****

**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

**Факултет „Компютърни системи и технологии“**

**Катедра „Компютърни системи“**

**КУРСОВ ПРОЕКТ по „Програмиране за интернет“**

**Изготвил:** Любен Димитров Кирев, №121321051, група 222

**Ръководител:** доц. д-р инж. Антония **Ташева**

**София, 2022 г.**

Съдържание:

[1. Общо описание на възможностите на реализирания проект 3](#_Toc92675295)

[2. Структура (архитектура) на приложението 7](#_Toc92675296)

[3. Схема на таблиците и релациите им в базата данни 8](#_Toc92675297)

[4. Обяснение на реализирани функции 9](#_Toc92675298)

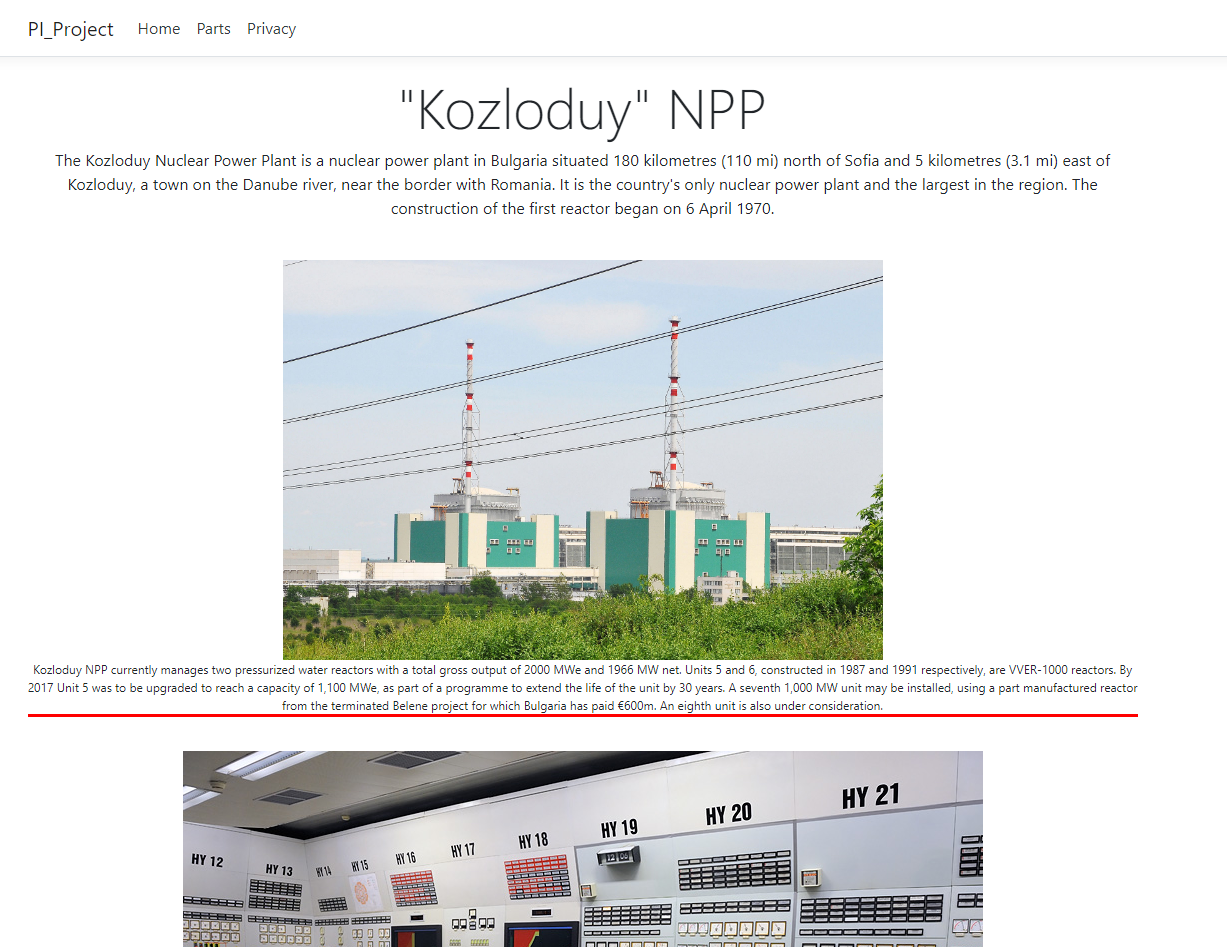
[5. Списък с фигури 10](#_Toc92675299)

# Общо описание на възможностите на реализирания проект

Реализираният проект е базиран на личен проект, който до в момента се използва в ЕАД „АЕЦ Козлодуй“ служещ за организация на резервни части за Пълномащабен Симулатор-1000, симулиращ експлоатацията и операциите на Блоково Щит Управление на блок 6 на АЕЦ „Козлодуй“. Проектът е построен с цел да се оползотвори плюсовете на междуплатформените възможности на ASP.NET фреймуърка представени пред мен в процеса на обучение по предмета „Програмиране за Интернет“ в учебния план за КСИ, ОКС магистър. Новопридобитите знания разширяват мирогледа за бъдещи възможности за програмно реализиране на идеи, като с помощта на Интернет приложения могат да достигнат по-лесно масите потребители извън тясно специализирания кръг от хора, който би използвало текущата изработка. Целта е да се улесни организацията на стотиците резервни части необходимо за безотказната работа на ПМС-1000 като се организират прегледно и се групират адекватно, за спестяване на време и усилия при необходимост на подмяна на дефект.

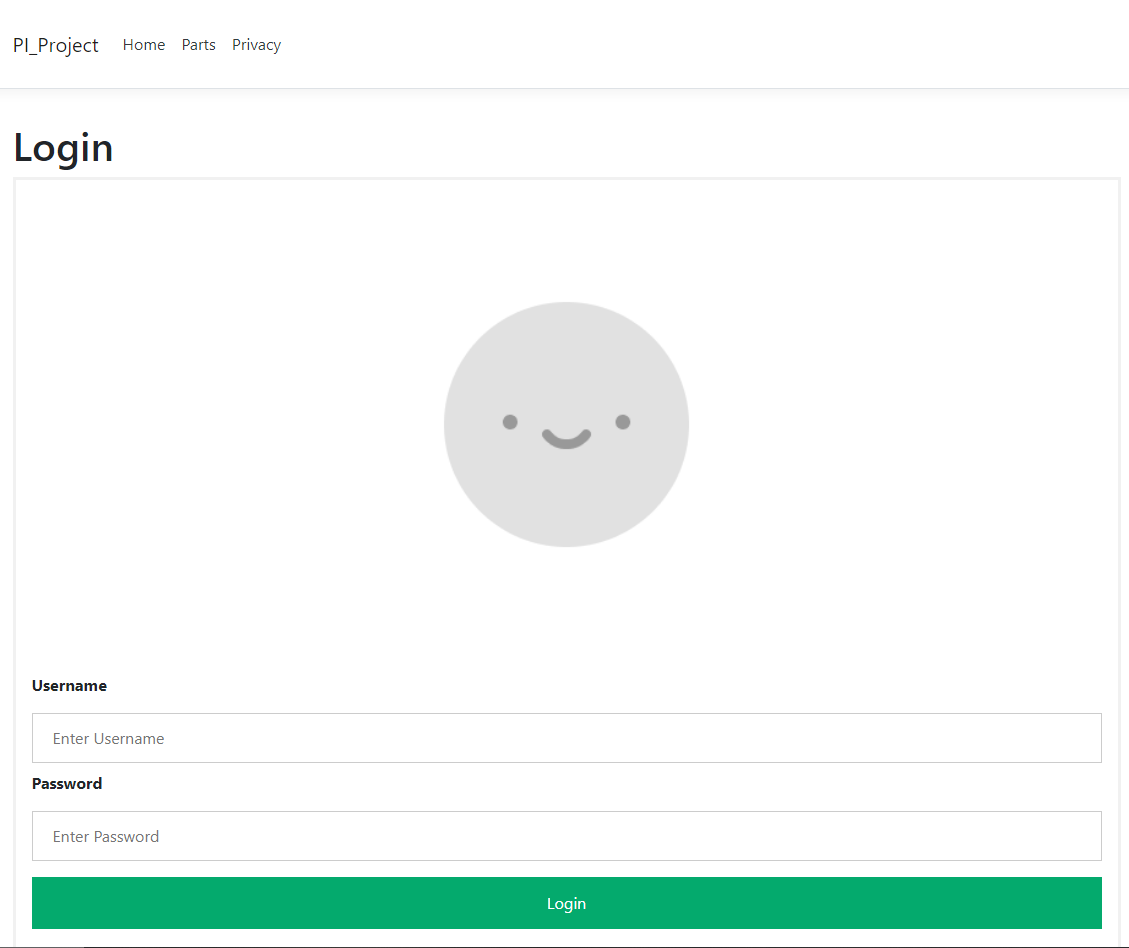
Проектът се реализира с framework-а ASP.NET използвайки архитектурата Model-View-Controller (MVC). Езиците, които са използвани са C#, HTML, CSS, SQLServer с помощта на Entity Framework Core. Архитектурата MVC се явява изключително подходяща за създаване на уеб-базирани приложения като се разделят основните дейности на няколко отделни части, което следва за подобрения в изпълнение на програмата, в създаване на програмата и в поддръжката на програмата, което във всеки проект се явява най-сложния и дълъг стадии в живота му. Контролерът е основния път на логиката, като той приема входните данни и ги обработва и връща информация. View-то е частта от архитектурата, където тази информация се визуализира за пред потребителя. Има за цел да е възможно най-интуитивно и прегледно за разбиране на изходната и входната информация. Моделът е описание на данните които ще се използват в програмата.

Приложението приветства потребителя с кратко въведение за АЕЦ „Козлодуй“ като има и показани три снимки въведени динамично чрез controller-view методология включително и текста в параграфите. Горе в навигационната лента, т.нар. меню на сайта, откриваме и линк към Privacy секцията където в реални условия трябва да се въведат данните за авторски права на собствениците на уеб-приложението.



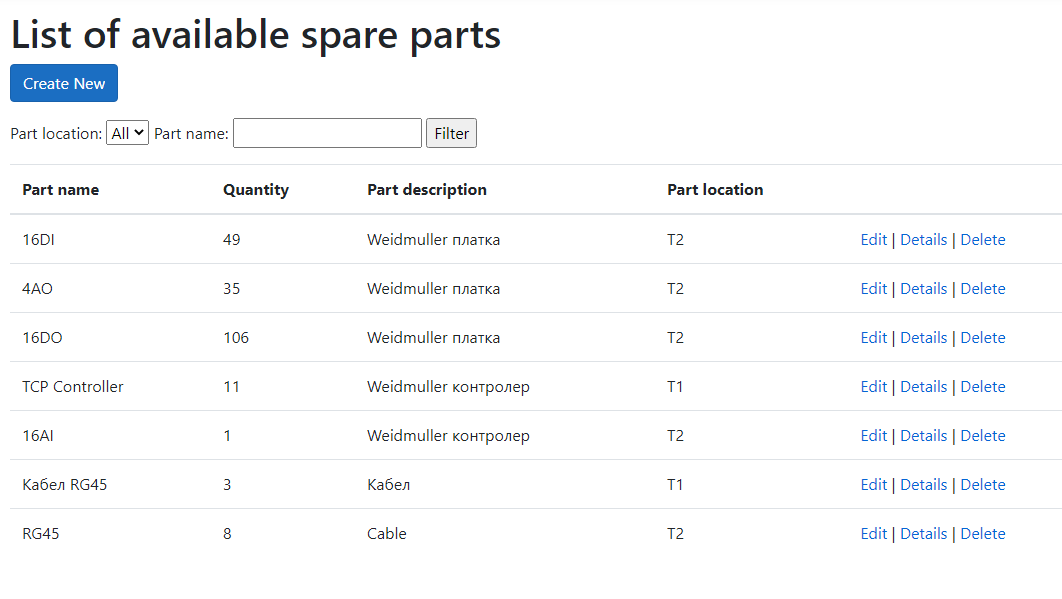
*Фиг. 1: Начален екран на уеб-приложението*

В линка за Parts където се случват реално изобразяването на данните и въвеждането/редактирането и/или изтриването от базата с данни, първо ни приветства форма за вход в системата. Идеята е, че информацията е класифицирана на много високо ниво като повечето данни в предприятието и въпреки, че системата е предназначена да е изцяло за вътрешна употреба, не бихме желали да се злоупотребява от неоправомощени лица. За цел демонстрация, са създадени двама потребителя „Pesho, test“ и „Ivan, test123“. Потребителите имат абсолютни администраторски права щом се валидират пред системата. Сигурността от неприятели е осигурена като за предаване на данни се използва метода „POST“, вместо „GET“.



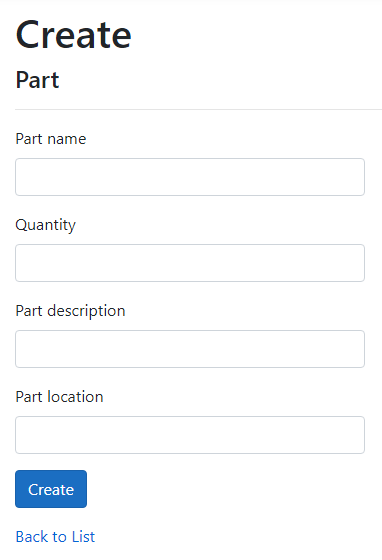
*Фиг. 2: Формата за вход в системата*

Потребителите нямат достъп до списъка с части докато не се автентифицират. След като успешно преминат проверката, се препращат към страницата с изписаните налични части. Има примерни попълнени в базата с данни. Всяка част има три допълнителни колони освен името ѝ, които са за количество, описание на частта и местоположение (в примерите в стая Т1 или Т2). Към списъка с всички части има и два допълнителни филтъра които правят търсенето по-прегледно за потребителя. Единият филтър е за търсене по име на частта, а другия за местоположението на частта с даденото име, или всички части на това място, ако името е празно.



*Фиг. 3: Списък с налични части*

В страницата има предоставена възможност за редактиране, детайли и/или изтриване на редица от базата с данни, като този списък чете директно от SQL Server базата. Има и наличен бутон за добавяне на нови попълнения към системата като има и проверка за автентичност на въведените данни преди да бъде дадена заявка към сървъра.



*Фиг. 4: Формата за създаване на ново попълнение*

# Структура (архитектура) на приложението

Приложението се състои от три контролера, четири View папки, съдържащи отделните .cshtml файлове, четири модела и един ViewModel и два .cs файла където съдържа инструкциите за работата на програмата.

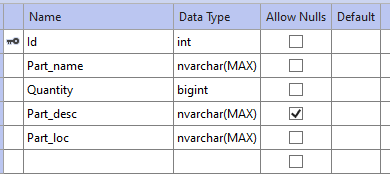
Първият контролер, HomeController, съдържа в себе си Index метода, където се дават данните които ще бъдат визуализирани на началната страница на сайта посредством Home/Index view-то. PartsController включва няколко функции, които са жизненоважни за програмата. Още в конструктора създаваме обекта за Entity Framework Core базата с данни, където посредством нея приемаме информацията записана в SQL Server. Имаме функция Index, която приема два аргумента: PartLocation и SearchString. От там намираме в базата с данни тези въведени според аргумента, като за местоположението на частите създаване ViewModel, който описва модела на view-то което ще бъде показано. Функциите се изпълняват асинхронно, като с цел да се запази конфиденциалността се използва POST метода за изпращане на данни обратно към сървъра от view-то. UsersController главната функция която се изпълнява е Login(string uname, string psw). В нея се осъществява проверката за кои лица са правоспособни за достъп и кои не са. Сравняват се входните параметри на функцията с такива в базата с данни. Ако се намери съответствие, може да продължи.

При изпълнението на повечето функции от контролерите е необходимо резултата да бъде визуализиран чрез View-та. Някои от главните View-та които се ползват са Home/Index, Parts/Index, Users/Index и Shared/\_Layout. Формирането със CSS се намира в папката wwwroot/css/site.css въпреки, че в някои view-та има директно въведен CSS в head-а за прегледност. В тях са описани с помощта на HTML дизайна и съдържанието на програмата, като е направена връзката и с ViewData[].

Моделите в програмата се използват за описание на данните които ще се използват. Те са Part, User, PartLocationViewModel и SeedData. В описанието на Part модела също са въведени и Regular Expression изрази за достоверност и отказоустойчивост на въведените в системата данни. В SeedData модела инициализираме базата с данни.

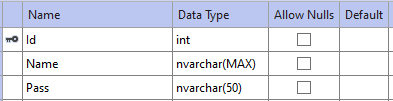
# Схема на таблиците и релациите им в базата данни

Самата база с данни се състои от две таблици, които не са свързани една с друга. Те са Part и Users.



*Фиг. 5: Таблицата Part*

Задължително поле за всички бази с данни е уникален идентификатор, ID, който служи за главен ключ. Part\_name е низ, който е задължителен; Quantity е естествено число, което е задължително; Part\_desc е низ, който не е задължителен; Part\_loc също се запазва като задължителен низ.



*Фиг. 6: Таблицата Users*

Освен полето Id, играе ролята на главен ключ, полетата Name и Pass са задължителни низове където се съхраняват съответно потребителското име и паролата на потребителя.

# Обяснение на реализирани функции.

HomeController/Index функцията попълва чрез ViewData[] директно параграфи към Home/Index view-то html кода.

PartsController/Index приема два аргумента PartLocation и SearchString съответно за избор на местоположението на частта и търсене по име на частта дадено от потребителя. Създаваме заявката към базата с данни за местоположението и за частите. С цел надеждност и отказоустойчивост на програмата, проверяваме възможността дали дадените низове са празни. Ако не са, присвояваме стойността на намерената част към променливата. От PartLocationViewModel-а взимаме лист с местоположенията и частите. Връщаме View с аргумент променливата на PartLocationViewModel.

PartsController/Details приема за аргумент уникалния идентификатор id за избраната част. Към View/Details подаваме информацията за намерената част.

Първият Create отваря view-то, а втория създава от въведените данни като аргумент се добавят към базата с данни по метода POST.

Подобно на Create, Edit е две функции. Първата намира според дадено id към базата с данни такава част, като има и проверка за валидността на данните. Ако всичко е наред, ще отвори view-то Edit. Другата функция приема данните от формата за обработка за достоверност. Проверява се дали съществува тази част, след което в try-catch блок с цел да не прекъсва работата на програмата се прави опит да се запазят новите данни на частта към съществуващата такава в базата с данни с метода POST.

Delete първо почва с избор по Id за частта която желаем да заличим от базата и отваряме view с тези данни. При потвърждаване на изтриването, според id-то изтриваме от базата с данни, запазваме текущото състояние и се връщаме обратно.

Функцията PartExists(int id) служи за проверка дали тази част вече се среща.

В UsersController/Index извършваме проверката за съществуването на такъв потребител и правилното въвеждане на данните му. Входни параметри се използват string uname и string psw. Създаваме списък с всички потребители в базата с данни като след това с помощта на foreach цикъл преминаваме през целия списък и проверяваме дали има съответствие на въведените данни с тези които се намират в БД. При успешно намиране ни препраща към Parts/Index; ако няма съответствие, ни връща обратно на същата страница. Целта е само хора с определен достъп да могат да боравят с информацията в базата с данни предоставена чрез уеб приложението.

# Списък с фигури:

* Фиг. 1: Начален екран на уеб-приложението, стр. 4
* Фиг. 2: Формата за вход в системата, стр. 5
* Фиг. 3: Списък с налични части, стр. 6
* Фиг. 4: Формата за създаване на ново попълнение, стр. 7
* Фиг. 5: Таблицата Part, стр. 8
* Фиг. 6: Таблицата Users, стр. 9