**Русенски университет „Ангел Кънчев“**

Катедра „Компютърни системи и технологии“

**ДИПЛОМЕН ПРОЕКТ**

**ТЕМА**: „ Разработване на Уеб приложение за проверка на валидността на програмен код“

**Изготвил:** Андриян Невелинов Кръстев

**Фак. ном.:** 113250

**Група:** 26б

**Специалност:** КСТ

**Курс:** IV

Дата на предаване: Проверил: \_\_\_\_\_\_\_\_

Русе

2015 г.

**Съдържание**

[1. Анотация. 6](#_Toc422431808)

[2. Увод. 7](#_Toc422431809)

[2.1. История на уеб. 7](#_Toc422431810)

[2.2. Уеб страница 8](#_Toc422431811)

[2.2. Уев браузър 9](#_Toc422431812)

[2.3. Уеб сървър 9](#_Toc422431813)

[2.4. Сигурност 10](#_Toc422431814)

[3. Обзор на съществуващите решения. Изводи. Цел и задачи. 11](#_Toc422431815)

[3.1. BGCoder. 11](#_Toc422431816)

[3.2. Codecademy. 14](#_Toc422431817)

[3.3. Code School 15](#_Toc422431818)

[3.4. Цел и задачи. 17](#_Toc422431819)

[4. Проектиране и описание на предлаганото решение. 17](#_Toc422431820)

[4.1. Изисквания към програмната система. 17](#_Toc422431821)

[4.2. Логически модел на програмната система. 18](#_Toc422431822)

[4.2.1. Use-case диаграми. 18](#_Toc422431823)

[4.2.2 Activity диаграми 19](#_Toc422431824)

[4.3. Архитектура на системата. 21](#_Toc422431825)

[4.4. Организация на данните - концептуален модел на БД. 23](#_Toc422431826)

[5. Реализация 24](#_Toc422431827)

[5.1. Структура на данните - избор на ключови полета, тип и размер. 24](#_Toc422431828)

[5.2. Избор на език и среда за програмиране. 26](#_Toc422431829)

[5.2.1. .NET Framework. 27](#_Toc422431830)

[5.2.2. ASP.NET 29](#_Toc422431831)

[5.2.3. C# 30](#_Toc422431832)

[5.2.4. Visual Studio 2013 31](#_Toc422431833)

[5.2.5. Microsoft SQL Server Management Studio 32](#_Toc422431834)

[5.3. Описание на програмните модули 32](#_Toc422431835)

[5.3.1. Слой на данните (Model). 32](#_Toc422431836)

[5.3.2. Слой за управление потребителския интерфейс (View). 34](#_Toc422431837)

[5.3.3. Слой за функционалността (Controller). 35](#_Toc422431838)

[5.4. Структура и организация на потребителския интерфейс. 36](#_Toc422431839)

[5.5. Формат на входните документи - формален и логически контрол. 37](#_Toc422431840)

[5.6. Формат на извежданите справки. 38](#_Toc422431841)

[6. Инструкции за работа с програмната система. 38](#_Toc422431842)

[6.1. Ръководство за потребителя. 38](#_Toc422431843)

[6.1.1. Ръководство за администратор. 39](#_Toc422431844)

[6.1.2. Ръководство за обикновен потребител. 41](#_Toc422431845)

[6.2. Инструкции и изисквания при инсталиране на системата. 43](#_Toc422431846)

[6.3. Изисквания към апаратното осигуряване 43](#_Toc422431847)

[7. Резултати от тестването на системата. 43](#_Toc422431848)

[8. Основни резултати. 45](#_Toc422431849)

[9. Изводи и препоръки. 45](#_Toc422431850)

**Списък на фигурите**

[Фигура 1 Най-използваните уеб браузъри за 2014 година 9](#_Toc422435063)

[Фигура 2. Модела клиент-сървър. 10](#_Toc422435064)

[Фигура 3. Изпращане на решение в BGCoder. 12](#_Toc422435065)

[Фигура 4. Преглед на резултати в BGCoder. 13](#_Toc422435066)

[Фигура 5. Преглед на състезания. 14](#_Toc422435067)

[Фигура 6. Въведение в даден проблем. 14](#_Toc422435068)

[Фигура 7. Решаване на даден проблем. 15](#_Toc422435069)

[Фигура 8. Видео урок в Code School. 16](#_Toc422435070)

[Фигура 9. Решаване на даден проблем в Code School. 16](#_Toc422435071)

[Фигура 10. Use-case диаграма на потребител с администраторски права. 19](#_Toc422435072)

[Фигура 11. Use-case диаграма на обикновен потребител. 19](#_Toc422435073)

[Фигура 12. Activity диаграма която изобразява процеса на логване. 20](#_Toc422435074)

[Фигура 13. MVC архитектура. 22](#_Toc422435075)

[Фигура 14. Заявка на потребител към уеб приложение с MVC архитектура. 23](#_Toc422435076)

[Фигура 15. Концептуален модел на БД 24](#_Toc422435077)

[Фигура 16. Релационнен модел на базата от данни. 25](#_Toc422435078)

[Фигура 17. Процесът на компилиране на код в .NET 29](#_Toc422435079)

[Фигура 18. ASP.NET технологии. 31](#_Toc422435080)

[Фигура 19. Visual Studio. 33](#_Toc422435081)

[Фигура 20. Клас диаграма на уеб приложението 37](#_Toc422435082)

[Фигура 21. Структура на потребителския интерфейс. 38](#_Toc422435083)

[Фигура 22. Преглед на резултат. 39](#_Toc422435084)

[Фигура 23. Администраторска зона 40](#_Toc422435085)

[Фигура 24. Създаване на състезание 41](#_Toc422435086)

[Фигура 25. Добавяне на задача. 41](#_Toc422435087)

[Фигура 26. Преглед на резултатите в дадено състезание 42](#_Toc422435088)

[Фигура 27. Налични състезания. 42](#_Toc422435089)

[Фигура 28. Присъединяване към състезание. 43](#_Toc422435090)

[Фигура 29. Решение на даден проблем. 43](#_Toc422435091)

[Фигура 30.Подготовка на тестовете. 45](#_Toc422435092)

[Фигура 31. Решение на задача. 45](#_Toc422435093)

**Списък на таблиците**

[Таблица 1. ASPNetUsers 25](#_Toc422435171)

[Таблица 2. UserContests 26](#_Toc422435172)

[Таблица 3. Contests 26](#_Toc422435173)

[Таблица 4. Questions 26](#_Toc422435174)

[Таблица 5. Answers 27](#_Toc422435175)

[Таблица 6. ASPNetUserRoles 27](#_Toc422435176)

[Таблица 7. ASPNetRoles 27](#_Toc422435177)

# **1. Анотация.**

Обект на настоящия дипломен проект е разработване на уеб приложение за проверка на валидността на програмен код. Това представлява автоматизирано приложение което на базата на дадено условие проверява до каква степен е верен кода. Идеята е следната: администратор (учител) създава състезание (изпит) в което могат да участват потребители (студенти/ученици). Към всяко едно състезание може да се добавят неограничен брой проблеми (задачи за решаване) като към всеки един проблем трябва да се прикачат следните неща:

* Условие на задачата
* Десет входни теста
* Десет изходни теста

Когато потребителя изпрати към приложението своя код на базата на тези десет входни теста се генерират съответно десет изходни теста. След което десетте изходни теста (който са качени от администратора) се сравняват с генерираните тестове от потребителя. На базата на това сравнение се оформят точките (оценката) на потребителя. Този процес може да бъде повтарян многократно докато потребителя е удовлетворен от резултати те си (оценката си).

# **2.Увод.**

## **2.1. История на уеб.**

Началото на уеб е поставено от Тим Бърнърс-Лий, сътрудник на Европейската организация за ядрени изследвания (CERN) в Женева, Швейцария. Първите хипертекстови системи се появяват още през 1960-те години, но Бърнърс-Лий пръв прави тази технология достъпна чрез Интернет. През 1989 година той прави предложение за създаване на хипертекстова система и през следващата година, заедно с белгиеца Робер Кайо, започва активна работа по нея. Към края на 1990 година са разработени основните компоненти на World Wide Web – мрежовият протокол HTTP, маркиращият език HTML, първият уеб браузър WorldWideWeb, първият уеб сървър CERN httpd и първите уеб страници, които описват самия проект.

На 6 август 1991 година Тим Бърнърс-Лий публикува съобщение в Usenet групата, с което на практика прави World Wide Web публична услуга, достъпна в Интернет. Месец по-късно в Станфордския линейно ускорителен център започва да функционира първият уеб сървър, разположен извън Европа. През следващите месеци системата постепенно се разпространява в научни центрове в различни части на света. През 1992 година се появяват първите графични браузъри след оригиналния WorldWideWeb, които вече са предназначени за по-широко разпространената операционна система UNIX. Година по-късно се появява браузърът Mosaic, който играе важна роля в популяризирането на World Wide Web. На 30 април 1993 година CERN обявява, че системата ще може да бъде използвана свободно и безплатно от всички. През 1994 година е основана организацията World Wide Web Consortium, включваща различни заинтересовани организации и имаща за цел утвърждаването на технически стандарти, свързани с функционирането на World Wide Web.

През втората половина на 1990-те години с все по-широкото разпространение на достъпа до Интернет World Wide Web започва да се превръща във важно маркетингово средство. Много предприятия създават свои уеб сайтове, появяват се първите специализирани сайтове за електронна търговия.

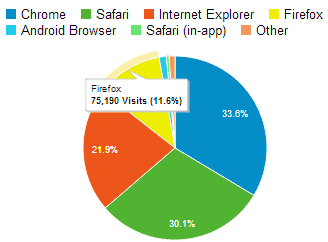
След 2002 г. достъпът до високоскоростен Интернет бързо нараства и се утвърждават успешни бизнес модели, базирани на World Wide Web, като тези на Google, eBay и Amazon.com. През следващите години възникват масови социални мрежи, като MySpace, Twitter и Facebook. Широко се разпространяват форми за споделяне на съдържание, като блоговете, или системи, при които съдържанието се създава и поддържа от самите потребители, като Wikipedia. Този нов модел за обмен на информация между множество индивидуални потребители често се описва с понятието Web 2.0, създадено от O’Reilly Media през 2004 г. Въпреки че терминът подсказва за една нова версия на Мрежата, явлението не се отнася до обновление на Интернет или на техническите стандарти на World Wide Web, а на промени в начините, по които тези стандарти се използват. Навлизат в употреба фреймовете и ярките фонове, както и анимираните елементи с монотонно повтарящо се действие. Думите се заместват от икони и меню бутони. [1]

## **2.2. Уеб страница**

Информацията в World Wide Web се съдържа в хипертекстови документи или информационни ресурси, наричани уеб страници. Достъпът до уебстраниците може да бъде осъществен чрез уеб браузър, а съдържанието им обикновено се показва на монитор на компютър (вкл. на екран на мобилно устройство). Тази информация обикновено се съхранява във формат от тип HTML или XHTML и може да предоставя навигация към други уеб страници чрез хипертекстови връзки. Обикновено уеб страниците включват текст, изображения (картинки, илюстрации, фотографии) и стилове във формат на файла от тип CSS, но може също да включват анимация, мултимедийно съдържание (файл с музика/звук, видеоклип с или без звук) или скриптове, написани на скриптов език (най-масово използвания е JavaScript).

## **2.3. Уеб браузър**

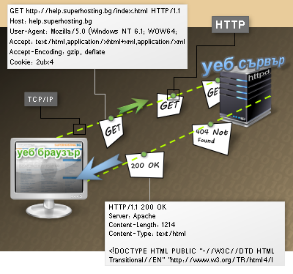
Уеб браузър е софтуерно приложение за извличане и представяне на информационни ресурси в World Wide Web. Информационния ресурс се идентифицира с Uniform Resource Identifier (URI/URL) и може да бъде уеб страница, изображение, видео или друг вид парче с информация. Хипервръските представляват ресурси който позволяват на потребителите лесно да навигират браузъри те си към съответния ресурс. Въпреки че, браузърите са предназначени основно за използване на World Wide Web, те също могат да се използват и за достъп до информация, предоставена от уеб сървъри в частни мрежи или файлове във файлови системи. Най-използваните уеб браузъри са : Firefox, Internet Explorer, Google Chrome, Opera, and Safari.[2]



Фигура 1 Най-използваните уеб браузъри за 2014 година [3]

## **2.4. Уеб сървър.**

Уеб сървърът е приложна програма, сървър, която позволява на даден компютър да предоставя информация на други компютри, под формата на страници с хипертекст използвайки модела клиент-сървър. Той предоставя информация на клиент под формата на HTML документ. Най-използваният клиент, който се свързва с уеб сървъра, е уеб браузърът. За пренасянето на информацията се използва протокола HTTP. Често Уеб сървърът се използва за предоставяне на информация и в други формати — най често това са изображения във форматите PNG, JPEG и GIF както и на XML документи и други. Най-популярните Уеб сървъри са Internet Information Server (IIS) на Microsoft и Apache. [4]



Фигура 2. Модела клиент-сървър.

## **2.5. Сигурност**

Всеки път, когато е зададена заявка от уеб сървъра за дадена страница, той може да я идентифицира и IP адресът, от който е дошла заявката обикновено се записва. Освен ако не е зададено обратното, повечето уеб браузъри записват уеб страницата, която е заявена и показана в историята и обикновено голяма част от съдържанието се кешира локално. Освен ако не се използва криптирането HTTPS, уеб заявките и отговорите, които се изпращат могат да бъдат видени, записани и кеширани от междинни системи.

Когато уеб страницата изисква и потребителят отговаря с личната си информация като истински имена, адрес, имейл, може да се осъществи връзка между текущия уеб трафик и потребителя. Ако уеб сайтът използва HTTP бисквитки, разпознаване на потребителско име и парола или други техники за проследяване, свързването между предишни или последващи уеб посещения с личната информация ще бъде възможно. Така уеб-базирана организация ще може да създаде профил на отделните потребители, които ползват сайта. Ще може да създаде запис за потребителя, който съдържа информация за неговите интереси, хобита, професия и други аспекти на техния демографски профил. Тези профили обикновено представляват интерес за търговци, рекламни агенции и други. В зависимост от условията за ползване на уебсайта и на действащите закони тази информация може да бъде продавана, разпространявана или предавана на други организации без знанието на потребителя. За много хора това значи малко повече от нежелани имейли във входящата поща.

Социалните мрежи правят опити да убедят потребителите си да представят техните истински имена, интереси и местоположение. Това прави преживяването в социалните мрежи по-реалистично и така задържа повечето си потребители. От друга страна, източниците на някои снимки и необмислени твърдения могат да бъдат идентифицирани и по-късно могат да съжаляват за публикуването им. Онлайн хулигани могат да използват лична информация, за да тормозят и преследват потребители. Модерните социални мрежи предоставят добър контрол на личните настройки за всяка отделна публикация.

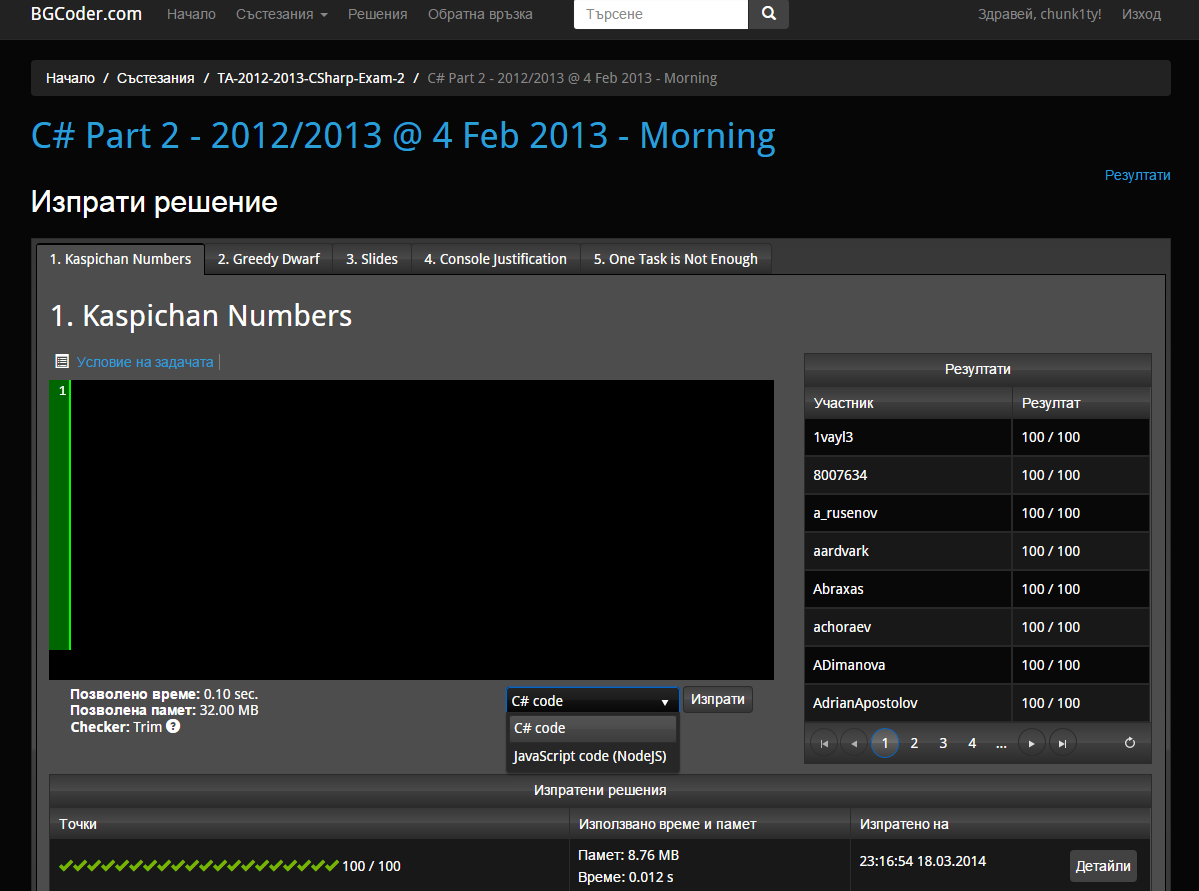
Някои снимки и видеота, публикувани в сайтове причиняват проблеми, тъй като лицето на даден човек може да бъде добавено в онлайн профил. С техника за разпознаване на лица, ще бъде възможно лице да бъде свързано с друго, до преди това анонимно. Заради кеширането и копирането на изображения премахването на снимка от уеб пространството е трудно.

# **Обзор на съществуващите решения. Изводи. Цел и задачи.**

В процеса за работа бяха разгледани и други уеб приложения за проверка на валидността на програмен код.

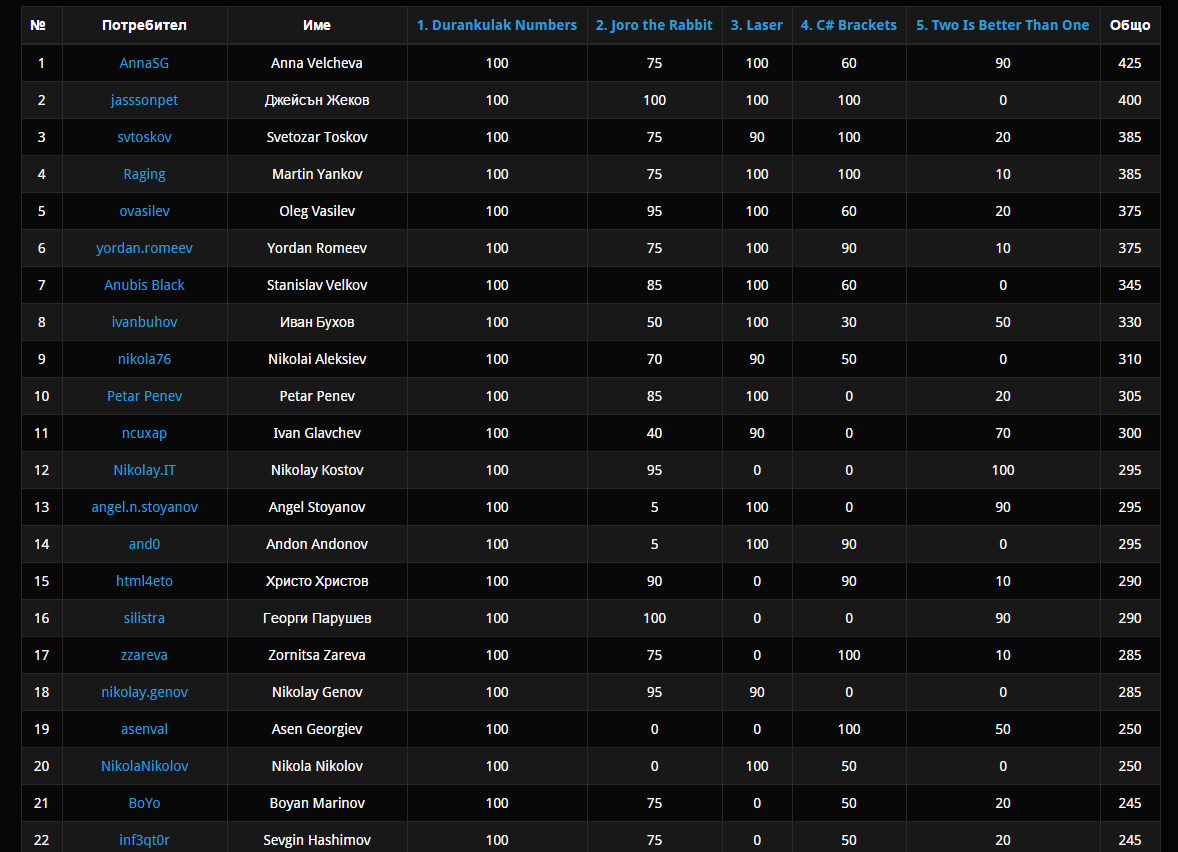
## **3.1. BGCoder.**

Началната страница на приложението показва активните състезание както и последните интересни новини свързани с него. BGCoder предлага по-богата функционалност при създаване на състезания и задачи към тях както и по-приятен потребителски интерфейс. Потребителя може да избира какъв език да използва за разрешаването на определен проблем като при създаването авторът определя кой са позволените езици за програмиране. Също така единственото условие за проверка на дадена задача е да се изпрати само кода който я решава (не се изпраща .exe файл). [5]



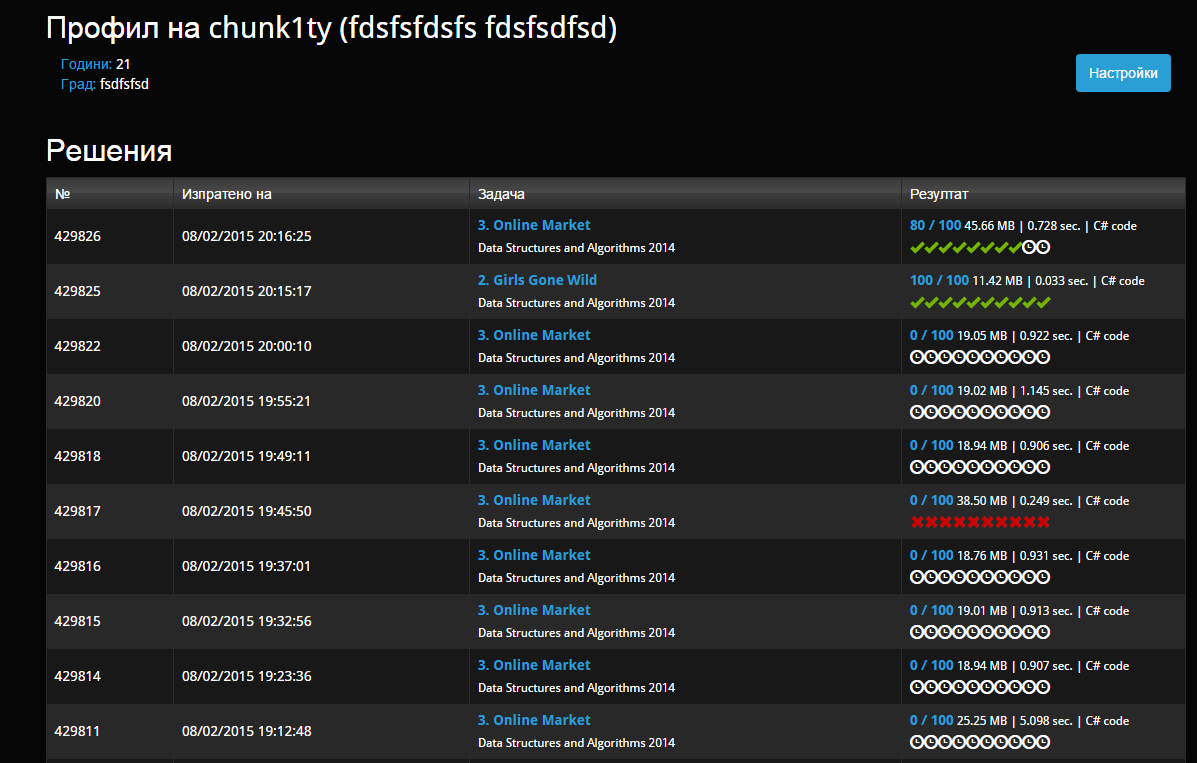
Фигура 3. Изпращане на решение в BGCoder.

Също така се дава възможност на потребителят да преглежда резултатите на другите участници в състезанието в което участва. Резултатите са визуализирани в приятен за потребителя вид.



Фигура 4. Преглед на резултати в BGCoder.

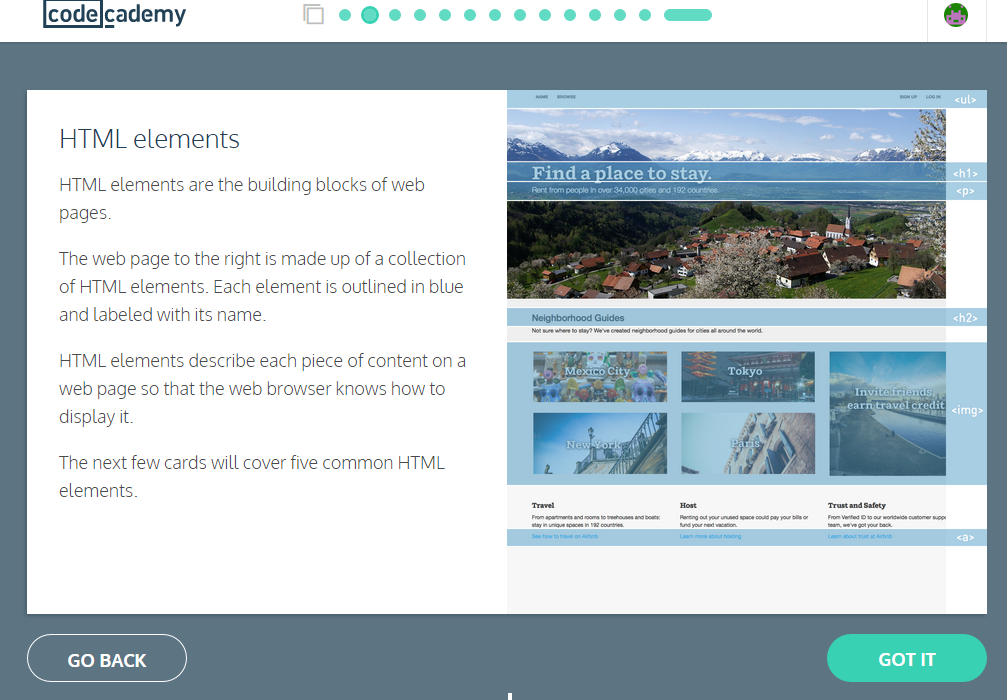
Предлага се възможност за преглед на всички състезание в който е участвал.



Фигура 5. Преглед на състезания.

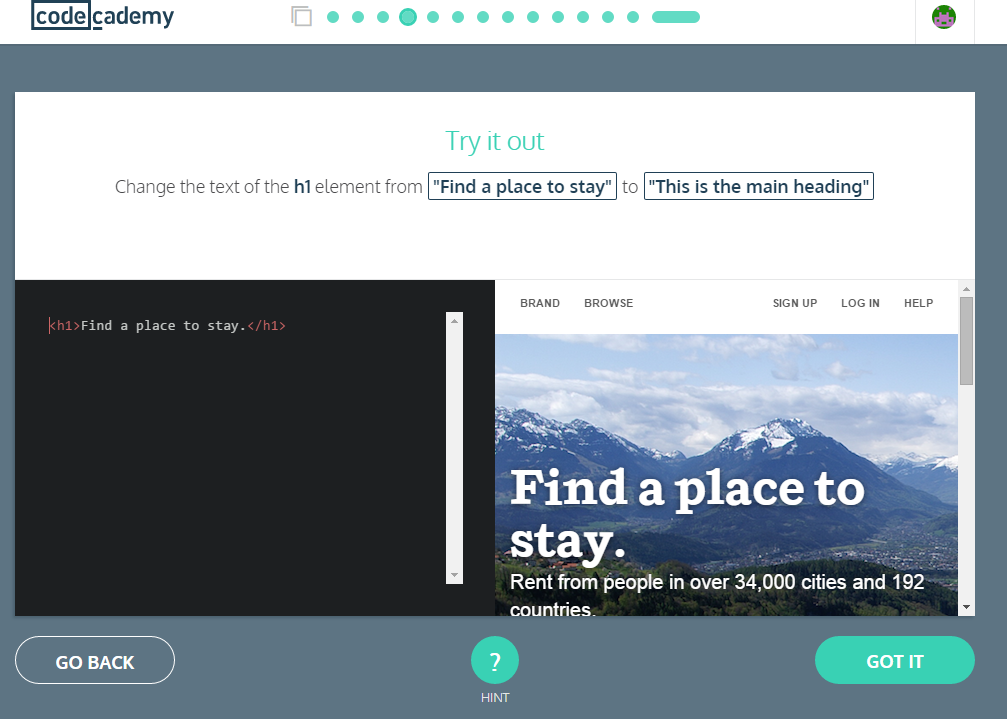
## **Codecademy.**

Най-популярния сайт за проверка на валидността на програмен код. Уеб приложението позволява валидация на богат набор от програмни езици като: HTML, CSS, JavaScript, jQuery, PHP, Python и Ruby. Формата на сайта не е състезателен а по- скоро с образователна цел. Всеки потребител може да избере някой от посочените езици и да пробва своите умения. Преди всяка задача има кратка въведителна част в която се описва даден проблем и начин по който да се разреши. [6]



Фигура 6. Въведение в даден проблем.

Ако задачата е сложна, потребителя има възможността да използва подсказка чрез натискане на бутона „Hint“.

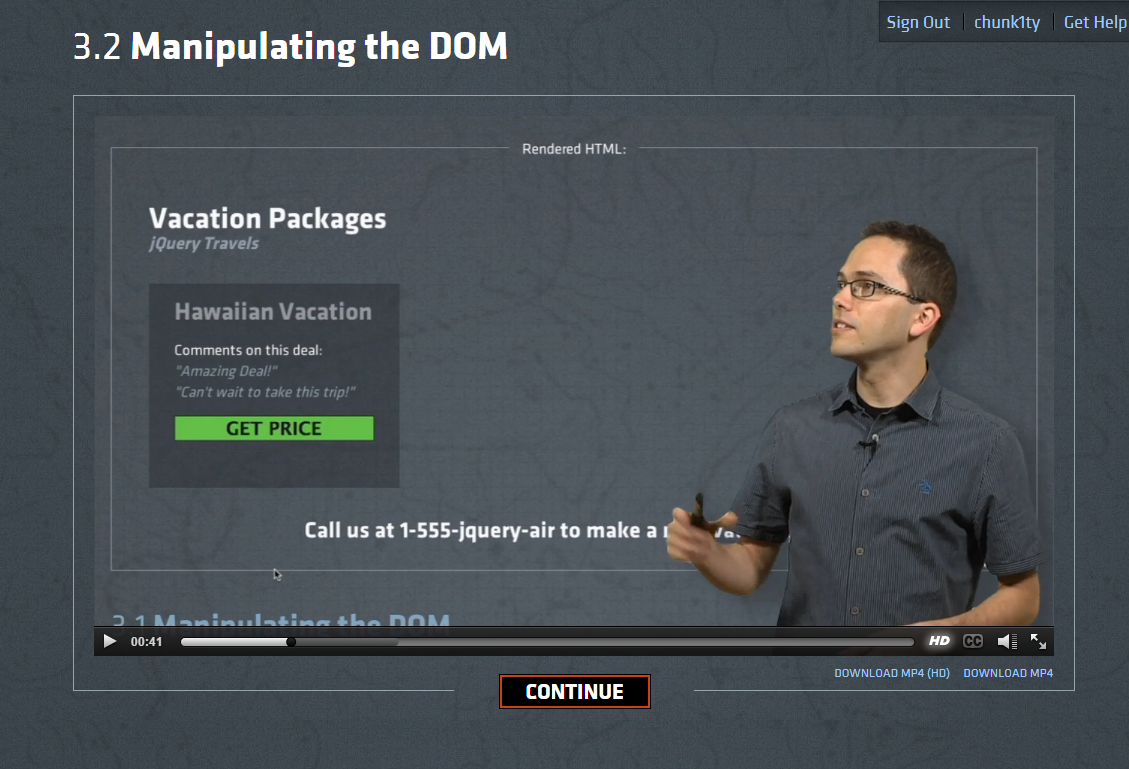


Фигура 7. Решаване на даден проблем.

С напредване на трудността приложението предлага и платени курсове който биха подобрили неговите умения.

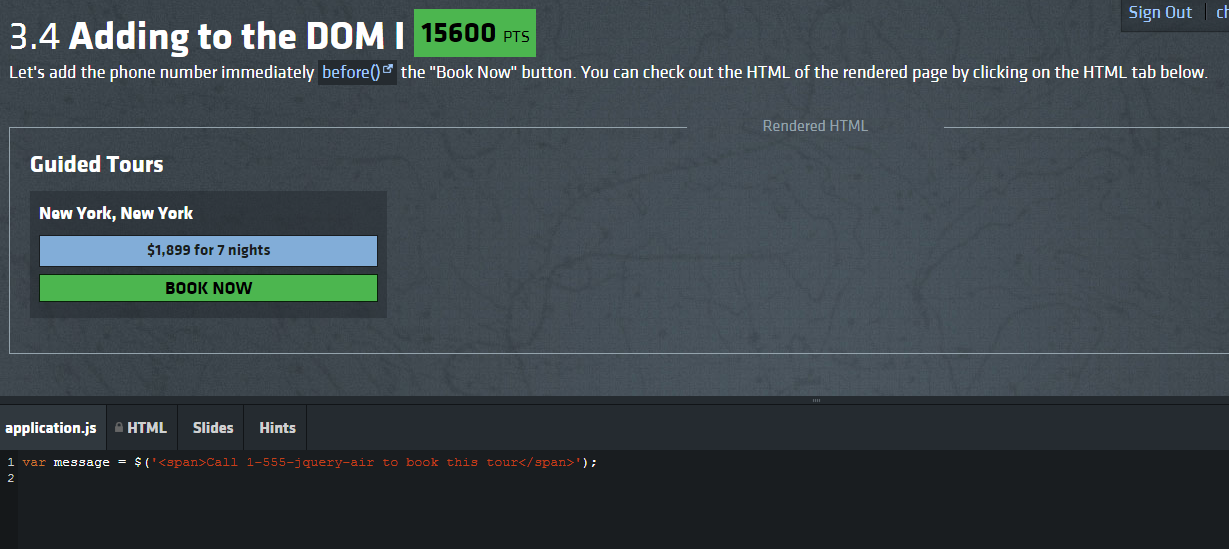
## **Code School**

Code School е уеб приложение също с образователен характер доста подобно на Codecademy. Предлага разнообразни курсове като: SQL, JavaScript, jQuery, Ruby и други. Потребителският интерфейс е приятен и улеснява работата с приложението. Подобно на Codecademy всеки курс започва с кратко въведение, но за разлика от него въведението е видео файл със средна продължителност около 4 минути. [7]



Фигура 8. Видео урок в Code School.

След като свърши видео файлът е ред на потребителя да пробва своите умения. И тук има възможност за подсказка. След всяка задача потребителя натрупва точки. При използване на подсказка се отнемат 75 точки. При събиране на определен брой точки се печелят различни награди предимно значки.



Фигура 9. Решаване на даден проблем в Code School.

Приложението също предлага и платени курсове с който потребителите биха подобрили значително техните умения.

## **Цел и задачи.**

Целта на проекта е да се проектира и реализира уеб приложение за проверка на валидността на програмен код, което да притежава интуитивен интерфейс но и да поддържа основните функционални възможности. За постигане на целта трябва да се изпълнят следните задачи:

* Да се направи преглед ба съществуващите решения и сравнителен анализ между тях.
* Да се формират функционалните изисквания към приложението.
* Да се избере най-подходящия интерфейс за приложението, като се залага основно на опростеността и добрата му функционалност.
* Да се избере подходящ начин за графично представяне на въведените от потребителя данни, така че те да са лесно четими и разбираеми.
* Да се избере подходяща архитектура на приложението.
* Да се изберат най-подходящите технологии за неговата реализация.
* Да се реализира и тества функционалността на приложението.

# 

# **Проектиране и описание на предлаганото решение.**

## **4.1. Изисквания към програмната система.**

Приложението има две групи потребители: администратори и обикновени потребители(студенти/ ученици). За да може да изпълнява пълноценно функциите си, програмната система трябва да предложи следните функции:

За двете групи:

* Да се предоставя на двете групи потребители възможност за създаване на акаунти.
* Да се предоставя на двете групи потребители възможност за редактиране на профилите си.

За администратори:

* Да се предлага възможност за създаване на състезание(изпит).
* Да се предлага възможност за изтриване на състезание(изпит).
* Да се предлага възможност за редактиране на състезание(изпит).
* Да се предлага възможност за създаване на задачи към дадено състезание(изпит)
* Да се предлага възможност за преглед на резултатите по дадена задача.
* Да се предлага възможност за преглед на резултатите на даден потребител (студент/ученик) в конкретно състезание.

За потребителите(студенти/ученици):

* Да се предлага възможност за присъединяване към дадено състезание.
* Да се предоставя възможност за описание (условие) на задачата.
* Да се предлага възможност за решаване конкретна задача.
* Да се визуализират точките на потребителя след решаването на дадена задача.
* Да се дава възможност за повторно решаване на проблема ако потребителя не е удовлетворен от постигнатите резултати.

## **4.2. Логически модел на програмната система.**

### **4.2.1. Use-case диаграми.**



Фигура 10. Use-case диаграма на потребител с администраторски права.



Фигура 11. Use-case диаграма на обикновен потребител.

### **4.2.2 Activity диаграми**

Activity диаграма, която описва достъпването да дадена страница от уеб приложението. Също така показва как потребителя взаимодейства с приложението в рамките на този процес.



Фигура 12. Activity диаграма която изобразява процеса на логване.

## 

## **4.3. Архитектура на системата.**

Приложението е реализирано чрез архитектурния шаблон за дизайн Model-View-Controller (MVC). Характерно за него е, че се разделя бизнес логиката от графичния интерфейс и данните в дадено приложение. Въпреки, че първоначално е разработен за персонални компютри, Model View Controller е широко приет като архитектура за World Wide Web приложения във всички основни езици за програмиране.

Модел - ядрото на приложението, съобразено с областта, за която се разработва. Обикновено това са данните от реалния свят, които сме моделирали и над които искаме да работим - да въвеждаме, променяме, визуализираме и т.н. Трябва да се направи разлика между реалния обкръжаващ ни свят и въображаемия абстрактен моделен свят, който е продукт на нашият разум, който ние възприемаме във вид на твърдения, формули, математическа символика , схеми и други помощни средства. Типичен пример за това е класа Question.cs, който описва въпросите за дадено състезание.

Изглед - тази част от изходния код на приложението, отговорна за показването на данните от модела. Например изгледът може да се състои от PHP шаблонни класове, JSP страници, ASP страници, JFrame наследници в Swing приложение. В текущия проект са използвани ASP страници.

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 в метода Index() който се намира в StudentController се вземат данните за всички текущи състезания и се подават към Изгледа където се визуализират. Също така когато се прави уеб графичен интерфейс това би довело до много лесна модификация на HTML кода дори от човек, който не е програмист - той ще гледа на шаблона просто като на обикновена HTML страница. [8]



Фигура 13. MVC архитектура.

Предимства:

* Моделът е независим от контролера и изгледа
* Моделът може да бъде планиран и осъществен независимо от другите части на системата
* За един и същи модел могат да бъдат осъществени различни изгледи.
* Контролерът и изгледът могат да бъдат променени, без да се налага промяна в модела.

Недостатъци

* Софтуерната система достига ново ниво на сложност, което я прави по-трудно разбираема за софтуерните разработчици, работещи по нея.

Примерно изпълнение на MVC при уеб приложение би било следното:

Потребителя изпраща заявка към сървъра посредством HTTP протокола. Тази заявка се обработва от Контролера. Ако има нужда Контролера взема нужните данни от модела ако ли не направо връща резултата към Изгледа. В крайна сметка точно този изглед е нещото което потребителя получава като отговор на неговата заявка. Визуално изглежда така:



Фигура 14. Заявка на потребител към уеб приложение с MVC архитектура.

## **4.4. Организация на данните - концептуален модел на БД.**

Уеб приложението ще обработва данни свързани с проверка на валидността на програмен код, а именно име на състезание, име на задача, изкарани точки върху дадена задача. Организацията на данните, необходима за реализацията на приложението, е показана чрез концептуален модел на базата данни.

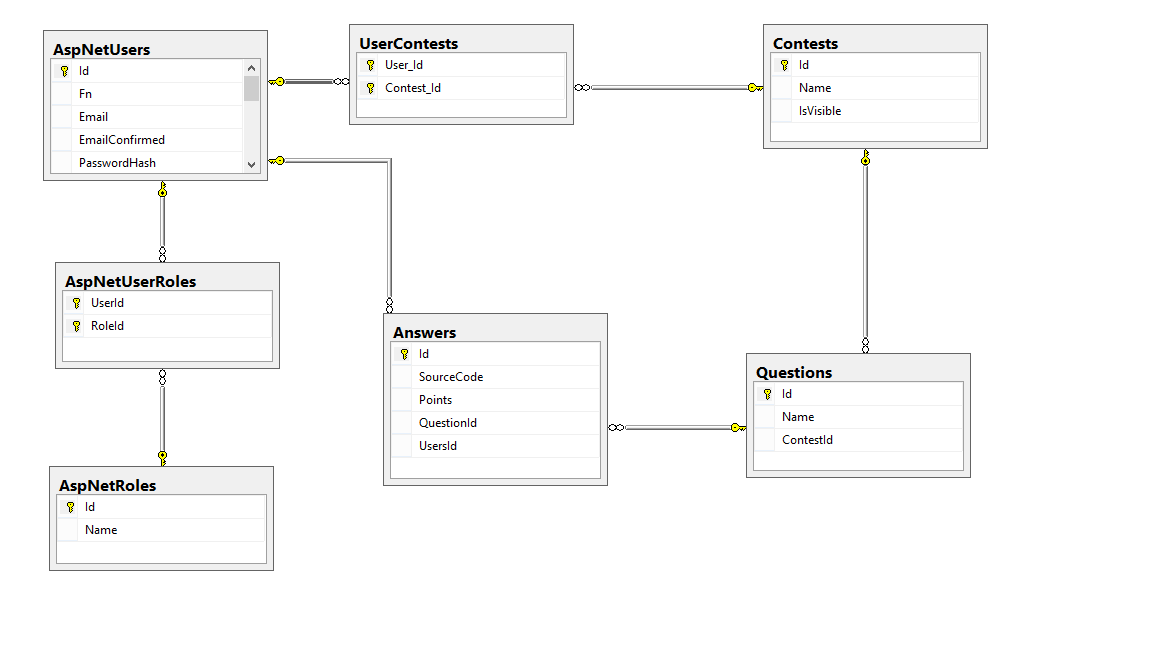


Фигура 15. Концептуален модел на БД

# **Реализация**

## **5.1. Структура на данните - избор на ключови полета, тип и размер.**

Реализацията на концептуалния модел на базата от данни е показана, чрез релационния ѝ модел, като в него са маркирани всички ключови полета за отделните таблици, както и чуждите такива.



Фигура 16. Релационен модел на базата от данни.

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | nvarchar(128) | not null,  **primary key** | Идентификационен номер на потребителя. |
| Fn | int | not null | Факултетен номер на студента. |
| Email | nvarchar(40) | not null | E-mail на потребителя, който служи за логване в приложението. |
| EmailConfirmed | bit | not null | Потвърждение на e-mail адреса. |
| PasswordHash | nvarchar(20) | not null | Хешираната парола на потребителят. |

Таблица 1. ASPNetUsers

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| User\_Id | nvarchar(128) | not null,  **primary key** | Идентификационен номер на потребителя. |
| Contest\_id | int | not null,  **primary key** | Идентификационен номер на състезанието. |

Таблица 2. UserContests

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на състезанието. |
| Name | nvarchar(100) | not null | Име на състезанието. |
| IsVisible | bit | not null | Видимо ли е състезанието. |

Таблица 3. Contests

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на въпроса. |
| Name | nvarchar(40) | not null | Име на въпроса. |
| ContestId | int | not null,  **foreign key** | Идентификационен номер на състезанието. |

Таблица 4. Questions

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на отговора. |
| SourceCode | nvarchar(MAX) | not null | Кодът който е отговор на въпроса. |
| Points | Int | not null | Точките който потребителя получава за отговора си. |
| QuestionId | int | not null,  **foreign key** | Идентификационен номер на въпроса. |
| UserId | nvarchar(128) | not null,  **foreign key** | Идентификационен номер на потребителя. |

Таблица 5.Answers

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| UserId | nvarchar(128) | not null,  **primary key** | Идентификационен номер на потребителя. |
| RoleId | nvarchar(128) | not null,  **primary key** | Идентификационен номер на ролята. |

Таблица 6. ASPNetUserRoles

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | nvarchar(128) | not null,  **primary key** | Идентификационен номер на ролята. |
| Name | nvarchar(50) | not null | Име на ролята. |

Таблица 7. ASPNetRoles

## **Избор на език и среда за програмиране.**

Реализацията на приложението в уеб програмирането се разделя на две : сървърна част и клиентска част. При клиентската част се използват езиците HTML, CSS и предимно JavaScript. Важно е да се отбележи, че при тази част за радост или съжаление няма голям избор на езици от което се стига до извода, че тези три езика са задължителни при изграждането на уеб приложение. При сървърната част обаче нещата не са така. „Голямата тройка“ (PHP, Java и APS.NET) са основните езици за програмиране, но също така съществуват и не по малко популярните Python(който е най-използвания език за програмиране за 2014 година), Ruby и Perl.

Настоящият дипломен проект е реализиран с помощта на ASP.NET технологиите и по конкретно с архитектурата MVC, който са част от платформата .NET Framework 4.5.

**Основни средства при реализацията:**

* Платформа: .NET Framework 4.5.
* Потребителски интерфейс: ASP.NET 5.
* Архитектура: MVC.
* Програмен език: C#.
* Уеб сървър: IIS 8.5.
* Среда за програмиране: Visual Studio 2013 Ultimate.
* Среда за работа със СУБД: Microsoft SQL Server Management Studio 2014 Professional

### **5.2.1. .NET Framework.**

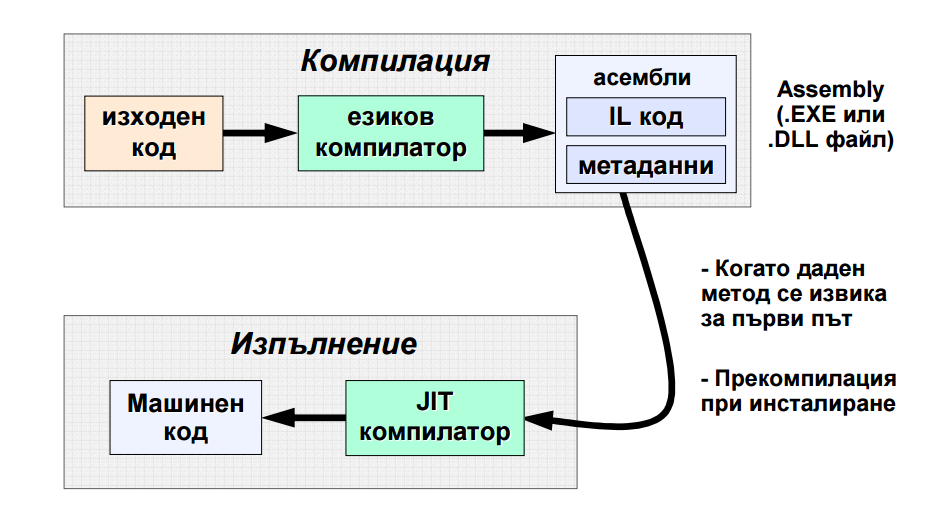
.NET Framework е съвременна платформа за разработка и изпълнение на приложения, която предоставя програмен модел, стандартна библиотека от класове(Framework Class Library) и среда за контролирано изпълнение на програмен код(Common Language Runtime). Тя поддържа различни езици за програмиране и позволява тяхната съвместна работа. .NET приложенията се пишат на езици от високо ниво (C#, Visual Basic, C++ и др.) и се компилират до междинен език от ниско ниво, наречен IL (Intermediate Language). По време на изпълнение IL програмите се компилират до инструкции за текущата хардуерна архитектура, съобразени с текущата операционна система, и след това се изпълняват от микропроцесора. Можем да разделим .NET Framework на два основни компонента:

* **Common Language Runtime (CLR)** – средата, в която се изпълнява

управляваният код на .NET приложенията. Представлява виртуална машина, която контролирано изпълнява .NET кода и осигурява различни услуги, като управление на сигурността, управление на паметта и др. Основните задачите които CLR изпълнява са:

* Изпълнение на IL кода;
* Управление на паметта и ресурсите;
* Управление на сигурността и осигуряване безопасността на типовете;;
* Управление на изключенията, управление на конкурентостта;
* Взаимодействие с неуправляващ код;
* Подпомагане процесите на дебъгване и оптимизиране.

При компилация на изходния код от някой от програмните езици на платформата, получаваме асембли. То представлява изпълним файл, съдържащ .NET управляван код и мета данни, които описват съдържанието на асемблито - имената на класовете и типовете, информация за членовете на класовете (методи, полета, свойства и други). При изпълнение на дадено асембли, CLR го зарежда в паметта, анализира мета данните му, извършва проверки на кода – дали е коректен спрямо IL стандарта, дали има необходимите права за изпълнение и др. След това с помощта на JIT (Just-In-Time) компилатор, междинният език се компилира до машинен код за текущия процесор. Компилираният вече код след това се изпълнява директно от процесора. Предимството на JIT компилацията е, че може да оптимизира кода за текущата хардуерна платформа по най-добрия начин. При неуправляващия код това не е възможно, защото кодът се компилира така, че да работи върху всички процесори, без да използва пълните възможности на текущата хардуерна платформа. По тази причина в някои случай управляваният код може да е дори по-бърз от неуправляващия, въпреки нуждата от JIT компилация, която отнема време.



Фигура 17. Процесът на компилиране на код в .NET

* **Framework Class Library (FCL)** - стандартната библиотека на .NET

Framework, в която се съдържат няколко хиляди дефиниции на типове, които предоставят богата функционалност при изграждането на .NET приложения: вход/изход, връзка с бази данни, работа с XML, изграждане на уеб-приложения, използване на уеб-услуги, изграждане на графичен потребителски интерфейс и др. [9]

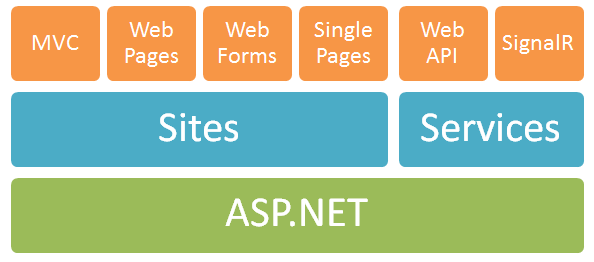
### **5.2.2. ASP.NET**

ASP.NET е уеб рамка (framework), разработена от "Майкрософт". За първи път е публикуван през януари 2002 година с версия 1.0 на .NET Framework, и е наследник на Microsoft Active Server Pages (ASP) технология. ASP.NET е изградена въз основа на Common Language Runtime (CLR), което позволява на програмистите да пишат ASP.NET код като използват .NET език по избор. За първи път е публикуван през януари 2002 година с версия 1.0 на .NET Framework, и е най-успешен наследник на Microsoft Active Server Pages (ASP) технология.

ASP.NET цели по добра производителност спрямо останалите скрипт-базирани технологии като компилира сървърно кода в един или повече DLL файлове на Уеб сървъра. Тази компилация става автоматично когато страницата бива заредена за пръв път (което от своя страна означава, че програмистът не трябва да изпълнява отделни компилации за страниците). Тази характеристика осигурява лекота на разработване предлагана от скриптовите езици с производителността на бинарните операции. Трябва да се има предвид обаче, че самата компилация може да причини забележимо забавяне при потребителя когато редактираната страница бива изискана за пръв път от Уеб сървъра, но това забавяне не би се появило отново преди следваща промяна.

Необходимите ресурси се поставят на виртуален хост на Internet Information Services сървър (или друг сървър съвместим с ASP.NET). Когато потребителят за пръв път поиска да зареди страница, .NET Framework парсва и компилира файловете в .NЕТ асембли и изпраща отговор; при последвалите зареждания се извикват от DLL файловете. ASP.NET има възможността да компилира цели сайтове на части от по 1000 файла при първо зареждане. Ако забавянето е значително или причинява проблеми, големината на самите части може да бъде променяна. [10]

На фигурата са показани технологиите който към настояща дата са част от ASP.NET.



Фигура 18. ASP.NET технологии.

### **5.2.3. C#**

Езикът C# е съвременен, обектно-ориентиран, типизиран език, с широка поддръжка на идеите на компонентно-ориентирания подход за разработка. C# е наследник на езика C++, но не наследява всичко, а само част от синтаксиса и някои негови силни страни. C# поддържа синтаксис за дефиниране и използване на свойства и събития, които играят важна роля при дефинирането и използването на компоненти.[11] Въпреки, че .NET Framework поддържа много други езици, за реализацията е избран езикът C# по няколко причини:

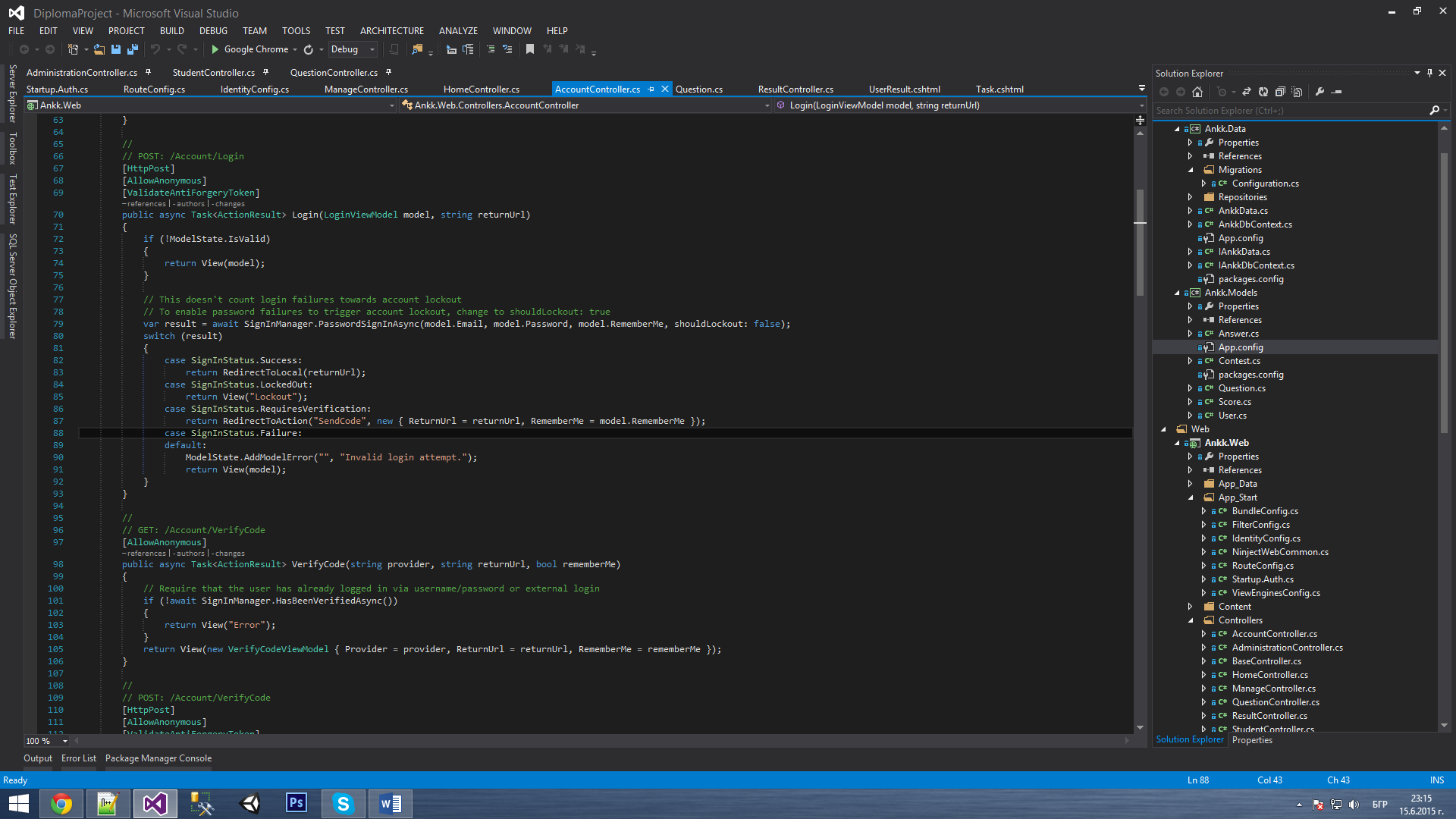
* Архитектите на езика са го проектирали специално за платформата .NET Framework, като още по време на дизайна са го съобразили с нейните особености. C# притежава максимално стегнат и ясен синтаксис, наследява простотата на Java, мощността на C++ и силните черти на Delphi.
* Езици като C++ и JavaScript не са проектирани специално за .NET Framework, а са адаптирани допълнително към него чрез редица изменения и добавки. В следствие на това те запазват някои синтактични особености, които не са удобни при работата с .NET.
* Общността на C# разработчиците е по-добре развита, отколкото на разработчиците на другите .NET езици. Поради голямата популярност на езика C# за него има по-добра поддръжка от инструментите за разработка.
* В България C# е най-популярният от .NET езиците и се използва най-масово в българските софтуерни компании.

### **5.2.4. Visual Studio 2013**

За реализацията на дипломния проект е използвано Microsoft Visual Studio. Visual Studio е мощно интегрирано средство за разработка на софтуерни приложения за Windows и за платформата .NET Framework. Използва се за разработка на конзолни и графични потребителски интерфейс приложения, както и Windows Forms или WPF приложения, уеб сайтове, уеб приложения и уеб услуги на всички поддържани платформи от Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Visual Studio предоставя 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. Могат да се добавят и плъгини, които повишават функционалността на почти всяко ниво — включително добавянето на поддръжка за source-control системи (като Subversion, Visual SourceSafe, GitHub и други), добавяне на нови инструменти като редактори и визуални дизайнери за domain-specific languages или инструменти за други аспекти (като например: Team Foundation Server, Team Explorer).

Visual Studio поддържа различни езици за програмиране (C#, VB.NET, C/C++, F#, XML/XSLT, HTML/XHTML, JavaScript, CSS и други) и различни технологии за разработка на софтуер (Win32, COM, ASP.NET, ADO.NET Entity Framework, Windows Forms, WPF, Silverlight и още десетки други Windows и .NET технологии).



Фигура 19. Visual Studio.

### **5.2.5. Microsoft SQL Server Management Studio**

С помощта на Microsoft SQL Server Management Studio реализацията и тестването на уеб приложението бяха по лесни. Management Studio е интегрирана среда за достъп, конфигуриране, управление и проектиране на Microsoft SQL Server. Комбинира широка група графични инструменти с множество скриптови редактори, за да осигури достъп на разработчиците и администраторите до SQL Server. Основната характерна черта на SQL Server Management Studio е обектния изследовател (Object Explorer), който позволява на потребителите търсене и изследване на обектите в сървъра.

## **5.3. Описание на програмните модули**

### **5.3.1. Слой на данните (Model).**

Под модул: Модели

В този под модул е имплементирана базата от данни и по точно всяка една от таблиците и взаимовръзките между тях.

Модел: Отговор

Предназначение: Имплементира таблица Answers.

Зависи от: БД.

Реализация: Answer.cs

Модел: Състезание

Предназначение: Имплементира таблица Contests.

Зависи от: БД.

Реализация: Contest.cs

Модел: Въпрос

Предназначение: Имплементира таблица Questions.

Зависи от: БД.

Реализация: Questions.cs

Модел: Потребител

Предназначение: Имплементира таблица AspNetUsers.

Зависи от: БД.

Реализация: User.cs

Под модул: База от данни

Обединява в едно всички таблици с помощта на дизайнерски шаблони (Repository pattern and Unit of Work). Благодарение на тях кодът става по гъвкав, независим и лесен за тестване.

Под модул: Контекст

Предназначение: Създават се пропъртита от моделите и с помощта на Entity Framework се създава базата.

Зависи от: БД.

Интерфейс: IAnkkDbContext.cs

Реализация: AnkkDbContext.cs

Под модул: Данни

Предназначение: Обединяване на всички хранилища в едно цяло (в този модул е имплементиран Unit of Work шаблона).

Зависи от: БД.

Интерфейс: IAnkkDbData.cs

Реализация: AnkkDbData.cs

Под модул: Хранилище

Предназначение: Позволява извършването на основните CRUD операции(в модула е имплементиран Repository шаблона).

Зависи от: БД.

Интерфейс: IGenericRepository.cs

Реализация: GenericRepository.cs

**public IQueryable<T> All()** - връща всички резултати от базата.

**public IQueryable<T> SearchFor(Expression<Func<T, bool>> conditions) -** търси по дадени критерии запис в базата.

**public void Add(T entity) -** добавя запис в базата.

**public void Update(T entity) -** променя запис в базата.

**public void Delete(T entity)** **-** изтрива запис от базата (като му се подаде конкретен запис)

**public T Delete(object id)** **-** изтрива запис от базата (като му се подаде съответния идентификационен номер)

### **5.3.2. Слой за управление потребителския интерфейс (View).**

В този слой се визуализира информацията в подходящ за потребителя начин. Всички „Изгледи“ се намират в папка Views в проекта. Основния шаблон е \_Layout.cshtml и се намира в папката Shared. Основните „Изгледи “ са :

Изглед: Акаунт

Предназначение: Визуализира основните операции с създаване, редактиране на потребителски акаунт.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Account.

Изглед: Администрация

Предназначение: Визуализира основните операции с създаване, редактиране на състезания както и създаване на задачи към дадено състезание. Важно е да се отбележи, че потребителя трябва да има администраторски права.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Administration.

Изглед: Въпрос

Предназначение: Използва се за създаване на задачи. Потребителя трябва да има администраторски права.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Question.

Изглед: Резултат

Предназначение: Визуализира резултатите на участниците в дадено състезание.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Result.

Изглед: Студент

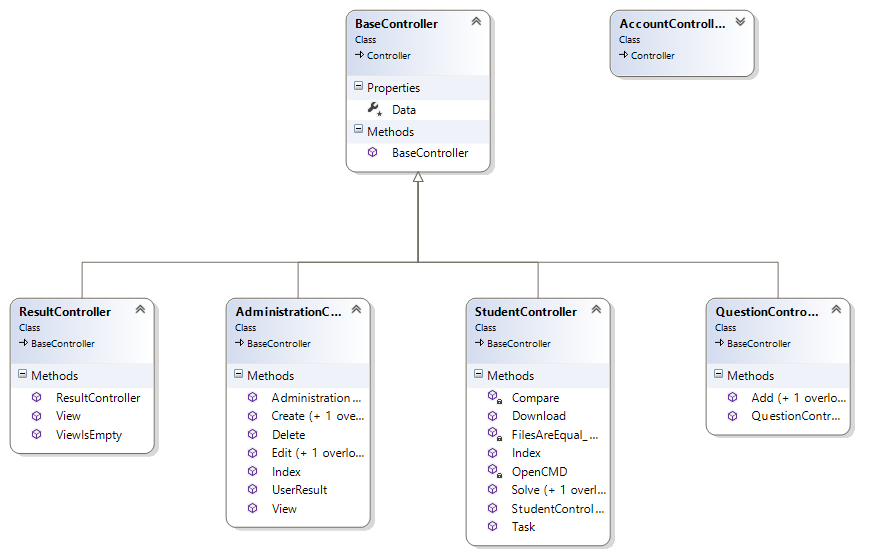
Предназначение: Дава възможност на обикновения потребител да участва в дадено състезание и да решава конкретните проблемите свързани с него.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Student.

### **5.3.3. Слой за функционалността (Controller).**

На фигура .. е представена клас диаграмата на уеб приложението. Важно е да се отбележи, че всички контролери наследяват класа Controller който се предоставя от самата архитектура (намира се в System.Web.Mvc).



Фигура 20. Клас диаграма на уеб приложението

Контролер: Акаунт

Предназначение: Извършва основните операции с създаване, редактиране на потребителски акаунт.

Зависи от: БД.

Реализация: AccountController.

Контролер: Администрация

Предназначение: Извършва основните операции с създаване, редактиране на състезания както и операциите свързани с създаване на задачи.

Зависи от: БД.

Реализация: AdministrationController.

Контролер: Въпрос

Предназначение: Извършва операция за създаване на задачи.

Зависи от: БД.

Реализация: QuestionController.

Контролер: Резултат

Предназначение: Извършва операциите чрез който се визуализират резултатите на участниците в дадено състезание.

Зависи от: БД .

Реализация: ResultController.

Контролер: Студент

Предназначение: Извършва операциите с който обикновения потребител има възможността да участват в дадено състезание и да решава проблемите му.

Зависи от: БД .

Реализация: StudentController.

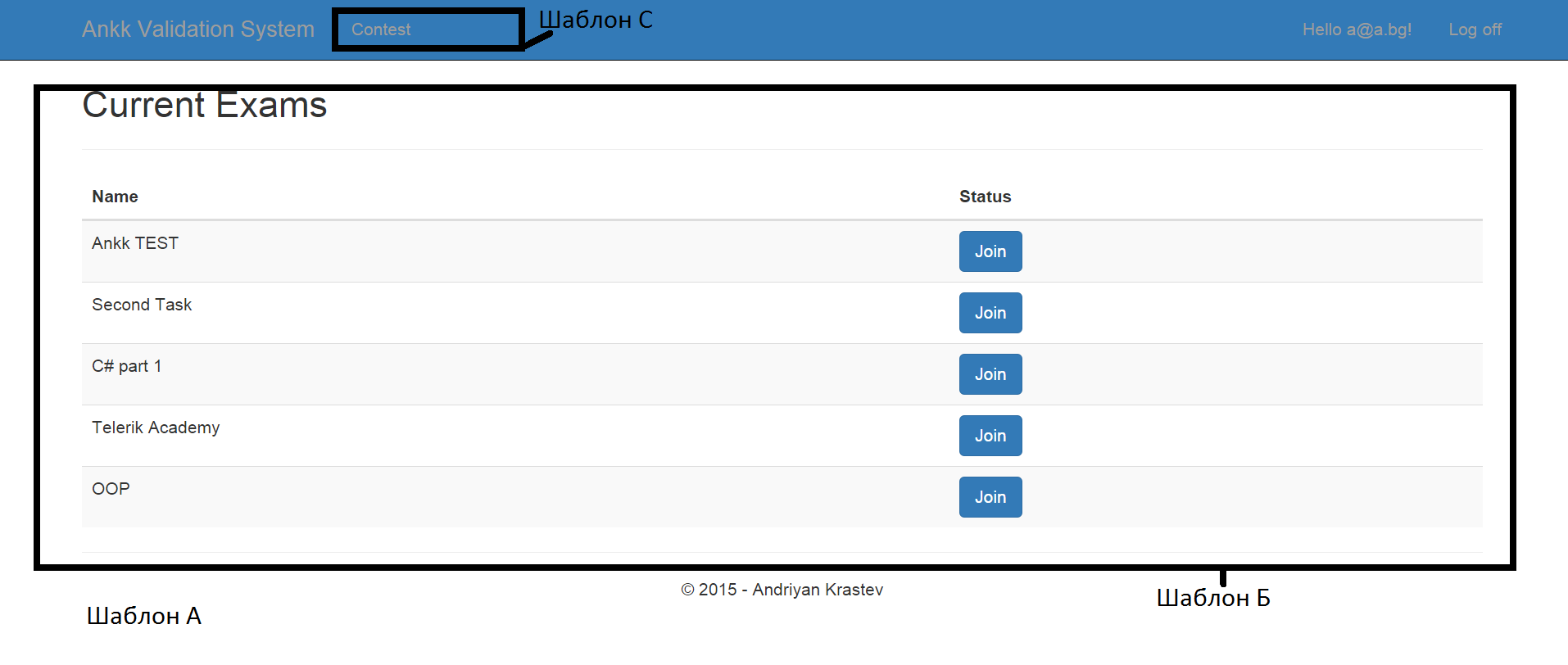
Контролер: Базов

Предназначение: Предоставя на останалите контролери, който го наследяват, да използват базата от данни. Също така решава едновременния проблем (concurrency problem).

Зависи от: БД .

Реализация: BaseController.

## **5.4. Структура и организация на потребителския интерфейс.**



Фигура 21. Структура на потребителския интерфейс.

Шаблон А е главния шаблон който се използва в цялото уеб приложение. Реализиран е в \_Layout.cshtml. Динамичните области за него са шаблон Б и шаблон С. Шаблон Б показва възможността на потребителя да избере към кое състезание да се присъедини. Ако потребителя има администраторски права то тогава във шаблон C, ще се появи още една възможност (Admin) чрез която ще има право да създава нови състезания както и създаването на определени проблеми по негово желание.

## **5.5. Формат на входните документи - формален и логически контрол.**

Регистрация на потребител:

**Е-mail:** При въвеждане се извършва проверка дали отговаря на даден регулярен израз.

**Факултетен номер:** Цяло число между 100000 и 999999. При въвеждане се извършва проверка за неговата валидност.

**Парола:** Символи между 6 и 100. При въвеждане се извършва проверка за неговата валидност.

Изпращане на решение на даден проблем:

**Код:** Минимално позволената дължина е 20 символа, а максимално няма.

**.Exe файл:** Проверява се дали файла е добавен.

Създаване на състезание:

**Име:** Символи между 3 и 60. При въвеждане се извършва проверка за неговата валидност.

**Видимост:** Извършва се проверка дали е използвано.

Създаване на задача:

**Име:** Символи между 3 и 60. При въвеждане се извършва проверка за неговата валидност.

**Описание:** Извършва се проверка дали е добавен файла.

**Входни данни:** Извършва се проверка дали са добавени файловете.

**Изходни данни**: Извършва се проверка дали са добавени файловете.

## **5.6. Формат на извежданите справки.**

Резултатите на даден участник в състезание се визуализират в приятен за потребителя вид, като се показват точките и кода който е използвал при решаването на проблема . Потребителите с администраторски права имат възможността да видят всичките решени проблеми на конкретен потребител в дадено състезание.



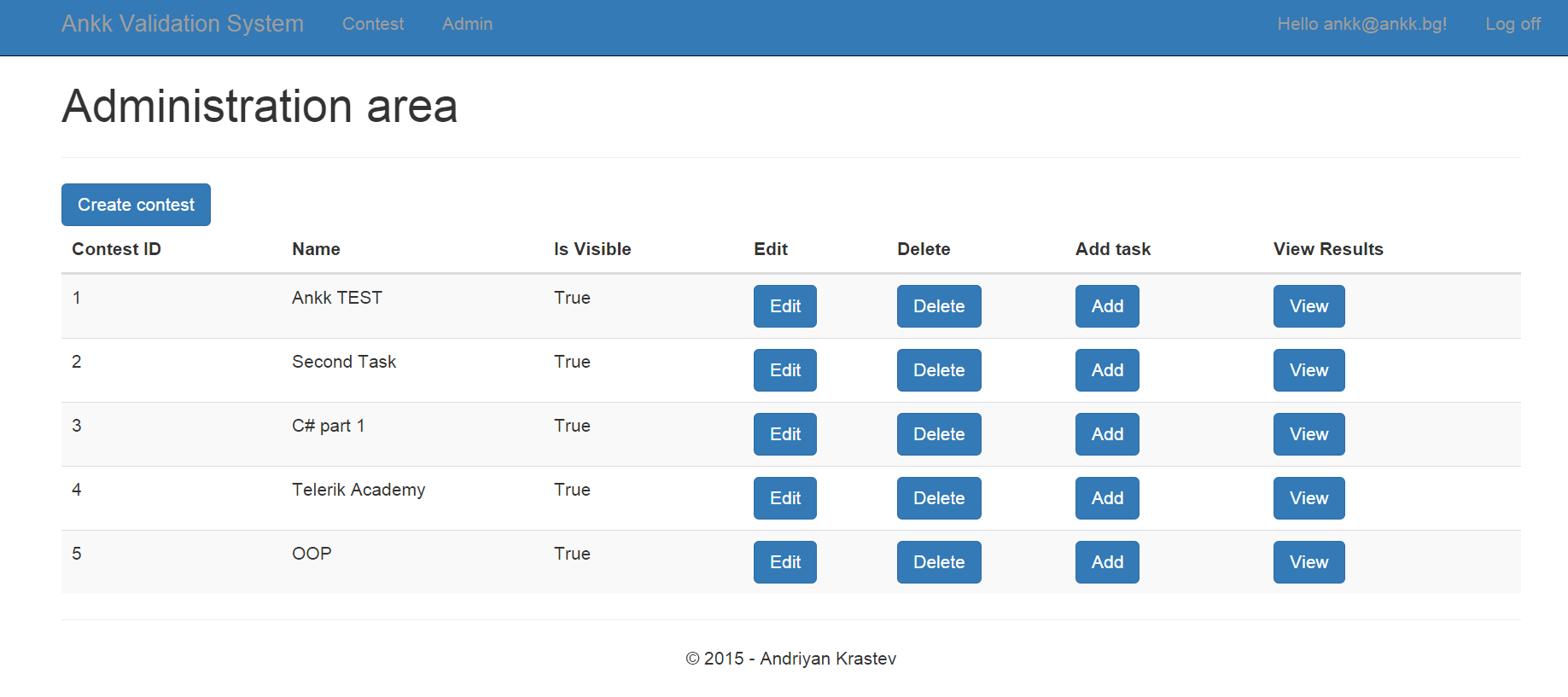
Фигура 22. Преглед на резултат.

# **Инструкции за работа с програмната система.**

## **6.1. Ръководство за потребителя.**

### **6.1.1. Ръководство за администратор.**

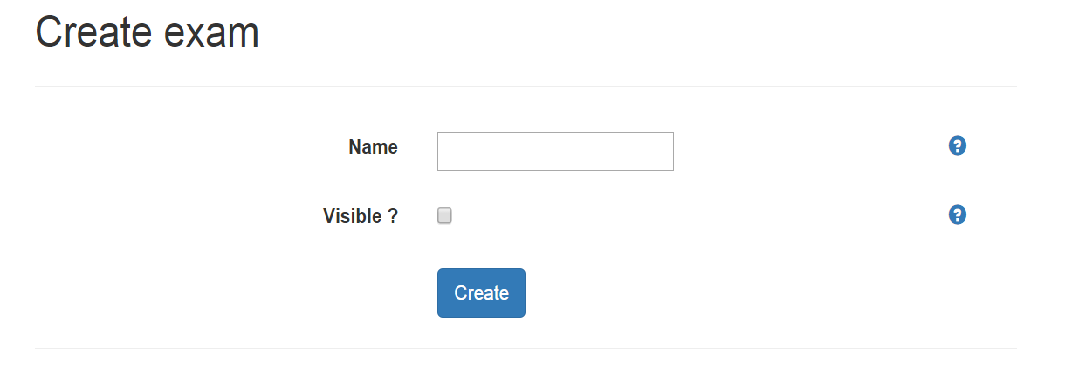
След логване в системата началната страница изглежда по следния начин.



Фигура 23. Администраторска зона

#### **6.1.1.1. Създаване на ново състезание**

След натискане на бутона „Create contest“ ще се появи възможност за създаване на ново състезание. В новия прозорец се дава име на състезанието и дали да е видимо за обикновените потребители.

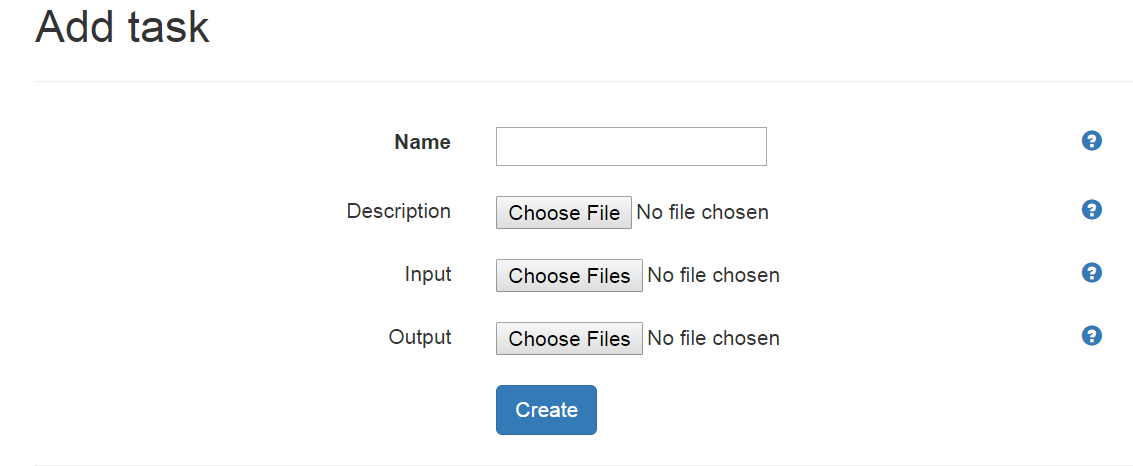


Фигура 24. Създаване на състезание

#### **6.1.1.2. Създаване на задача към състезание.**

След създаването на състезание уеб приложението препраща администратора към началната страница. На нея се визуализира името на създаденото състезание и дали е видимо. На същия ред се натиска бутона „Add“ чрез който се създава нова задача към състезанието. В новия прозорец трябва да се попълнят следните неща:

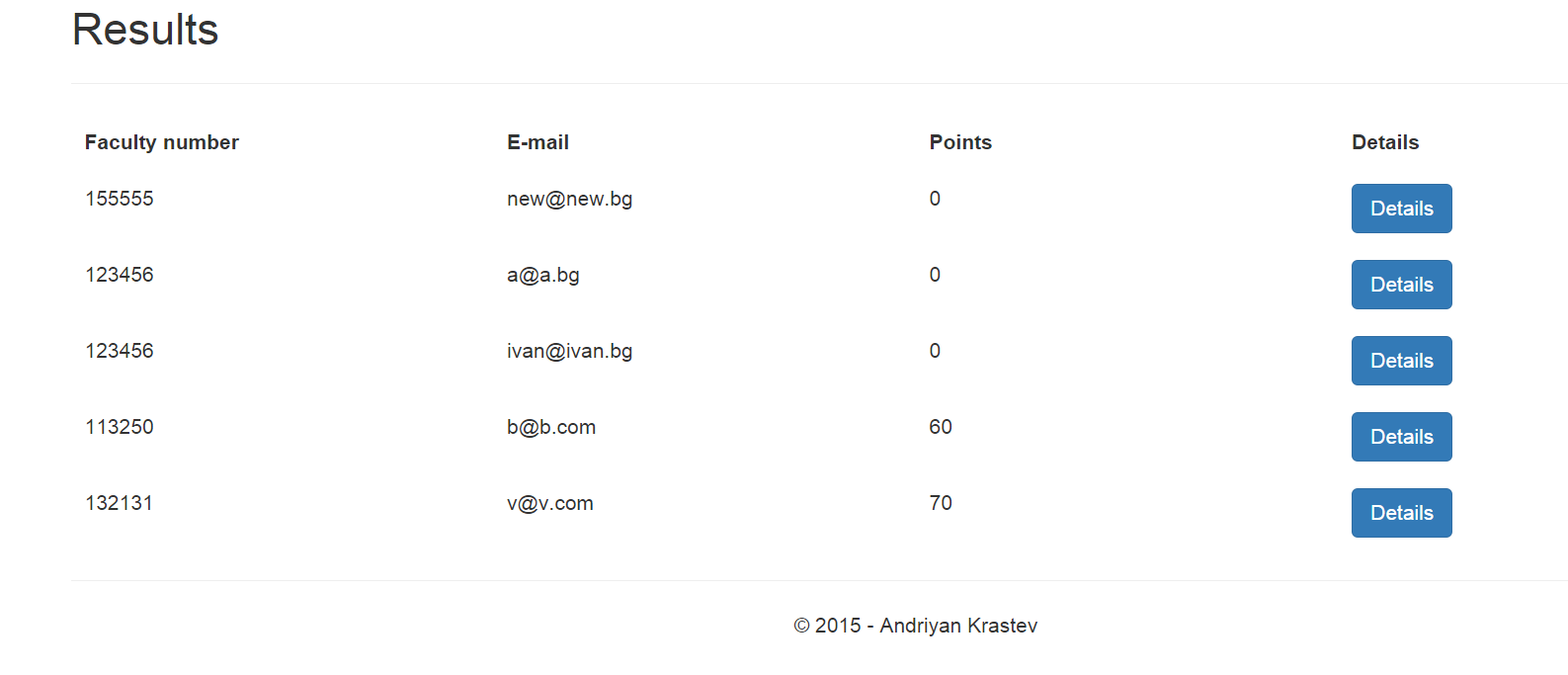
* Име на задачата
* Описание на задачата – текстов файл на който е описан конкретния проблем. Формата може да е .txt, .pdf или .doc.
* Входни данни – входните данни който се подават на задачата. Те трябва да са десет на брой във .txt формат и да са наименувани по следния начин**: test.i.out.txt** където i се изменя от 1 до 10.
* Изходни данни - изходните данни който са решение на входните тестове. Те трябва да са десет на брой във .txt формат и да са наименувани по следния начин: **test.i.in.txt** където i се изменя от 1 до 10. Номера на входния тест отговаря на номера на изходния тест.



Фигура 25. Добавяне на задача.

#### **6.1.1.3. Манипулация на състезание.**

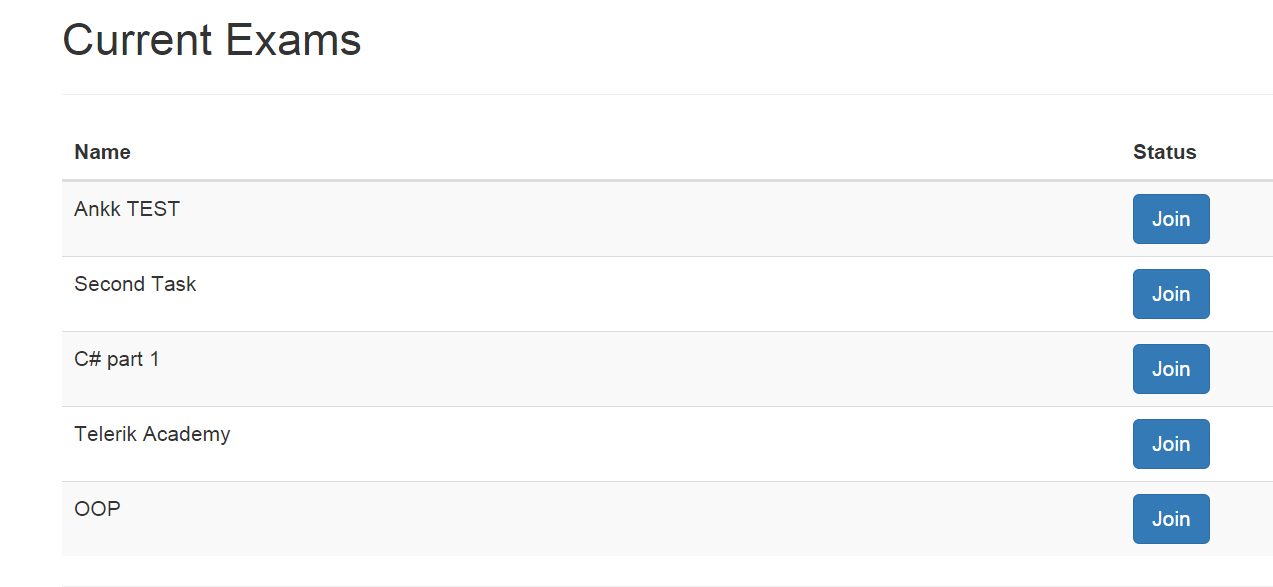
Манипулацията на дадено състезание става от началната страница. Всяко едно състезание може да се изтрива, променя, добавя нова задача и да се проверят резултатите на участниците в него.



Фигура 26. Преглед на резултатите в дадено състезание

### **6.1.2. Ръководство за обикновен потребител.**

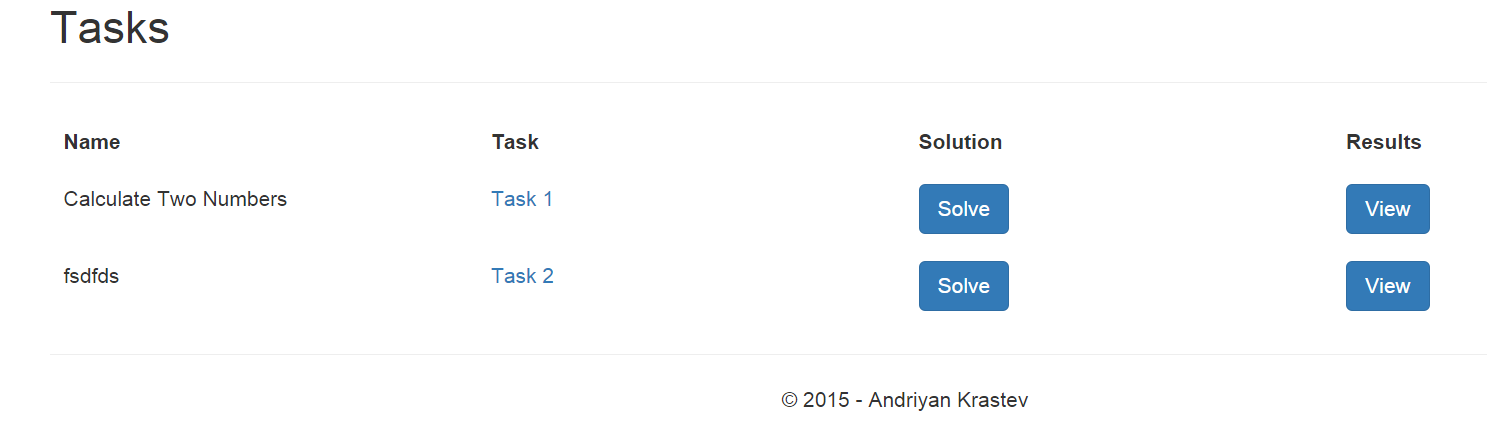
След логване в системата началната страница изглежда по следния начин.



Фигура 27. Налични състезания.

#### **6.1.2.1. Присъединяване към състезание.**

От началната страница потребителя избира в кое състезание да се присъедини и натиска върху бутона „Join“. В ново появилият се прозорец се визуализира информация за всяка една задача която е част от състезанието. В колоната „Task“ се намира описанието на всеки един проблем.



Фигура 28. Присъединяване към състезание.

#### **6.1.2.2. Решаване на задача.**

След като потребителя реши дадения проблем и натисне бутона “Solve” трябва да прикачи следните неща:

* В полето „My Code“ се слага решението на задачата.
* Upload file – избира се .exe файла който е решението на проблема.

Всички езици за програмиране който изкарват като резултат .exe файл могат да бъдат използвани.

След като се изпрати решението уеб приложението го проверява и показва на потребителя до каква степен е вярно.



Фигура 29. Решение на даден проблем.

## **6.2. Инструкции и изисквания при инсталиране на системата.**

За инсталиране на приложението са нужни следните неща:

* Visual Studio 2013
* .NET Framework 4.5
* Microsoft SQL Server Management Studio 2014
* Internet Information Services 8+

Приложението се стартира с помощта на Visual Studio. Базата от данни може да се прегледа от Management Studio.

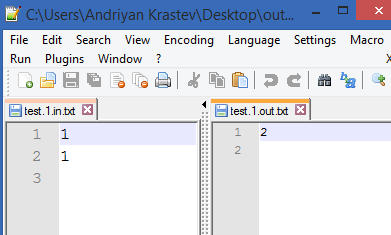
## **6.3. Изисквания към апаратното осигуряване**

За да се използва приложението е необходимо наличието на компютър с достъп до Интернет. Приложението работи адекватно с най използваните уеб браузъри –Chrome, Firefox и Internet Explorer.

# **Резултати от тестването на системата.**

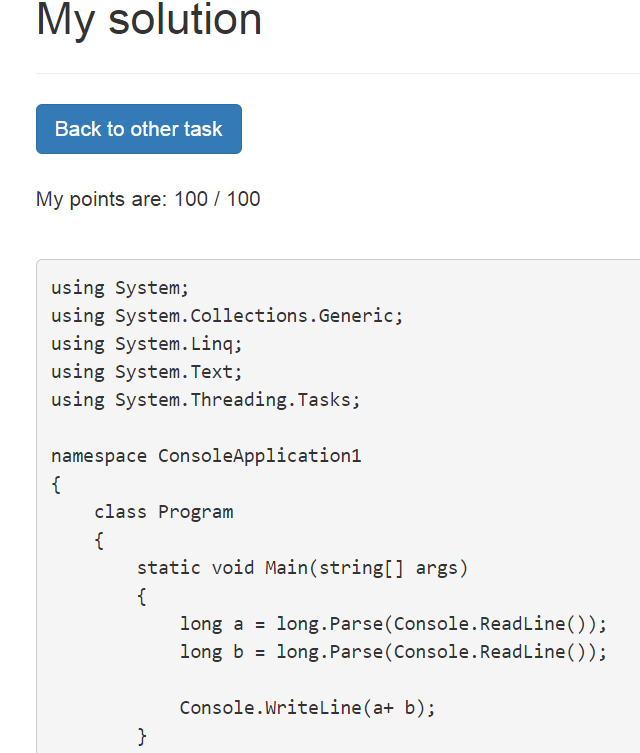
За тестване на приложението е необходимо да бъде създадено състезание и няколко задачи към него от потребител с администраторски права с цел тестване на цялата функционалност.

Нека нашето състезание се казва „Basic programing expressions“. Създаваме си задача към него, нека се казва „Sum of two numbers”, която събира две числа. Създаваме си входните тестове, както и изходните като внимаваме за тяхното именуване. Пример: във файла test.1.in.txt добавяме числата 1 и 1 (всяка на отделен ред), а в изходния файл добавяме 2. По същия начин създаваме още 9 теста.



Фигура 30.Подготовка на тестовете.

Състезанието е готово! В него има само една задача. Остава обикновения потребител да влезе в профила си и да реши „сложната“ задача.



Фигура 31. Решение на задача.

При коректно решена задача потребителя получава 100 точки. Добавянето на задачи към дадено състезание е неограничено.

# **8. Основни резултати.**

Проектирано и реализирано е уеб приложение за проверка на валидността на програмен код. То позволява на различни функционалности на различните групи потребители. За администраторите: създаване на състезания и задачи към тях, а обикновените потребители имат възможността се присъединят към тях и да решат поставените проблеми. Въз основа на решенията си обикновените потребители изкарват точки. Максималния брой точки за една задача е 100. Администраторите от своя страна могат: да променят състезанията; да преглеждат резултатите във всяко едно от тях; да преглеждат резултатите и решенията на всеки един потребител.

# **9. Изводи и препоръки.**

Разработеното уеб приложение предоставя на възможността за автоматична проверка на код при определено условие с определени тестове.

В бъдеще се планира функционалността на приложението да бъде разширена с :

* Премахване на .exe файла при изпращане на решение.
* Добавяне на филтри при администраторската зона с цел по лесно намиране на определен студент.
* Възможност за промяна на дадена задача.
* По лесен начин за качване на входните и изходни данни.

# **10. Източници.**

[1] <https://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1>

[2]<https://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80>

[3]<https://muse.union.edu/espanae/2014/03/24/top-browsers-winter-2014/>

[4]<http://help.superhosting.bg/what-is-web-server.html>

[5]<http://bgcoder.com/>

[6]<http://www.codecademy.com/>

[7]<https://www.codeschool.com/>

[8]<https://bg.wikipedia.org/wiki/Model-View-Controller>

[9]<http://www.devbg.org/dotnetbook/Nakov-Programming-.NET-Framework-Book-Volume-1-ver-1.03.html>

[10]<https://en.wikipedia.org/wiki/ASP.NET>

[11]<https://bg.wikipedia.org/wiki/C_Sharp>