Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj aplikace pro správu IT projektů a evidenci práce v malých firmách

František Jelínek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

František Jelínek

Informatika

Název práce

Vývoj aplikace pro správu IT projektů a evidenci práce v malých firmách

Název anglicky

Development of an application for managing IT projects and recording work in small companies

---

Cíle práce

Cílem práce je vytvořit webovou aplikaci pro malé začínající IT firmy. Aplikace obsahuje přehled projektů a úkolů. V aplikaci lze zaznamenávat svůj čas strávený prací (podpora homeoffice a flexibilní pracovní doby). Vedoucí projektu má možnost prostřednictvím aplikace zadávat do projektu různé úkoly a sledovat aktivitu zaměstnanců. Aplikace bude vyvíjena jako open source, což umožní malým nebo začínajícím firmám zmírnit náklady. Aplikace bude připravena na možnosti rozšíření.

Metodika

Literární rešerše, Obecná funkčnost ASP a C#, Určení požadavků na aplikaci, Analýza podobných aplikací na trhu, Vývoj aplikace, Návrh databáze, Propojení aplikace s databází

Doporučený rozsah práce

30-40

Klíčová slova

ASP, Vývoj aplikace, Webová aplikace, Open source, Řízení projektu

---

Doporučené zdroje informací

FREEMAN, Adam. Pro ASP.NET Core MVC 2. London: Apress, 2017. ISBN 978-1-4842-3149-4.

POLAT, Engin a Stéphane BELKHERAZ. ASP.NET Core MVC 2.0 cookbook : effective ways to build modern, interactive web applications with ASP.NET Core MVC 2.0. Birmingham, Mumbai: Packt, 2018. ISBN 978-1-78588-675-1.

VOGEL, Eric. Beginning Entity Framework Core 5 : from novice to professional. New York: Apress, 2021. ISBN 978-1-4842-6882-7.

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Martin Pelikán, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 21. 6. 2023

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 11. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Vývoj aplikace pro správu IT projektů a evidenci práce v malých firmách" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 12. 3. 2024 

Poděkování

Rád bych touto cestou poděkoval Ing. Martinovi Pelikánovi, Ph.D. za poskytnuté rady a ochotu při konzultacích.

Vývoj aplikace pro správu IT projektů a evidenci práce v malých firmách

Abstrakt

Tato bakalářská práce se zabývá vývojem webové aplikace, která je určena ke správě projektů v malých začínajících IT firmách. Aplikace je vyvíjena jako dostupná alternativa k již existujícím aplikacím pro správu projektů. Je zveřejněna jako open source, takže firmy ji mohou používat bez výdajů při překročení určitého množství projektů či uživatel. Také si ji mohou moci volně rozšířit o funkce, které budou postrádat. Teoretická část popisuje znalosti, informace a technologie, které je potřeba znát k vývoji aplikace. Také je její část věnována analýze již existujících podobných aplikací. Praktická část se z počátku zabývá návrhem jak entit, tak i designu. Také je v ní ukázáno, v jakém programu byla vytvořena databáze a jaké vývojové prostředí bylo zvoleno k vývoji aplikace samotné. Další částí je napsáno, jaké balíčky byly přidány do projektu a propojení aplikace s databází. V druhé polovině praktické části je popsáno přihlašování do aplikace a dále je popsána funkčnost aplikace prolnuta jak o ukázky z kódu, tak i o ukázky z aplikace z pohledu uživatele. Popsána je zde většina akcí, které lze v aplikaci provádět. Členěno je podle role, do které uživatel patří. V závěru se bakalářská práce zabývá možným rozšířením do budoucna.

Klíčová slova: ASP, .NET, Vývoj aplikace, C#, MVC, SQL server, Webová aplikace, Open source, Řízení projektu

Development of an application for managing IT projects and recording work in small companies

Abstract

This bachelor's thesis deals with the development of a web application designed for project management in small starting IT companies. The application is being developed as an accessible alternative to existing project management applications. It is released as open source, allowing companies to use it without expenses upon exceeding a certain number of projects or users. They can also freely extend it with features they may lack. The theoretical part describes the knowledge, information, and technologies necessary for application development. It also includes an analysis of existing similar applications. The practical part initially focuses on the design of entities and design. It also demonstrates the database creation in a specific program and the development environment chosen for the application development itself. Another section describes the packages added to the project and the integration of the application with the database. In the second half of the practical part, the login to the application is described, followed by the functionality of the application interspersed with code snippets and user interface examples. Most actions that can be performed in the application are described, categorized according to the user's role. The conclusion of the bachelor's thesis discusses possible future expansions.

Keywords: ASP, .NET, Application development, C#, MVC, SQL server, Web application, Open source, Project management

Obsah

1 Úvod

V současné době se na trhu vyskytuje velké množství začínajících firem, které podnikají v oblasti IT. Ať už vyvíjí desktopové aplikace, nebo webové stránky, tak potřebují aplikaci, ve které si realizované projekty rozloží na jednotlivé úlohy. Všechny dostupné webové aplikace, které umožňují správu projektů, mají značná omezení na počet uživatelů, počet projektů a také kapacitu úložiště. Jediné možné řešení, je si aplikaci platit v předplatném, aby byly kapacity navýšeny.

Webová aplikace vyvinuta v této bakalářské práci, by tedy měla nabízet plně funkční alternativu, která je naprosto zdarma, a může být dále rozšířena o funkce, které může firma postrádat. Aplikace v současném stavu nabízí tři uživatelské role, pro CIO (Chief Information Officer) firmy, dále pro projektové manažery a poslední role je určena pro běžné zaměstnance. Projektový manažeři zakládají projekty, plní je úlohami a přidělují jednotlivým zaměstnancům přístup k projektům. Zaměstnanci si mohou z projektů, do kterých mají přístup, vyberou ze seznamu úloh tu, kterou mají zájem zpracovat. Aplikace umožňuje zaměstnanci sledovat čas, který nad danou úlohou strávil. Zaměstnanec má pak přístup k výpisu práce, kterou odvedl, a také kolik by za ni měl dostat zaplaceno. CIO firmy má přístup ke všem uživatelům, jejich výpisům z práce na úlohách a také má přístup ke správě všech projektů. Hesla všech uživatelů jsou ukládána v hashované podobě.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvořit webovou aplikaci pro malé začínající IT firmy. Aplikace by měla obsahovat přehled projektů a úkolů. V aplikaci bude možné zaznamenávat svůj čas strávený prací (podpora homeoffice a flexibilní pracovní doby). Vedoucí projektu bude moci prostřednictvím aplikace zadávat do projektu různé úkoly a sledovat aktivitu zaměstnanců. Aplikace bude vyvíjena jako open source, což umožní malým nebo začínajícím firmám zmírnit náklady. Aplikace bude připravena na možnosti rozšíření.

2.2 Metodika

Teoretická část bakalářské práce popisuje obecnou funkčnost objektivně orientovaného programování. Dále se zaměřuje na framework .NET a programovací jazyk C#. V teoretické části je také popsána architektura MVC aplikací a HTTP protokol. Další částí jsou diagramy, které bývají tvořeny před začátkem vývoje a také typy grafických návrhů aplikací. Ke konci teoretické části je popsán SQL server, IDE pro vývoj webových aplikací a pro přístup a manipulaci s SQL serverem. Na samotném konci je popsána základní používaná metodologie projektového řízení a také analýza funkcí dvou dostupných webových aplikací pro řízení projektů.

Praktická část je věnována vytvoření databáze a entit na SQL serveru. Dále je popsán postup pro založení projektu ve Visual studiu, a jeho propojení s databází na SQL severu. V závěru vlastní práce jsou ukázky z vytvořené aplikace. Součástí praktické části jsou také obrázky obsahující kód, který je zde i popsán. Většina kódu je psána v jazyce C#. Stránky jsou vykreslovány pomocí HTML s prvky ASP. Ke stylování byl využit Boostrap, v malém množství i vlastní CSS. K zprovoznění funkce pro zaznamenávání času stráveného při práci bylo využito Java Scriptu.

Po zpracování aplikace je její kód nahrán do veřejného repozitáře, a tím zpřístupněn k volnému stahování a úpravám.

3 Teoretická východiska

3.1 Objektově orientované programování

Objektově orientované programování (OOP) je specifické programovací paradigma, které odlišuje programování od původního imperativního přístupu. OOP v dnešní době tvoří základ velkého množství současných programovacích jazyků. Kód se více propojuje s daty, což usnadňuje přenos dat mezi objekty.

Prvky programu, jsou seskupovány do entit, ty nazýváme objekty. Objekt je souborem metod a atributů. Metody popisují chování daného objektu, jsou zpětně použitelné a zapouzdřené uvnitř objektu. Atributy ukládají data objektů, přístup k nim může být modifikován.

Základními principy jsou zapouzdření, dědičnost, polymorfismus nebo abstrakce. [1][2]

3.1.1 Zapouzdření

Tento princip je využíván k zamezení přímého přístupu k atributům objektu. Objekty tedy nemohou přímo přistupovat k atributům jiných objektů. Přistupovat lze pouze k metodám a atributům, které jsou označeny jako veřejné. [1][3]

3.1.2 Dědičnost

Dědičnost v OOP umožňuje vytvářet nové třídy (podtřídy) na základě existující třídy (nadtřídy), umožňuje znovupoužití kódu a implementaci obecných vlastností. Podtřídy zdědí vlastnosti a metody nadtřídy, které lze následně rozšiřovat nebo upravovat podle konkrétních potřeb. Pomocí dědičnosti je vytvářena hierarchie tříd, která zlepšuje organizaci kódu. Může být využita pro implementaci polymorfismu, kdy můžeme pracovat s objekty různých podtříd stejným způsobem. Dědičnost také podporuje princip abstrakce, což umožňuje oddělit obecné koncepty a rozhraní od konkrétní implementace. [2][3]

3.1.3 **Polymorfismus**

Polymorfismus umožňuje pracovat s různými objekty pomocí stejných rozhraní nebo abstraktních tříd. Dále umožňuje třídám různých podtříd provádět stejné operace rozdílnými způsoby. Polymorfismus je základem pro využití polymorfních funkcí a virtuálních metod, tím lze dosáhnout dynamického určení volání metod během běhu programu na základě skutečného typu objektu. [1][3]

3.2 **.NET**

Je framework vyvinutý společností Microsoft. Jedná se o komplexní sadu nástrojů, knihoven, a runtime prostředí, které jsou využívány při vývoji a provozu rozmanitých aplikací a služeb. Dále nabízí nástroje jak pro tvorbu webových aplikací, desktopových aplikací, mobilních aplikací tak i pro vývoj cloudových služeb a mnoho dalšího.

V současné době je hlavní implementací .NET Core, lze jej označit za nástupce implementace .NET Framework je stále podporován, jelikož jej využívá mnoho současných aplikací, ale již není rozšiřován o nové funkce. Na rozdíl od svého předchůdce je. NET Core multiplatformní a je vyvíjen jako open-source.

Nejrozšířenějším programovacím jazykem pro platformu .NET je C#. Jedná se o velmi objektově orientovaný programovací jazyk s automatickou správou paměti. Podporuje zpracování výjimek (exceptions), lambda výrazy, reflexi, asynchronní operace nebo LINQ výrazy. Všechen zdrojový kód je organizován do tříd, které jsou součástí jednoho jmenného prostoru, který nazýváme namespace. Jedná se tedy o objektově orientovaný přístup. [4][5]

3.2.1 **Asynchronní volání**

Asynchronní volání se v jazyce C# obvykle provádí pomocí klíčového slova „async“ a „await“. Když označíme metodu jako „async“, umožníme jí tím běžet asynchronně. To znamená, že neblokuje hlavní vlákno a umožňuje jiným úlohám pokračovat ve vykonávání své činnosti. Asynchronní volání umožní objektu po zavolání metody provádět další operace. Označením „await“, čekáme na data, která byla zpracována asynchronně. [6][7]

3.2.2 **LINQ**

LINQ umožňuje psát dotazy nad různými zdroji dat přímo v kódu. Poskytuje jednotný způsob psaní dotazů nad kolekcemi objektů, databázemi, XML dokumenty nebo jinými zdroji dat. Pomocí LINQ lze také psát dotazy podobné SQL (Structured Query Language), tím zjednodušuje práci s daty v jazyce C#. LINQ provádí vyhodnocování dotazů až v okamžiku, kdy jsou výsledky skutečně potřeba. To znamená, že se minimalizuje zátěž na výpočetní výkon, protože se neprovádějí zbytečné výpočty. [8][9]

3.2.3 **Entity framework**

Entity Framework je technologie vývoje softwaru pro platformu .NET, která umožňuje pracovat s daty v databázích pomocí objektově orientovaných konceptů. Slouží jako objektově-relační mapování (ORM), to znamená, že zprostředkovává komunikaci mezi objekty v programu a databázovým modelem. Tím eliminuje potřebu psát přímé SQL dotazy a umožňuje vývojářům pracovat s daty pomocí objektového paradigmatu, což zvyšuje čitelnost kódu.

Entity Framework nabízí více možných postupů Database-First a Code-First. Model-First začínáme s vizuálním návrhem datového modelu, ze kterého se následně generuje databáze. Přístupem Database-First počítáme s již existující databází včetně tabulek, z nichž jsou vygenerovány entitní třídy (objekty, které mají stejnou strukturu, jako tabulka v databázi) i s databázovým kontextem.

Změny v Entity Framework jsou realizovány pomocí migrací. Migrace obsahuje změny, které je potřebné provést z důvodu, aby se model nacházel ve stejném stavu jako databáze. S databází pracujeme pomocí instance třídy v databázovém kontextu, ve kterém jsou tabulky reprezentovány kolekcemi entitních tříd (v jazyce C# se tyto kolekce označují jako DbSet). K dotazování na tyto kolekce lze využít výše zmíněné LINQ výrazy. [10][11]

3.3 Web API

API (Application Programming Interface) zprostředkovává komunikaci mezi klientem a serverem, za klienta si představíme webový prohlížeč. Existuje více způsobů, jak tuto komunikaci zařídit, např. REST, SOAP a RPC.

3.3.1 REST API

REST (Representational State Transfer) pochází od Roye Fieldinga, jedná se o API určené k přenosu dat mezi webovými systémy. Koncept není vázaný na žádný určený protokol ani na formát přenášených dat. V současnosti je nejpoužívanějším způsobem implementace REST API s využitím HTTP protokolu a přenos dat ve formátu JSON. [12]

3.3.2 SOAP API

SOAP (Simple Object Access Protocol) API je standardní způsob komunikace mezi systémy a aplikacemi. Používá XML, jako formát pro výměnu strukturovaných informací. SOAP API umožňuje aplikacím v různých operačních systémech a v jiných programovacích jazycích vzájemnou komunikaci. SOAP API je vyvíjený strukturovaně a formálně. Na rozdíl od REST, který je více flexibilní, SOAP poskytuje přesnější pravidla pro strukturu požadavků a odpovědí, ovšem může být pomalejší oproti REST. [13]

3.4 ASP .NET MVC

ASP.NET MVC je framework vyvinutý společností Microsoft a je určený pro vývoj webových aplikací. Postavený je na architektonickém vzoru Model-View-Controller.

Model reprezentuje datovou vrstvu aplikace (znázorňuje jednotlivé entity), View zobrazuje uživatelské rozhraní a Controller zpracovává uživatelské požadavky. Oddělení těchto oblastí je základním principem tohoto frameworku. Podporuje koncept routování, který mapuje URL adresy na akce v kontrolerech a umožňuje různé typy výstupů, včetně HTML stránek a JSONu.

S velmi rozmanitou sadou nástrojů pro validaci dat, správu relací, autorizaci a další úkoly, poskytuje efektivní prostředí pro vývoj. Nejčastěji je vyvíjeno ve Visual Studiu a platformou .NET Framework nebo v dnešní době spíše .NET Core.

Díky otevřené architektuře je snadné integrovat další knihovny a rozšíření. Podporuje moderní technologie jako AJAX, HTML5 a CSS3. Umožňuje práci s daty v

databázi pomocí již zmíněného balíčku Entity Framework. Možnost vytvářet jednotkové (Unit) testy a integrační testy pomáhá zajišťovat funkčnost kódu. [14][15]

3.5 HTTP protokol

HTTP (Hypertext Transfer Protocol) je síťový protokol na úrovni aplikační vrstvy, který je používán při komunikaci v síťovém modelu klient-server. Klient pracuje aktivně, tudíž zasílá požadavky serveru a ten odesílá zpět odpovědi. Komunikace je tedy vedena bezstatově, jelikož server neudržuje žádný stav mezi dvěma požadavky.

HTTP požadavek je tvořen čtyřmi podstatnými částmi. První částí je identifikátor cílového zdroje na serveru. Jako druhá je uvedena metoda požadavku, ta specifikuje akci, která má být provedena. Další částí požadavku je hlavička, ve které jsou uvedeny informace o požadavku. Posledním prvkem je tělo požadavku, které může být i prázdné. [16]

3.5.1 Metody a požadavky

GET je jedna z nejčastěji používaných HTTP metod, tvoří základ pro prohlížení webu a získávání dat z různých webových zdrojů. Používá se zejména pro získání dat ze serveru, a to prostřednictvím URL (Uniform Resource Locator). Požadavky mohou obsahovat parametry v URL, které umožňují klientovi specifikovat, jaké konkrétní data chce získat. Parametry jsou obvykle přidány za otazníkem v URL (např. ?parametr=hodnota) a mohou být použity k filtrování nebo třídění dat.

POST je metoda, která se používá k odesílání dat na webový server. Odesílat na server můžeme textové, binární nebo strukturovaná data. Tato data mohou být v těle HTTP požadavku a jsou často používána pro vytváření nebo aktualizaci dat na serveru.

PUT je používána k aktualizaci existujícího záznamu na webovém serveru nebo k vytvoření zdroje, pokud neexistuje. Tato metoda umožňuje klientovi odeslat data na server a specifikovat, kam by tato data měla být uložena. Klient odešle data, která by měla nahradit existující data na daném URL. Data, která mají být uložena nebo aktualizována, jsou umístěna v těle HTTP požadavku.

PATCH se používá k částečné aktualizaci dat na serveru. To znamená, že klient může odeslat pouze ty části dat, které se mají změnit, aniž by bylo nutné poslat celý zdroj. Tím se snižuje zátěž na síť a zvyšuje efektivita přenosu dat.

DELETE slouží k odstranění existujícího záznamu na webovém serveru. Tato metoda umožňuje klientovi požádat server, aby odstranil určitý záznam na základě URL. [17]

3.5.2 **Stavové kódy**

Odpověď serveru také obsahuje hlavičku a tělo, kromě toho je také odeslán i stavový kód. Stavový kód je tříciferné číslo a říká nám výsledek zpracování požadavku.

- Stavové kódy začínající číslem 1 značí informační kód;
- Stavové kódy začínající číslem 2 označují úspěch požadavku (200 OK);
- Stavové kódy začínající číslem 3 se používají k přesměrování;
- Stavové kódy začínající číslem 4 se používají pro chyby způsobené uživatelem (401 Unauthorized, 404 Not Found);
- Stavové kódy začínající číslem 5 značí chybu na straně serveru (500 Internal Server Error). [18]

3.6 **Diagramy**

Diagramy jsou využívány k návrhům, nebo dokumentacím různých softwarových systémů. Existuje několik druhů diagramů, každý slouží k znázornění jiné části systému.

3.6.1 **UML**

UML (Unified Modeling Language) je soubor grafických notací používaný pro vizualizaci, návrh a dokumentaci softwarových systémů. UML diagramy poskytují grafický náhled na různé aspekty systému a usnadňují porozumění jeho struktury a chování. Existuje několik typů UML diagramů, dva hlavní typy jsou diagram struktury a diagram chování.

Class diagramy zachycují strukturu systému a vztahy mezi třídami, atributy a jejich metody. Use case diagramy popisují funkční požadavky systému a interakce mezi aktéry a systémem. Sekvenční diagramy ukazují postupnost interakcí mezi objekty nebo aktéry v čase. Stavové diagramy modelují chování systému a jeho přechody mezi různými stavy.

UML diagramy jsou používány v procesu softwarového inženýrství od fáze analýzy a návrhu, až po implementaci a údržbu. Pomáhají komunikovat myšlenky mezi členy týmu, zákazníky.

Poskytuje standardizovanou a srozumitelnou notaci pro modelování nových systémů. Použití UML diagramů přispívá k lepší dokumentaci a správě softwarových projektů. [19][20]

3.6.2 Vývojový diagram

Vývojové diagramy jsou grafické reprezentace procesů, postupů nebo toku dat v systému. Tyto diagramy umožňují zobrazení kroků a interakcí v procesu. Každý prvek ve vývojovém diagramu, jako jsou akce, rozhodovací body, vstupy a výstupy, je reprezentován symboly nebo tvary, které jsou propojeny šipkami znázorňujícími tok dat nebo postupu. [21][22]

3.6.3 ERD diagramy

Entity-Relationship Diagrams (ERD) jsou využívané pro modelování datových struktur a vztahů v databázových systémech. Tyto diagramy pomáhají organizovat a reprezentovat entity a vztahy mezi nimi v databázovém schématu.

Základními složkami ERD jsou entity, atributy a vazby. Entity jsou objekty, které uchovávají data v databázi, jako jsou například lidé, místa, události nebo koncepty. Atributy jsou vlastnosti entit, které popisují různé aspekty entity, například jméno, datum narození nebo adresa.

Vazby určují spojení mezi jednotlivými entitami a označují, jak jsou jednotlivé entity propojeny. Existují různé typy vazeb v ERD, jako je jeden k jednomu (1:1), jeden ku mnoha (1:N) a mnoho k mnoha (M:N).

ERD diagramy mohou být vyjádřeny pomocí různých notací, včetně Chenova notace a Crow's Foot notace. Chenova notace používá k vyjádření entit obdélníky a k vyjádření vztahů čáry spojující entity, zatímco Crow's Foot notace používá symboly pro označení kardinality vztahů. [23][24]

3.7 Grafický návrh webů a aplikací

Grafické návrhy webů a aplikací jsou tvořeny před samotným vývojem, aby bylo předem známé, jak by měl výsledný produkt vypadat.

3.7.1 **Wireframe**

Wireframe je náčrt webové stránky, mobilní aplikace nebo jiného uživatelského rozhraní, který představuje strukturu a rozložení prvků bez detailního designu nebo obsahu. Náčrty jsou často vytvářeny na začátku procesu návrhu jako součást UX (user experience) a UI (uživatelského rozhraní).

Zahrnuje základní prvky, jako jsou textová pole, tlačítka, obrázky, menu a další, aby ilustroval strukturu a organizaci informací na stránce. Oproti prototypům nebo finálním designům jsou wireframy jednodušší a zaměřují se spíše na funkčnost a rozvržení než na vizuální estetiku.

Používání wireframů umožňuje designérům a vývojářům lépe porozumět požadavkům klienta a nalézt optimální řešení uživatelského rozhraní. Existují různé typy wireframů, včetně low-fidelity wireframů s hrubým rozložením prvků a high-fidelity wireframů s většími detaily a přesnějším vyobrazením.

Wireframy mohou být vytvářeny ručně na papíře nebo pomocí specializovaného software pro návrh uživatelských rozhraní. [25][26]

3.7.2 **UI Design**

UI (User Interface) Design je zaměřený na vytváření uživatelských rozhraní, která jsou intuitivní, efektivní a příjemná pro uživatele k interakci s nimi. Cílem UI designu je vytvořit prostředí, ve kterém uživatelé mohou snadno navigovat a plnit své úkoly.

UI design zahrnuje různé prvky, jako jsou tlačítka, menu, formuláře, ikony, textová pole a další, které jsou používány k interakci uživatele s aplikací nebo webovým stránkami. Důležitou součástí UI designu je také vizuální hierarchie, která pomáhá uživatelům porozumět struktuře obsahu a prioritám na stránce.

Barevná schémata, typografie a grafika jsou také klíčové prvky UI designu, které ovlivňují celkový vzhled a dojem uživatele. UI design musí být konzistentní napříč celou aplikací či webem.

Testování uživatelského rozhraní je nezbytnou součástí UI designu, která pomáhá identifikovat problémy a zlepšit uživatelský zážitek. [27][28]

3.8 SQL

SQL (Structured Query Language) je standardizovaný strukturovaný dotazovací jazyk, používaný pro správu relačních databázi a manipulaci s jejich daty. Umožňuje vytváření, editaci, upravování a mazání dat v databázi prostřednictvím dotazů.

Základními operacemi v jazyce SQL jsou SELECT, INSERT, UPDATE a DELETE, které umožňují provádět různé úkony v databázi. SELECT slouží k získání dat z tabulek, INSERT k vkládání nových záznamů do tabulek, UPDATE k aktualizaci existujících záznamů a DELETE k mazání záznamů.

SQL umožňuje také definovat strukturu databáze pomocí CREATE, ALTER a DROP příkazů pro vytváření, změnu a mazání tabulek, indexů, pohledů a dalších objektů.

Jedná se o deklarativní jazyk, což znamená, že programátor specifikuje, co chce provést, aniž by musel určovat, jak to provést. Existuje několik dialektů SQL, včetně T-SQL (Transact-SQL), PL/SQL (Procedural Language/Structured Query Language) a MySQL, které se používají v různých databázových systémech. [29][30]

3.8.1 SQL server

SQL Server je relační databázový systém vyvinutý společností Microsoft, který poskytuje spolehlivé prostředí pro ukládání dat a manipulaci s daty. SQL Server nabízí řadu funkcí, včetně transakční integrity, podpory pro více verzí a zálohování či obnovení dat.

SQL Server je kompatibilní s různými operačními systémy, včetně Windows a Linux, což umožňuje širokou škálu nasazení. Jeho architektura umožňuje horizontální i vertikální škálovatelnost, což znamená, že může efektivně zpracovávat zátěž od menších až po velké podnikové aplikace.

Nabízí také pokročilé funkce, jako jsou datové služby, analýza služeb a reporting služby, což umožňuje podporu business intelligence a analytických potřeb organizace. Jeho integrace s nástroji jako je Visual Studio a Azure umožňuje snadný vývoj, nasazení a správu aplikací. [29][30]

3.8.2 DML

SQL DML (Data Manipulation Language) je částí SQL, přes kterou manipulujeme s daty v databázích. Obsahuje základní příkazy pro vkládání, aktualizaci a mazání dat v tabulkách.

Důležitým příkazem DML je INSERT, který umožňuje vkládat nové záznamy do tabulky s určenými hodnotami. Příkaz UPDATE slouží k aktualizaci existujících záznamů v tabulce na základě specifikovaných podmínek. DELETE se používá k mazání záznamů z tabulky podle určených kritérií. Pomocí SELECT lze získat data z tabulek na základě různých kritérií a podmínek.

Poskytuje také možnosti pro vkládání, aktualizaci a mazání dat pomocí poddotazů a spojení (JOIN) mezi tabulkami. Další důležitou částí SQL jsou klauzule, které umožňují filtrovat, řadit a seskupovat data, jimiž jsou WHERE, ORDER BY, GROUP BY a HAVING. [29][31]

3.8.3 **DLL**

SQL DLL (Data Definition Language) je částí SQL, která umožňuje definovat strukturu databáze a datových objektů. Obsahuje příkazy pro vytváření, modifikací a mazání databázových objektů například tabulek, indexů, pohledů a procedur.

Klíčovým příkazem DLL je CREATE, který umožňuje vytvářet nové databázové objekty podle určených specifikací. ALTER se používá k modifikaci existujících databázových objektů, jako příklad můžeme uvést přidání nového sloupce do tabulky nebo změnu datového typu. DROP slouží k odstranění databázových objektů, jako jsou tabulky, indexy nebo pohledy, ze systému.

Pomocí DLL lze definovat primární a cizí klíče, které slouží k zachování integrity dat v databázi. Tato část je důležitá pro správu a návrh struktury databáze v rámci vývoje softwaru a správy informačních systémů. [29][31]

3.8.4 **T-SQL**

T-SQL (Transact-SQL) je rozšířením standardního SQL, které poskytuje pokročilé funkce a procedury pro práci s databázemi v produktech Microsoft SQL Server. Jedná se o vysoce výkonný dotazovací jazyk, který umožňuje vytvářet složité dotazy, procedury, funkce a triggerové události.

Poskytuje širokou škálu funkcí pro manipulaci s daty, včetně agregačních funkcí (SUM, COUNT, MIN, MAX), poddotazů, a pracování s řetězci a datumy.

Přes T-SQL lze také spravovat databázové objekty, vytvářet indexy, zálohovat či obnovovat databáze a provádět další správní úkoly. T-SQL podporuje transakce, což

umožňuje provádět sadu operací jako jednotnou jednotku, která je buď kompletně provedena, nebo kompletně zrušena. [29][32]

3.9 Vývojová prostředí

Vývojové prostředí (IDE) je software, který ke své práci využívají programátoři při vývoji různých aplikací. Obsahují editor zdrojového kódu, kompilátor a debugger. Kompilátor se stará o překlad a spuštění kódu. Pomocí debuggeru můžeme ladit a hledat chyby v kódu.

3.9.1 Microsoft SQL server Management studio

SQL Management Studio je nástroj od společnosti Microsoft, který slouží ke správě a údržbě databázových systémů SQL serveru. Jedná se o software, který poskytuje uživatelům grafické prostředí pro správu databází, tvorbu dotazů, správu bezpečnosti a monitorování výkonu. Uživatelé mohou využívat různé nástroje a funkce například: generátory skriptů, vizuální návrhy dotazů a správa replikace dat. Pomocí SQL Management Studia jsou uživatelé schopni spravovat a optimalizovat výkon SQL Server databází. [33]

3.9.2 Microsoft Visual studio

Visual Studio je integrované vývojové prostředí (IDE) vyvinuté společností Microsoft, poskytuje širokou škálu nástrojů pro vývoj softwaru. Tento software podporuje různé programovací jazyky, včetně C#, C++, F#, Python a dalších. Visual Studio obsahuje funkce jako je syntaxové zvýraznění, ladění kódu a tvorba grafických uživatelských rozhraní.

Díky své modulární architektuře, umožňuje Visual Studio integraci s různými verzovacími systémy (například Git), testovacími nástroji a cloudovými službami. [2][34]

3.9.3 Microsoft Visual studio code

Visual Studio Code je bezplatný open-source textový editor vyvinutý společností Microsoft. Tento editor poskytuje velké množství funkcí pro vývoj softwaru, včetně syntaxového zvýraznění, automatického doplňování kódu a integrace s verzovacími systémy.

Visual Studio Code je multiplatformní a podporuje vývoj v různých programovacích jazycích, včetně JavaScriptu, TypeScriptu, Pythonu a C#. Visual Studio Code také umožňuje rozšíření pomocí různých doplňků a rozšíření, které přizpůsobují funkčnost editoru podle potřeb vývojářů. [34]

3.10 **Projektové řízení**

Projektové řízení je uplatňování procesů, metod, dovedností, znalostí a zkušeností s cílem dosáhnout konkrétních cílů projektu v souladu s kritérii akceptace projektu v dohodnutých parametrech. V rámci projektového řízení vznikají definované konečné výstupy, které jsou striktně omezeny časovým rámcem a rozpočtem. Projektové řízení má konečný výsledek a omezený časový rámec na rozdíl od pouhého řízení. Z tohoto důvodu potřebuje projektový profesionál široké spektrum dovedností; často technické dovednosti a určité dovednosti v řízení lidí.

Projekt představuje jedinečný, dočasný úkol, podnikaný za účelem dosažení plánovaných cílů, které mohou být stanoveny ve formě výstupů, výsledků nebo přínosů. Projekt bývá označen za úspěšný, pokud dosáhne stanovených cílů v souladu s kritérii akceptace, v předem dohodnutém časovém rámci a rozpočtu. Čas, náklady a kvalita představují základní pilíře každého projektu. Projektový management zahrnuje několik klíčových prvků. Jedním z nich je definování důvodu, proč je projekt nezbytný. Další fáze zahrnuje zachycení požadavků na projekt, specifikaci kvality výstupů, odhadování zdrojů a časových rámců. Součástí procesu je také příprava obchodního plánu, který slouží k odůvodnění investice do projektu. [35]

3.11 **Metodologie projektového řízení**

Metodologie projektového řízení představuje soubor zásad, nástrojů a technik, které se využívají k plánování, provedení a řízení projektů. [36]

3.11.1 **Vodopád**

Jedná se o sekvenční vývojový proces, který prochází všemi fázemi projektu (analýza, návrh, vývoj, testování). Je nejlineárnější a nejtradičnější přístup k řízení. Model využívá postupy, které vyžadují úspěšné dokončení předchozí fáze při přechodu na následující.

Metodologie se provádí chronologicky daným procesem a na základě pevných dat funguje. Výhodou metodologie vodopádu je přímočarost a definovanost. Další výhodou je zpozorování chyby již během analýzy a návrhu.

Naopak mezi nevýhody patří doba realizace projektů, jelikož s chronologickým přístupem může být časový interval delší. Taktéž v této metodologii za nevýhodu považujeme Deadline creep – čekání na předchozí úspěšnou fázi. [36][37]

3.11.2 **Agilní**

Je inkrementální a nelineární přístup k projektovému řízení. Práce touto metodou je rychlá a lze ji optimalizovat v průběhu procesu. Nabízí dynamický způsob práce a je vhodná při vývoji produktů a softwarů.

Výhody Agilní metodologie jsou flexibilita, rychlé řešení problémů a lepší jejich funkčnost. Mezi nevýhody patří méně předvídatelné výsledky. [36][38]

3.12 **Analýza dostupných webových aplikací pro správu projektů**

Pro určení požadavků na aplikaci, bylo potřeba provést anýzu webových aplikací, které si již touto problematikou zabývají. Byly proto vybrány dvě aplikace, které jsou k těmto účelům běžně v praxi používány.

3.12.1 **Freelo**

Jedná se o webovou aplikaci, ve které je možné zakládat projekty, ve kterých se dále tvoří úkoly, jež lze případně přiřazovat jednotlivým uživatelům. Každému novému úkolu lze zadat název, popis a je možné rovnou přiřadit jeho řešitele. Řešitel úkolu může být i nezvolený a v tomto případě si každá osoba s přístupem k projektu může stát jeho řešitelem. Jako uživatel s přístupem do projektu, je vidět na kolik času bylo odpracováno a kolik peněz by podle toho případně mělo být vyplaceno. Hodinová částka za práci se určuje na uživatele, nikoliv na úlohu. V daném projektu jsou také na výběr možnosti pro vypsání práce odvetné, a lze tisknout rovnou z aplikace faktury. V aplikaci se dále nachází kalendář pro sledování důležitých dat. Všechny informace jsou ukryté za přihlášením. Aplikace dále zaznamenává čas, který uživatel tráví práci na úkolu.

Za omezení lze považovat, zpoplatnění aplikace při potřebě většího množství uživatel, nebo projektů. V bezplatné verzi, může mít uživatel pouze tři aktivní projekty současně. Co se týče omezení uživatel přiřazených k projektu, tak ve verzi zdarma lze mít

k jednomu projektu přiřazené pouze tři další uživatele. Dostupná kapacita databáze je pro jeden projekt omezena na 500MB. [39]

3.12.2 **Jira**

V této aplikaci je při zakládání nového projektu na výběr z mnoha šablon, podle typu projektu, který chceme realizovat. V detailu projektu, je pak možnost do tzv. TO DO seznamu přidávat úkoly, které je potřeba splnit. Pro přístup do projektu, do něj uživatel musí být pozván. Možnou velkou výhodou je spárování s aplikací pro sdílení a verzování souborů, jako například GitHub.

Omezení na bezplatnou verzi se vztahuje především na počet uživatelů. V bezplatné verzi lze do jednoho projektu přiřadit maximálně 10 lidí. Další věcí, která chybí v bezplatné verzi jsou role pro uživatele, které zde nejdou měnit, a uživatelé, které přidáme tak mají pouze jedno roli. Úložiště je omezeno na 2GB. [40]

4 Vlastní práce

4.1 Funkční požadavky

Webová aplikace má všechen svůj obsah skrytý za přihlašovacím formulářem. Každý uživatelský účet má svoji roli. Role definuje, kam může daný uživatel přistupovat, a na které akce má oprávnění. Uživatelé se přihlašují pomocí přihlašovacího jména a hesla. Heslo se do databáze ukládá v zahashované podobě. Role rozdělujeme tři, první má nad aplikací úplnou kontrolu, druhá má kontrolu pouze nad určitými částmi aplikace a poslední role může aplikaci pouze využívat bez jakýchkoliv administrátorských úkonů.

Aplikace je členěna do tří základních sekcí. První sekcí je správa uživatelů na úrovni aplikace. Přístup do ní, je určen pouze pro nejvyšší roli. Je zde k nalezení seznam všech uživatelů, kteří mají do aplikace přístup. Dále se zde nachází možnost nového uživatele přidat, současného uživatele odebrat, nebo se podívat na výpis práce vybraného uživatele.

V další sekcí jsou projekty. Zde se liší možnosti operací, které zde lze provádět podle role přihlášeného uživatele. Role nejvyšší si může prohlížet všechny projekty a má nad nimi plnou kontrolu. Role druhá, se středními pravomocemi, si může mít kontrolu pouze nad projekty, které daný uživatel sám vytvořil. Také může být přidán do cizích projektů, ale v nich si může obsah procházet pouze jako normální uživatel. Nejnižší role poskytuje možnost procházet pouze projekty, do kterých byl uživatel s touto rolí přidán. Na detailu projektu si jeho zakladatel, nebo uživatel s nejvyšší rolí, můžou procházet seznam lidí s přístupem do projektu, můžou stávajícím členů přístup odepřít, novým přístup udělit a také si vypsat pro uživatele seznam odvedené práce na daném projektu. Každý s přístupem do projektu si zde také může zobrazit seznam úloh, které je potřeba dokončit. Když uživatel začne na úloze pracovat, pustí si záznam času, který bude počítat uživatelův čas strávený prací na vybrané úloze. Po jeho dokončení uvidí čas, jak on tak i jemu nadřízené role.

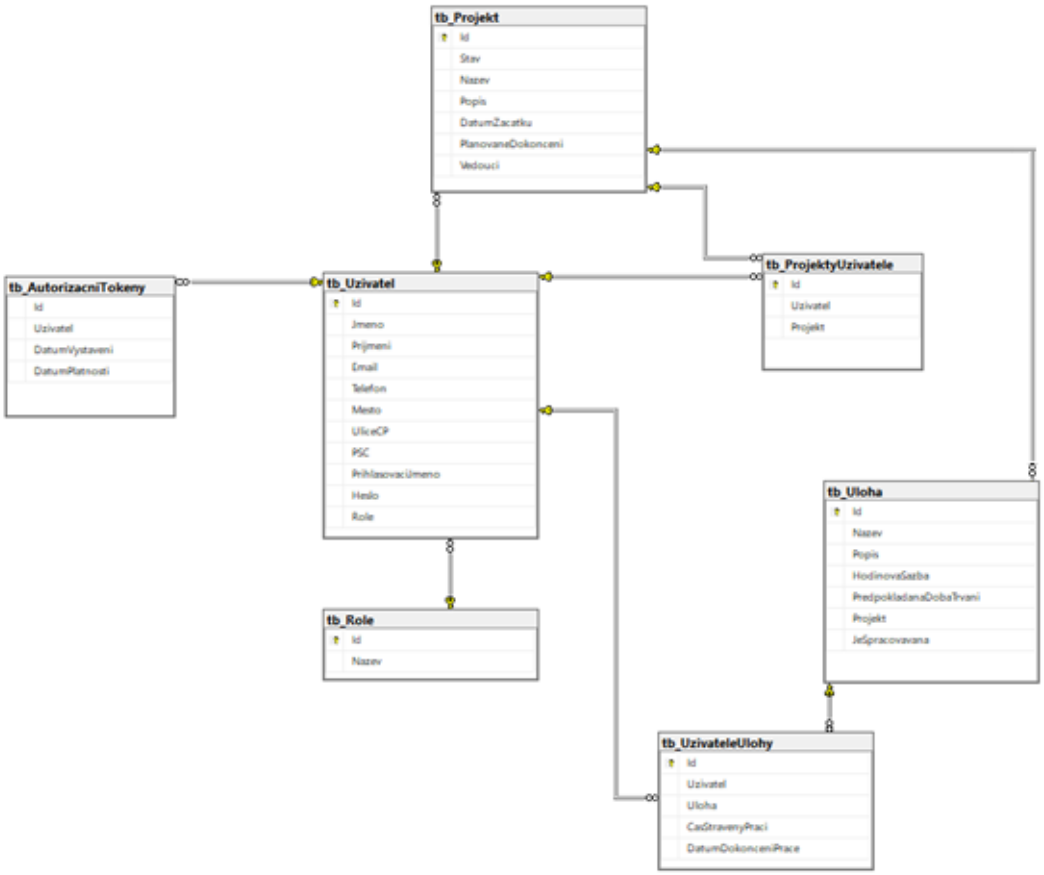
Poslední sekcí je Můj profil, zde najde každý uživatel svoje osobní informace, které si může podle potřeby měnit. Také je zde možnost změny hesla, a výpis práce, kterou odvedl. Všechny výpisy práce mají filtrování na období, případně i na vybraný projekt. Počítá se čas celkem strávený prací podle vybraných parametrů a také ukazují, Kolik by měl uživatel dostat vyplaceno za daný časový interval, nebo za vybraný projekt.

4.2 Nefunkční požadavky

Webová aplikace je vytvářena pouze pro počítače, kvůli budoucím rozšířením, na které bude potřeba, aby aplikace běžela na stejném zařízení, na kterém bude uživatel provádět svoji pracovní činnost. Aplikace by měla být kompatibilní se všemi aktuálními webovými prohlížeči. Zabezpečení je řešeno autorizačním tokenem, který se ukládá do databáze, a odkaz na něj je ukládán do session webového prohlížeče. Obsahy všech stránek jsou zabezpečeny již na úrovni serveru, tudíž pokud uživatel nemá dostatečná práva, nebo dokonce ani uživatelský účet, přijde mu jako odpověď ze serveru vždy jen přihlašovací formulář, nebo zůstane na původní stránce. Pro práci s databází jsou hojně používány výrazy LINQ, kvůli zjednodušení práce s daty.

4.3 Vytvoření databáze

Pro vytvoření entit byl použit postup database-first. Na začátku byl vytvořen ERD diagram pro zobrazení entit a jejich vazeb. Diagram můžeme vidět na obrázku níže.



Obrázek 1 - ERD diagram

4.3.1 Vytvoření databáze

Z webových stránek společnosti Microsoft, byla stažena verze SQL serveru, jedná se o verzi SQL Server 2022 Express. Po nainstalování serveru, na něm byla vytvořena databáze. Program, který byl použit pro práci se serverem se jmenuje Microsoft SQL server management studio 18. Po vytvoření databáze následovalo vytvoření tabulek. Každá tabulka má svůj unikátní identifikátor (atribut Id). Na tento sloupec lze nastavit automatické číslování, aby byla zajištěna unikátnost záznamu. Tento sloupec je také nastaven jako primární klíč. Jako ukázka vytvoření databáze, poslouží tabulka určená pro projekt. Celá tabulka je na obrázku níže.

```
CREATE TABLE tb_Projekt(
    Id int IDENTITY(1,1) primary key,
    Stav varchar(20) NOT NULL,
    Nazev varchar(100) NOT NULL,
    Popis varchar(max),
    DatumZacatku datetime NOT NULL,
    PlanovaneDokonceni datetime,
    Vedouci int NOT NULL
)
```

Obrázek 2 - Vytvoření tabulky tb_Projekt

Příkaz CREATE TABLE slouží k vytvoření nové tabulky, hned za příkaz píšeme název tabulky a do kulatých závorek pak píšeme, jaké bude mít atributy. IDENTITY(1,1) zajišťuje, že atribut bude mít výše zmíněné automatické číslování. První číslo v závorce značí, od jakého čísla se bude začínat a druhé číslo udává o kolik se bude každý nový záznam navyšovat. Dále spojení slov primary key označuje, že atribut bude primárním klíčem tabulky, dohromady to znamená, že atribut Id nikdy nebude obsahovat hodnotu NULL.

Na dalších řádcích definujeme ostatní atributy entity. Každý řádek začíná názvem atributu, na druhé pozici je datový typ a na poslední zda může atribut nabývat hodnoty NULL (v defaultním nastavení může NULL nabývat).

Datový typ int označuje, že atribut bude celé číslo. Varchar slouží k ukládání textových řetězců, a umí pracovat s daty všech abeced (umožňuje ukládat písmena s diakritikou). Číslo v závorce udává, kolik znaků může řetězec maximálně obsahovat. Jak

si můžeme všimnout, atribut Popis nemá v závorce číslo, ale má tam Max, do takového políčka lze uložit řetězec o délce až 2^31-1 bytů.
Dalšími používanými typy jsou například Datetime, Time nebo Bit. Datetime využíváme pro ukládání data i času zároveň. Naopak Time, ukládá pouze čas, ale oproti prvnímu zmíněnému lze ukládat s přesností na milisekundy. Bit může nabývat pouze hodnot 1 nebo 0 (true/false), takže se používá obdobně jako v jazyce C# datový typ bool.

4.3.2 Nastavení cizích klíčů

Aby mezi sebou tabulky měly vazby, musíme je propojit, k tomu slouží cizí klíče. Například k určení, zda má uživatel přístup k nějakému projektu, byla vytvořena tabulka, která obsahuje záznamy k propojení tabulek tb_Uzivatel a tb_Projekt. Jako první byla vytvořena tabulka tb_ProjektyUzivatele.

```
CREATE TABLE tb_ProjektyUzivatele(
    Id int IDENTITY(1,1) primary key,
    Uzivatel int NOT NULL,
    Projekt int NOT NULL
)
```

Obrázek 3 - Vytvoření mezitabulky tb_ProjektyUzivatele

Atribut Uzivatel dostal datový typ int, a ukládá Id uživatele z tabulky tb_Uzivatel. Atribut Projekt ukládá Id z tabulky tb_Projekt. Dále byly nastaveny vazby.

```
ALTER TABLE tb_ProjektyUzivatele
ADD CONSTRAINT FK_Uzivatel_ProjektyUzivatele FOREIGN KEY (Uzivatel) REFERENCES tb_Uzivatel(Id);

ALTER TABLE tb_ProjektyUzivatele
ADD CONSTRAINT FK_Projekt_ProjektyUzivatele FOREIGN KEY (Projekt) REFERENCES tb_Projekt(Id);
```

Obrázek 4 - Nastavení vazeb pro mezitabulku

Nastavené vazby jsou typu 1:N, jeden uživatel má přístup k více projektům a zároveň do jednoho projektu má přístup více uživatel. Záznam v tabulce vypadá následovně:

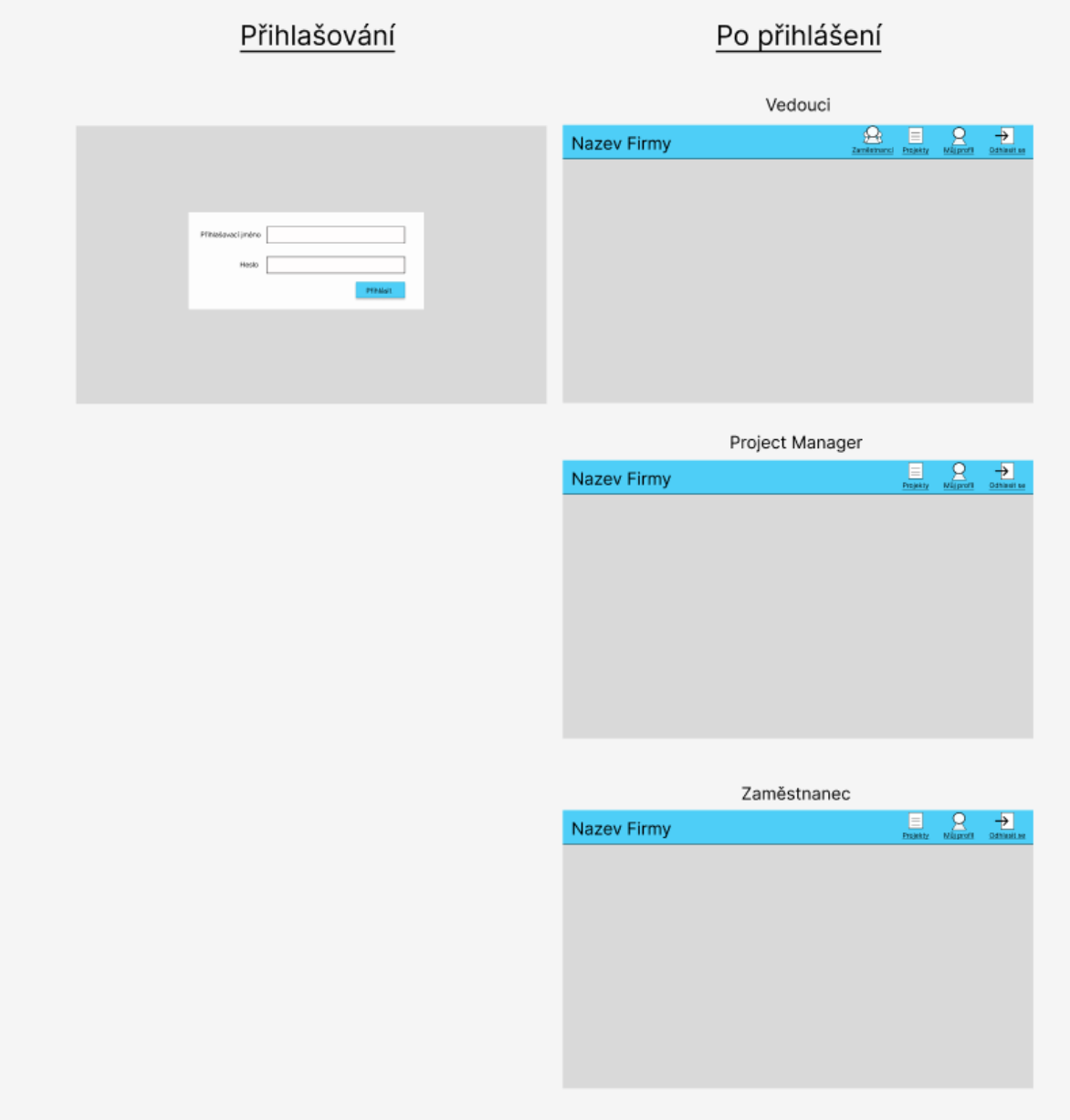
Id	Uzivatel	Projekt
15	4	8

Obrázek 5 - záznam z tabulky tb_ProjektyUzivatele

Ze záznamu lze vyčíst, že uživatel, jehož identifikační číslo je 4 má přístup do projektu s identifikačním číslem 8.

4.4 Grafický návrh

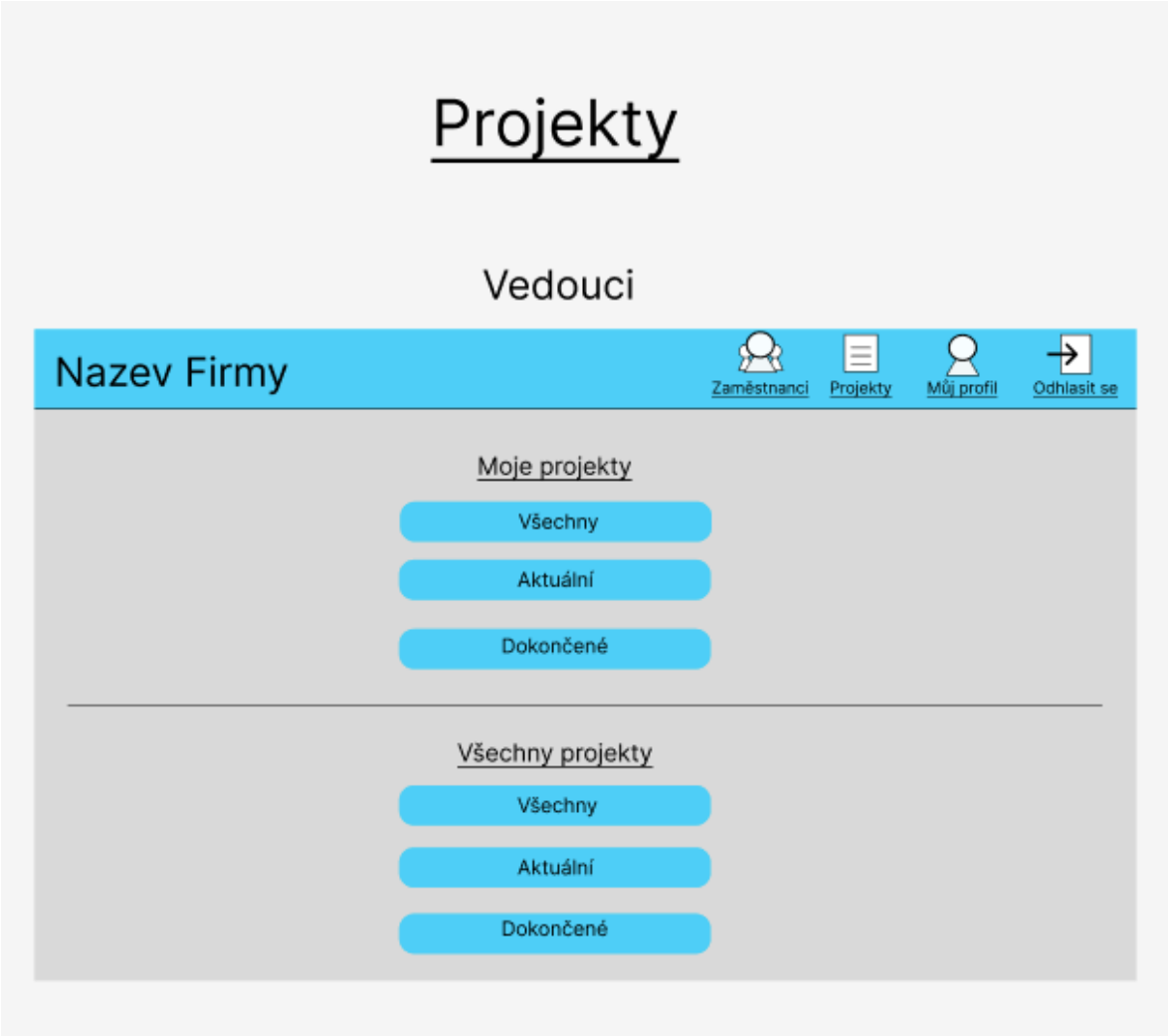
Dalším krokem byl grafický návrh aplikace, k základnímu navržení, kam by měla aplikace směřovat. V tomto kroku byl navržen základní design, kam by aplikace měla směřovat. Návrh byl vytvořen pomocí online nástroje Figma.



Obrázek 6 - Návrh přihlašování

Jako první bylo zhotoveno, jak by mohl vypadat přihlašovací formulář, a také pohled jednotlivých rolí po přihlášení. Můžeme si všimnout, že vedoucí (nejvyšší role) má k dispozici sekci zaměstnanci hned z úvodního menu.

Dále pak byly navrženy i ostatní stránky, a to vždy z pohledu všech rolí. Na následujícím obrázku je návrh sekce pro rozcestník na projekty při přístupu nejvyšší role.

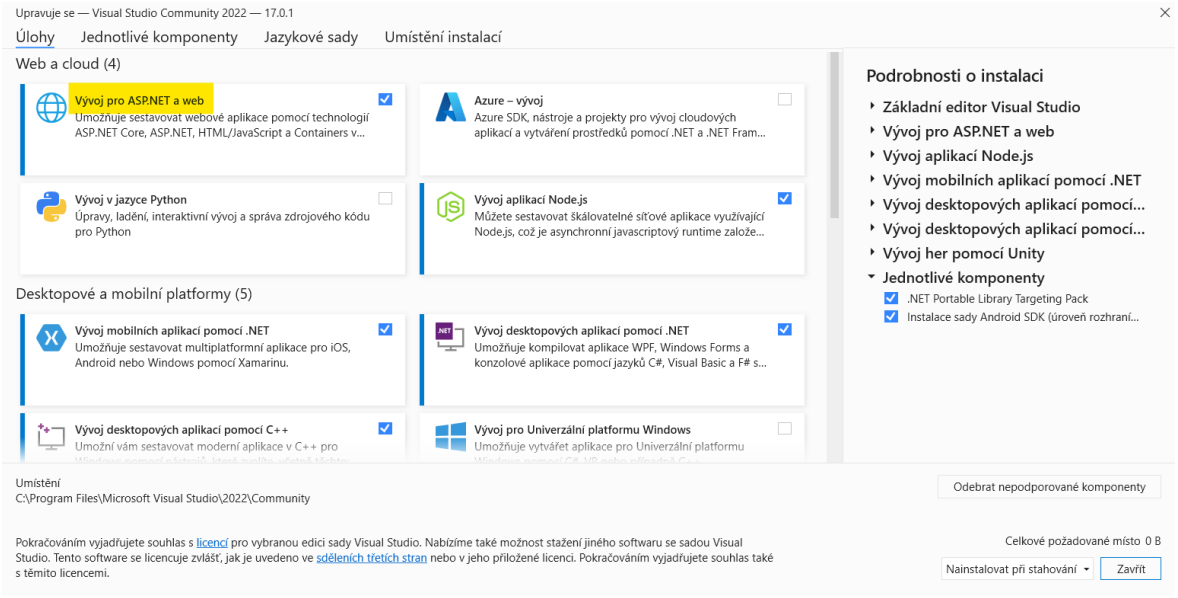


Obrázek 7 - Návrh sekce Projekty

4.5 Založení projektu

4.5.1 Stažení a instalace IDE pro vývoj

Aplikační a logická část projektu byla napsána v Microsoft visual studio verze 2022 community. Z této stránky byl stažen instalační program, po jehož spuštění se nainstaluje Visual studio installer, ve kterém již vybíráme, jaké věci budeme vytvářet, teda pro jaké doinstalujeme podporu.

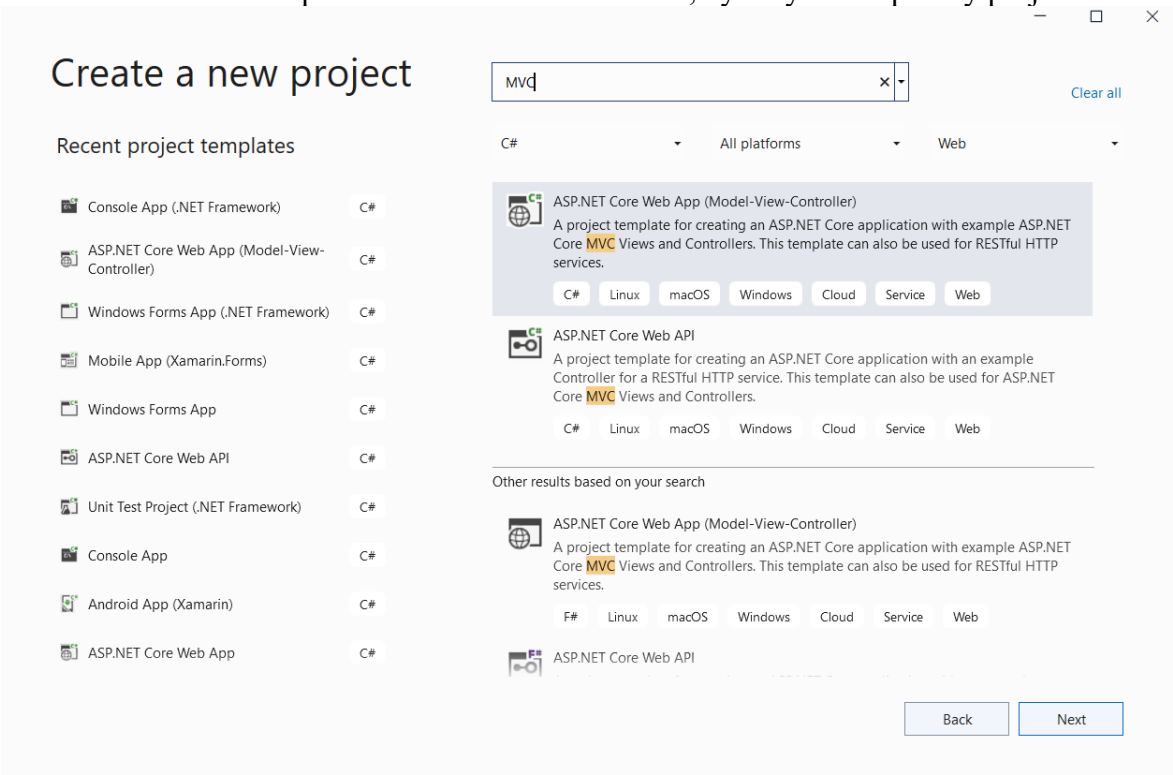


Obrázek 8 - Visual studio installer

Pro webovou aplikaci v ASP.NET bylo potřeba instalovat žlutě zvýrazněnou úlohu s názvem „Vývoj pro ASP.NET a web“. Na kartě Jazykové sady byla k vývoji zvolena pouze anglická jazyková sada.

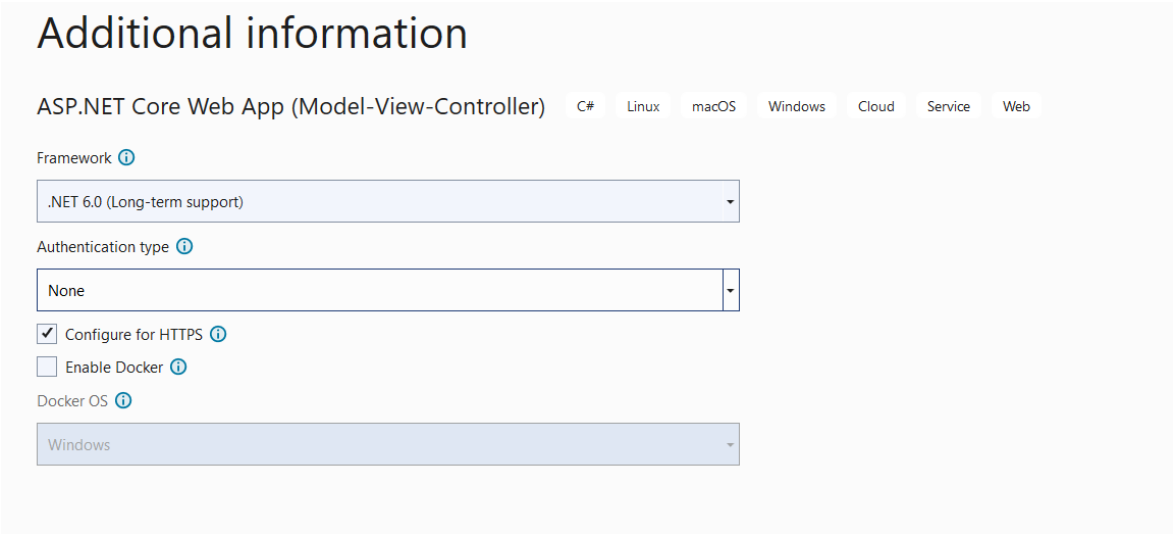
4.5.2 Vytvoření projektu ve Visual studiu

Dalším krokem po nainstalování Visual studia, bylo vytvořit správný projekt.



Obrázek 9 - Výběr správného projektu

Pomocí vyhledávacího políčka, kam byla zadána klíčová zkratka (MVC), byl hned nalezen správný projekt, poznat to lze z jeho popisu a klíčových slov pod ním. Dalším krokem bylo zadat název projektu, a posledním krokem byl výběr verze .NET a konfigurace pro HTTPS protokol. Verze .NET byla zvolena 6.0.



Obrázek 10 - Verze .NET a konfigurace pro HTTPS

4.6 Propojení aplikace s databází

4.6.1 Instalace NuGet balíčků

Pro práci s databází v projektu jsou využívány různé knihovny, stahují je ve formě balíčků. Prvním a nejdůležitějším balíčkem je EntityFramework (popsaný výše v teoretické části). Dalšími použitými balíčky byly Microsoft.EntityFrameworkCore.Desing, Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Tools a také Microsoft.VisualStudio.Web.CodeGeneration.Desing. Poslední zmíněný slouží k automatickému předvytvoření hlavních funkcí aplikace, jako jsou jednotlivé stránky. Například vytvoření nového záznamu do databáze.

Balíčky lze instalovat dvěma způsoby a to v prvním případě přes správce NuGet balíčků ve Visual studiu, kde lze i vyhledávat mezi balíčky, druhým způsobem je spuštění příkazu v konzoli.

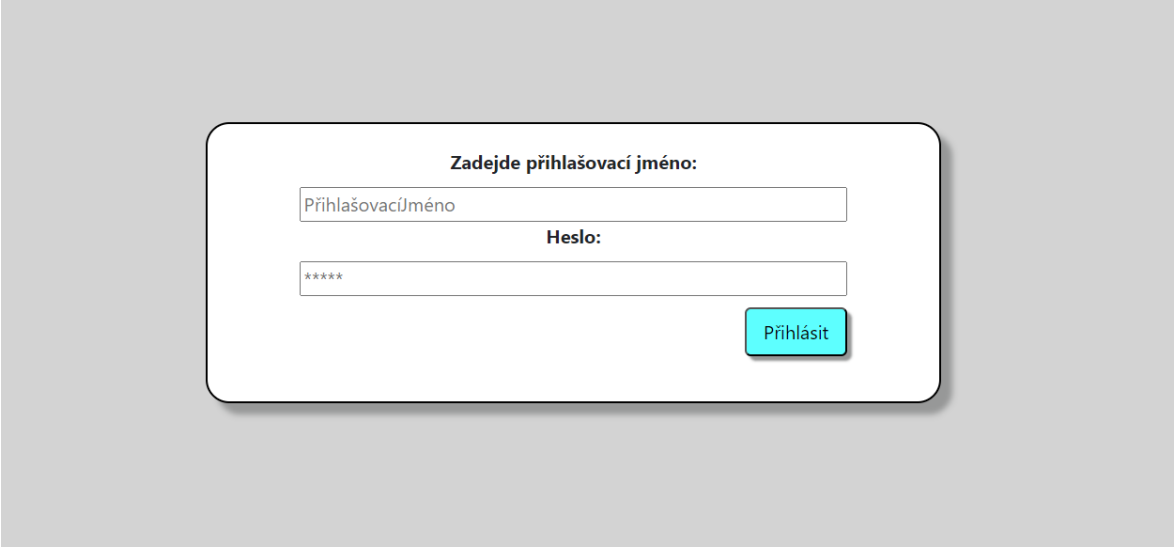
4.6.2 Vytvoření databázového kontextu a modelů entit

Po instalaci všech potřebných balíčků se přistoupilo k vytvoření databázového kontextu a modelů podle tabulek z databáze. Příkazem Scaffold-DbContext vytvoříme propojení mezi projektem a databází. Příkazu musíme zadat název serveru, na kterém databázi najde. Při přístupu přes účet na SQL serveru také název uživatele a jeho heslo, dále povolit důvěřování certifikátům. Dalším parametrem je název balíčku, aby příkaz věděl, který k propojení použít. Výstupní složka byla nastavena na složku Modely, jedná se o složku v projektu, do které se kontext a entity založí. Poslední parametr (Force) přepíše všechny již existující entity, například kdyby byl špatně založený model entity (měl jiné atributy než tabulka v databázi) tento parametr zajistí, aby se model opravil.

Obrázek 11 - Zadání příkazu Scaffold-DbContext

4.7 Přihlašování do aplikace

Před připojením do aplikace a procházením obsahu je nutné se přihlásit. Do přihlašovacího formuláře se zadají přihlašovací údaje (přihlašovací jméno a heslo).



Obrázek 12 - Přihlašovací formulář

Po zadání údajů a kliknutí na tlačítko Přihlásit začne na serveru ověřování zadaných údajů. První server ověří, zda existuje uživatel se zadaným přihlašovacím jménem, a pokud ano, pokračuje s ověření hesla. Ověření probíhá způsobem, při kterém server vytvoří stejným hashovacím algoritmem hash z řetězce zadaného do vstupu pro heslo, a výslednou hash porovná s hashí hesla z databáze. Pokud bylo heslo zadáno správně, vytvoří se příslušnému uživateli autorizační token. Ten je založen jak do databáze, tak do session uživatelova webového prohlížeče. Token pak má platnost následujících 12 hodin, do odhlášení, nebo do vypnutí webového prohlížeče. K hashování je použit algoritmus SHA256. Ve funkci se ke vstupu na hash přidává jeden znak, z důvodu možných uniklých databází hashů.

```
5 references
public static class Encryption
{
    5 references
    public static string Encrypt(string vstup)
    {
        vstup = vstup + "s";

        using (SHA256 sha256 = SHA256.Create())
        {
            byte[] bytes = Encoding.UTF8.GetBytes(vstup);
            byte[] hash = sha256.ComputeHash(bytes);

            StringBuilder result = new StringBuilder();
            for (int i = 0; i < hash.Length; i++)
            {
                result.Append(hash[i].ToString("x2"));
            }

            return result.ToString();
        }
    }
```

Obrázek 13 - Ukázka metody pro hashování řetězců

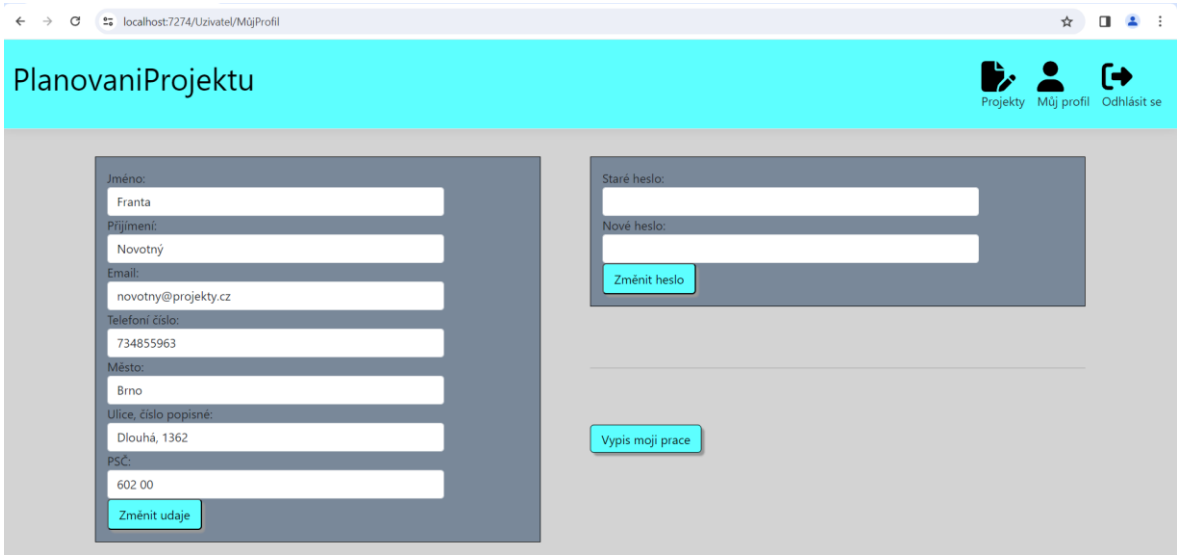
4.8 Aplikace z pohledu uživatele

Běžným uživatelem je v případě této aplikace zaměstnanec na nižší úrovni. Má přístup do dvou hlavních sekcí a to jsou Projekty a Můj Profil. Také je v hlavním menu možnost odhlášení se z aplikace.

Obrázek 14 - Menu po přihlášení z pohledu zaměstnance

4.8.1 Sekce Můj Profil

V sekci Můj Profil má uživatel možnost změny svých osobních údajů, dále možnost změny hesla a poslední možností je výpis z odvedené práce. Podle URL adresy, lze rozpoznat, že stránka na obrázku č. 15 se nachází v projektu ve složce Uzivatel a název stránky je MůjProfil.



Obrázek 15 - Sekce Můj Profil

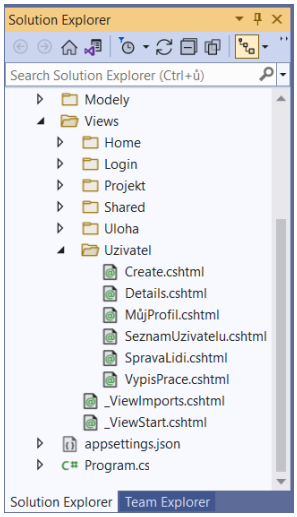
Osobní údaje jsou automaticky předvyplněné, o jejich předvyplnění se stará controller, který je prostředníkem mezi daty a rozhraním pro uživatele. O poslání správných dat se na controlleru UzivatelController (controller určený pro manipulaci s daty o uživatelích) stará metoda s názvem MůjProfil. Tato metoda jako první zjistí, zda uživatel, který ji vyvolal, má platný autorizační token. K tomu byla navržena metoda CheckAutorizacniToken, která kontroluje platnost tokenu. V případně neplatnosti tokenu uživatele přesměruje na přihlašovací formulář a v případě platnosti tokenu do ViewBagu uloží roli, do které uživatel patří. V posledním kroku jsou z databáze vytažena pomocí LINQ výrazu data z tabulky tb_Uzivatel pro aktuálně přihlášeného uživatele, který je zjištěn z autorizačního tokenu a nalezen pomocí Id.

```
public IActionResult MůjProfil()
{
    if (CheckUser.ChechAutorizationToken(HttpContext.Session.GetString("AutorizacniToken")) == null)
        return RedirectToAction("Login", "Login");
    else
        ViewBag.Autorizovano = CheckUser.ChechAutorizationToken(HttpContext.Session.GetString("AutorizacniToken"));

    return View(_conn.TbUzivatels.Where(x => x.Id == CheckUser.IdUzivatele(HttpContext.Session.GetString("AutorizacniToken"))).FirstOrDefault());
}
```

Obrázek 16 - Můj profil na controlleru

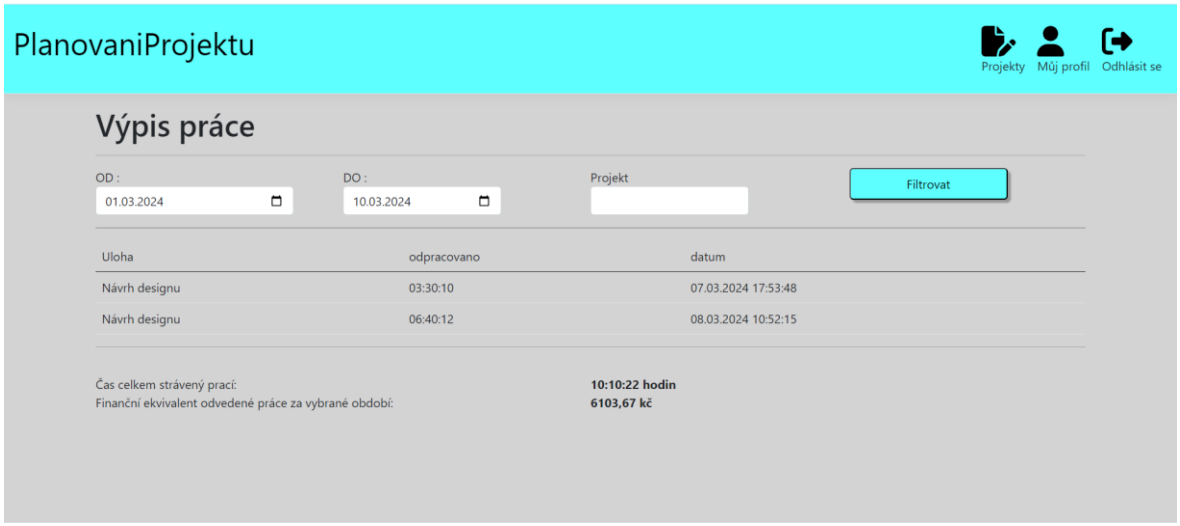
Správnou HTML stránku, kterou je potřeba vykreslit, pozná aplikace podle hierarchie stránek. Hledá ji ve složce Views, dále hledá složku, která se jmenuje stejně jako controller, ze kterého je volána (voláme z controlleru UzivatelController, což znamená, že aplikace hledá složku s názvem Uzivatel). V této složce hledá název již konkrétní HTML stránky, její název musí být shodný s metodou controlleru, která byla volána. V tomto případě se tedy jedná o stránku MůjProfil.cshtml.



Obrázek 17 - Hierarchie stránek

Po kliknutí na tlačítko „Výpis mojí práce“ je uživatel přesměrován na stránku, která obsahuje filtry a tabulku. V základu po kliknutí se tabulka vyplní všemi daty o veškeré práci, kterou uživatel odvedl. Pomocí filtrů OD a DO může nastavit období, za které má být práce vypsána (například za minulý měsíc). Dále pokud chce svoji práci omezit na jeden určitý projekt, může toho docílit pomocí políčka projekt, kde se nachází seznam všech projektů, na kterých uživatel pracoval. Pod tabulkou je k nalezení kolik si uživatel vydělal prací na úlohách projektů, a také kolik času odpracoval celkem.

Na obrázku 18, je zobrazen výpis práce přihlášeného uživatele. Můžeme si všimnout, že práce, kterou vypisujeme je pouze mezi 1. - 10. březnem a není nijak omezena, do jakého projektu byla zpracována. V tabulce vidíme vždy název úlohy, která byla plněna, kolik na ní času uživatel strávil času, a také datum a čas dokončení. Pod tabulkou je součet času vybraných projektů, kterým uživatel věnoval práci, a také vypočteno na základě finančního ohodnocení úloh, kolik by mělo být zaměstnanci za odvedenou práci vyplaceno.

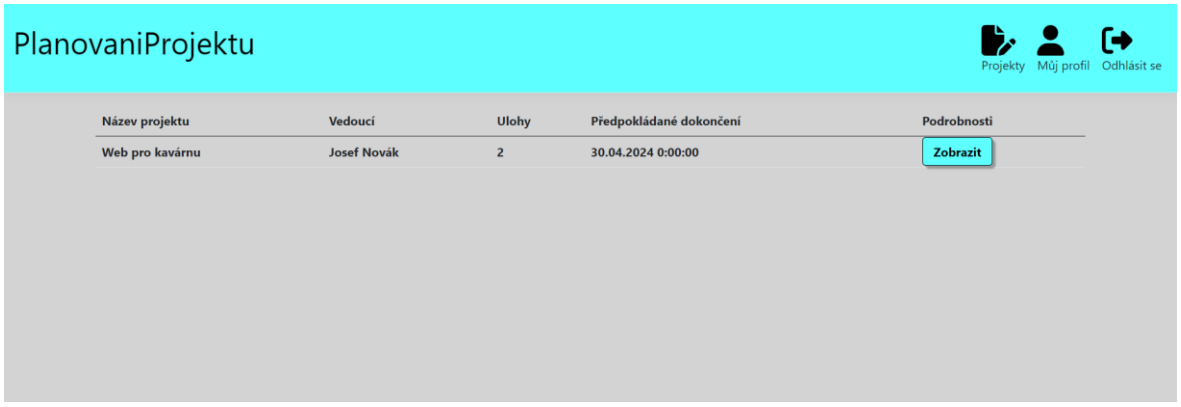


Obrázek 18 - Výpis práce ze sekce Můj Profil

4.8.2 Sekce Projekty

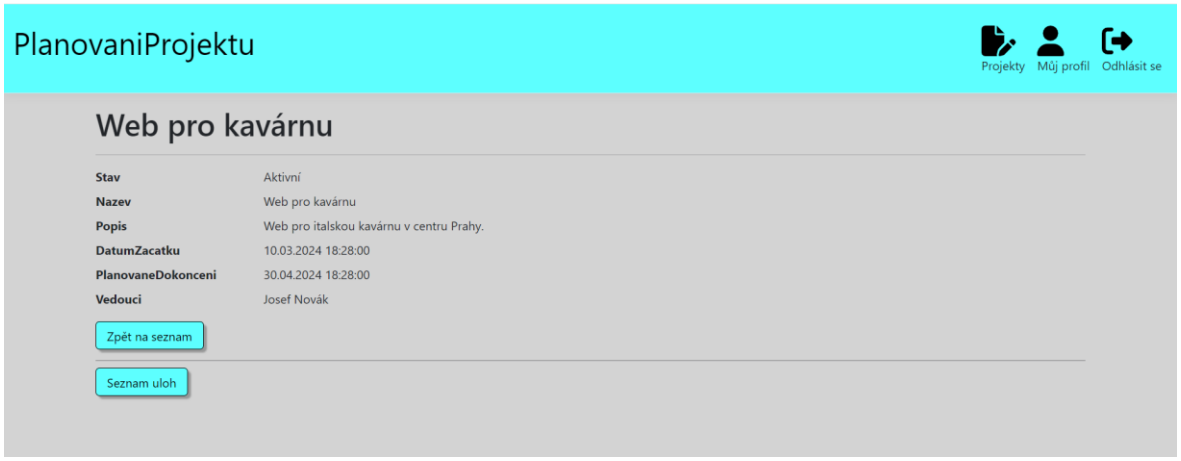
V sekci s projekty, uživatel jako první vidí rozcestník, kde má na výběr, jaké projekty může procházet. Lze zvolit ze tří možností. Těmi jsou výpis pouze aktuálních projektů, které mají momentálně aktivní nějaké úlohy k práci. Druhou možností je výpis projektů, které již jsou dokončené, a uživatel se na nich podílel. Poslední možností je výpis všech projektů, jak aktuálních, tak i těch dokončených.

Po vybrání jedné z možností, se zobrazí výpis projektů, které spadají do vybrané kategorie. V tabulce se zobrazí pouze projekty, do kterých uživatel dostal přístup. Prvním záznamem je název projektu, následuje jméno vedoucího projektu. Ve sloupci pro úlohy, se vypisuje počet úloh, na kterých je možné aktuálně pracovat. Také je k vidění do kdy by měl být projekt dokončený, a lze si zobrazit podrobnosti.



Obrázek 19 - Seznam projektů z pohledu uživatele

V detailu projektu, lze najít více informací o projektu. Je zde možnost vypsat si seznam úloh, na kterých zde lze pracovat, v tomto příkladu tedy dvě.



Obrázek 20 - Detail projektu z pohledu uživatele

V seznamu úloh máme ke každé úloze uvedený název, o jakou práci se jedná, hodinovou sazbu za odvedenou práci, a předpokládanou dobu práce, kterou úloha zabere. Po kliknutí na tlačítko Detail se zobrazí detail úlohy, obdobně jako tomu bylo u projektů.

Obrázek 21 - Seznam úloh z pohledu uživatele

Na detailu úlohy je tlačítko s názvem „Začít pracovat na úkolu“. Po kliknutí na něj, se zobrazí stránka, na které si uživatel může zapnout časovač, který bude zaznamenávat, jak dlouho úlohu vykonává. Na stránce se zobrazuje název a popis úlohy. Dále je zde tlačítko pro zapnutí a vypnutí časovače. Tlačítko k uložení, pošle do databáze informace, na jaké úloze bylo pracováno, kdo na ni pracoval, a kolik času práci věnoval. Poslední tlačítko, označí úlohu jako splněnou, a nebude se již dále v seznamu zobrazovat. Aplikace je momentálně navržena tak, že na každé úloze může v jeden okamžik pracovat pouze jeden uživatel.

Obrázek 22 - Stránka pro práci na úloze

Na následujícím obrázku je k vidění část kódu, která zajišťuje správnou funkčnost časovače. Jedná se také o jedinou část aplikace, která využívá ve větší míře Java Script. Funkce startStop se spustí či zastaví kliknutím na tlačítko nesoucí stejný název. V prvním kroku nastavíme hodiny, minuty a sekundy na čas, který je aktuálně na časovači zobrazen. Pokud je časovač spuštěn, tak každou sekundu dojde k zavolání funkce aktualizovatCas, která přičítá sekundu při každém zavolání a kontroluje, zda již není přesaženo 60 sekund, aby se přičetla minuta. Stejně tomu je i u hodin, pokud ovšem časovač přejde víc jak 24 hodin, bude vyresetován. Ovšem není předpokládáno, že by uživatel pracoval na jednom úkolu déle, jelikož autorizace vyprší již po dvanácti hodinách. Proto by uživatel neměl v jedné relaci pracovat déle než tuto dobu.

```
function startStop() {

    var pocatecniHodinyElement = document.getElementById("hodiny");
    hodiny = pocatecniHodinyElement.innerHTML;
    var pocatecniHodinyElement = document.getElementById("minuty");
    minuty = pocatecniHodinyElement.innerHTML;
    var pocatecniHodinyElement = document.getElementById("sekundy");
    sekundy = pocatecniHodinyElement.innerHTML;

    if (spusteno) {
        spusteno = false;
        clearInterval(casovac);
    } else {
        spusteno = true;
        casovac = setInterval(aktualizovatCas, 1000);
    }
}

function aktualizovatCas() {
    sekundy++;
    if (sekundy === 60) {
        sekundy = 0;
        minuty++;
        if (minuty === 60) {
            minuty = 0;
            hodiny++;
            if (hodiny === 24) {
                sekundy = 0;
                minuty = 0;
                hodiny = 0;
            }
        }
    }
    document.getElementById("hodiny").innerHTML = formatujCas(hodiny);
    document.getElementById("minuty").innerHTML = formatujCas(minuty);
    document.getElementById("sekundy").innerHTML = formatujCas(sekundy);
```

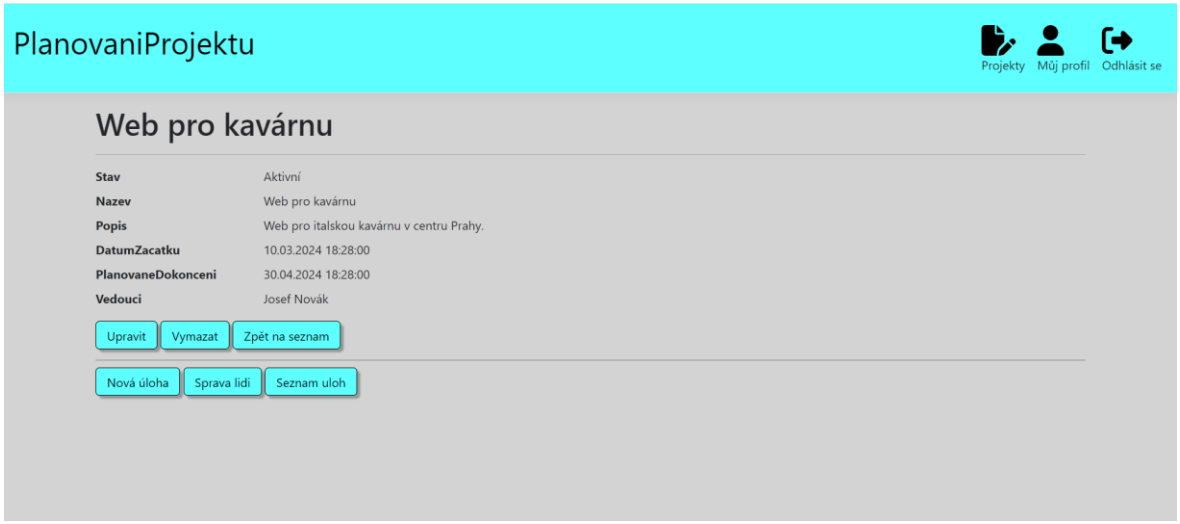
Obrázek 23 - Časovač v Java Scriptu

4.9 Aplikace z pohledu Projekt manažera

Roli projektového manažera, je udělována tomu, kdo se stará o vedení jednotlivých projektů. Má tedy možnost nové projekty zakládat a také je zpětně upravovat. Přístup má pouze do projektů, které sám vede, nebo do kterých ho přiřadil jejich vedoucí. Na projektech které založil, má možnost správy lidí. Tato možnost umožňuje přidávat a odebírat uživatele, kteří mají přístup k projektu. Dále je možné si zobrazit kolik, který uživatel odpracoval na úlohách projektu. Ale to pouze aktuálního projektu, nelze zde měnit výpis úloh podle projektu. Nemá tedy z této pozice přístup ke všem osobním údajům všech uživatelů a tudíž ani k výpisu práce uživatelů, kterou odvedli na jiných projektech.

V sekci Projekty má tedy projektový manažer navíc možnost přidat nový projekt, a v detailu na projektu má na výběr z více akcí (obrázky 24 a 25).

Obrázek 24 - Rozcestník projektů z pohledu projektového manažera



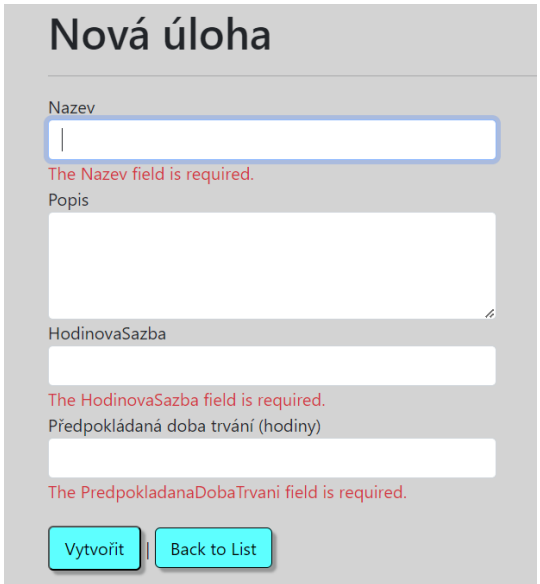
Obrázek 25 - Detail projektu z pohledu jeho vlastníka

Zakladatel projektu může editovat všechny informace o projektu, případně projekt i odstranit. Dále může do projektu přidávat nové úlohy, spravovat přístup lidem do projektu, a stejně jako uživatel zobrazit seznam úloh.

4.9.1 Vytvoření nové úlohy

Pro vytvoření nové úlohy, musí její zakladatel zadat několik informací. Jako jsou název, hodinová sazba a předpokládaná doba trvání. Popis je volitelná informace. Na obrázku 27 je k vidění ASP validátor, který znemožňuje založení nové úlohy bez vyplnění povinných polí, nebo špatně vyplněné úlohy. Controller, který je určený k práci s úlohami, zjistí, z jakého projektu byl formulář vyvolán a správně přiřadí úlohu k projektu.

Obrázek 26 - Formulář pro vytvoření nové úlohy



Obrázek 27 - Ochrana proti odeslání nevyplněné úlohy

Na řádku 14 se nachází element, který obstarává validaci formuláře. Podklady k validaci bere podle modelu entity, který je v projektu uložen ve složce Models. Pozná tedy, jaká omezení jsou nastavená pro které pole, z čehož určí, zda musí být pole vyplněné, nebo jestli je v požadovaném formátu (například email). Pod tímto elementem následují čtyři elementy div, které mají class form-group. V tomto elementu je tedy očekáván element label, který informuje uživatele, jaká informace se má zadávat. Následuje input, do něj uživatel zapisuje hodnotu samotnou. Klíčová vlastnost asp-for přiřazuje input k atributu entity. V tomto případě se tedy na řádku 17 nachází vstup pro atribut úlohy, který je pojmenovaný jako Nazev. V následujícím elementu je pro něj nastavena validace.

Na řádku číslo 13 je element pro celý formulář. Tento formulář obsahuje vlastnost asp-action ve které je napsáno, jaká metoda má být zavolána a přijme informace, které uživatel vyplnil. Řádek 36 obsahuje tlačítko, kterým se formulář odešle. První dojde k validaci, a pokud je vše správně vyplněno, jsou data zaslána na příslušné metodě.

```
11  <div class="row">
12      <div class="col-md-4">
13          <form asp-action="VytvoreniUlohy">
14              <div asp-validation-summary="ModelOnly" class="text-danger"></div>
15              <div class="form-group">
16                  <label asp-for="Nazev" class="control-label"></label>
17                  <input asp-for="Nazev" class="form-control" />
18                  <span asp-validation-for="Nazev" class="text-danger"></span>
19              </div>
20              <div class="form-group">
21                  <label asp-for="Popis" class="control-label"></label>
22                   <textarea asp-for="Popis" class="form-control" style="height: 100px;" /> </textarea>
23                  <span asp-validation-for="Popis" class="text-danger"></span>
24              </div>
25              <div class="form-group">
26                  <label asp-for="HodinovaSazba" class="control-label"></label>
27                  <input asp-for="HodinovaSazba" class="form-control" />
28                  <span asp-validation-for="HodinovaSazba" class="text-danger"></span>
29              </div>
30              <div class="form-group">
31                  <label asp-for="PredpokladanaDobaTrvani" class="control-label">Předpokládaná doba trvání (hodiny)</label>
32                  <input asp-for="PredpokladanaDobaTrvani" class="form-control" />
33                  <span asp-validation-for="PredpokladanaDobaTrvani" class="text-danger"></span>
34              </div>
35              <div class="form-group" style="margin-top: 20px;">
36                  <button type="submit" value="Vytvořit" class="loginButton"> </button>
37                  <a class="loginButton btnSeznam" asp-action="Index">Back to List</a>
38              </div>
39          </form>
40      </div>
41  </div>
```

Obrázek 28 - Kód stránky pro vytvoření nového projektu s ASP prvky

Metoda, na kterou byla data zaslána, přijme data jako celý objekt. První dojde k ověření, zda je uživatel přihlášený. Pokud přihlášený není, je vrácen na přihlašovací formulář. Pokud je přihlášený, ale nemá dostatečná práva, je vrácen na detail projektu, ze kterého akci vyvolal. Pokud uživatel měl dostatečná oprávnění, uloží se informace do ViewBagu. Poté následuje kód, který nejprve přiřadí atributu Projekt hodnotu ze session, ve které je uloženo, z jakého projektu byla akce vytváření vyvolána. Na řádku 46 je příslušná úloha přidána do kolekce všech úloh, na následujícím řádku jsou změny uloženy a úloha je zapsána do databáze. Uživatel je pak vrácen na detail projektu, ze kterého akci vyvolal.

```
0 references
36  public ActionResult VytvoreniUlohy(TbUloha uloha)
37  {
38      if (CheckUser.ChechAutorizationToken(HttpContext.Session.GetString("AutorizacniToken")) == null)
39          return RedirectToAction("Login", "Login");
40      else if (CheckUser.ChechAutorizationToken(HttpContext.Session.GetString("AutorizacniToken")) == "Uzivatel")
41          return RedirectToAction("Details", "Projekt", new { id = Convert.ToInt32(HttpContext.Session.GetString("CisloUlohy")) });
42      else
43          ViewBag.Autorizovano = CheckUser.ChechAutorizationToken(HttpContext.Session.GetString("AutorizacniToken"));
44
45      uloha.Projekt = Convert.ToInt32(HttpContext.Session.GetString("CisloUlohy"));
46      _conn.TbUlohas.Add(uloha);
47      _conn.SaveChanges();
48
49      return RedirectToAction("Details", "Projekt", new { id = uloha.Projekt });
50  }
51
```

Obrázek 29 - Založení nové úlohy na controlleru

4.9.2 Správa uživatelů pro jednotlivé projekty

Aby běžný uživatel měl přístup do projektu, a mohl zpracovávat úlohy, musí mu do něj prvně být povolen vstup. Správu uživatel tedy lze nalézt po kliknutí na tlačítko „Správa lidí“ (viz obrázek 25). Zobrazí se seznam uživatelů, který obsahuje všechny uživatele, těm lze měnit práva na přístup do projektu. Důležitý je sloupec s názvem Akce, v něm se nachází tlačítko na odebrání či na přidání uživatele. Tlačítko výpis práce, zobrazí seznam odpracovaných úloh, stejně jako tomu je v sekci Můj Profil, pouze s rozdílem, že projektový manažer zde nemá možnost vybírat úloh podle projektu, ale zobrazují se vždy jen úlohy vypracované do aktuálního projektu.

Obrázek 30 - Správa přístupu do projektu

Vykreslování tlačítka podle toho, zda uživatel má, či nemá přístup do projektu, je realizován pomocí podmínky na samotném pohledu. Controller vytvoří seznam identifikačních čísel zaměstnanců, a podmínka kontroluje, zda aktuálně vykreslovaný uživatel je již členem projektu. Pokud ano, tak je vykresleno tlačítko pro odebrání uživatele, pokud členem není, je vykresleno tlačítko pro jeho přidání.

```
<td>
    @if (@ViewBag.Lidi.Contains(@item.Id))
    {
        <a asp-action="OdebratOsobuZProjektu" asp-route-id="@item.Id" asp-route-projekt="@ViewBag.Projekt"
    }
    else
    {
        <a asp-action="PridatOsobuDoProjektu" asp-route-id="@item.Id" asp-route-projekt="@ViewBag.Projekt"
    }
</td>
```

Obrázek 31 - Vykreslování tlačítka podle přístupu k projektu

4.10 Aplikace z pohledu vedoucího

Vedoucí firmy, často označováni jako CIO, by měl mít přístup ke všem informacím o všech projektech a zaměstnancích. Proto uživatel s touto rolí má k dispozici úplnou správu uživatelů a také všech projektů.

4.10.1 Správa všech uživatelů v aplikaci

Uživateli s nejvyšší rolí je k vidění sekce Uživatelé. po kliknutí na ni, se objeví tabulka, ve které jsou všichni uživatelé, kteří mají v aplikaci účet. Ve sloupci s názvem Akce jsou pro každého uživatele možné akce dvě. První akcí je detail, ve kterém najdeme všechny informace o uživateli, jako je bydliště nebo email. Na detailu lze také uživatele odstranit z aplikace. Druhá akce „Výpis práce“, zobrazí stejnou tabulku a stejné filtry, jako kdyby do něj bylo vstoupeno se sekce Můj Profil. Vedoucí firmy tak tedy může vidět, všechny zaměstnance a sledovat, v jakém období, či na jakém projektu odpracovali kolik hodin.

Poslední možností zde, je založení nového uživatele. K jeho vytvoření stačí vyplnit přihlašovací jméno, dočasné jednorázové heslo a roli, do které bude uživatelský účet patřit. Heslo by mělo být uživateli doporučeno změnit, a osobní údaje si případně může doplnit po prvním přihlášení také.

Obrázek 32 - Seznam všech uživatelů v aplikaci

4.10.2 **Správa všech projektů v aplikaci**

Pokud přihlášený uživatel spadá do role Vedoucí, potom se nabízí navíc sekce Všechny projekty, kde jsou k nalezení i projekty, které sám nezaložil. Uživatel v této roli nemusí být do žádných projektů přidáván přes Správu Lidí, jelikož má ke všem projektům přístup i bez toho. Proto se uživatelé v nejvyšší roli ani neukazují v nabídce pro přidání či odebrání přístupu. Uživatel v této roli má také možnost založit si svůj projekt, a přidávat si do něj lidi.

Obrázek 33 - Správa projektů z pohledu vedoucího

5 Výsledky a diskuse

Povedlo se vytvořit aplikaci, která obsahuje základní oblasti pro správu projektů a také má všechny informace chráněné za přihlašovacím formulářem. Pro aplikaci po jejím otestování funkčnosti byl založen veřejný repozitář na stránce GitHub. Také byla doložena záloha databáze se základními daty pro vyzkoušení funkčnosti. Všechen kód je tak veřejně dostupný a může být stahován uživateli, kteří by aplikaci měli zájem používat.

Aplikace tak nyní obsahuje správu uživatelů, správu projektů a správu úloh jednotlivých projektů. Za omezení nyní lze považovat, že na každé úloze v jeden okamžik může pracovat pouze jeden zaměstnanec.

Možným rozšířením do budoucna by mohlo být přidání stavu na jednotlivé záznamy o práci, zda již byly proplacené. Případně přidat k úlohám jednotlivé technologie, které je potřeba na jejich zpracování ovládat.

Během vývoje aplikace dělaly největší problém verze jednotlivých balíčků, kdy při přechodu na novou verzi balíčku jednoho, bylo nutné zvedat verzi i ostatních balíčků, jinak kompilátor hlásil chybu. Dalším problémem se ukázalo odstraňování záznamů, na které byly vázány cizí klíce. Občas tedy během vývoje docházelo k pádům aplikace z důvodu, že se v nějaké proměnné neočekávaně objevila hodnota null. Posledním problémem bylo správné propojení s databází, kde příkaz Scaffold-DbContext neprošel na první pokus, chyby byly hlášeny ve spojitosti s důvěryhodností k certifikátu.

Teoretickým omezení může být, že databáze byla tvořena na SQL serveru, což nemusí být nejlepším řešením pro menší firmy, kterým je aplikace primárně určena. Řešením by bylo vyměnit balíčky pro připojení na SQL server, za balíčky pro připojení k databázi například MySQL, které jsou hojně využívané pro webové aplikace. Poté stačí změnit connection string, který se nachází v databázovém kontextu aplikace, a jako další si databázi pomocí migrací přesunout na jiný databázový engine. Tabulky jednotlivých entit by měly být vygenerovány automaticky.

6 Závěr

Závěrem lze napsat, že se v průběhu bakalářské práce se podařilo vytvořit webovou aplikaci, které umožňuje agilní přístup k řízení projektů v oblasti IT. V aplikaci lze nalézt správu zaměstnanců, také přehled všech projektů a jejich úloh. Přístup do jednotlivých projektů lze pro uživatele aplikace modifikovat. Zaměstnanci mají možnost, aby aplikace zaznamenávala jejich odpracovaný čas, a rovnou si zde mohou prohlédnout seznam odpracovaných úloh, a kolik by jim mělo být za práci vyplaceno.

Zdrojový kód aplikace byl nahrán na GitHub, aby k němu měl každý přístup, a mohl si ho stáhnout pro svoje využívání.

Aplikace je připravena na možnosti dalšího rozšíření. Například lepší správu úloh, které již byly proplacené, nebo tisk odvedené práce na fakturu.

Použité technologie a metody, se ukázaly jako dostatečné pro vývoj webové aplikace. Doporučenou metodologii pro vedení projektů této aplikace, je dozajista agilní metodologie.

7 Seznam použitých zdrojů

[1] *Object-oriented programming (OOP)*. Online. , Sarah. Techtarget. 2021. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP. [cit. 2024-03-06].
[2] SARCAR, Vaskaran. Simple and Efficient Programming with C#. Apress, 2021. ISBN 978-1-4842-7322-7.
[3] *Objektově orientované programování (C#)*. Online. Microsoft Learn. 2024. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/fundamentals/tutorials/oop. [cit. 2024-03-03].
[4] *Build. Test. Deploy*. Online. Microsoft. 2024. Dostupné z: https://dotnet.microsoft.com/en-us/. [cit. 2024-03-06].
[5] VOGEL, Eric. Beginning Entity Framework Core 5: From Novice to Professional. Apress, 2007. ISBN ISBN 978-1-4842-6882-7.
[6] *Asynchronní volání synchronních metod*. Online. Microsoft Learn. 2024. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/standard/asynchronous-programming-patterns/calling-synchronous-methods-asynchronously. [cit. 2024-03-06].
[7] HALJUK, Petr. *Ponořme se do async/await v jazyce C#.* Online. PROFINIT. 2021. Dostupné z: https://profinit.eu/blog/ponorme-se-do-async-await-v-jazyce-c-1-cast/. [cit. 2024-03-06].
[8] POLAT, Engin a BELKHERAZ, Stephane. ASP.NET Core MVC 2.0 Cookbook. Packt>, 2018. ISBN 978-1-78588-675-1.
[9] *Úvod do dotazů LINQ (C#)*. Online. Microsoft Learn. 2023. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/linq/get-started/introduction-to-linq-queries. [cit. 2024-03-06].
[10] *What is Entity Framework?* Online. Entity Framework Tutorial. 2023. Dostupné z: https://www.entityframeworktutorial.net/entityframework6/what-is-entityframework.aspx. [cit. 2024-03-06].
[11] BELOUCHE, Younes. *Code First Approach vs. Database First Approach*. Online. Medium. 2023. Dostupné z: https://medium.com/codex/code-first-approach-vs-database-first-approach-a3830c0cc9b6. [cit. 2024-03-06].
[12] *What is a REST API?* Online. Red Hat. 2020. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api. [cit. 2024-03-06].
[13] *What Is a SOAP API?* Online. Postman. 2023. Dostupné z: https://blog.postman.com/soap-api-definition/. [cit. 2024-03-06].
[14] FREEMAN, Adam. Pro ASP.NET Core MVC 2. Seventh edition. London: Apress, ISBN 978-1-4842-3149-4.
[15] *ASP.NET MVC Pattern*. Online. Microsoft. 2024. Dostupné z: https://dotnet.microsoft.com/en-us/apps/aspnet/mvc. [cit. 2024-03-06].
[16] ŠTRÁFELDA, Jan. *HTTP*. Online. Jan Štráfelda. Dostupné z: https://www.strafelda.cz/http. [cit. 2024-03-03].
[17] *HTTP Request Methods*. Online. W3 School. 2024. Dostupné z: https://www.w3schools.com/tags/ref_httpmethods.asp. [cit. 2024-03-03].
[18] ŠTRÁFELDA, Jan. *Stavový kód*. Online. Jan Štráfelda. Dostupné z: https://www.strafelda.cz/stavovy-kod. [cit. 2024-03-03].
[19] Úvod do UML Zdroj: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy. Online. ITnetwork.cz. 2024. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy . [cit. 2024-03-03].

[20] *Jednoduchý návod k UML diagramům a modelování databází*. Online. Microsoft. 2019. Dostupné z: https://www.microsoft.com/cs-cz/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling. [cit. 2024-03-03].
[21] *Vývojový diagram*. Online. ASPOSE. 2021. Dostupné z: https://products.aspose.app/diagram/cs/pages/what-is-flowchart. [cit. 2024-03-03].
[22] *Vývojové diagramy*. Online. ITnetwork.cz. 2024. Dostupné z: https://www.itnetwork.cz/navrh/uml/vyvojove-diagramy. [cit. 2024-03-03].
[23] *What is an Entity Relationship Diagram (ERD)?* Online. Lucidchart. 2024. Dostupné z: https://www.lucidchart.com/pages/er-diagrams#discoveryTop. [cit. 2024-03-03].
[24] *What is Entity Relationship Diagram (ERD)?* Online. Visual Paradigm. 2024. Dostupné z: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/. [cit. 2024-03-03].
[25] *What is wireframing?* Online. Figma. 2023. Dostupné z: https://www.figma.com/resource-library/what-is-wireframing/. [cit. 2024-03-03].
[26] KAĎOUSKOVÁ, Barbora. *Co je wireframe webu, proč ho potřebujete a jak ho vytvořit?* Online. Rascasone. 2023. Dostupné z: https://www.rascasone.com/blog/co-je-wireframe-predstavujeme-5-duvodu-proc-je-pro-klienty-drateny-model-dulezity. [cit. 2024-03-03].
[27] *What is UI design?* Online. Figma. 2023. Dostupné z: https://www.figma.com/resource-library/what-is-ui-design/. [cit. 2024-03-03].
[28] *Proč je důležitý UI a UX design?* Online. Skillmea. 2023. Dostupné z: https://skillmea.cz/blog/proc-je-ux-design-dulezity. [cit. 2024-03-03].
[29] DAVIDSON, Louis. Pro SQL Server Relational Database Design and Implementation. Sixth Edition. Apress, 2021. ISBN 978-1-4842-6497-3.
[30] HUGHES, Adam. *Microsoft SQL Server*. Online. TechTarget. 2019. Dostupné z: https://www.techtarget.com/searchdatamanagement/definition/SQL-Server. [cit. 2024-03-03].
[31] *SQL | DDL, DQL, DML, DCL and TCL Commands*. Online. GeeksforGeeks. 2023. Dostupné z: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/. [cit. 2024-03-03].
[32] *What is Transact SQL (T-SQL) and Its Type of Functions?* Online. Simplilearn. 2023. Dostupné z: https://www.simplilearn.com/tutorials/sql-tutorial/transact-sql. [cit. 2024-03-03].
[33] *What is SQL Server Management Studio (SSMS)?* Online. Microsoft Learn. 2023. Dostupné z: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16. [cit. 2024-03-06].
[34] *GitHub Copilot a Visual Studio 2022*. Online. Microsoft Visual Studio. 2024. Dostupné z: https://visualstudio.microsoft.com/cs/. [cit. 2024-03-06].
[35] CALDER, Simon. *What is Project Management Triangle and Why it is Important?* Online. Simon Sez IT. 2024. Dostupné z: https://www.simonsezit.com/article/project-management-triangle/. [cit. 2024-03-06].
[36] WESTLAND, Jason. *Top 10 Project Management Methodologies: An Overview*. Online. Project Manager. 2021. Dostupné z: https://www.projectmanager.com/blog/project-management-methodology. [cit. 2024-03-06].
[37] *Waterfall Methodology: A Complete Guide*. Online. Adobe. 2022. Dostupné z: https://business.adobe.com/blog/basics/waterfall. [cit. 2024-03-06].
[38] *Beginner's Guide to Agile Project Management*. Online. Adobe. 2022. Dostupné z: https://business.adobe.com/blog/basics/agile. [cit. 2024-03-06].

[39] *Freelo*. Online. 2023. Dostupné z: https://www.freelo.io/cs. [cit. 2024-03-13].
[40] *Atlassian*. Online. *2023*. Dostupné z: https://www.atlassian.com/software/jira.[cit. 2024-03-06].

8 Seznam obrázků

Přílohy

Příloha A
Na odkazu se nachází zdrojový kód aplikace, záloha databáze s jedním účtem pro přístup a readme soubor, který obsahuje velmi stručnou nápovědu ke zprovoznění aplikace.
https://github.com/Fanda301/PlanovaniProjektu