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: Разработка на уеб платформа за изготвяне и следене на хранителен режим**

*Ученик: Надежда Любенова Антонова*

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

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

***Консултант:*** *Живко Маджев*

Сопот, 2023 г.

**Съдържание**

**Увод (Въведение) 5**   
 **ПЪРВА ГЛАВА.** Проучване……

1.1. Предпоставка за създаване 6

1.2. Анализ на съществуващи решения 6

1.3. Цел на дипломната работа 7

**ВТОРА ГЛАВА.** Проектиране………

2.1. Използвани технологии и подготовка на работната среда

2.1.1. Microsoft Visual Studio

2.1.2. C#(C Sharp).....

2.1.3. Трислойна архитектура на приложението MVC

2.1.4. ASP.NET Core

2.1.5. HTML…

2.1.6. CSS……

2.1.7. JavaScript

2.1.9. База от данни

2.2. Изисквания към софтуерния продукт

2.2.1. Софтуерни

2.2.2. Интерфейсни

2.2.3. Хардуерни

**ТРЕТА ГЛАВА.** Описание на реализирания проект

3.1. Структура на базата данни……

3.2. Структура на MVC приложението – Controllers Class Diagram

3.2.1. Описание и конфигуриране на моделите за работа и обмен

3.2.2. Описание на CRUD операциите - особености на отделни ACTION

3.2.3. Работа с View моделите (където е необходимо)

3.2.4. Описание на Razor Pages и Razor Views (TagHelpers и HtmlHelpers)

3.3. Използване на технологии и алгоритми

3.3.1 Как работи схемата Route Engine

3.3. Как работи шаблона на Contoller с GET и POST заявки към Actions

**Заключение**

Индекс на фигурите

фиг. 1

фиг. 2

фиг. 3

фиг. 4

фиг. 5

фиг. 6

фиг. 7

фиг. 8

фиг. 9

фиг. 10

фиг. 11

фиг. 12

фиг. 13

фиг. 14

фиг. 15

фиг. 16

фиг. 17

фиг. 18

фиг. 19

фиг. 20

**Увод**

Плановете за хранителен режим са предназначени да осигурят енергиино ефективен здравословен комфорт и спестяване на време за човека.

Здравето на човека като състояние на пълно физическо, психично и социално благополучие, е резултат от комплексното въздействие на екологични, социалноикономически, медицински и психо-емоционални фактори. Върху човека въздействат много фактори, като някои от тях укрепват здравето, а други го увреждат (замърсена жизнена среда, шум, нервно-психично напрежение, неправилно хранене, намалена двигателна активност, злоупотреба с алкохол и никотин). Човекът трябва не само да се предпазва от неблагоприятните фактори на природната и социалната среда, но да познава добре и да се възползва оптимално от т.нар. здравословни фактори на околната среда. Здравето и болестта са две форми на съществуване на човека, но между тях няма резки граници

Диета е контролирано приемане на храни и течности с цел постигане на определени резултати, измервани в килограми, мускулна маса, мазнини и др. С други думи това е хранителен режим. Управлението на пропорцията между белтъчините, мазнините и въглехидратите, които се приемат с храната е от основно значение за постигане на максимално добри резултати при управление на даден хранителен режим или при определено коригиране на килограми, поради заболяване, практикуване на спорт или някаква друга необходимост.

Постигането и поддържането на здраве е продължаващ, непрекъснат процес, който се повлиява от еволюцията на знанията за здравословен начин на живот и практиките за грижата за здравето, личните усилия и медицинската помощ.

**ГЛАВА 1.**

**ПРОУЧВАНЕ**

**1.1. Предпоставка за създаване на продукта**

Основната дейност на уеб платформата е свързан с изготвянето и следене на хранителен режим. Всеки режим е индивидуален, направен по строго определени стъпки.

При създаването на сайта за хранителните режими първо направихме проучване на българското и чуждестранното интернет пространство за съществуващи сайтове на подобна тематика - как са организирани тези сайтове, какъв софтуер е използван за тяхната реализация, до колко тези сайтове са изчерпателни, по какъв начин е поднесена информацията. Необходимостта от такова проучване възникна, за да може да се очертае нужния модел за разработката и за да се проучи какво потребителя интуитивно очаква да види в подобен род сайтове. Дефинирахме следните критерии за оценяване на подобни Web –сайтове:

• Въведение - разглеждане на заглавната част на даден сайт, до колко тя е атрактивна и увлекателна, до колко може да задържи вниманието на потребителя.

• Определяне потребителите на дадения сайт – кой би използвал дадения сайт.

• Интерфейс на сайтът - как изглежда сайтът графично, до колко е атрактивен или претрупан, интуитивна ли е ориентацията по страниците.

• Изчерпателност и организация на информацията - каква информация съдържа дадения сайт и до колко тя е покрила засегнатата област и по какъв начин е структурирана.

• Навигация - до колко лесно и интуитивно става предвижването по страниците на сайта.

• Скорост на зареждане - с каква скорост се зарежда сайта, до колко са оптимизирани графичните ресурси.

• Софтуер - какъв софтуер е използван за изработването на сайта.

• Динамичност на сайта - има ли възможност потребителя да общува със сайта.

**1.2. Анализ на съществуващи решения и реализации**

Търсейки в мрежата подобни по тематика Web сайтове установихме, че съществуват много малко, които да предлагат подробна и систематизирана по съответен хранителен план информация за методиката за изготвянето и изпълнението на хранителния режим. Повечето страници са посветени на ограничен брой планове. Съществуват много сайтове, които са посветени на плановете за изчистване на излишни мазнини и стягане на тялото, и сваляне на килограми, чрез калориен дефицит. Тяхно предимство е, че информацията е достъпна от едно място и потребителят може лесно да сравни начина на изработване на плановете за отделните видове. Това, което не ни допадна в страниците, които прегледахме, е, че са претрупани с много реклами отвличащи вниманието на посетителя и не са изцяло ориентирани към проучването на дадения план. Голяма част от информацията за плановете на другите видове открихме предимно в web сайтове за здраве, но тя не е изчерпателна .

Обща черта на повечето от сайтовете, посветени на изготвянето на план е, че те не предлагат никакъв начин за комуникация с посетителите. Много малко от тях са атрактивни и предлагащи достатъчен графичен материал за описанието на плановете. Сравнение с един от сайтовете:

**Диети**

[*https://smartdiet.bg*](https://smartdiet.bg)

• Въведение - Заглавната страница е без ефекти и предлага кратък

увод относно съдържанието на сайта.

• Определяне потребителите на дадения сайт – Сайтът е достъпен

за всички българо- говорещи потребители.

• Интерфейс на сайта - Сайтът е семпъл, няма вграден специфичен

елемент, който да отличи плановете от такива на други сайтове.

Липсва меню с препратки към съответно другите страници,

даващи повече информация по темата.

• Изчерпателност и организация на информация - информацията

според мен е сравнително пълна, има различни мнения на клиенти,

както и информация за специфични основни нива, като метаболитно

и психологическо. Има достатъчно снимков материал, който

да допълва описанието на плановеге.

• Навигация - Навигацията е затруднена. Това, което не ми допадна

е, че няма меню, от което да отидеш на дадена страница, а трябва

от footer-a. Липсва карта на сайта, която да очертава,

каква информация е включена.

• Скорост на зареждане - Снимковият материал е добре подбран

и направен, но страниците, които съдържат видеа, се

зареждат значително по-бавно.

• Софтуер - Използван е само HTML.

• Динамичност на сайта – Сайтът е статичен. Потребителят няма

възможност да взаимодейства със сайта по никакъв начин.

**1.3. Цел на дипломната работа**

Целта на този проект е да се разработи уеб платформа за изготвяне и следене на хранителен режим, където потребителите могат да се свързват с даден треньор от платформата, с когото желаят да работят**.** След регистрация и одобрение на заявката от треньора, потребителите могат да преглеждат изготвения за тях индивидуален режим, както и да следят своя прогрес.

След избора на треньор и одобрението му, ще се направи разговор с клиента, на който ще снема пълен антропометричен, здравен и хранителен статус. Също така хранителният режим трябва да е съобразен с неговата възраст, тегло, физическо и здравословно състояние, както и с начина му на живот. На базата на информацията, която получи, ще създаде специално за клиента персонализиран хранителен режим, който ще отговаря на нуждите на организма му и ще е спрямо неговите вкусови предпочитания. Персонално изработеният хранителен режим включва най-необходими покупки, разнообразни варианти на всяко хранене (най-често между 4 и 6) и междинни хранения (над 5), базирани на предварително попълнен въпросник, в който потребителите уточняват предпочитани храни, както и такива които искат да бъдат избегнати на всяка цена.  
Суплементарен план (хранителни добавки), ако имат желание да използват такива.

Хранителният режим може да бъде допълнително съобразен с предпочитани брой хранения дневно, работотно време и други индивидуални фактори. Потребителите ще имат възможността да отбелязват всяко хранене, изразходена енергия.

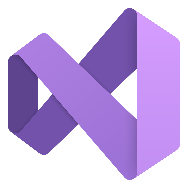
Химичен състав на храните – кратко описание на всяка една храна калории съдържа. Консултации за всяка седмица от режимите, за да се дискутира всичко необходимо и да се направят нужните промени по режимите, ако такива са необходими – на базата на данните, които се предоставят и това как се чувстват.

**ГЛАВА 2.**

**ПРОЕКТИРАНЕ НА СТРУКТУРАТА НА САЙТА**

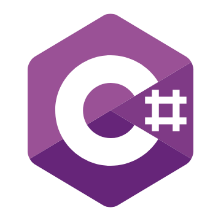
**2.1. Използвани технологии и подготовка на работната среда /**технологии: Visual Studio, MVC, ASP.Net Core; средства: C#, HTML, CSS, JavaScrip, Bootstrap**/**

Настоящата система е разработена на база на едни от последнитетехнологии, предлагани от Microsoft.

 **2.1.1. Microsoft Visual Studio** е мощна [интегрирана среда за разработка](https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%B0%D0%BD%D0%B0_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D0%B7%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0" \o "Интегрирана среда за разработка) на софтуерни приложения за [Windows](https://bg.wikipedia.org/wiki/Windows) и за платформата [.NET Framework](https://bg.wikipedia.org/wiki/.NET_Framework). Използва се за разработка на конзолни и [графични потребителски интерфейс](https://bg.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D0%BD_%D0%BF%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Графичен потребителски интерфейс) приложения, както и Windows Forms или WPF приложения, [уеб сайтове](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1%D1%81%D0%B0%D0%B9%D1%82" \o "Уебсайт), [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" \o "Уеб приложение) 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" \o "Уеб услуга) на всички поддържани платформи от Microsoft Windows, [Windows Mobile](https://bg.wikipedia.org/wiki/Windows_Mobile), Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Visual Studio предоставя мощна интегрирана среда за писане на код, компилиране, изпълнение, дебъгване (както за високо така и за машинно ниво), тестване на приложения, дизайн на потребителски интерфейс (форми, диалози, уеб страници, визуални контроли и други), моделиране на данни, моделиране на класове, изпълнение на тестове, пакетиране на приложения и стотици други функции.

Visual Studio поддържа различни [езици за програмиране](https://bg.wikipedia.org/wiki/%D0%95%D0%B7%D0%B8%D1%86%D0%B8_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5" \o "Езици за програмиране) (например: [C#](https://bg.wikipedia.org/wiki/C_Sharp), VB.NET, [C](https://bg.wikipedia.org/wiki/C_(%D0%B5%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5))/[C++](https://bg.wikipedia.org/wiki/C%2B%2B), [F#](https://bg.wikipedia.org/w/index.php?title=F_Sharp&action=edit&redlink=1), [XML](https://bg.wikipedia.org/wiki/XML)/[XSLT](https://bg.wikipedia.org/wiki/XSLT), [HTML](https://bg.wikipedia.org/wiki/HTML)/[XHTML](https://bg.wikipedia.org/wiki/XHTML), [JavaScript](https://bg.wikipedia.org/wiki/JavaScript), [CSS](https://bg.wikipedia.org/wiki/CSS) и други) и различни технологии за разработка на софтуер (Win32, COM, [ASP.NET](https://bg.wikipedia.org/wiki/ASP.NET), ADO.NET Entity Framework, Windows Forms, WPF, Silverlight и още десетки други Windows и .NET технологии).

**2.1.2. C#** е обектно ориентиран език за програмиране, разработен от Microsoft като част от софтуерната платформа .NET. Стремежът още при създаването 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 (Visual Basic и Java). Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика – инструкциите, които компютърът изпълнява. Програмите на C# представляват един или няколко файла с разширение .cs, в които се съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# до изпълним код и в резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll).

**2.1.3. Трислойна архитектура MVC - „Model-view-controller“.** В [софтуерното инженерство](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%BE%D1%82%D0%BE_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%81%D1%82%D0%B2%D0%BE" \o "Софтуерното инженерство) **многослойната архитектура** (наричана още N-слойна архитектура) е [архитектура](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0" \o "Софтуерна архитектура) от тип [клиент-сървър](https://bg.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D1%8A%D1%80%D0%B2%D1%8A%D1%80" \o "Клиент-сървър), в която [интерфейсът](https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5)" \o "Интерфейс (програмиране)), обработката на приложения и съхранението и обработката на [данни](https://bg.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D0%B8" \o "Данни) са логически разделени на отделни модули. Най-разпространената форма на многослойна архитектура е трислойната архитектура.

Многослойната архитектура осигурява модел, по който разработчиците могат да създават гъвкави приложения, които могат да бъдат използвани многократно. При разделянето на едно приложение на слоеве, разработчиците имат възможността да добавят или променят отделен слой, вместо да преработват цялото приложение.

Всички съвременни бизнес приложения имат достъп до различен вид данни като част от тяхната основна функционалност. След като [релационните бази данни](https://bg.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B8_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "Релационни бази данни) добиха широка популярност, IT индустрията се преориентира от еднослоен модел (mainframe) към модела **[клиент-сървър](https://bg.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D1%8A%D1%80%D0%B2%D1%8A%D1%80" \o "Клиент-сървър)**. При него са налице **клиент**, който включва презентационна логика и голямата част от бизнес логиката на приложението, както и **сървър** включващ в себе си хранилище за данни (data storage) и част от бизнес логиката под формата на готови и съхранени заявки към базата данни.

Трислойната архитектура обикновено се състои от презентационен слой ([потребителски интерфейс](https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Потребителски интерфейс)) на най-високо ниво в приложението. Той служи за прякото взаимодействие с потребителя и изпращането на заявки към бизнес слоя. Не трябва да има директна връзка между този слой и слоят за [бази данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "Бази данни). Бизнес слоят служи за обработка на данните и работните процеси. Той комуникира както с презентационния слой, така и с базите данни. Слоят за данни комуникира само с бизнес логиката и служи за съхранение данни и тяхното използване. Често отделните модули се разработват и на различни платформи. Разработена е от Джон Дж. Донован, в неговата компания OEC (Open Environment Corporation), основана в [Кеймбридж](https://bg.wikipedia.org/wiki/%D0%9A%D0%B5%D0%B9%D0%BC%D0%B1%D1%80%D0%B8%D0%B4%D0%B6_(%D0%9C%D0%B0%D1%81%D0%B0%D1%87%D1%83%D0%B7%D0%B5%D1%82%D1%81)" \o "Кеймбридж (Масачузетс)), Масачузетс.

Освен стандартните предимства на модулния [софтуер](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80" \o "Софтуер), трислойната архитектура е проектирана да позволява да бъде променен или заменен независимо, който и да е от трите слоя, за да отговаря на промени в изискванията или технологиите. Например, промяна на [операционната система](https://bg.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционна система) на презентационния слой би засегнала само кода за [потребителският интерфейс](https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Потребителски интерфейс).

В повечето случай, [потребителският интерфейс](https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "Потребителски интерфейс) работи на десктоп система и използва стандартен графичен интерфейс, функционалната логика за обработка на данните, която може да се състои от един или повече отделни модула, работещи на десктоп система или сървърно приложение, и РСУБД (релационните системи за управление на бази данни) работещи на сървър [база данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "База данни) или [мейнфрейм](https://bg.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B9%D0%BD%D1%84%D1%80%D0%B5%D0%B9%D0%BC" \o "Мейнфрейм), който съдържа логиката за съхранението и обработката на базите данни. Междинният слой може самият да бъде многослоен. В такъв случай цялостната архитектура се нарича N-слойна архитектура.

Трислойната архитектура се състои от следните три слоя:

В областта на уеб разработката трислойната архитектура често се използва в уеб сайтове, най-често т.нар. „електронни магазини“, които се състоят от 3 слоя:

1. Front-end слой (Клиентски слой) – този слой, е съдържанието, което се възпроизвежда от [уеб браузърът](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80" \o "Уеб браузър) и достига до крайния потребител. То може да бъде генерирано статично или динамично.
2. Среден слой – той реализира логиката на самото уеб приложение. Това се осъществява със [сървър](https://bg.wikipedia.org/wiki/%D0%A1%D1%8A%D1%80%D0%B2%D1%8A%D1%80" \o "Сървър), който обработва и генерира динамично съдържание. Този сървър може да е с платформа един от следните езици – [Ruby on Rails](https://bg.wikipedia.org/wiki/Ruby_on_Rails), Java EE, [ASP.NET](https://bg.wikipedia.org/wiki/ASP.NET), [PHP](https://bg.wikipedia.org/wiki/PHP), ColdFusion, [Perl](https://bg.wikipedia.org/wiki/Perl), Node.js.
3. Back-end слой – той представлява [база-данни](https://bg.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "База данни), която се състои, както от самата информация в таблици, така и от система за управлението и (RDBMS).

**2.1.4. ASP.NET Core** е безплатна софтуерна рамка за уеб разработка, с отворен код. Също така тя се явява и следващата стъпка в еволюцията ASP.NET. Тя е разработена съвместно от Microsoft и общността, която е събрала през годините на своето развитие. ASP.NET Core е модуларна софтуерна рамка, която може да върви както на пълната .NET рамка, така и на крос-платформената .NET Core. Въпреки, че е нова софтуерна рамка, изградена върху нов web stack, тя има висока степен на съвместимост с ASP.NET MVC.

Спрямо своите предшественици ASP.NET Core поддържа нова функция – т.нар. „side by side versioning”. При нея различни приложения, които използват една и съща машина, могат да таргетират различни версии на ASP.NET Core, в зависимост от версиите (и нуждите) си. Това не е възможно с по-стари издания на ASP.NET.

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

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

В повечето случаи HTML кодът е написан в текстови редактори, с файлов формат .html, .htm, dhtml и се качва и хоства на [сървъри](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. За да се илюстрира как се включва текст в HTML код: <маркер> Някакъв текст. </край на маркера>. [уеб браузърите](https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D0%B1%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80" \o "Уеб браузър) са програмирани от своя страна така, в повечето случаи, макар че някои браузъри могат да имат съответно проблеми на версията, за да могат да прочетат HTML документите и да ги покажат на екрана като уеб страници. Браузърите не показват самите HTML тагове, освен ако не се отиде в менюто за да се направи това, така че те „интерпретират“ (тоест [парсват](https://bg.wikipedia.org/w/index.php?title=Parsing&action=edit&redlink=1" \o "Parsing (страницата не съществува))) съдържанието на страницата като код и текст за да могат след работа на процесора да покажат желаното уеб-съдържание.

Основното предимство на HTML е, че уеб-страниците, които са го включват в кода си, могат да се разглеждат чрез показването им от браузъра на екрана на повечето устройства. Уебстраницата може да има дизайн, който дори изглежда с добър дизайн с помощта на [CSS](https://bg.wikipedia.org/wiki/CSS) или „правилно оформен“ (например с помощта на [C#](https://bg.wikipedia.org/wiki/C)), както върху монитора на [персоналния компютър](https://bg.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D0%B5%D0%BD_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%8A%D1%80" \o "Персонален компютър), но също и върху миниатюрния дисплей на [пейджър](https://bg.wikipedia.org/w/index.php?title=%D0%9F%D0%B5%D0%B9%D0%B4%D0%B6%D1%8A%D1%80&action=edit&redlink=1" \o "Пейджър (страницата не съществува)) или дисплея на [мобилен телефон](https://bg.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B1%D0%B8%D0%BB%D0%B5%D0%BD_%D1%82%D0%B5%D0%BB%D0%B5%D1%84%D0%BE%D0%BD" \o "Мобилен телефон).

HTML може да прикрепя скриптове писани на езици като [JavaScript](https://bg.wikipedia.org/wiki/JavaScript), който е помощен за HTML, и това променя поведението на дадена уеб страница. [Cascading Style Sheets](https://bg.wikipedia.org/wiki/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](https://bg.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C) поддържа както HTML, така и CSS, и насърчава използването на CSS в HTML страниците още от 1997. Това допринася за разделяне съдържанието и структурата на уеб страниците от тяхното визуално представяне.

**2.1.5. 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. В резултат на това HTML кодът се превръща в сложен и нечетлив, а всяка промяна в проекта на даден сайт изисквала корекцията да бъде нанасяна в целия сайт страница по страница. Използвайки CSS, настройките за форматиране могат да бъдат поставени в един-единствен файл и тогава промяната ще бъде отразена едновременно на всички страници, които използват този CSS файл.  
  
 **2.1.6. JavaScrip** представлява обектно-ориентиран скриптов език, който се използва в множество платформи. Той е малък и лек език, който е безполезен,ако се ползва самостоятелно. Проектиран е за лесно вграждане в други продукти и приложения като уеб браузърите. Ядрото на JavaScript съдържа множество от обекти като масиви, дати, математически функции и стандартни елементи като оператори, структури и декларации(statements).

Най-често използваната форма на JavaScript e като език от страна на клиента(client-side). В този случай скриптът се включва в HTML документ или е рефериран в него, за да бъде интерпретиран от браузъра. Потози начин една уеб страница престава да съдържа само статичен HTML код, но може да включва и програмни кодове, които да осъществяват взаимодействие с потребителя, да указват контрол върху браузъра и динамично да генерират HTML код.

JavaScript кода се изпълнява, когато потребител изпраща резултата от попълнена от него форма. Попълнените данни се изпращат до сървъра само в случай, че всички полетана форматаса попълнени коректно (напр. имейл адрес). Този език позволява прихващането на събития като натискане на бутон, пренасочване към друга страница от линкове и други

**Bootstrap -** Това е CSS рамка. Bootstrap е сред най-популярните HTML, CSS и JS рамки. Това е безплатен набор от инструменти за създаване на уебсайтове, който включва типографски елементи, готови уеб форми, бутони, менюта и други уеб елементи. Основният фокус на рамката признава развитието на сложни мобилни проекти. Важно предимство на Bootstrap е голяма общност, която е пет пъти по-голяма от конкурентитеси.

Основната характеристика на Bootstrap е, че тя не е само CSS рамка, но също така и Javascript библиотека. Bootstrap е разработил готови за използване стилове и скриптове, които са свързани с документа чрез писане на необходимите класове и атрибути на HTML елементи.

Bootstrap е разпознат като много полезен инструмент за мобилно оформление, защото мрежата от елементи, разработени в нея, е доста гъвкава, за да покаже сайта на различни диагонали с високо качество.

**2.1.7. База от данни** (БД, още база от данни) представлява колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. В първоначалния смисъл на понятието, използван в компютърната индустрия, базата от данни се състои от записи, подредени систематично, така че компютърна програма да може да извлича информация по зададени критерии.

Поддръжката на база от данни се осъществява от т.нар. [система за управление на бази от данни](https://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%BE%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8" \o "Система за управление на бази от данни) (СУБД).

Система за управление на бази данни е компютърно приложение ([софтуер](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80" \o "Софтуер)) създадено за комуникация между потребителя, други приложения, както и други БД, с цел да се сравнят и анализират данни. Общото специфично предназначение на СУБД е да позволи определянето, създаването, заявки, актуализацията и администрирането на бази данни. Добре известни СУБД включват MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA, и IBM DB2. Бази данни не са съвместими с различните СУБД, за това различните СУБД работят със стандартни като SQL и ODBC или JDBC, за да позволи на всяко приложение да работи с различни СУБД, а така и с различни БД. Управлението на БД често се избира от модела им, които те подкрепят. Най-използвани системи от бази данни от 1980 г. насам са всички поддържани релационния модели на езика SQL. Често срещано е СУБД да се нарича само „база данни“.

**2.2.** Изисквания към софтуерният продукт

**2.2.1.** Софтуерни

**Windows Server 2019** е деветата версия на операционната система Windows Server от Microsoft, като част от семейството операционни системи Windows NT. Това е втората версия на сървърната операционна система, базирана на платформата Windows 10, след Windows Server 2016. Windows Server е операционната система, която свързва локалните среди с услуги на Azure, което разрешава хибридни сценарии и максимизират съществуващите инвестиции, включително:

* Уникални хибридни възможности с Azure за разширяване на вашия център за данни и максимизиране на инвестициите
* Усъвършенствана многослойна защита за подобряване на положението със защитата
* По-бързи иновации за приложения, което позволява на разработчиците и ИТ специалистите да създават нови приложения и да ги модернизират, както и
* Несравнима хиперконвергентна инфраструктура за развиване на вашата инфраструктура на център за данни

**Internet Information Services(IIS)** е пакет от сървърни услуги, включващ уеб сървър, пощенски сървър и сървър за споделяне на файлове. Създаден от Microsoft, той е конкурентът на Apache и sendmail при Unix системите. Най-важния от горепосочените компоненти е уеб сървърът. Той поддържа всички стандарти за уеб като HTTP 1.0/1.1. За разлика от Apache, той може да работи само под Windows. IIS Express, олекотена (4,5–6,6 MB) версия на IIS, се предлага като самостоятелен безплатен сървър и може да се инсталира на Windows XP със Service Pack 3 и следващите версии на Microsoft Windows. IIS 7.5 Express поддържа само HTTP и HTTPS протоколи. Той е преносим, съхранява конфигурацията си за всеки потребител, не изисква административни привилегии и се опитва да избегне конфликт със съществуващи уеб сървъри на същата машина. IIS Express може да бъде изтеглен отделно или като част от WebMatrix или Visual Studio 2012 и по-нови версии. По подразбиране IIS Express обслужва само локален трафик.

**Microsoft SQL Server** е система за управление на релационни бази данни, разработена от Microsoft. Като сървър на база данни, това е софтуерен продукт с основна функция за съхраняване и извличане на данни, както се изисква от други софтуерни приложения - които могат да работят или на същия компютър, или на друг компютър в мрежа (включително Интернет). Microsoft SQL Server 2019 е предназначено за фирми и организации от всякакъв мащаб – малките, средни и корпоративни предприятия могат да избират измежду различни издания на SQL Server, подходящи за техните нужди.

**SQL Server Management Studio**(**SSMS**) – (Студио за управление на [SQL Server](https://bg.wikipedia.org/wiki/Microsoft_SQL_Server)) е софтуерно приложение, стартирано за първи път с [Microsoft](https://bg.wikipedia.org/wiki/Microsoft) [SQL Server 2005](https://bg.wikipedia.org/w/index.php?title=SQL_Server_2005&action=edit&redlink=1), 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](https://bg.wikipedia.org/w/index.php?title=Object_Explorer&action=edit&redlink=1), който позволява на потребителя да търси, избира и да работи с всеки от обектите на сървъра. Приложението има и „експресна“ версия, която може да бъде изтеглена безплатно.

**Azure Web Apps** е базирана на облачни изчисления платформа за хостване на уебсайтове, създадена и управлявана от Microsoft. Това е платформа като услуга (PaaS), която позволява публикуване на уеб приложения, работещи на множество рамки и написани на различни езици за програмиране (.NET, node.js, PHP, Python и Java), включително собствени на Microsoft и такива на трети страни. Той предлага автоматично мащабиране и висока достъпност, поддържа както Windows, така и Linux и позволява автоматизирани внедрявания от GitHub, Azure DevOps или всяко Git repo.

**Статичният IP адрес** е IP адрес, който не се променя с течение на времето. IP (интернет протокол) адресите са цифрови знаци, които позволяват пакети с данни да бъдат изпращани и получавани от нашите мрежи и устройства. Повечето IP адреси са динамични, което означава, че се променят от време на време. Статичният IP, от друга страна, винаги е една и съща последователност от числа. Статичните IP адреси обикновено се използват от сървъри или друго важно оборудване. IP адресът е уникален номер, присвоен на всяко устройство в TCP/IP мрежа. IP адресите идентифицират компютри и устройства и им позволяват да комуникират помежду си.

**Система за имена на (DNS)** представлява [разпределена база от данни](https://bg.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%BE%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8) за компютри, услуги или други ресурси свързани към [интернет](https://bg.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) или частни мрежи, с чиято помощ се осъществява преобразуването на [имената](https://bg.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%B9%D0%BD) на хостовете в [IP адреси](https://bg.wikipedia.org/wiki/IP_%D0%B0%D0%B4%D1%80%D0%B5%D1%81). Това улеснява работата на потребителите на интернет услуги. Вместо да въвежда IP адрес (комбинация от цифри), за да достигне до даден ресурс в мрежата, потребителят може просто да въведе неговото име (домейн).

Информацията за IP адресите и имената на домейни се съхранява на DNS сървърите. DNS е разпределена дървовидна система от обвързани чрез логическа йерархия [сървъри](https://bg.wikipedia.org/wiki/%D0%A1%D1%8A%D1%80%D0%B2%D1%8A%D1%80). В основата на тази структура са сървърите, съхраняващи:

* [*домейни от първо ниво*](https://bg.wikipedia.org/wiki/%D0%94%D0%BE%D0%BC%D0%B5%D0%B9%D0%BD%D0%B8_%D0%BE%D1%82_%D0%BF%D1%8A%D1%80%D0%B2%D0%BE_%D0%BD%D0%B8%D0%B2%D0%BE) (*top-level domains*) – например [.com](https://bg.wikipedia.org/wiki/.com), [.org](https://bg.wikipedia.org/wiki/.org), [.edu](https://bg.wikipedia.org/wiki/.edu) и т.н. и
* множество *домейни на държавно ниво* (*country-level domains*) – [.bg](https://bg.wikipedia.org/wiki/.bg) (за България), [.fi](https://bg.wikipedia.org/wiki/.fi) (за Финландия), [.fr](https://bg.wikipedia.org/wiki/.fr) (за Франция) и т.н.

Следващото ниво образуват *регистрираните домейни* (*registered domains*) – about.com, abv.bg, pirin.com и т.н. *Местните домейни* (*local domains*), наричани още *поддомейни* (*subdomains*), като compnetworking.about.com, sdyn.pirin.com, се определят и администрират от собствениците на съответните главни домейни. За разделяне на различните равнища се ползва точка

**2.2.2.** Потребителски интерфейс

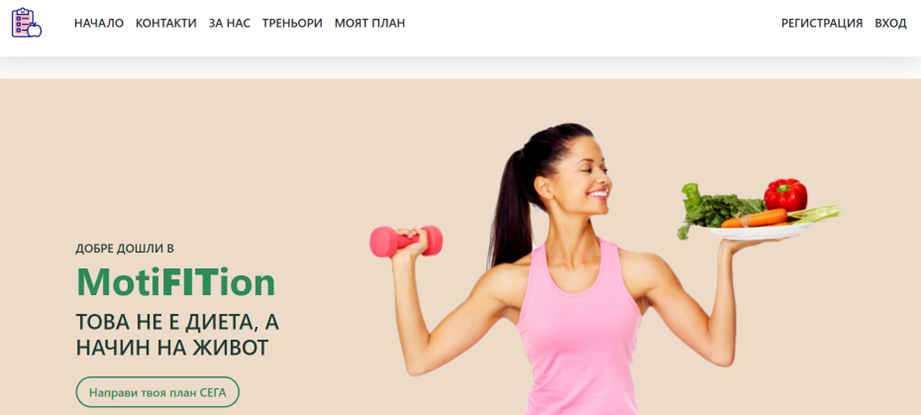
Графичния интерфейс ще се състои от няколко страници. Страниците могат да бъдат разделени на такива, които са видими за обикновения потребител и такива – видими за администратора. В тази точка ще опиша всяка една от тях. Заглавната част на всички страници ще е с емблемата на сайта за здравословното хранене. Дизайнът на потребителския интерфейс, както и използваните технологии са съобразени с функционалността на приложението. Уеб платформата се състои от публична част, в която са включени:

* ***Начална страница***

Първата страница, която виждат потребителите, е страница с меню на темите, както и тяхното описание. Под менюто може да се види анимация и името на платформата, под която е поставен мотивиращ цитат, както и бутон, отвеждаш към ,,Регистрация”. След това е представена обща информация за идеята на сайта. Обяснена е кратка информация относно това, какво представлява хранителният режим и бутон, водещ към страница ,,За нас”. Също така са представени някои от треньорите и техните професии, също с бутони, водещи към страница ,,Треньори”. Накрая на страницата е показан пример с резултатите на клиент, който е пробвал план, изработен спрямо неговите изисквания.

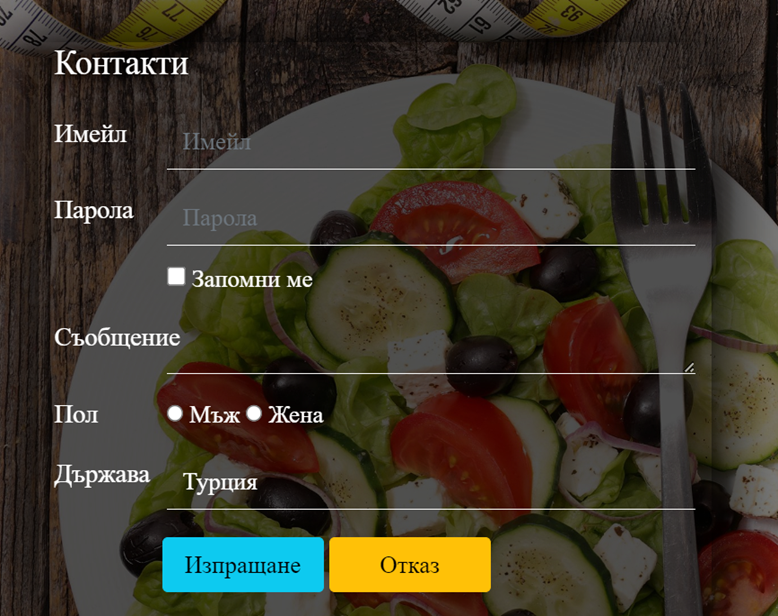
Също така ако посетителя е вече регистриран в страниците на форума има възможност да влезе неговите страници със своето потребителско име и парола. Ако не е регистриран може да направи това използвайки хипервръзката Регистрация.

Връзките Контакти, За нас, Треньори, Регистрация и Вход се намират в менюто на страницата, което в зависимост коя страница е показана съдържа различна информация.



*Фиг. Начална страница на сайта*

* ***Страница ,,Контакти”-***Когато дадено лице иска да зададе своя въпрос му е дадена форма, която е с полета за имейл, парола, съобщение(въпроса, който иска да зададе), пол и държава.



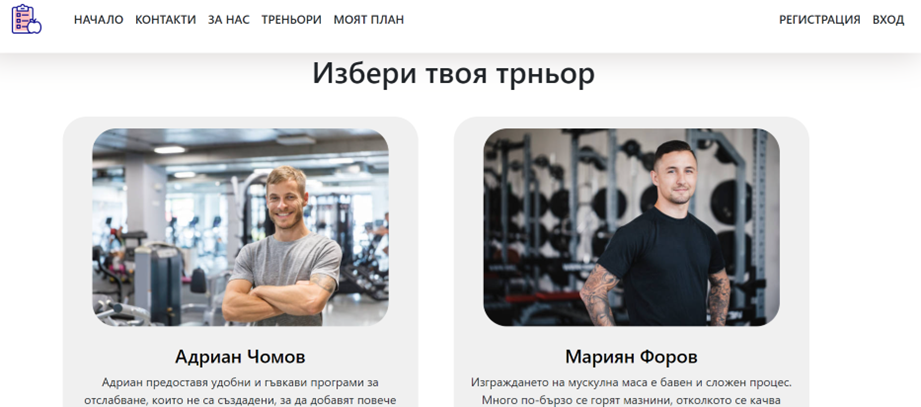
*Фиг. Контакти на сайта*

* ***Страница ,,За нас”*** -в тази страница е представена подробна информация за екипа. Мисията на екипа, с какво е по-различен от другите и как би могъл да помогне на клиентите. Също и принципите, които са важни при изпълнението на хранителния план.



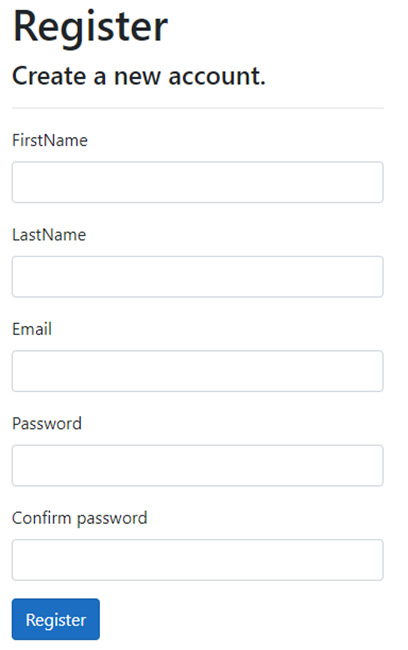
*Фиг. Контакти на сайта*

* ***Страница ,,Треньори”-*** в тази страница е представена подробна информация за треньорите и техните програми. Какво представляват програмите, тяхната трудност и какво е необходимо за тях.



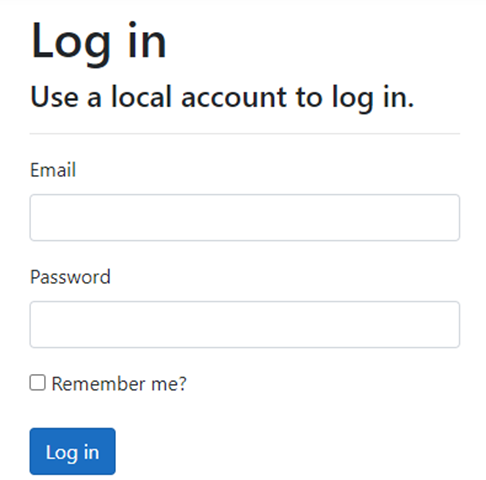
*Фиг. Контакти на сайта*

* **Страница ,,Регистрация”-** Посредством тази страница даден посетител може да се регистрира като потребител на платформата. За целта трябва да предостави определена информация за себе си като попълни формата показана на фиг.. Формата има задължителни полета за попълване (потребителско име, парола, e – mail адрес). При неправилно въведена информация, излиза съобщение което указва къде е грешката. Възможни грешки са: непозволени символи за потребителско име и парола, потребителското име вече заето, неправилен e – mail адрес, първоначално и повторно въведените пароли не съвпадат. При регистрация дадения потребител Web базирано представяне на традициите и обичаите на България 41 получава осведомително писмо, което дава информация за потребителското име и парола, на указания от него e – mail адрес при регистрацията.



*Фиг. Контакти на сайта*

* ***Страница ,,Вход”-***Всеки път когато клиента иска да види своя план, той ще трябва да влезе в своя акаунт. За да влезе е необходимо да се попълни форма с полетата имейл и парола. Също така ако е забравил паролата си, клиента може да влезе в линка ,,Забравена парола”. А ако не е регистриран, има линк, който го отвежда в страница ,,Регистрация”.



*Фиг. Контакти на сайта*

**Страници достъпни за администратора.**

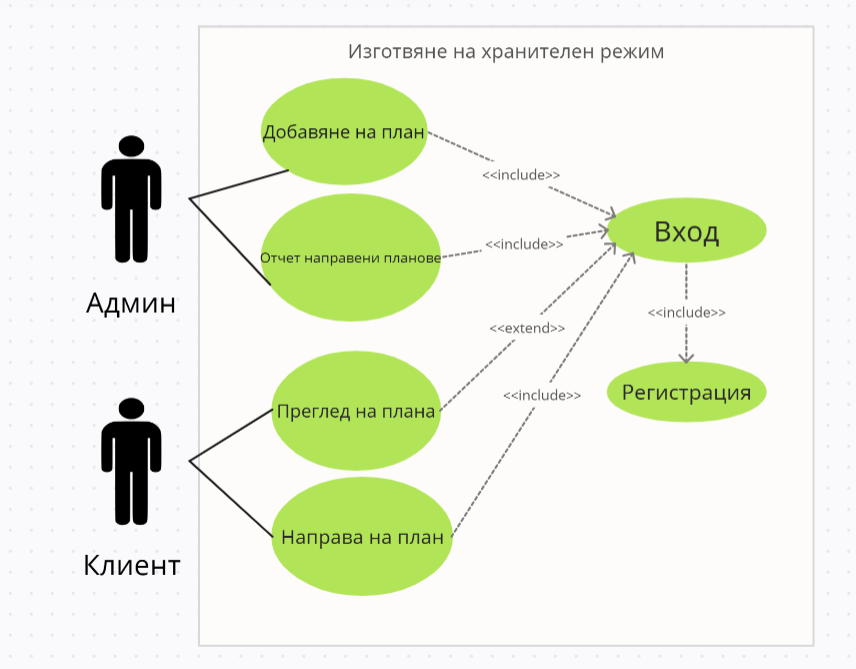
Всички страници достъпни за обикновения потребител са такива и за администратора. Когато даден посетител влезе в страниците на форума като администратор в долния край на всяка страница е достъпна препратката Администраторски панел (фигура 21). Страницата Администраторски панел дава достъп до разширените администраторски функции.

Администраторът има следните възможности: да търси даден потребител по неговия e – mail и потребителско име, да изтрие даден потребител, да забрани или разреши добавянето на теми от даден IP адрес, да спре e-mail осведомяването до даден адрес.

**Изготвяне на Use Cases Diagrams**

Диаграмите на случаите на употреба (Use-case Diagrams) са основен вид

диаграми за специфициране и документиране на различните потребителски изисквания към една система. По-точно, те се използват за специфициране на актьорите за системата (потребители и външни системи) и за определяне на функционалните изисквания към разработваната софтуерна система от гледна точка на различните типове потребители и външни клиенти.



*фиг.*

**2.2.3. Системни изисквания**

**Клиент**

**• *Операционна система* –** Windows 98; Windows 98 SE, Windows

2000, Windows ME, Windows Server 2003, Windows XP, Windows

Vista

**• *Софтуер*** - NET Framework 2.0 Redistributable, Microsoft Internet

Explorer 6.0 със Service Pack 1 или по-нова версия, Microsoft Web

Service Enhancements 3.0

**• *Оперативна памет* –** минимум 128 MB, препоръчително 256 MB

или повече

**• *Процесор* -** тактова честота 400 MHz или повече

**• *Дисково пространство* -** минимум 800MB (включително 200MB за

.NET Framework Redistributable)

**Приложен сървър**

**• *Операционна система* –** Windows 2000 Server; Windows Server

2003

**• *Софтуер* -** NET Framework 2.0 Redistributable, Microsoft Web

Service Enhancements 3.0

**• *Оперативна памет* –** 1 GB или повече

**• *Процесор* -** тактова честота 1 GHz или повече

**• *Дисково пространство* -** минимум 2GB (включително 200MB за

.NET Framework Redistributable)

**Сървър за база от данни**

**• *Операционна система* –** Windows 2000 Server; Windows Server

2003

**• *Софтуер* –** Microsoft SQL Server 2005

**• *Оперативна* *памет* –** 1 GB или повече

**• *Процесор* -** 1 GHz или повече

**• *Дисково пространство* –** 100GB или повече

**ГЛАВА 3.**

**ПРОГРАМНА РЕАЛИЗАЦИЯ НА WEB БАЗИРАН ИЗГОТВЯНЕ НА ХРАНИТЕЛЕН РЕЖИМ**

**3.1. Структурата на базата от данни**

Базата данни е храниище за данни, мястото където/от което може да се съхраняват/извличат данни, когато е необходимо. Съществуват много начини за съхранение и извличане на данни. Един от начините, използван в настоящата дипломна работа е използването на SQL заявки.

3.1.1. Първични и външни ключове – за да има достъп до данните, съхранени в таблици, трябва да може да се различава уникално всеки от редовете на таблицт. Затова идентификаторът трябва уникално да определя всички данни в таблицата. В примерен случай за таблица Employee се създава допълнителна колон, която има уникална стойнст за всеки от редовете в таблицата-(EMP\_id). Този уникален идентификатор се нарича първичен ключ. Аналогично се определят първичните ключове за останалите таблици.

Вънен ключ-се нарич колона или няколко колони, които задават връзка между двете таблици. Добавянето на колоната с първите ключове на едната таблица към друга таблица задава връзката между двете таблици(напр.: в таблца EMP първичният ключ е EMP\_id, докато в таблица EMPD този атрибут е с наменование: EMPD\_EMP\_id).

3.1.2. Нормализация – тя е процес на намаляване на излишъка от данни. Представлява разделяне на даннте в две или повече таблици, докато повтарящите се групи от данни се позиционират в две отделни таблици.

В случая постигнатите резултати са:

1. Всички таблици имат уникални идентификатори
2. Таблиците не съдържат повтарящи се стойности или колони
3. Всички таблици съдържат данни от еднотипнти обекти
4. За всички таблици се избегват колони, позволяващи празни стойности

3.1.3. Дизайн на базата данни

След прилагането на всички концепции описани дотук, се получава структурата на базата данни във вида, показан на фиг.

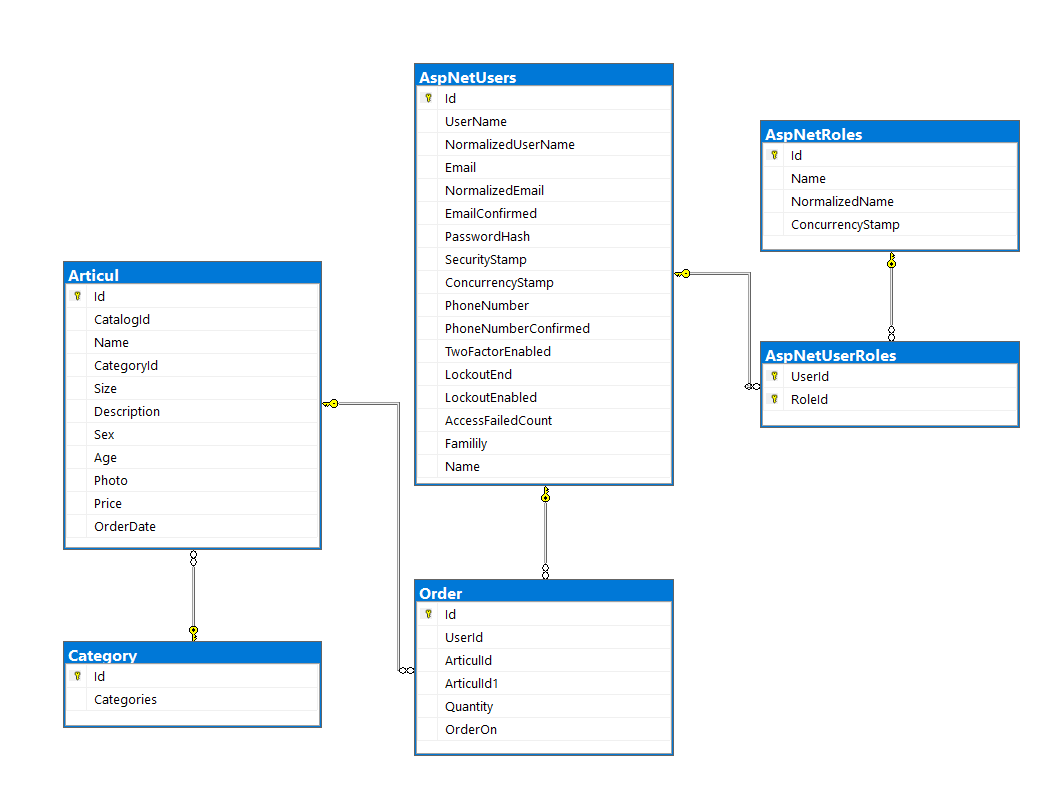


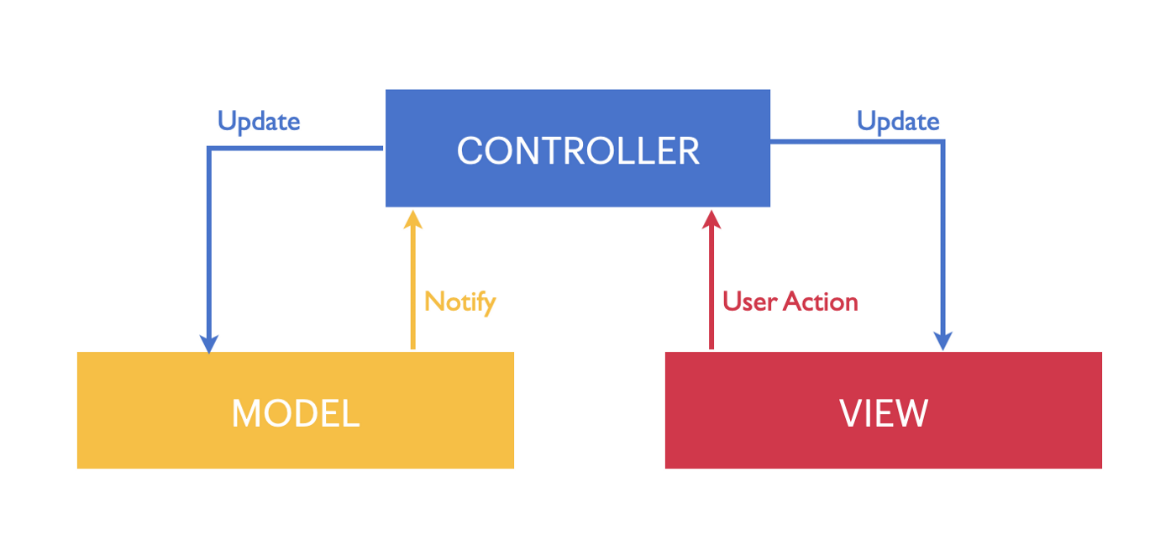
Таблица AspNetUsers - съдържа информация за потребителите.

Таблицата Articul – съдържа информация за артикулите.

Таблицата Order – съдържа за извършените поръчки.

Таблицата Category – определя категорията продукти.

**3.2. Структура на MVC приложението – Controllers Class Diagram**



**Модел-Изглед-Контролер** (Model-View-Controller или MVC) е архитектурен [шаблон за дизайн](https://www.wikiwand.com/bg/%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.

* **Модел** – ядрото на приложението, предопределено от областта, за която се разработва; обикновено 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. Например в банково приложение това са класовете, описващи клиентите, техните сметки, транзакциите, които са осъществили и т.н., както и класовете за извършване на операции над тези обекти (engines) – например клас Transfer с методи като createInterBankTransfer(), createInnerBankTransfer(), getCash() и т.н.
* **Изглед** (англ. View) – е изходящият поток от информация (*това, което приложението изпраща като отговор до дисплея, респективно – до потребителя, в следствие на неговата заявка*). Възможни са няколко различни изгледа на една и съща информация, като например различни диаграми за мениджмънт на даден ресурс или различни таблици. 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" \o "JFrame (страницата не съществува)) 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" \o "Графичен интерфейс) се прави и каква платформа се използва
* **Контролер** – тази част от сорс кода (клас или библиотека), която взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните, и чак след това ги дава на изгледа. Например може да бъде създаден един малък [обект](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5)" \o "Обект (програмиране)), в който да бъдат сложени данните за транзакцията – като в контролера бъдат взети данните за транзакцията от модела, бъдат преведени датите от [UNIX](https://bg.wikipedia.org/wiki/UNIX) формат в четим от потребителя формат, бъде преобразувана валутата от долари в евро например, бъде закръглено до втория знак вместо да се виждат данните както са в модела (и в базата) до 10-ия. 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.

**3.2.1. Описание на CRUD операциите- особености на оделните ACTION**

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 "Insert", "R" със "Select", "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. Такива вариации са:

-**Create операция** - прави това, което подсказва името. Това означава създаване на запис в базата данни. Този запис може да бъде акаунт, потребителска информация, публикация или задача.

public IActionResult Create()

{

return View();

}

// POST: Dishesses/Create

// To protect from overposting attacks, enable the specific properties you want to bind to.

// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

-**Edit операция** - операцията READ означава получаване на достъп до входовете или записите в потребителския интерфейс. Тоест, виждайки го. Отново записът може да бъде всичко - от потребителска информация до публикации в социални медии и други.

Този достъп може да означава, че потребителят получава достъп до създадените записи веднага след създаването им или след търсенето им. Търсенето е реализирано, за да позволи на потребителя да филтрира записите, от които не се нуждае.

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

{

if (id == null)

{

return NotFound();

}

var dishess = await \_context.Dishess.FindAsync(id);

if (dishess == null)

{

return NotFound();

}

return View(dishess);

}

-**Delete операция** - Изтриването означава да се отървете от запис от потребителския интерфейс и базата данни.

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

{

if (id == null)

{

return NotFound();

}

var dishess = await \_context.Dishess

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

if (dishess == null)

{

return NotFound();

}

return View(dishess); }

-**Update операция** - UPDATE е операцията, която ви позволява да променяте съществуващи данни. Тоест редактиране на данните.

За разлика от READ, операцията UPDATE променя съществуващите данни, като прави промени в тях.

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

{

if (id == null)

{

return NotFound();

}

var dishess = await \_context.Dishess

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

if (dishess == null)

{

return NotFound();

}

return View(dishess);

}

**ACTION -** Тъй като контролерът е просто обикновен .NET клас, той може да има полета, свойства и методи. Особено интересни са методите на клас Controller, тъй като те са връзката между браузъра (и следователно потребителя) и вашето приложение. Поради тази причина методите на клас Controller се наричат действия - метод обикновено съответства на действие във вашето приложение, което след това връща нещо на браузъра/потребителя.

Тъй като браузърът работи чрез извикване на URL, имате нужда от нещо, което превежда URL адресите към съответен контролер и действие (метод). Например браузърът може да поиска URL като /cars/1/ и след това искате вашият CarsController да обработва тази заявка с метод/действие, наречено Details . Това се прави с концепцията за маршрутизирането. /маршрутизирането е това, което свързва URL адресите с действията на вашите контролери/.

Когато създавате своите контролери, имайте предвид, че всички публични методи в клас Controller се считат за действие . Това означава, че ако сте дефинирали всеобхватни правила за маршрутизиране за вашия контролер (а това е често срещано нещо), всички методи във вашия клас Controller на теория могат да бъдат достигнати с помощта на URL. Така че, ако имате методи на вашия контролер, които не искате крайният потребител да може да извиква, не забравяйте да го маркирате като private . Като алтернатива, ако наистина имате нужда от метод да бъде публичен, но да не е достъпен чрез URL, можете да маркирате метода с атрибута [NonAction] .

**3.2.2. Работа с View моделите (където е необходимо). Разлика с използванет на ViewBag и ViewData ( описание на двата обекта)**

ViewBag в ASP.NET MVC се използва за прехвърляне на временни данни (които не са включени в модела) от контролера към изгледа.

Вътрешно това е свойство на динамичен тип на класа ControllerBase, който е базовият клас на класа Controller. Следващият пример демонстрира как да прехвърляте данни от контролера към изглед с помощта на ViewBag.

namespace MVC\_BasicTutorials.Controllers

{

public class StudentController : Controller

{

IList<Student> studentList = new List<Student>() {

new Student(){ StudentID=1, StudentName="Steve", Age = 21 },

new Student(){ StudentID=2, StudentName="Bill", Age = 25 },

new Student(){ StudentID=3, StudentName="Ram", Age = 20 },

new Student(){ StudentID=4, StudentName="Ron", Age = 31 },

new Student(){ StudentID=5, StudentName="Rob", Age = 19 }

};

// GET: Student

public ActionResult Index()

{

ViewBag.TotalStudents = studentList.Count();

return View();

}

}

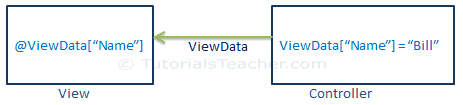
}

Вътрешно ViewBag е обвивка около ViewData. Той ще хвърли изключение по време на изпълнение, ако името на свойството ViewBag съвпада с ключа на ViewData.

В ASP.NET MVC ViewData е подобен на ViewBag, който прехвърля данни от Controller към View. ViewData е от речников тип, докато ViewBag е от динамичен тип. И двете обаче съхраняват данни в един и същи речник вътрешно.

ViewData е речник, така че съдържа двойки ключ-стойност, където всеки ключ трябва да бъде низ.

Следната фигура илюстрира ViewData.



ViewData прехвърля данни само от контролера към изгледа, а не обратното. Важи само по време на текущата заявка.

Следващият пример демонстрира как да прехвърляте данни от контролера към изглед с помощта на ViewData.

public ActionResult Index()

{

IList<Student> studentList = new List<Student>();

studentList.Add(new Student(){ StudentName = "Bill" });

studentList.Add(new Student(){ StudentName = "Steve" });

studentList.Add(new Student(){ StudentName = "Ram" });

ViewData["students"] = studentList;

return View();

}

ViewData и ViewBag използват един и същ речник вътрешно. Така че не можете да имате съвпадения на ViewData Key с името на свойството на ViewBag, в противен случай ще хвърли изключение по време на изпълнение.

**Следващата фигура илюстрира ViewBag.**

ViewData и ViewBag се използват за една и съща цел - за прехвърляне на данни от контролер към изглед. ViewData не е нищо друго освен речник на обекти и е достъпен чрез низ като ключ. ViewData е свойство на контролера, което излага екземпляр на класа ViewDataDictionary. ViewBag е много подобен на ViewData. ViewBag е динамично свойство (динамична ключова дума, която е въведена в .net framework 4.0). ViewBag е в състояние да задава и получава стойност динамично и може да добавя произволен брой допълнителни полета, без да го преобразува в строго въведени. ViewBag е само обвивка около ViewData.

TempData е речник, който е извлечен от клас TempDataDictionary. TempData се съхраняват данни точно като сесия на живо за кратко време. TempData съхранява данни за времето на HTTP Request, което означава, че съхранява данни между две последователни заявки. TempData ни помага да прехвърляме данни между контролери или между действия. TempData вътрешно използват променливи на сесията. Имайте предвид, че TempData работи само по време на текущата и последваща заявка. Обикновено се използва за съхраняване на еднократни съобщения. С помощта на метода TempData.Keep() можем да запазим стойност в обекта TempData след приключване на заявката.

Пример за TempData

//Controller Code

public ActionResult Index()

{

List<string> Student = new List<string>();

Student.Add("Jignesh");

Student.Add("Tejas");

Student.Add("Rakesh");

TempData["Student"] = Student;

return View();

}

//page code

<ul>

<% foreach (var student in TempData["Student"] as List<string>)

{ %>

<li><%: student%></li>

<% } %>

</ul>

° С#

Имаме три опции: ViewData, ViewBag и TeampData за предаване на данни от контролера към преглед и в следващата заявка. ViewData и ViewBag са почти сходни и ни помага да прехвърляме данните от контролера към изглед, докато TempData също работи по време на текущите и следващите заявки.

**3.2.3. Описание на Razor Pages и Razor Views (TagHelpers и HtmlHelpers)**

Razor Pages предоставят алтернативна технология на системата Model-View-Controller. Razor Pages ви позволяват да създавате страници с Razor код, който може да обработва заявки. До известна степен тази функционалност е подобна на начина, по който работят уеб формуляри, които представляват aspx страница и има C# логически файл, свързан с тази страница. В тази връзка Razor Pages предоставят алтернатива на стандартния MVC модел за изграждане на приложение.

Razor Pages е най-подходящ за изграждане на малки приложения.

**Razor Views**

Razor View е синтаксис за маркиране, който ни помага да пишем HTML и код от страна на сървъра в уеб страници, използвайки C#. Това е език за маркиране от страна на сървъра, но изобщо не е език за програмиране.

**ЗАКЛЮЧЕНИЕ**

Разработен е уеб сайт за изработване и следене на хранителен режим. За да се постигне готовността на проекта се използват следните технологии HTML,ASP.NET,C#,CSS.Интерфейса е лесен за употреба,както по потребители запознати с работата на новите технологии.Нужно е доизграждане,развиване и изчистване на дизайна.Текущата разработка предоставя основни функции за работа на оналйн магазин,има лесен за изполазване интерфейс,достъпен е за потребителите.Осигурена сигурност за нашите клиенти.Предстои популириация и разпространение,както на сайта така и на предлаганите продукти.

**ИЗПОЛЗВАНА ЛИТЕРАТУРА**

**ASP.NET:източник:** [**https://bg.wikipedia.org/wiki/ASP.NET**](https://bg.wikipedia.org/wiki/ASP.NET)

**C#:източник:** <https://bg.wikipedia.org/wiki/C_Sharp>

**CRUD:източник:** <https://softuni.bg/blog/what-is-crud-mssql>

**HTML:източник:** <https://bg.wikipedia.org/wiki/HTML>

**CSS:източник:** <https://bg.wikipedia.org/wiki/CSS>

**JavaScrip:източник:** [**https://bg.wikipedia.org/wiki/JavaScript**](https://bg.wikipedia.org/wiki/JavaScript)

**Bootstrap:източник:** [**https://bg.wikipedia.org/wiki/Bootstrap**](https://bg.wikipedia.org/wiki/Bootstrap)