:

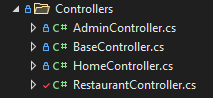
Разделянето на отговорностите между различните класове, което води до по-лесна поддръжка и разширяемост на приложението.

Избягване на повтаряща се логика в контролерите и моделите.

Улесняване на тестването на приложението.

За да използваме сървиси в нашето приложение, трябва да създадем класове, които да наследяват интерфейси и да реализират методите на тези интерфейси.

**Контролери и екшъни в проекта**



Контролерите са AdminController, BaseController, HomeController и RestaurantController.

AdminController – Контролерът, отговарящ за администарторските права.

Екшъни: Index();

BaseController – Базов контролер, който включва юзър мениджър в себе си.

Екшъни: няма, но има полета и конструктор.

HomeController – Контролер, който се генерира при създаването на мвс проект.

Еквшъни: създадени по дефуолт.

RestaurantController – Глваният контролер, който включва в себе си CRUD операциите за създаване, прочитане, редактиране и изтриване на ресторанти.

Екшъни:

- Index() - визуализира създадените ресторанти в сайта.

- Add() – контролер с круд операция за създаване на ресторант.

- Details() – контролер за визуализиране на детайлите на ресторантите по Id.

- Update() – за редактиране на вече съществуващите ресторанти.

- Delete() – за изстриване на ресторант по id.

**CRUD операции**

CRUD операциите са основните операции, които могат да бъдат извършвани върху база данни и са съкращение от Create (Създаване), Read (Четене), Update (Обновяване) и Delete (Изтриване). В тази документация ще разгледаме подробно всяка от тези операции и как можем да ги реализираме в нашето уеб приложение.

*Create (Създаване):*

Операцията за създаване позволява да добавяме нови записи към базата данни. В MVC тази операция може да бъде реализирана чрез HTTP POST заявка, която изпраща данните от формата за създаване към сървъра. В контролера можем да създадем метод с атрибут [HttpPost], който ще приема данните от формата за създаване и ще ги запише в базата данни. Това може да бъде направено чрез извикване на метода Add() на обекта на базата данни и запазване на промените чрез метода SaveChanges().

*Read (Четене):*

Операцията за четене ни позволява да извличаме информация от базата данни. В MVC тази операция може да бъде реализирана чрез HTTP GET заявка, която извлича данните от сървъра и ги визуализира на уеб страницата. В контролера можем да създадем метод, който да връща данните, като използва метода DbSet<TEntity>.ToList() за извличане на всички записи от дадена таблица. След това можем да ги представим на уеб страницата, като използваме HTML, CSS и Razor View Engine.

*Update (Обновяване):*

Операцията за обновяване ни позволява да променяме съществуващи записи в базата данни. В MVC тази операция може да бъде реализирана чрез HTTP PUT заявка, която изпраща променените данни от формата за обновяване към сървъра. В контролера можем да създадем метод с атрибут [HttpPut], който ще приема данните от формата за обновяване и ще ги запише в базата данни. Това може да бъде направено чрез извикване на метода Update() на обекта на базата.

*Delete (Изтриване)*

Изтрива запис от базата данни.

При извършване на операция CRUD е важно да се има предвид също и валидацията на данните. Това означава да се проверява дали входните данни от потребителя отговарят на определени критерии, преди да се извърши операцията. Например, при добавяне на нов ресторант, трябва да се проверява дали името и адресът на ресторанта са валидни, т.е. не съдържат специални символи или числа.

Освен това, при операциите CRUD е важно да се има предвид и обработката на грешки. Ако възникне грешка при извършване на операцията, трябва да се информира потребителя за това и да му се предостави възможност да коригира входните данни или да опита операцията отново.

Когато проектираме приложение, което извършва CRUD операции, трябва да помислим и за сигурността на данните. Необходимо е да се гарантира, че само оторизирани потребители имат достъп до данните и че данните не могат да бъдат модифицирани от злонамерени потребители.

И накрая, при извършване на CRUD операции е важно да се има предвид и производителността на приложението. Ако имаме голям брой данни, операциите могат да бъдат бавни и да забавят работата на приложението. Затова е необходимо да се изберат подходящи алгоритми и да се оптимизират заявките към базата данни, за да се постигне максимална производителност.

**Автентикация и авторизация**

Автентикацията и авторизацията са важни части от създаването на сигурно уеб приложение. В MVC има множество начини за реализация на тези функции.

За автентикацията в MVC може да се използва интегрирания в .NET Core механизъм на Identity, който предоставя готови компоненти за регистрация, вход и управление на потребителски профили. Той може да се интегрира в уеб приложение чрез съответния пакет от NuGet и да бъде настроен за използване с дадената база данни.

С помощта на Identity може да се осъществи вход на потребители в системата чрез потребителско име и парола или чрез външен доставчик на идентичност, като Google или Facebook. Може да се използва и двуфакторна аутентикация, която повишава нивото на сигурност.

За авторизацията, MVC предоставя редица начини за контрол на достъпа до ресурси в приложението. Това може да се постигне чрез атрибути, поставени върху контролери и действия, като [Authorize], които позволяват да се поставят ограничения на потребителите, които могат да имат достъп до определени ресурси.

С помощта на ролите на потребителите и правата, може да се реализира и грануларен контрол на достъпа до отделни действия в приложението.

За повече сложни сценарии на авторизация, може да се използват собствени филтри, които да позволяват на потребителите да имат достъп само до определени ресурси, в зависимост от техните права.

Всички тези функции могат да бъдат използвани за създаване на сигурно и защитено уеб приложение.

**5. Заключение**

Заключението за уеб приложението, което позволява добавяне на ресторанти и правене на резервации, е, че това е много полезен и функционален продукт, който може да бъде използван от потребителите на ресторантите за лесно добавяне на нови ресторанти, създаване на менюта и правене на резервации. Използването на .NET Core и Razor View Engine позволява създаването на ефективно уеб приложение, което работи бързо и е удобно за потребителите.

Бъдещите подобрения на уеб приложението могат да включват добавяне на функции за потребителска регистрация и автентикация, за да се осигури поверителност и сигурност на данните на потребителите. Също така, може да се добавят различни опции за търсене и филтриране на ресторантите и резервациите, за да се улесни навигацията на потребителите. Друго подобрение може да бъде добавяне на функционалност за плащане и онлайн поръчки на храна. Оставяне на отзиви за ресторантите или създаване на профил за потребителите, където да могат да запазват и преглеждат своите резервации. Също така, може да се развие мобилно приложение за по-лесен достъп до приложението.

Като цяло, това уеб приложение е много полезен инструмент за ресторантите и клиентите им, който може да бъде развиван и подобряван за да отговори на нуждите на всички заинтересовани страни.

**Приноси**

Това уеб приложение предоставя много полезна функционалност, която може да помогне на потребителите да открият нови ресторанти, да направят резервации и да разглеждат менютата на ресторантите. Това може да спести много време и усилия при търсенето на подходящо място за хранене.

Приносът на това уеб приложение може да бъде много голям за ресторантьорите, които искат да привлекат нови клиенти. С добавянето на своите ресторанти в приложението, те могат да достигнат до по-голям брой хора и да увеличат видимостта си.

Също така, потребителите могат да споделят мненията си и да дават оценки на ресторантите, което може да помогне на другите потребители да изберат подходящо място за хранене, но тази функция е за бъдещо подобрение на сайта.

**6.Списък на използваната литература**

Galloway, J. (2021). Building Web Applications with ASP.NET Core MVC. Packt Publishing.

C# Yellow Book (2021). Rob Miles.

Microsoft Docs - ASP.NET Core MVC (https://docs.microsoft.com/en-us/aspnet/core/mvc/)

Microsoft Docs - Entity Framework Core (https://docs.microsoft.com/en-us/ef/core/)

Microsoft Docs - Razor View Engine (https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor)

Microsoft Docs - Authentication and Authorization in ASP.NET Core (https://docs.microsoft.com/en-us/aspnet/core/security/authentication/)

Microsoft Docs - SQL Server Management Studio (<https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15>)

**7.    Приложения**

Уеб приложението улеснява работата на ресторантьорите и клиентите. Клиентите вместо да се обадят по телефона, могат направо да влязат в сайта и да разглеждат ресторантите и могат да направят резервации.

Другите предимства на сайтовете за ресторантите са:

Изложба на информацията за ресторанта: Сайтовете на ресторантите предоставят информация за ресторанта, включително менюта, работното време, адреса, контактна информация и др. Това прави по-лесно за потенциалните клиенти да се запознаят с ресторанта и да вземат решение дали да го посетят.

Възможност за онлайн резервации: Сайтовете на ресторантите предоставят възможност за онлайн резервации, което прави процеса по резервация на маса по-удобен за клиентите. Онлайн резервациите могат да се направят в удобно време, без да се налага да се обаждат на ресторанта.

Увеличаване на видимостта: Сайтовете на ресторантите могат да повишат видимостта на бизнеса онлайн. Това може да привлече нови клиенти и да повиши популярността на ресторанта.

Улесняване на комуникацията с клиентите: Сайтовете на ресторантите предоставят възможност за клиентите да се свържат с ресторанта по лесен и бърз начин, например чрез формуляр за контакт или онлайн чат.

Повишаване на лоялността на клиентите: Сайтовете на ресторантите могат да предоставят информация за специални оферти, наградни програми и други инциативи, които могат да повишат лоялността на клиентите и да ги привличат към ресторанта отново и отново.

Анализ на данни: Сайтовете на ресторантите могат да предоставят важна информация за потребителските предпочитания, което може да помогне на управителите на ресторанта да вземат по-добри решения за бизнеса си.

**7.1. Наръчник за работа на потребителя**

Регистрирайте се или влезте в своя профил, за да можете да започнете да правите резервации в ресторанти.

Изберете желания от вас ресторант от списъка на наличните ресторанти в приложението.

Важно е да се отбележи, че някои ресторанти може да изискват допълнителна информация или да имат свои собствени правила за резервации. Поради това е препоръчително да се запознаете с правилата на всяко отделно място за хранене преди да направите резервация. Освен това, не забравяйте да проверявате вашия профил за актуална информация и да отменяте или променяте резервациите си при необходимост.

**7.2. Програмен код**

Kодът е качен в гитхъб - https://github.com/koizafilex123/Reservator