**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

**ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ**

**Катедра “** **Компютърни системи”**

**Дипломна работа**

**Тема:** **Софтуерна система за разпределяне и съставяне на дипломни задания**

**Изготвил: Стефан Николов Петров**

**Факултет: КСТ**

**Фак. №: 121314032**

**Проверил:** **гл. ас. д-р Антония Ташева**

* заглавна страница;
* утвърдено дипломно задание;
* съдържание;
* увод;
* изложение на дипломната работа по глави;
* заключение;
* списък на използвани литературни източници;
* приложения.

Съдържание

[Увод 3](#_Toc453710894)

[1 Анализ на подобни системи. Цели и задачи. 4](#_Toc453710895)

[2 Обзор на използваните технологии. 5](#_Toc453710896)

[2.1 ASP.NET MVC 5](#_Toc453710897)

[2.1.1 ASP.NET MVC 6](#_Toc453710898)

[2.1.2 Изглед (View) 6](#_Toc453710899)

[2.1.3 Модел (Model) 7](#_Toc453710900)

[2.1.4 Контролер 7](#_Toc453710901)

[2.2 C# 8](#_Toc453710902)

[2.3 MS SQL - база данни 9](#_Toc453710903)

[2.3.1 Релационна база данни 9](#_Toc453710904)

[2.3.2 Отношение 9](#_Toc453710905)

[2.3.3 Нормализация на базата данни 10](#_Toc453710906)

[2.4 jQuery 11](#_Toc453710907)

[2.5 CSS 12](#_Toc453710908)

[3 Проектиране на системата. 13](#_Toc453710909)

[3.1 Съставяне на моделите за базата данни 14](#_Toc453710910)

[3.2 Съставяне на хранилище за елементите на базата (repository) 15](#_Toc453710911)

[3.3 Създаване на слой с услуги (Service layer pattern) 16](#_Toc453710912)

[3.4 Създаване на подходящи контролери и изгледи 17](#_Toc453710913)

[3.4.1 Регистрация на потребители 17](#_Toc453710914)

[3.4.2 Избор на роли в приложението 18](#_Toc453710915)

[3.4.3 Създаване и редактиране на дипломи от преподавателя 18](#_Toc453710916)

[3.4.4 Редактиране на дипломи от студенти 18](#_Toc453710917)

[4 Програмна реализация. 19](#_Toc453710918)

[5 Ръководство на потребителя 19](#_Toc453710919)

[6 Заключение 19](#_Toc453710920)

[7 Списък на използвани литературни източници 19](#_Toc453710921)

[8 Приложения 19](#_Toc453710922)

# Увод

Уводът съдържа резюме на дейностите извършени от дипломанта в процеса на дипломното проектиране.

Управление на дипломните задания (менажиране)

Лесен избор на дипломна работа

Намиране на подходящ ръководител или различни преподаватели за консултации

Повечето избори стават от сайт със статични теми като този

https://www.fmi.uni-sofia.bg/lecturers/softeng/milenp/42243543c438-437430-43443843f43b43e43c43d438-44043043143e442438

# Анализ на подобни системи. Цели и задачи.

# Обзор на използваните технологии.

## ASP.NET MVC



**ASP.NET** е новото поколение уеб рамка (framework), разработена от Microsoft. 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, но да не се бърка, че не е подобрена версия на ASP. ASP.NET е изградена въз основа на Common Language Runtime (CLR), което позволява на програмистите да пишат ASP.NET код като използват .NET език по избор. 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.

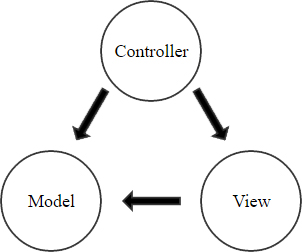
ASP.NET цели производителност спрямо останалите скрипт-базирани технологии (включително класическия ASP) като компилира сървърно кода в един или повече DLL файлове на Уеб сървъра. Тази компилация става автоматично когато страницата бива заредена за пръв път (което от своя страна означава, че програмистът не трябва да изпълнява отделни компилации за страниците). 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.

ASP.NET предлага три рамки за създаване на уеб приложения: Web Forms, ASP.NET MVC, и ASP.NET Web Pages. И трите рамки са стабилни и добре развити, и с всяка от тях могат да се създадат големи уеб приложения. Всички рамки предоставят навсякъде предимствата и характеристиките на ASP.NET.

### ASP.NET MVC



**ASP.NET MVC** е платформа, създадена от Microsoft, която служи за изработване на уеб приложения, използвайки модела **Model-View-Controller** (MVC). Платформата използва C#, HTML, CSS, JavaScript и бази данни. ASP.NET MVC е съвременно средство за изграждане на уеб приложения, което не замества изцяло уеб формите. Платформата включва нови тенденции в разработката на уеб приложения, притежава много добър контрол върху HTML и дава възможност за създаване на всякакви приложения. ASP.NET MVC може да бъде много лесно тествана и допълвана, защото е изградена от отделни модули, които са изцяло независими едни от други. Чрез платформата се създават цялостни приложения, които се стартират, а не единични скриптове (като при PHP например).



Фигура 2.1 Схема на Модел-Изглед-Контролер

### Изглед (View)

Изгледите са тези, които определят как ще бъде визуализиран потребителският интерфейс (UI) на приложението. В ASP.NET MVC се поддържат средства (engines) за генериране на изгледи.

Когато потребител взаимодейства с изглед, данните се предават от изгледа до метод за действие, който от своя страна може да създаде друг изглед. Едно MVC приложение може да има няколко контролери, всеки от които може да съдържа множество методи за действие, а всяко действие може да създаде различен изглед. Изгледите са организирани в папки, като името им се определя от това на свързания контролер.

### Модел (Model)

Моделът представлява част от приложението, което реализира домейн логиката, също известна като бизнес логика. Домейн логиката обработва данните, които се предават между базата данни и потребителския интерфейс. Например, в една система за инвентаризация, моделът отговаря за това дали елемент от склада е наличен. Моделът може да бъде част от заявлението, което актуализира базата данни когато даден елемент е продаден или доставен в склада. Често моделът съхранява и извлича официална информация в базата данни.

Изглед модел (ViewModel)

Изглед модела позволява да се оформят няколко изгледа от един или повече модела от данни или източници в един обект. Този модел е оптимизиран за потребление и изпълнение.

### Контролер (Controller)

Контролери са класове, които се създават в MVC приложението. Намират се в папка Controllers. Всеки един клас, който е от този тип, трябва да има име завършващо с наставка "Controller". Контролерите обработват постъпващите заявки, въведени от потребителя и изпълняват подходящата логика за изпълнение на приложението. Класът контролер е отговорен за следните етапи на обработка:

* Намиране и извикване на най-подходящия метод за действие (action method) и валидиране, че може да бъде извикан.
* Взимане на стойности, които да се използват като аргументи в метода за действие.
* Онстраняване на всички грешки, които могат да възникнат по време на изпълнението метода за действие.
* Осигуряване на клас WebFormViewEngine по подразбиране за отваряне на страници с изглет от тип ASP.NET.

Контролера е клас който се наследява от базовия клас System.Web.Mvc.Controller. Всеки публичен метод в контролера е показан като "controller action". Ако искаме даден метод да не бъде извикан, трябва да сложим "NonAction" атрибут върху неговото име. По подразбиране "Index()" действието е извикано за контролера, когато друго такова не е изрично упоменато.

## C#



C# е език от високо ниво, който прилича на Java и C++ и донякъде на езици като Delphi, VB.NET и C. Всички C# програми са обектно-ориентирани. Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика инструкциите, които компютърът изпълнява. В днешно време C# е един от най-популярните езици за програмиране. На него пишат милиони разработчици по цял свят. Тъй като C# е разработен от Майкрософт като част от тяхната съвременна платформа за разработка и изпълнение на приложения .NET Framework, езикът е силно разпространен сред Microsoft-ориентираните фирми, организации и индивидуални разработчици. Езикът C# и платформата .NET Framework се поддържат и контролират от Microsoft, но постепенно се отварят и за външния свят чрез миграция към отворен код с усилията на .NET фондацията (www.dotnetfoundation.org) стартирала през 2014 г. Поради години наред затворено развитие на езика C# и .NET платформата останали големи световни софтуерни корпорации като Google, Apple, IBM, Oracle и SAP базират своите решения на Java или други платформи и не използват C# като основен език за разработка на своите продукти. Езикът C# се разпространява заедно със специална среда, върху която се изпълнява, наречена Common Language Runtime (CLR). Тази среда е част от платформата .NET Framework, която включва CLR, пакет от стандартни библиотеки, предоставящи базова функционалност, компилатори, дебъгери и други средства за разработка. Благодарение на нея CLR програмите са преносими и след като веднъж бъдат написани, могат да работят почти без промени върху различни хардуерни платформи и операционни системи. Най-често C# програмите се изпълняват върху MS Windows, но .NET Framework и CLR се поддържа и за мобилни телефони и други преносими устройства, базирани на Windows Mobile. Под Linux, FreeBSD, MacOS X и други операционни системи, C# програмите могат да се изпълняват върху свободната .NET Framework имплементация Mono, която обаче не се поддържа официално от Microsoft. Тенденциите са Microsoft постепенно да започне да поддържа C# и .NET под Linux.

## MS SQL - база данни

### Релационна база данни

**Релационна база данни** е тип база данни, която съхранява множество данни във вид на релации, съставени от записи и атрибути (полета) и възприемани от потребителите като таблици. Релационните бази данни понастоящем преобладават при избора на модел за съхранение на финансови, производствени, лични и други видове данни.

Софтуерът, който се използва за организиране и управление на този вид бази данни се нарича най-общо система за управление на релационни бази данни (СУРБД).

### Отношение

**Отношение** (relationship, в някои източници с това значение е натоварен терминът релация) се нарича зависимост, съществуваща между две таблици, когато записи от първата таблица могат да се свържат по някакъв начин със записи от втората таблица. Три са възможните видове отношения, още известни като кардиналности или кардинални числа (cardinality):

* „едно към едно“ (1:1),
* „едно към много“ (1:N),
* „много към много“ (M:N).

**Отношение „едно към много“**

Отношението от вид „едно към едно“ е налице, когато всеки запис от една таблица е свързан с най-много един запис от втора таблица и всеки запис от втората таблица е свързан най-много един запис от първата таблица. Този вид отношение е специално, защото е единственото, при което двете таблици могат да споделят един общ първичен ключ. Възможно е обаче и първичните им ключове да са различни и отношението да се създава с използване на външен ключ, като първичният ключ на едната таблица, без значение коя, се включи в структурата на другата.

**Отношение „едно към много“**

Отношение „едно към много“ между две таблици съществува тогава, когато един запис от първата таблица, наречена родителска, може да бъде свързан с много записи от втората таблица, наречена дъщерна, но запис от дъщерната таблица може да бъде свързан само с един запис от родителската таблица. Отношението между двете таблици се създава като копие на първичния ключ на родителската таблица се включи в структурата на дъщерната таблица, за която той представлява външен ключ. В литературата се среща и кардиналността „много към едно“ (N:1), която е вариант на „едно към много“. Това е най-често срещаният вид отношение между таблици.

**Отношение „много към много“**

Отношението „много към много“ съществува когато един запис от едната таблица може да се свърже с много на брой записи от втората таблица, и един запис от втората може да се свърже с много на брой записи от първата таблица. За да се създаде на практика това отношение, се използва нова, свързваща или асоциираща таблица, която съдържа копия на първичните ключове на двете таблици. От една страна свързващата таблица представлява сложен първичен ключ на отношението, а от другата страна, всеки от първичните ключове на изходните таблици играе ролята на външен ключ за свързващата таблица.

### Нормализация на базата данни

Нормализацията изпълнява следните задачи:

* Изключване на повтаряща се информация в таблиците.
* Декомпозиция на един типов обект на няколко.
* Минимизиране на аномалиите при съхраняване, изтриване и промяна на данните.
* Създаване на отворена към бъдещи промени структура.
* Създаване на структура, свеждаща до минимум влиянието на структурни изменения върху вече създадени приложения.

Нормализацията, т.е. привеждането в нормална форма, включва набор от практики по отстраняването на повторения сред данните, което от една страна води до икономия на памет и повишено бързодействие, а от друга страна предпазва от аномалии при манипулирането на данните (вмъкване, актуализиране и изтриване) и от загуба на тяхната цялост. В процеса на нормализация се осигурява оптимална структура на базата от данни, основаваща се на взаимозависимостта между данните. Структурата на таблиците се трансформира, с цел да се оптимизират функционалните зависимости на съставните им атрибути.

## jQuery



JQuery е една от най-известните и най-използваните библиотеки, алтернатива на JavaScript, публикувана в началото на 2006 от Джон Резиг. В основата си jQuery опростява достъпа до всеки елемент на дадена уеб-страница, като по този начин позволява лесно изграждане на динамична функционалност в страниците. JQuery е създаден с цел да промени начина, по който се пише JavaScript. Синтаксисът му е сравнително лесен за научаване и употреба, а възможностите и гъвкавостта на jQuery подпомагат за изключителни резултати. Поддържа се от всички съвременни браузъри, което е още една от многото причини за голямата му популярност.

JQuery е безплатна библиотека с отворен код, лицензиран под MIT лиценз. JQuery се използва в 55% от 10000-те най-посещавани сайтове, което я прави най-популярната JavaScript библиотека днес

JQuery архитектурата позволява на разработчиците да създават приставки (plug-in), като по този начин разширяват нейната функционалност. В момента има на разположение в интернет над 16 хиляди jQuery приставки, които обхващат широк спектър от функционалности, като помощни приложения тип Ajax, уеб услуги, мрежови масиви от данни, динамични списъци, XML и XSLT инструменти, drag and drop приложения, събития, управление на бисквитки, модални прозорци и други.

## CSS



CSS е съкращение от Cascading Style Sheets и представлява отделен език, съдържащ множество "инструменти", с които може да се промени външния вид на HTML страниците. От определена гледна точка CSS е нещо като "надстройка" на HTML. Официално спецификацията на CSS се поддържа от W3C.

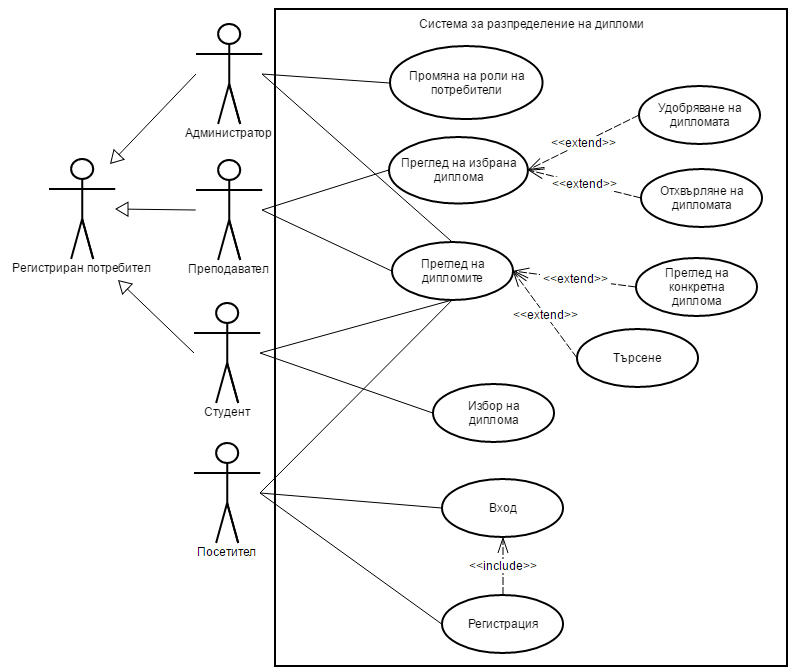
CSS е създаден с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Преди стандартите за CSS, установени от W3C през 1995 г., съдържанието на сайтовете и стила на техния дизайн са писани в една и съща HTML страницата. В резултат на това HTML кода се превръща в сложен и нечетлив, а всяка промяна в проекта на даден сайт изисквала корекцията да бъде нанасяна в целия сайт страница по страница. Използвайки CSS, настройките за форматиране могат да бъдат поставени в един единствен файл, и тогава промяната ще бъде отразена едновременно на всички страници, които използват този CSS файл.

CSS позволява да се определя как да изглеждат елементите на една HTML страница - шрифтове, размери, цветове, фонове, и др. CSS кодът се състои от последователност от стилови правила, всяко от които представлява селектор, последван от свойства и стойности.

Селекторите се използват за да покажат към кои елементи на HTML документа трябва да бъде прилаган съответният стил. Съществуват много видове селектори. Някои селектори позволяват постигане на динамичност на страницата до определена степен. Например само с помощта на CSS могат да бъдат направени изскачащи менюта, хипервръзки, които при посочване променят цвета си и др.

# Проектиране на системата.

В настоящата дипломна работа ще предложа решение за изграждане на софтуерна система за разпределение и съставяне на дипломни задания. Целта на тази система е да спомогне абсолвентите и ръководителите на дипломни задания, като предложи едно лесно за употреба web базирано приложение.



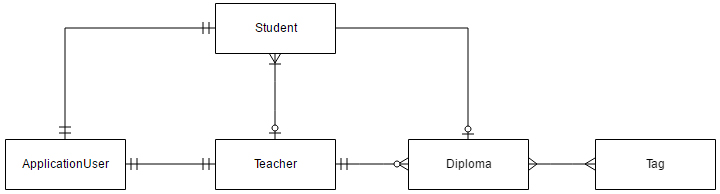
Фигура 3.1 Use case диаграма

На фиг. 3.1 е представена use case диаграма на система за разпределение и съставяне на дипломни работи. Представени са основните действия, които могат да се извършат от отделните потребители с различни роли.

## Съставяне на моделите за базата данни



При съставяне на план за приложение е необходимо да се поставят солидни основи. В този проект основата е правилно съставена база данни и начина по който отделните елементи са свързани помежду си.



Приложение за съставяне на дипломни работи се нуждае от потребител, с който да се извършват операции за идентификация в системата. ApplicationUser е такова entity. То пази в себе си информацията за отделния потребител, като имена, телефон, email и т.н.

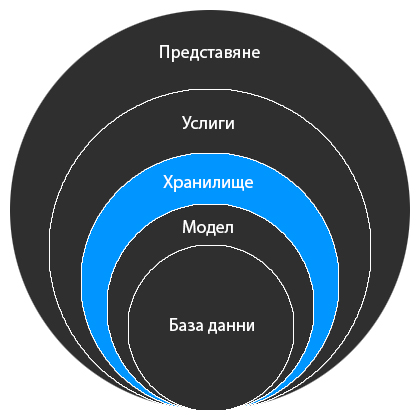
Класовете Student и Teacher са класове, в които се запазва допълнителна информация за потребителя. Изнасяйки ги в отделни класове, ще се намали отговорността на ApplicationUser класа и в същото време един потребител може да се възползва от архитектурата и на двата класа (един потребител ще може да е и студент и преподавател).

* В класа Student се запазва всякаква информация касаеща отделния студент (факултетен номер, коя диплома е избрал).
* Класът Teacher се съдържат полета с информация за конкретен учител. В този клас се записват дипломите, които преподавателя е добавил, както и студентите, които е поел като дипломанти.

Класът Diploma съдържа информация за индивидуална диплома, като полета за имена, съдържание, дата на задаване, както и служебна информация за дипломата, която да бъде използвана в логическата част на приложението.

Класът Tag представлява ключова дума, която да определя дипломната работа. Съдържа проста информация като име, дата на създаване, както и дипломи, които използват този таг.

## Съставяне на хранилище за елементите на базата (repository)



Използвайки repository pattern, отделяме основната част на приложението (web частта) от базата данни. Тестване на бизнес логиката става по-лесно, като по този начин не сме зависими от базата данни и можем да работи отделно от вида и типа й. Улеснява се достъпът до ресурсите, когато имаме достъп до тях от няколко места, като прилага консистентни правила и логика за работа. Този шаблон улеснява поддръжката и разчитането на кода като разделя бизнес логиката от данните и услугите.

В този слой се дефинират някои основни действия свързани с моделите като:

* Връщане на всички елементи от даден тип (модел или таблица от базата данни)
* Взимане на запис по ключ от базата данни
* Добавяне на елемент от даден тип към базата данни
* Изтриване на елемент от базата данни (маркиране на елемента като изтрит и пълно изтриване)
* Записване на промените по базата данни

## Създаване на слой с услуги (Service layer pattern)



Service layer е модел, прилаган при дизайн ориентиран към услугите, и има за цел да ги организира, в рамките на едно общо място като логически слоеве. Услуги, които се категоризират с определена функционалност обуславят един слой. Това помага за намаляване на сложността на управлението на слоевете на услуги, тъй като услугите, които принадлежат към един и същи слой адресат по-малък набор от дейности.

Употребата на услуги (services) улеснява допълнително работата с данните, както и unit тестването на контролерите на web приложението. Достъпът през service дава възможност за настройка на върнатата информация според нуждите на приложението (например сортиране по име на студентите).

Услугите, които са необходими са директно свързани с типовете модели, които има приложението.

Услуга за операции с дипломи, трябва да има някои фундаментални операции като:

* Избиране на диплома по ID
* Избиране на всички дипломи
* Избиране на дипломи принадлежащи на даден преподавател

Услугата за работа с преподаватели, трябва да има следните операции:

* Избор на преподавател по ID
* Избор на преподавател по ID на потребителския модел (ApplicationUser)
* Добавяне на диплома към колекцията от дипломи на преподавателя
* Добавяне на студент към колекцията от дипломанти на преподавателя

Услуга за работа със студенти, трябва да има следните операции:

* Избор на студент по ID
* Избор на студент по ID на потребителския модел (ApplicationUser)
* Избор на студент по факултетен номер
* Добавяне на диплома
* Премахване / промяна на избрана диплома

## Създаване на подходящи контролери и изгледи



За да мога да избера подходящи контролери за приложението е необходимо да се изследват целите на самото приложение. То има няколко основни изисквания като:

* Приложението трябва да може да поддържа регистрации
* Трябва да има потребители от различни роли (администратори, студенти, преподаватели)
* Всеки от типовете потребители има свой задачи (действия), които може да изпълнява

### Регистрация на потребители

За да се регистрира нов потребител, web приложението може да използва предоставената от Microsoft архитектура. Класовете, които се използват на готово, включват основна информация за потребителите. В това число и контролер за управление на потребителя.

Тези готови класове трябва да бъдат разширени за нуждите на приложението. Тъй като основната задача на проекта е да се улесни процеса на избор на диплома от абсолвентите, при регистрирането на потребител е подходящ момент за създаването на обект от тип студент, които да е обвързан с новия потребител.

### Избор на роли в приложението

Приложението трябва да има няколко основни роли. Потребители, които да управлява самото приложение (**администратори**). Тези потребители са няколко за целия проект. За създаване на тази функционалност трябва да има контролер, които да бъде достъпен само за тях (администраторите на системата). Функционалностите, които са необходими да има са промяна на част от потребителската информация и промяна на правата на останалите потребители (студенти, преподаватели).

Приложените трябва да има потребители от тип студенти.

Приложените трябва да има потребители от тип преподаватели.

### Създаване и редактиране на дипломи от преподавателя

### Редактиране на дипломи от студенти

# Програмна реализация.

# Ръководство на потребителя

# Заключение

Заключението резюмира основните характеристики и особености на представеното инженерно решение, неговите предимства и недостатъци, резултатите от експериментите, тяхната инженерна трактовка и предложения за по-нататъшна работа върху решаването на зададения проблем.

# Списък на използвани литературни източници

# Приложения

Приложенията съдържа изходен текст на програми, формати на структури от данни, екранни форми, получени резултати, електрически схеми, графични оригинали на печатни пратки, справочни данни и др.