



**WYŻSZA SZKOŁA
INFORMATYKI I ZARZĄDZANIA**
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA
Specjalność: Programowanie
Projekt: Projekt zespołowy

Eryk Winiarski

Nr albumu studenta 61995

Aplikacja do zarządzania pracownikami

Rzeszów 2022

Spis treści

1.	Wstęp	3
2.	Opis poszczególnych sesji.....	3
3.	Programy, technologie i wymagania	3
4.	Baza danych.....	5
5.	Kod programu.....	6
6.	Testy	10
7.	Podsumowanie	10

1. Wstęp

Projekt zawiera w sobie opis aplikacji do zarządzania pracownikami. Do bazy danych będą dodawane imiona i nazwiska, stanowisko oraz wynagrodzenie pracownika. Dane będą miały możliwość ciągłej edycji.

2. Opis poszczególnych sesji

Sesja I:

- Zdefiniowanie typu aplikacji i wybór narzędzi do jej zrealizowania,
- Zapoznanie się z literaturą na temat używanych technologii,
- Określenie problemu.

Sesja II:

- Określenie celów projektu,
- Wstępny opis funkcjonalności aplikacji,
- Określenie wymogów i ograniczeń.

Sesja III:

- Generowanie hipotez, wraz z rozwiązaniami.

Sesja IV:

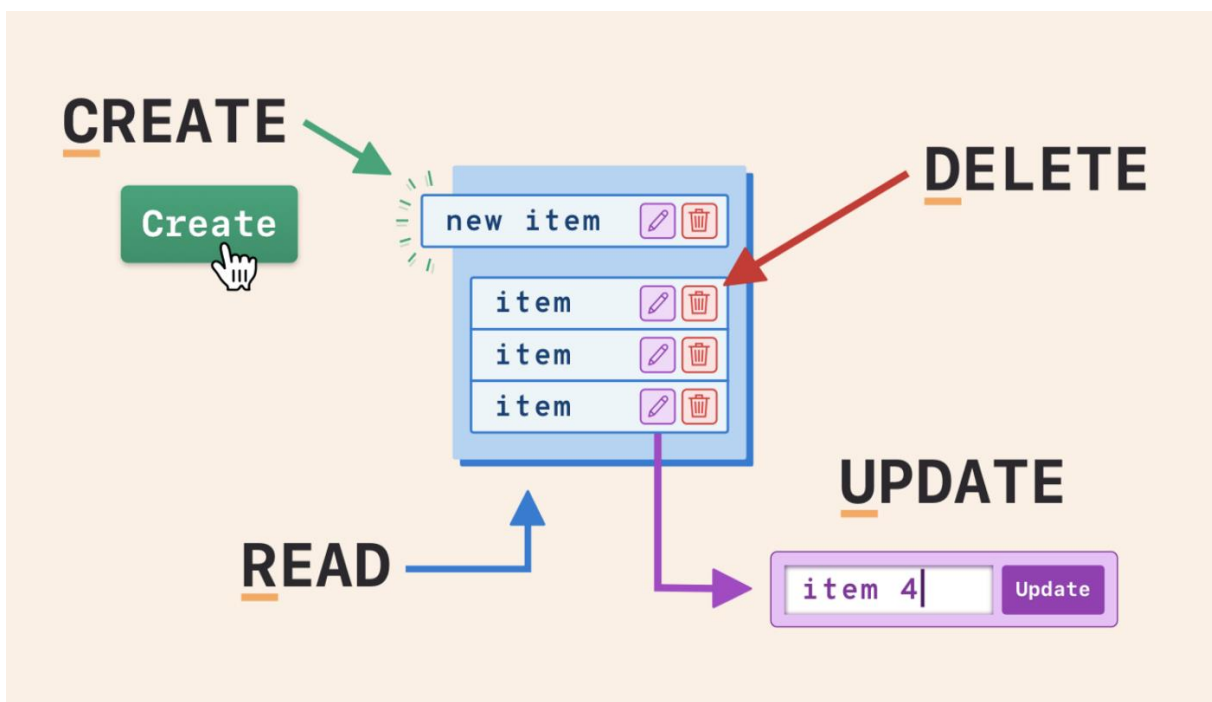
- Przedstawienie prototypu z działającymi rozwiązaniami,
- Określenie ram finalnego projektu.

3. Programy, technologie i wymagania

Visual Studio - zintegrowane środowisko programistyczne firmy Microsoft. Jest używane do tworzenia oprogramowania konsolowego oraz z graficznym interfejsem użytkownika.

C# - wieloparadygmatowy język programowania zaprojektowany w latach 1998–2001 przez zespół pod kierunkiem Andersa Hejlsberga dla firmy Microsoft. Program napisany w tym języku kompilowany jest do języka Common Intermediate Language (CIL), specjalnego kodu pośredniego wykonywanego w środowisku uruchomieniowym takim jak .NET Framework, .NET Core, Mono lub DotGNU.


CRUD - (od ang. create, read, update, delete, tłum. utwórz, odczytaj, aktualizuj, usuń) – cztery podstawowe funkcje w aplikacjach korzystających z pamięci trwałej, które umożliwiają zarządzanie nią.



ASP.NET - zbiór technologii opartych na frameworku zaprojektowanym przez firmę Microsoft. Przeznaczony jest do budowy różnorodnych aplikacji internetowych, a także aplikacji typu XML Web Services. ASP.NET rozszerza platformę deweloperską .NET o narzędzia i biblioteki przeznaczone specjalnie do tworzenia aplikacji internetowych.

4. Baza danych

Stworzona baza danych jest bardzo prosta i składa się z jednej tabeli.

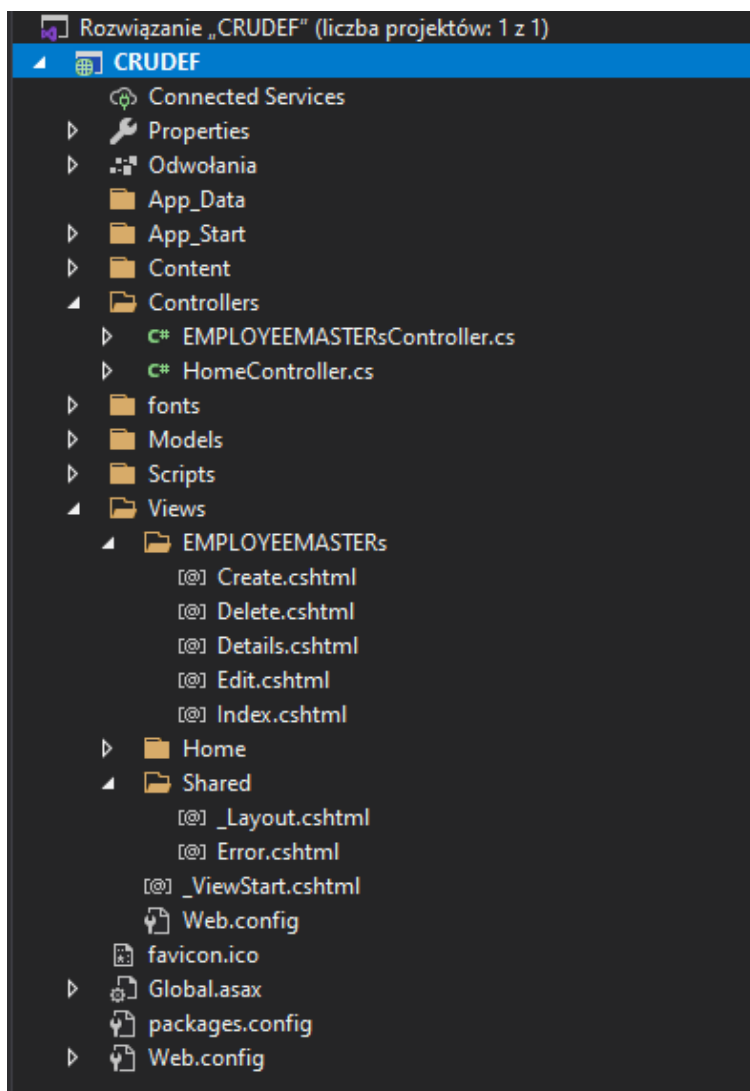
	Column Name	Data Type	Allow Nulls
	ID	int	<input type="checkbox"/>
	EMPCODE	int	<input type="checkbox"/>
	EMPNAME	varchar(100)	<input type="checkbox"/>
	DESIGNATION	varchar(50)	<input type="checkbox"/>
	SALARY	int	<input type="checkbox"/>
			<input type="checkbox"/>

Rysunek 1 Baza danych

Zawiera się w niej ID pracownika, jego kod, imię i nazwisko, „przeznaczenie” - stanowisko i wynagrodzenie.

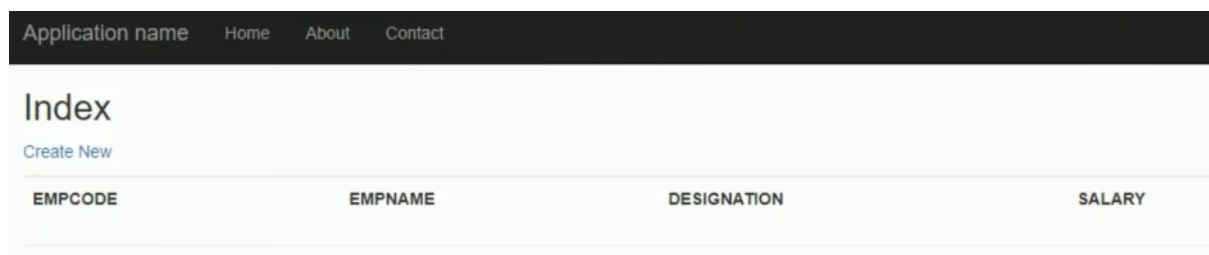
5. Kod programu

Po stworzeniu projektu w Visual Studio należy stworzyć odpowiedni model. Będzie to ADO.NET Entity Data Model. Łączymy przez to nasz program z wcześniej zbudowaną bazą danych. W następnym kroku tworzymy kontroler do naszej bazy danych.



Rysunek 2 Rozwiązanie

Na tym etapie nasze rozwiązanie prezentuje się następująco. Mamy już elementy odpowiedzialne za sterowanie bazą danych- tworzenie, usuwanie, edytowanie i podgląd danych.



Rysunek 3 Index

Żeby jednak prawidłowo wyświetlać dane pracowników należy dokonać zmiany w kodzie, a mianowicie w Views > Shared > _Layout.cshtml.

```
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
@Html.ActionLink("Nazwa aplikacji", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
</div>
<div class="navbar-collapse collapse">
<ul class="nav navbar-nav">
<li>@Html.ActionLink("Strona główna", "Index", "Home")</li>
<li>@Html.ActionLink("Informacje", "About", "Home")</li>
<li>@Html.ActionLink("Kontakt", "Contact", "Home")</li>
<li>@Html.ActionLink("Employee Master", "Index", "EMPLOYEEMASTERS")</li>
</ul>
</div>
</div>
</div>
```

Rysunek 4 Zakładka

Po dodaniu zakładki „Employee Master” można w pełni korzystać z programu i sprawdzić jego działanie.

Index

Create New				
EMPCODE	EMPNAME	DESIGNATION	SALARY	
123	Dawid	IT	8000	Edit Details Delete

Rysunek 5 Index

Po edycji kodu i wejściu w zakładkę Employee Master zobaczymy bazę z obecnie dodanymi pracownikami.

Edit Employee

EMPCODE	<input type="text" value="123"/>
EMPNAME	<input type="text" value="Dawid"/>
DESIGNATION	<input type="text" value="IT"/>
SALARY	<input type="text" value="8000"/>
<input type="button" value="Update"/>	

[Back to List](#)

Rysunek 6 Edit

Dane pracowników można dowolnie edytować.

Employee Details

EMPCODE	123
EMPNAME	Dawid
DESIGNATION	IT
SALARY	8000

| [Back to List](#)

Rysunek 7 Details

Można również sprawdzić dokładne dane pracownika.

Delete Employee

Are you sure you want to delete this?

EMPCODE	123
EMPNAME	Dawid
DESIGNATION	IT
SALARY	8000

[Delete](#) | [Back to List](#)

Rysunek 8 Delete

Istnieje również możliwość usunięcia pracownika z listy.

Create Employee

EMPCODE	<input type="text"/>
EMPNAME	<input type="text"/>
DESIGNATION	<input type="text"/>
SALARY	<input type="text"/>
	Create

[Back to List](#)

Rysunek 9 Create

Najważniejszym elementem jest dodawanie nowych pracowników do bazy.

6. Testy

Na poniższym fragmencie kodu zaprezentowane są przeprowadzone testy.

```
using System;
using Xunit;
using System.Web.Mvc;
using CRUDEF;
using System.Configuration;
namespace TestProject2
{
    Odwołania: 0
    public class UnitTest1
    {
        [Fact]
        Odwołania: 0
        public void HomeControllerTest()
        {
            var cont = new CRUDEF.Controllers.HomeController();
            Xunit.Assert.NotNull(cont);
        }
        //[Fact]
        //public void IndexControllerTest()
        //{
        //    var cont = new CRUDEF.Controllers.HomeController();
        //    Xunit.Assert.NotNull(cont.Index());
        //}
        [Fact]
        Odwołania: 0
        public void AboutControllerTest()
        {
            var cont = new CRUDEF.Controllers.HomeController();
            Xunit.Assert.NotNull(cont.About());
        }
        [Fact]
        Odwołania: 0
        public void ContactControllerTest()
        {
            var cont = new CRUDEF.Controllers.HomeController();
            Xunit.Assert.NotNull(cont.Contact());
        }
    }
}
```

7. Podsumowanie

Cała aplikacja po przetestowaniu działa w pełni prawidłowo. Można w przyszłości dodać do wyżej zaprezentowanego programu takie opcje jak:

- Wyszukiwanie pracownika po nazwie
- Waluta w której otrzymuje wynagrodzenie

Możliwości, które pozostawia projekt jest dużo. Jest to jego podstawowa wersja, która spełnia swoje zadanie przy mniejszej ilości pracowników.