

**Природо-математическа гимназия**

**„Васил Друмев“**

**гр. Велико Търново**

ДИПЛОМЕН ПРОЕКТ

за придобиване на трета степен на професионална квалификация

Професия: „Приложен програмист“

Специалност: „Приложно програмиране“

**Тема: Изграждане на комплекс от методи, средства и подходи при разработване на приложение**

**„Споделено пътуване“**

Ученик: Ръководител консултант:

Боян Росенов Божанов Милен Василев

Велико Търново, 2025 г.

Съдържание

[**Увод** 3](#_Toc132804249)

[**Част 1. Технологии за реализиране на уеб-базирани информационни системи** 4](#_Toc132804250)

[**1.1** **Особености на уеб-базираните информационна система** 4](#_Toc132804251)

[**1.2** **Езици за реализиране на уеб приложения** 5](#_Toc132804252)

[**1.2.1 HTML (HyperText Markup Language)** 6](#_Toc132804253)

[**1.2.2 CSS (Cascade style sheets)** 8](#_Toc132804254)

[**1.2.3 Bootstrap** 11](#_Toc132804255)

[**1.2.4 JavaScript** 11](#_Toc132804256)

[**1.2.5 Езикът C#** 13](#_Toc132804257)

[**1.2.6 Технологията ASP.NET** 14](#_Toc132804258)

[**Част 2. Модели и езици за бази данни и СУБД** 16](#_Toc132804259)

[**2.1 Модели и езици за бази данни** 17](#_Toc132804260)

[**2.2 СУБД** 18](#_Toc132804261)

[**Част 3. Реализиране на разработка на приложение „Баш Майстори БГ“** 20](#_Toc132804262)

[**3.1 Design pattern** 20](#_Toc132804263)

[**3.1.1 Repository design pattern** 20](#_Toc132804264)

[**3.1.2 MVC design pattern** 22](#_Toc132804265)

[**3.1.3 Service design pattern** 23](#_Toc132804266)

[**3.2 Модели в приложението „Баш Майстори БГ“** 25](#_Toc132804267)

[**3.2.1 Потребители (AspNetUsers)** 25](#_Toc132804268)

[**3.2.2 Специалност** 27](#_Toc132804269)

[**3.2.3 Заявка** 27](#_Toc132804270)

[**3.3 Контролери** 29](#_Toc132804271)

[**Заключение** 31](#_Toc132804272)

[**Използвана литература** 32](#_Toc132804273)

# **Увод**

С нарастващата нужда от устойчиви и икономически ефективни начини за пътуване, съвременните платформи за споделено пътуване предлагат удобни решения за свързване на шофьори и пътници с общи маршрути. В този контекст представям документацията за проекта „СПОДЕЛЕНО ПЪТУВАНЕ“ – информационна система, предназначена да улесни организацията и участието в съвместни пътувания с автомобил.

Целта на дипломната работа е да се разработи и документира цялостна уеб базирана система, която да обслужва както обикновени потребители (пътници), така и шофьори, желаещи да споделят разходите по своите пътувания. Чрез платформата потребителите могат да разглеждат налични пътувания, да резервират място като спътници или да създадат собствено пътуване като шофьори.

В основата на системата е двукомпонентна архитектура, състояща се от база данни и уеб приложение със слой за бизнес логика, който осигурява комуникация между потребителския интерфейс и данните. Проектът предлага разнообразни функционалности, включително управление на потребителски профили, създаване и преглед на пътувания, кандидатстване за пътуване и администраторски контрол.

В настоящата документация ще бъде представена подробна информация за проектирането, функционалностите и реализацията на „СПОДЕЛЕНО ПЪТУВАНЕ“, както и взаимодействието между различните компоненти на системата. Ще бъде разгледано използването на валидация на данни, падащи менюта, потребителска регистрация и автентикация, съобразено с нуждите и изискванията на проекта.

Тази документация има за цел да предостави ясен преглед на проекта „СПОДЕЛЕНО ПЪТУВАНЕ“ и да подпомогне разбирането на неговата структура, предназначение и възможности. В следващите раздели ще бъдат описани в детайли отделните модули на системата, използваните технологии и постигнатите резултати.

За реализацията на този проект са използвани HTML, CSS, Bootstrap, JavaScript, C#, ASP.NET Core и Entity Framework Core.

# **Част 1. Технологии за реализиране на уеб-базирани информационни системи**

## **Особености на уеб-базираните информационна система**

Уеб информационна система, наричана също и уеб-базирана информационна система, е софтуерна система, която използва уеб технологии за доставяне на информация и услуги до потребители или други информационни системи и приложения. Основната цел на тази система е да публикува и поддържа данни, като използва хипертекстови принципи. [1]

Типичната уеб информационна система се състои от едно или повече уеб приложения, които имат специфични функции, ориентирани към компоненти, заедно с информационни елементи и други не-уеб компоненти. Обикновено front-end на системата е уеб браузър, докато back-end се състои от база данни.

С развитието на уеб технологиите и нарастващия интерес на потребителите и разработчиците, понятието за уеб сайт се развива от прост набор от HTML страници към уеб-базирани информационни системи (WIS). WIS предоставят средства за достъп до комплексни данни и интерактивни услуги чрез Мрежата. Примери за WIS включват е-бизнес приложения, CRM и приложения за веригата на доставки.

Въпреки че традиционните и уеб-базирани информационни системи имат много прилики, съществуват и значителни различия. Уеб-базираните информационни системи представляват различни предизвикателства за разработчика, който трябва да се справя с неизвестни условия на мрежата и потребителите, различни типове данни и други въпроси, като управление на съдържанието, представяне и използваемост. Въпреки че уеб-базираните информационни системи са платформено независими по отношение на информацията за доставка, което е една от причините, поради които те са толкова популярни, уеб разработчиците трябва да се справят с много различни мрежи и да ги вземат под внимание по време на процеса на разработване на системата.

В контраст, потребителите на традиционните информационни системи често са служители, работещи в рамките на една организация или служба. Анализът на изискванията на потребителите за WIS е далеч по-голямо предизвикателство за разработчиците.

## **Езици за реализиране на уеб приложения**

Скриптовите езици се превръщат в програмни езици с общо предназначение поради тяхната гъвкавост и функционалност. Те се използват широко там, където времето на разработчиците е ценно и е необходимо да се избегне дълъг процес на компилация. С тях се пишат програми, предназначени за множество цели, като например:

* Вграждане в уеб страници. Тези програми не се компилират, а се интерпретират от браузъра при зареждането на страницата на клиентския компютър. С помощта на стандартните средства на скрипт езика и на обектите, дефинирани от DOM (Document Object Model), програмистът може да получи достъп до всеки елемент от уеб страницата и динамично да промени неговото съдържание и стилово оформление.
* Реализиране на различни динамични ефекти като показване и скриване на съдържание, придвижване и позициониране на обекти, анимация и т.н.
* Съединяване в едно цяло различни пакети приложни програми.
* Създаване на големи интернет приложения.
* Автоматизиране на различни процеси.

За да се разбере по-добре същността на скриптовите езици, е необходимо да се изходи от второто им наименование - езици за сценарии. Сценарий е последователност от операции, които потребителят може да извършва на компютъра. Сценариите обикновено се интерпретират, а не компилират, което позволява бързо и лесно изпълнение.

Скриптовите езици са програмни езици, които автоматизират определени задачи, които би трябвало да се изпълняват ръчно. Те са особено полезни в случаи, когато времето за разработка на програми е по-ценно от времето за изпълнение на програмата. Скриптовите езици могат да се използват масово в голям брой приложения, като например вграждане в уеб-страници, реализиране на динамични ефекти и създаване на големи Интернет приложения.

Скриптовите езици могат да бъдат динамично изпълними или предварително компилирани. Динамично изпълнимите скриптови езици четат инструкциите от програмния файл на минимални функционални блокове и изпълняват тези блокове без да четат останалия код. Предварително компилираните скриптови езици прочитат програмата в началото, компилират я цялата в машинен код или някакъв вътрешен формат и след това я изпълняват.

Скриптовите езици имат няколко предимства пред готовите програми, транслирани в машинен код. Те имат по-сложен инструментариум и поддържат по-прогресивни техники на програмиране, което съкращава времето за разработка на програми и ги прави по-ефективни. Скриптовите езици позволяват бързо да се откриват грешки в програмата и да се реализират доработки на кода без да се изчаква приключване на компилацията. Освен това, скриптовите езици са сравнително лесни за изучаване, дори и без съответната подготовка.

### **1.2.1 HTML (HyperText Markup Language)**

HTML (HyperText Markup Language) е основният език за описание на уеб страници, който се използва за създаване на графични и текстови елементи на уеб страниците. HTML е стандарт в Интернет, като правилата за него се определят от международния консорциум W3C. Най-новата версия на стандарта е HTML5.

Описание на документа става чрез специални елементи или маркери, които се състоят от етикети или тагове (HTML tags). HTML елементите са основната градивна единица на уеб страниците. Чрез тях се оформят отделните части от текста на една уеб страница, като заглавия, цитати, раздели, хипертекстови препратки и т.н. Най-често HTML елементите са групирани по двойки и имат отварящ таг <h1> и затварящ таг </h1>.

HTML е основната технология, която контролира това, което уеб браузърът показва на екрана.

Основните категории команди в езика HTML, включени във версиите 3.2 и 4.01, са предназначени за специфициране на стилови формати и управление на текстовия поток (Flow Control), включване на графични изображения (Images), създаване на хипервръзки (Links), интегриране на аудио с външни графични обекти (Sound and Maps), вмъкване на видео таг (Video) и на youtube клипове директно от тяхната страница чрез тага (Iframe), създаване на интерактивни формуляри (Forms), разделяне на документа на отделни полета (Frames), включване на външни приложения (Applet), написани на езика Java, и осъществяване на връзка с външни информационни структури (CGI-script).

HTML е език за маркиране, който се използва за създаване на уеб страници и документи. Той позволява на уеб браузърите да интерпретират и показват съдържанието на уеб страници, включително текст, изображения и други материали. HTML таговете се използват за оформление на съдържанието, като се определят характеристики като заглавия, параграфи, линкове и други елементи.

В допълнение към HTML, дизайнерите на уеб страници използват Cascading Style Sheets (CSS), за да задават стилове на съдържанието, като например цветове, шрифтове и разположение. CSS се използва за засилване на дизайна и визуалния аспект на уеб страници.

HTML файловете обикновено се пишат в текстови редактори и се хостват на сървъри, свързани към Интернет. Те съдържат текстово съдържание с маркери, които инструктират браузърите как да показват съдържанието на страницата. Уеб браузърите прочитат HTML документите и ги превръщат в уеб страници, като скриват HTML таговете и показват само съдържанието на страницата.

Едно от основните предимства на HTML е, че документите, създадени по този начин, могат да се разглеждат на различни устройства, включително компютри, таблети и мобилни устройства. Създаването на HTML-базирани уеб страници може да се извършва с помощта на обикновен текстов редактор или специализирани инструменти за създаване на уеб страници, като Microsoft FrontPage, Macromedia Dreamweaver, Notepad и други.

Фигура 1 Пример за HTML форма за създаване на модел “Rating“

### **1.2.2 CSS (Cascade style sheets)**

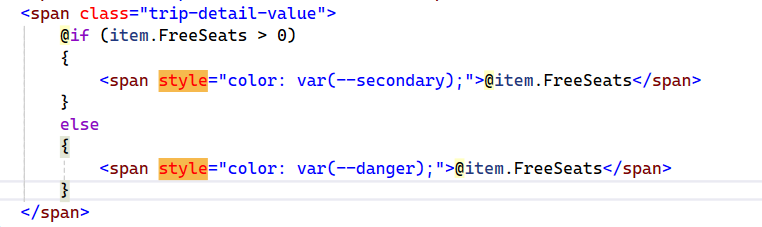
CSS (Cascading Style Sheets) е език за описание на стилове, който се използва основно за определяне на представянето на документи, написани на езици за маркиране (като HTML и XML). Официалната спецификация на CSS се поддържа от W3C (World Wide Web Consortium).

CSS позволява на уеб дизайнерите да определят визуални стилове на HTML елементите, като например шрифтове, размери, цветове, фонове и други. Кодът на CSS се състои от последователност от стилови правила, като всяко правило се състои от селектор, последван от декларации, като всяка декларация се състои от свойство и стойност.

CSS е много удобен за форматиране на текст, шрифтове, списъци, таблици и други елементи на уеб страницата. Комбинацията от селектор и декларации определя стиловете, които ще се приложат върху избраните HTML елементи.

Синтаксисът на CSS е много прост и лесен за разбиране. Всяко правило започва със селектор, който определя върху кой елемент ще се приложат стиловете. След това следват декларации, като всяка декларация е комбинация от свойство и стойност, разделени с двоеточие и заключени в скоби.

Фигура 2 Пример за CSS код

Фигура 3 CSS код приложен чрез “style” атрибут

Ако има повече от един стил, специфициран за един HTML елемент, ще се използва този с най-висок приоритет. Приоритетът е както следва:

1. Стиловете, вградени в HTML елементите с атрибута style.

2. Вътрешните стилове, зададени във <style> елемент в същия HTML файл.

3. Външните стилове, зададени в отделен CSS файл.

4. Стилът по подразбиране на браузъра.

HTML етикетите са първоначално създадени, за да определят съдържанието на документа, като използват синтаксис като <h1> за големи заглавия и <p> за параграфи. С добавянето на нови спецификации към HTML и увеличаването на изискванията към визията на сайта, стана все по-трудно да се създават страници само с HTML. Тук влиза в игра CSS, който ни позволява да разделим стиловете от съдържанието и да създадем по-сложни дизайни. CSS ни позволява да съхраняваме повтарящите се форматирания във външен файл и така да не пишем всеки път атрибутите за тях. Това ни спестява работа и ни позволява да правим по-лесни промени в изгледа на уебсайта.

Синтаксисът на CSS е съставен от три части: селектор (selector), характеристика (property) и стойност (value). (фиг. 2)

Селекторът е нормален HTML елемент (таг), който трябва да се промени, характеристиката е атрибутът за смяна, като всяка характеристика си има стойност. Характеристиката и стойността са отделени с двоеточие и са заградени с фигурни скоби:

div {color: red}

Ако стойността представлява низ съдържащ няколко думи – необходимо е да се огради с кавички:

span {font-family: "sans serif"}

Ако трябва да се специфицира повече от една характеристика, е необходимо да се разделят с “;”.

H1 {text-align:center; color:blue}

Селекторите с еднакви характеристики могат да се групират по между си и така да се спести писане на код. Това става като се разделят със запетая. В следващия пример са групирани всички header елементи. След прилагането на стила съответния текст ще бъде оцветен в зелено:

h1,h2,h3,h4,h5,h6 { color: red }

CSS притежава още едно важно свойство, така нареченият клас Селектор (class selector). С него може да се зададат различни стилове за един и същи HTML елемент в уеб страницата. Ако например имаме два параграфа, единият от тях е с дясно подравняване а другият с централно, с клас селектор това би изглеждало така:

p.right {text-align: right}

p.center {text-align: center}

Друга полезна опция в CSS e id селекторът. Той се различава от класа селектор, по това че последния може да се приложи върху няколко елемента върху страницата, а id селекторът е приложим само за един. Той трябва да е уникален в рамките на уеб страницата. Долу е дефиниран стил, който свързва елемента <h2> с id стойността “heading2”

h2# heading2{text-align: start; color: blue }

Добра практика е като се пише код да се оставят коментари, такава възможност притежава и CSS. Коментарът започва с "/\*" и завършва с "\*/". Всичко между тях се игнорира от браузъра.

### **1.2.3 Bootstrap**

Bootstrap е библиотека с отворен код, която съдържа дизайн шаблони за HTML и CSS, включително компоненти за интерфейса на уеб-приложения и/или уеб-страници, като типография, форми, бутони, навигация и други. Въпреки това, има няколко вградени файлове, които не е препоръчително да се редактират, като файлове с разширения .css и .js. Библиотеката включва няколко JavaScript компонента във формата на jQuery плъгини, които предоставят допълнителни елементи на потребителския интерфейс, като диалогови прозорци и пояснения, и разширяват функционалността на някои от съществуващите елементи на интерфейса.

### **1.2.4 JavaScript**

JavaScript 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 кода 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. JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). JavaScript е най-широко разпространеният език за програмиране в интернет. Прието е JavaScript програмите да се наричат скриптове.

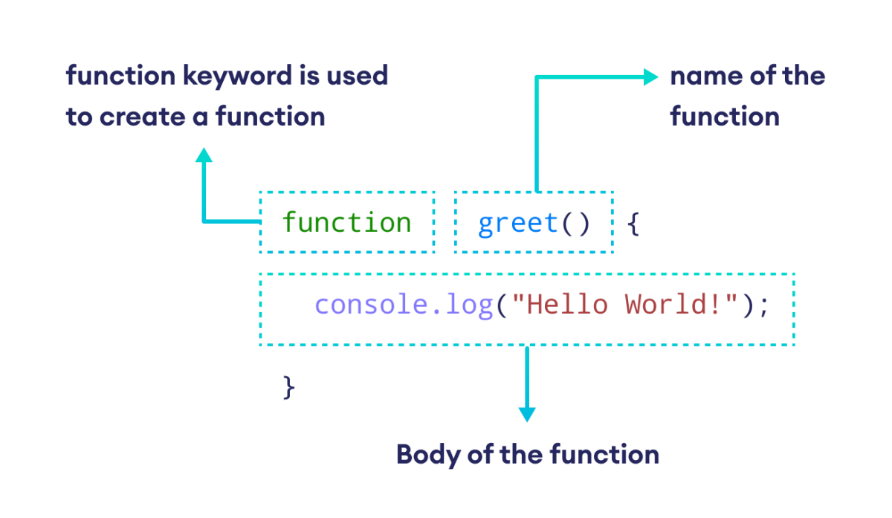
JavaScript може да влияе на почти всяка част от браузъра. Браузъра изпълнява JavaScript кода в цикъла на събития т.е. като резултат от действия на потребителя или събития в браузъра (например document.onLoad).

Основни задачи в повечето JavaScript приложения са:

* Зареждане на данни чрез AJAX.
* Ефекти с изображения и HTML елементи: скриване/показване, пренареждане, влачене, слайд шоу, анимация и много други.
* Управление на прозорци и рамки.
* Разпознаване на възможностите на браузъра.
* Използване на камерата и микрофона.
* Създаване на 3D графики WebGL.
* По-добър и гъвкав потребителски интерфейс

Какво не може да се прави с помощта на JavaScript:

* Не може да се записва информация на потребителския компютър или отдалечения сървър.
* Не може да се запазва информация директно в отдалечена база данни.
* Не може да се стартират локални приложения.



Фигура 4 Пример за JavaScript функция

### **1.2.5 Езикът C#**

C# е език за програмиране, разработен от Microsoft през 2000 година. Той е предназначен да бъде силно типизиран, обектно-ориентиран език, който може да се използва за различни цели - създаване на Windows приложения, уеб приложения, игри, мобилни приложения и други видове софтуер. C# е основен елемент от платформата .NET, която позволява на програмистите да създават приложения, които работят на различни операционни системи.

Езикът C# е синтаксисът на програмиране, който е много подобен на C и C++. Той е дизайниран за да бъде по-модерен и сигурен от тези езици, като същевременно запазва много от техните мощни възможности. Един от ключовите аспекти на C# е, че той е език със строго типизирани променливи, което означава, че програмата може да се компилира само ако типовете на променливите в програмата са коректни.

Освен това C# е езикът, който се използва за създаване на приложения, които се базират на архитектурата на Microsoft Windows. Той е много подходящ за създаване на Windows десктоп приложения и уеб приложения, които използват технологии като ASP.NET и Silverlight. Също така, C# е използван в голяма степен за разработване на игри в платформата Unity, която е популярна за създаване на мобилни игри.

Един от главните достойнства на C# е, че той има много добра интеграция с други езици за програмиране, което позволява на програмистите да използват различни езици и технологии, когато разработват софтуер. Например, C# може да се използва за създаване на програми, които използват бази данни, с помощта на технологии като LINQ (Language Integrated Query).

Също така, C# има богата библиотека от класове, която програмистите могат да използват за да улеснят програмирането. Библиотеката включва класове за работа с графични интерфейси, мултимедийни файлове, мрежови връзки и много други. Освен това, C# разполага с много добра документация и поддръжка от страна на Microsoft, която предоставя на програмистите широк избор от ресурси за учене, като курсове, уроци и форуми.

В последните години, C# се е развивал много бързо и предоставя нови функционалности за програмистите. Например, C# 8.0 представи нова функционалност като nullable типове, изключения от шаблони и асинхронни поточни методи. C# 9.0, който е пуснат през 2020 година, включва нови възможности като записи, модели на образци, улеснена употреба на null безопасност и много други.

C# е изключително мощен език за програмиране, който се използва в много различни области на софтуерното инженерство. Той е изключително полезен за програмисти, които работят върху създаването на мобилни приложения, игри, уеб приложения и Windows приложения. C# има силна поддръжка и развитие от Microsoft, което го прави един от най-предпочитаните езици за програмиране в индустрията.

### **1.2.6 Технологията ASP.NET**

ASP.NET е технология за уеб разработка, разработена от Microsoft, която позволява на програмистите да създават динамични уеб приложения и услуги. Технологията е базирана на програмния език C# или VB.NET и използва Microsoft .NET Framework, който предоставя много библиотеки и функционалности за програмиране на уеб приложения.

ASP.NET предлага различни подходи за програмиране на уеб приложения. Един от тях е ASP.NET Web Forms, който е бил първоначално предназначен да подобри процеса на програмиране на уеб приложения. Web Forms използва модел на програмиране, който е близък до този на програмирането на Windows Forms, където програмистът може да използва графичен потребителски интерфейс и компоненти. Сега, обаче, ASP.NET MVC е по-популярен подход за програмиране на уеб приложения и се състои от модел-изглед-контролер архитектура, която е подобна на модела на програмиране на Ruby on Rails.

ASP.NET MVC позволява на програмистите да разделят приложението на три слоя: модел, който управлява данните и извършва операции с тях, изглед, който представя данните на потребителя и контролер, който се грижи за приемането на заявките от потребителите и извършва съответните операции. Тази архитектура прави приложението лесно поддържаемо и просто за разбиране.

ASP.NET Core е нова версия на ASP.NET, която е направена да бъде по-бърза и лека от предишните версии. Тя е съвместима със множество операционни системи, включително Windows, Linux и macOS, и може да се използва за създаване на уеб приложения, уеб услуги, микросервизи и други.

С помощта на ASP.NET, програмистите могат да създават различни видове уеб приложения, като например онлайн магазини, социални мрежи, уеб услуги, уеб приложения за управление на данни и много други. ASP.NET предоставя множество вградени функционалности, като например автоматично управление на сесии и кеширане на данни. Това улеснява работата на програмистите, тъй като те не трябва да се занимават с тези задачи и могат да се концентрират върху разработването на бизнес логиката на приложението.

Технологията е съвместима с много бази от данни, като например Microsoft SQL Server, MySQL, Oracle и други. Това позволява на програмистите да изберат базата от данни, която най-добре отговаря на нуждите на приложението.

ASP.NET има голяма общност от програмисти, които споделят знания и опит в разработката на уеб приложения. Microsoft предоставя множество ресурси и инструменти, които могат да бъдат използвани за учене и подобряване на уменията в програмирането на уеб приложения.

С помощта на ASP.NET програмистите могат да създадат много гъвкави и мащабируеми уеб приложения, които могат да се използват от малки стартъп компании до големи корпорации. Технологията предоставя много инструменти за сигурност и оптимизация на уеб приложенията, които могат да се използват за защита на данните и оптимизация на производителността на приложението.

Съществуват множество уеб хостинг услуги, които поддържат ASP.NET приложения, което прави деплоя на приложенията много лесен и удобен.

В заключение, ASP.NET е мощна технология за уеб разработка, която предоставя на програмистите голям брой инструменти и функционалности, които им помагат да създават гъвкави, мащабируеми и сигурни уеб приложения. ASP.NET е подходяща за разработка на уеб приложения от различни отрасли, като например онлайн магазини, социални мрежи, уеб услуги и много други.

# **Част 2. Модели и езици за бази данни и СУБД**

База данни представлява колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. Терминът база данни се дефинира като записи, организирани в систематичен ред, който позволява на компютърни програми да извличат информация по зададени критерии. За по-добро извличане и сортиране на данните, всеки запис се организира като множество от елементи на данни (факти). Извлечените елементи в отговор на въпросите (запитванията) стават информация, която може да се използва за вземане на решения.

Системата за управление на бази данни (СУБД) е компютърна програма, която се използва за управление и задаване на въпроси в базите данни. От потребителска гледна точка СУБД трябва да притежава следните свойства:

* Устойчива памет - данните трябва да се съхраняват независимо от процесите; базите данни трябва да позволяват ефективен достъп до много големи обеми от данни - времето на достъп до данните трябва да е независимо от техния обем.
* Програмен интерфейс - СУБД предоставя мощен език за заявки на потребителя или на приложната програма, която използва данните.
* Управление на транзакции - СУБД поддържа едновременен достъп до данните; той се осъществява чрез множество процеси, наречени транзакции; всеки потребител, който работи с данните трябва да инициира такъв процес; СУБД поддържа изолация на транзакциите - всяка транзакция е атомарна, т.е. тя или завършва изцяло или бива изцяло отхвърлена; ако по време на изпълнение на една транзакция възникне проблем тя се отменя, заедно с всички изменения, които е причинила върху базата от данни.

От съществено значение за ефективното използване на базите данни е оптимизацията на заявките към тях. Това се постига чрез изграждане на индекси върху най-често използваните полета в таблиците, които ускоряват търсенето на записи по тези полета.

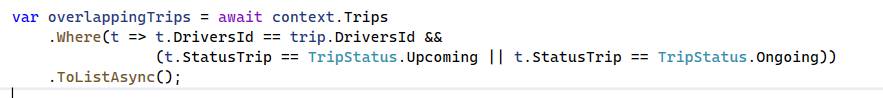
Освен това, в съвременните бази данни се използват различни технологии за съхранение на данни, като релационните, нерелационните и графовите бази данни. Всеки от тези видове бази данни има свои предимства и недостатъци и е подходящ за различни типове приложения.

В заключение, базите данни са от съществено значение за съвременните информационни системи и играят важна роля във вземането на решения в бизнеса и други области. Оттам и необходимостта от постоянно развитие на технологиите за управление на бази данни и оптимизация на тяхната работа.

## **2.1 Модели и езици за бази данни**

Базите данни са съвкупност от организирани данни, които се използват за съхраняване, манипулиране и извличане на информация. Един от най-важните аспекти при проектирането на бази данни е изборът на подходящ модел и език за реализация на базата данни.

1. **Релационни бази данни (RDBMS)**

Този модел е най-често използваният модел за създаване на бази данни. Релационните бази данни работят върху таблици, които имат редове и колони. Всяка таблица има уникален идентификатор, наречен "ключ", който свързва редовете с други таблица чрез "връзки". RDBMS използва SQL език за манипулиране на данните и създаване на заявки.

Фигура 5LINQ заявка (преобразува се в SQL заявка от Entity Framework Core)

Microsoft SQL Server е релационна база данни, която използва SQL езика. Тя е налице само за Windows операционните системи и може да се използва с Microsoft Azure за облачно съхранение на данни. Microsoft SQL Server е използвана в големи корпоративни приложения и включва функционалности за управление на данните, включително сигурност и репликация на данните.

1. **Нерелационни бази данни**

Нерелационните бази данни (NoSQL бази данни) са алтернатива на традиционните релационни бази данни. Те се използват за съхранение на големи масиви от структурирани и/или неструктурирани данни, които не могат да бъдат ефективно съхранени и обработвани с традиционни релационни бази данни. В следващите редове ще представим някои от най-популярните видове нерелационни бази данни:

Документни бази данни: Тези бази данни съхраняват данните в документи, които са обикновено в JSON или XML формат. Те са много гъвкави и удобни за съхранение на неструктурирани данни като текст, снимки, видео и други.

Ключ-стойностни бази данни: Тези бази данни съхраняват данни в стойности, които могат да бъдат достъпвани чрез ключове. Те са много бързи и ефективни при работа с много голям обем данни.

Колонни бази данни: Тези бази данни съхраняват данни в колони, вместо в редове, което ги прави много ефективни при работа с големи обеми данни и търсене на конкретна информация.

Графови бази данни: Тези бази данни съхраняват данни като графи, като връзките между данните са важни. Те са много полезни за съхранение на данни за социални мрежи, транспортни мрежи, връзки между хора и други.

Нерелационните бази данни са много гъвкави и удобни за съхранение на различни видове данни, които не могат да бъдат съхранени ефективно в релационни бази данни. Те са много полезни за приложения, които работят с големи обеми данни, които се променят често и изискват бърз достъп и търсене на информация.

В заключение, изборът на база данни зависи от специфичните нужди на приложението, като се вземат предвид също така и неговите мащабируемост, сигурност и производителност. Различните модели и езици за бази данни предоставят различни функционалности, които могат да се използват за различни видове приложения.

## **2.2 СУБД**

СУБД (Система за Управление на Бази от Данни) е програмен продукт, който се използва за управление на данните в една или няколко свързани бази от данни. Тези бази от данни съхраняват информация, която може да бъде много разнообразна - от текстови документи и изображения до структурирани данни като таблици и списъци. СУБД-тата осигуряват удобен интерфейс за работа с данните, който може да бъде използван от различни потребители - от начинаещи до професионалисти.

Най-известните СУБД-та са MySQL, PostgreSQL, Oracle, Microsoft SQL Server, MongoDB, SQLite и др. Всяко от тях има свои предимства и недостатъци и е подходящо за различни сценарии на използване.

Microsoft SQL Server е СУБД, която е разработена специално за използване на Windows операционни системи. Тя е платена, но се предлага в различни пакети в зависимост от нуждите на потребителите. Microsoft SQL Server е много бърза и мощна, като има вградени инструменти за управление и администриране на бази от данни. Тя е също така съвместима с различни програмни езици и платформи, като .NET и Java.

MongoDB е NoSQL СУБД, която се използва за съхранение на голямо количество неструктурирани данни като документи, изображения и видео файлове. MongoDB е много гъвкава и позволява много бързо и лесно мащабиране на приложенията. Тя е идеална за уеб приложения, които работят с големи количества неструктурирани данни.

СУБД-тата играят много важна роля в съвременните информационни технологии и са необходими за управление на големи количества данни. В зависимост от нуждите на потребителя, той може да избере най-подходящото СУБД за неговите нужди - от малки приложения до големи корпоративни системи. Въпреки че СУБД-тата могат да бъдат доста сложни, те предлагат изключително голяма гъвкавост и функционалност, което ги прави важен инструмент в информационната технология.

# **Част 3. Реализиране на разработка на приложение „Споделено Пътуване“**

Фирма „Рион“ извършва всякакви видове ремонтни дейности за дома в гр. Велико Търново. Управителния съвет взема решение дейността на фирмата да се разшира и в други градове на страната. За релаизиране на тази задача, фирмата се нуждае от приложение, което освен да рекламира дейността ѝ, трябва да осигурява връзка с клиенти.

## **3.1 Design pattern**

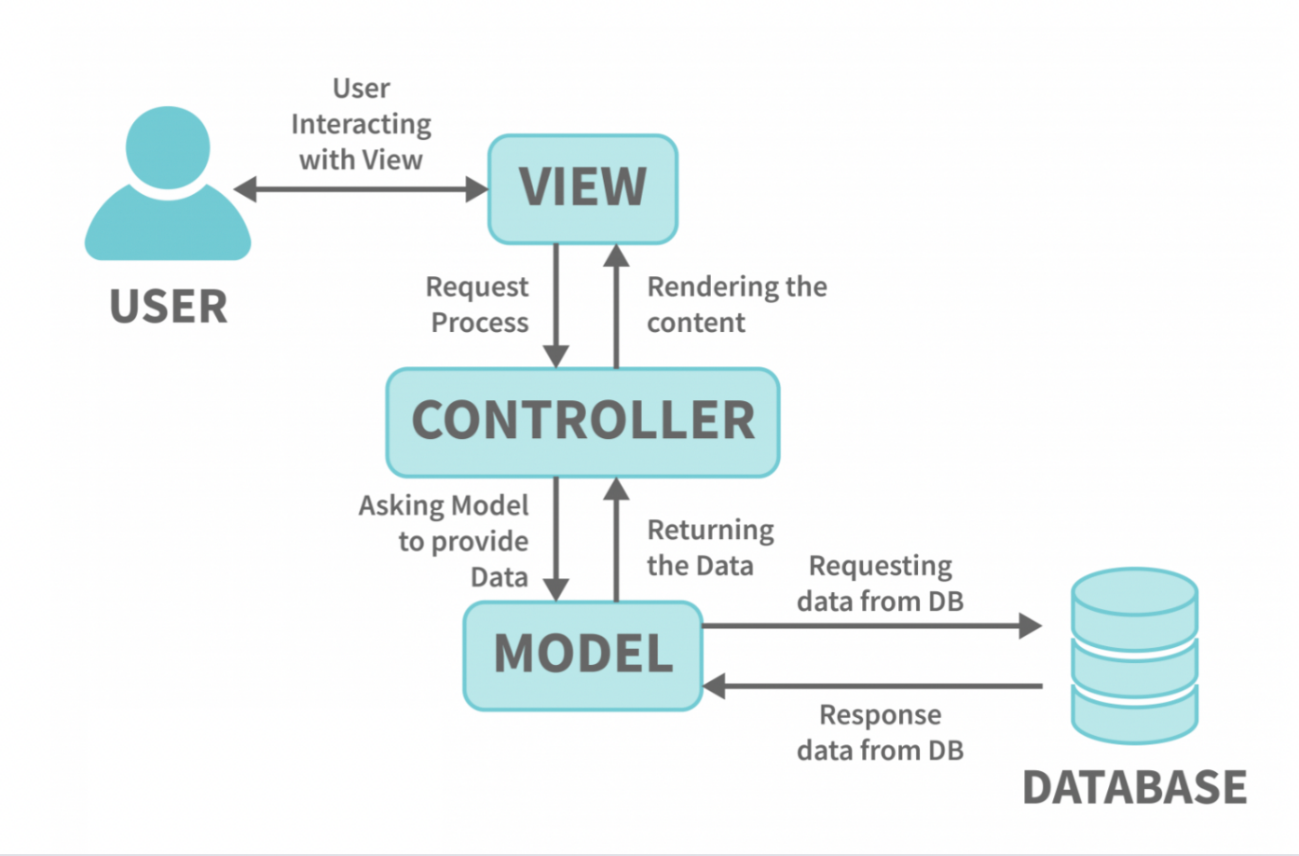
Design pattern (дизайн шаблон) е един препоръчителен начин за решаване на често срещани проблеми в софтуерното проектиране. Те представляват решения на проблеми, които са се появили многократно в различни контексти на даденото софтуерно приложение.

### **3.1.2 MVC design pattern**

MVC е съкращение от "Model-View-Controller" (Модел-Изглед-Контролер) и е един от най-популярните дизайн патерни за програмиране на уеб приложения.

Той разделя приложението на три основни компонента: Модел, Изглед и Контролер.

* Моделът представлява данните и бизнес логиката на приложението. Той обработва всички заявки за данни от Контролера и осъществява комуникацията с базата данни.
* Изгледът е отговорен за визуализацията на данните и общуването с потребителя. Той използва Модела, за да взема необходимата информация и да я представи на потребителя по желания начин.
* Контролерът посредничи между Изгледа и Модела, обработва заявките на потребителя и връща съответните отговори. Той работи с Изгледа, за да осигури правилното визуализиране на информацията и с Модела, за да извършва операции с данните.

MVC е особено полезен за големи приложения, тъй като разделя логиката на приложението на отделни компоненти, което улеснява тестването и поддръжката на кода.

Фигура 6 Структурата MVC

В C# програмирането може да се използва MVC с помощта на ASP.NET Core фреймуърка, който предоставя готова инфраструктура за работа с Модела, Изгледа и Контролера, както и други полезни функционалности за разработката на уеб приложения.

За да се използва ASP.NET Core фреймуъркът, е необходимо да се създаде проект на Visual Studio и да се избере шаблон за ASP.NET Core Web App.

След това може да се започне работа върху Модела, Изгледа и Контролера.

Моделът може да бъде съставен от класове, които представят съответните обекти в приложението и съдържат методи за работа с тях.

Изгледът може да бъде съставен от HTML файлове и Razor шаблони, които използват C# код за визуализиране на данните от Модела.

Контролерът може да бъде съставен от класове, които обработват заявките от Изгледа и извършват нужните операции върху Модела, като връщат подходящи отговори на Изгледа.

ASP.NET MVC фреймуъркът използва маршрутизация за да определи кой Контролер да се използва за обработка на всяка заявка.

MVC е много гъвкав и може да се използва за разработка на уеб приложения от всякакъв тип и сложност. Освен това, той се използва широко в индустрията и предоставя стандартизиран начин за организиране на кода и поддръжка на приложенията.

### **3.1.3 Service design pattern**

Service design pattern (Шаблон за проектиране "Сервиз") е дизайн патерн, който се използва за изграждане на сложни приложения, където е необходима комуникация между различни компоненти или модули.

Service pattern се основава на идеята за изграждане на независими сервизи, които предоставят определени функционалности на останалите части на приложението. Тези сервизи имат ясно дефинирани интерфейси, които други части на приложението могат да използват, за да получат необходимите им данни.

Service pattern обикновено се състои от следните елементи:

* Интерфейс - дефинира методите, които сервизът предлага на другите компоненти на приложението.
* Реализация - включва кода, който извършва необходимите операции и имплементира методите от интерфейса.
* Инжектиране на зависимости - това е механизмът, чрез който приложението може да свърже различни компоненти помежду си и да ги интегрира със сервизите.

Service pattern се използва за улесняване на комуникацията между компонентите на приложението и за повторно използване на кода. Той улеснява тестването и поддръжката на приложението, като осигурява стандартизиран начин за достъп до различни функционалности на приложението.

В C# програмирането Service pattern се използва често в комбинация със Dependency Injection (DI). Dependency Injection е механизъм, който автоматично свързва различните компоненти на приложението помежду си и ги инжектира в нужните места в кода.

Така Service pattern и DI заедно помагат за постигане на добра архитектура на приложението, улесняват мащабируемостта му и увеличават гъвкавостта и поддръжката на кода.

За да бъде успешно използван Service pattern, трябва да бъде правилно дефиниран интерфейсът на сервизите и да се спазват добрите практики за организиране на кода. Това включва използването на правилни имена на класовете и методите, разделянето на кода на подходящи слоеве, дефинирането на подходящи модели на данните и други.

### **3.1.3 Service в приложението “Споделено Пътуване“**

#### **3.1.3.1 TripSchedulerService**

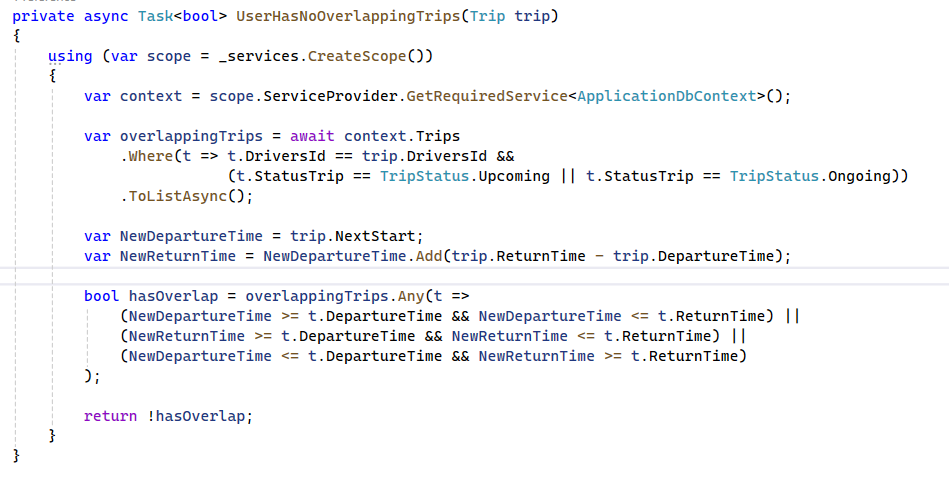
Фоновата услуга в системата има за цел автоматичното създаване на нови пътувания въз основа на предварително дефинирани повтарящи се шаблони. Това осигурява удобство за шофьори, които извършват регулярни курсове, тъй като не е необходимо ръчно да създават всяко пътуване. Основната логика на услугата се изпълнява от метода ExecuteAsync, който работи във фонов режим и се активира периодично на всеки 30 секунди.

При всяко изпълнение, услугата проверява за шаблони на повтарящи се пътувания, при които моментът за следващо изпълнение вече е настъпил. За всеки намерен шаблон системата анализира дали шофьорът, свързан с него, вече има насрочени пътувания, които биха се припокрили по време с новото, предстоящо пътуване. Тази проверка се извършва чрез метода UserHasNoOverlappingTrips.

Ако методът установи, че няма припокриващи се пътувания, се създава ново пътуване, базирано на данните от шаблона, с актуализирани дати и часове. След успешното създаване на пътуването, системата обновява времето за следващо изпълнение на съответния шаблон, така че да е готов за следващото автоматично създаване. Всички промени, включително новите записи за пътуванията и актуализираните стойности в шаблоните, се запазват в базата данни.

Фигура 7 Методът “ExecuteAsync“

Методът UserHasNoOverlappingTrips играе ключова роля в този процес. Неговата задача е да провери дали даден шофьор вече има пътувания, които се припокриват с времето на новосъздаваното пътуване. Ако няма такова припокриване, методът връща стойност true, което позволява на системата да създаде новото пътуване. В случай на припокриване, създаването се пропуска, за да се избегнат конфликти в графика на шофьора.

Фигура 8 Методът “UserHasNoOverlappingTrips“

#### **3.1.3.2 TripStatusUpdaterService**

TripStatusUpdaterService е фоновa услуга, която работи автоматично и има за цел да актуализира статуса на пътуванията според текущото време. Тя се изпълнява периодично, като на всеки 30 секунди проверява базата данни за пътувания, чието състояние трябва да бъде променено.

Основният метод в услугата е ExecuteAsync, който съдържа логиката за актуализиране. Ако е достигнато времето за тръгване на дадено пътуване, неговият статус се променя от „Предстоящо“ (Upcoming) на „В процес“ (Ongoing). Съответно, ако е достигнато времето за връщане, статусът се обновява от „В процес“ на „Завършено“ (Completed).

След всяка такава актуализация, промените се записват в базата данни. Услугата също така логва информация за успешно извършените промени или за възникнали грешки по време на процеса.

Фигура 9 Методът “ExecuteAsync“

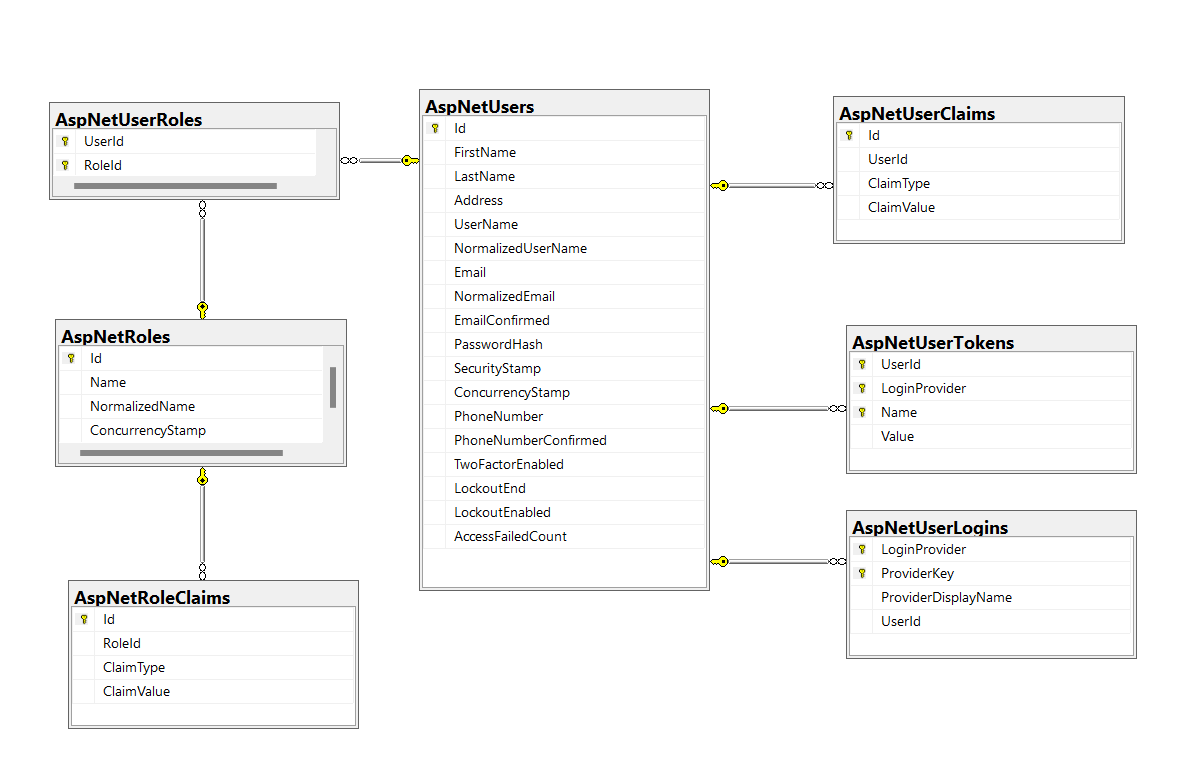
Service pattern е мощен инструмент за управление на сложни приложения и осигуряване на стабилно и мащабируемо приложение. Правилното му използване може да подобри архитектурата на приложението, да улесни тестването и поддръжката му и да ускори разработката на нови функционалности.

## **3.2 Модели в приложението „Споделено Пътуване“**

Модел (Model) - това е компонентът, който управлява данните в приложението. Той се грижи за извличането, обработката и запазването на данните. Моделът може да предоставя данните на контролера или директно на изгледа.  
За тази цел е използван Microsoft SQL Server в комбинация с Entity Framework Core, в контекста на ASP.NET Core приложение. Това позволява създаване на модел на данните, управление на базата данни и извършване на операции като създаване, четене, актуализиране и изтриване на данни по лесен и удобен начин. Entity Framework Core предоставя мощен ORM (Object-Relational Mapping) инструмент, който улеснява взаимодействието с базата данни, като позволява манипулирането на данните с помощта на обекти и класове, вместо да се пишат SQL заявки директно. Базата от данни се състои общо от 11 таблици. 7 от тях са автоматично генерирани и 6 създадени в процеса на разработка.

Седемте автоматично генерирани таблици:

* AspNetUsers – Съдържа информация за потребителите в системата.
* AspNetRoles – Съхранява ролите на потребителите.
* AspNetUserRoles – Установява връзката между потребителите и техните роли.
* AspNetRoleClaims – Съдържа допълнителна информация за ролите.
* AspNetUserClaims – Съдържа допълнителна информация за потребителите.
* AspNetUserLogins – Съхранява информация за външните идентификатори на потребителя.
* AspNetUserTokens – Съхранява токени за потвърждаване на идентичността на потребителите.



Фигура 10 Автоматично създадените таблици

### **3.2.1 Потребители (AspNetUsers)**

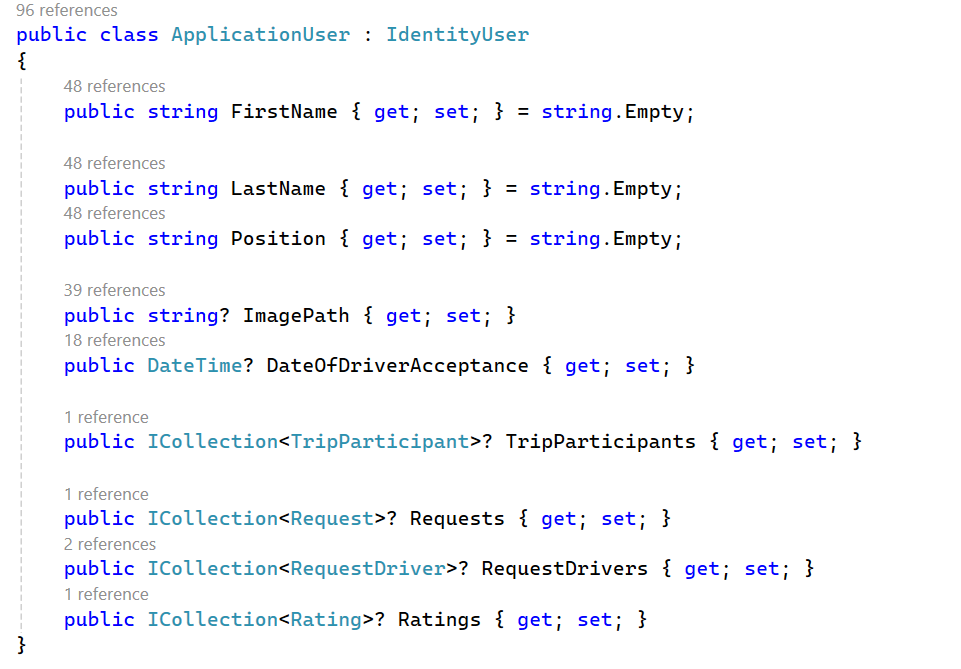
Потребителите, които ASP.NET създава автоматично съдържат следните полета:

* Id
* UserName
* NormalizedUserName
* EmailConfirmed
* PasswordHash
* SecurityStamp
* ConcurrencyStamp
* PhoneNumber
* PhoneNumberConfirmed
* TwoFactorEnabled
* LockoutEnd
* LockoutEnabled
* AccessFailedCount

Ние искаме потребителите да имат следните допълни полета:

* Собствено име
* Фамилно име
* Снимка (ако е шофьор)
* Дата на приемане (когато е станал шофьор)
* Роля (за нея отговаря автоматично създадена таблица)

За целта създаваме клас ApplicationUser, в който добавяме допълнителните полета. Класът ApplicationUser наследява класа IdentityUser, който е дефаултния клас за потребители от ASP.NET.

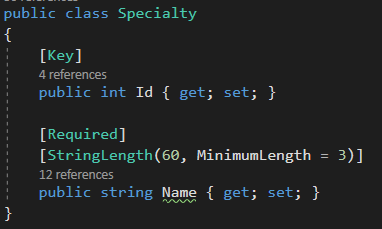


Фигура 11 Моделът “ApplicationUser“

### **3.2.2 Специалност**

Специалността се характеризира с:

* Название

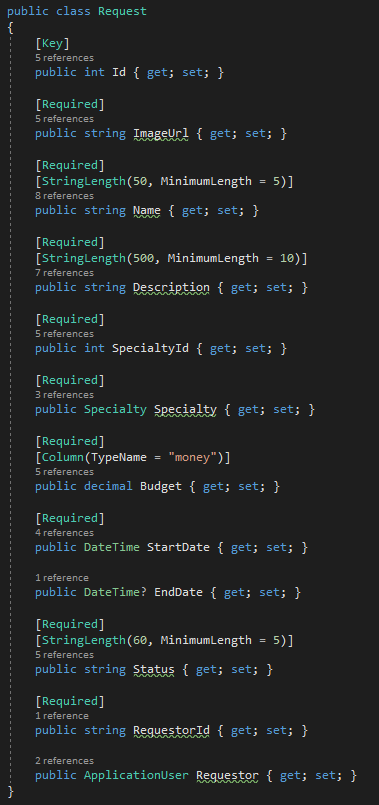


Фиг. 9 Класът Specialty

### **3.2.3 Заявка**

Заявката се характеризира с:

* Възложител
* Описание
* Тип специалист (избира се от клиент)
* Бюджет на възложителя
* Дата на подаване
* Дата на завършване
* Статус

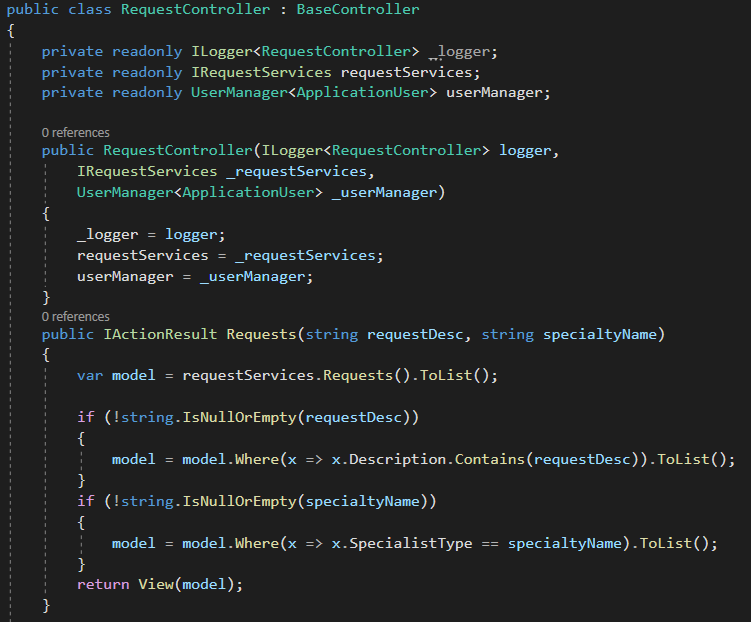


Фиг. 10 Класът Request

## **3.3 Контролери**

Контролер (Controller) - това е компонентът, който управлява логиката на приложението. Той приема заявките от потребителите и обработва данните, които са въведени. Контролерът изпраща заявките към модела, за да получи нужните данни и след това ги предава на изгледа.

В контролерите на проекта се извикват сървисите за дадения контролер, в които е реализирана по-голямата част от логиката. Във фигури 12 и 13 може да видите част от бизнес логиката реализирана в контролер от приложението:



Фиг. 12



Фиг. 13

# **Заключение**

Приложението "Баш Mайстори БГ", написано на C#, е много полезен софтуерен продукт, който предоставя възможност на потребителите да намерят майстори за различни видове строителни и ремонтни дейности.

Приложението има ясен и лесен за използване потребителски интерфейс, който предоставя на потребителите удобен начин за създаване на ремонтни заявки и свързване с квалифицирани майстори.

Системата на приложението е базирана на модел-изглед-контролер (MVC) архитектура, която прави кода лесен за поддръжка и разширение. Моделите в приложението съхраняват информацията за ремонтните заявки, потребителите и категориите на заявките. Така потребителите могат да пояснят от какъв специалист имат нужда.

Приложението е създадено с използването на C# и .NET фреймуърк, което гарантира стабилност и бързина на приложението. Кодът е написан съгласно най-добрите практики в софтуерното инженерство, като приложението е тествано и оптимизирано за най-добро изпълнение.

Като предложения за доразвиване мога да спомена следното:

* Качване на проекта и базата му данни в облачното пространство
* Осъществяване на връзки с реални фирми
* Осъществяване на по-добра комуникация с клиентите

# **Използвана литература**

1. Иванов И, „Разработване на уеб-базирана софтуерна система“, София, 2015
2. Джон Дъкет, „HTML & CSS: Design and Build Web Sites“
3. Светлин Наков, "Принципи на програмирането със C#" / "Въведение в програмирането със C#"
4. http://www.w3schools.com/
5. https://getbootstrap.com/

5. <https://www.microsoft.com/en-us/sql-server>

6. <https://developer.mozilla.org/en-US/docs/Glossary/MVC>

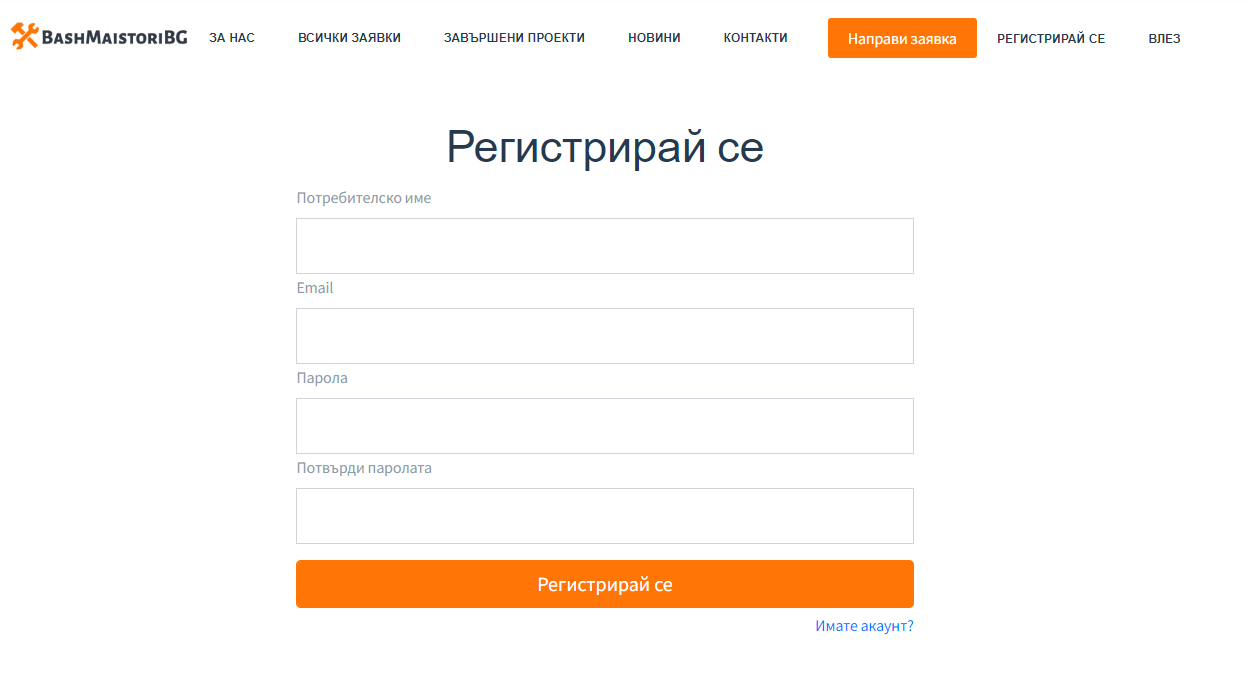
**Работа с приложението „Споделено пътуване“**

**Регистрация**

Страницата за регистрация в сайта предлага възможност на потребителите да създадат нов акаунт. За да се регистрират, потребителите трябва да въведат своите потребителско име, Email, парола и да я потвърдят в полето за повторение на паролата. След въвеждане на данните, системата ще извърши проверка за валидността на въведения. Ако има грешка във въведените данни, потребителят ще бъде уведомен за грешка и му се изискват корекции.

За да се гарантира висока сигурност на данните, паролата трябва да бъде достатъчно силна. Затова системата изисква паролата да съдържа поне една главна буква (A-Z), поне една малка буква (a-z), поне една цифра (0-9) и поне един знак (!;?;...) Ако паролата не отговаря на тези изисквания, системата ще помоли потребителя да я промени, за да отговаря на стандартните параметри.

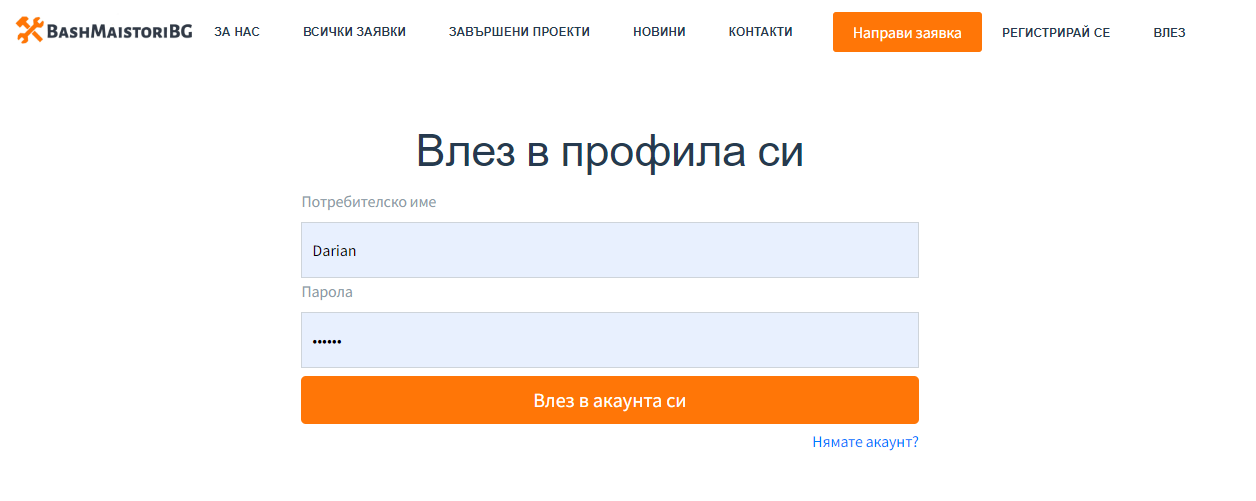
След успешна регистрация, потребителите ще може да влязат в своя акаунт и да започнат да използват функционалностите на сайта. Бутонът "Регистрирай се" ще изпрати заявката за регистрация на потребителя и ще го препрати към страницата за вход, където той ще може да влезе в системата със своя акаунт.

****

Фиг. 14

**Вход**

Страницата за вход в сайта предоставя възможност на потребителите да влязат в своя акаунт. Ако вече имат регистриран акаунт, могат да използват опцията "вход като регистриран потребител", където да въведат своите данни за вход, като email адрес и парола, и да влязат в системата. В случай, че потребителят все още не е регистриран, може да натисне линка "Нямате акаунт?" и да бъде пренасочен към страницата за регистрация, където да създаде нов акаунт като въведе своите потребителско име, email адрес и желана парола, като тя трябва да отговаря на изискванията за силна парола, включваща поне една цифра, една малка буква и една главна буква и един знак. След успешен вход в системата, потребителят ще може да използва функционалностите на сайта. В случай на неуспешен вход, потребителят ще бъде уведомен за грешка.



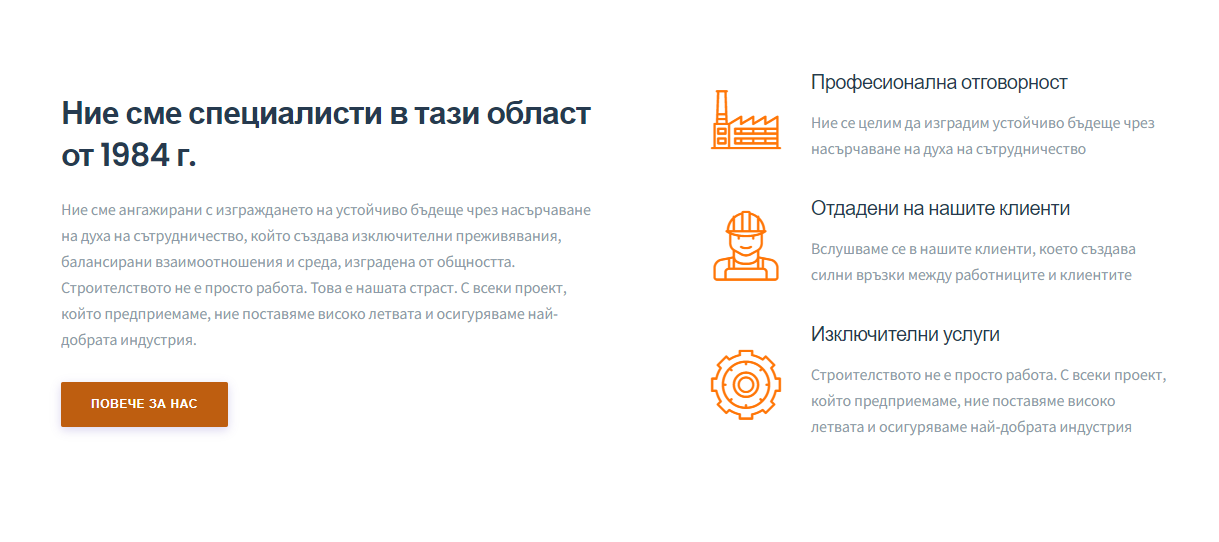
Фиг. 15

**Начална страница**

Приложението „Баш Майстори БГ“ е уеб-приложение за фирма „Рион“, която извършва всякакви видове ремонтни дейности за дома в гр. Велико Търново. На началната страница има обща информация за фирмата и препратки към страниците „За нас“ и „Завършени проекти“.



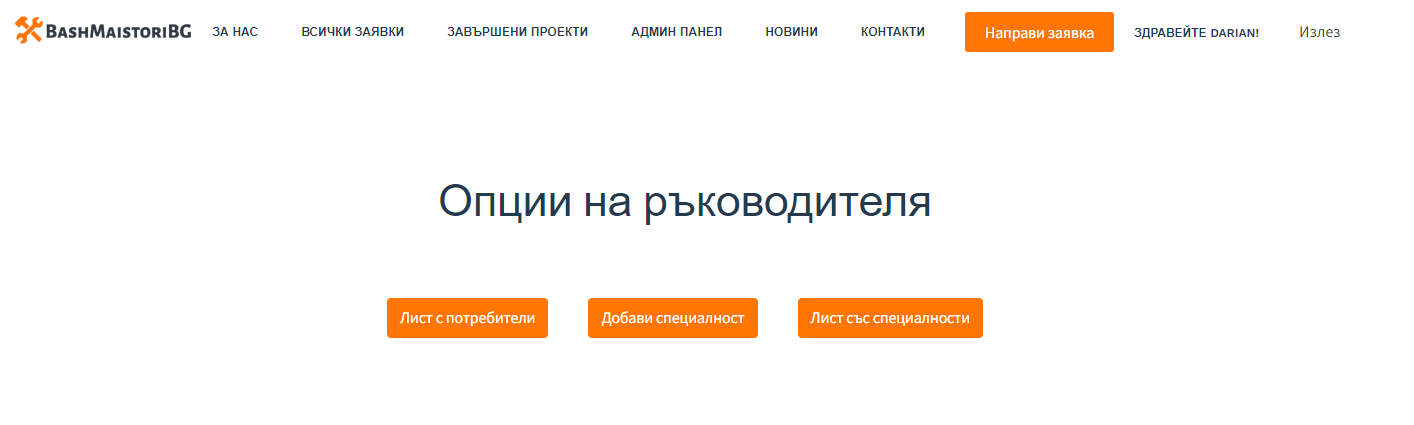
Фиг. 16



Фиг. 17

**Админ панел (опции на ръководител)**

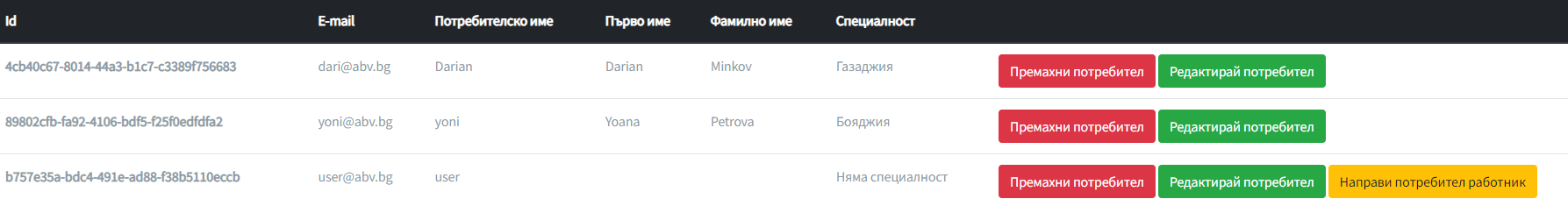
На страницата с опциите на ръководителя може да видите 3 бутона, „Лист с потребители“, „Добави специалност“ и „Лист със специалности“. Администраторът (ръководителят) може да редактира, трие, повишава клиенти в служители и да понижава служители в клиенти. Също така само той може да добавя, трие и редактира специалности.

****

Фиг. 18

**Лист с потребители**

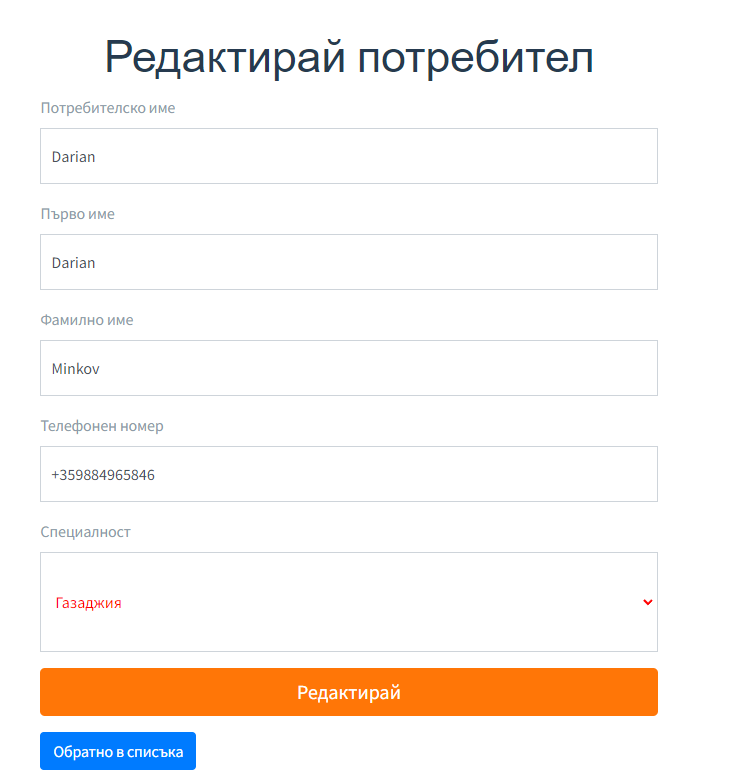
Когато ръководител натисне бутона „Лист с потребители“ той бива отпратен на страницата от фигура 19. Там той има възможност да редактира, да премахне или да повиши или понижи даден потребител.



Фиг. 19

**Редакция на потребител**

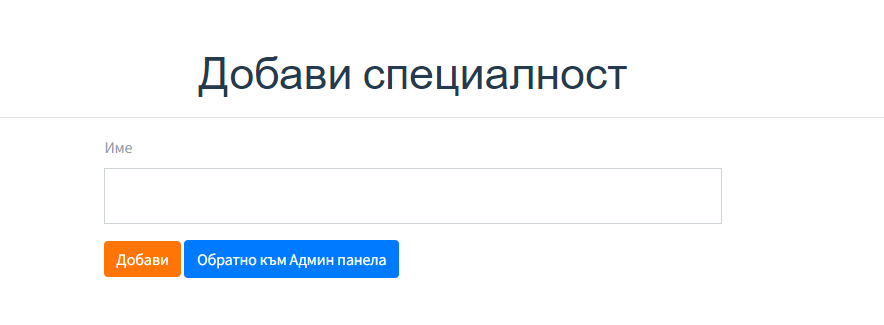
При натискане на бутона „Редактирай потребител“, на екрана се показва страницата на фигура 20. Там един администратор (ръководител) може да променя данните на даден потребител. Съществува бутон „Обратно в списъка“, чрез който администраторът се връща към списъка с всички потребители.



Фиг 20.

**Добави специалност**

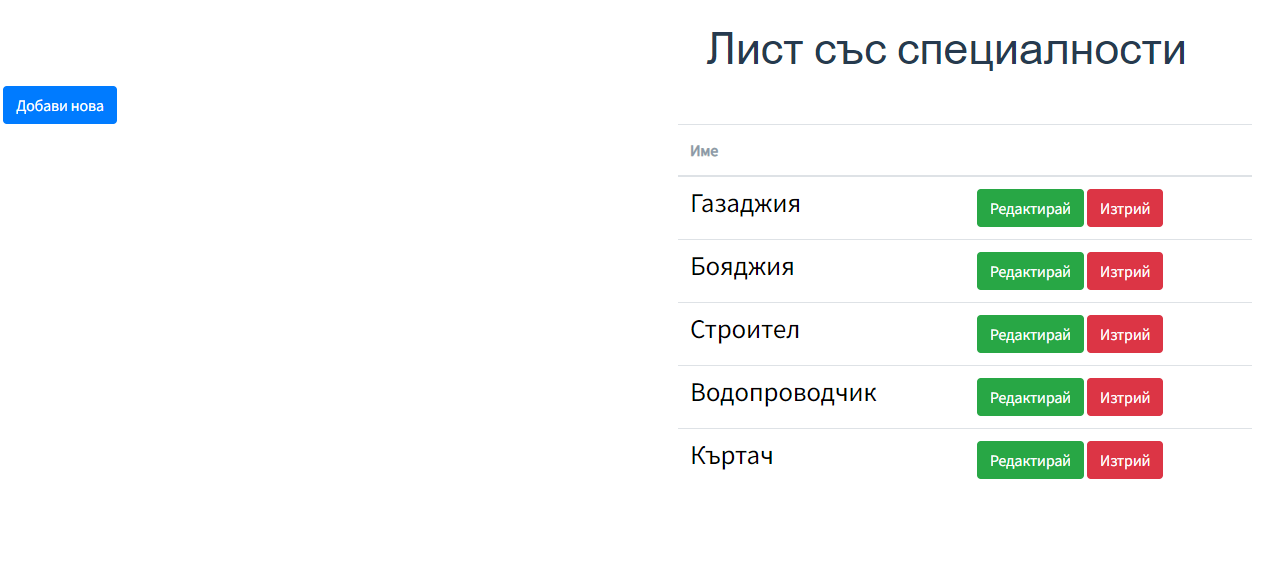
Всеки ръководител може да добавя специалности в базата данни, които са важни за определянето и разграничаването на майсторите и техните задачи и нуждите на клиентите.



Фиг. 21

**Лист със специалности**

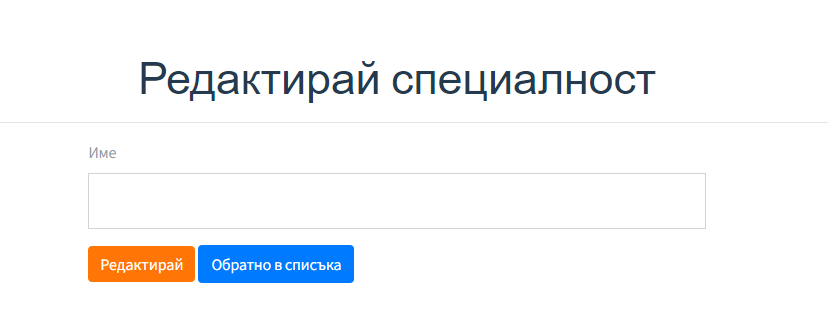
Администраторите могат да преглеждат всички специалности в един списък. Така се улеснява прегледа на всички специалности от ръководителите. Бутонът „Добави нова“ ни отвежда към страницата за добавяне на специалност.



Фиг. 22

**Редактиране на специалност**

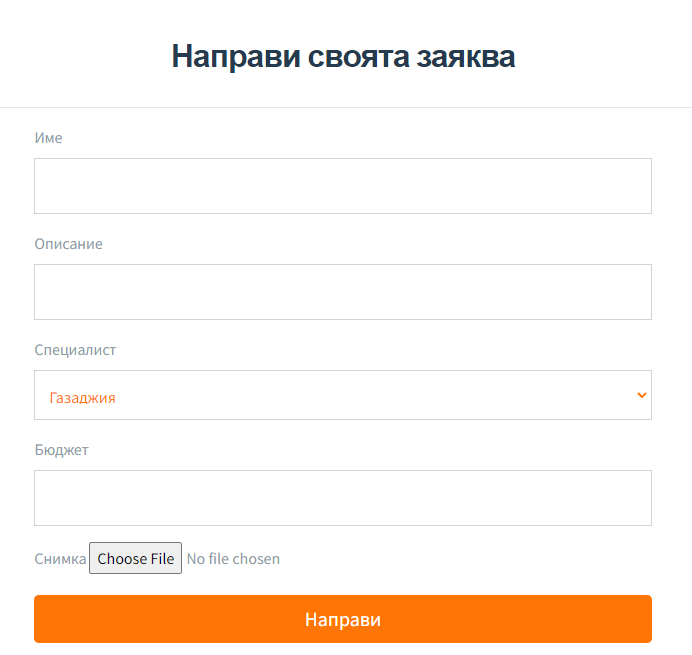
Администраторите (ръководителите) могат да редактират наимонаванието на дадена специалност чрез бутона „Редактирай“ от листа със специалности. Бутонът „Обратно в списъка“ ни отвежда в списъка със специалности.



Фиг. 23

**Създаване на заявка**

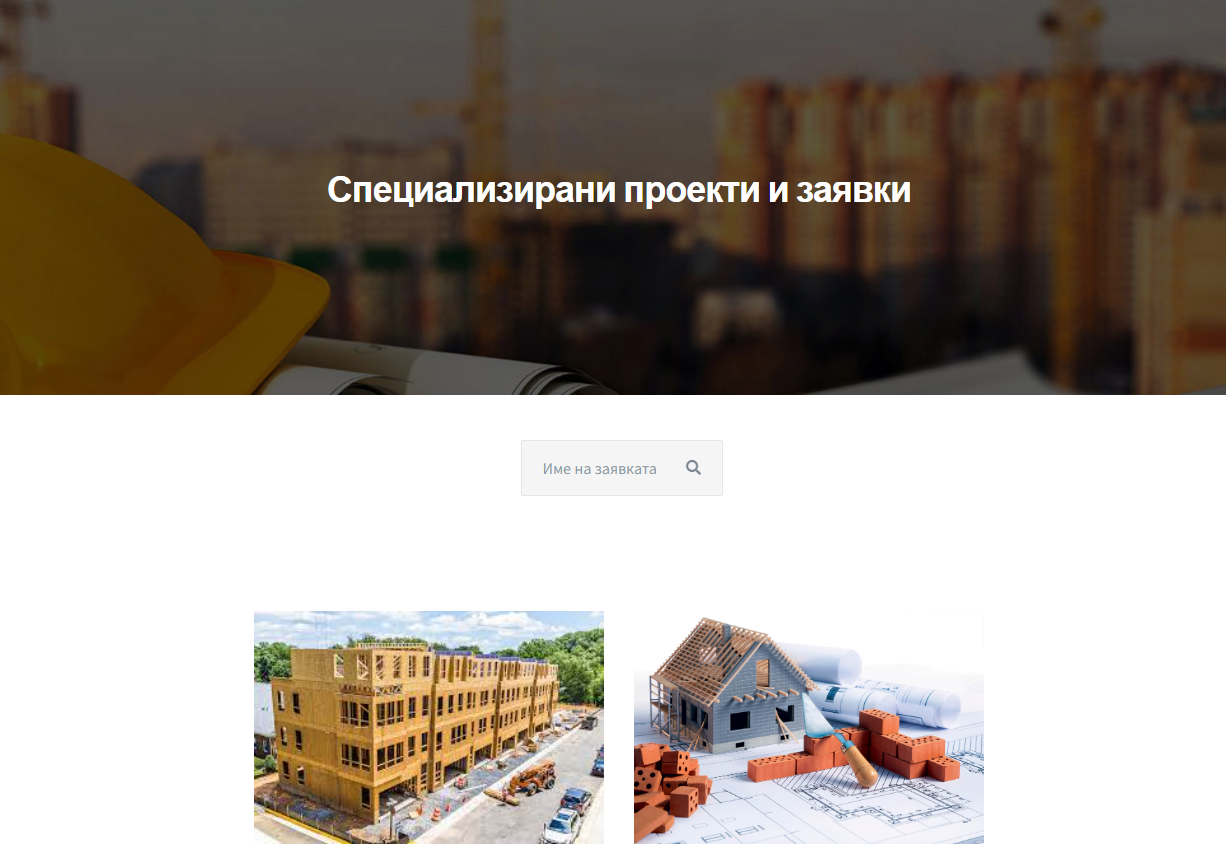
Чрез формата за създаване на заявка клиентите се свързват с работниците и ръководителите отностно своите строителни и ремонтни нужди. Клиентът избира вида специалист, от който има нужда. Видът специалист може да бъде избран от падащо меню, в което са заредени всички данни за специалности от базата данни.



Фиг. 24

**Списък със заявки**

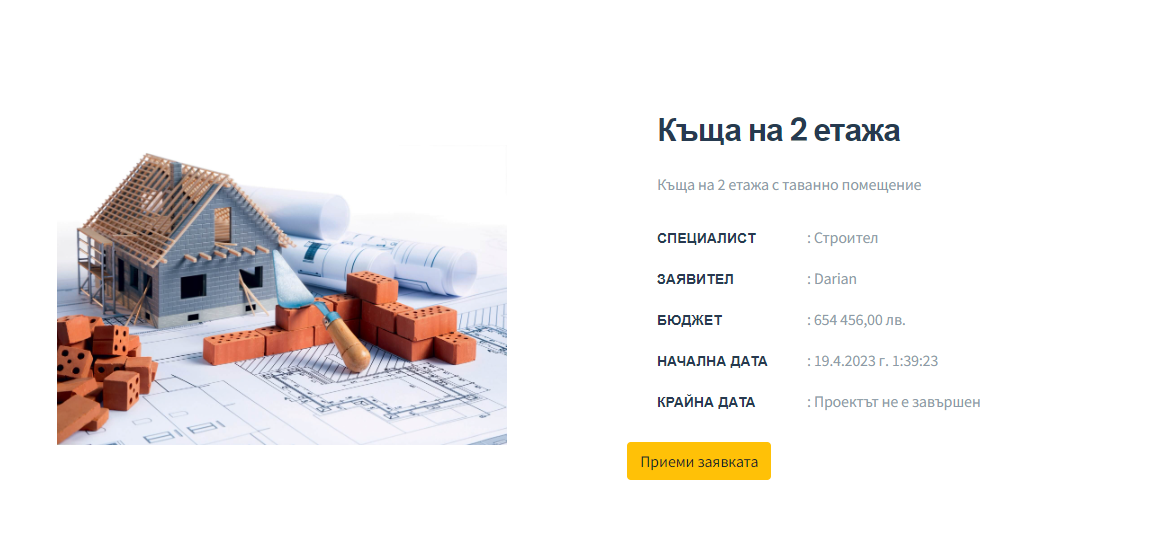
Страницата, на която е реализиран списъка със заявки представлява всички заявки отправени към строителната фирма и нейните служители от клиенти. Чрез интересен дизайн са показани снимки, които клиенти са прикачили към своите заявки. Тези снимки най-често представляват примерен модел на това как трябва да изглежда завършен проект.



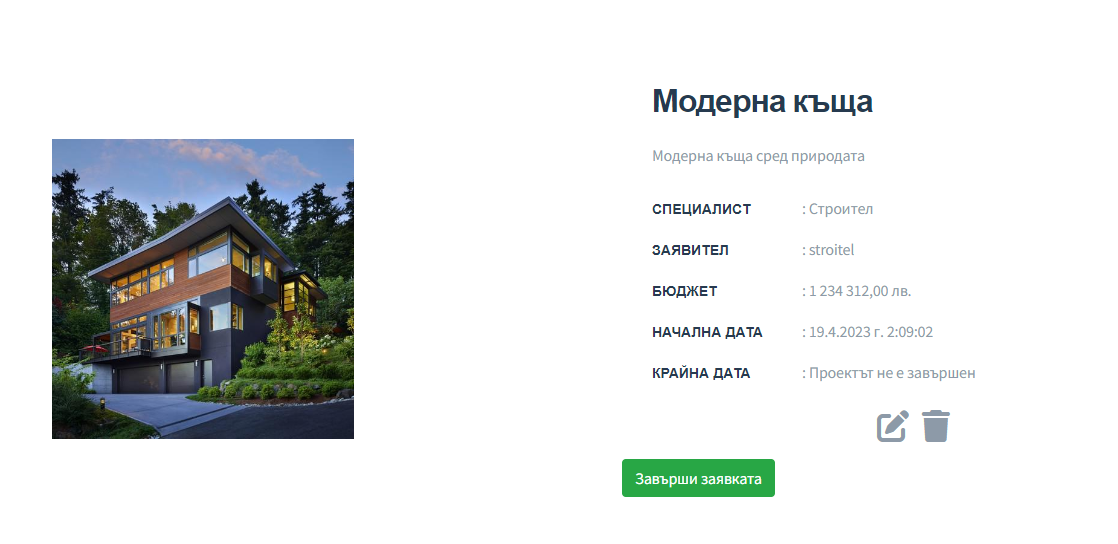
Фиг. 25

**Детайли на заявка**

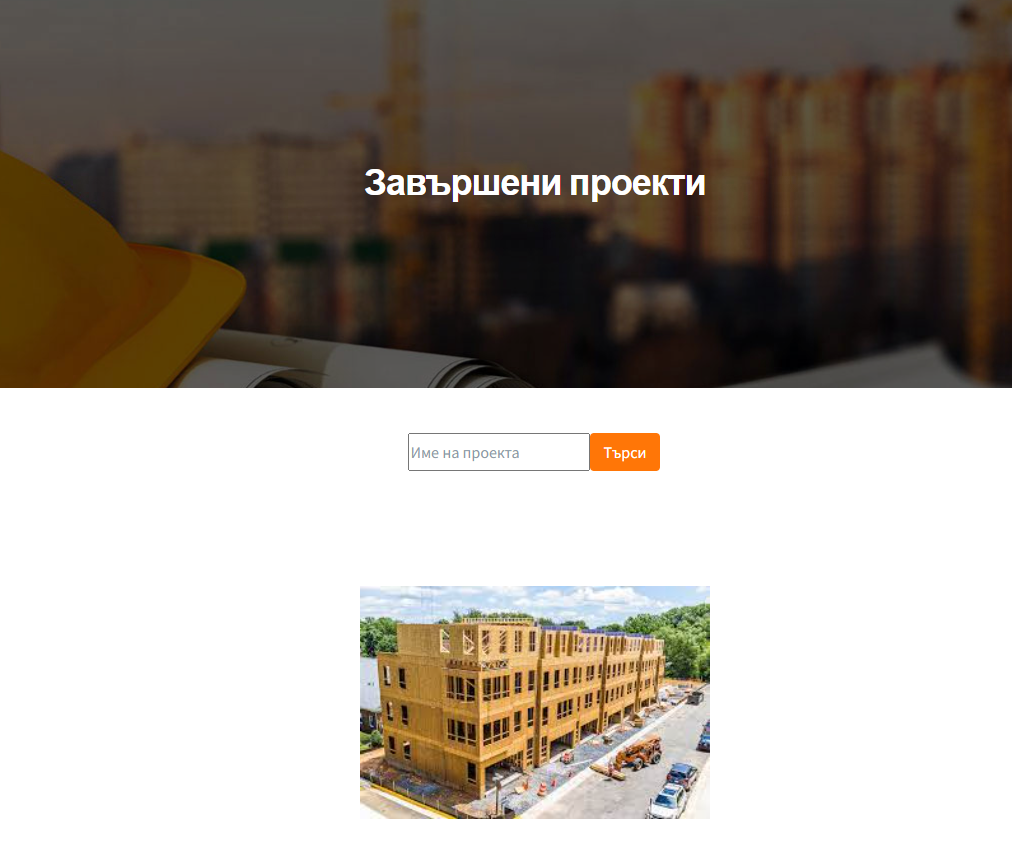
На страницата на детайлите за всяка заявка е представена по-подробна информация. Показано е всяко едно свойство на всяка заявка. Бутонът „Приеми заявката“ във фигура 26 е видим само за работниците, които са от дадената специалност. При натискането на бутона статусът на заявката се променя на „Завърши заявката“. (фиг.27) Когато бутонът „Завърши заявката“ бъде натиснат статусът на заявката се променя отново и бива премахната от страницата със заявките. За сметка на това се появява на страницата със завършените проекти (фиг. 28), където има информация за всеки завършен проект.



Фиг. 26



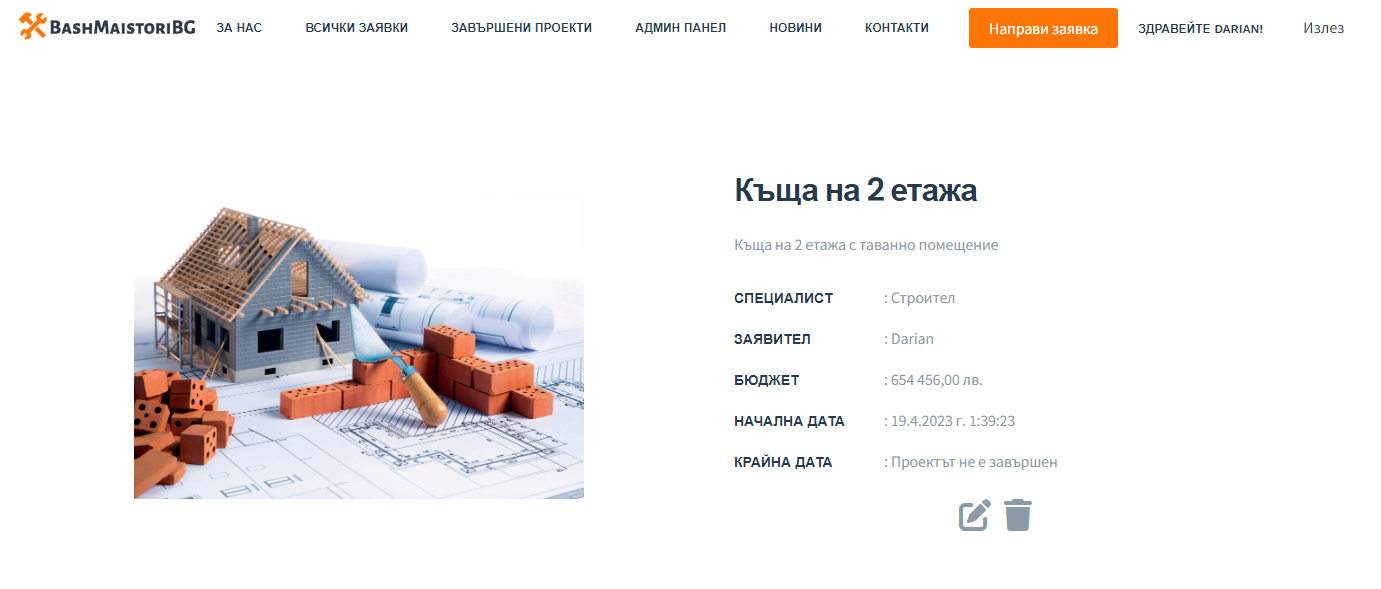
Фиг. 27



Фиг. 28

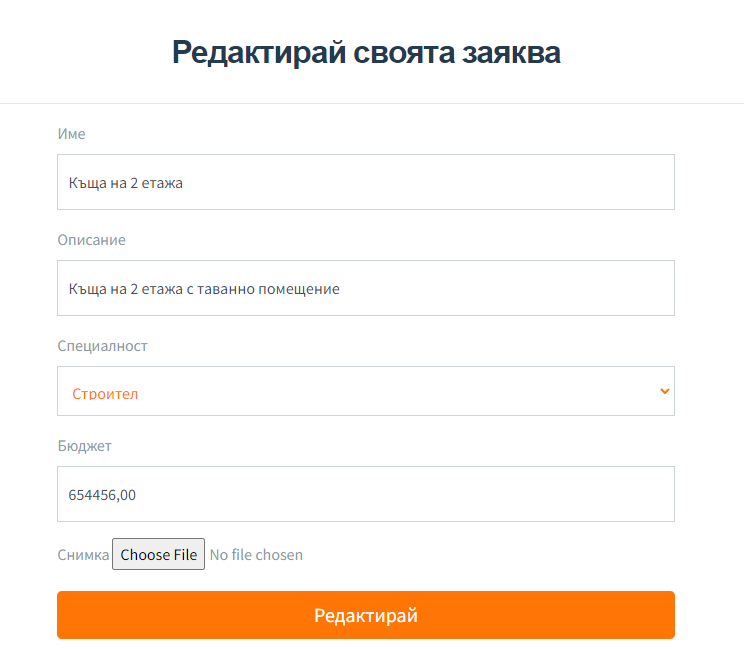
**Редактиране и изтриване на заявки**

Редактирането на заявките става само и единствено от клиента, който е създал дадената заявка. Само когато текущия потребител и създалият заявката съвпадат бутоните за редакция и изтриване са налични.



Фиг. 29

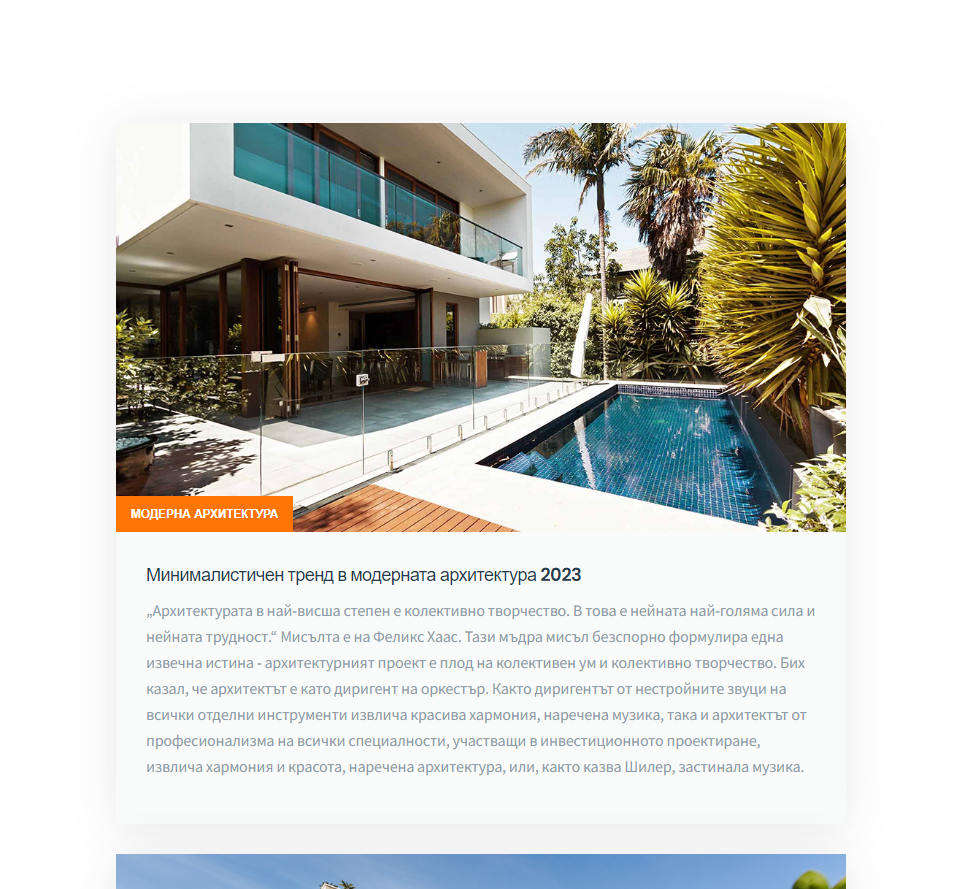
Заявките се редактират чрез форма за редактиране (фиг. 29). Във формата са попълнени предварително данните на заявката, която ще редактираме, за да можем да видим какво искаме да променим и какво да оставим както е.



Фиг. 30

**Страница с новини**

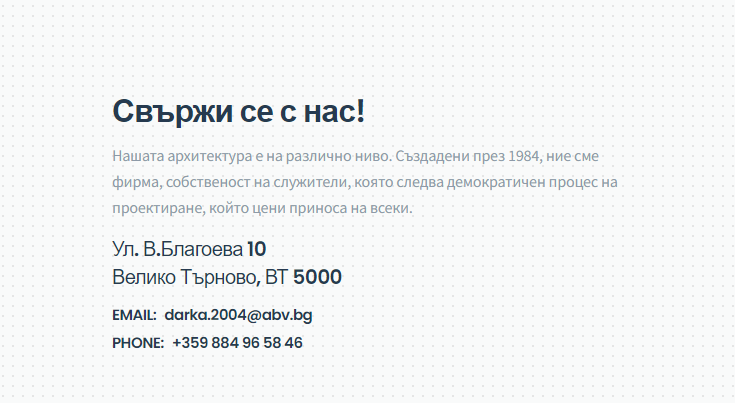
На тази страница може да намерите най-новите новини свързани със строителството, архитектурата и дизайна.



Фиг. 31

**Страница с контакти**

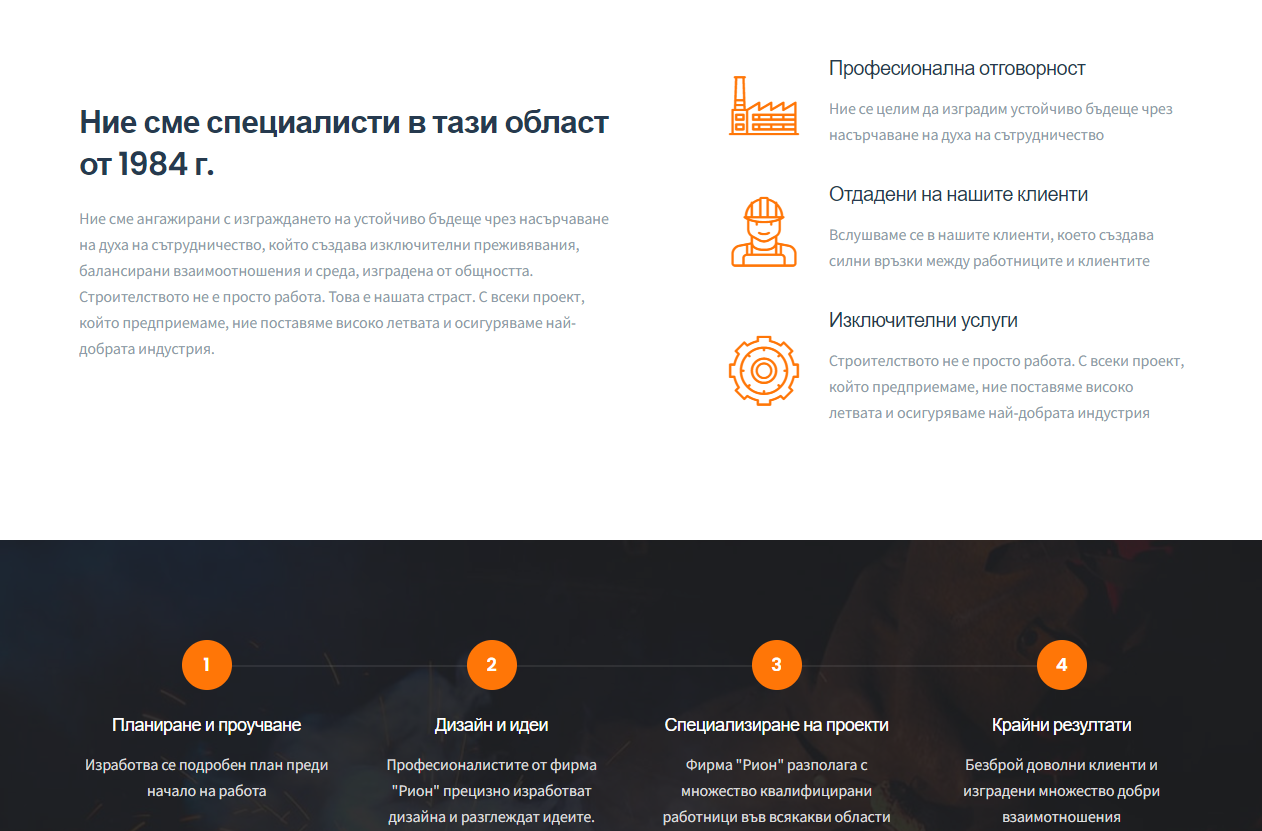
На страницата с контактите може да намерите методите, чрез които да се свържете с фирмата при нужда от помощ или отговор на въпроси.



Фиг. 32

**Страницата „За нас“**

На тази страница може да разбере малко повече за фирмат „Рион“ и нейната история.



Фиг. 33