**Русенски университет „Ангел Кънчев“**

Катедра „Компютърни системи и технологии“

**ДИПЛОМЕН ПРОЕКТ**

**ТЕМА**: „ Уеб базирана информационна система за анкети и гласувания“

**Изготвил:** Андриян Невелинов Кръстев

**Фак. ном.:** 153708

**Група:** 26

**Специалност:** КСТ

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

Русе

2016 г.

**Съдържание**

[**1. Анотация.** 5](#_Toc454047559)

[**2.Увод.** 5](#_Toc454047560)

[**2.1. История на уеб.** 5](#_Toc454047561)

[**2.2. Уеб страница** 6](#_Toc454047562)

[**2.3. Уеб браузър** 7](#_Toc454047563)

[**2.4. Уеб сървър.** 7](#_Toc454047564)

[**2.5. Сигурност** 8](#_Toc454047565)

[**3.** **Обзор на съществуващите решения. Изводи. Цел и задачи.** 9](#_Toc454047566)

[**3.1.** **Survey.bg** 9](#_Toc454047567)

[**3.2.** **SurveyMonkey** 11](#_Toc454047568)

[**3.3.** **Surveyplanet** 13](#_Toc454047569)

[**3.4.** **Цел и задачи.** 15](#_Toc454047570)

[**4.** **Проектиране и описание на предлаганото решение.** 15](#_Toc454047571)

[**4.1. Изисквания към програмната система.** 16](#_Toc454047572)

[**4.2. Логически модел на програмната система.** 16](#_Toc454047573)

[**4.2.1. Use-case диаграми.** 16](#_Toc454047574)

[**4.2.2. Activity диаграми** 17](#_Toc454047575)

[**4.3. Архитектура на системата.** 19](#_Toc454047576)

[**4.4. Организация на данните - концептуален модел на БД.** 22](#_Toc454047577)

[**5.** **Реализация** 22](#_Toc454047578)

[**5.1. Структура на данните - избор на ключови полета, тип и размер.** 22](#_Toc454047579)

[**5.1.** **Избор на език и среда за програмиране.** 26](#_Toc454047580)

[**5.2.1. .NET Framework.** 26](#_Toc454047581)

[**5.2.2. ASP.NET** 28](#_Toc454047582)

[**5.2.3. C#** 29](#_Toc454047583)

[**5.2.4. Visual Studio 2015** 30](#_Toc454047584)

[**5.2.5. Microsoft SQL Server Management Studio** 31](#_Toc454047585)

[**5.3. Описание на програмните модули** 31](#_Toc454047586)

[**5.3.1. Слой на данните (Model).** 32](#_Toc454047587)

[**5.3.2. Слой за управление потребителския интерфейс (View).** 34](#_Toc454047588)

[**5.3.3. Слой за функционалността (Controller).** 35](#_Toc454047589)

[**5.4. Структура и организация на потребителския интерфейс.** 37](#_Toc454047590)

[**5.5. Формат на входните документи - формален и логически контрол.** 37](#_Toc454047591)

[**5.6. Формат на извежданите справки.** 38](#_Toc454047592)

[**6.** **Инструкции за работа с програмната система.** 39](#_Toc454047593)

[**6.1. Ръководство за потребителя.** 39](#_Toc454047594)

[**6.1.1. Ръководство за администратор.** 39](#_Toc454047595)

[**6.1.2. Ръководство за обикновен потребител.** 42](#_Toc454047596)

[**6.2. Инструкции и изисквания при инсталиране на системата.** 43](#_Toc454047597)

[**6.3. Изисквания към апаратното осигуряване** 43](#_Toc454047598)

[**7.** **Резултати от тестването на системата.** 43](#_Toc454047599)

[**8. Основни резултати.** 45](#_Toc454047600)

[**9. Изводи и препоръки.** 45](#_Toc454047601)

[**10. Източници.** 46](#_Toc454047602)

**Списък на фигурите**

[Фигура 1 Най-използваните уеб браузъри за 2015 година 7](#_Toc422435063)

[Фигура 2. Модела клиент-сървър. 8](#_Toc422435064)

[Фигура 3. Създаване на анкета в Surveybg.. 10](#_Toc422435065)

[Фигура 4. Създаване на въпроси и отговори в Surveybg.. 11](#_Toc422435066)

[Фигура 5. Завършена анкета в Surveybg.. 11](#_Toc422435067)

[Фигура 6. Начална страница в SurveyMonkey.. 12](#_Toc422435068)

[Фигура 7. Създаване на въпроси и отговори в SurveyMonkey.. 13](#_Toc422435069)

[Фигура 8. Завършена анкета в SurveyMonkey. 13](#_Toc422435070)

[Фигура 9. Начална страница в Surveyplanet.. 14](#_Toc422435071)

[Фигура 10. Създаване на въпроси и отговори в Surveyplanet.. 15](#_Toc422435071)

[Фигура 11. Преглед на анкета в Surveyplanet.. 15](#_Toc422435071)

[Фигура 12. Завършена анкета в Surveyplanet... 15](#_Toc422435071)

[Фигура 13. Use-case диаграма на потребител с администраторски права. 16](#_Toc422435072)

[Фигура 14. Use-case диаграма на обикновен потребител. 18](#_Toc422435073)

[Фигура 15. Activity диаграма която изобразява процеса на логване. 19](#_Toc422435074)

[Фигура 16. Activity диаграма която изобразява процеса на попълване на дадена анкета. 20](#_Toc422435074)

[Фигура 17. MVC архитектура. 20](#_Toc422435075)

[Фигура 18. Заявка на потребител към уеб приложение с MVC архитектура. 22](#_Toc422435076)

[Фигура 19. Концептуален модел на БД 24](#_Toc422435077)

[Фигура 20. Релационнен модел на базата от данни. 25](#_Toc422435078)

[Фигура 21. Процесът на компилиране на код в .NET 29](#_Toc422435079)

[Фигура 22. ASP.NET технологии. 31](#_Toc422435080)

[Фигура 23. Visual Studio. 33](#_Toc422435081)

[Фигура 24. Клас диаграма на уеб приложението 37](#_Toc422435082)

[Фигура 25. Структура на потребителския интерфейс. 38](#_Toc422435083)

[Фигура 26. Преглед на резултат. 39](#_Toc422435084)

[Фигура 27. Администраторска зона 40](#_Toc422435085)

[Фигура 28. Създаване на на анкета 41](#_Toc422435086)

[Фигура 29. Създаване на въпроси и отговори. 41](#_Toc422435087)

[Фигура 30. Добавяне на учасници 42](#_Toc422435088)

[Фигура 31. Имайл съобщение за начало на анкета. 42](#_Toc422435089)

[Фигура 32. Активни анкети. 43](#_Toc422435090)

[Фигура 33. Въпроси и отговори. 43](#_Toc422435091)

[Фигура 34. Успешно попълнена анкета 45](#_Toc422435092)

[Фигура 35. Начален панел. 45](#_Toc422435092)

[Фигура 36. Преглед на въпроси 46](#_Toc422435092)

[Фигура 37. Добавяне на учасник 47](#_Toc422435092)

[Фигура 38. Резултати от анкетирането. 48](#_Toc422435093)

**Списък на таблиците**

[Таблица 1. ASPNetUsers 25](#_Toc422435171)

[Таблица 2. Participants 26](#_Toc422435172)

[Таблица 3. ParticipantAnswers 26](#_Toc422435173)

[Таблица 4. RateSystems 26](#_Toc422435174)

[Таблица 5. Questions 27](#_Toc422435175)

[Таблица 6. QuestionAnswers 27](#_Toc422435176)

[Таблица 7. ASPNetUserRoles 28](#_Toc422435176)

[Таблица 8. ASPNetRoles 28](#_Toc422435177)

# **1. Анотация.**

Обект на настоящия дипломен проект е разработване на уеб базирана информационна система за анкети и гласувания. Приложението цели улесняването на целият процес при гласуване, като го направи електронно. Всеки потребител може да попълва анкетите навсякъде по всяко време (дори и от личният си телефон). Попълването става еднократно без право на корекции. Администраторът (преподавателят) избира участниците (студентите) във всяка една анкета както и продължителността и. Броят на въпросите, отговорите и участниците във всяка една анкета е неограничен. Въпросите могат да са два типа – с единичен отговор, с множествен отговор. След като се добавят участници има възможност да се уведомят (чрез изпращане на имейли) кога е началото и краят на анкетата в която са добавени. Администраторът получава постоянна информация как протича анкетата, като му се предоставя подробна информация за всеки един въпрос.

# **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.

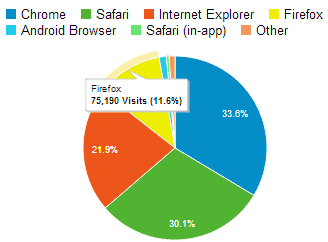
След 2002 г. достъпът до високоскоростен Интернет бързо нараства и се утвърждават успешни бизнес модели, базирани на World Wide Web, като тези на Google, eBay и Amazon.com. През следващите години възникват масови социални мрежи, като MySpace, Twitter и Facebook. Широко се разпространяват форми за споделяне 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**

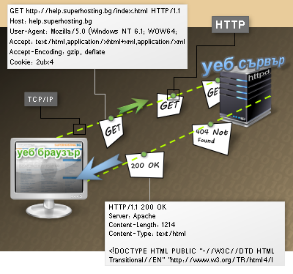
Уеб браузър е софтуерно приложение за извличане и представяне на информационни ресурси в World Wide Web. Информационния ресурс се идентифицира с Uniform Resource Identifier (URI/URL) и може да бъде уеб страница, изображение, видео или друг вид парче с информация. Хипервръските представляват ресурси който 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, които са показани във вид на диаграма на фигура 1.[2]



*Фигура 1 Най-използваните уеб браузъри за 2015 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 (фигура 2). 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.



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.

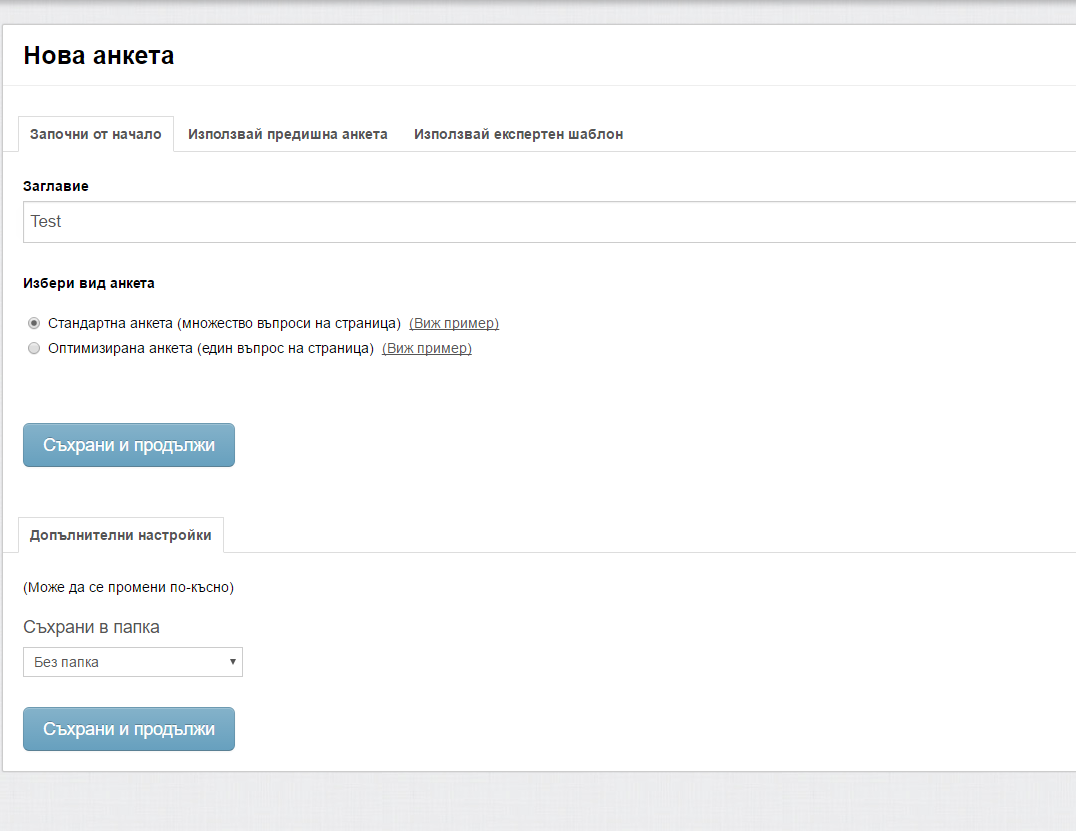
Социалните мрежи правят опити да убедят потребителите си да представят техните истински имена, интереси и местоположение. Това прави преживяването в социалните мрежи по-реалистично и така задържа повечето си потребители. От друга страна, източниците на някои снимки и необмислени твърдения могат да бъдат 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.**

## **Survey.bg**

Началната страница на приложението показва подробна информация относно анкетите – кои са активни, броят на отговорилите за последните 7 дни и информация за последните анкети. Survey.bg предлага по-богата функционалност при създаване на анкети (фигура 3) и въпроси към тях както и по-приятен потребителски интерфейс. Потребителят може да избира как да са визуализирани въпросите – всичките на една страница или по един въпрос на страница. Дава се възможност да се използва стара анкета, както и богат набор от експертни шаблони. [4]

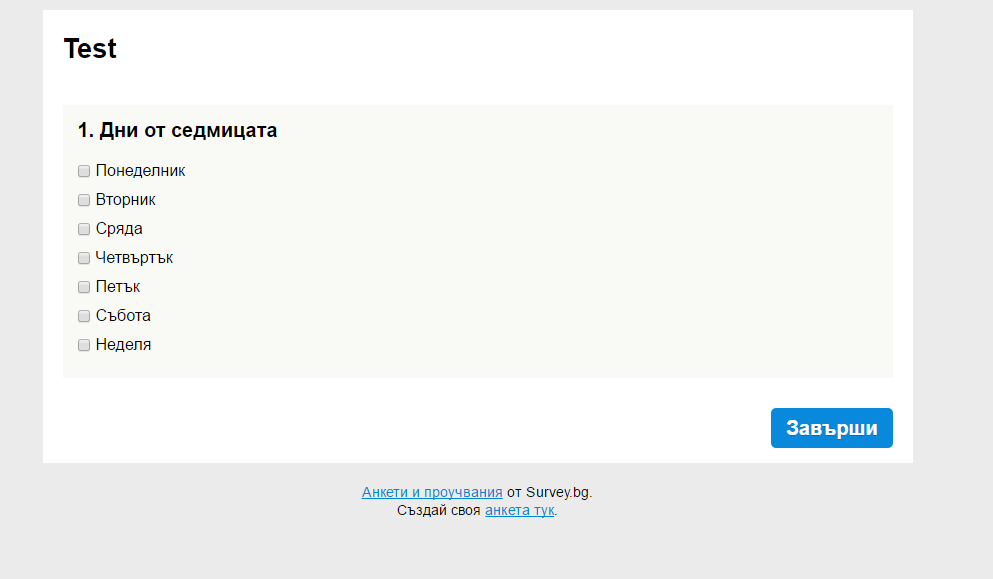
Фигура 3. Създаване на анкета в Surveybg.

Добавянето на въпроси и отговори (фигура 4) е улеснено до максимална степен. Видовете въпроси може да са: мултивариантни, матрици, полета, форми, графични, класификация и други, а този на отговорите се определя от вида на самият въпрос.



Фигура 4. Създаване на въпроси и отговори в Surveybg.

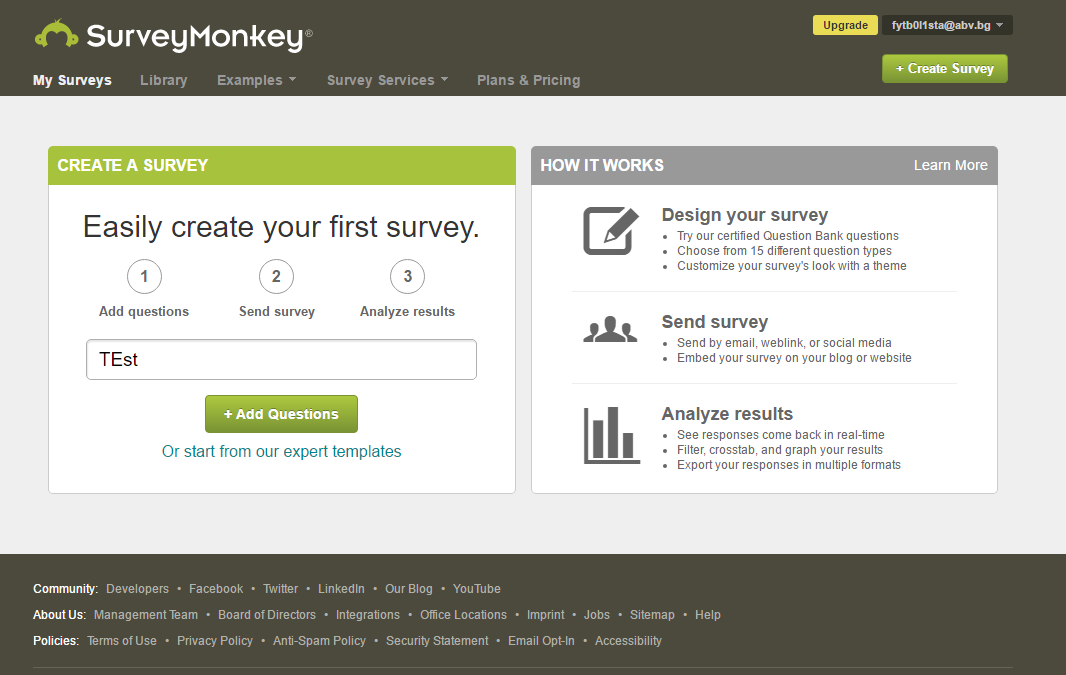
Анкета в Surveybg може да се попълни от всеки който намери линк за нея. Крайният резултат е показан на фигура 5.



Фигура 5. Завършена анкета в Surveybg.

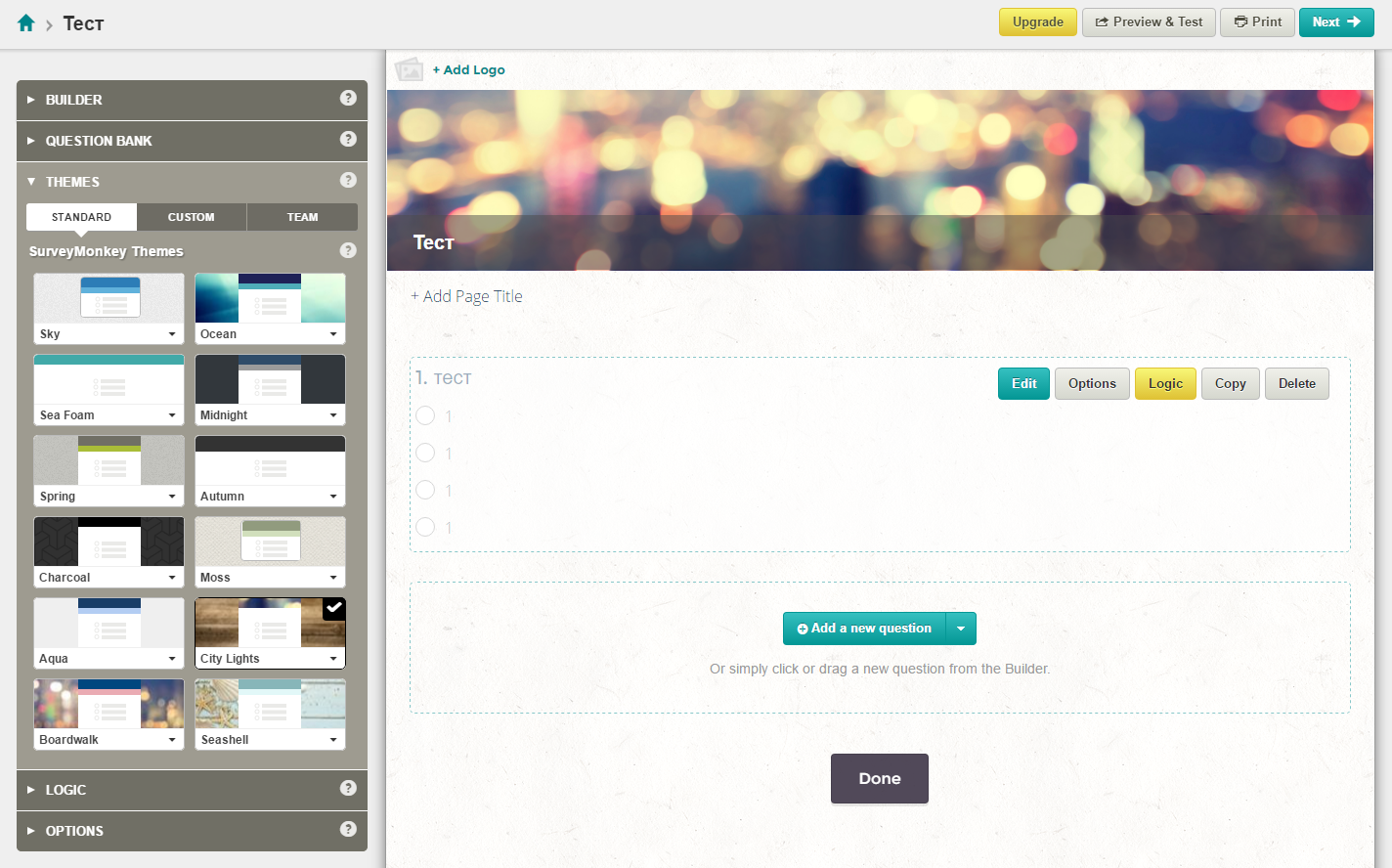
## **SurveyMonkey**

SurveyMonkey е най-големият и популярен сайт за анкетиране (фигура 6). Предлага изключително богата гама от функционалност, с която се управлява лесно и приятно анкетирането. Подобно на Surveybg предлага готови шаблони, с които създаването на анкетата става изключително бързо. При създаването на анкета се предоставя възможността да се избере категория. [5]



Фигура 6. Начална страница в SurveyMonkey.

Добавянето на въпроси и отговори е интуитивно. Освен разнообразния вид на въпросите, SurveyMonkey дава възможност да се избере и тема за фон на анкетата, която и придава изискан и елегантен вид. На фигура 7 са показани малка част от фоновите теми.



Фигура 7. Създаване на въпроси и отговори в SurveyMonkey.

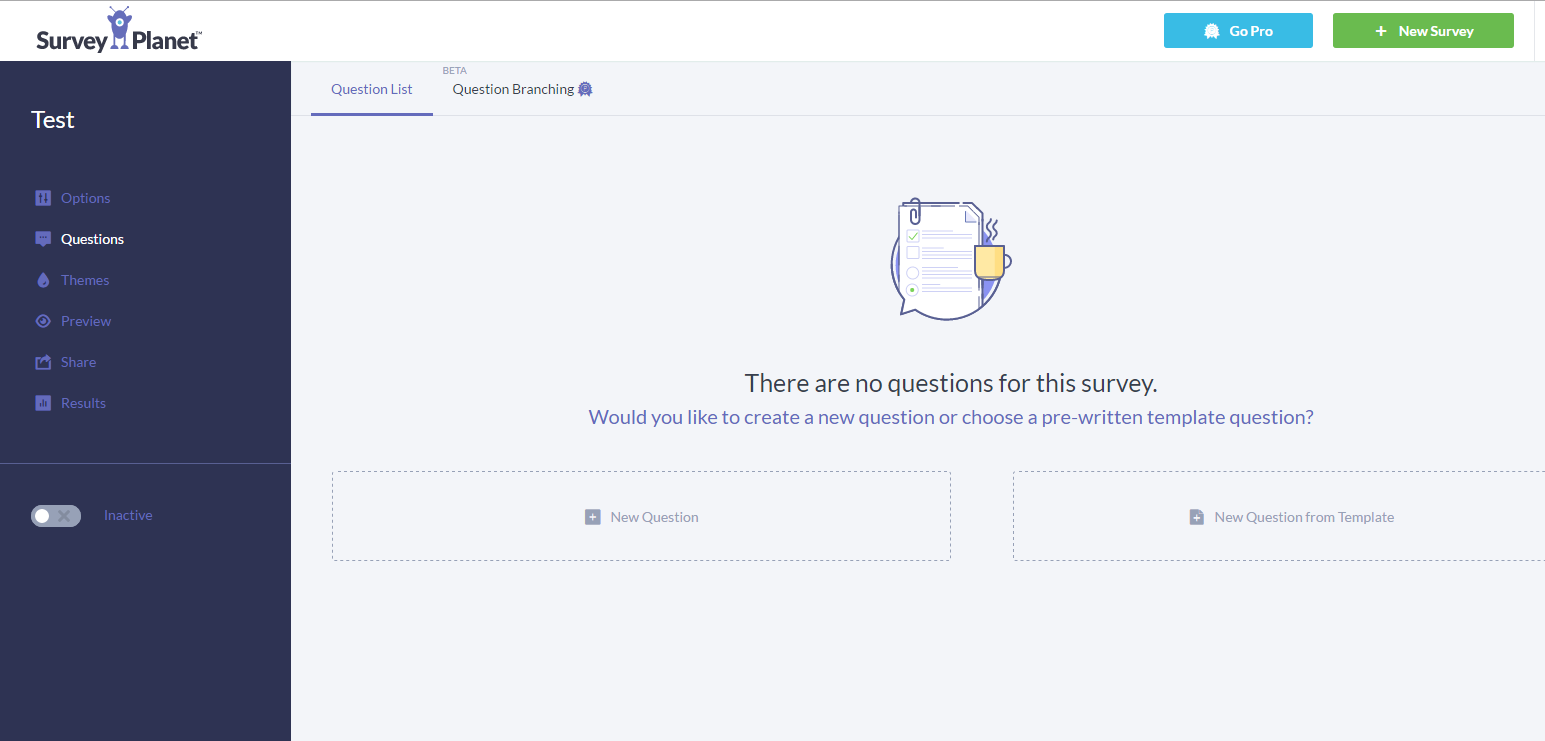
И тук е достатъчно да се знае само URL-а на анкетата за да се гласува. Завършената анкета е показана на фигура 8.



Фигура 8. Завършена анкета в SurveyMonkey.

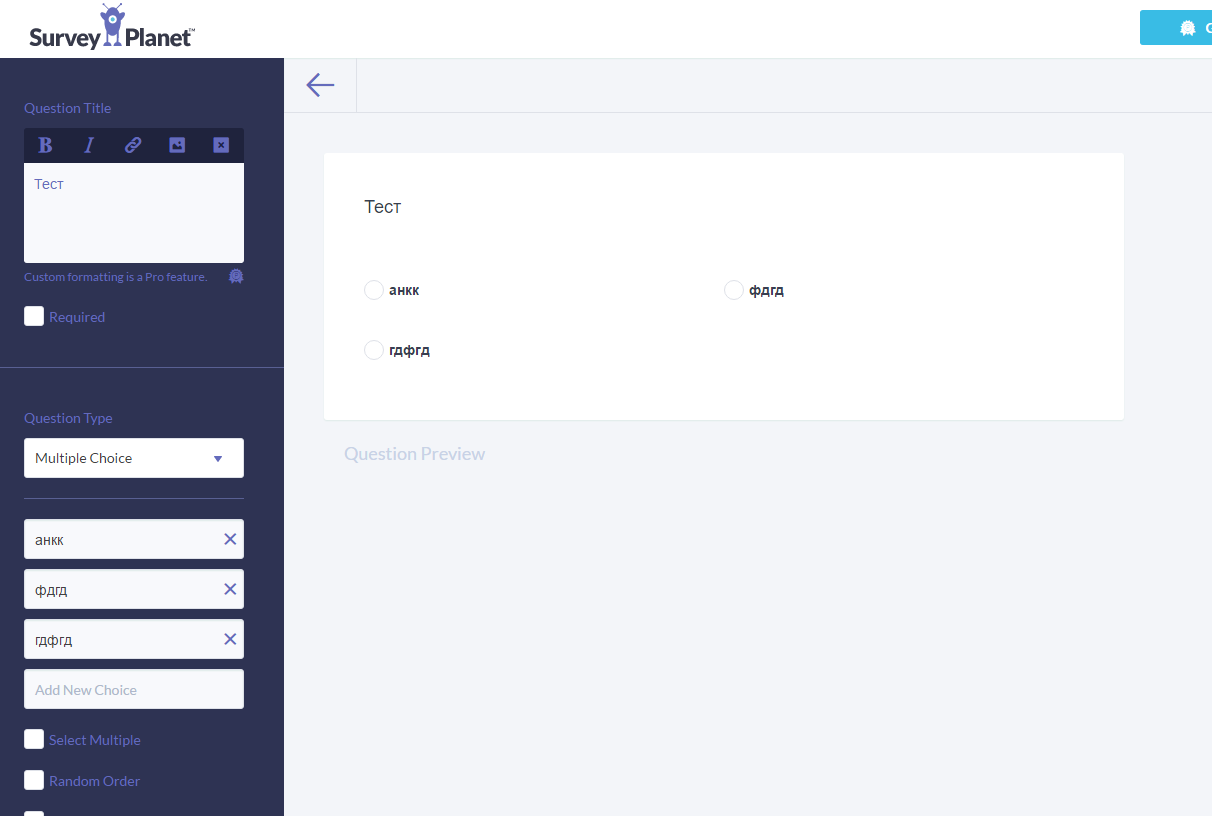
## **Surveyplanet**

Surveyplanet притежава интуитивен, опростен и изчистен интерфейс, показан на фигура 9, благодарение на който създаването на анкети е приятно и лесно. Подобно на Surveybg и SurveyMonkey типът на въпросите и отговорите е разнообразен. [6]



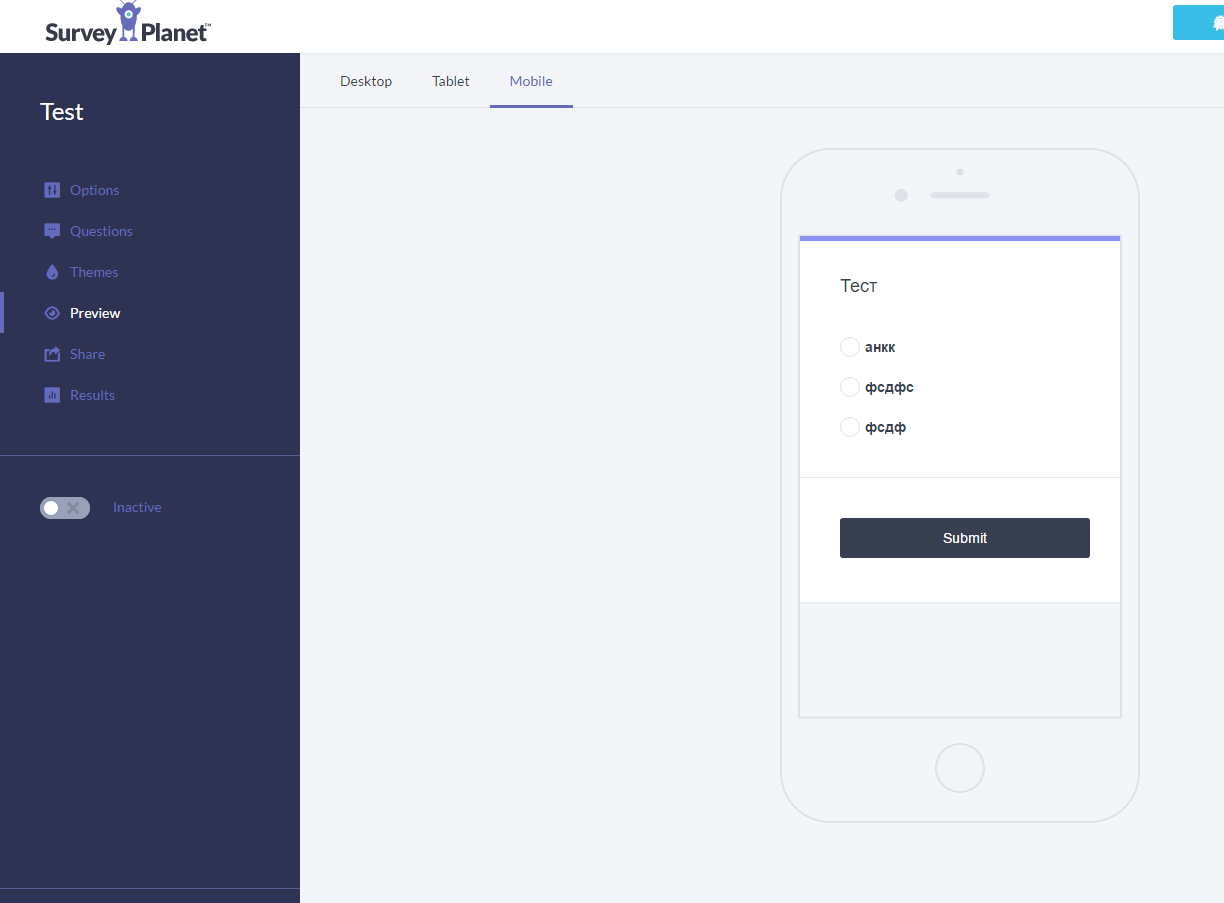
Фигура 9. Начална страница в Surveyplanet.

Важно да се отбележи, че за разлика от предходните приложения, тук въпросите и отговорите се преглеждат още при самото създаване. Отговорите могат да бъдат подредени в няколко колонки. На фигура 10 е показано как изглежда интерфейсът при добавяне на въпроси и отговори.



Фигура 10. Създаване на въпроси и отговори в Surveyplanet.

И тук се предоставя възможността да се избере фонова тема. Опцията за преглед ни позволява да погледнем как би изглеждала анкетата на различните устройства (фигура 11).



Фигура 11. Преглед на анкета в Surveyplanet.

И тук попълването може да стане само ако се знае URL-а, но за разлика от предходните уеб приложения, преди да се започне може да се задължи потребителят да въведе своят имейл адрес. Завършеният вид на анкетата е показан на фигура 12.



Фигура 12. Завършена анкета в Surveyplanet.

## **Цел и задачи.**

Целта на проекта е да се проектира и реализира уеб базирана информационна система за анкети и гласувания, 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 на 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. 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 система за гласуване.
* Да се предлага възможност за изтриване на система за гласуване.
* Да се предлага възможност за редактиране на система за гласуване.
* Да се предлага възможност за създаване на въпроси към определена система за гласуване.
* Да се предлага възможност за изтриване на въпроси към определена система за гласуване.
* Да се предлага възможност за редактиране на въпроси към определена система за гласуване.
* Да се предлага възможност за преглед на резултатите в дадена система за гласуване.

За потребителите:

* Да се предлага възможност за попълване отговори на въпроси към дадена система за гласуване.

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

### **4.2.1. Use-case диаграми.**

На фигури 13 и 14 са показани use-case диаграмите за двете групи потребители.



Фигура 13. Use-case диаграма на потребител с администраторски права.



Фигура 14. 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. На фигура 15 е по показана activity диаграма която показва процесът на логване, а на 16 процесът на попълване на дадена анкета.



Фигура 15. Activity диаграма която изобразява процеса на логване.



Фигура 16. Activity диаграма която изобразява процеса на попълване на дадена анкета.

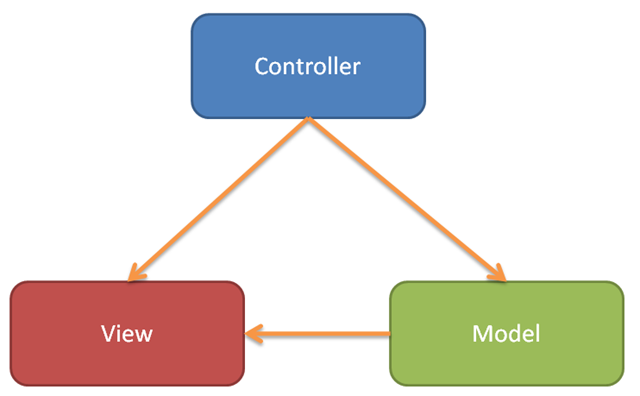
## **4.3. Архитектура на системата.**

Приложението е реализирано чрез архитектурния шаблон за дизайн Model-View-Controller (MVC) (фигура 17). 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.

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 HomeController се вземат данните за всички текущи анкети 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. [7]



Фигура 17. 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 е представено на фигура 18.



Фигура 18. Заявка на потребител към уеб приложение с MVC архитектура.

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

Уеб приложението ще обработва данни свързани със система за гласуване, а именно име на системата, име на въпрос, име на отговор. Организацията на данните, необходима за реализацията на приложението, е показана чрез концептуален модел на базата данни (фигура 19).



Фигура 19. 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 20).



Фигура 20. 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. |
| FirstName | nvarchar(40) | not null | Име на потребителя. |
| LastName | nvarchar(40) | not null | Фамилия на потребителя. |
| 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** |
| Id | int | not null,  **primary key** | Идентификационен номер на участникът. |
| User\_Id | nvarchar(128) | not null,  **foreign key** | Идентификационен номер на потребителя. |
| RateSystemId | int | not null,  **foreign key** | Идентификационен номер на системата за гласуване. |
| IsVoted | bit | not null | Гласувал ли е участникът. |

Таблица 2. Participants

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на участникът. |
| ParticipantId | nvarchar(MAX) | not null,  **foreign key** | Идентификационен номер на участникът. |
| QuestionAnswerId | int | not null,  **foreign key** | Идентификационен номер на отговорът. |
| CreatedOn | datetime | not null | Дата на създаване. |
| ModifiedOn | datetime | allow nulls | Дата на последната промяна. |

Таблица 3. ParticipantAnswers

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на системата за гласуване. |
| RateSystemName | nvarchar(100) | not null | Име на системата. |
| StartDateTime | datetime | not null | Начална дата. |
| EndDateTime | datetime | not null | Крайна дата. |
| CreatedOn | datetime | not null | Дата на създаване. |
| ModifiedOn | datetime | allow nulls | Дата на последната промяна. |
| IsDeleted | bit | not null | Изтрита ли е системата. |
| DeletedOn | datetime | allow nulls | Дата на изтриване. |

Таблица 4. RateSystems

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на въпроса. |
| QuestionName | nvarchar(40) | not null | Име на въпроса. |
| HasMultipleAnswers | bit | not null | Множествен отговор. |
| RateSystemId | int | not null | Идентификационен номер на системата. |
| CreatedOn | datetime | not null | Дата на създаване. |
| ModifiedOn | datetime | allow nulls | Дата на последната промяна. |
| IsDeleted | bit | not null | Изтрита ли е системата. |
| DeletedOn | datetime | allow nulls | Дата на изтриване. |

Таблица 5. Questions

|  |  |  |  |
| --- | --- | --- | --- |
| **Име на атрибут** | **Тип на данните** | **Ограничения** | **Предназначение** |
| Id | int | not null,  **primary key** | Идентификационен номер на отговора. |
| QuestionAnswerName | nvarchar(MAX) | not null | Наименование на отговора. |
| QuestionId | Int | not null,  **foreign key** | Идентификационен номер на въпроса. |
| CreatedOn | datetime | not null | Дата на създаване. |
| ModifiedOn | datetime | allow nulls | Дата на последната промяна. |

Таблица 6.QuestionAnswers

|  |  |  |  |
| --- | --- | --- | --- |
| **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 7. 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 8. 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 2015 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 10.
* Среда за програмиране: Visual Studio 2015 Community.
* Среда за работа със СУБД: Microsoft SQL Server Management Studio 2014 Professional

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

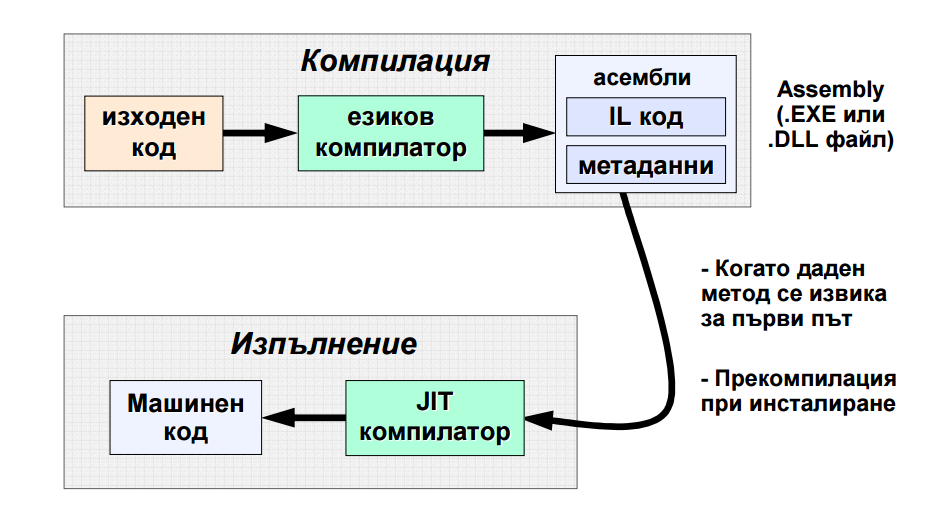
.NET Framework е съвременна платформа за разработка и изпълнение на приложения, която предоставя програмен модел, стандартна библиотека от класове(Framework Class Library) и среда за контролирано изпълнение на програмен код(Common Language Runtime). Тя поддържа различни езици за програмиране и позволява 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. Целият този процес е представен на фигура 21.



Фигура 21. 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. [8]

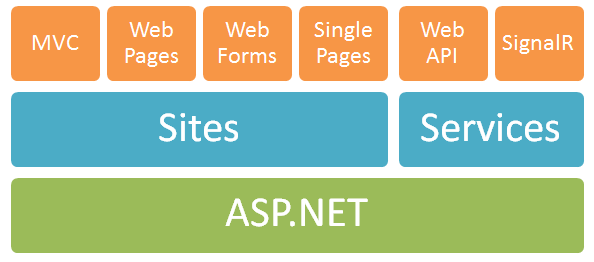
### **5.2.2. ASP.NET**

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. [9]

На фигура 22 са показани технологиите който към настояща дата са част от ASP.NET.



Фигура 22. 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.[10] Въпреки, че .NET Framework поддържа много други езици, за реализацията е избран езикът C# по няколко причини:

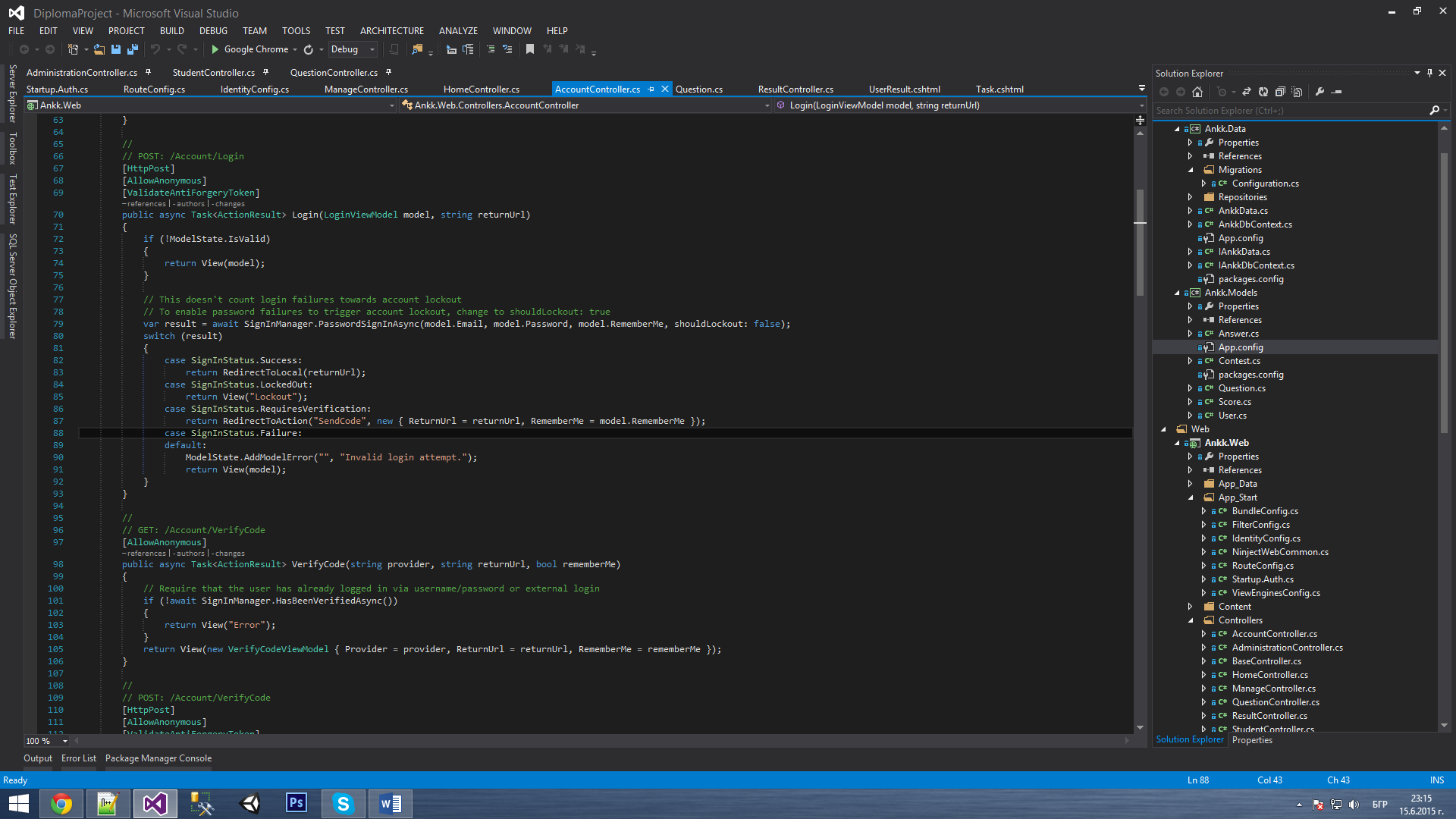
* Архитектите на езика са го проектирали специално за платформата .NET Framework, като още по време на дизайна са го съобразили с нейните особености. C# притежава максимално стегнат и ясен синтаксис, наследява простотата на Java, мощността на C++ и силните черти на Delphi.
* Езици като C++ и JavaScript 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 2015**

За реализацията на дипломния проект е използвано Microsoft Visual Studio (фигура 23). 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 23. 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: Участници

Предназначение: Имплементира таблица Participants.

Зависи от: БД.

Реализация: Participant.cs

Модел: Отговори на участниците

Предназначение: Имплементира таблица ParticipantAnswers.

Зависи от: БД.

Реализация: ParticipantAnswer.cs

Модел: Въпрос

Предназначение: Имплементира таблица Questions.

Зависи от: БД.

Реализация: Questions.cs

Модел: Отговор на въпрос

Предназначение: Имплементира таблица QuestionsAnswers.

Зависи от: БД.

Реализация: QuestionsAnswer.cs

Модел: Системи за гласуване

Предназначение: Имплементира таблица RateSystems.

Зависи от: БД.

Реализация: RateSystem.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: IVoteSystemDbContext.cs

Реализация: VoteSystemDbContext.cs

Под модул: Хранилище

1. Хранилище (основно)

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

Зависи от: БД.

Интерфейс: IDbGenericRepository.cs

Реализация: DbGenericRepository.cs

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

**public T GetById(object id) -** търси по идентификационен номер 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)** **–** маркира записа като изтрит.

1. Изтриваемо Хранилище

Предназначение: Позволява изтриването (окончателно) на записи от базата.

Зависи от: БД.

Интерфейс: IDeletableEntityRepository.cs

Реализация: DeletableEntityRepository.cs

**public IQueryable<T> AllWithDeleted() –** показва всички записи включително и тези маркирани като изтрити.

**public void HardDelete(T entity) -**  изтрива запис от базата.

### **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: Система за гласуване

Предназначение: Визуализира основните операции за създаване, изтриване и редактиране на системи за гласуване. 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 RateSystem.

Изглед: Въпрос

Предназначение: Използва се за създаване на въпроси и отговори. 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 в определена система за гласуване.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката Result.

Изглед: Попълване на система

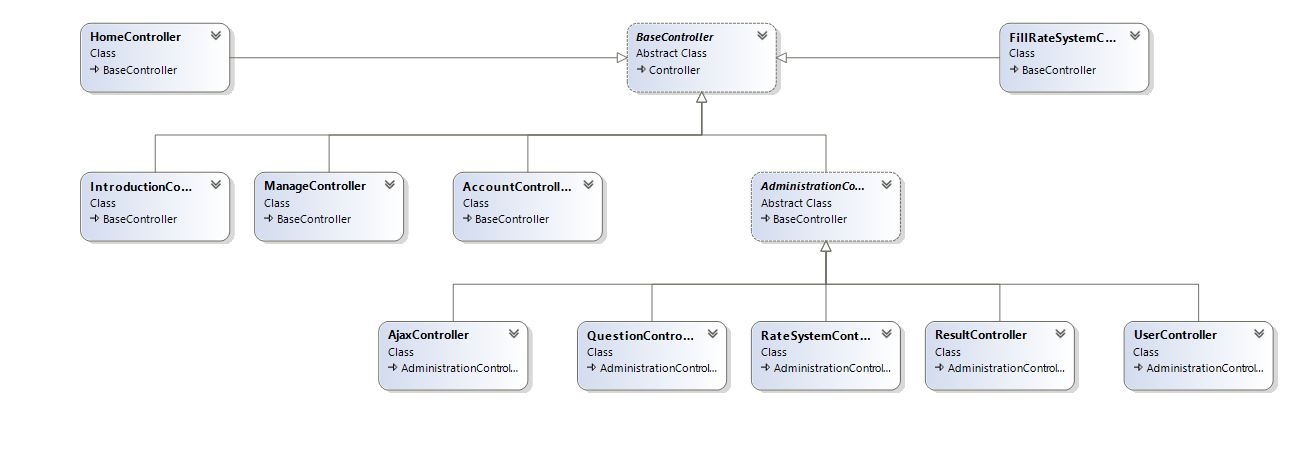
Предназначение: Дава възможност на обикновения потребител да участва в дадена система за гласуване.

Зависи от: Слоя на за управление на функционалността .

Реализация: Всички шаблони се намират в папката FillRateSystem.

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

На фигура 24 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 24. Клас диаграма на уеб приложението

Контролер: Администрация

Предназначение: Базов контролер за администрацията.

Зависи от: БД.

Реализация: AdministrationController.

Контролер: Динамичен

Предназначение: Извършва динамични операции свързани.

Зависи от: БД.

Реализация: AjaxController.

Контролер: Въпрос

Предназначение: Извършва операция за създаване на въпроси.

Зависи от: БД.

Реализация: QuestionController.

Контролер: Система за гласуване

Предназначение: Извършва основни CRUD операции върху система за гласуване.

Зависи от: БД.

Реализация: RateSystemController.

Контролер: Резултат

Предназначение: Извършва операциите чрез който се визуализират резултатите.

Зависи от: БД .

Реализация: ResultController.

Контролер: Потребител

Предназначение: Извършва операциите с който администраторът определя участниците в дадена система за гласуване.

Зависи от: БД .

Реализация: UserController.

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.

Контролер: Начален

Предназначение: Извършва основните операции свързани с визуализирането на началната страница след успешно логване.

Зависи от: БД.

Реализация: HomeController.

Контролер: Въведителен

Предназначение: Извършва основните операции свързани с визуализирането на началната страница на приложението.

Зависи от: БД.

Реализация: IntroductionController.

Контролер: Управителен

Предназначение: Извършва операция за управление на акаунтите.

Зависи от: БД.

Реализация: ManageController.

Контролер: Акаунт

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

Зависи от: БД.

Реализация: AccountController

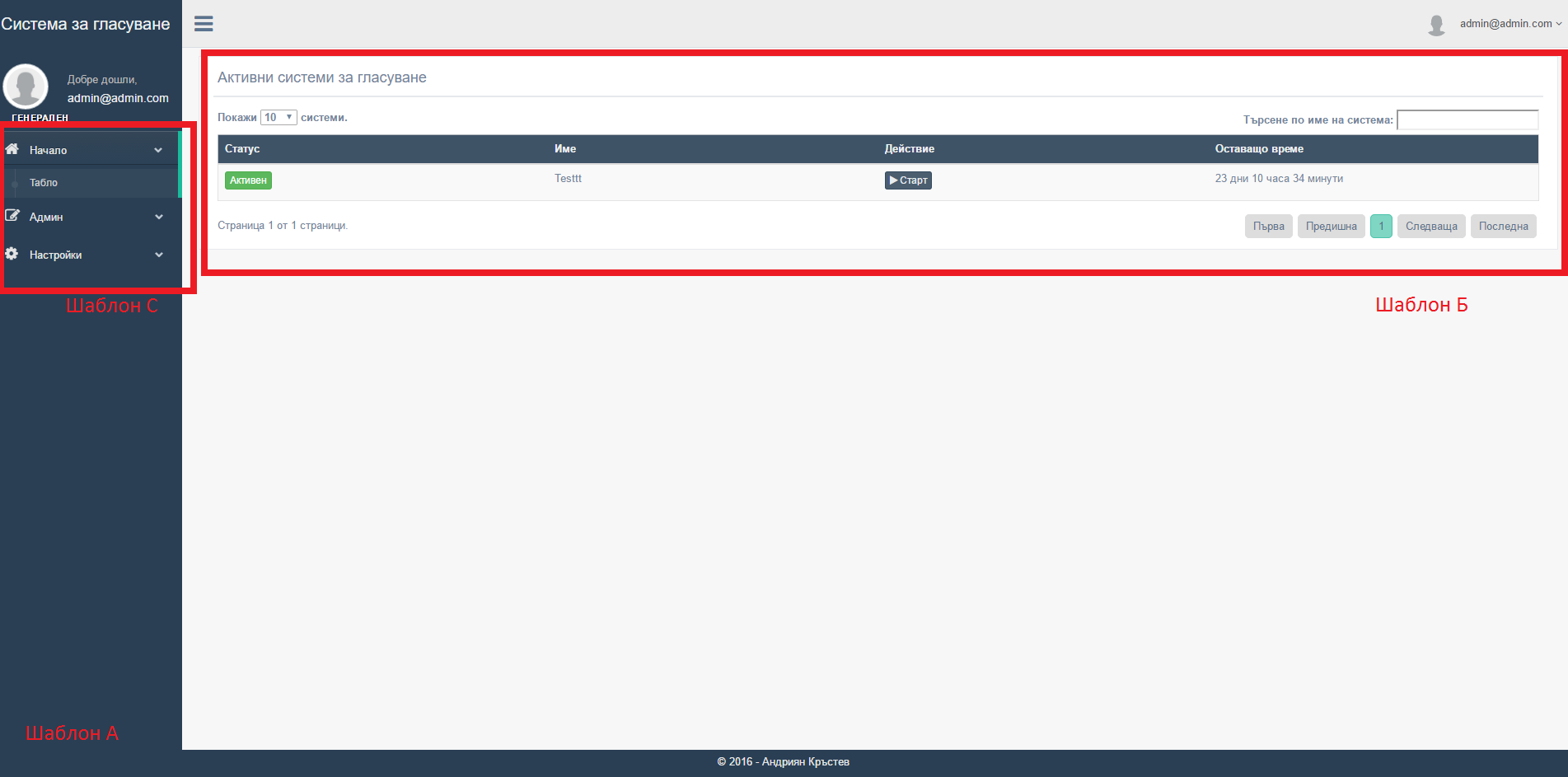
Контролер: Попълване на система

Предназначение: Извършва операциите с който обикновения потребител има възможността да попълва определена система за гласуване.

Зависи от: БД .

Реализация: FillRateSystemController.

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



Фигура 25. 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 коя анкета да се присъедини. Ако потребителя има администраторски права то тогава във шаблон C, ще се появи още една възможност (Админ) чрез която ще има право да създава нови анкети. Различните видове шаблони са показани на фигура 25.

## **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 валидността и.

**Име:** Символи между 2 и 20. При въвеждане се извършва проверка за валидността на името.

**Фамилия:** Символи между 2 и 20. При въвеждане се извършва проверка за валидността на фамилията.

Създаване на анкета:

**Име:** Символи между 5 и 100. При въвеждане се извършва проверка за неговата валидност.

**Начало:** Проверява се дали е въведена коректно датата.

**Край:** Проверява се дали е въведена коректно датата.

Създаване на въпрос:

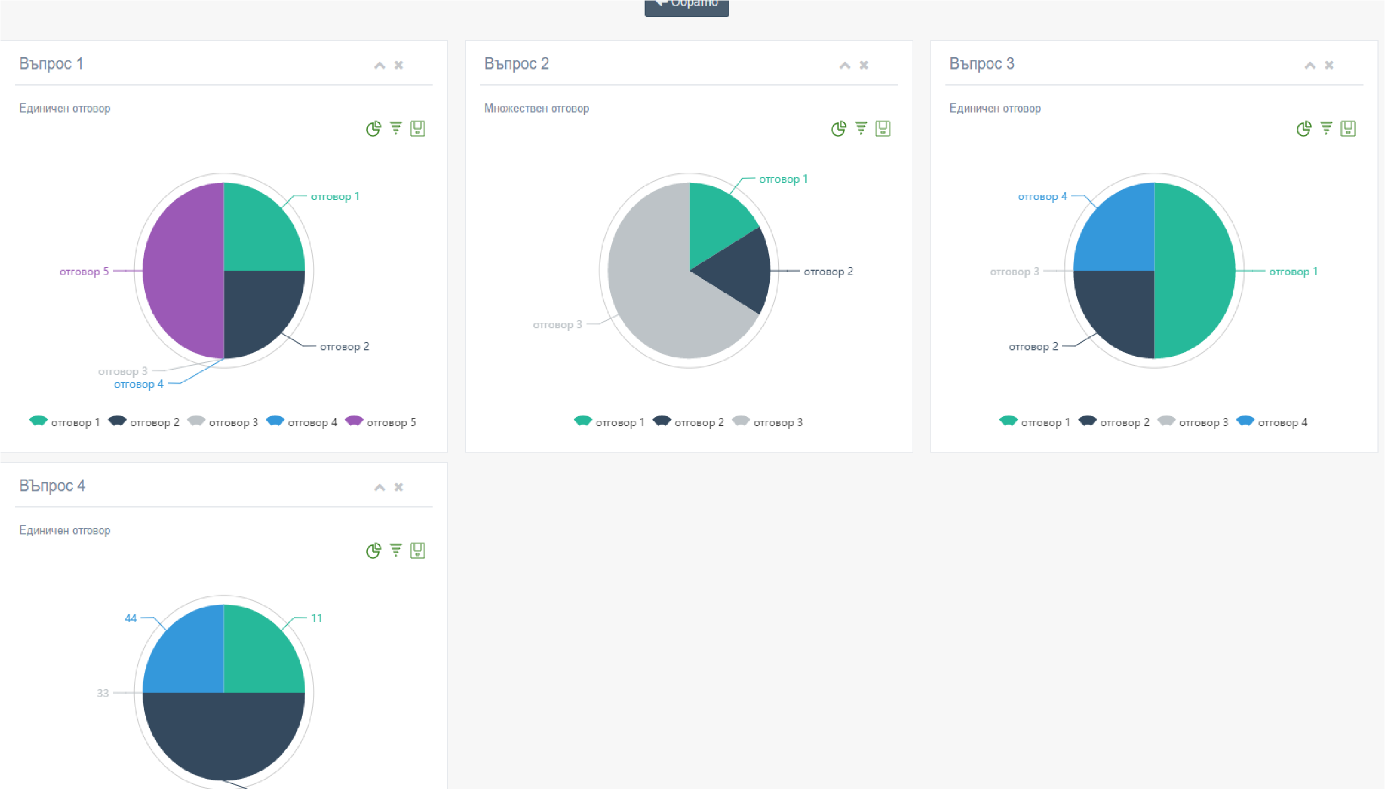
**Име:** Символи между 2 и 200. При въвеждане се извършва проверка за неговата валидност.

Създаване на отговор:

**Име:** Символи между 2 и 200. При въвеждане се извършва проверка за неговата валидност.

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

Резултатите от определена анкета се визуализират в приятен за потребителя вид, във вид на диаграми които са представени на фигура 26.



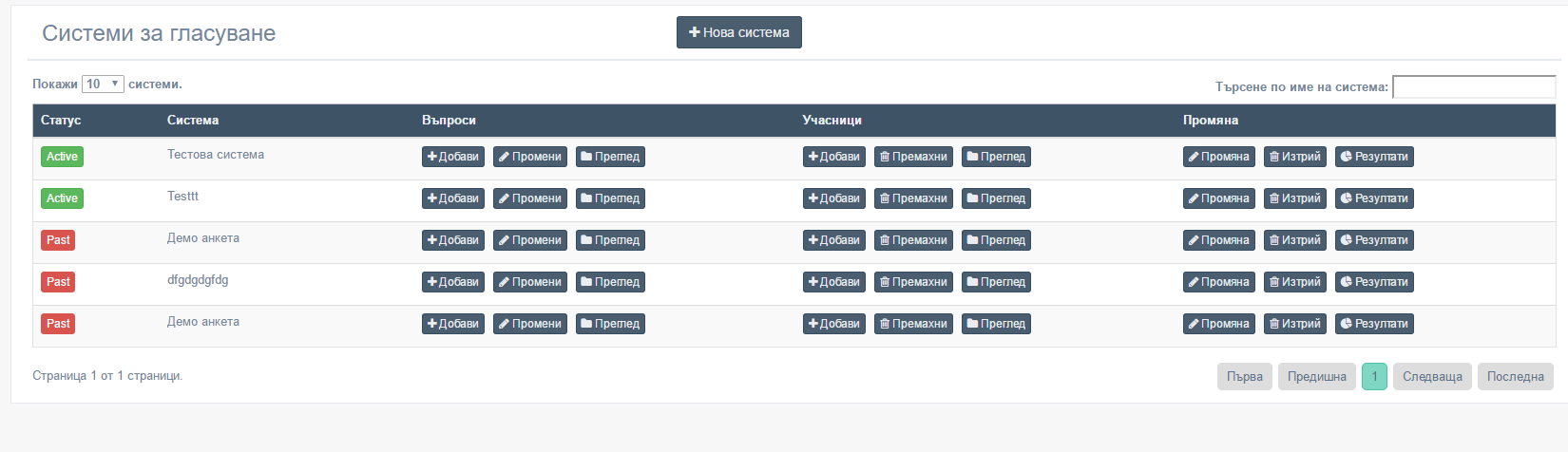
Фигура 26. Преглед на резултат.

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

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

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

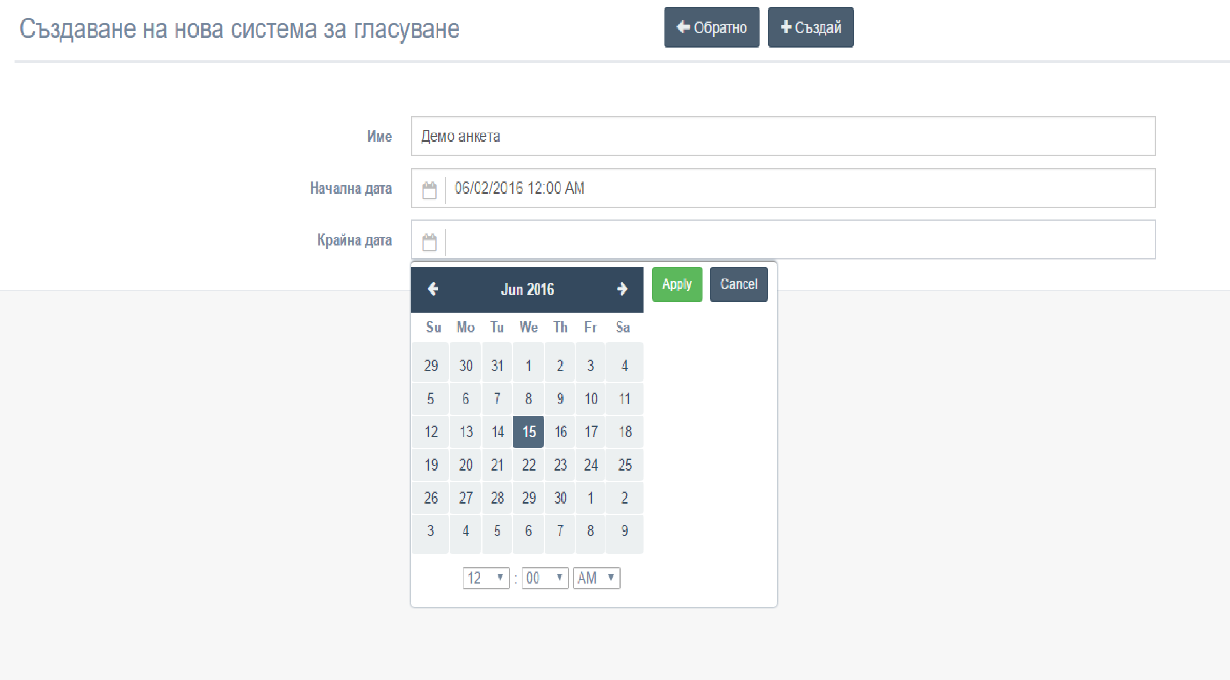
На фигура 27 е показана началната страница на администратор след успешно логване в информационната система.



Фигура 27. Администраторски панел

#### **6.1.1.1. Създаване на нова система**

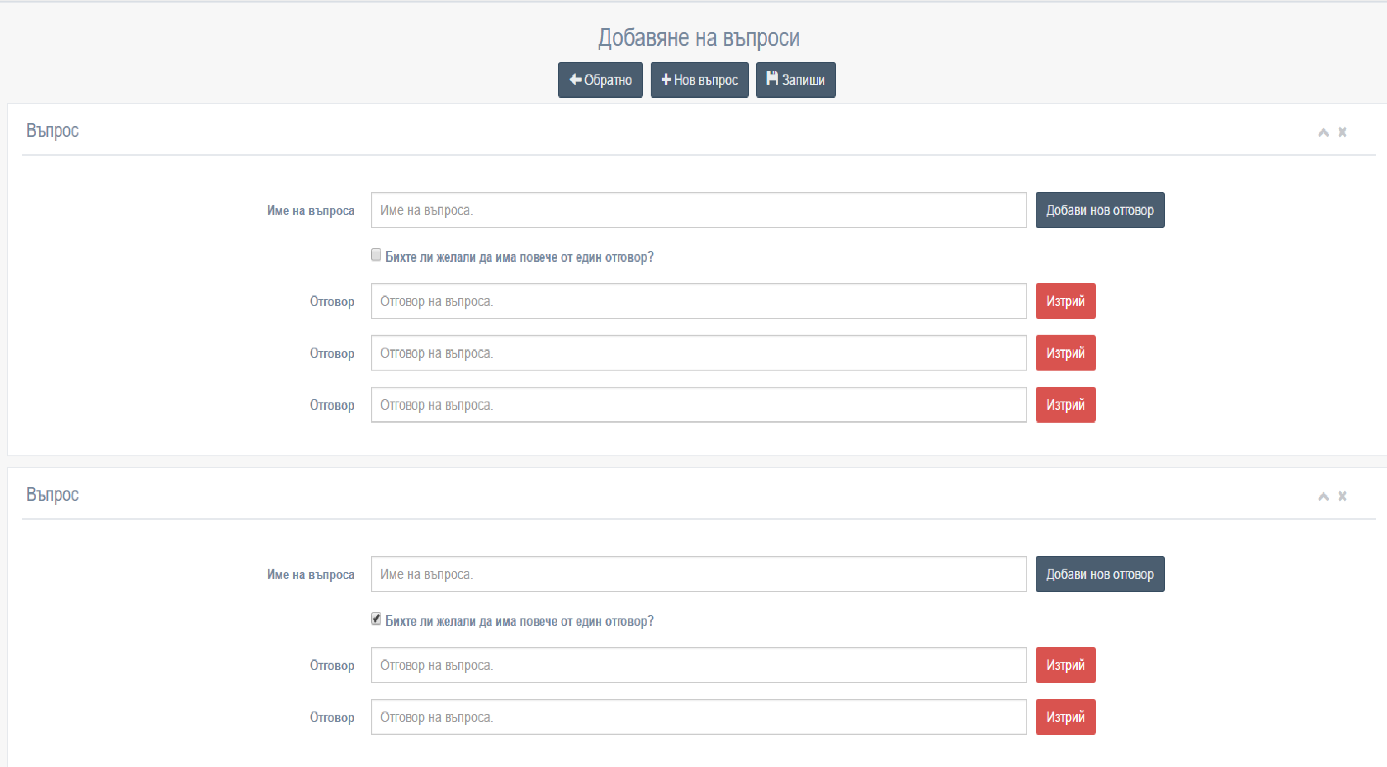
След натискане на бутона „Новa система“ (намира се в Админ/Табло) ще се появи възможност за създаване на нова анкета. В новия прозорец се дава име на анкетата, начало и край (фигура 28).



Фигура 28. Създаване на анкета

#### **6.1.1.2. Създаване на въпроси и отговори към анкета.**

След създаването на анкетата, уеб приложението препраща администратора към началната страница. На нея се визуализира името анкетата. От колонката Въпроси се натиска бутонът „Добави“. В новия прозорец се дава възможност за създаване на въпроси и отговори към тях (фигура 29).

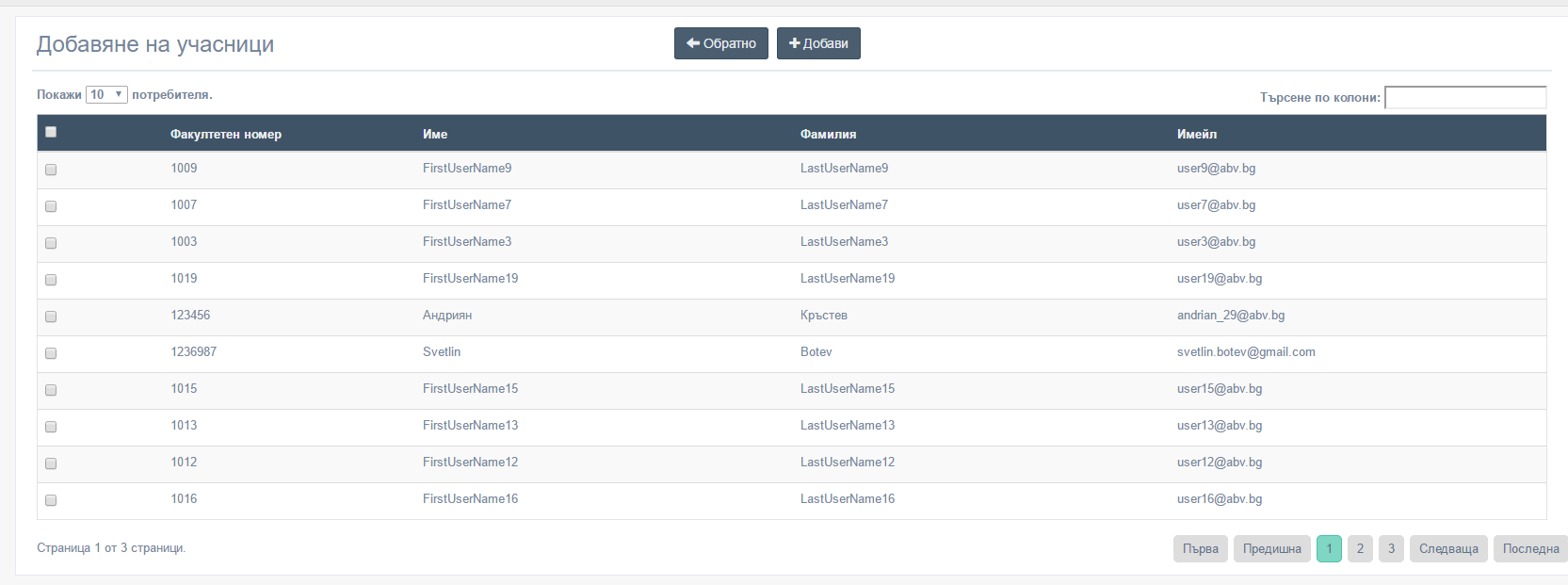


Фигура 29. Създаване на въпроси и отговори.

Броят на въпросите и както и на отговорите е неограничен. Създаването на нов въпрос става чрез натискане на бутонът „Нов въпрос“. За да се добави отговор на въпрос се натиска „Добави нов отговор“ срещу името на въпросът. След въвеждане на желаният брой въпроси и отговори се натиска „Запиши“ след което приложението автоматично записва въведената информация от администраторът. От администраторският панел се предоставя възможност за промяна на вече добавени въпроси и отговори както и техният преглед.

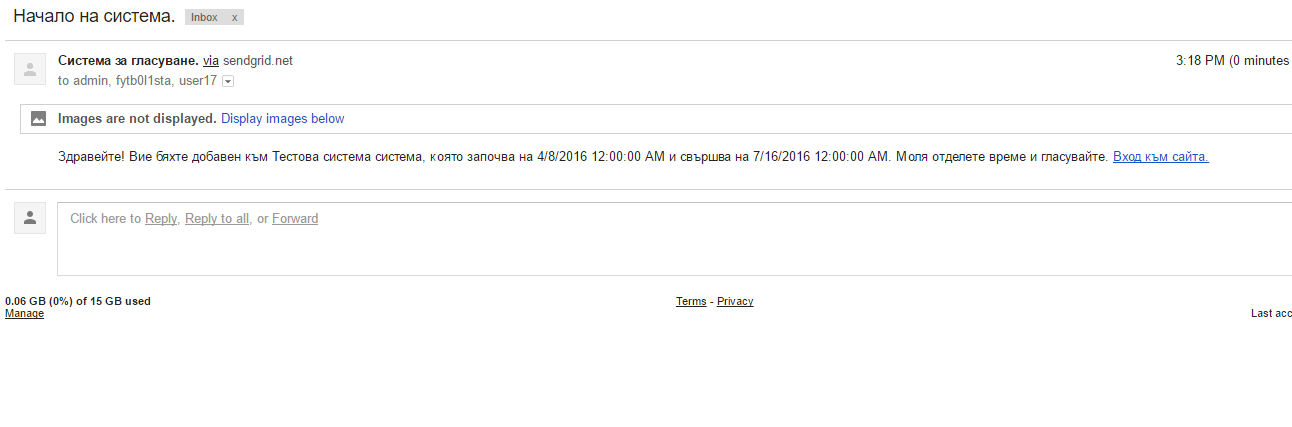
***6.1.1.3. Добавяне на участници към анкета.***

В администраторският панел от колонката Участници се натиска бутонът „Добави“ след което приложението препраща администраторът към таблица която визуализира всички регистрирани потребители в приложението. Таблицата е показана на фигура 30.



Фигура 30. Добавяне на участници.

Предоставя се възможност за филтриране по всяка една колона с цел максимално улесняване когато се търси определен потребител. Добавянето на потребител/и става след като бъде селектиран (най-лявата част на таблицата) и натискане на бутонът „Добави“. От администраторският панел се предоставя възможност за премахване и преглед на участниците към анкетата. Премахването на участници става по аналогичен начин както добавянето. В преглед прозорецът, администраторът има възможност да уведоми добавените участниците за началото на анкетата чрез натискане на бутонът „Изпрати имейл“. Приложението автоматично изпраща информация относно анкетата на всеки един от участниците на техните имейл адреси. Съдържанието на имейлът е показано на фигура 31.

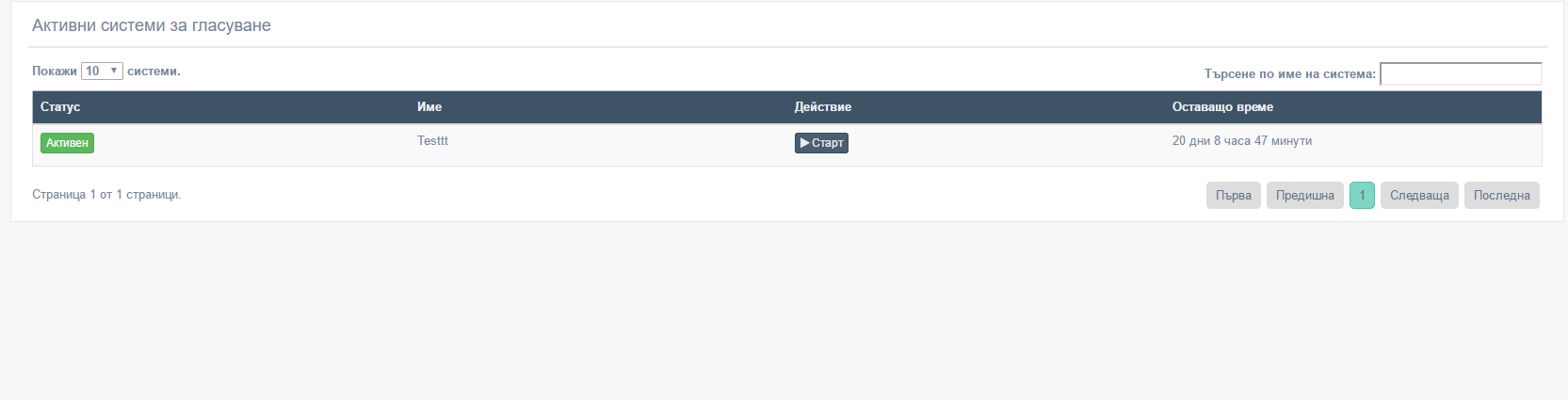
 Фигура 31. Имейл съобщение за начало на анкета.

#### **6.1.1.4. Обработка на анкета.**

В администраторският панел от колонката „Промяна“ се предоставя възможността за промяна, изтриване и преглед на резултатите за съществуваща анкета.

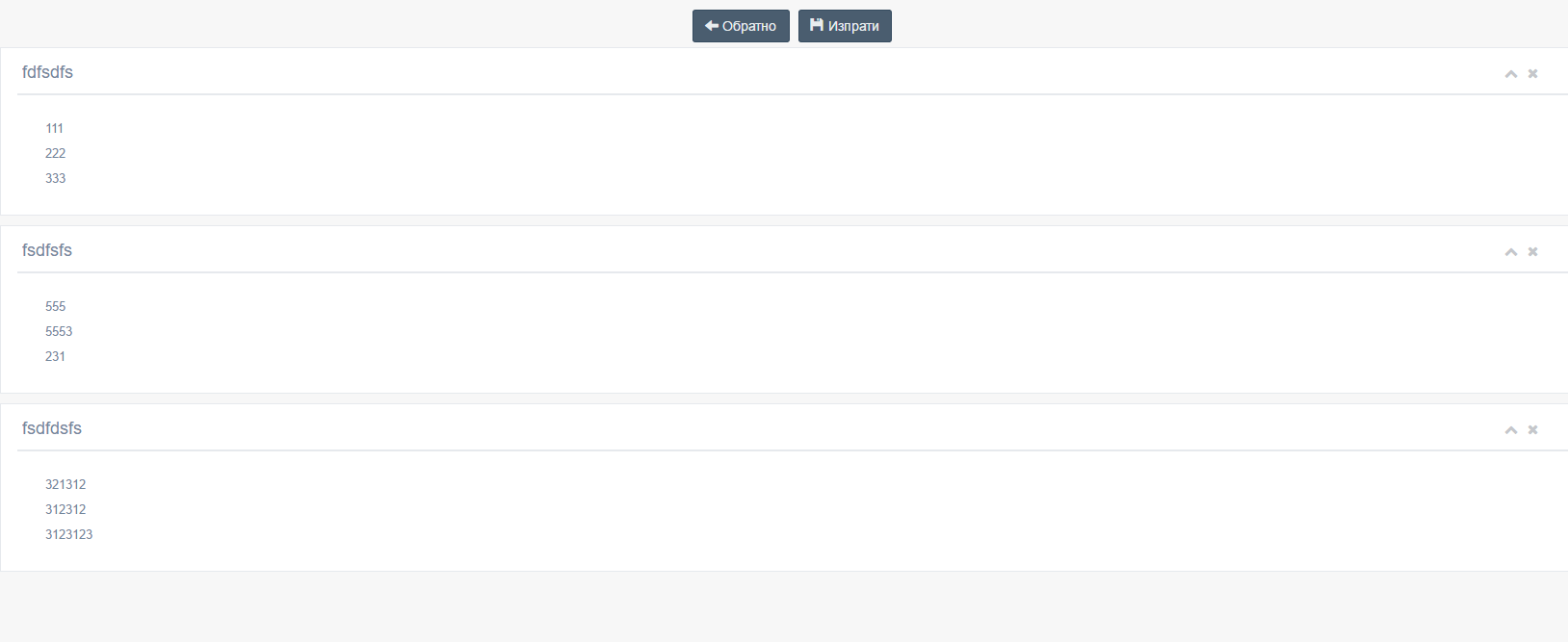
### **6.1.2. Ръководство за обикновен потребител.**

На фигура 32 е показана началната страница на обикновен потребител след успешно логване в информационната система.



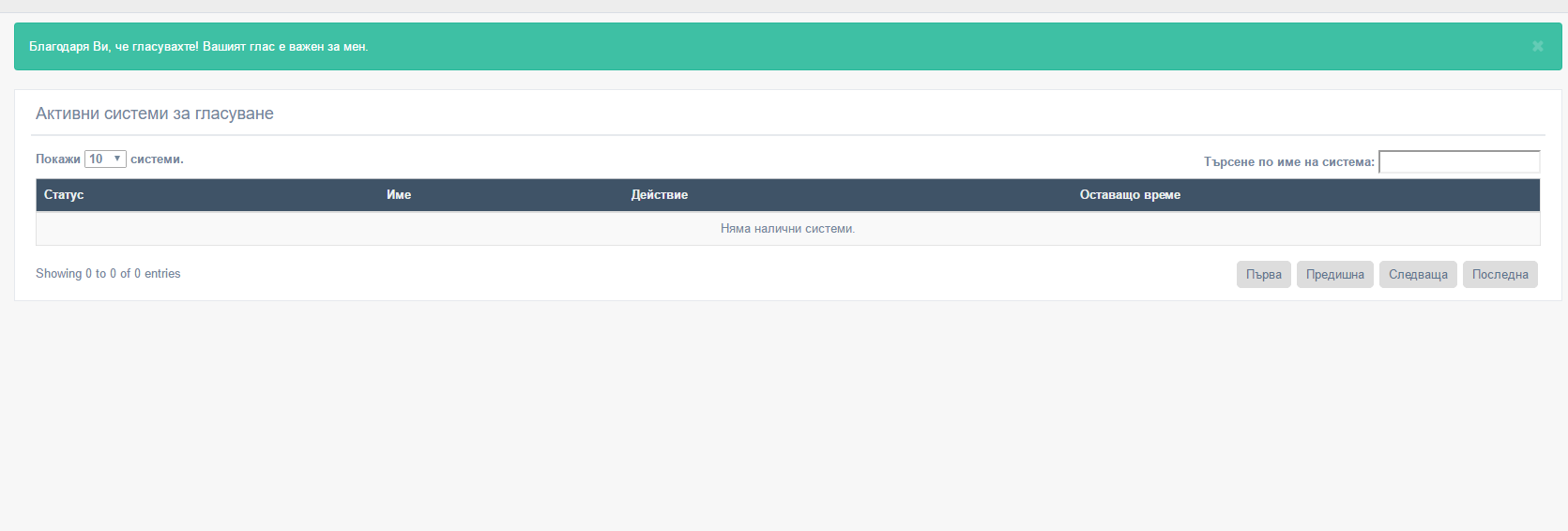
Фигура 32. Активни анкети.

В началната страница са визуализирани всички активни състезания към който е добавен текущият потребител. Той има права да гласува само един път за всяка анкета. Попълването започва след натискането на бутонът „Старт“. След което се визуализира прозорец с въпросите и отговорите към съответната анкета (фигура 33).



Фигура 33. Въпроси и отговори.

След като се изберат подходящите отговори, потребителят трябва да ги изпрати чрез натискане на бутонът „Изпрати“. Ако е отговорил на всички въпроси приложението го препраща на началната страница и съобщава, че гласуването е успешно (фигура 34).



Фигура 34. Успешно попълнена анкета.

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

За инсталиране на приложението са нужни следните неща:

* Visual Studio 2012+
* .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 както и с Android и iOS устройства.

# **Резултати от тестването на системата.**

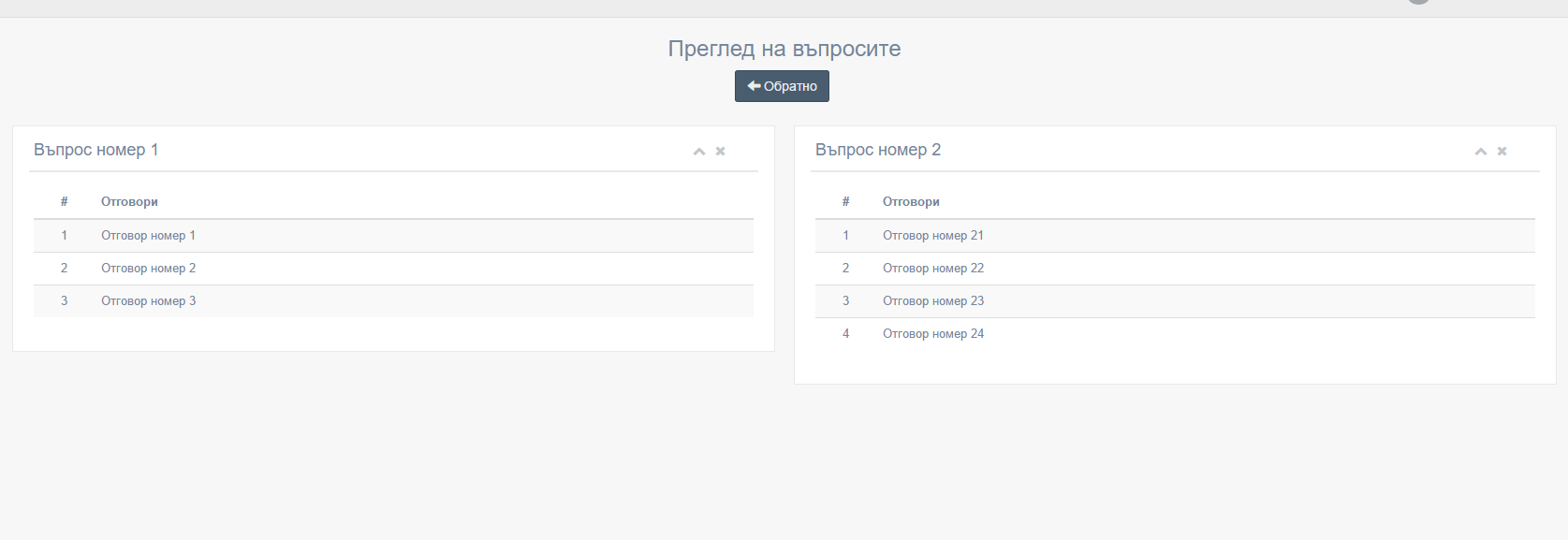
За тестване на приложението е необходимо да бъде създадена анкета и няколко въпроси към нея от потребител с администраторски права с цел тестване на цялата функционалност.

Нека нашето състезание се казва „Тестова анкета“, която да започва на 06/18/2016 и да свършва на 07/09/2016 (фигура 35).

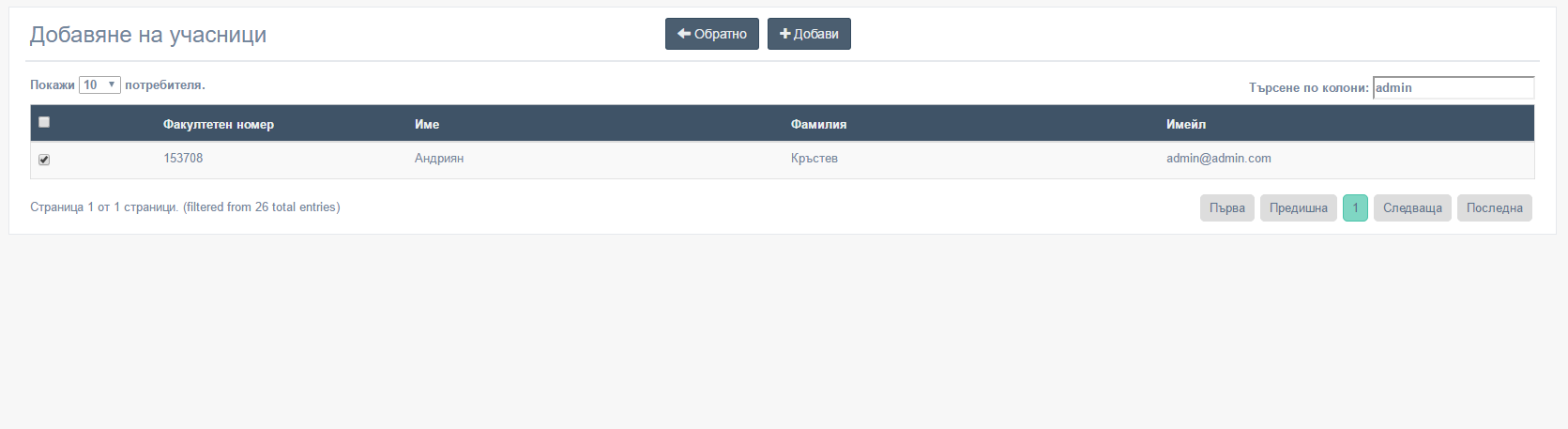


Фигура 35. Начален панел.

Към нея да добавим два въпроса. Нека единият да е с единичен отговор, а другият с множествен (фигура 36).

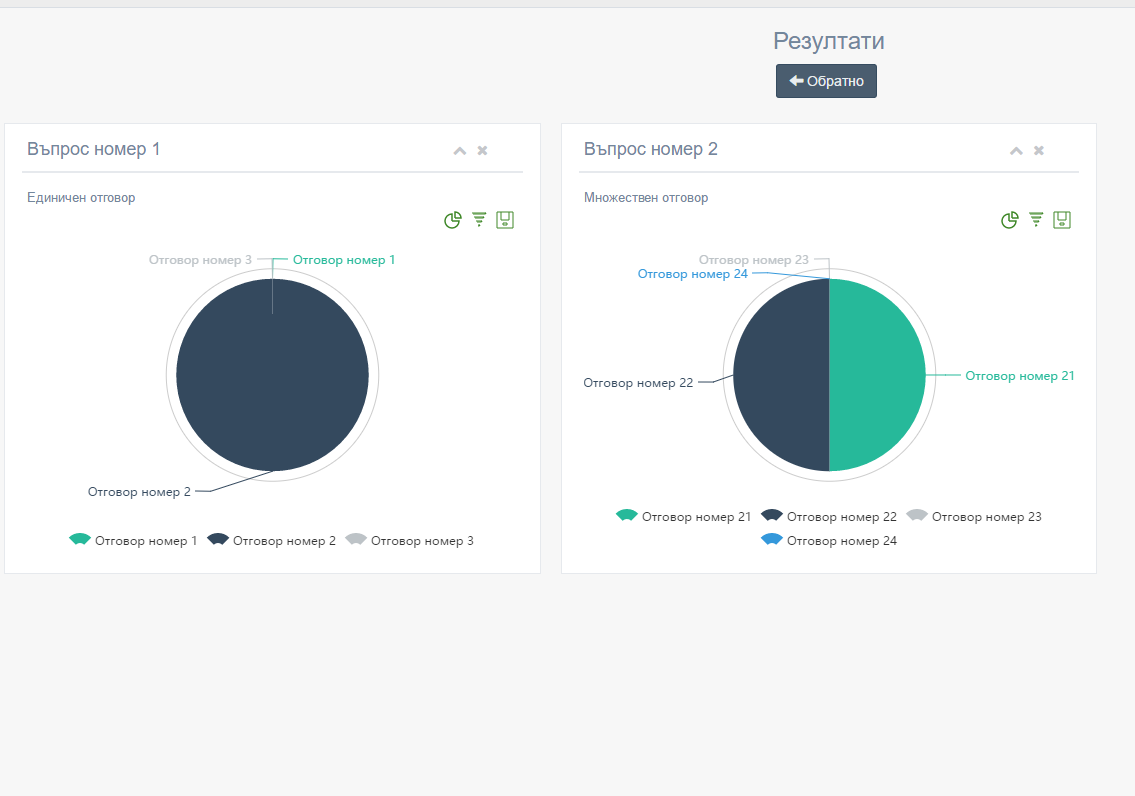
 Фигура 36. Преглед на въпроси.

След което добавяме и участник (фигура 37).



Фигура 37. Добавяне на участник.

Резултатът от попълването на анкетата е показан на фигура 38.



Фигура 38. Резултати от анкетирането.

На фигура 38 се вижда, че „Въпрос номер 2“ има два посочени отговора понеже е от тип множествен избор.

**8. Основни резултати.**

Проектирано и реализирано е уеб базирана информационна система за анкети и гласувания. Тя позволява на различни функционалности на различните групи потребители. За администраторите: създаване на анкети, въпроси и отговори както и добавяне на участници, а обикновените потребители имат възможността да гласуват в определена анкета. Обикновеният потребител може да гласува еднократно, като зависимост от типа на въпроса може да избере един или повече отговори. Администраторите от своя страна могат: да променят анкетата; да преглеждат резултатите във всяка една от тях.

# **9. Изводи и препоръки.**

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

* Добавяне на функционалност за преглед на вече изтрити системи за гласуване.
* Добавяне на функционалност за добавяне на профилна снимка.
* Подобряване на потребителският интерфейс за различните мобилни устройства.
* Кеширане на данните при преглед на резултати с цел малко заявки към сървърът.

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

[4]<https://survey.bg/>

[5]<https://www.surveymonkey.com/>

[6] <https://surveyplanet.com>

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

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

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

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