



Střední průmyslová škola a Vyšší odborná škola, Písek, Karla Čapka 402, Písek

18-20-M/01 Informační technologie

Maturitní práce

Správa povinných prací

Téma číslo 1

autor:

Richard Kropáček, B4.I

vedoucí maturitní práce:

Mgr. Milan Janoušek

Písek 2020/2021

Anotace

Výstupem mé maturitní práce je fungující systém, jehož úkolem je správa povinných prací odevzdaných za uplynulý školní rok žáky. Učitelé následně mohou práce procházet, stahovat a hodnotit.

Klíčová slova: C#; ASP.NET Core; Entity Framework Core; HTML; JavaScript; CSS; Bootstrap; MySQL; WebApp

Annotation

The output of my graduation thesis is a functioning system, the task of which is the administration of compulsory work submitted by pupils for the past school year. Teachers can then browse, download and evaluate the work.

Key words: C#; ASP.NET Core; Entity Framework Core; HTML; JavaScript; CSS; Bootstrap; MySQL; WebApp

Poděkování

Tímto bych chtěl poděkovat Mgr. Milanu Janouškovi za vedení mé Maturitní práce, cenné rady a odborný dohled. Děkuji také za pomoc všem, kteří se jakkoliv podíleli na realizaci této práce.

Obsah

1	Úvod	5
2	Teoretický úvod	6
2.1	Programovací jazyk C#	6
2.2	Razor	6
2.3	ASP.NET, ASP.NET CORE	7
2.3.1	ASP.NET	7
2.3.2	ASP.NET Core	8
2.4	EF6 a EF Core	8
2.4.1	Entity Framework 6	8
2.4.2	Entity Framework Core	8
2.5	MVC	8
2.5.1	Model	9
2.5.2	View	9
2.5.3	Controller	10
2.6	MySQL	11
3	Základní struktura ASP.NET Core	12
3.1	Model	12
3.2	View	15
3.3	Controller	15
3.3.1	Detail	15
3.3.2	Editace	17
3.3.3	Smazání	19
4	Databáze	21
5	Závěr	22

Přílohy	27
A Příloha	28

Kapitola 1

Úvod

Pro realizaci této práce jsem se rozhodl využít ASP.NET Core, což jest otevřený framework pro tvorbu webových aplikací vyvinutý společností Microsoft. Pro databázi jsem se rozhodl využít MySQL, což je velice rozšířený otevřený systém řízení báze dat. Webová aplikace je psána pomocí značkovacího jazyku HTML. Pro vizuální část stránky jsem využil Bootstrap 4.

Kapitola 2

Teoretický úvod

2.1 Programovací jazyk C#

C# je moderní, vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý společností Microsoft. Jazyk C# se řadí mezi typově bezpečné programovací jazyky. To znamená, že nedovoluje provádět operace, které mohou vést k chybám. Tento jazyk umožňuje vývojářům vytvářet mnoho druhů zabezpečených a robustních aplikací, které běží na platformě .NET¹.

C# je zároveň objektově orientovaný programovací jazyk orientovaný na součásti². Z toho vyplývá, že se zaměřuje na vytváření komponent, které jsou tvořeny často se opakujícími částmi kódu. C# také poskytuje jazykové konstrukce pro přímou podporu těchto konceptů, což z něj dělá přirozený jazyk pro tvorbu a používání těchto softwarových komponent.[13]

2.2 Razor

Razor je syntaxe pro ASP.NET používaná pro tvorbu dynamických webových stránek společně s programovacím jazykem C#. Pomocí Razor syntaxe lze vložit do webové stránky blok kódu, který se provede na straně serveru. Soubory využívající tuto syntaxi mají příponu .cshtml.

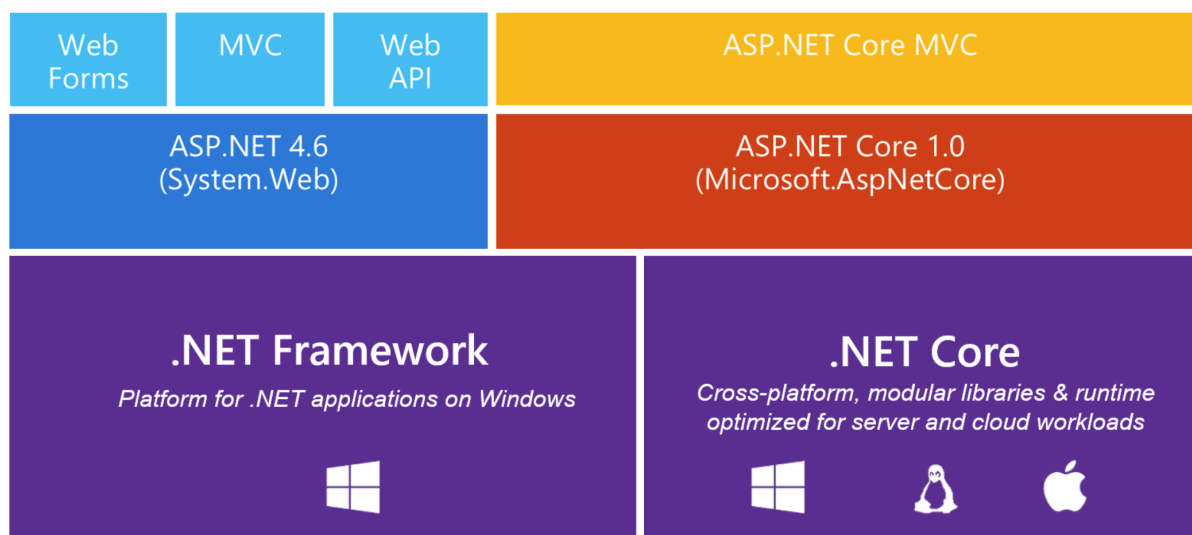
Razor syntaxe se skládá z Razor značek, C# a HTML. Výchozím Razor jazykem je HTML. Pro označení C# kódu využijeme symbolem @. Tímto dojde k přechodu z formátu HTML do C#. Razor vyhodnotí výrazy jazyka C# a vykresluje je ve výstupu HTML.

¹Vývojářská platforma pro vytváření webových, mobilních, desktopových, herních, IoT a dalších aplikací. Je podporovaná v systémech Windows, Linux a macOS.

²Component-oriented programming language

Kód HTML v souborech s příponou .cshtml se vykreslí serverem beze změny.[2]

2.3 ASP.NET, ASP.NET CORE



Obrázek 2.1: Porovnání ASP.NET Core a ASP.NET [3]

ASP.NET je webový framework obsahující sadu knihoven, které obsahují hotová řešení mnoha základních problémů, které ve webových technologiích vyvstávají. Může se jednat např. o bezpečnost, autentifikaci uživatele, práci s databázemi apod.

Tato technologie je založena na architektuře klient-server. Výstupem ASP.NET aplikace je HTML stránka. ASP.NET tedy běží na serveru a reaguje na dotazy uživatele/klienta. Pro tvorbu ASP.NET aplikace je potřeba znalost především programovacího jazyku C# a značkovacího jazyku HTML.[10]

2.3.1 ASP.NET

ASP.NET je webový framework s otevřeným zdrojovým kódem, vytvořený společností Microsoft, pro vytváření moderních webových aplikací. ASP.NET rozšiřuje platformu .NET o nástroje a knihovny určené právě pro vytváření webových aplikací.[5]

2.3.2 ASP.NET Core

ASP.NET Core je open-source a multiplatformní verze ASP.NET. Tato platforma je navržena tak, aby umožnila rychlé vyvíjení runtime komponent, API, překladačů apod. a zároveň poskytovala stabilní a podporovanou platformu pro udržení běhu aplikací.

Aplikace v ASP.NET Core lze, oproti dřívější verzi APS.NET Windows-only version, vyvíjet a spouštět v systémech Windows, Linux, macOS a Docker.[6]

2.4 EF6 a EF Core

Entity Framework je Object-relational mapping (ORM)³. To znamená, že se databázové tabulky přímo mapují na C# třídy. V projektu následně pracujeme pouze s objekty a framework za nás na pozadí generuje SQL dotazy. Díky tomu je výsledná aplikace tvořena především pomocí objektů.[11]

2.4.1 Entity Framework 6

Entity Framework 6 (EF6) je ORM, primárně navržený pro .NET Framework, ale zároveň s podporou pro .NET Core. EF6 je stabilní, podporovaný produkt, ale již se aktivně nevyvíjí.[12]

2.4.2 Entity Framework Core

Entity Framework Core (EF Core) je moderní ORM pro .NET. Podporuje dotazy LINQ, sledování změn, aktualizace a migrace schématu.

EF Core pracuje s SQL Server/SQL Azure, SQLite, Azure Cosmos DB, MySQL, PostgreSQL a mnoha dalšími druhy databází.[12]

2.5 MVC

Model-View-Controller (MVC) je návrhový vzor používaný k rozdělení webové aplikace na 3 komponenty.

³Objektově-relační mapování

- Model - Práce s daty
- View - Uživatelské rozhraní
- Controller - Logická část aplikace

Pomocí vzoru MVC pro webové aplikace jsou požadavky směřovány na controller, který je zodpovědný za práci s modelem. Model provádí akce a načítá data z databáze. Controller poté zvolí zobrazení (view), které se má zobrazit, a poskytne mu model. View už pouze vykreslí stránku na základě dat získaných z modelu.[4]

2.5.1 Model

Jedná se o dynamickou datovou strukturu aplikace, nezávislou na uživatelském rozhraní. Jejím úkolem je spravovat data, pravidla a logiku aplikace.

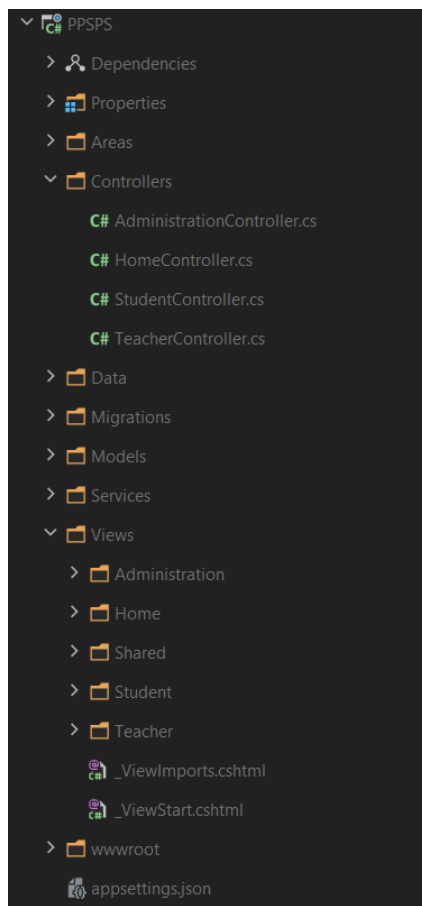
V ASP.NET Core MVC má model podobu třídy, kde veřejné metody představují položky v databázi, s kterou je třída provázána. Ukládáme ji do adresáře Models v rámci projektu. Pomocí atributů definuje pravidla, která se aplikují na klienta a server.[8]

2.5.2 View

Převádí data reprezentovaná objekty modelu do podoby vhodné k interaktivní prezentaci uživateli.[7] Může se jednat o jakoukoliv reprezentaci dat (graf, diagram, tabulka, atd.).

V ASP.NET Core se využívá syntaxe Razor. Ta poskytuje jednoduchý, čistý a lehký způsob vykreslení obsahu HTML stránek na základě view. Razor umožňuje vykreslit stránku pomocí C# a vytvářet webové stránky plně kompatibilní s HTML5.[8]

Každý controller může pro jedno view generovat vlastní obraz. Aby tedy nedocházelo ke kolizi 2 controlleru pro jedno view, vyžaduje MVC uložení view do adresáře Views v rámci projektu, a zde do podadresáře s názvem controlleru, který ho generuje (viz obr. 2.2).



Obrázek 2.2: Ukládání View v rámci projektu

2.5.3 Controller

Controller reaguje na vstup uživatele a provádí interakce s objekty datového modelu. Zjednodušeně to znamená, že řadič přijme vstup, ověří jej a následně ho předá modelu.[8]

Třída controlleru obsahuje veřejné metody označené jako Action method. V ASP.NET MVC musí každý název třídy controlleru končit klíčovým slovem "Controller". To znamená, že pro domovské stránky třídu nazýváme HomeController, pro stránky studenta StudentController, apod. Všechny tyto třídy ukládáme v rámci projektu do složky Controllers.

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

Listing 1: Controller - Action Method

2.6 MySQL

MySQL je otevřený systém řízení báze dat uplatňující relační databázový model. Jedná se o multiplatformní databázi. Komunikace s databází probíhá pomocí dotazovacího jazyka SQL.[9]

Kapitola 3

Základní struktura ASP.NET Core

Pro tuto webovou aplikaci jsem použil verzi .NET Core 3.1, ta byla vydána v prosinci roku 2019. V době, kdy jsem začínal pracovat na tomto projektu to také byla nejnovější verze.

V tomto projektu jsem dále použil ASP.NET Core Identity¹, které za mě vygenerovalo základní strukturu, se kterou jsem dále pracoval.

V poslední řadě bylo zapotřebí stáhnout přes NuGet package manager² Entity Framework Core a k němu nástroje potřebné k práci s MySQL databází, kterou v projektu využívám.

3.1 Model

Jak již bylo zmíněno v teorii, model se zabývá především prací s daty uloženými v databázi. Pro to, abych s nimi mohl efektivně pracovat, potřebuji vytvořit z jednotlivých položek databáze objekty. Všechny objekty musí být veřejné, tedy public.

Pro každou tabulku z databáze jsem si tedy vytvořil objekt (třidu) ve složce Models (viz obr. 3.1). Poté jsem si potřeboval nadefinovat každou její položku jako vlastnost tohoto objektu, kde datový typ reprezentuje způsob zápisu do databáze tzn.

- VARCHAR() - Public string NazevObjektu { get; set; }
- INT - Public int NazevObjektu { get; set; }
- DATETIME - public DateTime NazevObjektu { get; set; }

¹Rozhraní API, které podporuje funkce přihlášení uživatelského rozhraní (UI). Spravuje uživatele, hesla, data profilu, role, deklarace identity, tokeny, potvrzení e-mailu a další.[1]

²NuGet je správce balíčků, který vývojářům umožňuje sdílet opakovaně použitelný kód.

Dalším krokem bylo přiřazení atributů k jednotlivým vlastnostem. V ASP.NET Core se můžeme setkat jak s atributy, které definují vlastnosti (jedná se o primární klíč, název, formát, ...), např.

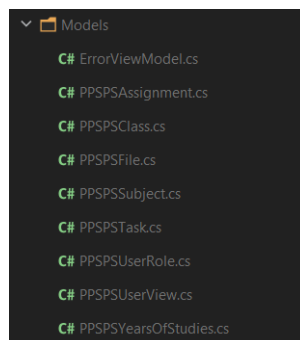
- Key - Označuje vlastnost jako primární klíč.
- Display("Název") - Název vlastnosti, která se zobrazí ve View.
- DisplayFormat(DataFormatString = "0: yyyy-MM-dd", ApplyFormatInEditMode = true) - Nastavuje formát zobrazení pro hodnotu vlastnosti.

Tak i s tzv. Atributy ověřování. Tyto atributy nám umožňují zadat pravidla pro vlastnosti objektu. V ASP.NET Core existuje několik předdefinovaných atributů, např.

- EmailAddress - Ověřuje, zda má vlastnost formát e-mailu.
- Range - Ověřuje, že hodnota vlastnosti spadá do zadaného rozsahu.
- Required - Ověří, že pole nemá hodnotu null.
- StringLength - Ověří, že hodnota vlastnosti řetězce nepřekračuje zadané omezení délky.

Atributy ověření umožňují vypsání chybové zprávy, která se zobrazí, pokud je zadán neplatný vstup do hodnoty vlastnosti. Tyto atributy je možné ověřit jak na straně klienta, tak na straně serveru.

K vytvoření modelu byla nutná již navržená databáze, tu popisuji v kapitole Databáze níže. Pro lepší představu přikládám zdrojový kód PPSPSSubject.cs, modelu popisující tabulku předmětů (viz zdrojový kód 2).



Obrázek 3.1: Složka Models

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace PPSPS.Models
{
    public class PPSPSSubject
    {
        [Key]
        [Column(TypeName = "varchar(767)")]
        public string Id { get; set; }

        [StringLength(45, ErrorMessage = "Název předmětu nesmí" +
            "mít víc, jak 45 znaků.")]
        [Display(Name = "Název předmětu")]
        [Column(TypeName = "varchar(45)")]
        public string SubjectName { get; set; }

        [StringLength(5, ErrorMessage = "Zkratka předmětu nesmí" +
            "mít víc, jak 5 znaků.")]
        [Display(Name = "Zkratka předmětu")]
        [Column(TypeName = "varchar(5)")]
        public string SubjectAbbreviation { get; set; }
    }
}

```

Listing 2: Model - Předmět

3.2 View

3.3 Controller

3.3.1 Detail

```
public async Task<IActionResult> UserOverview(string? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var user = await _context.Users
        .Include(c => c.Class)
        .AsNoTracking()
        .FirstOrDefaultAsync(u => u.Id == id);
    if (user == null)
    {
        return NotFound();
    }
    return View(user);
}
```

Listing 3: Controller - Detail

3.3.2 Editace

```
[HttpPost, ActionName("UserEdit")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> UserEdit_Post(string? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var userToUpdate = await _context.Users.FirstOrDefaultAsync(s => s.Id == id);
    if (await TryUpdateModelAsync<PPSPSUser>(
        userToUpdate,
        "",
        s => s.FirstName, s => s.LastName, s => s.Email,
        s => s.EmailConfirmed, c => c.ClassId))
    {
        try
        {
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(UsersOverview));
        }
        catch (DbUpdateException ex)
        {
            ModelState.AddModelError("", "Nebylo možné uložit změny. " +
                "Zkuste to znovu později a pokud " +
                "problém přetrvává, " +
                "obraťte se na správce systému.");
        }
    }

    PopulateClassesWithIdDropDownList(userToUpdate.ClassId);
    return View(userToUpdate);
}
```

```

public async Task<IActionResult> UserEdit(string? id)
{
    if (id == null)
    {
        return NotFound();
    }
    var user = await _context.Users.FindAsync(id);
    if (user == null)
    {
        return NotFound();
    }
    PopulateClassesWithIdDropDownList(user.ClassId);
    return View(user);
}

```

Listing 5: Controller - Editace b)

3.3.3 Smazání

```
public async Task<IActionResult> UserDelete(string? id,
bool? saveChangesError = false)
{
    if (id == null)
    {
        return NotFound();
    }

    var user = await _context.Users
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.Id == id);
    if (user == null)
    {
        return NotFound();
    }

    if (saveChangesError.GetValueOrDefault())
    {
        ViewData["ErrorMessage"] =
            "Smazání se nezdařilo. Zkuste to znovu později "
            + "a pokud problém přetrvává, " +
            "obraťte se na správce systému.";
    }

    return View(user);
}
```

Listing 6: Controller - Smazání a)

```

[HttpPost, ActionName("UserDelete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> UserDelete_Post(string? id)
{
    var user = await _context.Users.FindAsync(id);
    if (user == null)
    {
        return RedirectToAction(nameof(UsersOverview));
    }
    try
    {
        _context.Users.Remove(user);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(UsersOverview));
    }
    catch (DbUpdateException ex)
    {
        return RedirectToAction(nameof(UserDelete), new { id = id,
            saveChangesError = true });
    }
}

```

Listing 7: Controller - Smazání b)

Kapitola 4

Databáze

Kapitola 5

Závěr

Seznam tabulek

Seznam obrázků

2.1	Porovnání ASP.NET Core a ASP.NET [3]	7
2.2	Ukládání View v rámci projektu	10
3.1	Složka Models	13

Seznam zdrojových kódů

1	Controller - Action Method	11
2	Model - Předmět	14
3	Controller - Detail	15
4	Controller - Editace a)	17
5	Controller - Editace b)	18
6	Controller - Smazání a)	19
7	Controller - Smazání b)	20

Literatura

- [1] ANDERSON, R.: *Úvod do Identity ASP.NET Core*. Microsoft Corporation, 2020, [cit. 2021-4-10].
Dostupné z: <https://docs.microsoft.com/cs-cz/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio>
- [2] ANDERSON, R.; Mullen, T.; Vicarel, D.: *Razor Referenční informace k syntaxi pro ASP.NET Core*. Microsoft Corporation, 2020, [cit. 2021-4-8].
Dostupné z: <https://docs.microsoft.com/cs-cz/aspnet/core/mvc/views/razor?view=aspnetcore-5.0>
- [3] ASP.NET vs ASP.NET Core. 2017, [cit. 2021-4-4].
Dostupné z: <https://thienn.com/aspnet-vs-aspnetcore/>
- [4] Microsoft Corporation: *ASP.NET MVC Pattern*. 2021, [cit. 2021-4-5].
Dostupné z: <https://dotnet.microsoft.com/apps/aspnet/mvc>
- [5] Microsoft Corporation: *What is ASP.NET?* 2021, [cit. 2021-4-4].
Dostupné z: <https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet>
- [6] Microsoft Corporation: *What is ASP.NET Core?* 2021, [cit. 2021-4-4].
Dostupné z: <https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core>
- [7] Model–view–controller. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021, [cit. 2021-04-05].
Dostupné z: <https://cs.wikipedia.org/wiki/Model-view-controller>
- [8] Model–view–controller. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2021, [cit. 2021-04-05].
Dostupné z: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

- [9] MySQL. 2021, [cit. 2021-04-06].
Dostupné z: <https://cs.wikipedia.org/wiki/MySQL>
- [10] ČÁPKA, D.: *Lekce 1 - Úvod do ASP.NET*. 2012, [cit. 2021-4-4].
Dostupné z: <https://www.itnetwork.cz/csharp/asp-net-core/zaklady/tutorial-uvod-do-asp-dot-net>
- [11] ČÁPKA, D.: *Lekce 8 - Scaffolding a Entity Framework v ASP.NET MVC*. 2014, [cit. 2021-4-5].
Dostupné z: <https://www.itnetwork.cz/csharp/asp-net-mvc/zaklady/asp-dot-net-mvc-tutorial-scaffolding-entity-framework-editor-clanku>
- [12] VICKERS, A.; OLPROD: *Porovnání EF Core a EF6*. Microsoft Corporation, 2019, [cit. 2021-4-5].
Dostupné z: <https://docs.microsoft.com/cs-cz/ef/efcore-and-ef6/>
- [13] WAGNER, B.; OLPROD: *Prohlídka jazyka C#*. Microsoft Corporation, 2021, [cit. 2021-3-25].
Dostupné z: <https://docs.microsoft.com/cs-cz/dotnet/csharp/tour-of-csharp/>

Příloha A

Příloha