МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

**ПРОФЕСИОНАЛНА ГИМНАЗИЯ “ГЕН. ВЛАДИМИР ЗАИМОВ” гр. СОПОТ**

4330 гр. Сопот, ул. ”Иван Вазов” №1, тел./факс: /03134/ 83-31, 83-32, e-mail: [pgzaimov@yahoo.com](mailto:pgzaimov@yahoo.com)

**ДИПЛОМЕН**

**ПРОЕКТ**

**Тема: „ShiftOn“**

**Уеб сайт, организиращ график на ресторант**

*Ученик: Преслава Маринова Петрова*

***Професия:*** *код 481020 „Системен програмист“*

***Специалност:*** *код 4810201 „Системно програмиране“*

***Консултант****: Живко Маджев*

Сопот, 2023 г.

**Съдържание**

**Увод**

Въведение в предметната област 4

**Първа глава. Проучване**

1. Предпоставка за създаване 5

2. Анализ на съществуващи решения 6

3. Цел на проекта 7

**Втора глава. Проектиране**

1. Използвани технологии и подготовка на работна среда 8

2. Изисквания към софтуерния продукт

2.1. Софтуерни 13

2.2. Интерфейсни 15

2.3. Хардуерни 20

**Трета глава. Описание**

1. Структурата на базата данни 21

2. Структура на MVC приложението ?

3. Описание на CRUD операции ?

4. View модели ?

**Индекс на фигурите** ?

**Използвана литература** ?

**Увод**

**Въведение в предметната област**

„ShiftOn“ е повече от просто инструмент за планиране и проследяване на времето – това е по-добър начин за съвместна работа. Компаниите, които преминават към „ShiftOn“, са по-продуктивни и по-добре организирани. Те комуникират по-добре и са много по-ефективни от преди. Прекарвате по-малко време в планиране и повече време в инвестиране във вашия екип и бизнес. Нашата цел е да ви спестим време и пари, да ви дадем интелигентни прозрения, които да ви помогнат да вземете по-информирани бизнес решения. Нашите шаблони за планиране ви спестяват часове, като ви позволяват бързо да копирате и прилагате графици. Нашият интерфейс за плъзгане и пускане ви дава пълна свобода да правите промени в графика в движение, а нашата функция за заявка за почивен ден дава възможност на вашите служители да разпределят сами почивните си дни. Списъкът с функции е безкраен, както и желанието ни да направим нашия софтуер най-добрият. „ShiftOn“ е инструмент за разрастващ се бизнес, който ви помага бързо да създавате и поддържате непрекъснато развиващия се график на вашия екип. Създадохме го с мисъл за вас, така че да е пълен с удобни функции, от които се нуждаете, без ненужна бъркотия. В основата си планирането на смени е богат на функции календар, който следи кой кои дни работи и по кое време работи. Можете да планирате смени седмици предварително и където е подходящо, можете да копирате смени, за да намалите ръчното въвеждане. След като всички ваши служители са въведени в приложението, можете да започнете планирането на смени. Изграждането на графика на вашият ресторант често е голямо начинание. Всеки има ограничения, предпочитания и промени в последния момент. Ние ви даваме възможност да съставите график за минути, когато настъпят промени, можете лесно да плъзгате и пускате смени в приложението. Няма повече да действате като посредник, да извиквате списъци или безкрайни вериги за одобрение.

**ПЪРВА ГЛАВА.**

***Проучване***

1. **Предпоставка за създаване**

В днешно време организацията на по-големи заведения не е от най-лесните неща, а и изискват много време. Изграждането на графици на служителите на ресторант може да бъде хаотично. Нещата може да се изплъзват през пукнатините – предпочитания за наличност на персонала, заявки за отпуск, обаждания в последната минута. Вече няма да имате от какво да се притеснявате, защото създадохме уеб платформа, която да управлява графика на служителите в ресторант, за да бъде по-лесно и по-удобно управлението на дадено заведение.

Уеб платформата ще бъде лесно за планиране на смените на служителите на ресторант с всички функции, от които се нуждаете, и нито една, която да не ви е необходима. Планирането на смени може да бъде изтощително, за това „ShiftOn“ улеснява целият екип като всеки един служител ще има право на достъп до графика си по всяко време и да разполага с един източник на истина. Изградихме богати на стойност функции като шаблони за смени и възможност за копиране на смени. Ще можете да планирате смени за дни или седмици напред за част от времето. Електронните таблици са чудесни за много неща, но не и за планиране на графици. Те не са удобни за мобилни устройства и липсват функциите, които мениджърите и служителите търсят. Като например смяната на смени, заявките за почивни дни от служителите и възможност за влизане и излизане от уеб сайта като всичко се запазва автоматично. Администраторът ще вижда заявленията за почивни дни на всичките служители в ресторанта и ще може да одобрява или да отхвърля заявленията им според техният график, което прави управлението на смените бързо и безпроблемно.

С това приложение всеки един ресторант и заведение ще могат спокойно да организират служителите си без да им отнема много време. Кажете сбогом на човешката грешка, неявяванията и непредвидените разходи за труд.

1. **Анализ на съществуващи решения**

Има много уеб сайтове за организиране на график за персонала на ресторант и всичките предлагат удобни функции и екипна работа. Ние усъвършенствахме тези функции, като създадохме сайт, който е подобрен и безплатен за персонала и управителите на ресторант. Нашите функции са по-лесни за служителите и по-удобни за реализиране на графиците им. Подобрихме ги, като добавихме и заявки за почивни дни, които служителите на ресторанта могат да използват и трябва само да зададат датата, на която искат да почиват. Единственото нещо, което трябва да направи администратора с тези заявки е да ги одобри или отхвърли при нужда и те автоматично ще се добавят в графика на всички служители. Това улеснява организирането на екипа, защото няма да се налага да се занимава с излишни разговори по телефона, и да губи от времето си за реализация на подобренията върху ресторанта, а и върху екипа си.

Обикновените сайтове за създаване на график са сложни за разбиране и се искат много пари за реализацията на функциите им. Отнема много време да създадете графика си, а благодарение на нашите шаблони за графици ще можете лесно да плъзнете графика, който искате и да го промените по ваш избор. Това помага на управителите на ресторанта да създадат графици, които ще им спестят време и енергия. Те ще могат да създадат графика на служителите за секунди. С този софтуер ще разучим обичайните графици и как можете да започнете да оптимизирате графика за работа на смени, за да повишите производителността във вашата компания, без да е необходимо да разширяват мениджърския си екип. Ръчното планиране на смени и опитът да се следи производителността на всеки работник е огромно оперативно препятствие. Графика ни за смени на служителите на ресторант може да помогне на компанията ви да управлява своите ресурси, да подобри ефективността си и да предостави удобно и безпроблемно обслужване на клиентите.

Сравнение с другите уеб сайтове, „ShiftOn“ предлага лесни функционалности удобни за всеки един потребител на страницата, за да могат компаниите да нямат проблеми с организирането на ресторанта си.

1. **Цели на проекта**

Целта на този проект е да се реализира уеб платформа, която да управлява графика на служителите в ресторант, да подобри организацията на заведението, като създаде по-добра и лесна за използване платформа за създаване на графици.

Софтуерът за планиране на графици на служители на „ShiftOn“ съхранява всичко на едно място. Преглеждане наличността на служителите, отпуските, смените и събитията от едно място. Създаване на графици по ден и седмица. Копирайте графици всяка седмица. Когато влезнем като потребител ще можем да проверяваме графика и смените за седмицата, в която сме на работа, както и да преглеждаме служителите, с които ще работим. Служителите ще могат да пускат заявление за почивен ден, като изпратят на мениджъра датата, в която искат да почиват. Заявлението за почивен ден от служител, може да бъде одобрено или отхвърлено от управителя на ресторанта. Системният администратор на платформата се грижи за управлението на ресторантите и техните управители. Управителят на ресторант е потребител, регистриран от системния администратор, който ще може да преглежда графиците на всичките служители и да ги изтрива и редактира. След редактиране на графика, служителите на ресторанта ще бъдат уведомени онлайн. Подобряване отчетността на персонала, като администратора дава на екипа си 24/7 достъп до графика си онлайн. Информира персонала си, когато графика се промени.

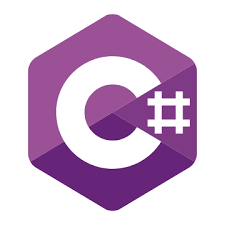
„ShiftOn“ осигурява яснота, когато вашият екип има най-голяма нужда от нея. С онлайн планирането на служители ще получавате изгледи в реално време на графика на екипа и лесно можете да правите промени в движение. Планирането на смени оставя много място за грешки. С нашия уеб сайт сте спокойни, че скъпоструващите грешки в графика, като пропуснати удари, ненужен извънреден труд и ранни влизания, няма да се натрупват. Софтуерът ви позволява да зададете правила, за да насочите вниманието си към проблемите, преди те да се превърнат в проблеми.

**ВТОРА ГЛАВА.**

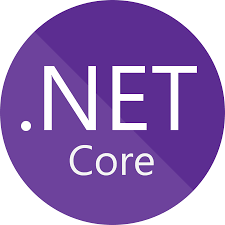
**Проектиране**

1. **Използвани технологии и подготовка на работна среда (Visual studio, C#, Bootstrap, CSS, ASP .Net, .Net Core, Java Script, Html)**
   1. **Картина, която съдържа текст, графична колекция

      Описанието е генерирано автоматичноVisual Studio** е мощна интегрирана среда за разработка на софтуерни приложения от Microsoft. Използва се за разработване на компютърни програми, включително уеб сайтове, уеб приложения, уеб услуги и мобилни приложения. Може да създава както собствен код, така и управляван код. Visual Studio включва редактор на код, поддържащ IntelliSense, както и рафакторинг на код. Интегрираният дебъгер работи както като дебъгер на ниво източник, така и като дебъгер на ниво машина. Други вградени инструменти включват профилиращ код, дизайнер за изграждане на GUI приложения, уеб дизайн, дизайнер на класове и дизайнер на схема на база данни. Той приема добавки, които разширяват функционалността на почти всяко ниво – включително добавяне и поддръжка на системи за контрол на източника и добавяне на нови набори от инструменти като редактори и визуални дизайнери за специфични за домейна езици или набори от инструменти за други аспекти на жизнения цикъл на разработката на софтуер. Visual Studio поддържа 36 различни езика за програмиране и позволява на редактора на код и програмата за отстраняване на грешки да поддържат почти всеки език за програмиране, при условие че съществува услуга, специфична за езика. Характеристики, който Visual Studio използва са редактор на код, дебъгер, дизайнер, редактор на свойства, инструменти за тестване и още много други.



* 1. **C#** е език за програмиране на високо ниво с общо предназначение, поддържащ множество парадигми. Той обхваща статично типизиране, силно типизиране, лексикален обхват, императивно, декларативно, функционално, общо, обектно-ориентирано и компонентно-ориентиране програмиране. C# е обектно-ориентиран език за програмиране, разработен от Microsoft като част от софтуерната платформа .NET. Стремежът още при създаване на C# езика е бил да се създаде прост, модерен, обектно-ориентиран език с общо предназначение. Основа за C# са C++, Java и донякъде езици като Delphi, VB.NET и C. Той е проектиран да балансира мощност (C++) с възможност за бързо разработване.
  2. **Картина, която съдържа графична колекция

     Описанието е генерирано автоматичноBootstrap** е безплатна рамка за [преден край за разработка с](https://www.techtarget.com/whatis/definition/front-end)[отворен код](https://www.techtarget.com/whatis/definition/Free-and-open-source-software-FOSS-or-free-libre-open-source-software-FLOSS) за създаване на уеб сайтове и уеб приложения. Създаден, за да даде възможност за бързо [реагиране](https://www.techtarget.com/whatis/definition/responsive-design) на уеб сайтове, ориентирани към [мобилни устройства](https://www.techtarget.com/searchmobilecomputing/definition/mobile-first) , Bootstrap предоставя колекция от синтаксис за дизайн на шаблони. Като рамка, Bootstrap включва основите за адаптивна уеб разработка, така че разработчиците трябва само да вмъкнат кода в предварително дефинирана мрежова система. Рамката [Bootstrap](https://www.theserverside.com/quiz/12-tough-Bootstrap-5-quiz-questions-for-web-developers) е изградена върху Hypertext Markup Language ( [HTML](https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language)) , cascading style sheets ( [CSS](https://www.theserverside.com/definition/cascading-style-sheet-CSS)) и [JavaScript](https://www.theserverside.com/definition/JavaScript) . Уеб разработчиците, използващи Bootstrap, могат да създават уебсайтове много по-бързо, без да губят време да се тревожат за основни команди и функции. Bootstrap прави отзивчивия уеб дизайн реалност. Това дава възможност на уеб страница или приложение да разпознават размера и ориентацията на екрана на посетителя и автоматично да адаптират дисплея съответно.
  3. **CSS** е език за програмиране и също описание на уеб дизайн програмни стилове (език за стилови файлове, style sheet language) – използва се основно за описание на онлайн представянето на уеббазиран документ, който написан на език за маркиране. Най-често се използва допълнително към чистия HTML, но се прилага и върху XML уебстраници и документи. Спецификацията на CSS официално се поддържа от W3C. CSS още в началото на развитието на www започва да се добавя към стандартния HTML с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Преди стандартите за CSS, установени от W3C през 1995 г., съдържанието на сайтовете и стила на техния дизайн са писани в една и съща HTML страницата.
  4. **ASP .Net** е технология за създаване на [уеб приложения](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) и [уеб услуги](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D1%83%D1%81%D0%BB%D1%83%D0%B3%D0%B0), разработена от „Microsoft “. За първи път е публикуван през януари 2002 г. с версия 1.0 на [.NET Framework](https://bg.wikipedia.org/wiki/.NET_Framework) и е наследник на Microsoft Active Server Pages (ASP) технологията, но не е подобрена версия на ASP. ASP.NET е изградена въз основа на Common Language Runtime (CLR), което позволява на програмистите да пишат ASP.NET код, като използват .NET език по избор. ASP.NET цели производителност спрямо останалите скрипт базирани технологии (включително класическия ASP), като компилира сървърно кода в един или повече DLL файлове на уеб сървъра. Тази компилация става автоматично, когато страницата бива заредена за пръв път (което от своя страна означава, че програмистът не трябва да изпълнява отделни компилации за страниците). Това комбинира лекотата на разработване, предлагана от скриптовите езици, с производителността на бинарните операции.
  5. **.Net Core** е най-новата платформа за разработка с общо предназначение, поддържана от Microsoft. Той работи на различни платформи и е преработен по начин, който прави .NET бърз, гъвкав и модерен. Това се оказва един от основните приноси на Microsoft. Разработчиците вече могат да създават приложения за Android, iOS, Linux, Mac и Windows с .NET, всички с отворен код. Това е рамка с отворен код, между платформи, разработена както от Microsoft, така и от неговата общност. Това е пълна реформа на ASP.NET, която комбинира структурата на MVC и Web API в една рамка. Някои от основните промени включват - междуплатформени ASP .NET приложения на Windows, Mac, Linux; здраво представяне; съвременни инструменти за опростяване на разработката; рамка с отворен код; по-бързо компилиране, асинхронни операции. Основните характеристики на .NET Core са: отворен код, кросплатформен, гъвкаво внедряване, инструменти на командния ред, модулен.
  6. **Java Script** е интерпретируем език за програмиране, разпространяван с повечето уеб браузъри. Поддържа обектно ориентиран и функционален стил на програмиране. Създаден е в Netscape през 1995 г. Най-често се прилага към HTML-а на интернет страница с цел добавяне на функционалност и зареждане на данни. Може да се ползва също за писане на сървърни скриптове JSON, както и за много други приложения. JavaScript не трябва да се бърка с Java, съвпадението на имената е резултат от маркетингово решение на Netscape. Javascript е стандартизиран под името EcmaScript. JavaScript е програмен език, който позволява динамична промяна на поведението на браузъра в рамките на дадена HTML страницата. JavaScript се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до Обектния модел на браузъра.
  7. **Html е** основният маркиращ език за описание и дизайн на уеб страници. 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 ъглови скоби (като например елемента <html>). HTML елементите са основната градивна единица на кода, който изграждат уеб страниците. 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](https://bg.wikipedia.org/wiki/%D0%A1%D1%8A%D1%80%D0%B2%D1%8A%D1%80" \o "Сървър), които са онлайн в интернет или са част от [www](https://bg.wikipedia.org/w/index.php?title=Www&action=edit&redlink=1) мрежата. Тези .html файлове съдържат програмно на таговете на HTML и текстово съдържание със маркери и коментари – също инструкции за [браузъра](https://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80" \o "Браузър), за това какъв точно тип е .html страницата, а също за това как да се показва текстът, особено що се отнася до езиковите характеристики. 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. Това допринася за разделяне съдържанието и структурата на уеб страниците и тяхното визуално представяне.

1. **Изисквания към софтуерния продукт**
   1. **Софтуерни**
2. **Windows Server 2019** е операционна система, която свързва локални среди с услуги на Azure, позволявайки хибридни сценарии, максимизирайки съществуващите инвестиции. Windows Server 2019 има следните функции: Съхранение (директни пространства за съхранение, услуга за мигриране на съхранение, реплика за съхранение, системни прозрения), Сигурност (защитени виртуални машини, подобрена разширена защита от заплахи), Администрация (център за администриране на Windows, SetupDiag).
3. **IIS Express** е лека, самостоятелна версия на IIS, оптимизирана за разработчици. IIS Express улеснява използването на най-актуалната версия на IIS за разработване и тестване на уеб сайтове. Той има всички основни възможности на IIS7 и по-нова версия, както и допълнителни функции, предназначени да улеснят разработката на уеб сайтовете, включително: Не работи като услуга и не изисква потребителски права на администратора за изпълнение на повечето задачи. IIS Express работи добре с ASP.NET и PHP приложения. Множество потребители на IIS Expres могат да работят независимо на един и същи компютър.
4. **SQL Server** е система за управление на релационни база данни, разработена от Microsoft. SQL Server има основна функция да съхранява и извлича данни, по заявки на други софтуерни приложения, които могат да вървят на същия или на друг компютър в дадена мрежа. MS SQL Server има над дузина различни изделия, които са подходящи за различни аудитории и различни натоварвания: от приложения за отделни персонални компютри до масивни приложения работещи постоянно в онлайн режим и обхващащи огромно количество компютри.
5. **SQL Server Management Studio (SSMS)** е софтуерно приложение, което 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. Той позволява на администраторите на база данни и разработчиците на база данни да конфигурират, управляват и администрират всички компоненти в рамките на SQL Server.
6. **Azure WebApps** е базирана на облачни изчисления платформа за хостване на уеб сайтове. Това е платформа като услуга, която позволява публикуване на уеб приложения, работещи на множество рамки и написани на различни езици за програмиране, включително собствени на Microsoft и такива на трети страни. Това е платформа за уеб хостинг, която поддържа множество технологии и езици за програмиране. Потребителите могат да създават уеб сайтове и да внедряват съдържание и код в уеб сайтовете. Поддържа съветник за създаване на уеб сайт или да създаде сайт въз основа на едно от няколко налични предварително конфигурирани изображения от галерията на уеб сайта.
7. **Domain Name System (DNS)** представлява разпределена база от данни за компютри, услуги или други ресурси свързани към интернет или частни мрежи, с чиято помощ се осъществява преобразуването на имената на хостовете в IP адреси. Това улеснява работата на потребителите на интернет услуги. Вместо да въвежда IP адрес, за да достигне до даден ресурс в мрежата, потребителят може просто да въведе неговото име (domain). Информацията за IP адресите и имената на домейни се съхранява на DNS сървърите.
   1. **Интерфейсни**
      1. **Функционални изисквания:**

За да се реализира уеб сайта „ShiftOn“, което организира графика на служители в ресторант, то трябва да поддържа следните функционалности:

Публични страници:

Публичните страници са: Начална страница и Контакти.

* В началната страница ще можем да видим за какво се използва уеб приложението ни, с какво нашето уеб приложение е по-добро и усъвършенствано от другите сайтове за графици на служителите в ресторант, с какви характеристики разполага то. Там ще научите за нашата технология за създаването му. Създадохме го за да е от полза на потребителите, които се затрудняват със организирането на екипа си. Публичната страница е видима за всички потребители, който искат да разгледат уеб сайта ни и да разберат повече за функциите му.
* В страницата Контакти ще можем да видим кои са главните администратори на нашето уеб приложение и също така да се свържем с тях. Там можем да зададем конкретен въпрос по работата на приложението.

Потребителска страница:

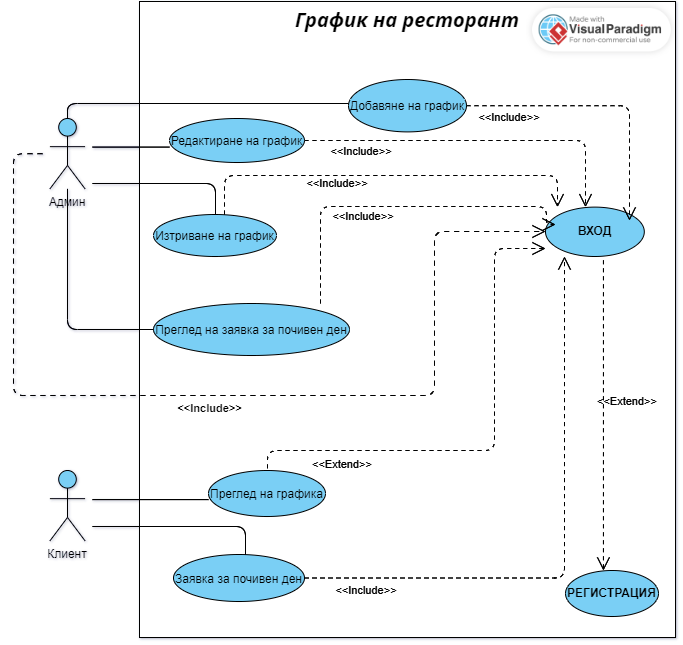
След регистрацията си потребителите ще имат възможност да влезнат в основната част на уеб сайта. Благодарение на многофункционалния ни интерфейс служителите ще могат да разглеждат графика за седмицата. Да проверяват коя смяна са и на каква позиция ще работят, а и с кого са на смяна. Този график ще излиза като таблица, където работните дни ще са оцветени в червено а почивните дни – в зелено. „ShiftOn“ улеснява служителите да поискат почивка за ваканция, събития, поръчки или отпуск по болест, като единственото, което трябва да направят е да избират датата, в която искат да почиват и да изпратят заявката към управителя на ресторанта и да очакват неговия отговор.

Администраторска страница:

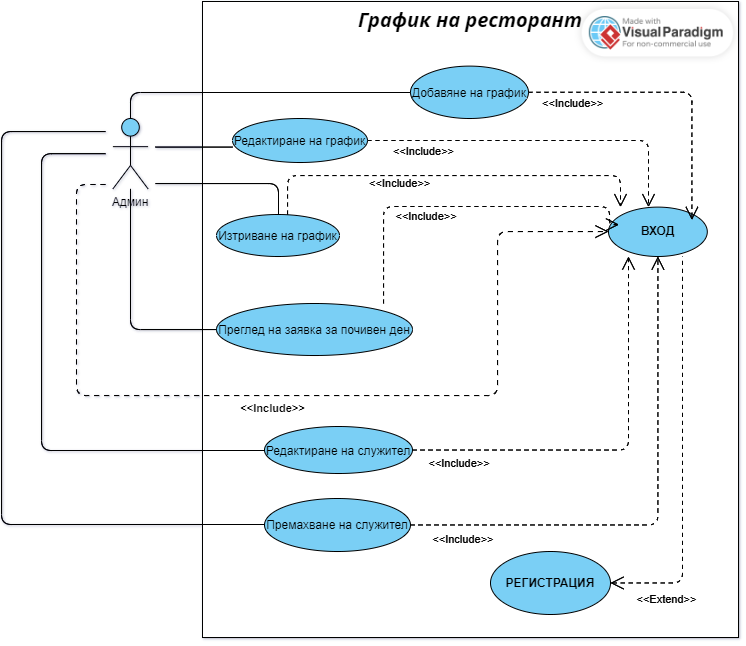
След регистрацията на администратор, той ще трябва да въведе данните на всичките си служите от екипа, за да може да започне съставянето графика им. В администраторската страница ще има таблица, на която са показани всички служители, като той ще има правото да добавя, премахва и редактира служителите на ресторанта. Също така ще има друга таблица, която е графика за седмицата и в него ще има всички служители каква смяна са и кога са на работа. Администратора ще може да променя, добавя и изтрива графика на служителите и да променя тяхната смяна като изтрива или добавя служители в дадена смяна. Той може да позволява на служителите да вземат няколко почивни дни, когато решат, и те автоматично ще бъдат добавени заедно за лесно проследяване. Всички одобрени заявки за отсъствие се преглеждат лесно чрез главната страница с график, като се гарантира, че всички са на една и съща страница. Или да си запазят нещата традиционно като оставят на мениджърите да определят фиксирани часове за почивка за всяка смяна. Администраторите ще могат да преглеждат, редактират, добавят или изтриват почивни дни за служителите. Рационализирането на планирането на служителите става по-лесно, благодарение на шаблоните за графици, удобни за ползване, за да може администраторите лесно да запълнят смените за минути, а не за часове. Проследяването на посещаемостта вече е по-лесно за администраторите. С един поглед ще могат да виждат кой на каква позиция работи.

* + 1. **Use cases Diagrams:**

Диаграмата на случаите на използване е графично изображение на възможните взаимодействия на потребителя със системата. Диаграма на случаи на употреба показва различни случаи на употреба и различни типове потребители, които системата има и често ще бъде придружена от други типове диаграми. Случаите на използване са представени или с кръгове, или с елипси. Актьорите често се изобразяват като фигурки. Самият случай на употреба може да разбие много подробности за всяка възможност, диаграмата на случая на употреба може да помогне за предоставяне на изглед на системата от по-високо ниво.



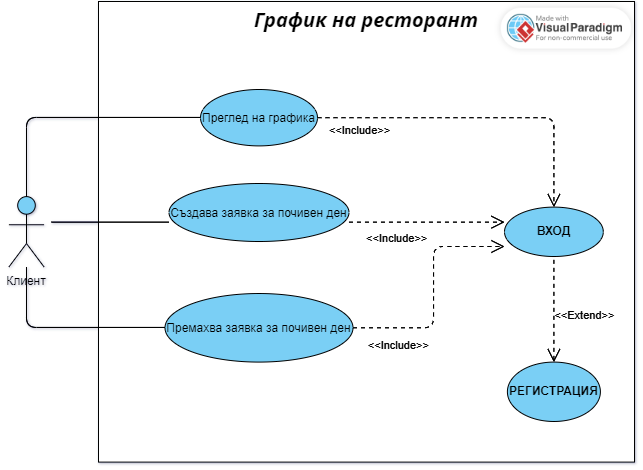
*фиг. … Use-cases diagram*



*фиг.… Admin Usе-cases diagram*

**Случаите на употреба на администратора са:**

* Добавяне на нов график. Целта му е при нужда да се задейства възможността да се добавя нов график. Успешните подусловия са – успешно добавен график, а неуспешните – има вече съществуващ график.
* Изтриване на съществуващ график. Целта му е при необходимост да се изтрие вече съществуващ график. Успешни подусловия – успешно изтрит график.
* Редактиране на съществуващ график. Целта му е да се задейства, когато има нужда от редакция на график. Успешни подусловия – успешно редактирахте графика.
* Преглед на заявките за почивен ден. Целта му е да одобрява или отхвърля заявките за почивни дни на служителите. Успешни подусловия – вие успешно одобрихте/отхвърлихте заявката за почивен ден.
* Вход в системата. Задейства се, когато даден администратор иска да се регистрира в приложението. Предусловията са, че потребителя иска да се регистрира в системата. Успешни подусловия – успешно влизане в системата, а неуспешните – грешно потребителско име или парола.

****

*фиг.… User Use-cases diagram*

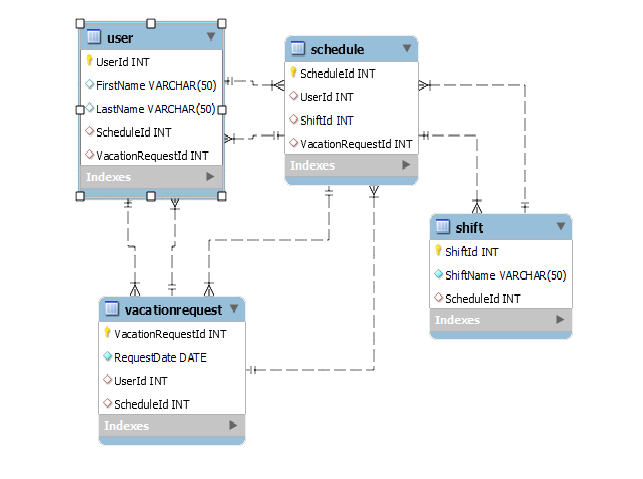
**Случаите на употреба на клиента са:**

* Преглед на графика. Целта му е да се задейства при необходимост, когато клиента иска да прегледа графика си за седмицата. Предусловията са успешно отваряне на системата и влизане в нея. Успешни подусловия – успешно влизане в системата.
* Създава заявка за почивен ден. Целта му е, когато служителя иска да има почивен ден да може да попита работодателя си онлайн. Успешните подусловия са – вие успешно пуснахте заявка за почивен ден, а неуспешните – не можете да почивате на тази дата.
* Регистрация в системата. Задейства се, когато даден потребител иска да използва цялата функционалност на приложението. Успешни подусловия – успешно регистриране в системата, а неуспешните – потребителското име е заето.
  1. **Хардуерни**

**ТРЕТА ГЛАВА.**

**Описание**

1. **Структура на базата данни**

База от данни е интегрирана съвкупност от данни, които са взаимно свързани, съхраняват се съвместно, и с минимално дублиране. Такава база данни осигурява оптимално използване на данни от много потребители отделно или едновременно. База данни се усъвършенства и утвърждава паралелно с развитието на електронно-изчислителна техника и алгоритмичните езици от високо ниво. Заедно с това се създават и системи за управление на база от данни (СУБД). Системи за управление на база от данни са компютърно базирани системи, които осигуряват връзка на потребителя с база от данни.

*фиг.… ER Diagram*

1. Таблица User – съдържа информация за потребителите;

public class User

{

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public Guid UserId { get; set; }

[Required]

[MaxLength(50)]

[DisplayName("First name")]

public string FirstName { get; set; }

[Required]

[MaxLength(50)]

[DisplayName("Last name")]

public string LastName { get; set; }

[Required]

public Guid ScheduleId { get; set; }

[Required]

public int VacationRequestId { get; set; }

public ICollection<Schedule> Schedules { get; set; }

public ICollection<VacationRequest> VacationRequests { get; set;}

}

* UserId – Първичен ключ от тип Guid, който подрежда данните на потребителя по азбучен ред;
* FirstName – Първото име на потребителя, което е задължително и е от тип VARCHAR и с дължина от 50 символа;
* LastName – Фамилното име на потребителя, което е задължително и е от тип VARCHAR и с дължина от 50 символа;
* ScheduleId – Вторичен ключ от тип INT, което е задължително поле и е свързва таблицата User с таблицата Schedule. Това е връзка 1:М (едно към много), където в тази таблица връзката е М.
* VacationRequestId - Вторичен ключ от тип INT, което е задължително поле и е свързва таблицата User с таблицата VacationRequest. Това е връзка 1:М (едно към много), където в тази таблица връзката е М.
  + 1. Таблица Shift – съдържа информация за смените;

public class Shift

{

[Key]

public int ShiftId { get; set; }

[Required]

public string ShiftName { get; set; }

[Required]

public Guid ScheduleId { get; set; }

public ICollection<Schedule> Schedules { get; set;}

}

* ShiftId – Първичен ключ от тип INT, който подрежда потребителя по азбучен ред;
* ShiftName – Името на смяната, което е задължително поле и е тип VARCHAR и с дължина от 50 символа;
* ScheduleId - Вторичен ключ от тип INT, което е задължително поле и е свързва таблицата Shift с таблицата Schedule.
  + 1. Таблица Schedule – съдържа информация за графика;

public class Schedule

{

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public Guid ScheduleId { get; set; }

[Required]

public int ShiftId { get; set; }

[Required]

public Guid UserId { get; set; }

[Required]

public int VacationRequestId { get; set; }

public Shift Shift { get; set; }

public User User { get; set; }

}

* ScheduleId – Първичен ключ от тип INT, който подрежда графика по азбучен ред;
* UserId – Вторичен ключ от тип INT, свързващ таблицата Schedule с таблицата User.
* ShiftId – Вторичен ключ от тип INT, свързващ таблицата Schedule с таблицата Shift.
* VacationRequestId - Вторичен ключ от тип INT, свързващ таблицата Schedule с таблицата VacationRequest.
  + 1. Таблица VacationRequest – съдържа информация за почивните дни;

public class VacationRequest

{

[Key]

public int VacationRequestId { get; set; }

[Required]

[DisplayName("Date")]

public DateTime RequestDate { get; set; }

public Guid UserId { get; set; }

public Guid ScheduleId { get; set; }

public User User { get; set; }

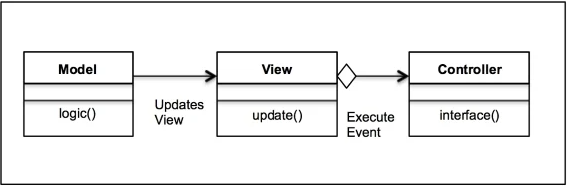
public Schedule Schedule { get; set; }

}

* VacationRequestId – Първичен ключ от тип INT, подреждащ почивните дни по азбучен ред;
* RequestDate – Избраната дата за почивен ден от потребителя, която е задължително поле и е от тип DateTime;
* UserId – Вторичен ключ от тип INT, свързващ таблицата VacationRequest с таблицата User. Това е връзка 1:М (едно към много), където в тази таблица връзката е 1.
* ScheduleId - Вторичен ключ от тип INT, свързващ таблицата VacationRequest с таблицата Schedule. Това е връзка 1:М (едно към много), където в тази таблица връзката е 1.

1. **Структура на MVC приложението**

Model-View-Controller(MVC) / Модел-Изглед-Контролер е архитектурен [шаблон за дизайн](https://bg.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%B7%D0%B0_%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD" \o "Шаблон за дизайн) в програмирането, основан на разделянето на бизнес логиката от графичния интерфейс и данните в дадено приложение.

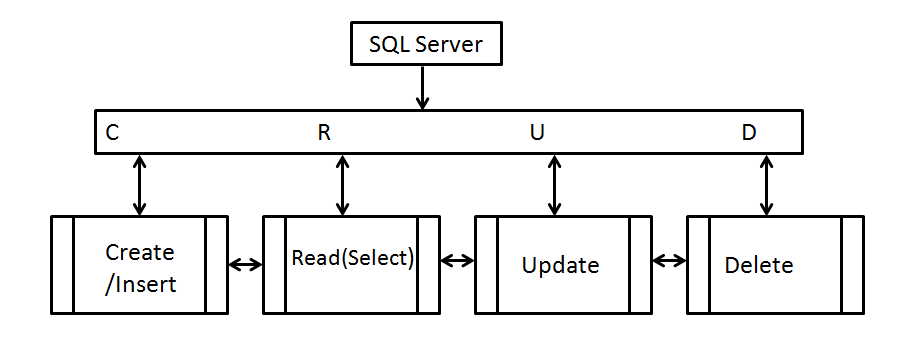


*фиг. ... Model-View-Controller*

* **Модел –** ядрото на приложението, предопределено от областта, за която се разработва. Oбикновено това са данните от реалния свят, които се моделират и над които се работи – въвеждане, промяна, показване и т.н. Трябва да се прави разлика между реалния обкръжаващ свят и въображаемият абстрактен моделен свят, който е продукт на разума, който се възприема като твърдения, формули, математическа символика, схеми и други помощни средства.
* **Изглед –**  тази част от изходния код на приложението, отговорна за показването на данните от модела. Например изгледът може да се състои от [PHP](https://bg.wikipedia.org/wiki/PHP) шаблонни класове, [JSP](https://bg.wikipedia.org/w/index.php?title=JSP&action=edit&redlink=1) страници, [ASP](https://bg.wikipedia.org/wiki/ASP) страници, [JFrame](https://bg.wikipedia.org/w/index.php?title=JFrame&action=edit&redlink=1) наследници в [Swing](https://bg.wikipedia.org/w/index.php?title=Swing&action=edit&redlink=1) приложение. Зависи от това какъв [графичен интерфейс](https://bg.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D0%BD_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) се прави и каква платформа се използва;
* **Контролер –** тази част от сорс кода, взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните, и чак след това ги дава на изгледа. Също така когато се прави уеб графичен интерфейс това би довело до много лесна модификация на [HTML](https://bg.wikipedia.org/wiki/HTML) кода дори от човек, който не е програмист – той ще гледа на шаблона просто като на обикновена HTML страница.

1. **Описание на CRUD операциите**

CRUD функциите обхващат всички основни функции, които трябва да са имплементирани в приложенията, базирани на релационна база данни. Всяка една от буквите на акронима е свързана със Structured Query Language (SQL) изявление – "C" с "Create", "R" с "Read", "U" с "Update" и "D" с "Delete". Съществуват вариации на модела, но всяка една от тях включва тези четири основни функционалности под една или друга форма. Акронимът CRUD идентифицира всички основни функции, които са присъщи на релационните база данни и приложения, използвани за тяхното управление. Четирите CRUD операции могат да извършват различни видове операции върху избрани данни в базата данни. Функциите за създаване, четене, актуализиране и изтриване на ресурси са основни компоненти на използваем модел за съхранение.



*фиг. ... CRUD операции*

1. **Update – променя запис в базата данни**

* Заявка Get

public async Task<IActionResult> Edit(Guid? id)

{

if (id == null || \_context.Users == null)

{

return NotFound();

}

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

return NotFound();

}

return View(user);

}

* Заявка Post

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(Guid id,[Bind("UserId, FirstName, LastName, ScheduleId, VacationRequestId")] User user)

{

if (id != user.UserId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(user);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(user.UserId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(user);

}

1. **Delete – трие записи от базата данни**

* Заявка Get

public async Task<IActionResult> Delete(Guid? id)

{

if (id == null || \_context.Users == null)

{

return NotFound();

}

var user = await \_context.Users

.FirstOrDefaultAsync(m => m.UserId == id);

if (user == null)

{

return NotFound();

}

return View(user);

}

* Заявка Post

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(Guid id)

{

if (\_context.Users == null)

{

return Problem("Entity set 'Context.Users' is null.");

}

var user = await \_context.Users.FindAsync(id);

if (user != null)

{

\_context.Users.Remove(user);

}

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

1. **Read – чете записи от базата данни**

public async Task<IActionResult> Details(Guid? id)

{

if (id == null || \_context.Users == null)

{

return NotFound();

}

var user = await \_context.Users

.FirstOrDefaultAsync(m => m.UserId == id);

if (user == null)

{

return NotFound();

}

return View(user);

}

* + 1. **Create – създава запис в базата данни**
* Заявка GET

public IActionResult Create()

{

return View();

}

* Заявка Post

[HttpPost]

[ValidateAntiForgeryToken]

Public async Task<IActionResult> Create(IFormCollectio collection)

{

if (ModelState.IsValid)

{

var user = new User()

{

UserId = Guid.NewGuid(),

FirstName = collection["firstName"],

LastName = collection["lastName"]

};

\_context.Add(user);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index), user.UserId);

}

return View(User);

}

* 1. **Работа с View модели.**

**View (Изглед)**

Докато контролерът обработва връзката между back-end и front-end, изгледът е визуалния резултат от действията на контролера. Така че, когато искаме да представим на потребителя нещо визуално, а това обикновено е, което искаме, когато разработваме уеб сайт, то се поставя в View. Контролерът генерира обект на модела и след това го предава на изгледа, който след това използва модела, за да представи визуално съдържанието на модела на потребителя. Изгледът използва разширението *.cshtml*, за да посочи допълнителните възможности. Прегледаните файлове обикновено се поставят в папка, наречена Views, в корена на вашия MVC проект. За да улесните .NET рамката и вашите контролери да намерят правилния изглед, създавате подпапка във вашата папка Views за всеки от вашите контролери, носеща името на контролера.

Картина, която съдържа текст, електроника, екранна снимка

Описанието е генерирано автоматичноМестоположението и именуването на вашите изгледи трябва да следват тази конвенция: */View/[NameController]/[Action\_Name].cshtml.* Ако е така, можете просто да извикате метода View() от действията на вашия контролер и да накарате .NET рамката автоматично да намери правилния изглед за вас:

public IActionResult Index()

{

return View();

}

Ако рамката не може да намери съвпадащ изглед с помощта на споменатата по-горе конвенция, тя ще търси на още едно място: */Views/Shared/[Action\_Name].cshtml*. Shared папка, която можете да добавите във вашата папка Views, обикновено се използва за файлове с Layout и Partial изгледи, споделени в множество контролери.

**Разлика в използването на ViewBag и ViewData.**

ViewData и ViewBag се използват за прехвърляне на данни от контролера към изгледа. ViewData е речник на обекти и е достъпен чрез низ като ключ. ViewData е свойство на контролера, което излага екземпляра на класа ViewDataDictionary. ViewBag е динамично свойство. То може да задава и получава динамично стойност и може да добавя произволен брой допълнителни полета, без да го преобразува в строго въведени.

|  |  |
| --- | --- |
| *ViewData* | *ViewBag* |
| Колекция от речници Key-Value | Това е тип обект |
| Речников обект и е собственост на класа ControllerBase | Динамично свойство на клас ControllerBase |
| По-бърз | По-бавен |
| Въведен в MVC 1.0 и е наличен в MVC 1.0 и по-нови версии | Въведен в MVC 3.0 и е наличен в MVC 3.0 и по-нови версии |
| Работи с .NET Framework 3.5 и по-нова версия | Работи с .NET Framework 4.0 и по-нова версия |
| По време на изброяването се изисква код за преобразуването на типа | В дълбочина се използва динамично, така че няма нужда от преобразуване на типа, докато изброявате |
| Стойността му става нула, ако е настъпило пренасочване | Стойността му става нула, ако е настъпило пренасочване |
| Лежи само по време на текуща заявка | Лежи само по време на текуща заявка |

*таблица. ... ViewData / ViewBag*

**Индекс на фигурите**

Фигура ?. Диаграма на случаите на използване / Use cases diagram ?

Фигура ?. Администраторска Диаграма на случаите на използване / Use cases diagram ?

Фигура ?. Клиентска Диаграма на случаите на използване / Use cases diagram ?

Фигура ?. База данни ER Diagram ?

Фигура ?. Модел-Изглед-Контролер / Model-View-Controller (MVC) ?

Фигура ?. CRUD операции ?

Таблица ?. ViewData / ViewBag ?

**Използвана литература**

1. ***Visual Studio -*** *https://en.wikipedia.org/wiki/Visual\_Studio*
2. ***ASP .NET*** *–* [*https://bg.wikipedia.org/wiki/ASP.NET*](https://bg.wikipedia.org/wiki/ASP.NET)
3. ***C#*** *- <https://en.wikipedia.org/wiki/C_Sharp_(programming_language)>*
4. ***CRUD*** *–* [*https://en.wikipedia.org/wiki/Create,\_read,\_update\_and\_delete*](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete)
5. ***HTML*** *- <https://bg.wikipedia.org/wiki/HTML>*
6. ***CSS*** *-* *<https://bg.wikipedia.org/wiki/CSS>*
7. ***JavaScript*** *-* *<https://en.wikipedia.org/wiki/JavaScript>*
8. ***Bootstrap*** *-* *<https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)>*
9. ***Windows Server 2019 -*** *https://en.wikipedia.org/wiki/Windows\_Server\_2019*
10. ***IIS Express -*** *https://en.wikipedia.org/wiki/Internet\_Information\_Services*
11. ***SQL Server -*** *https://en.wikipedia.org/wiki/Microsoft\_SQL\_Server*
12. ***SMSS -*** *https://en.wikipedia.org/wiki/SQL\_Server\_Management\_Studio*
13. ***Azure WebApp -*** *https://en.wikipedia.org/wiki/Azure\_Web\_Apps*
14. ***Domain Name System -*** *https://en.wikipedia.org/wiki/Domain\_Name\_System*
15. ***Use cases diagrams -*** *https://en.wikipedia.org/wiki/Use\_case\_diagram*
16. ***База данни -*** *https://en.wikipedia.org/wiki/Database*
17. ***MVC -*** *https://en.wikipedia.org/wiki/Model\_view\_controller*