Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Webový nástroj pro vedení SW projektů s možností týmové spolupráce

Bc. Miloslav Moravec

Diplomová práce
2019

**Univerzita Pardubice**
**Fakulta elektrotechniky a informatiky**
Akademický rok: **2018/2019**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Miloslav Moravec** |
| Osobní číslo: | **I17216** |
| Studijní program: | **N2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Webový nástroj pro vedení SW projektů s možností týmové spolupráce** |
| Zadávající katedra: | **Katedra softwarových technologií** |

Zásady pro vypracování:

Cílem diplomové práce je navrhnout a implementovat webový softwarový nástroj pro vedení a administraci SW projektů s možností týmové spolupráce. Nejprve bude provedena analýza aktuálně dostupných nástrojů v této oblasti, dojde k jejich komparaci a vyhodnocení požadavků na nově navrhovaný systém. Následně budou rozpracovány případy užití navrhovaného systému s detailním popisem scénářů a ověření několika vybraných klíčových scénářů. Na základě analýzy dojde k implementaci celého systému s využitím technologií MS SQL, ASP.NET MVC s možností využití dalších vhodných nástrojů (např. Javascript apod.).

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

**GALLOWAY, Jon, WILSON Brad, ALLEN K. Scott, MATSON David, Professional ASP.NET MVC 5. Indianapolis, IN: Wrox, a Wiley brand, [2014]. Wrox professional guides. ISBN 9781118794753.**
**SARKA, Dejan, RADIVOJEVIC, Milos. SQL Server 2017 Developer's Guide: A professional guide to designing and developing enterprise database applications [2018]. Packt Publishing; 2nd Revised edition edition. ISBN 9781788476195**
**ARLOW, Jim a Ila NEUSTADT UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9**

Vedoucí diplomové práce: **doc. Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání diplomové práce: **22. října 2018**

Termín odevzdání diplomové práce: **18. května 2019**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 17. listopadu 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 2. 2019

Miloslav Moravec

# PODĚKOVÁNÍ

Děkuji vedoucímu mé diplomové práce, panu doc. Ing. Michaelu Bažantovi, Ph.D., za cenné rady během psaní této práce. Dále bych rád poděkoval své rodině, která mi byla obrovskou oporou během celého studia. V neposlední řadě děkuji své přítelkyni za to, že to se mnou vydržela a byla chápavá v mé práci.

**ANOTACE**

Tato diplomová práce se zabývá softwary na správu projektů v odvětví softwarového inženýrství. Zřetel se též bere i na softwary umožňující práci lidí v týmu. Hlavním cílem praktické části práce je návrh a implementace takového softwaru s využitím frameworku ASP.NET MVC a databáze MS SQL.

**KLÍČOVÁ SLOVA**

ASP.NET MVC, MS SQL, Kanban, UML, SW projekt, správa projektů, týmová spolupráce.

**TITLE**

Web tool for Managing SW projects with team collaboration capabilities

**ANNOTATION**

This diploma thesis deals with project management softwares in the software engineering industry and softwares enabling people to collaborate in a team. The main goal is to create the design and implementation of this software with the use of framework ASP.NET MVC and the database MS SQL.

**KEYWORDS**

ASP.NET MVC, MS SQL, Kanban, UML, SW project, project management, team work.

# OBSAH

# SEZNAM ZKRATEK

| | |
|---|---|
| ASP | Active Server Pages |
| CD | Compact Disc |
| CSS | Cascading Style Sheets |
| DBMS | Database Management System |
| DDL | Data Definition Language |
| DML | Data Manipulation Language |
| EDMX | Entity Data Model XML |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| IBM | International Business Machines |
| IDE | Integrated Development Environment |
| IT | Information Technology |
| JSP | JavaServer Pages |
| LINQ | Language Integrated Query |
| MVC | Model-View-Controller |
| NuGet | New Get \| A new way to get libraries |
| ORM | Object-Relational Mapping |
| PHP | Hypertext Preprocesor |
| RAID | Redundant Array of Independent Disks |
| RDBMS | Relational Database Management System |
| SAN | Storage Area Network |
| SATA | Serial Advanced Technology Attachment |
| SQL | Structured Query Language |
| TOGAF | The Open Group Architecture Framework |
| UML | Unified Modeling Language |
| URL | Uniform Resource Locator |
| WIP | Work In Progress |
| XML | Extensible Markup Language |

# SEZNAM OBRÁZKŮ

# SEZNAM TABULEK

# ÚVOD

Tato práce se zabývá nástroji pro vedení softwarových projektů a nástroji umožňujícími týmovou spolupráci.

Každý softwarový projekt prochází několika fázemi vývoje: sběr požadavků, tvorba analýzy, návrh řešení, implementace, testování a nasazení. Všechny tyto fáze musí být nějakým způsobem dokumentovány. Výsledek z každé fáze je pak předáván jako vstup do fáze nové. Klasickou metodou pro dokumentaci těchto fází (zejména fází požadavků, analýzy a návrhu) jsou tzv. UML diagramy, jejíž účelem je graficky znázornit informace o projektu.

Práce v týmu se odvíjí od zvolené metodiky vývoje softwaru a s tou je pak spjatý i nástroj, který tým pro svou práci využívá. Neexistuje jednotný nástroj, který by vyhovoval všem cílovým skupinám. Každý nástroj je předurčený nějaké metodice a podle ní si jej pak týmy vybírají.

Praktická část této práce se snaží oprostit fázi požadavků a analýzy od UML diagramů a naopak se snaží zpracovávat projektové informace ve formě přehledných tabulek uchovávaných v databázi. Správa těchto informací je dále zastřešena pomocí webové aplikace umožňující vytváření týmů z přihlášených uživatelů. Jelikož tato aplikace nechce být nějak pevně spjatá s jakoukoliv metodikou vývoje softwaru, týmová spolupráce probíhá pouze v zadávaní jednoduchých úkolů mezi členy projektu.

Tato webová aplikace je vytvořena pomocí webového aplikačního frameworku ASP.NET MVC jehož použití a funkcionalita jsou popsány v rámci použitých technologií. Pro uchovávání dat celé aplikace byla použita databáze MS SQL, která je vysvětlena rovněž. Pro snadnou konverzi dat mezi databází a webovou aplikací bylo využito objektově-relačního mapování, které poskytl Entity Framework.

V této práci se též pojednává o testování webových aplikací, resp. uživatelského rozhraní, a to pomocí nástroje Selenium. Poslední kapitolou této práce je obsáhlá uživatelská příručka vysvětlující obsluhu výsledné aplikace.

### Typografické konvence

V této práci se používají následující typografické konvence:

*Kurzíva*

Používá se pro vyznačení názvů objektů a anotací.

`Styl písma Courier New`

Slouží k vyznačení názvů tříd.

# 1 VYMEZENÍ A CÍL PRÁCE

Cílem této práce je vytvoření webové aplikace vhodné pro správu softwarových projektů v týmových skupinách. Aplikace bude umožňovat v rámci svého prostředí vytvoření počáteční analýzy softwarového projektu. Dále bude umožňovat správu externích souborů týkajících se daného projektu.

Pokrok ve vývoji projektu bude zajištěn pomocí vzájemného zadávání úkolů vycházejícího z agilní metodiky Kanban.

## 1.1 Základní pojmy

### Softwarový projekt

Projekt je posloupností jedinečných, komplexních a souvisejících činností, které mají jeden cíl nebo účel a které musí být dokončeny v určitém čase, v rámci rozpočtu a dle specifikací. [1]

Posloupnost činností je založena na technických požadavcích, nikoliv na výsadách vedení. Pro určení sekvence je užitečné popřemýšlet o následujících vstupech a výstupech: [1]

- Co bude vstupní informací pro započetí této činnosti?
- Co bude výstupem z této činnosti?

Výstup z jedné aktivity nebo souboru činností se stává vstupem do jiné činnosti nebo souboru činností. Určení posloupnosti na základě omezování zdrojů nebo prohlášení typu: „Petr bude pracovat na aktivitě B, jakmile dokončí práci na aktivitě A.“ je třeba se vyhnout, protože to vytváří umělý vztah mezi činnostmi. Co když Petr nebude vůbec k dispozici? Omezování zdrojů není ignorováno při skutečném plánování aktivit. Rozhodnutí o tom, jaké prostředky použít a kdy je použít, přichází později v procesu plánování projektu. [1]

### Webová aplikace

Webová aplikace je počítačový program, který využívá webové prohlížeče a webové technologie k provádění úkolů přes internet. [2]

Webové aplikace používají kombinaci skriptů na straně serveru a skriptů na straně klienta. Skripty na straně serveru (např. PHP, ASP…) se zabývají ukládáním a vyhledáváním informací. Skripty na straně klienta (JavaScript, HTML) pak tyto informace poskytují klientům. To umožňuje uživatelům komunikaci s cílovou společností pomocí on-line formulářů. [2]

Webová aplikace vyžaduje, aby webový server dokázal spravovat požadavky od klienta, dokázal využívat aplikační server k provádění daných úkolů, včetně spolupráce s databází pro ukládání informací. Technologie aplikačního serveru se pohybuje od ASP.NET až po PHP či JSP. [2]

Typický průběh požadavku v rámci webové aplikace probíhá následovně: [2]

1. Uživatel zašle požadavek na webový server prostřednictvím webového prohlížeče.
2. Webový server předá tuto žádost příslušnému serveru webových aplikací.
3. Aplikační server provede požadovaný úkon – například dotazování v databází či zpracování nějakých dat. Poté vygeneruje výsledky požadovaných dat.
4. Aplikační server odesílá výsledky na webový server.
5. Webový server odpovídá klientovi skrze webový prohlížeč, na kterém se zobrazí požadované informace.

### DBMS

DBMS je databázový program. Z technického hlediska jde o softwarový systém, který používá standardní metodu katalogizace, vyhledávání a spouštění dotazů nad daty. DBMS spravuje příchozí data, organizuje je a poskytuje způsoby úpravy nebo extrahování dat uživateli či jinými programy. [3]

DBMS například zahrnují: MySQL, PostgreSQL, SQL Server či Oracle SQL. Vzhledem k tomu, že je k dispozici mnoho systémů pro správu databází, je důležité, aby existoval způsob komunikace mezi nimi. Z tohoto důvodu je většina databázových softwarů dodávána s ovladačem Open Database Connectivity (ODBC), který umožňuje integraci databáze s jinými. [3]

### RDBMS

RDBMS je DBMS navržené speciálně pro relační databáze. Relační databáze odkazuje na databázi, která ukládá data ve strukturovaném formátu pomocí řádků a sloupců. Umožňuje snadné vyhledávání a přístup ke konkrétním hodnotám v databázi. Databáze je označena jako relační, protože hodnoty v každé tabulce jsou stejného charakteru. Tabulky se též mohou spojovat s jinými tabulkami. Relační struktura umožňuje spouštět dotazy na více tabulkách najednou. [4]

Nejznámější DBMS aplikace spadají do kategorie RDBMS. Patří sem Oracle, MySQL, Microsoft SQL Server či IBM DB2. Příklady nerelačních databází zahrnují Apache HBase, IBM Domino a Oracle NoSQL Database. [4]

### Trigger

Trigger je speciální typ procedury, která se automaticky spouští při vzniku události na databázovém serveru. DML triggery se spustí, když se uživatel pokusí upravit data v tabulce pomocí DML příkazu (SELECT, INSERT, UPDATE, DELETE). DDL triggery se spouštějí v reakci na různé události vyvolané pomocí DDL příkazu (CREATE, ALTER, DROP). [5]

## 1.2 Životní cyklus vývoje softwaru

Tato podkapitola je inspirována zdrojem [6]. Životní cyklus vývoje systému (SDLC – System Development Life Cycle) je terminologie používaná k vysvětlení jakými kroky a fázemi software projde, než se dostane ke konečnému zákazníkovi. Tyto kroky přeměňují software z prvotní myšlenky na konečný produkt.

Povolání „softwarový vývojář“ existuje již od doby prvních počítačů a od stejné chvíle se začaly rozvíjet metody pro vývoj softwaru. Tyto metody se postupně přizpůsobují současnému počítačovému hardwaru, vývojovým nástrojům, a dokonce schopnosti organizovaného řízení týmů pro vývoj softwaru. Přístupy mohou mít tyto metody odlišné, ale cíl je vždy jasný: vyvíjet software levně a hlavně efektivně.

Software je komplexní produkt, který je vyvinut a dodáván prostřednictvím řady kroků. To je jediná věc, kterou mají všechny metody společné. Software, stejně jako všechny produkty, začíná jako nápad. Nápad se pak stává dokumentem, nebo prototypem, v závislosti na použité metodě. Každý dokument, schéma, část softwaru nebo jiný artefakt vytvořený v rámci jednoho kroku, se stává zároveň vstupem do kroku dalšího. Na konci celé posloupnosti kroků je software připravený na předání zákazníkovi. Sekvence těchto kroků používaných jednotlivými metodami se označuje jako Životní cyklus vývoje softwaru (SDLC).

Je velmi obtížné provádět komplexní týmové úsilí (jako je vývoj softwaru) bez nějakého plánu. Každá metodika vývoje softwaru je plánový rámec pro vývoj softwaru. Existuje mnoho diskusí o tom, jaká metoda je celkově nejlepší, či která je nejlépe vhodná pro určitý typ softwaru a jak měřit úspěch ve vývoji softwaru. Jedna věc je ovšem jistá: každý plán je lepší, než žádný plán. Nejlépe tuto myšlenku vystihl Benjamin Franklin, když řekl: „If you fail to plan, you are planning to fail.“. Bez strukturovaného plánu mají vývojové týmy tendenci přecházet do „stáda koček“. Vývojáři nevědí, co by měli vytvořit. Projektoví manažeři nemají ponětí o tom, jakého pokroku jejich tým právě dosáhl. Bez plánu nemá podnik ani šanci rozhodnout, zda konečný produkt splňuje všechny jejich požadavky.

Formálně definovaná metoda pro vývoj softwaru ve formě SDLC dosahuje řady výhod: (i) společná slovní zásoba pro každý krok, (ii) definované komunikační kanály mezi vývojovými týmy a zúčastněnými stranami, (iii) jasné role a odpovědnosti mezi vývojáři, projektanty, obchodními analytiky a projektovými manažery, (iv) jasně definované vstupy a výstupy z jednoho kroku do druhého, (v) deterministická definice úplnosti, která může být použita k potvrzení, zda je krok skutečně splněný.

### 1.2.1 Fáze SDLC

Následující fáze jsou zhruba shodné u všech metod. Vyskytují se zpravidla v tomto pořadí, ačkoli mohou být i kombinovány tak, že se několik kroků provádí paralelně.

Agilní metodiky mají tendenci tyto kroky vmáčknout do jednoho rychle se opakujícího cyklu. Metody vodopádu pak procházejí každý z těchto kroků postupně pomocí principu, kdy výstup z jednoho kroku se stává automaticky vstupem do kroku následujícího.

### Plánování

Fáze plánování zahrnuje aspekty řízení projektu a produktu. Mezi tyto aspekty může patřit:

- přidělování zdrojů (lidských i materiálových),
- plánování kapacit,
- plánování projektu,
- přibližné vyčíslení konečné ceny,
- zajištění práv a bezpečnostních opatření.

Výstupem plánovací fáze jsou plány projektu, odhady nákladů a požadavky na zakázku. V ideálním případě spolupracují týmy projektových manažerů a vývojářů s operačními a bezpečnostními týmy, aby zajistili, že budou zastoupeny všechny perspektivy.

### Požadavky

Podnik musí komunikovat s IT týmy, aby jim mohl přednést své požadavky na nový rozvoj a vylepšení. Fáze požadavků pak tyto požadavky shromažďuje. O požadavcích zpravidla s IT týmy komunikuje vždy odborník na danou podnikovou problematiku, aby jim požadované vylepšení co nejlépe vysvětlil.

Architekti, vývojové týmy a výrobní pracovníci spolupracují s malými a středními podniky na dokumentaci obchodních procesů, které je třeba automatizovat pomocí softwaru. Výstup této fáze je v metodikách vodopádového typu dokument, který všechny tyto požadavky shromažďuje. Naopak agilní metodiky mohou vytvářet seznam nevyřízených úkolů, které mají být dokončeny.

### Návrh

Jakmile jsou požadavky pochopeny, softwaroví architekti a vývojáři mohou začít vytvářet software. Proces návrhu používá zavedené vzorce pro aplikační architekturu a vývoj softwaru. Architekti mohou použít architekturní rámec, jako je TOGAF, k sestavení aplikace z existujících komponent podporujících opakované použití a standardizaci.

Vývojáři používají osvědčené návrhové vzory pro konzistentní řešení algoritmických problémů. Tato fáze může také zahrnovat tvorbu rychlých prototypů, jejichž řešení se porovnává a hledá se to nejlepší. Výstup této fáze zahrnuje:

- Dokumenty obsahující seznam návrhových vzorů a komponentů vybraných pro projekt;
- Prototypový kód představující základní kámen pozdějšího vývoje;

### Vývoj softwaru

Tato fáze vytváří vývojový software. V závislosti na metodologii může být tato fáze prováděna buď v tzv. sprintech (agilní metodiky), nebo může probíhat jako jeden blok úsilí (vodopád). Bez ohledu na metodiku by vývojové týmy měly vytvářet software co nejrychleji. Zainteresované strany v oblasti podniku by měly být pravidelně zapojovány, aby překontrolovaly, zda jsou plněna jejich očekávání. Výstupem této fáze je testovatelný funkční software.

### Testování

Testovací fáze je pravděpodobně jedna z nejdůležitějších. Není možné dodávat kvalitní software bez testování. Existuje široká škála testů potřebných ke změření kvality:

- testy kvality kódu,
- jednotkové testy,
- integrační testy,
- výkonnostní testy,
- zabezpečovací testy,
- testy uživatelského rozhraní.

Nejlepším způsobem, jak zajistit, aby testy probíhaly pravidelně a nikdy nebyly vynechány, je jejich automatizace. Testy lze automatizovat pomocí nástrojů kontinuální integrace. Výstupem testovací fáze je funkční software, připravený k nasazení do výrobního prostředí.

### Nasazení

Fáze nasazení je v ideálním případě vysoce automatizovaná fáze. U podniků vyšší úrovně je tato fáze téměř neviditelná; software je nasazen v okamžiku, kdy je připraven. U podniků nižší úrovně, nebo v některých vysoce regulovaných odvětvích, tento proces zahrnuje některá manuální schválení. I v tomto případě je však nejlepší, aby bylo samotné nasazení plně automatizováno. Nástroje automatického nasazování aplikací (ARA – Application Release Automation) se používají ve středních a velkých firmách k automatizaci zavádění aplikací do produkčního prostředí. Systémy ARA jsou obvykle integrovány s nástroji pro kontinuální integraci. Výstupem této fáze je uvolnění fungujícího softwaru do produkce.

### Provoz a údržba

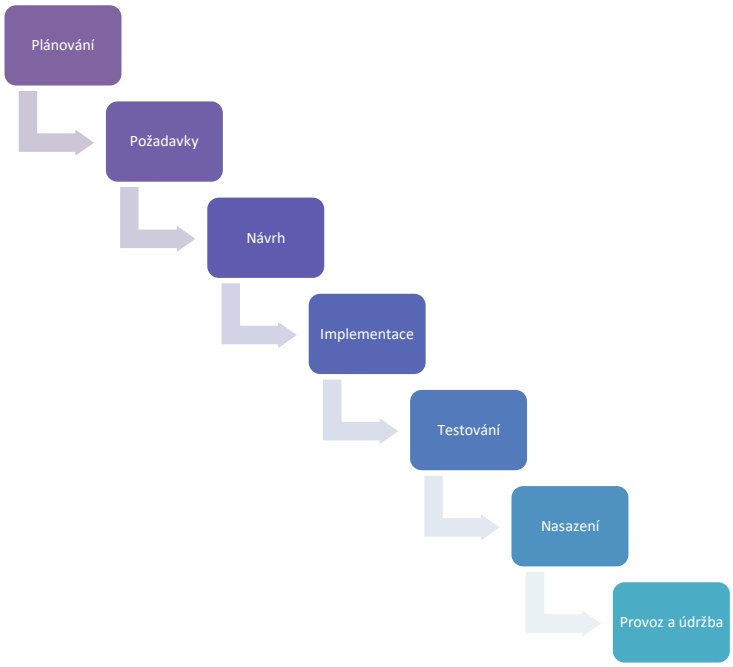
Fáze provozu a údržby je tzv. „konec začátku“. Životní cyklus vývoje softwaru zde nekončí. Software musí být neustále sledován k zajištění řádného provozu. Chyby a závady objevené za běhu aplikace musí být hlášeny a řešeny, což často vrací software zpět do výroby. Opravy chyb nemusí projít znovu celým cyklem, je ale nutný alespoň zkrácený proces, který zajistí, že oprava s sebou nepřináší jiné problémy.

### 1.2.2 Modely SDLC

V následujícím textu budou představeny vybrané modely SDLC, a to konkrétně vodopád [7], iterativní model [8], spirála [9] a agilní model [6].

#### Vodopád

Model vodopád poprvé popsal v roce 1970 počítačový vědec Winston Walker Royce ve svém článku *Managing the Development of Large Software Systems*. Jedná se o posloupnost fází (popsanou výše), které na sebe striktně navazují s důrazem na to, zda aktuální fáze byla úplně dokončena. Jinými slovy, do další fáze nelze přejít, dokud aktuální fáze není hotová. Obrázek 1 znázorňuje princip této posloupnosti, která směřuje vždy jedním směrem právě jako vodopád.




**Obrázek 1: Vodopádový model**

V běžné praxi odpovídá model vodopádu časovému plánu projektu, kde 20–40 % času je věnováno plánování, sběru požadavků a tvorby analýzy, 30–40 % času implementaci a zbytek času je investován do testování a nasazení výsledného produktu. Samotná organizace projektu musí být vysoce strukturovaná. Většina středních a velkých projektů obsahuje podrobný soubor postupů a kontrol regulující všechny procesy v rámci projektu.

Vodopád je často považován za klasický přístup při vývoji softwaru. Stal se základem pro provádění vysoce předvídatelných projektů s dobře definovanými a zřídka se měnícími požadavky. Tento model je vhodný pro aplikace ve vojenských, zdravotnických či finančních odvětvích; zkrátka pro většinu systémů řídících průmyslové procesy.

Avšak projekty na vývoj softwaru jsou vždy velmi různorodé a metodologie vodopádu, která je založena na striktním plánu, moc dobře nereaguje na agresivní časově citlivé projekty bez předem jasných požadavků. Na tyto projekty je pak třeba použití pružnějšího přístupu, který

zajistí rychlejší uvedení na trh a přizpůsobení se rychle se měnícím požadavkům. Metodiky s takovýmto přístupem se nazývají agilní a budou představeny na konci této kapitoly.

Člověk si může myslet, že vodopád je zcela zastaralá metodika, ale není to tak úplně pravda. Model vodopádu má své výhody a může být úspěšně využit pro kritické projekty, u nichž se pravděpodobně nemění požadavky.

Mezi hlavní výhody vodopádu patří:

- Logická struktura, která je snadno srozumitelná a spravovatelná.
- Každá fáze má jasně definované výstupy, které je třeba přezkoumat a schválit.
- Požadavky jsou podrobně zdokumentovány a zůstávají nezměněny v průběhu celého procesu.
- Důkladná dokumentace usnadňuje rychlejší pochopení problematiky pro nové vývojáře.
- Tento model je vhodný pro vývojový proces zaměřený na mezníky a poskytuje jasný časový plán pro projekt.
- Rozsáhlé plánování zajišťuje předvídatelnější konečné výsledky z hlediska rozpočtu a rozsahu.

Mezi hlavní nevýhody vodopádu patří:

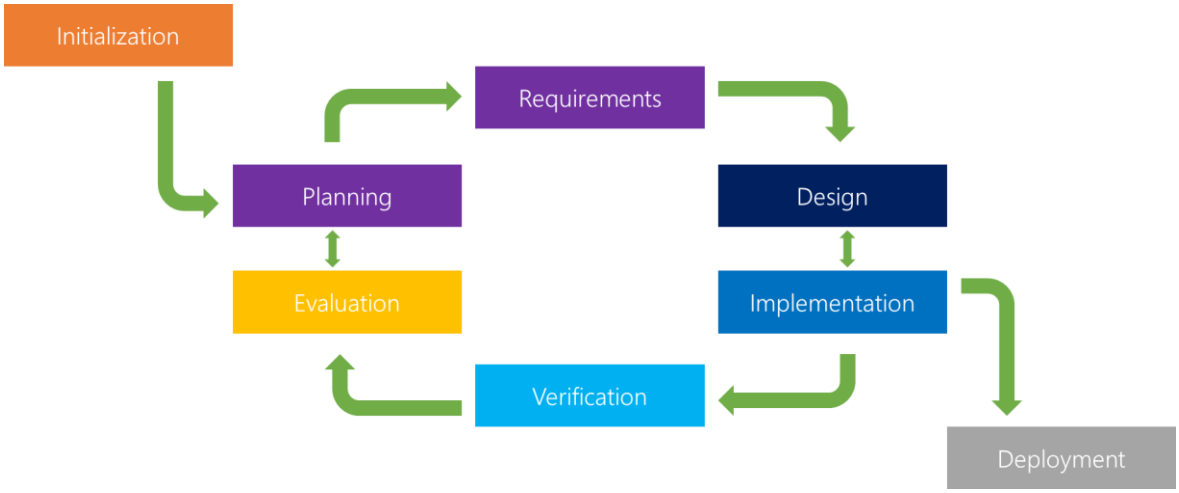
- Nedostatečná flexibilita – lineární posloupnost fází ztěžuje možné návraty zpět ve vývoji.
- Nalezení chyby v požadavcích či v návrhu ve fázi testování rapidně zvyšuje náklady na její odstranění.
- Model může být špatnou volbou pro projekty citlivé na čas nebo pro ty, které nemají dobře definované požadavky.
- Silný důraz na kontrolu činí tento model nevhodný pro prostředí s vysokou mírou nejistoty či rizikem.

### Iterativní model

Iterativní model se zaměřuje na menší implementace softwaru již od prvních fází vývoje, které postupně na sebe nabalují více a více složitosti s širší funkcionalitou. Tento proces nabalování trvá tak dlouho, dokud není výsledný software kompletní.

Na rozdíl od tradičního vodopádového modelu, který se zaměřuje na přísný postupný vývoj, je iterativní model považován za cyklický proces. Po počáteční fázi plánování se cyklicky opakuje několik menších kroků, přičemž každé dokončení cyklu představuje nějaké

vylepšení či zvětšení softwaru. Vylepšení lze rychle rozpoznat a v každé iteraci implementovat, což dovolí další iteraci být alespoň o něco lepší než ta poslední.




**Obrázek 2: Iterativní model [8]**

Prvním krokem v každém cyklu je plánování společně se sběrem požadavků. Jedná se o přípravu na nadcházející fázi cyklu, kdy se podrobně určí, co má být v tomto cyklu splněno. Jakmile je toto dokončeno provádí se analýza a návrh. V této části cyklu se vytváří příslušné obchodní logiky, databázové modely a jiné artefakty, které budou v další fázi cyklu potřebné. Následuje krok, ve kterém všechny dosavadní plánovací dokumenty, specifikace a návrhy jsou kódovány a implementovány do projektu. V dalším kroku se pak tato implementace testuje a hledají se možné chyby, které se po implementaci mohli vyskytnout. Po dokončení všech předchozích kroků následuje vyhodnocení celkového vývoje včetně právě dokončené fáze. V této chvíli se tým společně s klienty sejdou a přezkoumají, v jakém bodě se projekt nachází a kam by ho chtěli posunout dál. Tento cyklus se opakuje, dokud není v posledním vyhodnocovacím kroku projekt prohlášen za dokončený a nepřistoupí se k jeho nasazení.

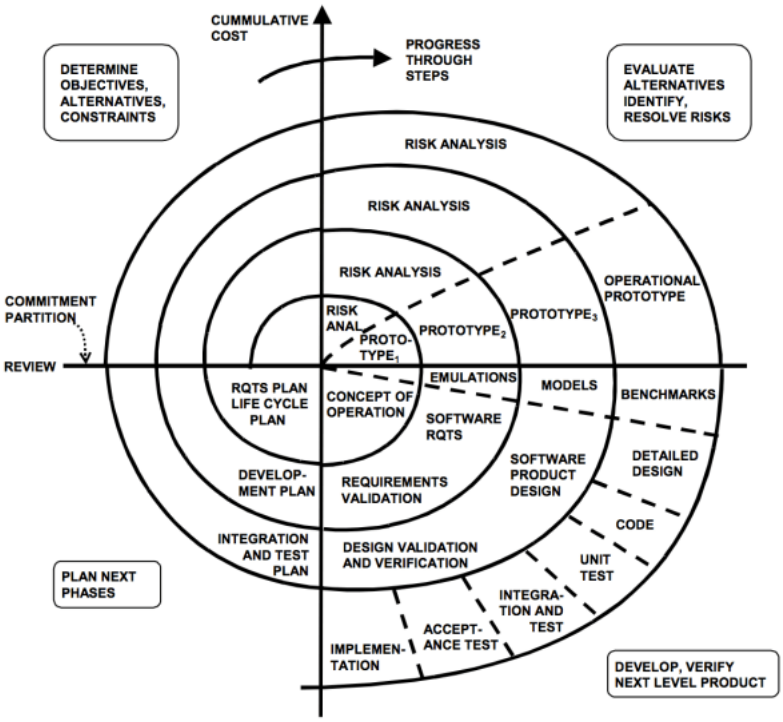
### Spirála

Spirálový model, který poprvé popsal v roce 1986 Barry Boehm, je jedním z nejdůležitějších modelů SDLC poskytující podporu pro řízení rizik. Ve své schematické reprezentaci vypadá jako spirála s mnoha smyčkami. Přesný počet smyček spirály není znám a může se lišit dle projektu. Každá smyčka spirály představuje jednu fázi procesu vývoje softwaru. Přesný počet fází potřebných k vývoji produktu může být změněn vedoucím projektu v závislosti na riziku projektu. Protože projektový manažer dynamicky určuje počet fází, je jeho role při vývoji pomocí spirálového modelu obzvláště důležitá.

Rádius spirály v jakémkoliv bodě představuje dosavadní náklady projektu; úhel pak znázorňuje dosavadní pokrok v dané fázi. Na následujícím obrázku (Obrázek 3) jsou znázorněny různé fáze spirálového modelu.

Každá fáze spirálového modelu je rozdělena do čtyř kvadrantů, které po sobě následují ve směru hodinových ručiček. Pro lepší pochopení budou tyto kvadranty označeny dle popisu na obrázku.

1. **Determine objectives, alternatives, constraints:** Na začátku každé fáze se identifikují cíle, které by se měly v této fázi splnit. Tyto cíle se analyzují a navrhují se zároveň i možná alternativní řešení.
2. **Evaluate alternatives identify, resolve risks:** Během druhého kvadrantu se všechna řešení vyhodnocují a vybírá se to nejlepší. Poté se vyhodnocují rizika spojená s tímto řešením, která se zároveň eliminují.
3. **Develop, verify next level product:** V rámci třetího kvadrantu se vybrané řešení implementuje a ověřuje pomocí testování.
4. **Plan next phases:** Čtvrtý kvadrant slouží k naplánování další fáze.




**Obrázek 3: Spirálový model [10]**

Jak již bylo zmíněno, spirálový model poskytuje podporu pro řízení rizik. Rizikem je každá nepříznivá situace, která by mohla ovlivnit úspěšné dokončení softwarového projektu. Nejdůležitějším úkolem při vývoji pomocí spirálového modelu je odhalení zatím neznámých rizik hned po zahájení projektu. Model spirály podporuje zvládnutí rizik tím, že poskytuje prostor pro vytváření prototypů v každé fázi vývoje softwaru. Liší se tak velmi od vodopádového modelu, kde první prototyp spatří světlo světa až v konečných fázích vývoje.

Model spirály je též nazýván jako Meta Model, protože zahrnuje všechny ostatní modely SDLC. Například jediná smyčka spirály vlastně reprezentuje iterativní vodopádový model. Model spirály zahrnuje postupný přístup klasického vodopádového modelu. Používá též přístup prototypového modelu tím, že na počátku každé fáze vytvoří prototyp jako techniku zpracování rizik. Také lze model spirály považovat za podporu evolučního modelu – iterace po spirále mohou být považovány za evoluční úrovně, díky nichž je celý systém postaven.

### Agilní model

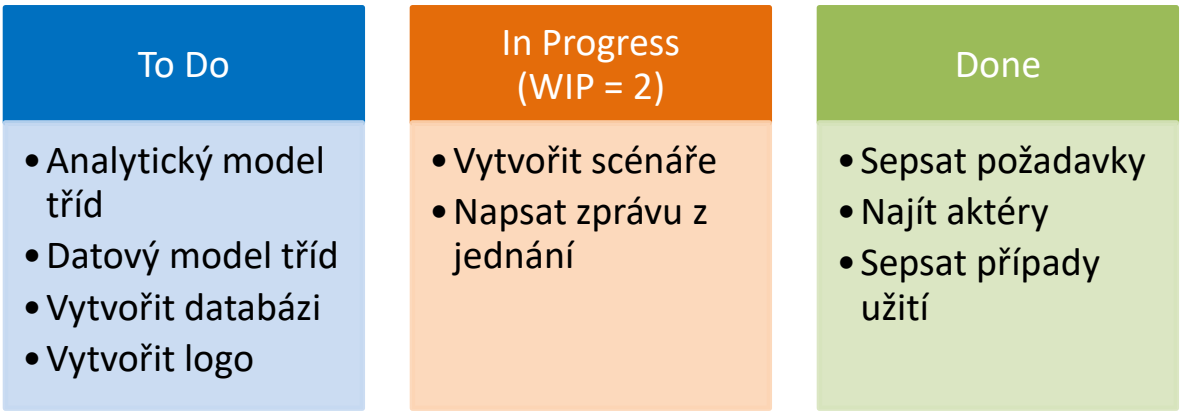
Manifest agilního vývoje softwaru byl navržen a podepsán skupinou vývojářů v roce 2001. Manifest se zabývá klíčovými problémy vodopádu, které vedly k problémům při poskytování softwaru. Tam, kde je vodopád jednosměrná cesta, je agilní model flexibilní rámec, který umožňuje nejistotu. Agilní model zdůrazňuje týmovou práci, prototypy a zpětné vazby, které mohou změnit směr vývoje v reakci na měnící se požadavky.

Agilní model se od doby podepsání manifestu rozrostl na několik jeho variant. Patří sem například Scrum, který definuje konkrétní role a události, jako součást svých praktik nebo Kanban, který je jednodušší a má méně předpisů s větší flexibilitou. Agilní týmy často tyto varianty kombinují, aby přizpůsobili proces vývoje tomu, který jim vyhovuje lépe. V dalším textu budou varianty Scrum a Kanban vysvětleny podrobněji.

Scrum je iterativní, přírůstkový framework pro projekty a vývoj softwaru. Struktura vývoje se dělí na cykly nazývané sprinty. Každý sprint není delší než jeden měsíc a plynule navazuje na předchozí bez sebemenší přestávky. Sprinty jsou časově rozvržené – končí určitým datem, ať byla práce dokončena, či ne. Na začátku každého sprintu vybírá tým položky ze seznamu požadavků, který je seřazen dle priority. Zároveň se tým zavazuje ke splnění vybraných požadavků do konce daného sprintu. Během sprintu se vybrané položky již nemění. Každý den se tým krátce shromáždí, aby zhodnotil své dosavadní pokroky a navrhl další kroky potřebné k dokončení zbývající práce. Na konci sprintu dojde k vyhodnocení výsledků dosavadní práce společně s klientem, od kterého dostane tým zpětnou vazbu, která může být začleněna do dalšího sprintu. Scrum se vyznačuje fungujícím produktem na konci sprintu, který je skutečně „hotový“; v případě softwaru to znamená kód, který je integrován, plně testován a potenciálně schopný nasazení. [11]

Kanban je metoda pro správu tvorby produktů s důrazem na nepřetržité dodávky, aniž by byl přetížen vývojový tým. Kanban je založen na třech základních principech: [12]

- **Vizualizujte, co dnes děláte:** vidět všechny položky v kontextu mezi sebou může být velmi informativní.
- **Omezte množství probíhající práce (WIP):** pomáhá vyvážit přístup založený na průtoku, týmy by neměly dělat spousty odlišných úkolů najednou.
- **Vylepšujte průtok:** Když je něco dokončeno, další úkol ze seznamu nevyřízených (s nejvyšší prioritou) je započat.




**Obrázek 4: Kanban Board**

Obrázek 4 znázorňuje tzv. Kanban Board neboli klasickou tabuli, která obsahuje tři sloupce: To Do, In Progress a Done. Sloupec To Do představuje prioritní frontů úkolů, které je třeba vyřešit. Pořadí úkolů ve frontě se může lišit dle priorit nově přicházejících úkolů. Druhý sloupec In Progress obsahuje úkoly, na kterých se momentálně pracuje. U každého úkolu může být i informace o tom, kdo na úkolu pracuje a jak dlouho se už úkol v tomto sloupci nachází. Poslední sloupec Done pak zobrazuje seznam úkolů, které již byly dokončeny.

Za zmínku stojí, že prostřední sloupec má přiřazenou konstantu WIP s hodnotou 2. WIP je zkratka pro limit „Work In Progress“ neboli, že v jednom okamžiku lze maximálně pracovat jen na dvou úkolech na jednou, aby bylo dosaženo vyváženého průtoku úkolů. Po dokončení úkolu se tento přesouvá do sloupce Done a tým, který na úkolu pracoval si buď vezme první úkol ze sloupce To Do, nebo pomůže týmu, který již delší dobu pracuje na nějakém úkolu.

# 2 NÁSTROJE PRO TVORBU UML DIAGRAMŮ

Každý projekt začíná se souhrnem požadavků a uvědoměním si, co vlastně od požadovaného softwaru očekáváme. Kromě požadavků se dále sepisují aktéři, případy užití, tvoří se scénáře a konečně analytické a datové modely tříd.

Pro přehlednou správu a uchovávání těchto informací slouží nástroje, ve kterých se dají modelovat tzv. UML diagramy. UML (Unified Modeling Language) je univerzální jazyk pro modelování systémů. Poskytuje vizuální syntaxi, která se využívá při sestavování jednotlivých modelů. Jazyk UML je využíván mnoha metodikami jako jejich vlastní syntaxe pro vizuální modelování. [13]

V následujícím textu bude představeno několik nástrojů pro modelování UML diagramů.

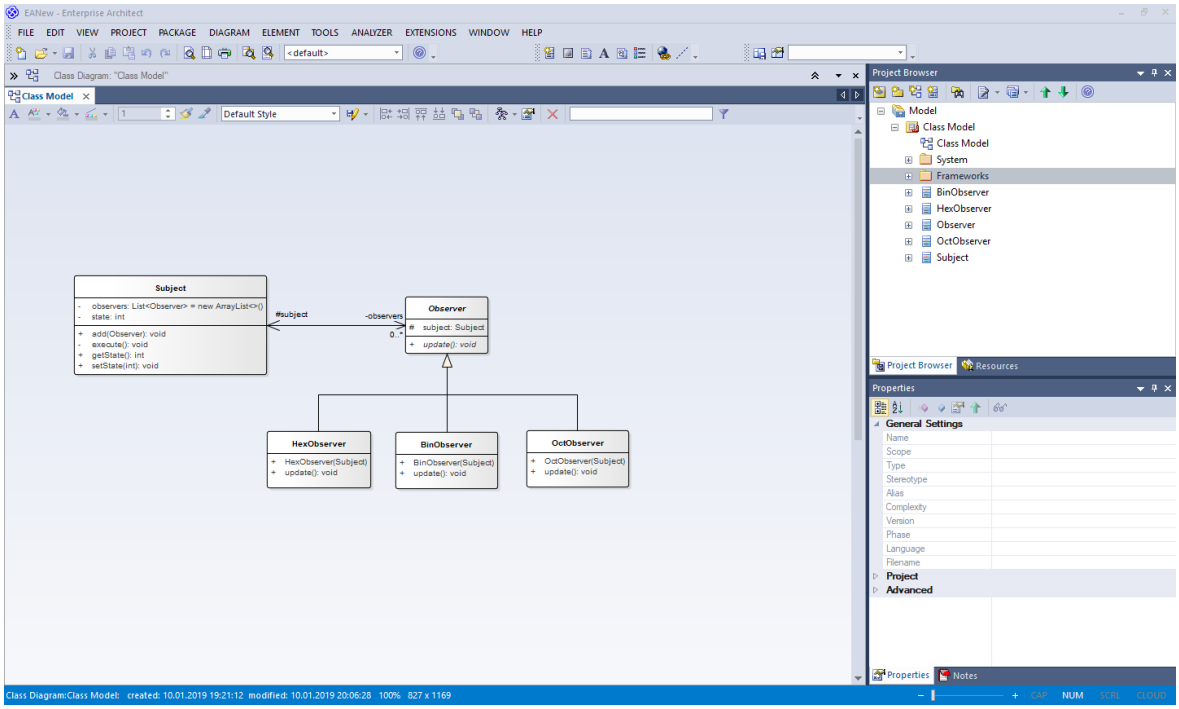
## 2.1 Enterprise Architect

Enterprise Architect (dále jen „EA“) je komplexní UML nástroj pro analýzu a návrh softwaru. Mezi jeho základní vlastnosti patří: [14]

- modelování business procesů,
- shromažďování požadavků prostřednictvím analýzy, návrhu a konstrukce,
- plná sledovatelnost systému,
- intuitivní rozhraní,
- škálovatelnost,
- víceúčelovost,
- výkonnost či
- integrita.

Modelování ale není to jediné, co EA nabízí. Obrovským bonusem tohoto programu je jeho integrita s jinými programy (Microsoft Office, Microsoft Visual Studio, Eclipse a další) či programovacími jazyky. EA dokáže například ze zdrojového kódu sám sestavit přesný diagram tříd. A naopak z digramu tříd předepsat zdrojový kód (samozřejmě bez vyplněných metod a funkcí).

Následovat nyní bude představení uživatelského rozhraní. Na následujícím obrázku (Obrázek 5) se nachází rozložení celého prostředí, které se skládá z horní lišty obsahující nabídku elementů vhodných do aktuálního modelu. Dále pak z panelu Project Browser, který zobrazuje stromovou hierarchii aktuálně otevřeného projektu se všemi soubory a modely, které obsahuje. Pod tímto panelem se dále nachází panel Properties zobrazující detailní informace k aktuálně vybrané položce. Konečně hlavní okno, které je uprostřed celého prostředí, slouží k modelování vytouženého diagramu. V případě otevření více diagramů se dělí pak toto okno na záložky, mezi kterými lze proklikávat.



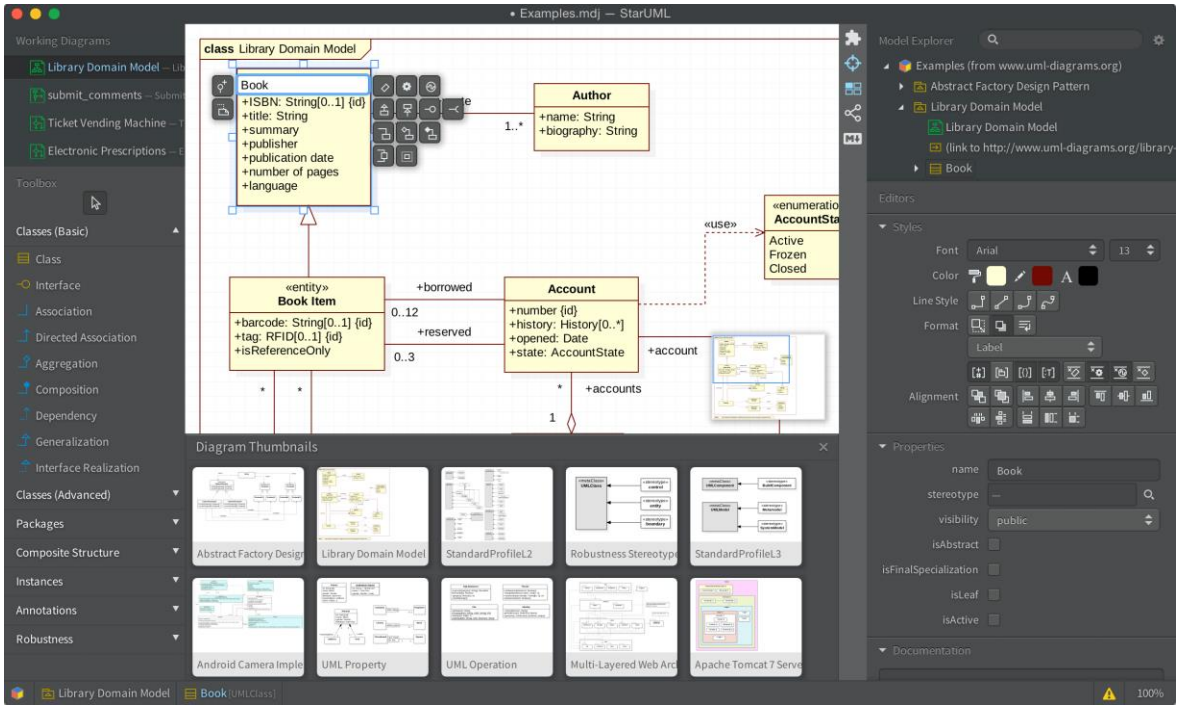

**Obrázek 5: Enterprise Architect**

## 2.2 StarUML

Dalším nástrojem, který bude představen jako pomocník při modelování UML diagramů, je StarUML společnosti MKLab. Je to sofistikovaný softwarový modelovací nástroj zaměřený na podporu agilního a stručného modelování. Základní vlastnosti tohoto programu jsou: [15]

- podpora pro více platforem (MacOS, Windows i Linux),
- tvorba ERD diagramů (Entity-Relationship Digram),
- tvorba DFD diagramů (Data-Flow Diagram),
- tvorba vývojových diagramů,
- otevřené API,
- podpora exportování dat do HTML či
- možnost rozšíření programu od třetích stran.

Na obrázku 6 se nachází uživatelské prostředí programu StarUML. Skládá se opět z několika panelů a jednoho hlavního okna uprostřed obrazovky. Toto okno ale nemá „záložkovitý“ charakter jako tomu bylo u EA. Zde se otevřené diagramy přepínají pomocí panelu Working Diagrams, který se nachází v levém horním rohu obrazovky. Pod ním se nachází Toolbox obsahující elementy potřebné pro vytváření diagramů. Pravá strana programu je téměř stejná jako u EA. Opět se zde nachází panel zobrazující stromovou strukturu našeho projektu (Model Explorer) a panel s podrobnostmi a nastavením právě zvoleného elementu.




**Obrázek 6: StarUML [15]**

Tento panel se dále dělí na odvětví stylů, vlastností a dokumentace. Poslední dolní panel Diagram Thumbnails nabízí předlohy vybraných modelů, na kterých může uživatel začít stavět nebo se jimi inspirovat.

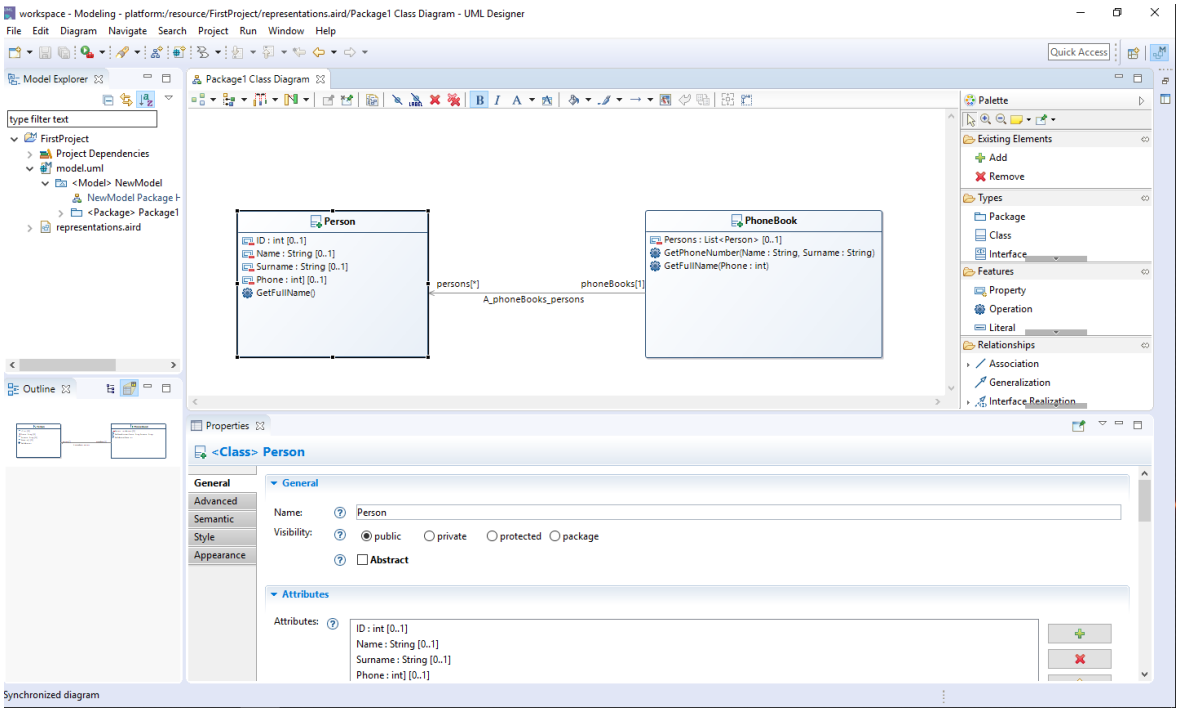
## 2.3 UML Designer

Poslední nástroj, který bude zmíněn je UML Designer společnosti Obeo. Je založen na vývojovém prostředí Eclipse a Sirius. Tento program poskytuje sadu diagramů pro práci s modely UML. Program je typu open-source a umožňuje stejně jako StarUML rozšíření funkcionalit od třetích stran. I u tohoto nástroje budou zmíněny hlavní vlastnosti a funkce.

UML Designer poskytuje: [16]

- hierarchii balíčků,
- diagram tříd,
- diagram komponentů,
- diagram případů užití,
- diagram aktivit,
- sekvenční diagram a mnoho dalších.

Jak je vidno z následujícího obrázku (Obrázek 7) uživatel, který se již někdy setkal s prostředím programu Eclipse, se brzy zorientuje i v tomto.




**Obrázek 7: UML Designer**

Panel nacházející se na levé straně se nazývá Model Explorer a jak bývá zvykem, zobrazuje stromovou strukturu právě otevřeného projektu. Naopak panel na pravé straně (Palette) nabízí doslova paletu různých elementů, ze kterých se později skládá výsledný model. Panel Properties nacházející se ve spodu obrazovky umožňuje doplňovat informace o právě vybraném elementu. A konečně uprostřed obrazovky se nachází hlavní pracovní okno, zde opět „záložkovitého“ charakteru, ve kterém se požadovaný model vytváří.

## 2.4 Srovnání UML nástrojů

V předchozím textu byly představeny některé nástroje umožňující tvorbu UML diagramů. Nyní je na čase provést analýzu a srovnání těchto programů. Zejména budou sledovány nabízené funkcionality a uživatelská přívětivost. Všechny tyto aspekty jsou zahrnuty v následující tabulce (Tabulka 1).

Z tabulky je zřejmé, že nástroj Star UML takřka všechny požadavky na tvorbu UML diagramů splňuje v plné míře. Dokonce je dostupný na všech platformách a jeho zkušební verze je neomezená. Na rozdíl od konkurenčního nástroje Enterprise Architect, který je drahý a dostupný pouze na platformě Windows. Na poslední pozici se umístil UML Designer, který je sice zadarmo a dostupný na všech platformách, ale pár požadovaných funkcionalit mu chybí.

**Tabulka 1: Srovnání UML nástrojů**

| | Enterprise Architect | StarUML | UML Designer |
|---|---|---|---|
| **Zkušební verze** | Dostupná na 30 dní. | Dostupná neomezeně, ale bez některých funkcionalit. | Nepotřebná. |
| **Integrace s jinými programy** | Ano | Ano | Ano |
| **Převod kódu na UML diagram a zpět** | Ano | Ano | Ano |
| **Tvorba dokumentace** | Ano | Ano | Ne |
| **Modelování business procesů** | Ano | Ano | Ne |
| **Platformy** | Windows | Windows, MacOS, Linux | Windows, MacOS, Linux |
| **Možnost barevné úpravy modelu** | Ano | Ano | Ano |
| **Práce v týmu** | Ano | Ano | Ne |
| **Tvorba komunikačního diagramu** | Ano | Ano | Ne |
| **Tvorba sekvenčního diagramu** | Ano | Ano | Ano |

# 3 NÁSTROJE UMOŽŇUJÍCÍ TÝMOVOU SPOLUPRÁCI

V minulé kapitole byl brán zřetel na nástroje pro tvorbu UML diagramů. Další pozornost bude věnována nástrojům umožňující týmovou spolupráci. Jsou to většinou webové aplikace umožňující sestavování užšího okruhu lidí, resp. týmu, které si mezi sebou zadávají úkoly a dbají na jejich včasné dokončení. Jednou z nejznámějších aplikací toho typu je Jira.

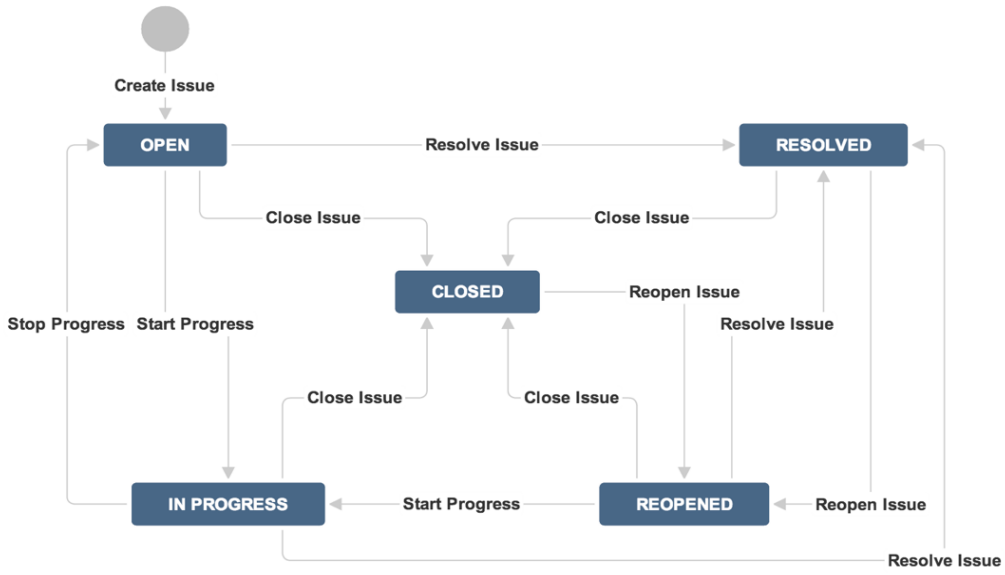
## 3.1 Jira

Jira je softwarový nástroj společnosti Atlassian. Projekt v tomto nástroji je definován jako kolekce úkolů, které odpovídají požadavkům dané organizace. Projektem pak může být například:

- vývoj softwaru,
- marketingová kampaň,
- helpdesk systém,
- systém správy požadavků,
- systém požadavků na vylepšení webových stránek.

Součástí projektu je logické seskupení úkolů v rámci daného projektu. Takové seskupení se nazývá komponenta. Každý projekt může sestávat z různých komponent (nebo z žádné), v závislosti na potřebách organizace. [17]

Dalším důležitým pojmem v tomto nástroji je workflow (pracovní postup). Workflow je soubor stavů a přechodů, kterým prochází úkol během jeho životního cyklu. Následující schéma (Obrázek 8) zobrazuje vestavěný pracovní postup v Jiře.



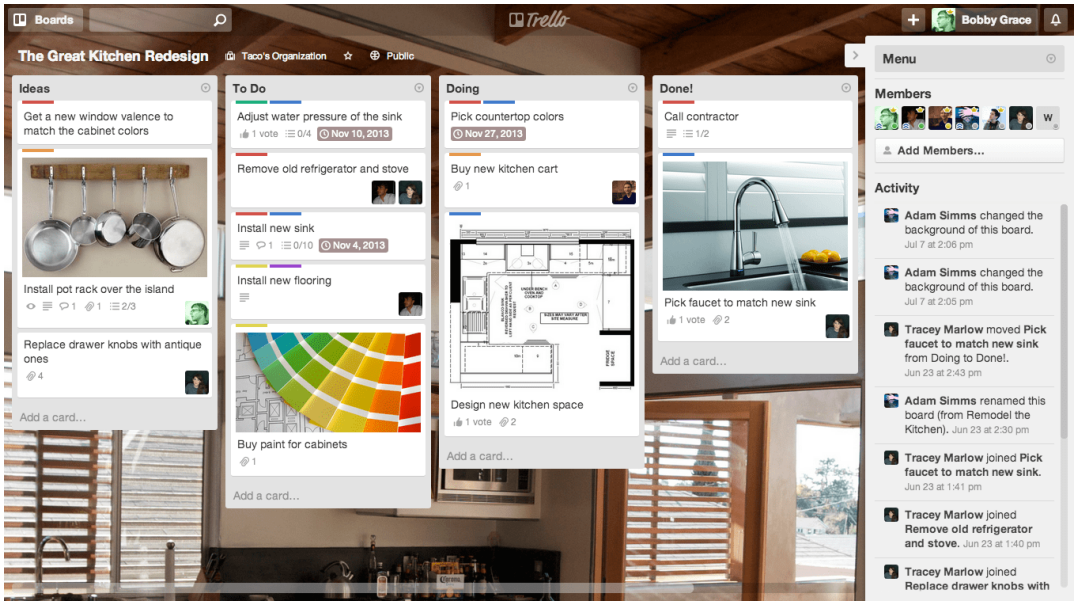

**Obrázek 8: Jira Workflow [18]**

Nedílnou součástí tohoto nástroje je pak tvorba reportů, které v rámci úkolů zobrazují statistiky konkrétních lidí, projektů, verzí a dalších důležitých aspektů. Tyto reporty jsou automaticky vytvářeny v případě, že má projekt povolené sledování v čase. Jira poskytuje celou řadu reportů. Zde bude zmíněno jen několik hlavních. [19]

- Average Age Report
  - Zobrazuje průměrnou dobu života nevyřešených úkolů.
- Created vs Resolved Issues Report
  - Zobrazuje počet vyřešených úkolů ku počtu nevyřešených za dané časové období.
- Recently Created Issues Report
  - Zobrazuje rychlost vytváření úkolů.
- Resolution Time Report
  - Zobrazuje průměrnou dobu potřebnou k vyřešení úkolu.
- Time Since Issues Report
  - Zobrazuje počet vytvořených úkolů k danému datu.

## 3.2 Trello

Další nástroj podporující týmovou spolupráci je Trello. Byl vytvořen v roce 2010 společností Fog Creek Software jako webová aplikace. V roce 2017 bylo Trello přesunuto pod firmu Atlassian, pod kterou je i Jira. Dnes již má přes 5 milionů uživatelů a mobilní aplikace na platformách iOS i Android. [20]




**Obrázek 9: Trello [21]**

Hlavním děním celého programu je nástěnka, což je v podstatě sada karet, která slouží ke komunikaci s týmem. Ke každé takové kartě lze přidávat komentář, nahrávat přílohy, upravovat termíny a mnoho dalšího. Kalendář v Trellu obsahující důležité termíny dokončení jednotlivých úkolů, lze snadno propojit přes formát iCal s libovolnou aplikací, která toto povoluje. Umožňuje to tak sledovat deadliny úkolů nejen přímo v prostředí Trello, ale i v soukromém diáři telefonu, tabletu apod. [21]

Trello se výborně hodí pro práci v agilním modelu Kanban, o kterém byla řeč v kapitole 1.2.2 Na obrázku 9 lze spatřit již zmíněnou nástěnku s kartami. Panel na pravé straně pak obsahuje seznam členů aktuálního týmu a list aktivit, které provedli ostatní členové za poslední dobu.

## 3.3 Monday

Posledním nástrojem, který bude zmíněn v této kapitole je platforma Monday. Je to vizuální nástroj pro správu projektů, který pomáhá přetvářet způsob spolupráce týmu. Jedná se o jednoduchý a intuitivní nástroj, který umožňuje lidem řídit práci, dodržovat lhůty a vytvářet kulturu transparentnosti. Snaží se eliminovat potřebu dlouhých e-mailových podprocesů a zbytečných schůzek. Veškerá komunikace a soubory jsou centralizovány na jednom místě, takže se nemůže stát, že by někomu unikla nějaká klíčová informace. [22]

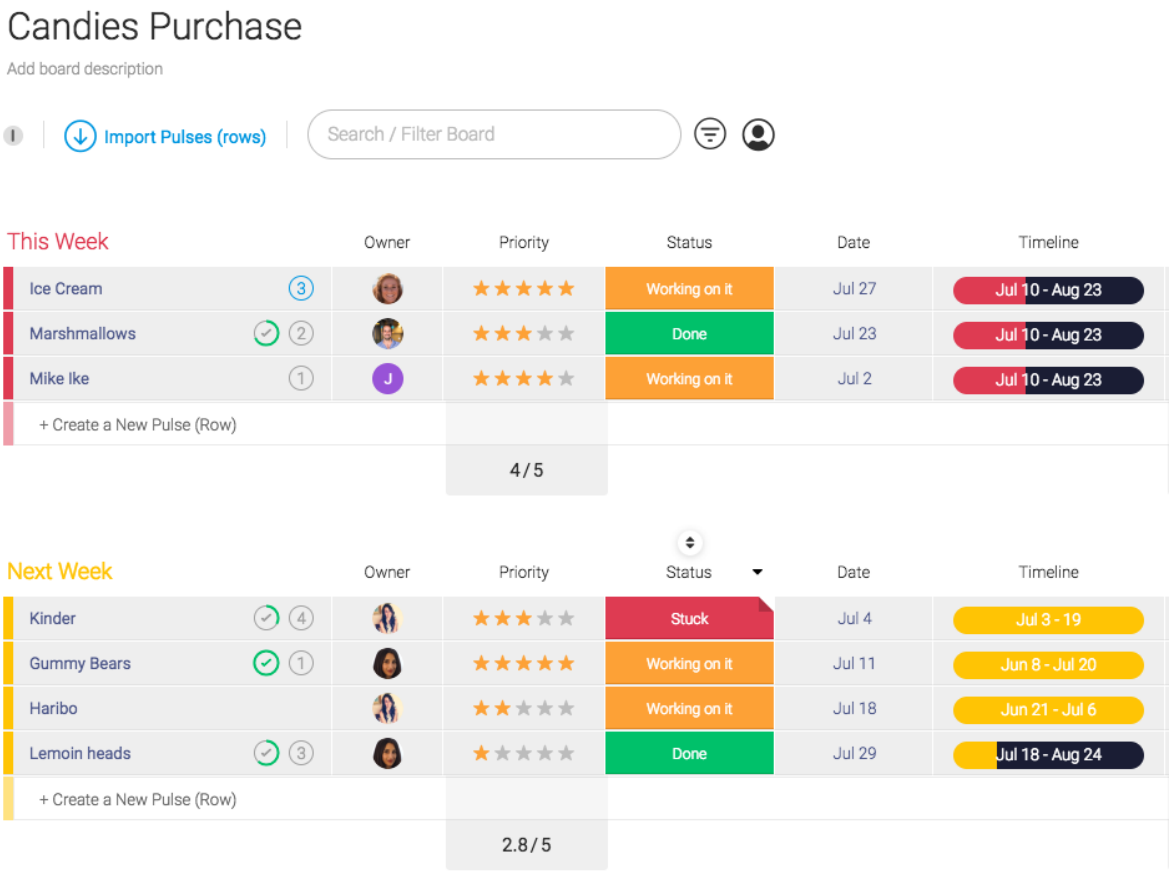
Po přihlášení do aplikace je prvním krokem nového uživatele vytvoření tzv. boardu. Board je místo, ve kterém se dá uspořádat vše potřebné – projekt, seznam úkolů apod. Boardy se dělí na:

- Main boards;
    - Jsou viditelné každému členovi vašeho týmu.
    - Vše vytvořené v této sekci bude pro ostatní uživatele přístupné a přehledné.
- Shareable boards;
    - Používají se při potřebě sdílet board i s jiným uživatelem mimo tým.
- Private boards;
    - Tyto boardy vidí jen uživatel, který je vytvořil. Umožňují ale i sdílení s jinými uživateli.

Dalším krokem je výběr šablony. Šablonu si může vytvořit uživatel sám, nebo si vybere jednu z předdefinovaných. Šablony jsou kategorizovány podle druhů použití na: Marketing, Content Production, Project Management, Sales & Customers, Freelancers, Design a doporučené.

Board je sestaven ze skupin. Skupina je na boardu barevně kódovaná část obsahující úkoly. Skupina může představovat časové období (týden, měsíc…), konkrétní krok projektu či cokoliv jiného. Do těchto skupin se pak již zadávají samotné úkoly. Úkol s sebou nese prioritu, status, časové období vykonávání, deadline vykonání, samozřejmě vykonavatele

úkolu a pomocné tagy. Může nést i jiné informace, ale to již záleží na zvolené šabloně. Následující obrázek (Obrázek 10) zobrazuje, jak může takový board vypadat. Nachází se zde dvě skupiny: This Week a Next Week, ve kterých jsou přidány vždy nějaké úkoly. [23], [24]




**Obrázek 10: Monday [23]**

Další předností tohoto nástroje je jeho obrovská integrita s jinými softwary. Dokonce dokáže spolupracovat i s jeho konkurentem Trello.

## 3.4 Srovnání týmových nástrojů

Jira, Trello, Monday – to jsou nástroje, které byly představeny v rámci podpory týmové spolupráce. Srovnání bude těžké, přeci jenom každý umí něco jiného. Nejvíce se asi odlišuje nástroj Jira, jelikož umožňuje správu úkolů a podporuje agilní metodiky vývoje softwaru. Zbylé programy jsou univerzální do různých odvětví týmového managementu. Následující tabulka (Tabulka 2) představuje porovnání základních požadavků na software podporující týmovou spolupráci.

Z tabulky vychází jako vítěz nástroj Jira. Dokáže totiž vše, co se dá od nástroje pro plánování softwarových projektů požadovat. Kdyby prioritou v hledání nebyl program na řízení softwarových projektů ale obecně týmový nástroj, tak by byl možným vítězem nástroj Monday.

**Tabulka 2: Srovnání týmových nástrojů**

| | Jira | Trello | Monday |
|---|---|---|---|
| **Zkušební verze** | Na cloudu 7 dní. Na serveru 30 dní. | Dostupná neomezeně, ale s horšími funkcionalitami. | Dostupná na 14 dní. |
| **Integrace s jinými programy** | Ano | Ano | Ano |
| **Sledování v čase** | Ano | Ne | Ano |
| **Tvorba reportů** | Ano | Ne | Ano |
| **Tvorba dokumentace** | Ano | Ano | Ne |
| **Správa problémů** | Ano | Ne | Ne |
| **Agilní metodiky** | Ano | Ne | Ne |
| **Mobilní aplikace** | Android, iOS | Android, iOS | Android, iOS |
| **API** | Ano | Ano | Ano |
| **Autentizace** | Email | Email, Google, SSO | Email |

# 4 POUŽITÉ TECHNOLOGIE

V předchozích kapitolách byly provedeny rešerše na nejpoužívanější nástroje z odvětví UML a týmové spolupráce. Aktuální kapitola se zaměří na technologie, které byly použity v rámci praktické části této diplomové práce.

Praktická část diplomové práce počítá s webovou aplikací běžící na systému Microsoft SQL Server. Samotná aplikace je pak napsaná za pomoci frameworku ASP.NET MVC, který využívá programovacího jazyka C#. Následující text popisující SQL Server vychází ze zdroje [25]. Podkapitola týkající se frameworku ASP.NET je inspirována zdrojem [26].

## 4.1 SQL Server

SQL Server je relační systém řízení báze dat (RDBMS) vyvinutý společností Microsoft. Je schopen spravovat osobní databáze o pár megabajtech až po multiserverové databázové systémy řídící terabajty informací.

Produkt SQL Server se skládá z několika různých komponent. Hlavní součástí je databázový engine, což je páteřní aplikační služba v balíčku SQL Serveru pro ukládání, zpracování a zabezpečení dat. Data se ukládají a zpracovávají v relačním formátu jako XML dokumenty, jako prostorové údaje a jako indexy sloupců. Hlavní úkoly databázového enginu jsou následující:

- zajištění spolehlivého ukládání dat;
- poskytování prostředků k rychlému načtení dat;
- poskytování konzistentního přístupu k datům;
- řízení přístupu k datům pomocí zabezpečení;
- prosazování pravidel integrity dat pro zajištění spolehlivosti a konzistentnosti;

### 4.1.1 Spolehlivé ukládání dat

Spolehlivé ukládání dat začíná již na hardwarové úrovni. Není to striktní zodpovědnost databázového enginu, ale je to nezbytná součást dobře vybudované databáze. Přestože se dá například na jednotku SATA umístit celá databáze SQL (nebo dokonce vypálit kopii na CD), je vhodné zachovat data na nějakém typu úložiště, které disponuje redundancí, jako tomu je u polí RAID nebo SAN. Tato pole mohou přežít selhání hardwaru na úrovni disku bez ztráty dat.

Pomocí zvoleného hardwaru pak databázový engine spravuje všechny datové struktury potřebné k zajištění spolehlivého ukládání dat. Řádky dat jsou uloženy na stránkách velikosti 8 kB. Osm takových stránek tvoří prostor a databázový engine sleduje, do jaké míry jsou přiděleny tabulky a indexy.

### 4.1.2 Rychlý přístup k datům

SQL Server poskytuje vytváření indexů umožňující rychlý přístup k datům.

Dalším způsobem, jak zajistit rychlý přístup k datům, je udržování často přístupných dat v paměti. Nadbytečná paměť pro instanci serveru se využívá jako datová cache paměť. Když jsou požadovány stránky z databáze, databázový engine zkontroluje server, zda jsou již požadované stránky v paměti cache. Pokud nejsou, načte je z disku a uloží do paměti cache. Pokud v této paměti není aktuálně místo, pak stránky, které se nejdelší dobu nevyužívají, jsou z této paměti odebrány. V případě, že tyto stránky obsahují nezapsané změny, jsou tyto před odebráním zapsány na disk.

### 4.1.3 Konzistentní přístup k datům

Rychlé získání údajů neznamená mnoho, pokud jsou informace, které obdržíme, nepřesné. SQL Server se řídí souborem pravidel, které zajišťují konzistenci dat.

Obecná myšlenka s konzistentním přístupem k datům je umožnit pouze jednomu uživateli změnit data a zabránit ostatním číst data z databáze, zatímco probíhají změny. Datové a transakční konzistence jsou udržovány v SQL Serveru pomocí uzamykání transakcí. Transakční konzistence má několik úrovní shody, z nichž každá poskytuje kompromis mezi přesností dat a souběžností.

### 4.1.4 Řízení přístupu

SQL Server řídí přístup prováděním zabezpečení na několika úrovních. Bezpečnost je vynucena na úrovni serveru, databáze, schématu a objektu. Přístup na úrovni serveru je vynucen buď pomocí uživatelského jména a hesla k serveru, nebo prostřednictvím integrovaného zabezpečení sítě, který používá k přihlášení identifikační přihlašovací údaje klienta.

### 4.1.5 Integrita dat

Některé databáze musí sloužit potřebám více než jedné aplikace. Podniková databáze, která obsahuje cenné informace, může mít deset různých oddělení, které chtějí získat přístup do částí databáze pro různé potřeby.

V tomto případě je nepraktické očekávat, že vývojáři každé aplikace budou agregovat na stejném souboru standardů pro zachování integrity dat. Například jedno oddělení by mohlo umožnit rozšíření telefonních čísel, zatímco jiné oddělení nemusí tuto funkci vůbec potřebovat. Jedno oddělení může považovat za důležité udržovat vztah mezi údaji zákazníka a údaji prodejce, zatímco jiný může mít zájem pouze o údaje zákazníků.

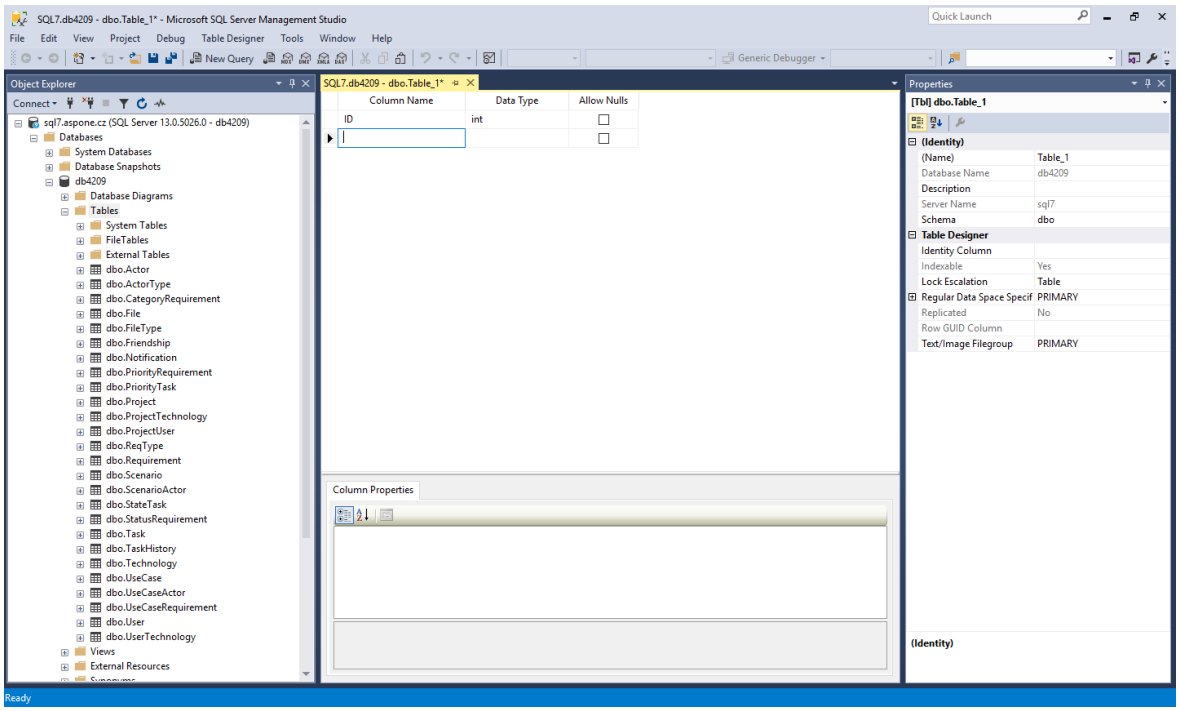
Nejlepším způsobem, jak zajistit, aby data zůstávala konzistentní a použitelná pro všechny, je prosadit soubor pravidel o celistvosti dat v samotné databázi. Toho lze dosáhnout omezením integrity dat a dalšími mechanismy integrity dat, jako jsou například triggery.

### 4.1.6 SQL Server Management Studio (SSMS)

SSMS je integrovaná aplikace, která poskytuje přístup k většině grafickým nástrojům, které lze použít k provádění administrativních a vývojových úloh na SQL Serveru. SSMS byl představen s SQL Serverem 2005 a nahradil tak programy: Enterprise Manager, Query Analyzer a Analysis Manager, které byly k dispozici v SQL Serveru 2000. Společnost

Microsoft konsolidovala všechny tyto nástroje do jednoho se zaměřením na poskytnutí nástroje, který vyhoví potřebám vývojářů i databázových správců. Tuto konsolidaci podpořili v SQL Serveru 2012 tím, že využívají prostředí Visual Studia v SSMS.

Následující obrázek (Obrázek 11) zobrazuje prostředí tohoto programu.




**Obrázek 11: SQL Server Management Studio**

Činnosti, které lze v prostředí SSMS provádět jsou například:

- správa několika serverů v praktickém rozhraní;
- správa databázových uživatelů a databázových rolí;
- vytváření, upravování a plánování automatizovaných úloh pomocí SQL Server Agenta;
- zálohování a obnovování databáze;
- vytváření nových databází;
- procházení obsahů tabulek;
- vytváření a spravování databázových objektů (jako jsou tabulky, indexy a procedury);
- generování DDL skriptů pro databáze a objekty databáze;
- vytváření, upravování, spouštění a ladění Transact-SQL (T-SQL) skriptů;
- definování, implementace a správa služeb SQL Serveru;

## 4.2 ASP.NET MVC

ASP.NET MVC je framework pro vytváření webových aplikací, který aplikuje základní architektonický vzor Model-View-Controller do frameworku ASP.NET.

Model-View-Controller (MVC) je důležitým architektonickým vzorem v oblasti počítačové vědy již mnoho let. Původně nazvaný Thing-Model-View-Editor v roce 1979, byl později zjednodušen na Model-View-Controller. Jedná se o výkonný a elegantní prostředek pro oddělení logiky přístupu k datům a logiky zobrazení. Velmi dobře se tak uplatňuje ve webových aplikacích. MVC odděluje uživatelské rozhraní (UI) aplikace do tří hlavních aspektů:

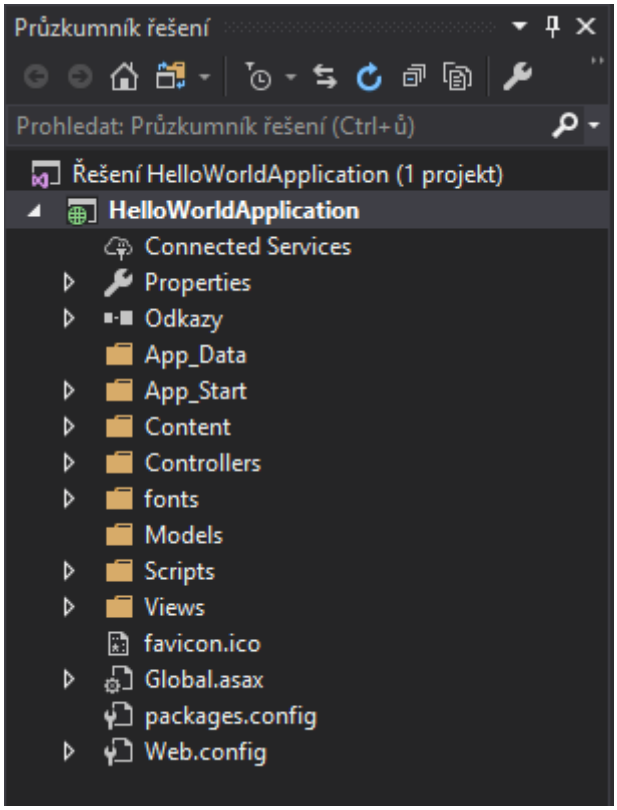
- **Model:** Sada tříd popisující data, s nimiž se pracuje. Popisuje též business pravidla, jak budou data měněna a jak s nimi bude manipulováno.
- **View:** Definuje, jak se bude zobrazovat uživatelské rozhraní aplikace.
- **Controller:** Sada tříd, která se stará o komunikaci od uživatele, celkový chod aplikace a aplikační logiku.

Vzor MVC se často používá ve webovém programování. V kombinaci s ASP.NET je přeložen zhruba takto:

- **Model:** Jedná se o třídy, které reprezentují oblast toho, co nás zajímá. Objekty těchto tříd často zapouzdřují data uložená v databázi, stejně tak kód, který manipuluje s daty a vynucuje business logiku specifickou pro danou oblast. V ASP.NET MVC to nejčastěji slouží jako datová přístupová vrstva využívající nějakého nástroje jako Entity Framework nebo NHibernate v kombinaci s vlastním kódem obsahujícím specifickou logiku pro danou oblast.
- **View:** Šablona pro dynamické generování HTML kódu.
- **Controller:** Speciální třída, která řídí vztah mezi View a Modelem. Reaguje na vstup uživatele, promluví si s Modelem a rozhodne, která View má být vykreslena (pokud existuje). V ASP.NET MVC je tato třída běžně označována příponou Controller.

### 4.2.1 Struktura MVC aplikace

Při vytváření nové ASP.NET MVC aplikace Visual Studio automaticky přidá několik souborů a adresářů do projektu, jak je znázorněno na obrázku níže (Obrázek 12).




**Obrázek 12: Struktura ASP.NET MVC aplikace**

Projekty ASP.NET MVC vytvořené pomocí šablony webových aplikací mají osm adresářů nejvyšší úrovně, které jsou uvedeny v následující tabulce (Tabulka 3).

**Tabulka 3: Adresáře v projektu ASP.NET MVC**

| Adresář | Účel |
|---|---|
| **/Controllers** | Obsahuje třídy typu Controller, které zpracovávají URL požadavky. |
| **/Models** | Obsahuje třídy typu Model, které reprezentují a manipulují s daty. |
| **/Views** | Obsahuje šablony UI, které jsou odpovědné za vykreslení HTML výstupu. |
| **/Scripts** | Obsahuje JavaScript soubory a skripty (.js). |
| **/fonts** | Obsahuje pár webových fontů od šablony Bootstrap. |
| **/Content** | Obsahuje CSS, obrázky a další potřebné soubory, kromě skriptů. |
| **/App_Data** | Obsahuje datové soubory ke čtení či zapsání. |
| **/App_Start** | Obsahuje kód se základní konfigurací aplikace. |

### 4.2.2 NuGet

Microsoft nemůže poskytnout každý kousek kódu, který by vývojář mohl potřebovat. Miliony vývojářů pracují na platformě .NET, z nichž každý má svůj malý problém, který

potřebují vyřešit. Čekání na řešení problémů od společnosti Microsoft není účinné a ani nemá smysl.

Dobrou zprávou je, že existují vývojáři, kteří nečekají na to, až budou mít podobný problém. Naopak, vyřeší své vlastní problémy (i problémy ostatních) pomocí užitečných knihoven, které si sami napíší a poté je distribuují ostatním vývojářům přes internet.

S takovou knihovnou se pak dají dělat tři velké věci. Nalézt ji, nainstalovat ji a udržovat ji. Jak si ale vývojáři najdou knihovnu? Když ji najdou, jak ji využijí ve svých projektech? Jakmile ji nainstalují, jak budou sledovat její aktualizace?

Tato část prochází rychlým příkladem kroků nezbytných k instalaci knihovny ELMAH bez pomoci NuGet. ELMAH (Error Logging Module And Handler), který je označen jako modul odchytávání chyb, se používá k zobrazení informací o neošetřených výjimkách v rámci webové aplikace. K využití této knihovny jsou potřebné následující kroky:

1. **Najít ELMAH.** Díky svému jedinečnému jménu to je snadné s jakýmkoliv webovým vyhledávačem.
2. **Stáhnout si správný balíček zip.** Zobrazí se několik souborů zip, a tak není vždy triviální vybrat ten správný.
3. **„Odblokovat“ balíček.** Soubory stažené z internetu jsou označeny metadaty, které specifikují, že pocházejí z „webové zóny“ a jsou potenciálně nebezpečné. Tato značka je někdy označována jako MOTW (Mark of the Web – značka webu). Odblokování souboru zip před rozšířením je důležité; v opačném případě je možné, že kód nebude fungovat správně, protože každý soubor v balíčku má přidanou bitovou sadu.
4. **Ověřit hash balíčku s tím, který poskytuje hostitelské prostředí.** Pro bezpečnost a ověření, že byl stažen správný soubor je nutné ověřit hash souboru s tím, který je uveden na stránce pro stahování.
5. **Rozbalit obsah balíčku do známého umístění, například do složky lib, aby se dalo odkazovat na sestavu.** Vývojáři obvykle nechtějí přidávat sestavy přímo do adresáře bin, protože nechtějí přidat adresář bin k řízení zdroje.
6. **Přidat odkaz na knihovnu.** Přidat odkaz na knihovnu v projektu Visual Studia.
7. **Aktualizovat web.config.** ELMAH vyžaduje trochu konfigurace. Obvykle se hledá dokumentace, aby se zjistilo správné nastavení.

Všechny tyto kroky jsou potřebné pro knihovnu ELMAH, která nemá žádné jiné závislé knihovny. Pokud má knihovna závislosti, pak při každé aktualizaci knihovny se musí najít správná verze každé závislosti a pro každou z nich provést předchozí kroky. Jedná se o

bolestný soubor úkolů, které se musí provést pokaždé, když budete chtít nasadit novou verzi vaší aplikace. Mnoho týmů se tak raději drží starých verzí svých závislostí.

Tyhle starosti řeší NuGet. NuGet automatizuje všechny tyto běžné a nudné úkoly pro balíček i jeho závislosti. Odstraňuje většinu problémů se začleněním otevřené knihovny třetí strany do zdrojového stromu projektu.

NuGet je součástí programu Visual Studio od verze 2012 a 2013; dříve vyžadoval samostatnou instalaci s aplikací Visual Studio 2010. V aplikaci Visual Studio jsou na výběr dva způsoby interakce: dialogové okno Spravovat balíčky NuGet a konzola Správce balíčků.

K instalaci balíčku ELMAH pomocí NuGet jsou zapotřebí tyto kroky:

1. Zadat jméno ELMAH do vyhledávacího pole. Pokud se nabídne několik výsledků, vybírá se podle popisu, autora či počtu stažení.
2. Po nalezení balíčku kliknout na tlačítko Instalovat. Toto stáhne balíček a všechny jeho závislosti a zasadí jej do našeho projektu.

Po instalaci balíčku se v projektu provede několik změn. Po první instalaci balíčku do projektu se vytvoří nový soubor s názvem packages.config, který pak obsahuje seznam instalovaných balíčků v projektu.

## 4.3 Entity Framework

Nový ASP.NET MVC 5 projekt automaticky obsahuje referenci na Entity Framework (EF). EF je objektově-relační mapovací (ORM) framework, který ví, jak ukládat .NET objekty v relační databázi, a naopak jak je z databáze získávat pomocí dotazů LINQ.

EF podporuje vývojové modely: database-first, model-first a code-first. Code-first znamená, že můžeme začít ukládat a načítat informace z SQL serveru bez vytvoření databázového schéma. Píší se pouze obyčejné C# třídy a EF určuje, jak a kde ukládat instance těchto tříd.

EF, stejně jako ASP.NET MVC, následuje řadu konvencí, které vývojáři usnadňují život. Pokud by vývojář například chtěl v databázi uložit objekt typu Album, EF předpokládá, že chce vývojář uložit data do tabulky s názvem Album. Pokud má objekt vlastnost s názvem ID, EF si to přeloží tak, že tato vlastnost drží hodnotu primárního klíče a nastaví v databázi tomuto sloupci automatickou inkrementaci.

Stejně tak má EF konvence i pro cizí klíče, názvy databází a další. Tyto konvence nahrazují veškeré mapování a konfiguraci, které vývojáři historicky poskytli objektově-relačnímu frameworku.

Rozdíl mezi database-first, model-first a code-first bude vysvětlen ze zdroje [27].

### 4.3.1 Model-First

Pro uživatele, kteří nejsou obeznámeni s většinou návrhových nástrojů IDE, jako jsou XML Schema DataSet (XSD) a vizuální rozhraní Entity Designer Model XML (EDMX), pak přístup Model-First může být spíše matoucí. Klíčem k pochopení je uznat skutečnost, že slovo „model“ zde má za cíl definovat vizuální diagram vytvořený pomocí nástrojů návrhu. Tento diagram bude potom využíván frameworkem pro automatické generování SQL skriptu zdrojových kódů datového modelu.

Model-First v podstatě znamená pracovat s vizuálním diagramem a nechat Entity Framework vytvářet / aktualizovat zbytek odpovídajícím způsobem.

Výhody Model-First:

- Schopnost vytvářet schéma databáze a diagram tříd jako celek pomocí vizuálního návrhového nástroje, který může být skvělý, když je datová struktura velká.
- Kdykoli se databáze změní, model lze odpovídajícím způsobem aktualizovat bez ztráty dat.

Nevýhody Model-First:

- Automatické SQL skripty generované diagramy mohou v případě aktualizací vést ke ztrátě dat. Jednoduchým řešením je vytváření skriptů na disku a jejich ruční úprava, která ale vyžaduje slušné SQL znalosti.
- Řešení schématem může být obtížné, zvláště pokud se vyžaduje přesná kontrola nad modelovými třídami; vývojář nebude mít vždy možnost získat to, co chce, protože skutečný zdrojový kód bude automaticky generován nástrojem.

### 4.3.2 Database-First

Vzhledem k nevýhodám Model-First může vyplývat, že Database-First je ta pravá cesta. To může být pravda, pokud vývojář již bude mít k dispozici existující databázi. V takovém případě je přístup Database-First podobný Model-First, s výjimkou toho, že jde opačným směrem; namísto ručního návrhu EDMX a generování skriptu SQL k vytvoření databáze, je nejprve nutné vytvořit databázi a z ní vygenerovat EDMX návrh.

Database-First se dá shrnout tak, že se nejprve vybuduje databáze, z které pak Entity Framework vytvoří zbytek odpovídajícím způsobem.

Výhody Database-First:

- Pro projekty s již existující databází je toto ta práva cesta, protože ušetří potřebu ji znovu vytvářet.

- Riziko ztráty dat bude omezeno na minimum, protože v databázi budou vždy provedeny změny či aktualizace.

Nevýhody Database-First:

- Ještě menší kontrola nad generovanými třídami modelu (a jejich zdrojovým kódem) než pomocí Model-First; vyžadování rozsáhlých znalostí o konvencích a normách EF.

### 4.3.3 Code-First

Code-First je vlajkovou lodí Entity Frameworku od verze 4, která umožňuje elegantní a vysoce efektivní vývojový pracovní proces datového modelu. Přístup Code-First umožňuje vývojáři definovat objekty používající pouze standardní třídy, aniž by byl potřeba jakýkoliv návrhový nástroj, soubory mapování XML nebo těžkopádné hromady generovaného kódu.

Code-First znamená napsaní tříd datových modelů, které se budou používat v rámci nového projektu a ponechání Entity Frameworku vygenerovat odpovídající databázi.

Výhody Code-First:

- Není třeba diagramů a vizuálních nástrojů, které mohou být skvělé pro malé až střední projekty, protože dokáží ušetří spoustu času.
- Umožnění vývojářům dodržovat konvenční přístup ke konfiguraci a současně přecházet na vlastní implementaci při potřebě přizpůsobit mapování databáze.

Nevýhody Code-First:

- Dobrá znalost programovacího jazyka a ORM konvencí je vyžadována.
- Údržba databáze může být někdy obtížná, stejně jako zpracování aktualizací bez ztráty dat.

# 5 ANALÝZA

Tato diplomová práce předpokládá funkční webovou aplikaci, jejichž účelem bude správa softwarových projektů s možností týmové spolupráce.

Zadání této práce by znělo asi takto: Představme si projekt, na který jsou navázány požadavky a aktéři. Z těchto požadavků a aktérů jsou uskupeny případy užití, ke kterým jsou vytvořeny scénáře, včetně jejich alternativ. Kromě těchto položek jsou dále na projekt navázány i soubory, které mají nějaký vztah s projektem. Nedílnou součástí toho všeho je pak uživatel, který projekt vytvořil a má možnost k němu přizvat jiné uživatele, aby se na něm taktéž podíleli. Takto vytvořený tým lidí si pak mezi sebou zadává úkoly podle metodiky Kanban.

Jak bylo zmíněno v předchozích kapitolách – na začátku každého softwarového projektu je jeho analýza, a právě tomu se bude věnovat tato kapitola. Nejprve budou představeny požadavky na systém, dále očekávaní aktéři a konečně případy užití, ke kterým budou vyhotoveny scénáře. Vybrané scénáře pak budou ověřeny pomocí sekvenčních diagramů. K tomu ale, aby se tyto scénáře dali ověřit, bude nejprve představen analytický model tříd.

## 5.1 Požadavky

Požadavky na systém jsou zobrazeny v tabulce 4, kde jsou i kategorizovány a rozděleny dle typu na funkční a nefunkční. Kategorie byly určeny dle použití v systému. Konečné kategorie požadavků tedy jsou: Projekt, Uživatel, Úkoly, Požadavky, Aktéři, Scénáře, Soubory, Tým a Systém.

**Tabulka 4: Seznam požadavků na systém**

| ID | Požadavek | Typ | Kategorie |
|---|---|---|---|
| **R1** | Systém bude umožňovat vytvoření nového projektu | Funkční | Projekt |
| **R2** | Systém bude umožňovat editaci projektu | Funkční | Projekt |
| **R3** | Systém bude umožňovat sdílení projektu s vybranými uživateli | Funkční | Projekt |
| **R4** | Systém bude umožňovat vytváření dokumentace o projektu ve formátu PDF | Funkční | Projekt |
| **R5** | Systém bude umožňovat odstranění projektu | Funkční | Projekt |
| **R6** | Systém bude umožňovat registraci nového uživatele | Funkční | Uživatel |
| **R7** | Systém bude umožňovat přihlášení uživatele do systému | Funkční | Uživatel |
| **R8** | Systém bude umožňovat úpravu osobních údajů | Funkční | Uživatel |

| | | | |
|---|---|---|---|
| **R9** | Systém bude umožňovat vyhledání uživatele podle jména | Funkční | Uživatel |
| **R10** | Systém bude umožňovat přidávání jiných uživatelů do kontaktů | Funkční | Uživatel |
| **R11** | Systém bude umožňovat zrušení účtu | Funkční | Uživatel |
| **R12** | Systém bude umožňovat zobrazení profilu uživatele | Funkční | Uživatel |
| **R13** | Systém bude umožňovat vytvoření nového úkolu | Funkční | Úkoly |
| **R14** | Systém bude umožňovat přiřazení řešitele k úkolu | Funkční | Úkoly |
| **R15** | Systém bude umožňovat změnu stavu úkolu | Funkční | Úkoly |
| **R16** | Systém bude umožňovat vypsání seznamu úkolů | Funkční | Úkoly |
| **R17** | Systém bude umožňovat zobrazení kumulativního grafu úkolů | Funkční | Úkoly |
| **R18** | Systém bude umožňovat editaci úkolu | Funkční | Úkoly |
| **R19** | Systém bude umožňovat odstranění úkolu | Funkční | Úkoly |
| **R20** | Systém bude umožňovat změnu hodnoty WIP | Funkční | Úkoly |
| **R21** | Systém bude umožňovat vytvoření nového požadavku | Funkční | Požadavky |
| **R22** | Systém bude umožňovat vypsání seznamu požadavků | Funkční | Požadavky |
| **R23** | Systém bude umožňovat editaci požadavku | Funkční | Požadavky |
| **R24** | Systém bude umožňovat vypsání seznamu kategorií požadavků | Funkční | Požadavky |
| **R25** | Systém bude umožňovat vytvoření nové kategorie požadavků | Funkční | Požadavky |
| **R26** | Systém bude umožňovat editaci kategorie požadavků | Funkční | Požadavky |
| **R27** | Systém bude umožňovat odstranění požadavku | Funkční | Požadavky |
| **R28** | Systém bude umožňovat odstranění kategorie požadavků | Funkční | Požadavky |
| **R29** | Systém bude umožňovat vytvoření nového aktéra | Funkční | Aktéři |
| **R30** | Systém bude umožňovat editaci aktéra | Funkční | Aktéři |
| **R31** | Systém bude umožňovat vypsání seznamu aktérů a jejich uplatnění v případech užití a scénářích | Funkční | Aktéři |
| **R32** | Systém bude umožňovat import aktérů z CSV souboru | Funkční | Aktéři |
| **R33** | Systém bude umožňovat odstranění aktéra | Funkční | Aktéři |

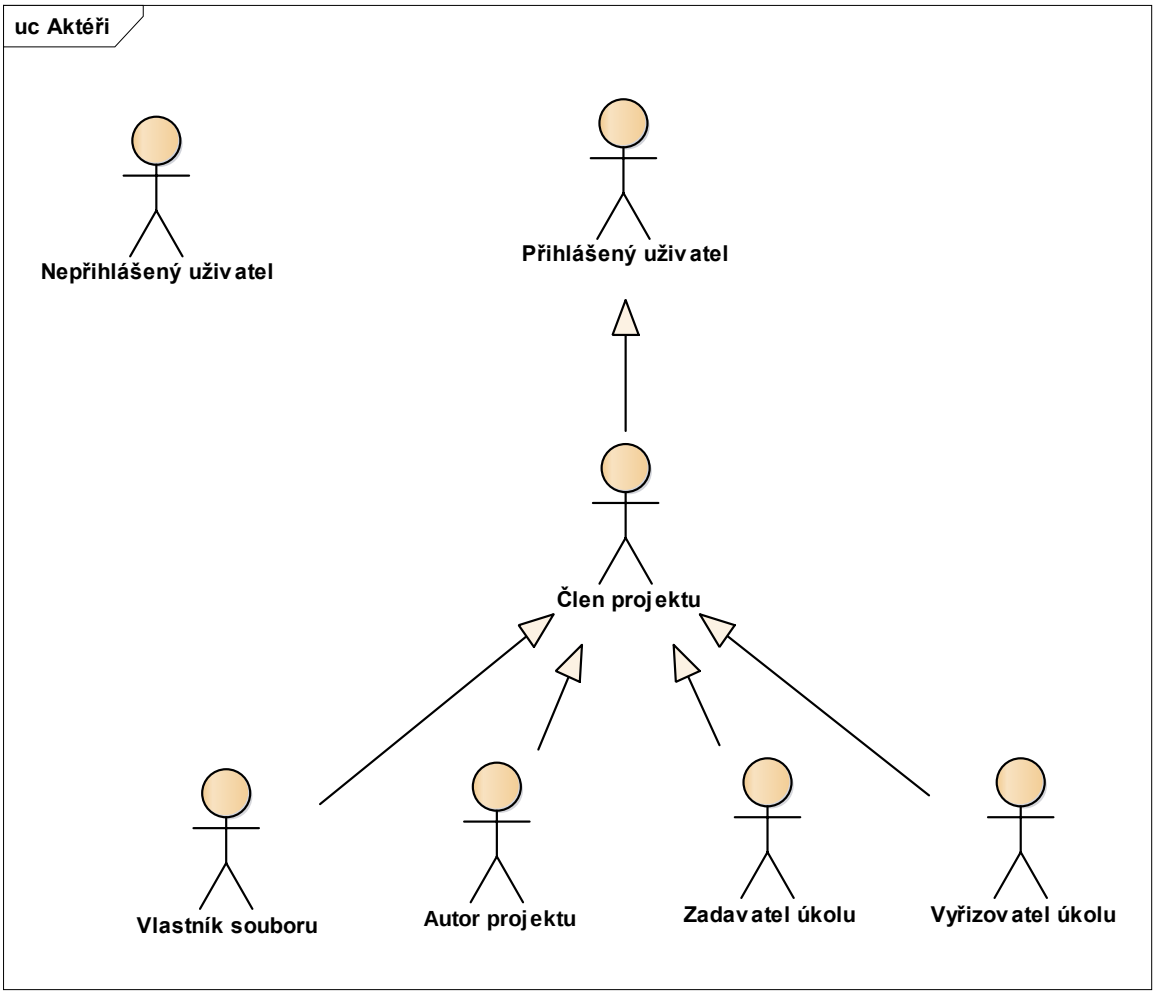
| | | | |
|---|---|---|---|
| **R34** | Systém bude umožňovat vytvoření nového případu užití | Funkční | Aktéři |
| **R35** | Systém bude umožňovat editaci případu užití | Funkční | Aktéři |
| **R36** | Systém bude umožňovat vypsání případů užití | Funkční | Aktéři |
| **R37** | Systém bude umožňovat přidání scénáře k případu užití | Funkční | Aktéři |
| **R38** | Systém bude umožňovat odstranění případu užití | Funkční | Aktéři |
| **R39** | Systém bude umožňovat zobrazení matice sledovatelnosti požadavků | Funkční | Aktéři |
| **R40** | Systém bude umožňovat vytvoření scénáře k případu užití | Funkční | Scénáře |
| **R41** | Systém bude umožňovat editaci scénáře | Funkční | Scénáře |
| **R42** | Systém bude umožňovat vytvoření alternativního scénáře k hlavnímu | Funkční | Scénáře |
| **R43** | Systém bude umožňovat zobrazení scénářů ve stromové struktuře | Funkční | Scénáře |
| **R44** | Systém bude umožňovat odebrání scénáře | Funkční | Scénáře |
| **R45** | Systém bude umožňovat nahrávání souborů k projektu | Funkční | Soubory |
| **R46** | Systém bude umožňovat stažení nahraného souboru | Funkční | Soubory |
| **R47** | Systém bude umožňovat odebrání souboru | Funkční | Soubory |
| **R48** | Systém bude umožňovat vypsání seznamu souborů | Funkční | Soubory |
| **R49** | Systém bude umožňovat přidávat nové členy do projektu | Funkční | Tým |
| **R50** | Systém bude umožňovat odebírat členy z projektu | Funkční | Tým |
| **R51** | Systém bude umožňovat vypsání seznamu členů týmu | Funkční | Tým |
| **R52** | Systém bude rozesílat členům týmu upozornění o změnách v projektu | Funkční | Systém |
| **R53** | Systém bude chránit uživatelská data | Nefunkční | Systém |
| **R54** | Systém bude bezchybně hlídat vstupy všech formulářů | Nefunkční | Systém |
| **R55** | Systém bude okamžitě reagovat na akce uživatele | Nefunkční | Systém |

## 5.2 Aktéři

Po sběru požadavků následuje hledání aktérů. Aktér představuje uživatelskou roli bezprostředně využívající systém. V tomto případě byli nalezeni následující aktéři:

- Nepřihlášený uživatel;
    - Uživatel, který není přihlášen v systému. Má právo na registraci, či přihlášení (pokud je již v systému registrován).
- Přihlášený uživatel;
    - Uživatel, který je úspěšně přihlášen v systému. Má právo na běžné uživatelské úkony týkající se správy jeho účtu, vytvoření si vlastního projektu, či pracování na cizím projektu.
- Autor projektu;
    - Přihlášený uživatel, který vlastní určitý projekt. Má právo na úpravu či smazání svého projektu, tvorbu dokumentace a úpravu hodnoty WIP.
- Člen projektu;
    - Člen projektu je buď sám autor projektu, nebo někdo, komu dal autor přístup do svého projektu.
- Vlastník souboru;
    - Vlastník souboru je někdo z členů projektu, který v rámci tohoto nahrál nějaký soubor. Pouze vlastník má právo na odstranění tohoto souboru.
- Zadavatel úkolu;
    - Zadavatel úkolu je člen projektu, který nějakému svému kolegovi (nebo sobě samému) zadal úkol. Pouze on má právo tento úkol upravit či odstranit. Může též měnit stav tohoto úkolu, i když to je hlavně práce vyřizovatele.
- Vyřizovatel úkolu;
    - Vyřizovatel úkolu je člen projektu, který dostal od zadavatele nějakou povinnost. On a zadavatel mají právo na změnu stavu tohoto úkolu.

Jak zobrazuje následující obrázek (Obrázek 13), mezi aktéry fungují určité dědičnosti. Vlastník souboru, Autor projektu, Zadavatel a Vyřizovatel úkolu dědí od Člena projektu. Sice má každý určitá svá privilegia, ale všichni mají stejná práva člena projektu. Člen projektu zase dědí od Přihlášeného uživatele. Jediným aktérem, který je lehce „odstrčený“ je nepřihlášený uživatel.

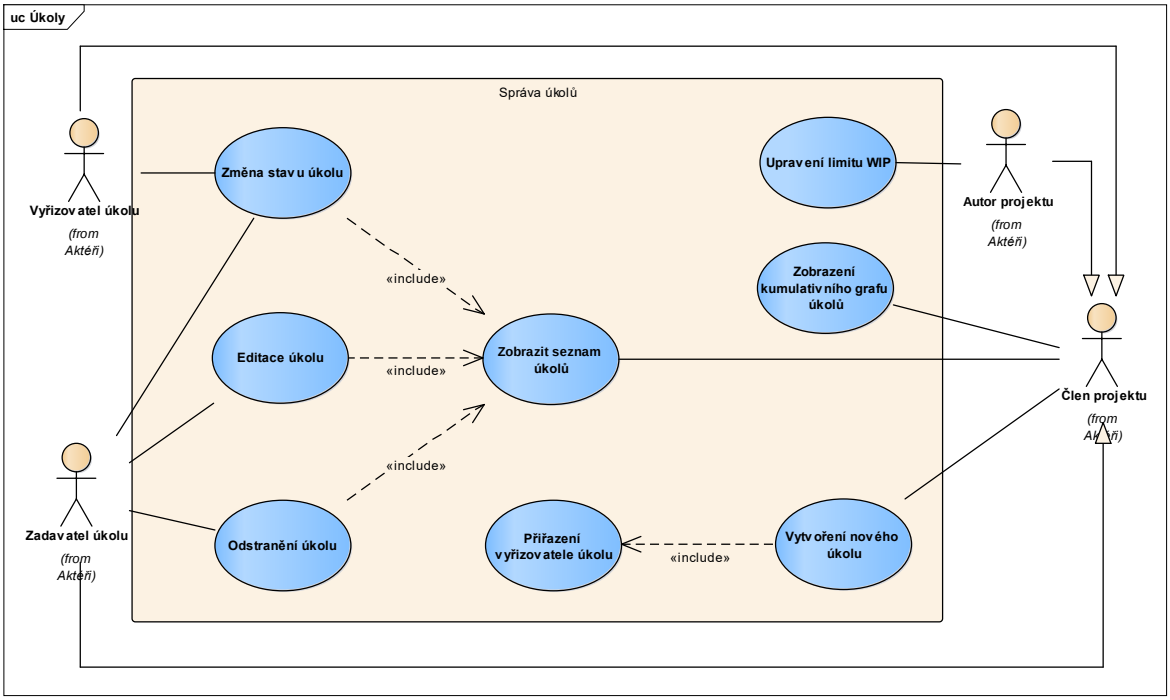



**Obrázek 13: Diagram aktérů**

## 5.3 Případy užití

Případ užití představuje službu, kterou nabízí systém aktérovi. Diagram případů užití zobrazuje, jak a kdo tyto služby využívá.

Případy užití se odvíjejí od výše uvedených požadavků. Pro přehlednost zde budou představeny pouze případy užití týkající se správy úkolů (Obrázek 14). Zbylé diagramy se nacházejí v příloze (Příloha A).




**Obrázek 14: Diagram případů užití**

Tento diagram zobrazuje případy užití týkající se správy úkolů. V diagramu se nachází čtyři aktéři, kteří využívají systém odpovídajícím způsobem:

- Autor projektu má právo na úpravu hodnoty limity WIP, která uvádí, kolik úkolů může být v jeden čas v rozpracovaném stavu.
- Vyřizovatel úkolu má právo, nebo spíše povinnost, měnit stav úkolu podle jeho aktuálního plnění. K tomu, aby mohl najít úkol, kterému chce změnit stav, si musí zobrazit seznam všech úkolů.
- Zadavatel úkolu má právo na editaci a odstranění úkolu, který zadal. Stejně tak má právo na změnu stavu tohoto úkolu. K dosažení všech těchto akcí musí nejprve daný úkol najít díky zobrazení všech úkolů.
- Člen projektu je univerzální uživatel, který má právo zobrazit si kumulativní graf úkolů, vytvořit nový úkol, a zároveň přiřadit vyřizovatele k tomuto úkolu. Vyřizovatelem tohoto úkolu může být i on sám.

Po vytvoření diagramu případů užití se tyto porovnávají s požadavky a hledá se, zda každý požadavek má k sobě adekvátní případ užití, který ho realizuje. Matici sledování těchto realizací zobrazuje Obrázek 15. Ostatní matice pro další kategorie požadavků se nacházejí v příloze (Příloha B).

| Source \ Target | Úkoly::Editace úkolu | Úkoly::Odstranění úkolu | Úkoly::Přiřazení vyřizovatele úkolu | Úkoly::Upravení limitu WIP | Úkoly::Vytvoření nového úkolu | Úkoly::Změna stavu úkolu | Úkoly::Zobrazení kumulativního grafu úkolů | Úkoly::Zobrazit seznam úkolů |
|---|---|---|---|---|---|---|---|---|
| Úkoly::S 13 Systém bude umožňovat vytvoření nového úkolu | | | | | ← | | | |
| Úkoly::S 14 Systém bude umožňovat přiřazení řešitele k úkolu | | | ← | | | | | |
| Úkoly::S 15 Systém bude umožňovat změnit stav úkolu | | | | | | ← | | |
| Úkoly::S 16 Systém bude umožňovat zobrazení seznamu úkolů a informací o nich | | | | | | | | ← |
| Úkoly::S 17 Systém bude umožňovat zobrazení kumulativního grafu úkolů | | | | | | | ← | |
| Úkoly::S 18 Systém bude umožňovat editaci úkolu | ← | | | | | | | |
| Úkoly::S 19 Systém bude umožňovat odstranění úkolu | | ← | | | | | | |
| Úkoly::S 20 Systém bude umožňovat úpravu konstanty WIP | | | | ← | | | | |

**Obrázek 15: Matice sledovatelnosti požadavků**

## 5.4 Scénáře

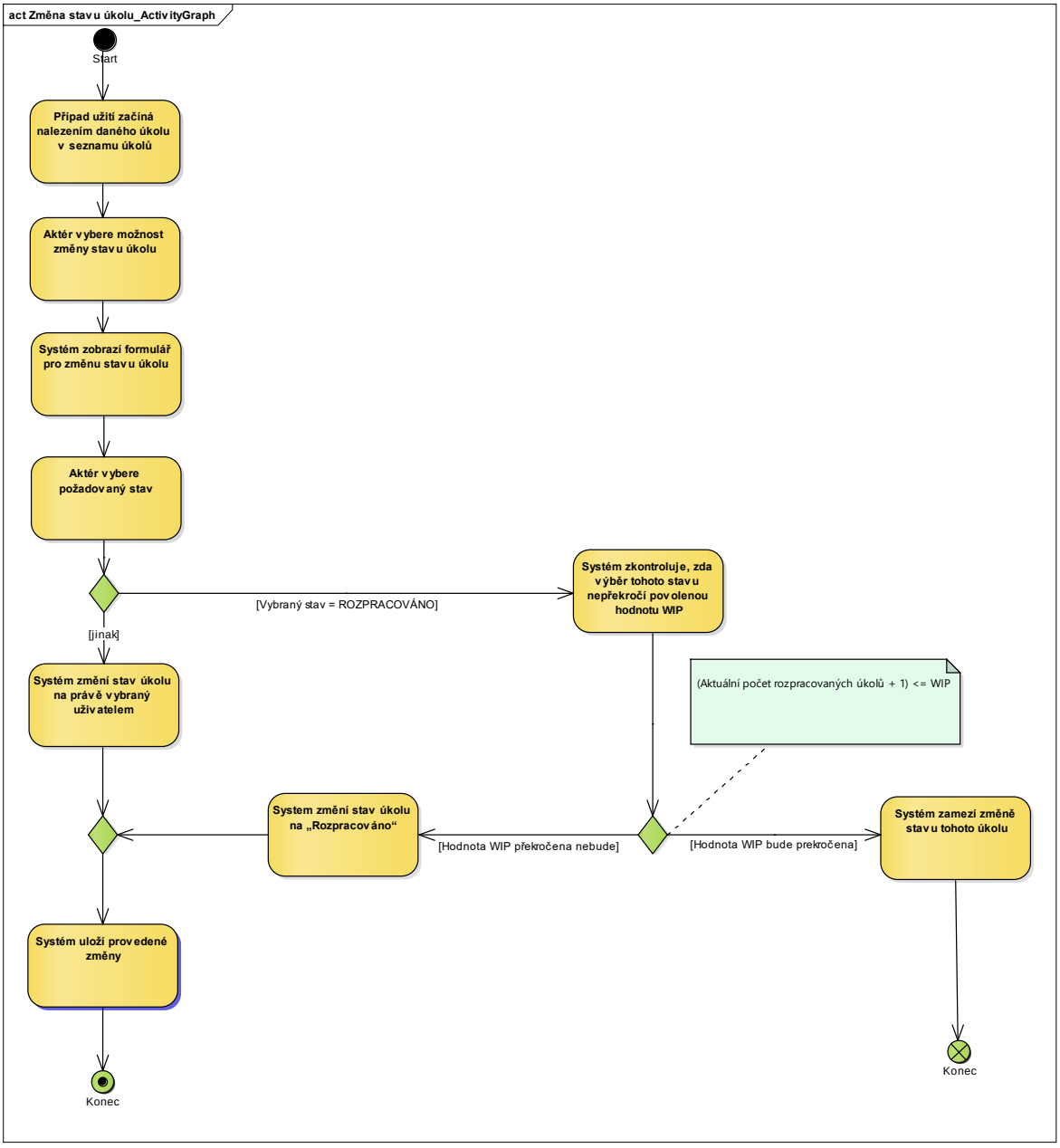
Po definování všech případů užití následuje podrobný popis, jakým způsobem každý případ užití probíhá. K takovému popisu slouží scénáře. Každý případ užití by měl mít jeden hlavní scénář, který popisuje bezproblémový průběh případu. Jelikož nikdo není dokonalý a chybička se může vloudit, definují se k hlavnímu scénáři ještě alternativní, které definují speciální průběh událostí v rámci možné situace. Nedoporučuje se k alternativnímu scénáři vytvářet další alternativní scénář, protože by byl systém příliš složitý. Všechny alternativní scénáře by měly být navázány pouze na jeden hlavní scénář.

V rámci kategorie Úkoly byl vybrán případ užití Změna stavu úkolu, ke kterému náleží následující scénář (Tabulka 5) a diagram aktivit (Obrázek 16). Další scénáře z této kategorie se nachází v Příloze C.

**Tabulka 5: Hlavní scénář k případu užití Změna stavu úkolu**

| Případ užití: Změna stavu úkolu |
|---|
| ID: 1 |
| Stručný popis:<br>Vyřizovatel nebo zadavatel úkolu změní jeho stav. |
| Hlavní aktéři:<br>Vyřizovatel úkolu nebo Zadavatel úkolu (Aktér) |
| Vedlejší aktéři:<br>Žádní. |

| |
|---|
| Vstupní podmínky:<br>1. Aktér je přihlášen v systému a má otevřený projekt. |
| Hlavní scénář:<br>1. Případ užití začíná nalezením daného úkolu v seznamu úkolů.<br>2. Aktér vybere možnost změny stavu úkolu.<br>3. Systém zobrazí formulář pro změnu stavu úkolu.<br>4. Aktér vybere požadovaný stav.<br>5. Pokud se vybraný stav rovná stavu „Rozpracováno“, pak:<br>5.1. Systém zkontroluje, zda výběr tohoto stavu nepřekročí povolenou hodnotu WIP.<br>5.2. Pokud hodnota překročena nebude, pak:<br>5.2.1. Systém změní stav úkolu na „Rozpracováno“.<br>5.3. Jinak:<br>5.3.1. Systém zamezí změnit stav tohoto úkolu.<br>6. Nebo:<br>6.1. Systém změní stav úkolu na právě vybraný uživatelem.<br>6.2. Systém uloží provedené změny. |
| Výstupní podmínky:<br>Stav úkolu byl úspěšně změněn. |
| Alternativní scénáře:<br>Žádné. |




**Obrázek 16: Diagram aktivit**

## 5.5 Analytický model tříd

Dalším stavebním kamenem při tvorbě analýzy je analytický model tříd. K jeho vytvoření se nejčastěji používá metoda podstatných jmen a sloves, která vychází z původního zadání projektu, požadavků, případů užití či ze slovníčku pojmů. Tato metoda slouží k mapování názvů tříd, atributů a metod. Všechna klíčová slova, která se nachází v projektu, budou rozdělena na podstatná jména a slovesa. Ze sloves se pak stávají názvy metod a z podstatných jmen názvy tříd či atributů. Rozdělení v rámci tohoto projektu je uvedeno v následující tabulce (Tabulka 7).

**Tabulka 6: Metoda podstatných jmen a sloves**

| Klíčové slovo | Slovní druh | Prvek analytické třídy |
|---|---|---|
| Aktér | Podstatné jméno | Třída |
| Název aktéra | Podstatné jméno | Atribut |
| Správa aktérů | Podstatné jméno | Třída |
| Přidání aktéra | Sloveso | Metoda |
| Odstranění aktéra | Sloveso | Metoda |
| Editace aktéra | Sloveso | Metoda |
| Vypsání aktérů | Sloveso | Metoda |
| Uživatel | Podstatné jméno | Třída |
| Jméno uživatele | Podstatné jméno | Atribut |
| Příjmení uživatele | Podstatné jméno | Atribut |
| Uživatelské jméno | Podstatné jméno | Atribut |
| Heslo uživatele | Podstatné jméno | Atribut |
| Správa uživatelů | Podstatné jméno | Třída |
| Registrace uživatele | Sloveso | Metoda |
| Zrušení uživatelského účtu | Sloveso | Metoda |
| Úprava uživatelského účtu | Sloveso | Metoda |
| Projekt | Podstatné jméno | Třída |
| Autor projektu | Podstatné jméno | Atribut |
| Název projektu | Podstatné jméno | Atribut |
| Popis projektu | Podstatné jméno | Atribut |
| Správa projektů | Podstatné jméno | Třída |
| Vytvoření projektu | Sloveso | Metoda |
| Odstranění projektu | Sloveso | Metoda |
| Editace projektu | Sloveso | Metoda |
| Otevření projektu | Sloveso | Metoda |
| Člen projektu | Podstatné jméno | Třída |
| Správa členů projektu | Podstatné jméno | Třída |
| Přidání člena projektu | Sloveso | Metoda |
| Odebrání člena projektu | Sloveso | Metoda |
| Vypsání členů projektu | Sloveso | Metoda |

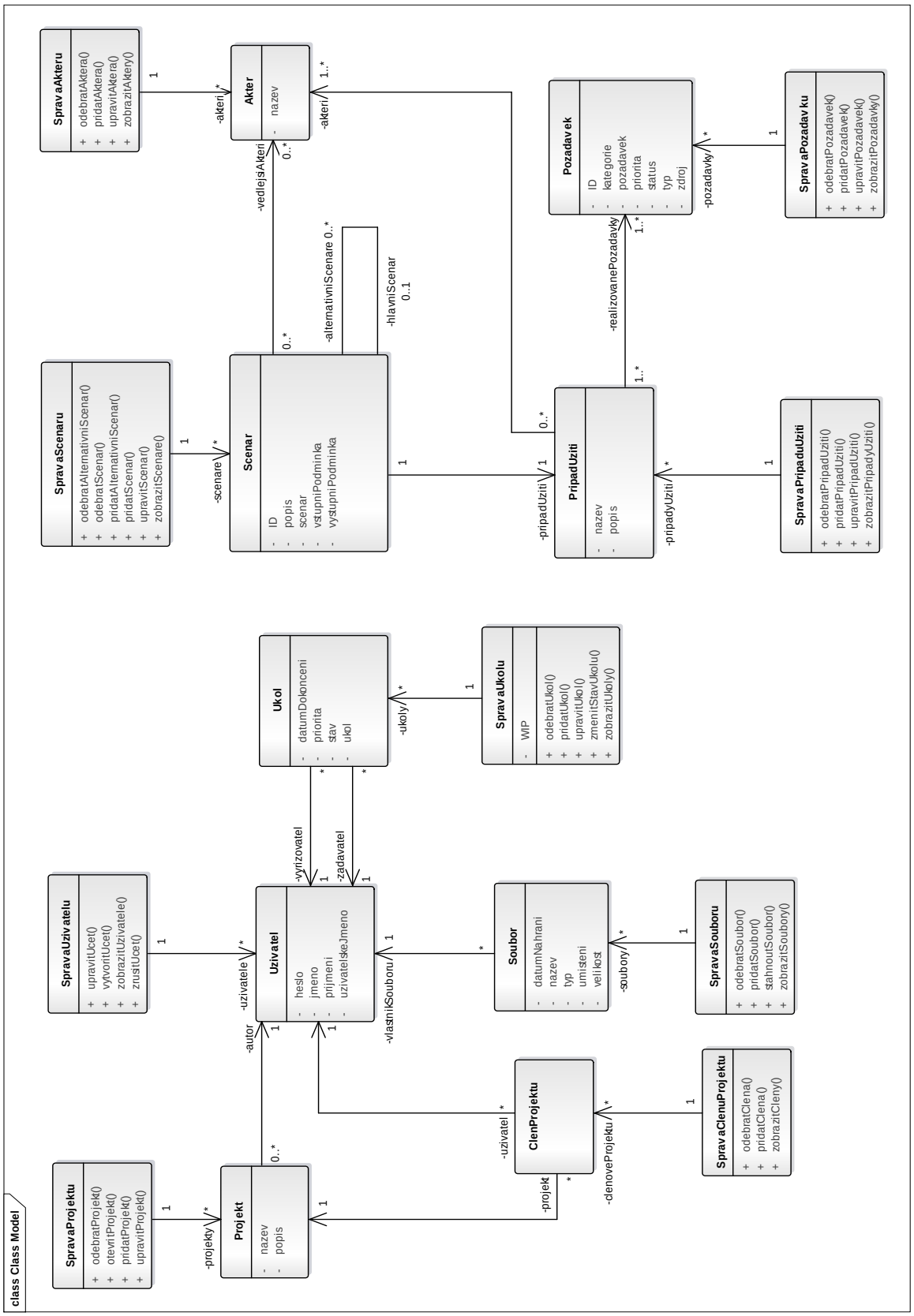
| Požadavek | Podstatné jméno | Třída |
|---|---|---|
| Znění požadavku | Podstatné jméno | Atribut |
| Zdroj požadavku | Podstatné jméno | Atribut |
| Kategorie požadavku | Podstatné jméno | Atribut |
| Priorita požadavku | Podstatné jméno | Atribut |
| Typ požadavku | Podstatné jméno | Atribut |
| Status požadavku | Podstatné jméno | Atribut |
| Identifikátor požadavku | Podstatné jméno | Atribut |
| Správa požadavků | Podstatné jméno | Třída |
| Vytvoření požadavku | Sloveso | Metoda |
| Odstranění požadavku | Sloveso | Metoda |
| Upravení požadavku | Sloveso | Metoda |
| Vypsání požadavků | Sloveso | Metoda |
| Případ užití | Podstatné jméno | Třída |
| Název případu užití | Podstatné jméno | Atribut |
| Popis případu užití | Podstatné jméno | Atribut |
| Realizované požadavky | Podstatné jméno | Atribut |
| Aktéři případu užití | Podstatné jméno | Atribut |
| Správa případů užití | Podstatné jméno | Třída |
| Vytvoření případu užití | Sloveso | Metoda |
| Odstranění případu užití | Sloveso | Metoda |
| Editace případu užití | Sloveso | Metoda |
| Vypsání případů užití | Sloveso | Metoda |
| Scénář | Podstatné jméno | Třída |
| Identifikátor scénáře | Podstatné jméno | Atribut |
| Popis scénáře | Podstatné jméno | Atribut |
| Vstupní podmínka scénáře | Podstatné jméno | Atribut |
| Hlavní aktéři scénáře | Podstatné jméno | Atribut |
| Vedlejší aktéři scénáře | Podstatné jméno | Atribut |
| Hlavní scénář | Podstatné jméno | Atribut |
| Výstupní podmínka scénáře | Podstatné jméno | Atribut |
| Alternativní scénáře | Podstatné jméno | Atribut |

| | | |
|---|---|---|
| Správa scénářů | Podstatné jméno | Třída |
| Přidání scénáře | Sloveso | Metoda |
| Odebrání scénáře | Sloveso | Metoda |
| Úprava scénáře | Sloveso | Metoda |
| Zobrazení scénáře | Sloveso | Metoda |
| Vypsání scénářů | Sloveso | Metoda |
| Přidání alternativního scénáře | Sloveso | Metoda |
| Odebrání alternativního scénáře | Sloveso | Metoda |
| Úkol | Podstatné jméno | Třída |
| Zadavatel úkolu | Podstatné jméno | Atribut |
| Vyřizovatel úkolu | Podstatné jméno | Atribut |
| Zadání úkolu | Podstatné jméno | Atribut |
| Datum dokončení úkolu | Podstatné jméno | Atribut |
| Stav úkolu | Podstatné jméno | Atribut |
| Priorita úkolu | Podstatné jméno | Atribut |
| Správa úkolů | Podstatné jméno | Třída |
| Vytvoření úkolu | Sloveso | Metoda |
| Odstranění úkolu | Sloveso | Metoda |
| Editace úkolu | Sloveso | Metoda |
| Vypsání úkolů | Sloveso | Metoda |
| Změna stavu úkolu | Sloveso | Metoda |
| Soubor | Podstatné jméno | Třída |
| Název souboru | Podstatné jméno | Atribut |
| Cesta k souboru | Podstatné jméno | Atribut |
| Velikost souboru | Podstatné jméno | Atribut |
| Typ souboru | Podstatné jméno | Atribut |
| Datum nahrání souboru | Podstatné jméno | Atribut |
| Vlastník souboru | Podstatné jméno | Atribut |
| Správa souborů | Podstatné jméno | Třída |
| Nahrání souboru | Sloveso | Metoda |
| Odstranění souboru | Sloveso | Metoda |
| Stažení souboru | Sloveso | Metoda |

| Vypsání souborů | Sloveso | Metoda |
|---|---|---|

Z takto předepsané tabulky podstatných jmen a sloves byl sestrojen analytický model tříd, který je na Obrázku 17. Na tomto obrázku lze nalézt celkem 9 tříd, jejichž název začíná slovem „Sprava“. Takové třídy zastupují z hlediska MVC architektury tzv. kontroléry. Každý kontrolér je přímo napojen na jednu modelovou třídu, o kterou se „stará“, resp. uchovává její instance a zprostředkovává nad nimi operace. Tyto modelové třídy samozřejmě nekomunikují pouze jen se svými kontroléry, ale i mezi sebou, a to následujícím způsobem: Vazba mezi třídami `Projekt` a `Uzivatel` říká, že každý projekt v sobě obsahuje odkaz na autora projektu neboli uživatele, který tento projekt vytvořil. Třída `ClenProjektu` slouží ke spojení třídy `Uzivatel` a `Projekt` ve smyslu vytvoření práva daného uživatele k tomuto projektu přistupovat. Třída `Soubor` má též vazbu se třídou `Uzivatel`. Zde se jedná o zachování informace o vlastníkovi daného soubor, resp. kdo daný soubor do systému nahrál. Poslední třída, která se přímo dotýká třídy `Uzivatel` je třída `Ukol`. U každého úkolu je potřeba vědět kdo z uživatelů úkol vytvořil a kdo ho má splnit. V tomto případě se tedy v modelu nacházejí dvě vazby ze třídy `Ukol` na třídu `Uzivatel`. Další třídou v modelu je třída `PripadUziti` obsahující seznam instancí třídy `Pozadavek`, které jsou daným případem užití realizovány. Stejně tak tato třída obsahuje i seznam instancí třídy `Akter`, které tento případ užití používají. Podobný seznam instancí třídy `Akter` obsahuje i třída `Scenar`. V tomto případě se ale jedná o vedlejší aktéry. Seznam hlavních aktérů má třída `Scenar` k dispozici také, a to díky vazbě na třídu `PripadUziti`. Třída `Scenar` je jedinou třídou v celém modelu, která odkazuje i sama na sebe. Je to z toho důvodu, že hlavní scénář obsahuje seznam alternativních scénářů, které jsou stejného charakteru.

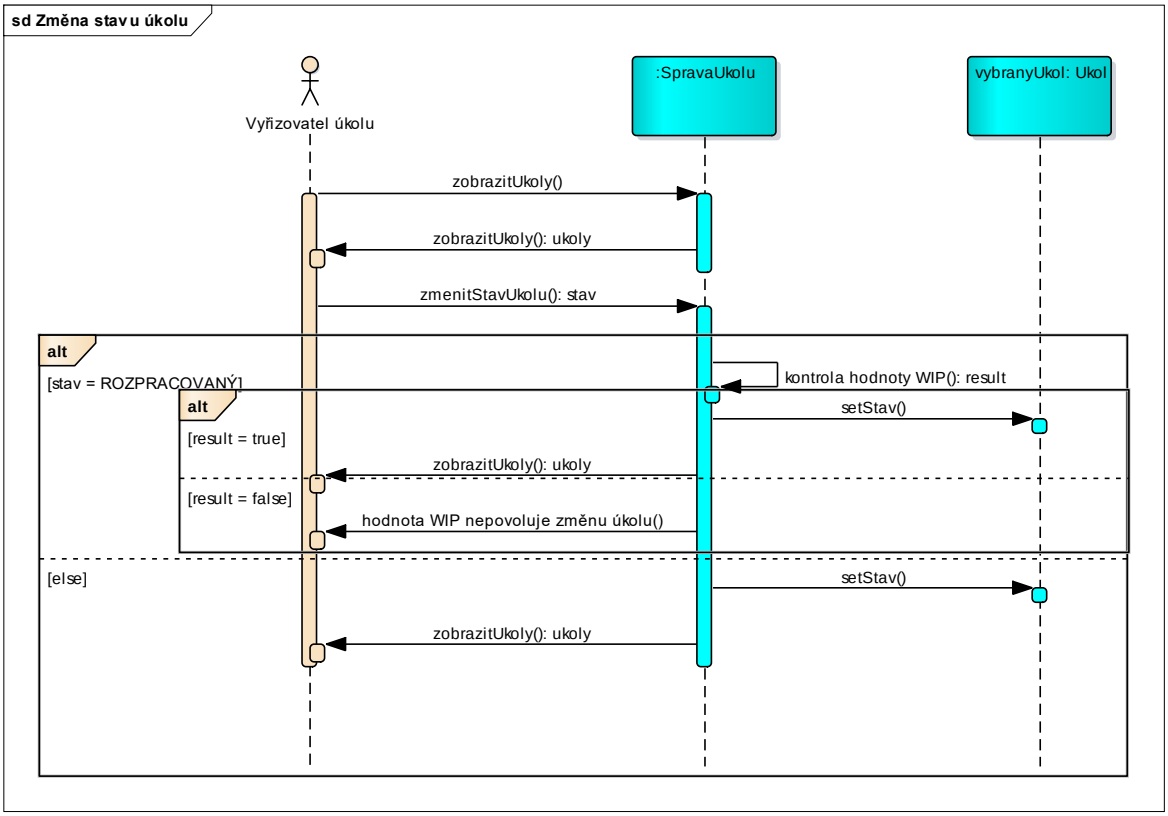
Z analytického modelu se později vytváří model návrhový. Ten je uveden v Příloze D.




**Obrázek 17: Analytický model tříd**

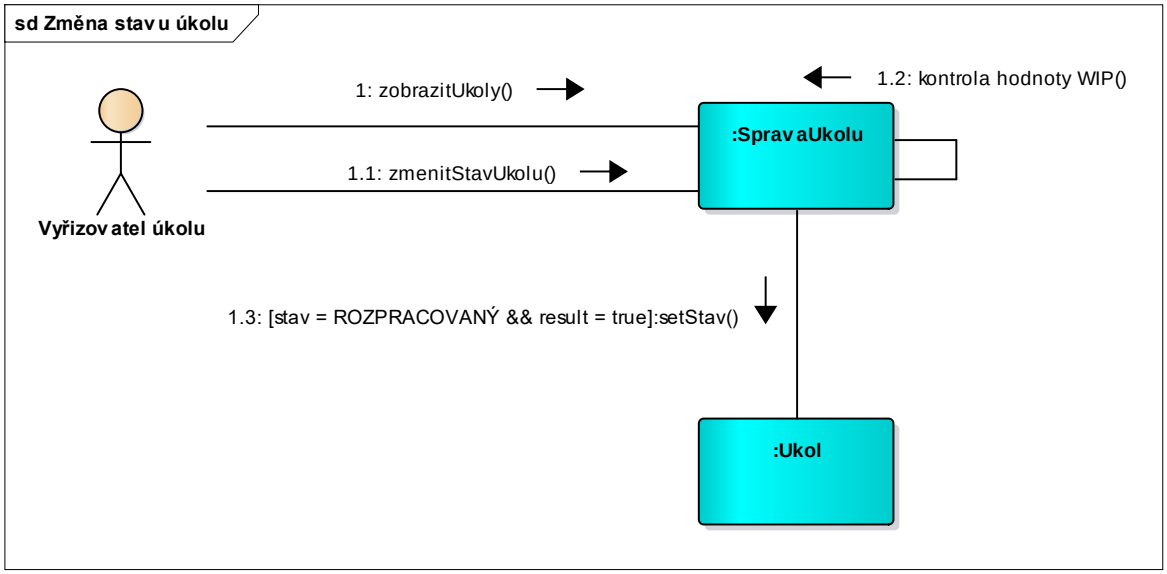
## 5.6 Realizace případů užití

Funkčnost a správnost případů užití, resp. scénářů se ověřuje pomocí sekvenčních či komunikačních diagramů. Ty zobrazují vzájemnou interakci mezi analytickými třídami a ověřují tak jejich schopnost spolupráce v rámci realizace daného scénáře. Pro účely demonstrace bude realizován scénář Změna stavu úkolu, který je popsán v kapitole 5.4. Sekvenční diagram je zobrazen na následujícím obrázku (Obrázek 18).




**Obrázek 18: Sekvenční diagram**

Jak lze vidět z obrázku, sekvenční diagram je velice podrobný. Je v něm jasně vidět čára života každého objektu, stejně tak jako zprávy, které si objekty předávají. U komunikačního diagramu (Obrázek 19) to tak podrobné již není. Diagram znázorňuje pouze volání metod nebo posílání požadavků.




**Obrázek 19: Komunikační diagram**

# 6 IMPLEMENTACE A TESTOVÁNÍ

Po vytvoření analýzy a návrhového modelu tříd následuje konečně fáze vývoje. V rámci této diplomové práce, která počítá s webovou aplikací, byl prvním krokem návrh databázového modelu, který nepřímo vychází z návrhového modelu tříd. Tento databázový model, který byl vytvořen v prostředí Microsoft SQL Server Management Studio, sestává z entit, které v návrhovém modelu představovaly pouze třídy tzv. modelové. Výraz model se již objevil v kapitole vysvětlující architekturu MVC (Model-View-Controller), která říká, že model je třída, která udržuje data. Proto se databázovému modelu říká datový. Datový model lze nalézt v Příloze E.

Po založení ASP.NET MVC projektu bylo zapotřebí vytvořit třídy, která budou udržovat data. K tomuto krásně posloužil právě databázový model, ze kterého Entity Framework pomocí své metody Database-First vytvořil modelové třídy. Ke každé takové modelové třídě musí existovat třída označená stereotypem Controller, která implementuje operace prováděné nad daným modelem. Ke každé operaci, implementované ve třídě typu Controller, pak existuje zobrazení, které vykresluje většinou nějaký formulář potřebný k provedení dané akce. V případě této webové aplikace jsou to soubory *.cshtml*, což jsou klasické HTML soubory s přidanou hodnotou toho, že dovolují do sebe zapisovat kód programovacího jazyka C#.

## 6.1 Kontrolér

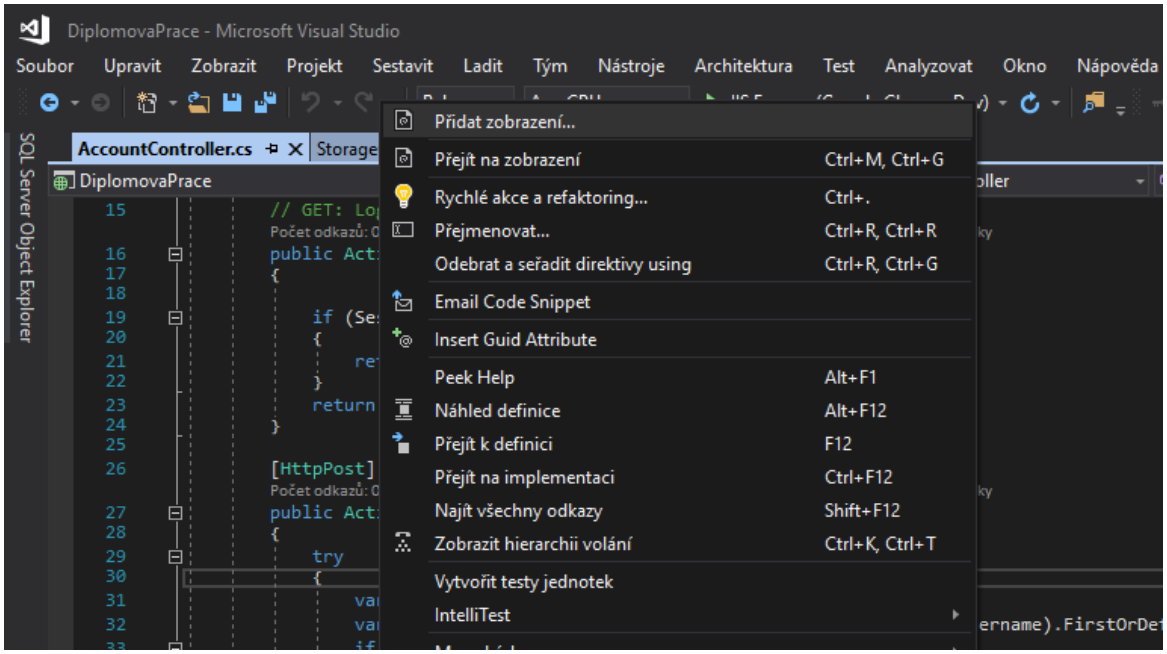
V ASP.NET všechny třídy typu Controller musí být potomky systémové třídy `Controller`, která pak umožňuje přístup k *Session, Request, Response* a jiným webovým objektům. Všechny veřejně přístupné metody v této třídě představují operace, na které se lze dotázat ve webovém prohlížeči pomocí adresy URL. Zápis takové URL adresy vypadá například takto: „https://www.mojestranka.cz/Account/Login“. *Account* zde představuje název kontroléru a *Login* název metody, která se pomocí této adresy zavolá. Další věcí, kterou je třeba rozlišovat, je druh HTTP volání, které si metodu žádá. Pro každý druh volání je implementována zvláštní metoda se stejným názvem, ale s odlišnou implementací. Metody jsou odlišeny pomocí anotace, která určuje, pro které HTTP volání je zrovna určena. Anotace se zpravidla zapisuje nad hlavičku metody do hranatých závorek s názvem daného volání (např. *[HttpGet]* nebo *[HttpPost]* a další). Toto tvrzení se dá přeložit tak, že pokud uživatel sám zavolá metodu pomocí zadání URL adresy do prohlížeče, zavolá se metoda s anotací *[HttpGet]*. Pokud se ovšem jedná o metodu, která má za úkol zpracovat nějaký formulář, volá se metoda s anotací *[HttpPost]*. Návratovým typem z těchto metod je abstraktní třída `ActionResult`, jejímž potomkem je například třída `ViewResult`, která má za úkol vykreslit předpokládané zobrazení nebo třída `RedirectToRouteResult`, která v podstatě volá jinou metodu, ať už z téže kontroléru, či z úplně jiného. Následující úryvek kódu zobrazuje, jak může vypadat implementace třídy typu Controller. Tento příklad vychází ze třídy `AccountController`, která se stará o registraci, či přihlášení uživatele do systému. Atribut má tato třída pouze jeden, a to atribut *db* typu `SDTEntities`, který představuje databázové spojení a přístup k databázovému modelu.

```
public class AccountController : Controller
   {
       private SDTEntities db = new SDTEntities();
       // GET: Login
       public ActionResult Login()
       {

           if (Session["userID"] != null)
           {
               return RedirectToAction("Index", "Home");
           }
           return View();
       }

       [HttpPost]
       public ActionResult Login(User user)
       {
           try
           {
               var password = GetHashString(user.Password);
               var userDetails = db.Users.Where(
               u => u.Username == user.Username).FirstOrDefault();
               if (userDetails == null)
               {
                   ViewBag.ErrorMessage = "Neexistující uživatelské jméno.";
                   return View();
               }
               if (userDetails.Password != password)
               {
                   ViewBag.ErrorMessage = "Zadané heslo není správné.";
                   return View();
               }

               Session["userID"] = userDetails.ID;
               Session["userName"] = userDetails.Username;
               Session["avatar"] = userDetails.Avatar;

               userDetails.LastActive = DateTime.Now;
               db.Entry(userDetails).State = EntityState.Modified;
               db.SaveChanges();
           }
           catch (Exception ex)
           {
               Console.WriteLine(ex.Message);
               ViewBag.ErrorMessage = "Něco se pokazilo. Opakujte prosím akci.;
               return View();
           }

           return RedirectToAction("Index", "Home");
       }
```

## 6.2 Zobrazení

Jak již bylo nastíněno výše, každá veřejně přístupná metoda ve třídě typu Controller k sobě musí mít adekvátní zobrazení ve formě CSHTML souboru. V prostředí Microsoft Visual Studia se dá takové zobrazení jednoduše vygenerovat, jak je ostatně vidět na Obrázku 20. Po kliknutí pravým tlačítkem myši do těla metody se objeví nabídka s možností „Přidat

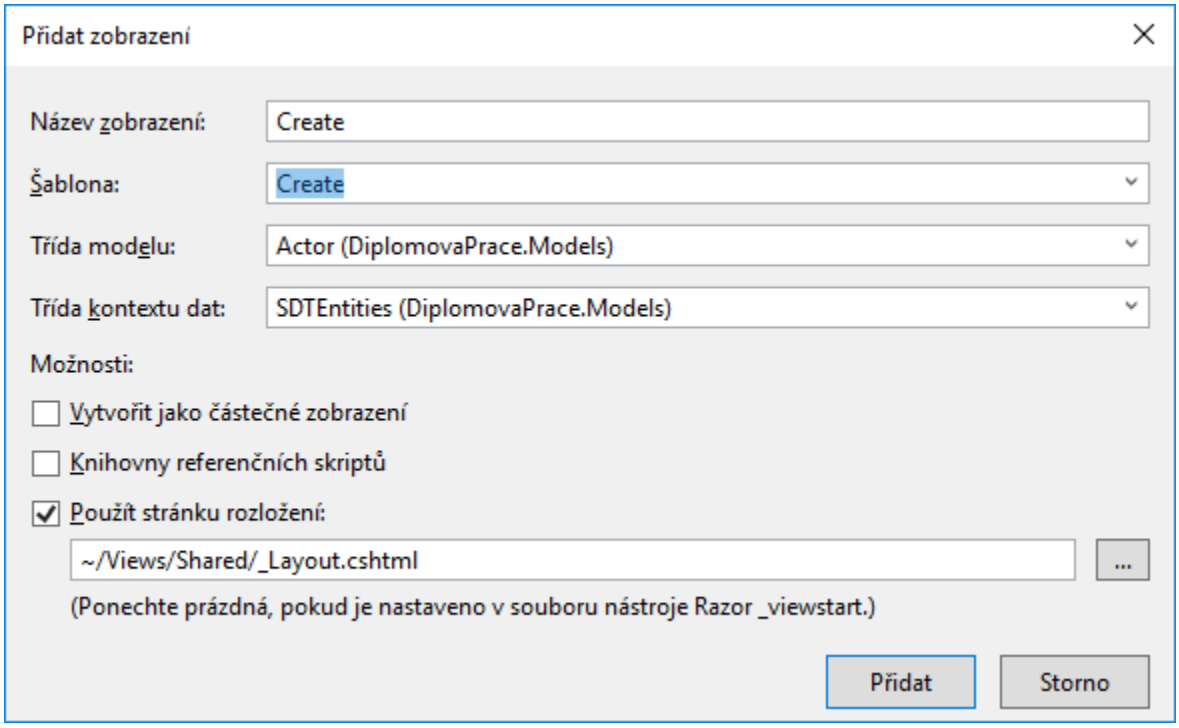
zobrazení…“, pokud již toto zobrazení existuje, lze využit i možnosti „Přejít na zobrazení“, které tento soubor otevře pro úpravu.




**Obrázek 20: Přidání zobrazení k metodě**

Po kliknutí na volbu „Přidat zobrazení…“ se objeví dialogové okno, do kterého se vyplní název zobrazení, může se vybrat šablona (Create, Delete, Details, Edit, Empty, List), ze které se předpřipraví předpokládaný formulář, a vybírá se modelová třída, která bude hrát v daném zobrazení klíčovou roli.

Na následujícím obrázku (Obrázek 21) lze vidět, jak může takové okno vypadat vyplněné. Je zřejmé, že nové zobrazení se bude jmenovat „Create“, šablona bude typu „Create“, čili se dá předpokládat vygenerovaný formulář obsahující vstupy adekvátní k atributům modelové třídy, v tomto případě třídy `Actor`. Další zde zaškrtnutou možností je použití stránky rozložení. Neboli, že se zachová layout hlavní stránky a k tomuto zobrazení nebude vytvářena speciální HTML hlavička či nějaké CSS styly.




**Obrázek 21: Dialogové okno při přidání zobrazení**

```
1   @model DiplomovaPrace.Models.Actor
2
3   @{
4       ViewBag.Title = "Create";
5       Layout = "~/Views/Shared/_Layout.cshtml";
6   }
7
8   <h2>Create</h2>
9
10  @using (Html.BeginForm())
11  {
12      @Html.AntiForgeryToken()
13
14      <div class="form-horizontal">
15          <h4>Actor</h4>
16          <hr />
17          @Html.ValidationSummary(true, "", new { @class = "text-danger" })
18          <div class="form-group">
19              @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2" })
20              <div class="col-md-10">
21                  @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-control" } })
22                  @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
23              </div>
24          </div>
25
26          <div class="form-group">
27              <div class="col-md-offset-2 col-md-10">
28                  <input type="submit" value="Create" class="btn btn-default" />
29              </div>
30          </div>
31      </div>
32  }
```

**Obrázek 22: Kód vygenerovaného zobrazení**

Na Obrázku 22 se nachází zdrojový kód vygenerovaného zobrazení dle předchozího nastavení. Na řádku číslo 1 se nachází informace o zvolené modelové třídě. Na řádku 4 se pak dá upravit titulek HTML stránky, který se přenese do hlavní HTML hlavičky stránky rozložení (layoutu), jejichž umístění udává řádek číslo 5. Na řádku 10 pak začíná HTML formulář zapsaný pomocí syntaxe Razor, která je pro ASP.NET MVC typická. Jako parametr metody *Html.BeginForm* pak přijde jméno kontroléru a metoda, která formulář po odeslání uživatelem vyhodnotí. Jelikož má třída `Actor` pouze jediný atribut Name, je zde pro něj připravena metoda *@Html.EditorFor*, která se později přeloží na klasický HTML tag

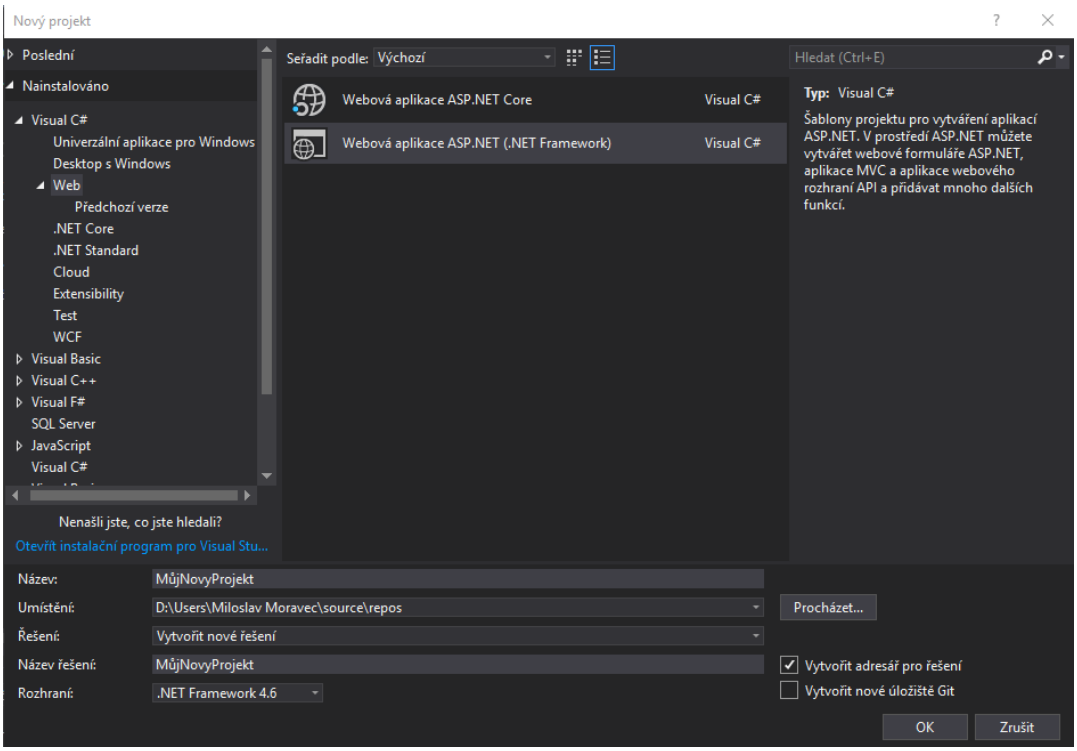
<input/>. Metoda *@Html.LabelFor* nedělá nic jiného, než že vygeneruje HTML tag <Label> se zvoleným označením atributu, které se dá v modelové třídě upravit pomocí anotace *[DisplayName]*. Modelová třída `Actor` vypadá například takto:

```
public partial class Actor
    {
        public Actor() { }

        public int ID { get; set; }
        [DisplayName("Jméno aktéra")]
        [Required(ErrorMessage = "Jméno aktéra je povinná položka")]
        [StringLength(50, ErrorMessage = "Maximální délka jména je 50 znaků")]
        public string Name { get; set; }
    }
```

Jak je vidět, pomocí anotace *[DisplayName]* se dá pohodlně z jednoho místa měnit zobrazovaný název daného atributu. Stejně tak se dá nastavit znění zprávy v případě nějaké chyby. Anotace *[Required]* se volá při chybě nevyplnění dané položky ve formuláři a anotace *[StringLength]* hlídá maximální velikost zadané hodnoty atributu. Znění takové chyby se pak vypíše v zobrazení do HTML tagu vygenerovaném metodou *@Html.ValidationMessageFor*. Tato validace ale úzce spolupracuje s javascriptovou knihovnou jQuery, takže je důležité mít tuto k dispozici.

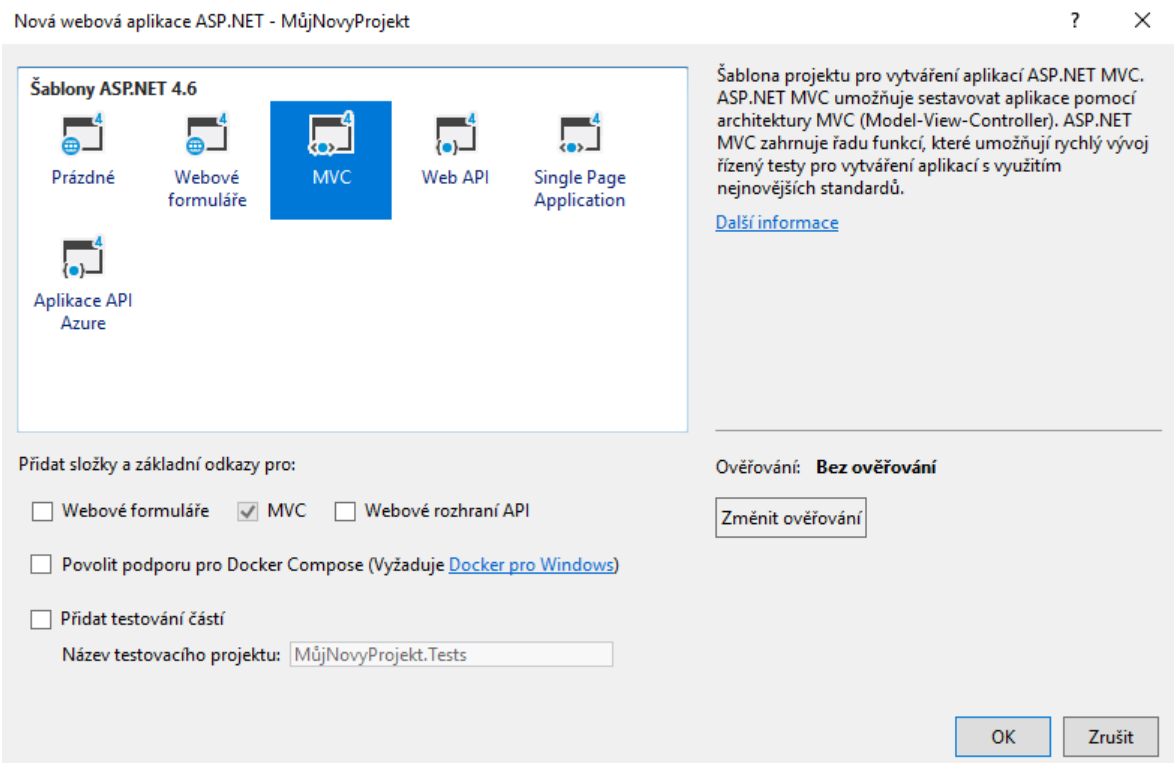
## 6.3 Založení projektu

Následující text vrátí pozornost opět na začátek implementace.



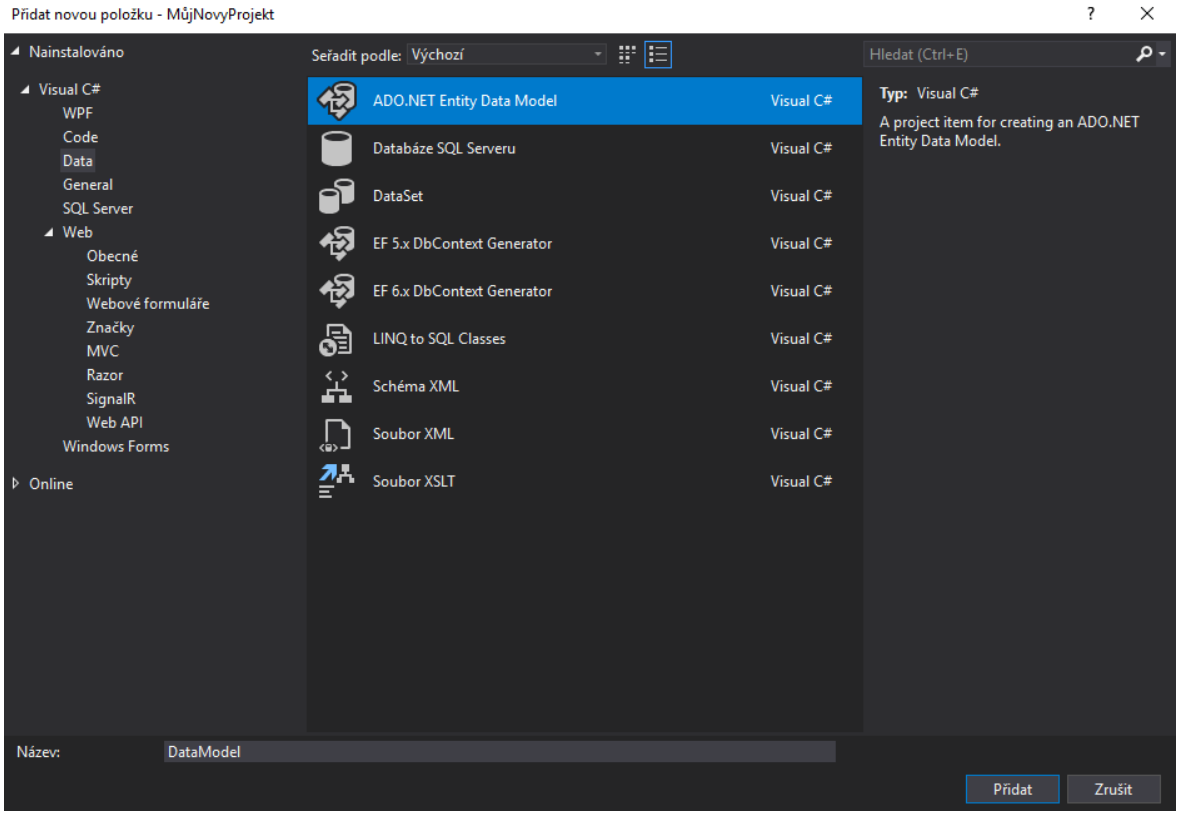

**Obrázek 23: Založení projektu**

Jak je možné vidět na Obrázku 23, tak pro založení nového ASP.NET projektu stačí kliknout na volbu „Nový projekt“ a v kategorii Web vybrat možnost „Webová aplikace ASP.NET (.NET Framework)“. Po potvrzení vybrané volby aplikace se objeví dialogové okno (Obrázek 24), ve kterém se vybírá šablona aplikace. Každá šablona má trochu jinou strukturu, v tomto případě bude vybrána možnost MVC. Aplikaci jde též nastavit i ověřování. Ověřování se týká přihlášení. Mohou to být individuální uživatelské účty, pracovní a školní účty nebo ověřování pomocí systému Windows. Při výběru volby „Individuální uživatelské účty“ bude mít projekt implementovanou možnost přihlášení přes Facebook, Twitter, Google a Microsoft. V tomto případě bude aplikace bez ověřování.



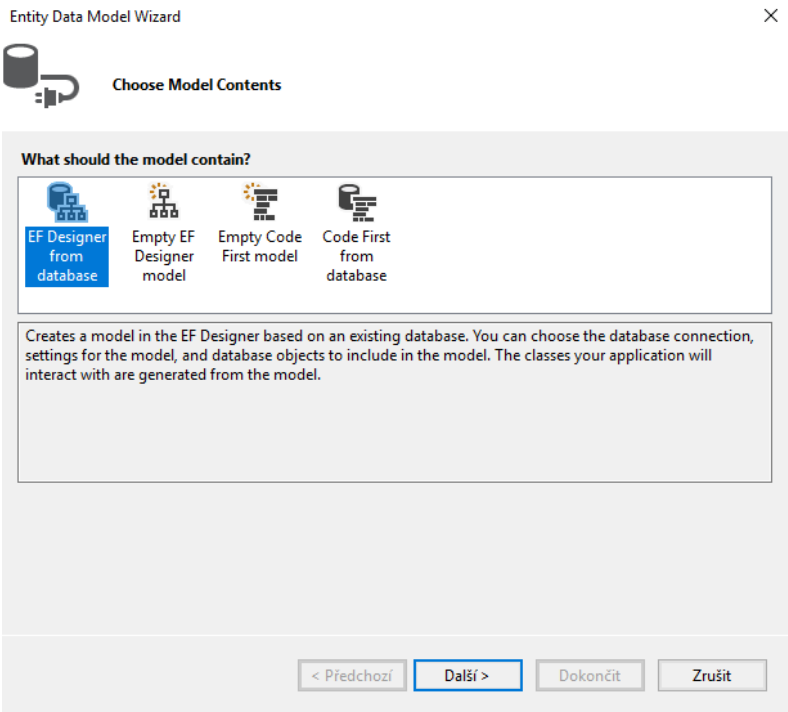

**Obrázek 24: Konfigurace nového projektu**

Po tomto posledním dialogovém okně je aplikace konečně vytvořena. Disponuje přesně takovou adresářovou strukturou, o které se píše v kapitole 4.2. Dalším úkolem je tedy vytvořit modelové třídy, ve kterých se budou uchovávat data a stav aplikace. Následuje tedy kliknutí pravým tlačítkem myši a vybrání možnosti „Přidat“ → „Nová položka…“. Objeví se okno (Obrázek 25), ve kterém se vybere položka „ADO.NET Entity Data Model“ ze záložky Data. Toto je hlavní prostředek Entity Frameworku pro objektově-relační mapování.



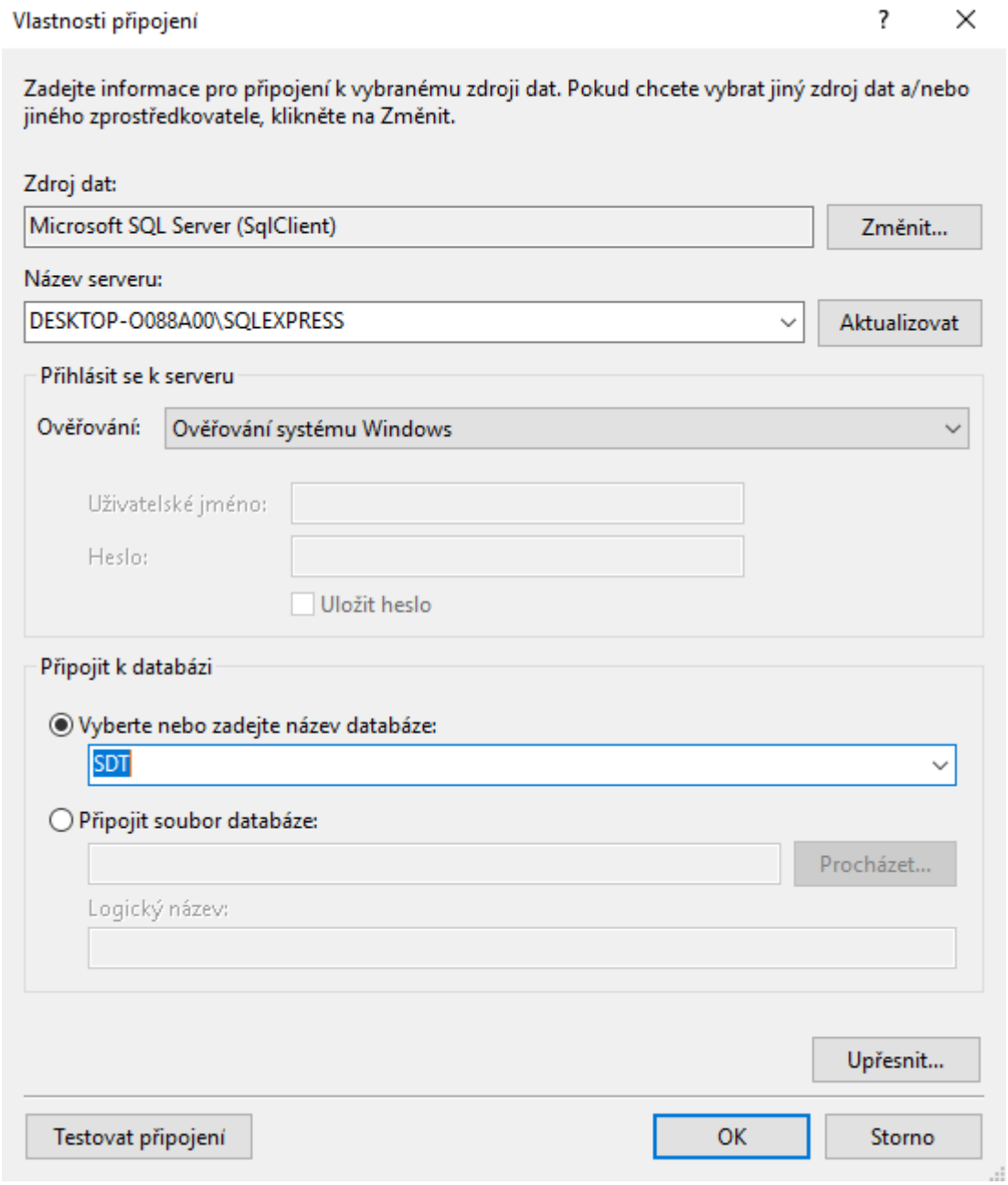

**Obrázek 25: Entity Data Model**

V dalším kroku (Obrázek 26) se vybírá, jakou metodu ORM mapování požadujeme, zda Database-First, Code-First, či Model-First. Pro tento projekt byla vybrána metoda Database-First.




**Obrázek 26: Výběr ORM mapování**

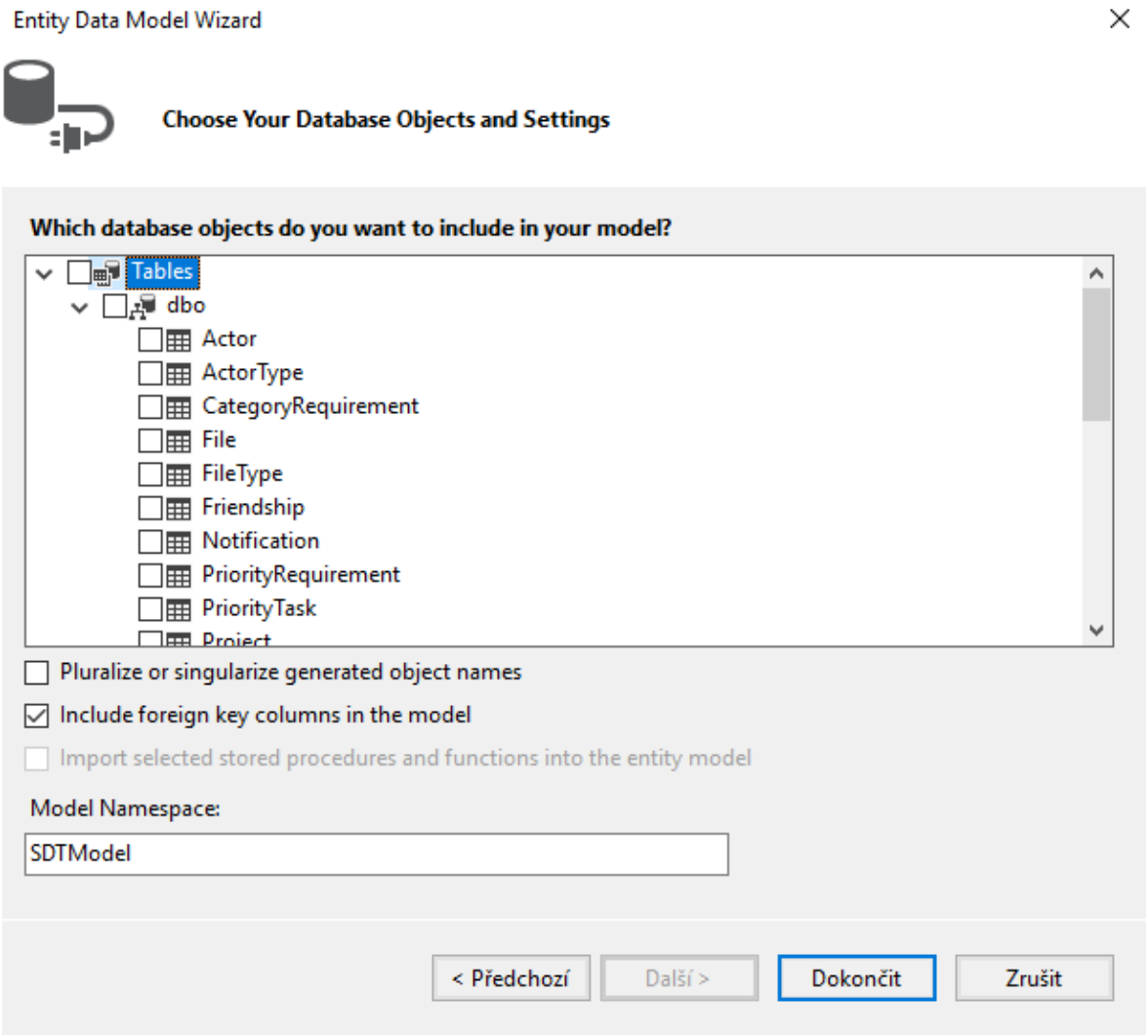
Jelikož byla vybrána metoda Database-First, dá se předpokládat, že dalším krokem bude připojení vytvořené aplikace k databázi. Po kliknutí na tlačítko „New Connection…“ se objeví dialogové okno (Obrázek 27), ve kterém se vyplní název serveru a případně přihlašovací údaje. Pokud jsou tyto dvě položky validní, zobrazí se v dolní části okna seznam databází, které byly na serveru nalezeny. Uživatel si poté vybere tu, se kterou chce v aplikaci pracovat. V tomto případě je to databáze SDT.




**Obrázek 27: Připojení k databázi**

Z vytvořeného databázového připojení se vytvoří tzv. „Connection string“, který se automaticky vkládá do souboru Web.config. Přes tento řetězec se pak aplikace připojuje do databáze a už se nikdy nepotřebuje znovu dotazovat na přihlašovací údaje či název serveru.

Na Obrázku 28 je vyobrazen posledním krok, což je výběr tabulek z databáze, které se mají stát třídami v právě vytvořené aplikaci. Z databáze se ale nemusí brát jen tabulky, mohou to být veškeré objekty, jako například procedury či funkce, ze kterých Entity Framework dokáže vytvořit standardní C# metody.




**Obrázek 28: Výběr databázových objektů**

V tomto okně se také zaškrtává volba „Pluralize or singularize generated object names“. Jedná se o jednotná či množná čísla názvů objektů. Pokud bude například třída `UseCase` uchovávat seznam objektů typu `Requirement`, bude se tato kolekce jmenovat „Requirements“. Doporučuje se tedy už při tvorbě databáze používat anglických slovíček na jména tabulek a sloupců. Pokud by se totiž daná třída jmenovala `Pozadavek`, Entity Framework by z kolekce jejich objektů udělal „Pozadaveks“, a to nevypadá hezky.

Modely jsou tedy vytvořené. Dalším krokem už je pak tvorba kontrolérů, které již byly vysvětleny na začátku této kapitoly. V další podkapitole bude brán zřetel na testování výsledného softwaru.

## 6.4 Testování

V této části bude představeno testování funkčnosti aplikace provedené metodou testování uživatelského rozhraní. Uživatelské rozhraní lze testovat buď ručně (tester kliká do formuláře a kontroluje, zda se aplikace chová dle předepsaných požadavků a omezení), anebo se využije testovacích nástrojů, mezi které patří například: Ranorex, Selenium, QTP, Cucumber, SilkTest, TestComplete či Squish GUI Tester. Pro tuto diplomovou práci byl zvolen nástroj Selenium WebDriver.

Selenium je bezplatný (open source) automatizační nástroj pro testování webových aplikací napříč různými webovými prohlížeči. Není to pouze jeden jediný nástroj, ale celá sada softwarů, z nichž každý se zabývá různými testovacími potřebami organizace. Tato sada se skládá ze čtyř komponent: [29]

- Selenium Integrated Development Environment (IDE);
- Selenium Remote Control (RC);
- WebDriver;
- Selenium Grid;

WebDriver je framework pro webovou automatizaci, který umožňuje provádět testy nad různými prohlížeči (Selenium IDE podporuje pouze Firefox). WebDriver dále umožňuje použití programovacího jazyku při vytváření testovacích skriptů. Podporované programovací jazyky jsou: Java, C#, PHP, Python, Perl a Ruby. [30]

V praxi může takový testovací kód vypadat například takto:

```
1 using System;
2 using System.Threading;
3 using Microsoft.VisualStudio.TestTools.UnitTesting;
4 using OpenQA.Selenium;
5 using OpenQA.Selenium.Chrome;
6 using OpenQA.Selenium.Support.UI;
7
8 namespace DiplomovaPrace.Test
9 {
10     [TestClass]
11     public class UITest
12     {
13         private IWebDriver driver = new ChromeDriver(@"");
14
15         [TestMethod]
16         public void TestLogin()
17         {
18             driver.Navigate().GoToUrl("http://www.milamoravec.cz");
19             driver.FindElement(By.Id("Username")).SendKeys("Skipper");
20             driver.FindElement(By.Id("Password")).SendKeys("špatné heslo");
21             driver.FindElement(By.TagName("button")).Click();
22             string message = driver.FindElement(By.TagName("strong")).Text;
23             string expectedMessage = "Zadané heslo není správné.";
24             Assert.AreEqual(expectedMessage, message);
25         }
```

Na řádcích 4, 5 a 6 se nachází import knihoven potřebných pro práci s nástrojem Selenium. Na řádku 13 se provádí deklarace třídy `IWebDriver` se současnou inicializací na ChromeDriver, tudíž prohlížeč, na kterém bude kód testován bude Chrome. Do konstruktoru třídy `ChromeDriver` se vkládá cesta ke složce, ve které se knihovna ChromeDriver nachází. Pro přehlednost představovaného kódu je tento konstruktor prázdný.

Testovací metoda *TestLogin* funguje následovně:

1. WebDriver otevře adresu dle zadaného URL.
2. WebDriver na dané stránce nalezne webový element, jehož ID je *Username* a zapíše do něj uživatelské jméno „Skipper“.
3. WebDriver na dané stránce nalezne webový element, jehož ID je *Password* a zapíše do něj heslo „špatné heslo“.
4. WebDriver na dané stránce nalezne webový element, jehož tag je *button* a klikne na něj.
   4.1. Jelikož se na dané stránce nachází jen jediné tlačítko, a to tlačítko pro přihlášení, provede se přihlášení. Zadané heslo pro daného uživatele ale není správné, a tak by se mělo objevit upozornění v HTML tagu *strong* říkající, že zadané heslo není správné.
5. WebDriver na dané stránce nalezne webový element, jehož tag je *strong* a jeho obsah uloží do proměnné *message*.
6. Do proměnné *expectedMessage* se uloží očekávaná zpráva o tom, že heslo není správné.
7. Provede se porovnání získané zprávy s očekávanou, zda aplikace opravdu vrátila předvídanou chybu.

Tímto způsobem byla ověřena většina důležitých funkcionalit aplikace.

# 7 UŽIVATELSKÁ PŘÍRUČKA

Poslední kapitolou této diplomové práce je uživatelská příručka, která by měla uživateli představit všechny možnosti a funkcionality, které vytvořená aplikace nabízí. Taktéž by měla uživatele seznámit s ovládáním aplikace a jejím nasazením do reálného světa.

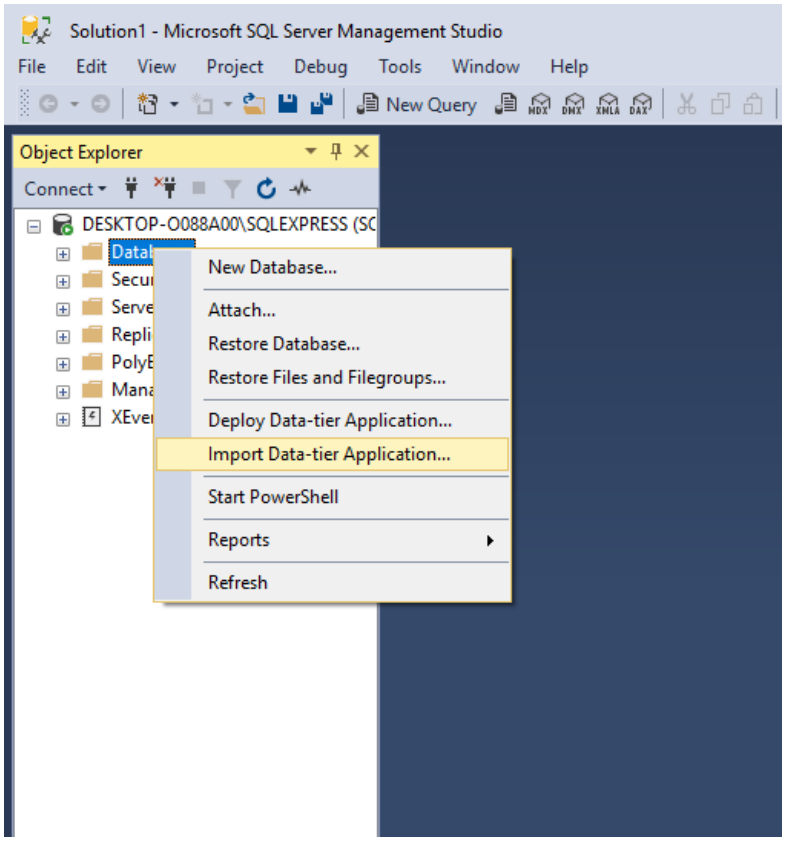
## 7.1 Nasazení aplikace

K nasazení aplikace jsou potřeba následující programy a nástroje:

- Microsoft SQL Server,
- Microsoft SQL Management Studio,
- Microsoft Visual Studio.

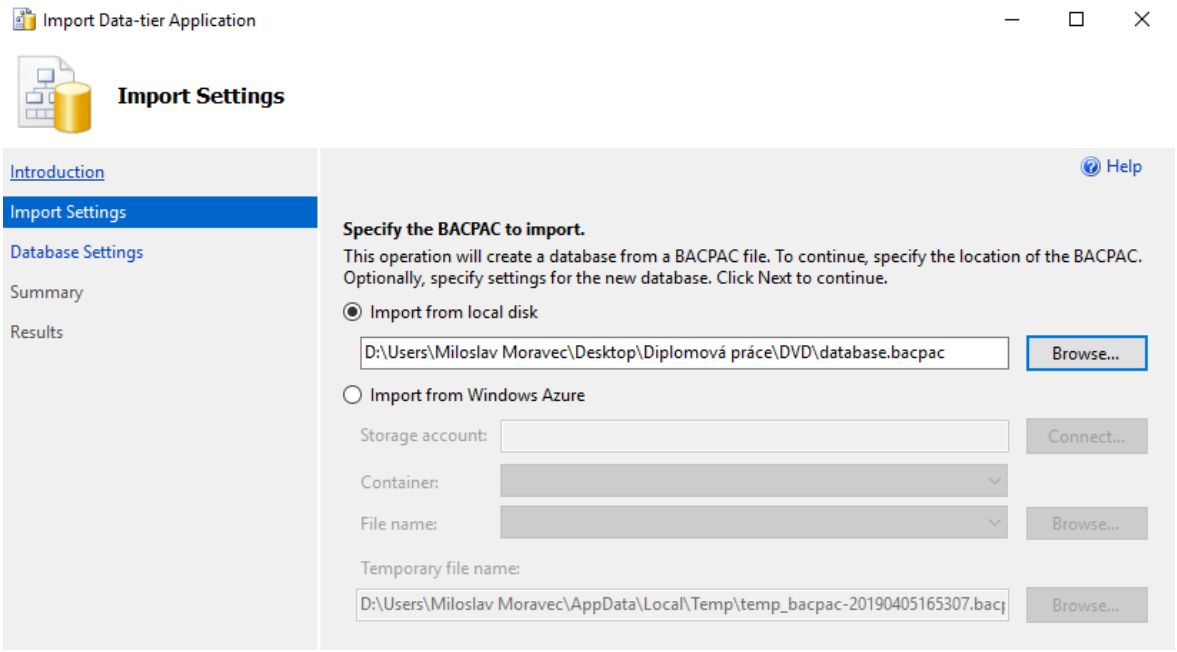
Instalace a konfigurace prvních dvou nástrojů je přehledně popsána ve zdroji [28].

V Příloze F se nachází soubor *database.bacpac* ten je nutné naimportovat do programu Microsoft SQL Management Studio, a vytvořit tak z něj databázi.



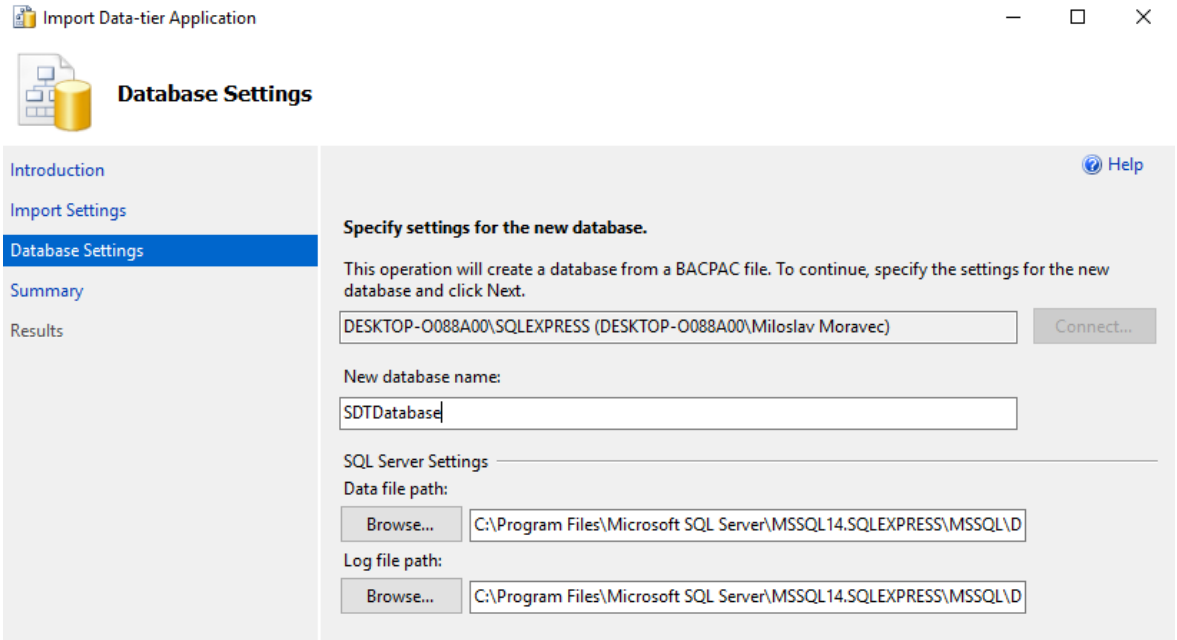

**Obrázek 29: Import databáze, krok I**

Dle obrázku 29 se po kliknutí pravým tlačítkem na složku *Databases* objeví nabídka, ze které se vybere možnost „Import Data-tier Application…“. Následně se objeví dialogové okno (Obrázek 30) s kolonkou „Import from local disk“, díky níž se vybere soubor (v tomto případě *database.bacpac)* udržující data pro vytvoření databáze.




**Obrázek 30: Import databáze, krok II**

Posledním krokem (Obrázek 31) je zvolení názvu databáze. V tomto případě byl zadán název *SDTDatabase*, ale je to čistě libovolné. V dalším okně se pouze klikne na tlačítko Finish a vytvoří se databáze.



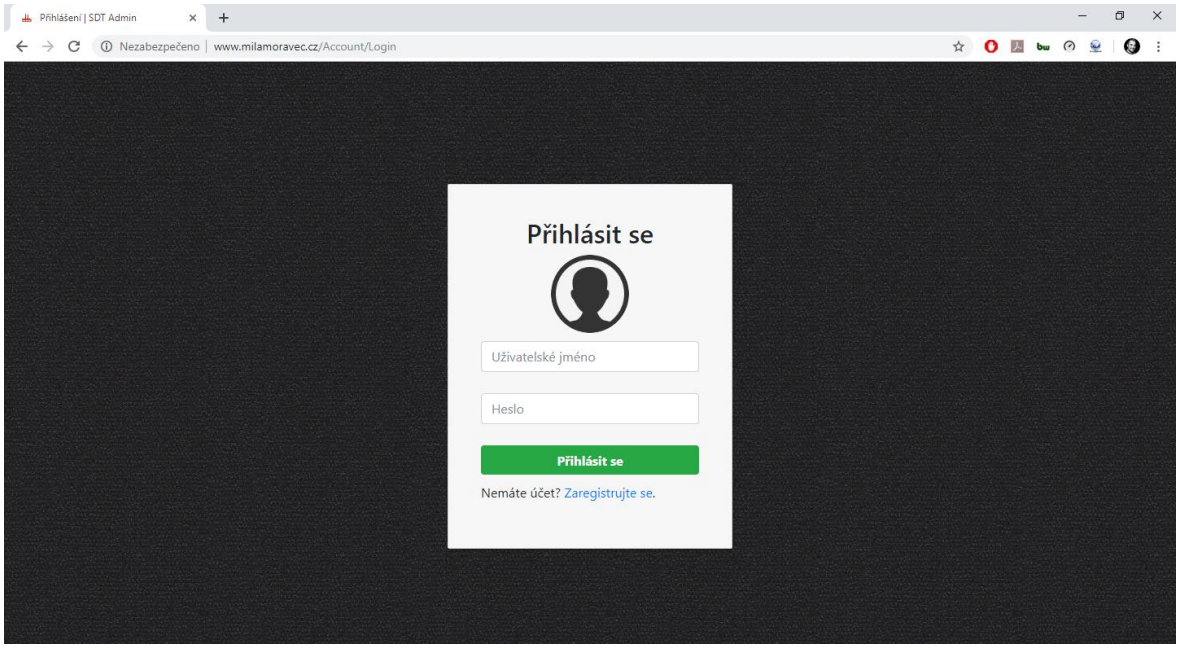

**Obrázek 31: Import databáze, krok III**

Tímto je práce v programu Microsoft SQL Management Studio ukončena a pozornost bude mít vývojové prostředí Microsoft Visual Studio. V něm se otevře projekt s názvem *DiplomovaPrace* (vizte Příloha F). Dalším úkolem bude svázání tohoto projektu s právě

vytvořenou databází. Projekt má pro tyto potřeby předem odstraněny datové třídy představující tabulky databáze. Dalším krokem bude tedy jejich vytvoření pomocí postupu popsaného na straně 64 této práce. Po vytvoření modelových tříd by nyní měla být aplikace plně připravena k použití.

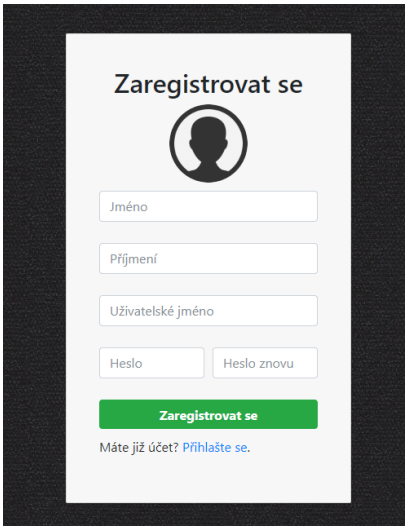
## 7.2 Registrace a přihlášení

Po spuštění aplikace se jako první objeví přihlašovací okno (Obrázek 32), které uživatele (po zadání uživatelského jména a hesla) přihlásí k jeho uživatelskému účtu.




**Obrázek 32: Přihlašovací okno**

Pokud ale zatím uživatel žádným účtem nedisponuje, musí kliknout na odkaz „Zaregistrujte se“, který ho přenese na registrační formulář (Obrázek 33).




**Obrázek 33: Registrační formulář**

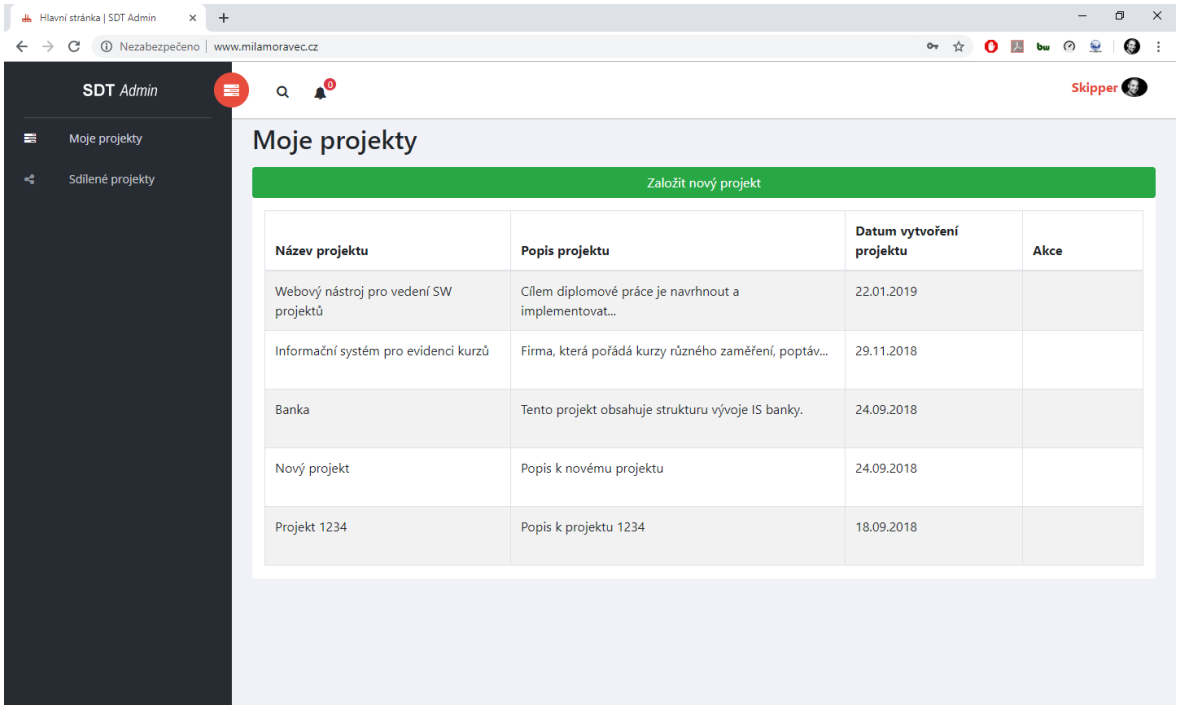
Každý nový uživatel musí při registraci vyplnit následující položky:

- jméno,
- příjmení,
- uživatelské jméno,
- heslo a
- heslo ještě jednou pro kontrolu.

Při registraci mohou nastat dva druhy chyb. První chybou je, že uživatel si vybral takové uživatelské jméno, které již někdo vlastní a musí si tedy vymyslet jiné. Druhou chybou je pak neshoda zadaných hesel. Je možné, že se uživatel upsal, a tudíž musí provést registraci znovu.

Po úspěšné registraci systém znovu zobrazí přihlašovací okno, kde uživatel své přihlašovací údaje uplatní k prvnímu vstupu do systému.

## 7.3 Uživatelské prostředí




**Obrázek 34: Hlavní stránka**

Po přihlášení do systému se jako první objeví hlavní stránka (Obrázek 34) zobrazující seznam vlastních vytvořených projektů uživatele. U každého projektu lze vidět jeho název, popis a datum vytvoření. Nad tímto seznamem se nachází zelené tlačítko „Založit nový projekt“, které uživateli otevře formulář k vytvoření projektu.

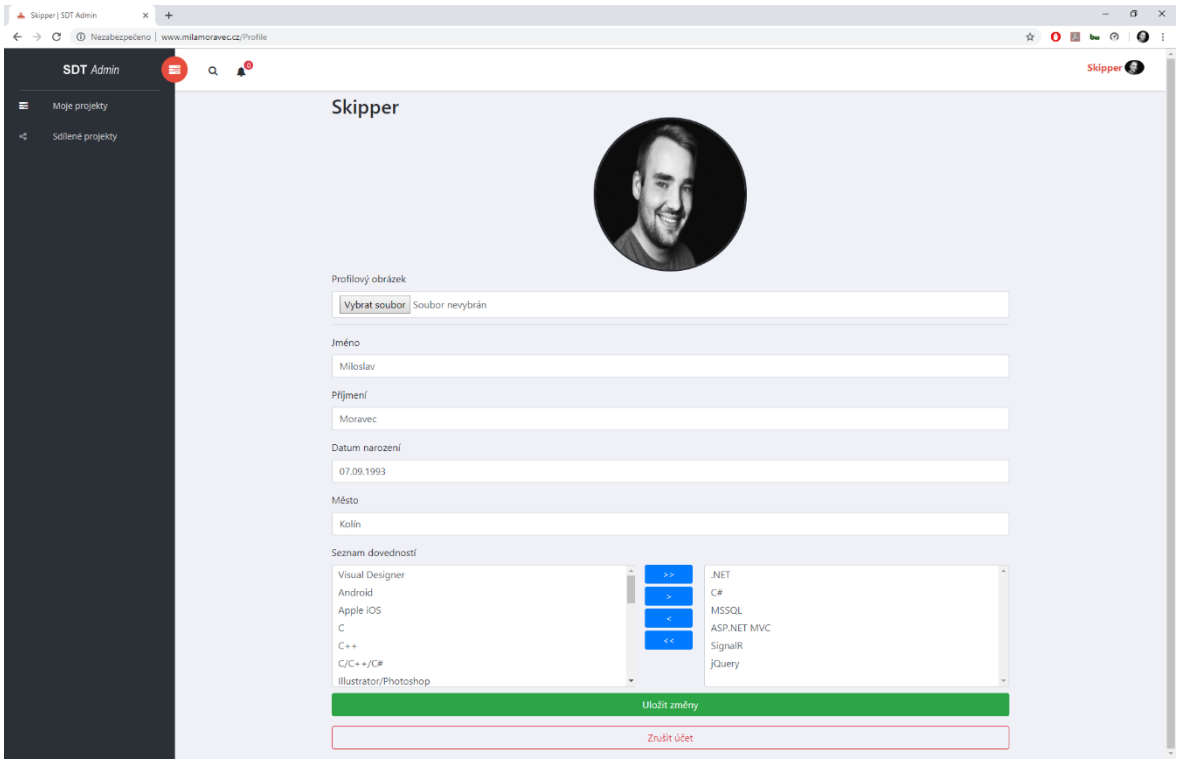
Na levé straně okna se nachází tmavý panel s dvěma záložkami. První záložka („Moje projekty“) představuje právě otevřené okno se seznamem vlastních projektů. Druhá

záložka („Sdílené projekty“) pak vypisuje stejným způsobem seznam cizích projektů, ke kterým má uživatel přístup, ale nevlastní je.

Horní bílá lišta se skládá ze tří komponent. Z lupy sloužící k vyhledávání jiných uživatelů, ze zvonečku upozorňujícího na provedené akce v nějakém projektu a z uživatelského jména, na které když se klikne, objeví se tři možnosti akce: „Můj profil“ (slouží k úpravě osobních údajů), „Kontakty“ (zobrazí seznam kontaktů či žádostí o navázání kontaktu) a „Odhlásit se“ (pro bezpečné odhlášení ze systému).

### 7.3.1 Úprava osobních údajů

Po kliknutí na tlačítko „Můj profil“ se zobrazí následující okno (Obrázek 35) s formulářem sloužícím k editaci osobních údajů či zrušení uživatelského účtu.

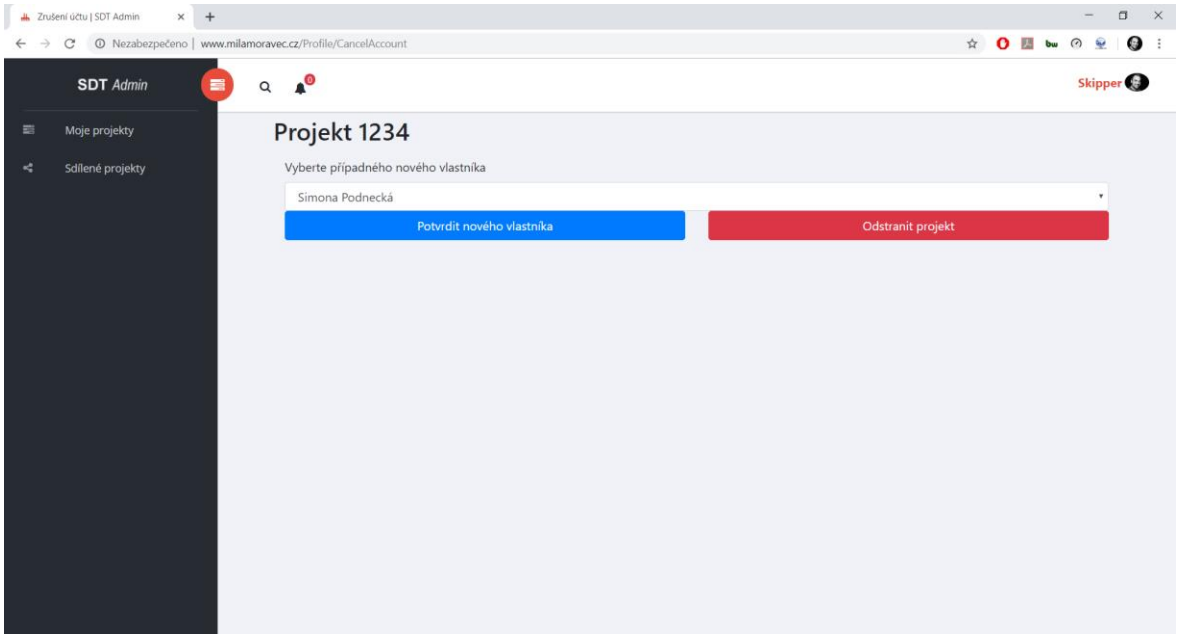



**Obrázek 35: Úprava osobních údajů**

Formulář nabízí možnost nahrání profilového obrázku, úpravu jména a příjmení, vyplnění datumu narození či města, ve kterém uživatel bydlí. Dále disponuje výběrem dovedností ze seznamu technologií a programovacích jazyků, díky kterým může dát uživatel světu vědět v čem pracuje a co ho zajímá. Na konci formuláře se dále nachází tlačítko „Zrušit účet“, o kterém bude řeč v následující části textu.

### 7.3.2 Zrušení účtu

Účet lze zrušit stisknutím tlačítka, o kterém bylo řečeno v předcházejícím textu. Po stisknutí tlačítka systém zkontroluje, zda uživatel vlastní nějaké projekty, ke kterým mají přístup jiní uživatelé. Pokud tato situace nastane, zobrazí se následující okno (Obrázek 36), které postupně projde všechny uživatelovy projekty, ke kterým má někdo přístup a nabídne mu,



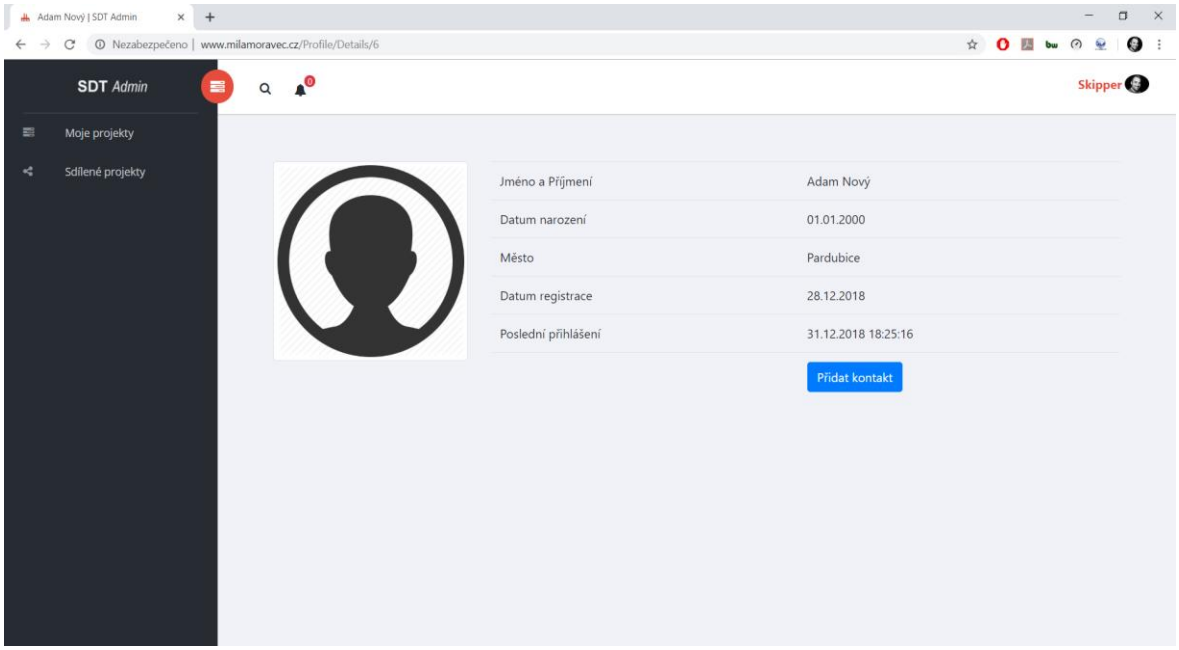

**Obrázek 36: Výběr nového vlastníka projektu**

zda si vybere z těchto lidí člověka, kterému projekt předá a novým vlastníkem se stane on, nebo se projekt úplně odstraní. Po provedení této akce u všech projektů systém zruší uživatelský účet a uživatele odhlásí.

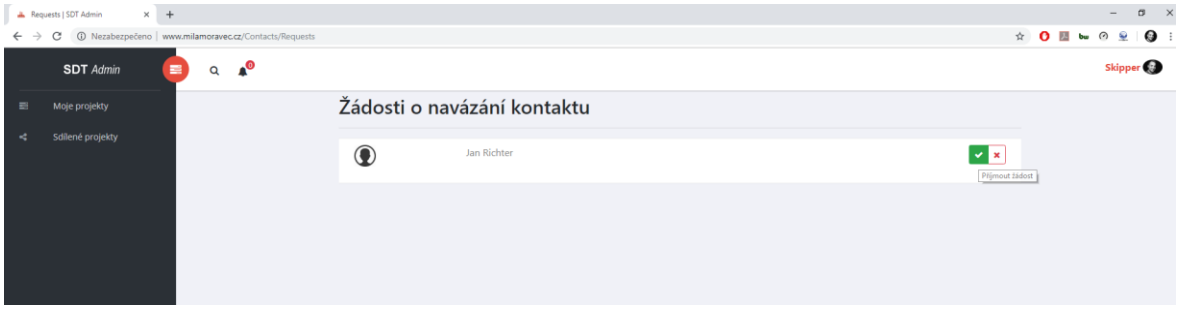
### 7.3.3 Kontakty

V této sekci aplikace se nachází seznam uživatelů, se kterým je daný uživatel v kontaktu. Z tohoto seznamu lze daný kontakt odstranit a ukončit tak s uživatelem jakoukoliv vazbu. Navazování kontaktů je důležité pro týmovou spolupráci, protože tým lze skládat pouze z lidí, které se nachází v kontaktech. Než vznikne nový kontakt je nutné daného uživatele vyhledat pomocí lupy v horní části aplikace dle jeho jména. Zobrazí se profil uživatele (Obrázek 37) s tlačítkem „Přidat kontakt“. Po kliknutí na toto tlačítko se vytvoří žádost, kterou daný uživatel musí schválit. Po schválení této žádosti uživatelem je úspěšně kontakt vytvořen. Informaci o schválení žádosti či o nově přichozí dostává uživatel formou upozornění (zvoneček v horní části obrazovky).

Upozornění uživatele dovede k seznamu žádosti o navázání kontatu (Obrázek 38), ve kterém se rozhodne, zda žádost příjme, či nepříjme. V okamžiku, kdy žádost příjme, odešle se upozornění odesílateli žádosti informující ho o skutečnosti, že žádost byla schválena.




**Obrázek 37: Přidání kontaktu**



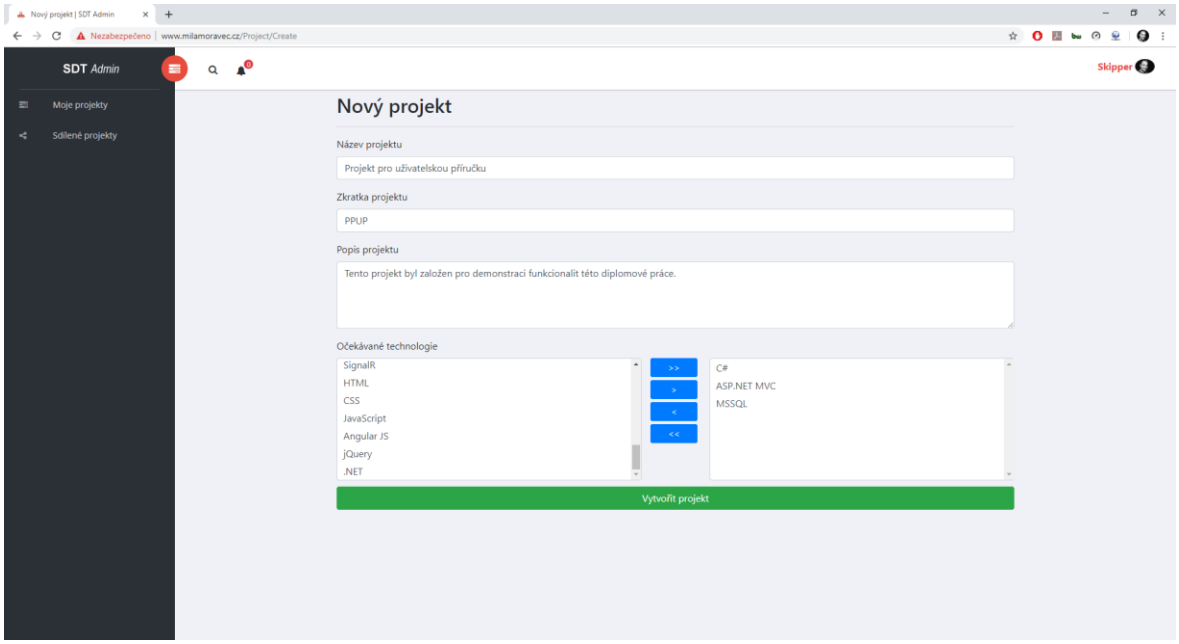

**Obrázek 38: Žádost o navázání kontaktu**

## 7.4 Správa projektu

Následující podkapitola se bude zabývat správou projektu a všech jeho částí, tj. správou požadavků, správou aktérů, správou případů užití a správou scénářů.

### 7.4.1 Založení projektu

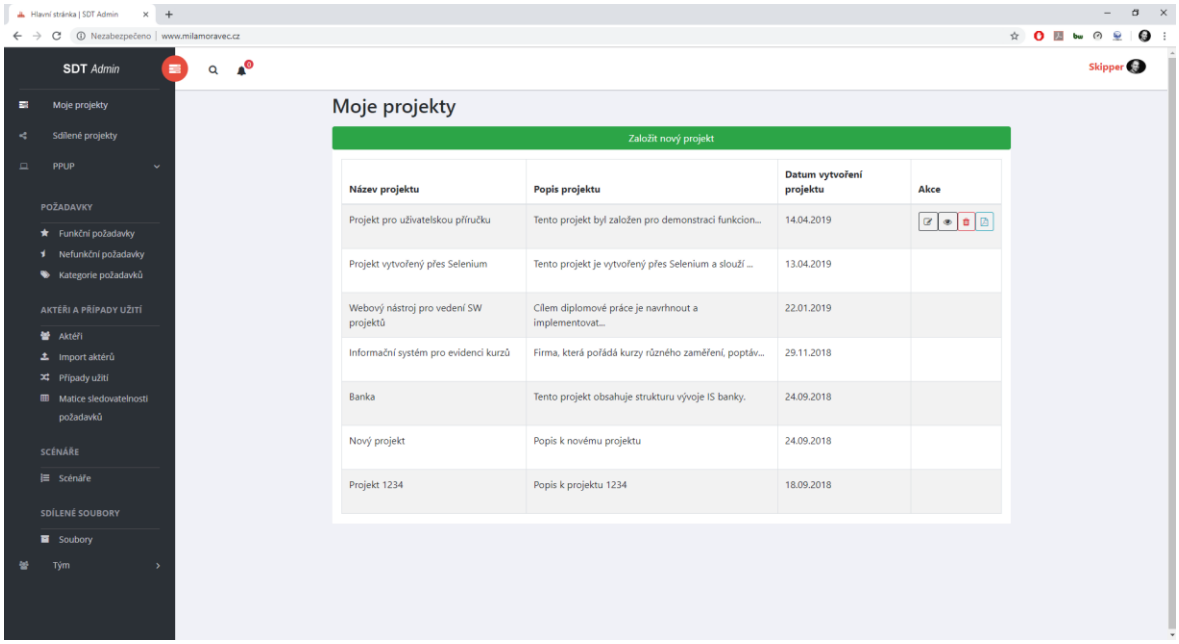
Na hlavní stránce (Obrázek 34) v záložce „Moje projekty“ se nachází tlačítko „Založit nový projekt“, které zobrazí formulář (Obrázek 39) pro založení nového projektu. Na tomto formuláři se nachází textová pole pro zadání názvu projektu a popisu projektu. Zkratka projektu se může samozřejmě vyplnit také, ale v základu se vyplňuje automaticky podle prvních písmen delšího názvu projektu. Dále se ze seznamu technologií vybírají očekávané technologie, které se budou pravděpodobně využívat při implementaci tohoto projektu. Za zmínku stojí, že uživatel si ve svém profilu mohl vybrat z těch samých technologií ty, které ovládá. Tato skutečnost pak bude pomáhat pro volbu nových členů projektu, protože se bude vybírat podle nejvíce shod mezi technologiemi projektu a dovednostmi uživatele.




**Obrázek 39: Založení projektu**

Po úspěšně vytvořeném projektu se tento objeví v seznamu projektů (Obrázek 40). Každý projekt s sebou nese čtyři tlačítka: Úprava projektu (přepsání nějakých informací, které se uvádí při vytváření projektu), Otevření projektu (zobrazí projekt včetně všech jeho náležitostí na levé liště), Odstranění projektu a tlačítko pro stažení dokumentace k projektu.

Projekt, který byl právě vytvořen je zároveň i otevřen a zobrazen ve formě záložky na levé liště.

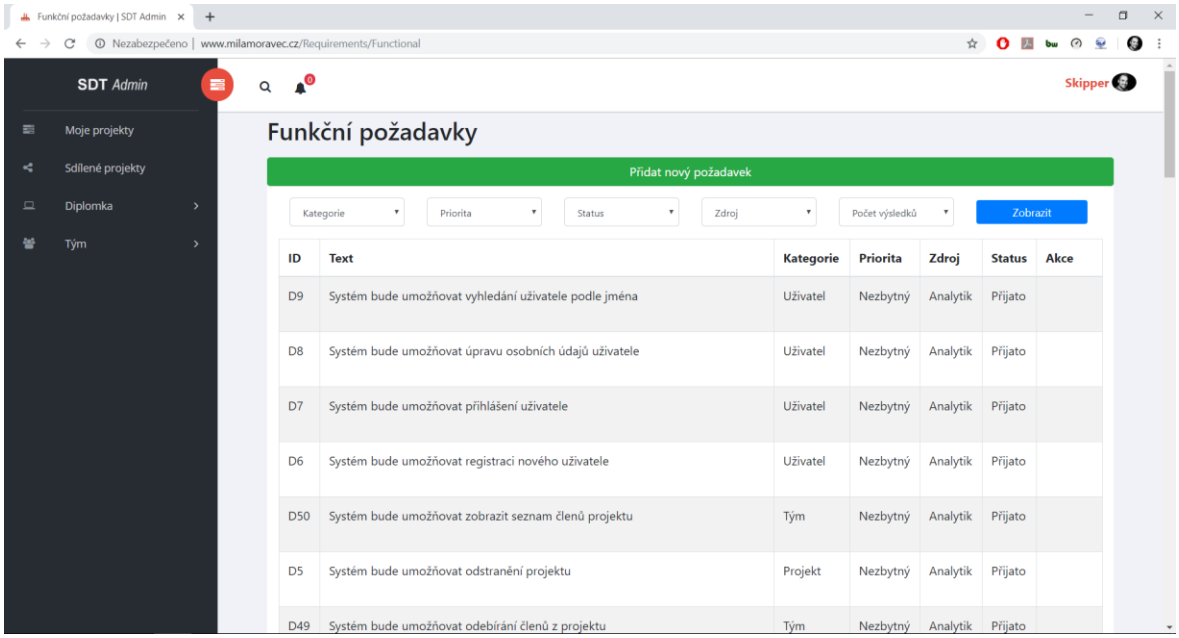



**Obrázek 40: Otevření projektu**

### 7.4.2 Správa požadavků

První položkou na levé liště pod právě otevřeným projektem jsou požadavky. Dělí se zde na podkategorie: Funkční požadavky, Nefunkční požadavky a Kategorie požadavků. Po kliknutí na možnost „Funkční požadavky“ se zobrazí seznam funkčních požadavků (Obrázek 41) udržující informace o požadavcích, jako jsou:

- ID požadavku,
- znění požadavku,
- kategorie požadavku,
- priorita požadavku,
- zdroj požadavku,
- status požadavku.




**Obrázek 41: Požadavky**

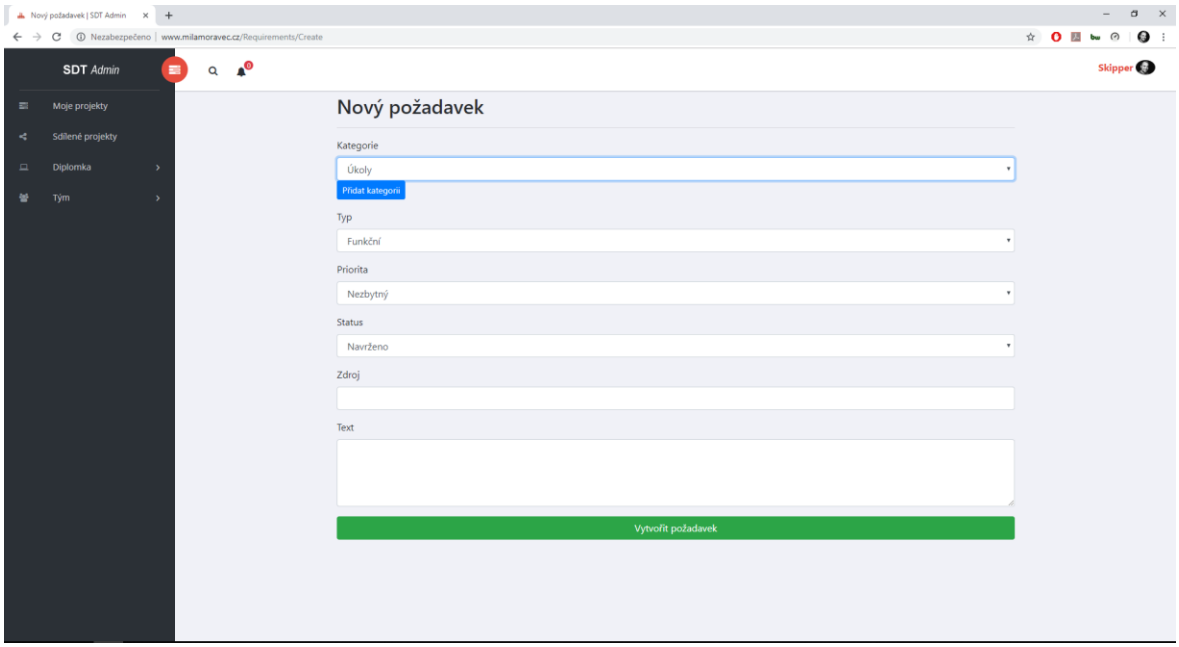
Podle těchto atributů požadavku lze požadavky i filtrovat včetně ovlivnění počtu výsledků, které se zobrazí na obrazovce. Každý požadavek pak lze upravit, či odstranit v kolonce Akce.

Vytvoření nového požadavku se provádí pomocí formuláře (Obrázek 42), který se vyvolá po kliknutí na tlačítko „Přidat nový požadavek“. V tomto formuláři se vyplňují právě výše zmíněné atributy.

Kategorii požadavku lze v případě nutnosti vytvářet i v tomto formuláři, jinak pro toto patří samostatná položka „Kategorie požadavků“ nacházející se na levé liště. Typ požadavku je dvojí, může být funkční, nebo nefunkční. Priorita požadavku pak obsahuje následující možnosti:

- Nezbytný.
    - Povinný požadavek, jež je základem systému.

- Možný.
    - Důležitý požadavek, který lze vynechat.
- Eventuální.
    - Požadavek, jež je nepovinný (podává se, je-li čas).
- Chceme mít.
    - Požadavek, který může být zahrnut do dalších verzí systému.




**Obrázek 42: Přidání nového požadavku**

Status požadavku říká, zda již byl požadavek schválen. Lze mu nastavit hodnotu s následujících možností:

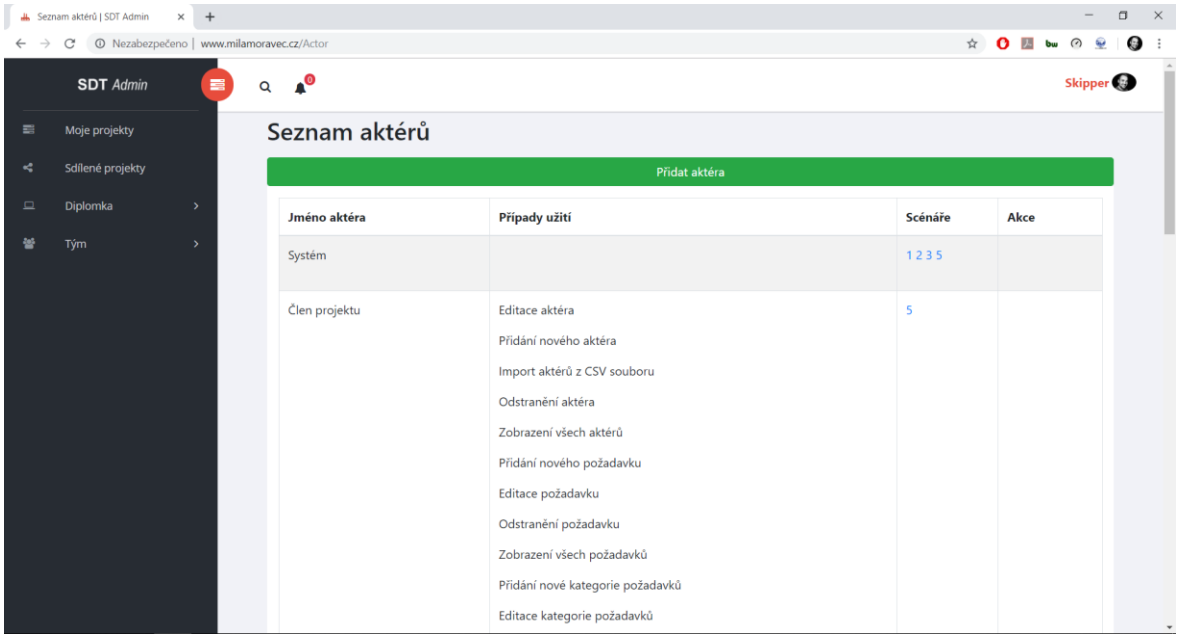
- Navrženo.
    - O požadavku se stále diskutuje, a ještě nebyl odsouhlasen.
- Přijato.
    - Požadavek byl schválen a může být implementován.
- Odmítnuto.
    - Požadavek byl odmítnut a nebude implementován.
- Začleněno.
    - Požadavek byl implementován v určité verzi.

Posledními položkami jsou zdroj, který udává, kdo požadavek navrhl a samotný text požadavku.

Po kliknutí na tlačítko „Vytvořit požadavek“ je tento vytvořen a zařazen do seznamu požadavků, ve kterém je možné ho upravovat či odstranit. Odstranění požadavku je podmíněno tím, zda není požadavek realizován nějakým případem užití.

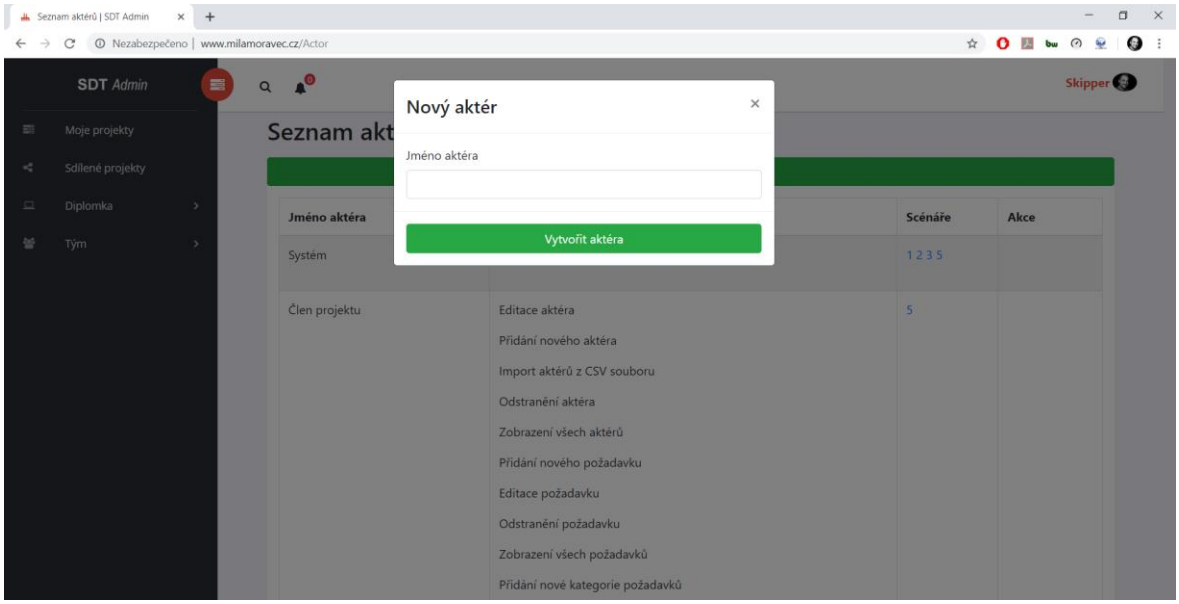
### 7.4.3 Správa aktérů

Další položkou v otevřeném projektu jsou aktéři. V seznamu aktérů (Obrázek 43) se ke každému aktérovi uvádí, v jakých případech užití a scénářích se daný aktér vyskytuje. Scénáře jsou zde znázorněny svým ID v podobě hypertextového odkazu, který umožňuje si daný scénář okamžitě prohlédnout.



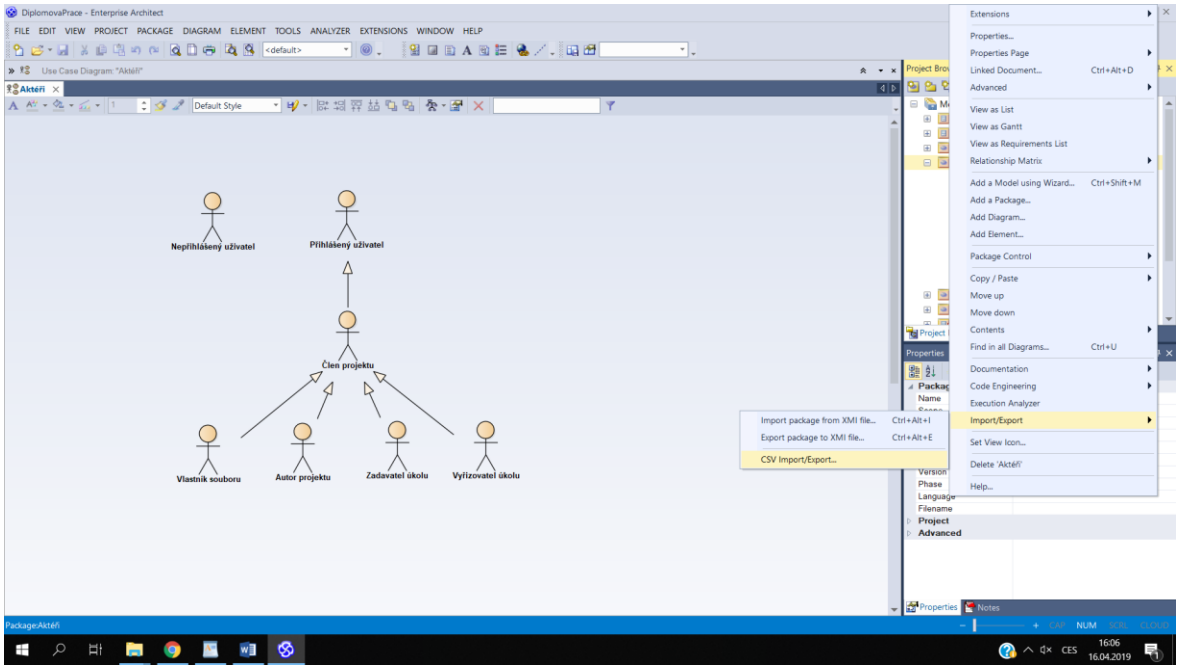

**Obrázek 43: Seznam aktérů**

Každá položka v seznamu aktérů lze samozřejmě upravit a odstranit. Odstranění je opět podmíněno nevyužitelností aktéra v žádném případu užití či scénáři. K vytvoření aktéra je zapotřebí zadání jeho jména. Z tohoto důvodu nemá tvorba aktéra žádný speciální formulář, ale pouze dialogové okno (Obrázek 44) sloužící k této akci.



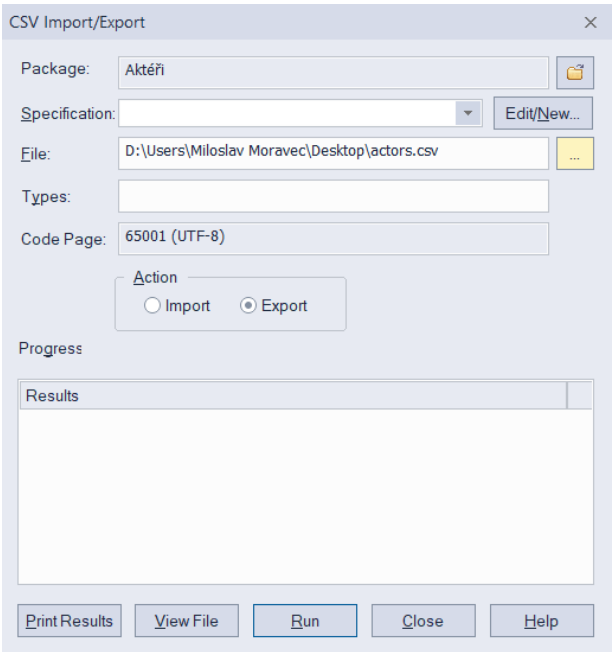

**Obrázek 44: Vytvoření aktéra**

Aktéry lze nahrát i z externího CSV souboru. Počítá se s využitím programu Enterprise Architect, který umožňuje UML model s aktéry uložit do CSV souboru. Dle obrázku 45 se po kliknutí pravým tlačítkem myši na balíček udržující model aktérů vybere možnost „Import/Export“ a posléze volba „CSV Import/Export“.



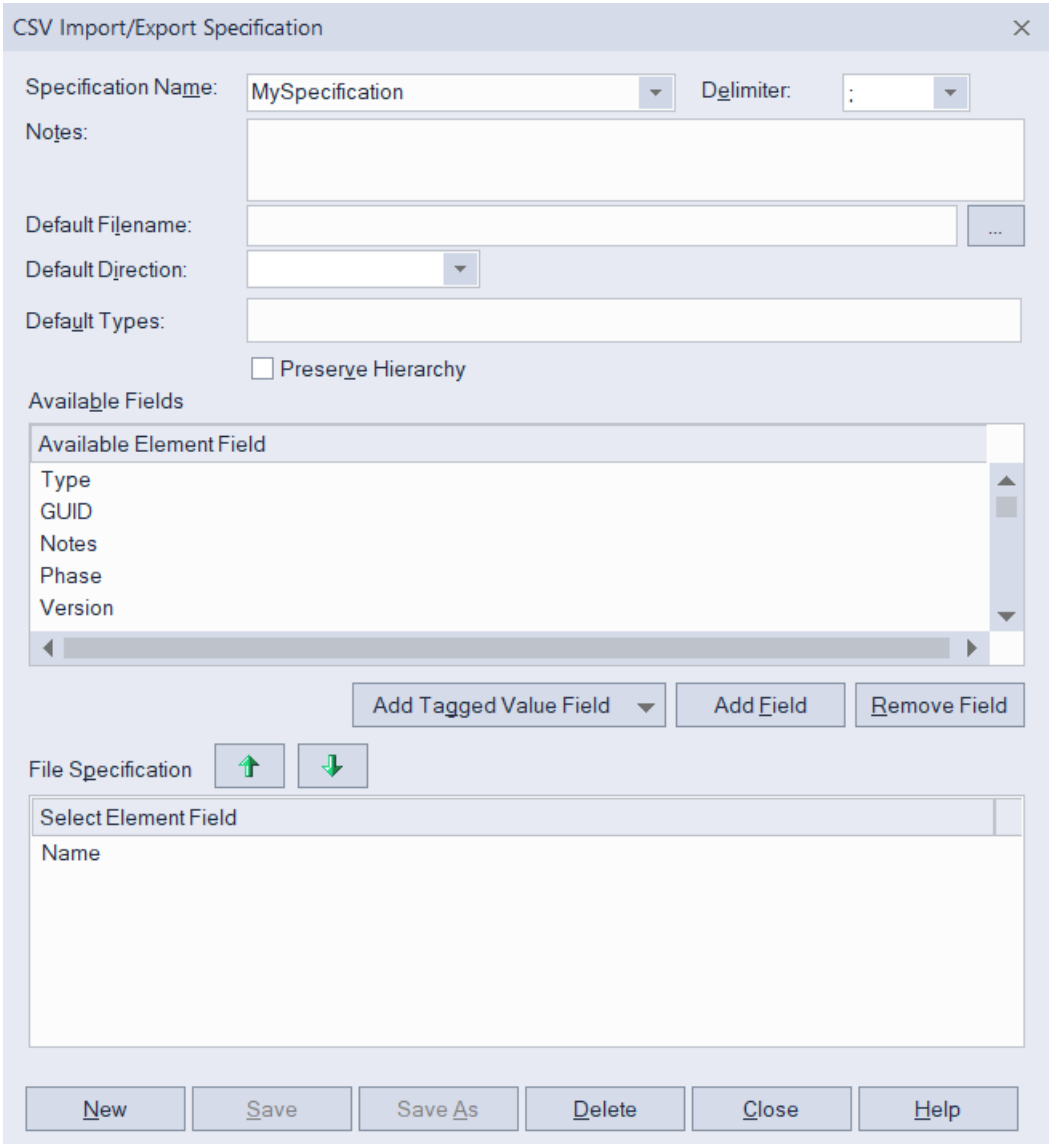

**Obrázek 45: Export aktérů z Enterprise Architect I**

Objeví se následující dialogové okno (Obrázek 46), ve kterém se specifikuje v kolonce „File“ cílová cesta k souboru, který se má vytvořit. Dále se v sekci „Action“ zvolí možnost Export.




**Obrázek 46: Export aktérů z Enterprise Architect II**

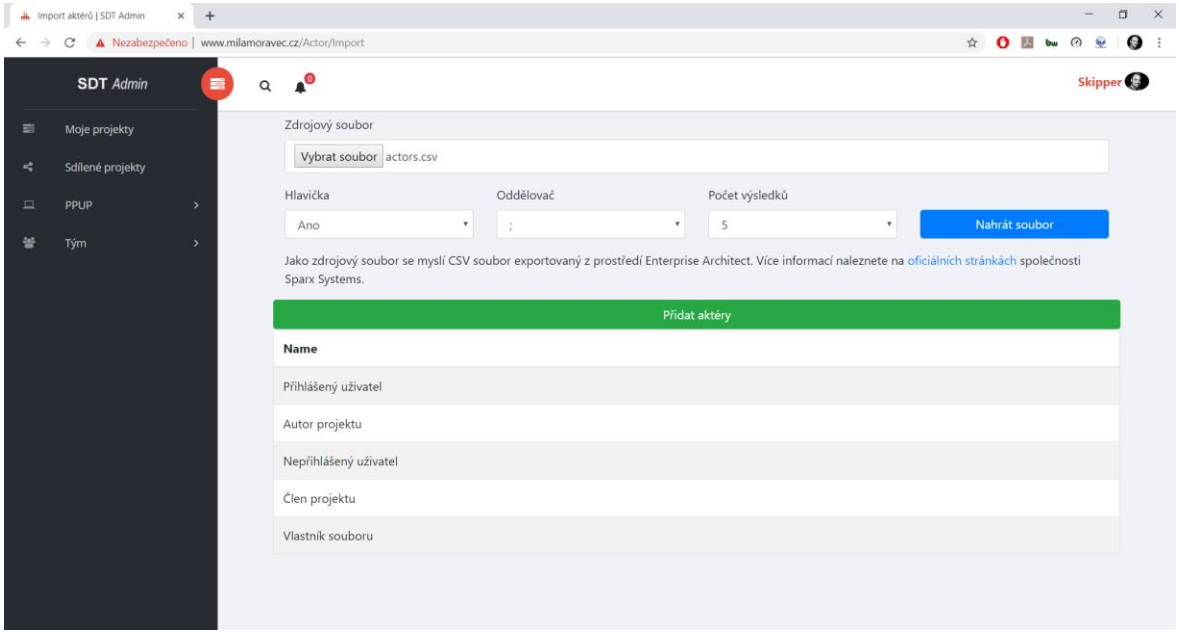
Posledním, ale zároveň nejdůležitějším krokem je definování atributů, které mají být v CSV souboru zobrazeny. Toto se uvádí v sekci „Specification“, kde se klikne na tlačítko „Edit/New“. Zobrazí se dialogové okno (Obrázek 47), ve kterém se v sekci „Specification Name“ zadá jméno specifikace a v sekci „Delimiter“ se vybere oddělovač. V sekci „Available Fields“ se nachází dostupné atributy ke každému aktérovi. V tomto případě stačí vybrat pouze atribut Name a potvrdit výběr tlačítkem „Add Field“. Následuje tlačítko „Save“ pro uložení specifikace a „Close“ pro zavření dialogu. V základním dialogu (Obrázek 46) se tato nově vybraná specifikace vybere ze seznamu specifikací a provede se export stisknutím tlačítka „Run“.




**Obrázek 47: Export aktérů z Enterprise Architect III**

Takto vytvořený CSV se posléze jednoduše nahraje do aplikace pomocí formuláře „Import aktérů“ (Obrázek 48). V této části se vybere daný CSV soubor, zvolí se, zda má CSV soubor hlavičku, nebo nemá (v tomto případě ano), vybere se oddělovač a v případě obsáhlého CSV souboru se dá nastavit, kolik výsledků se má zobrazit. V tabulce, která se následně po

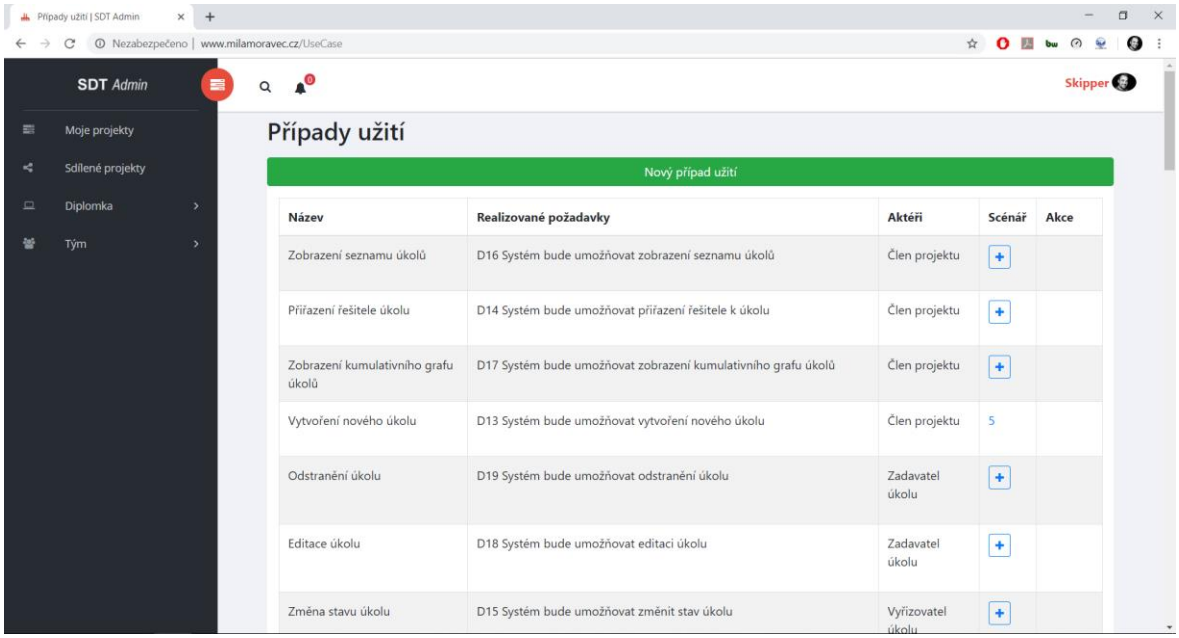
kliknutí na tlačítko „Nahrát soubor“ zobrazí je pouze náhled sloužící ke kontrole, zda jsou data v pořádku. Až po stisknutí tlačítka „Přidat aktéry“ se skutečně tyto data vloží do systému.




**Obrázek 48: Import aktérů**

### 7.4.4 Správa případů užití

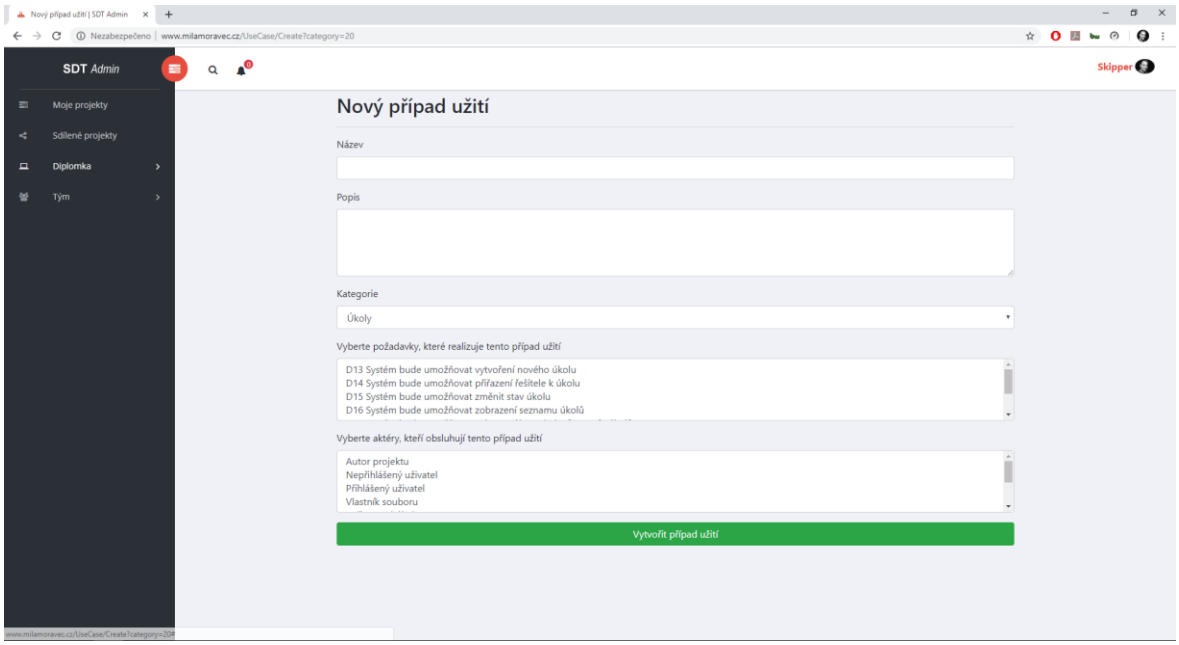
Po tvorbě požadavků a aktérů následuje tvorba případů užití. Dle následujícího obrázku (Obrázek 49) se ke každému případu užití udržuje jeho název, seznam požadavků, které jsou tímto případem užití realizovány, seznam aktérů, kteří daný případ užití obsluhují a scénář.




**Obrázek 49: Seznam případů užití**

U případu užití, který zatím nemá definovaný scénář se zobrazuje tlačítko „+“ umožňující přejít k formuláři pro vytvoření scénáře k tomuto případu užití. V opačném případě se vypisuje ID scénáře ve formě hypertextového odkazu, jak tomu bylo i u seznamu aktérů.

Vytvoření nového případu užití se provádí, jak je už zvykem, přes tlačítko „Nový případ užití“, které zobrazí odpovídající formulář (Obrázek 50).



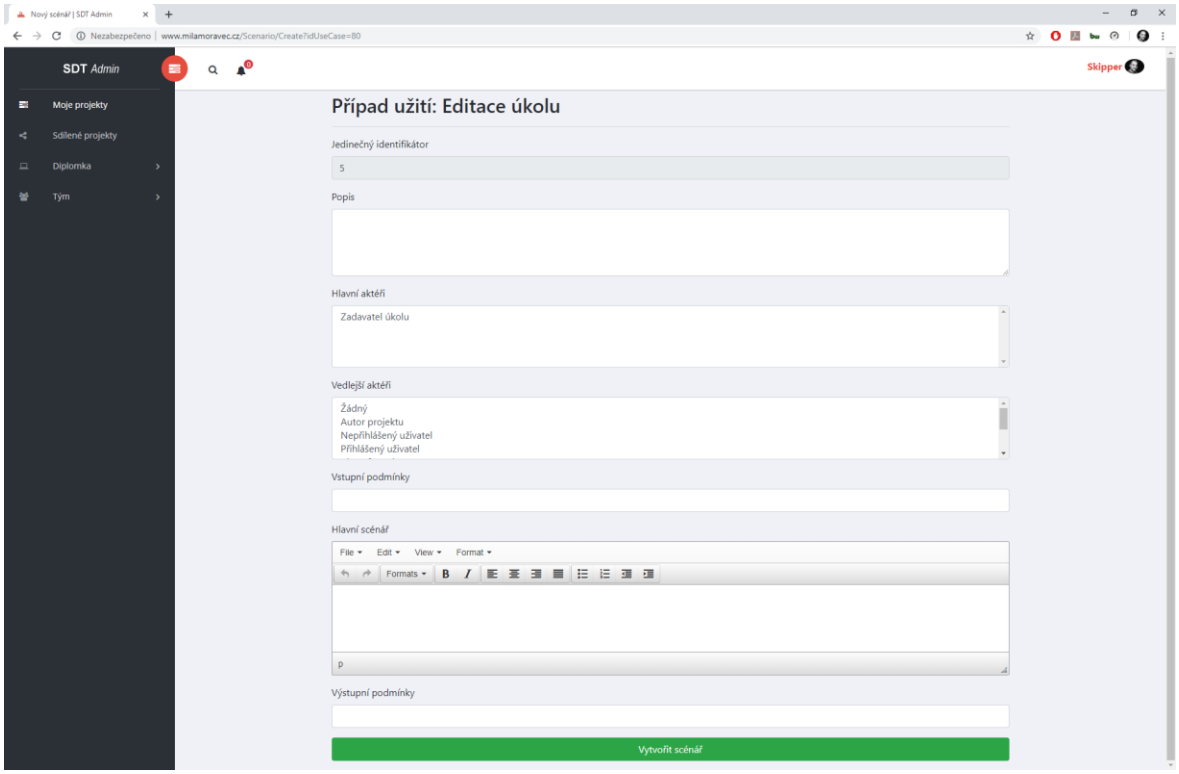

**Obrázek 50: Nový případ užití**

V tomto formuláři se vyplňuje název případu užití, nepovinně popis případu užití, a hlavně se vybírá z nadefinovaných funkčních požadavků, které tento případ užití realizují. Jelikož může být v systému několik funkčních požadavků a bylo by obtížné v nich hledat ty, které jsou momentálně potřeba, existuje možnost vyfiltrování požadavků jen z určité kategorie požadavků.

### 7.4.5 Správa scénářů

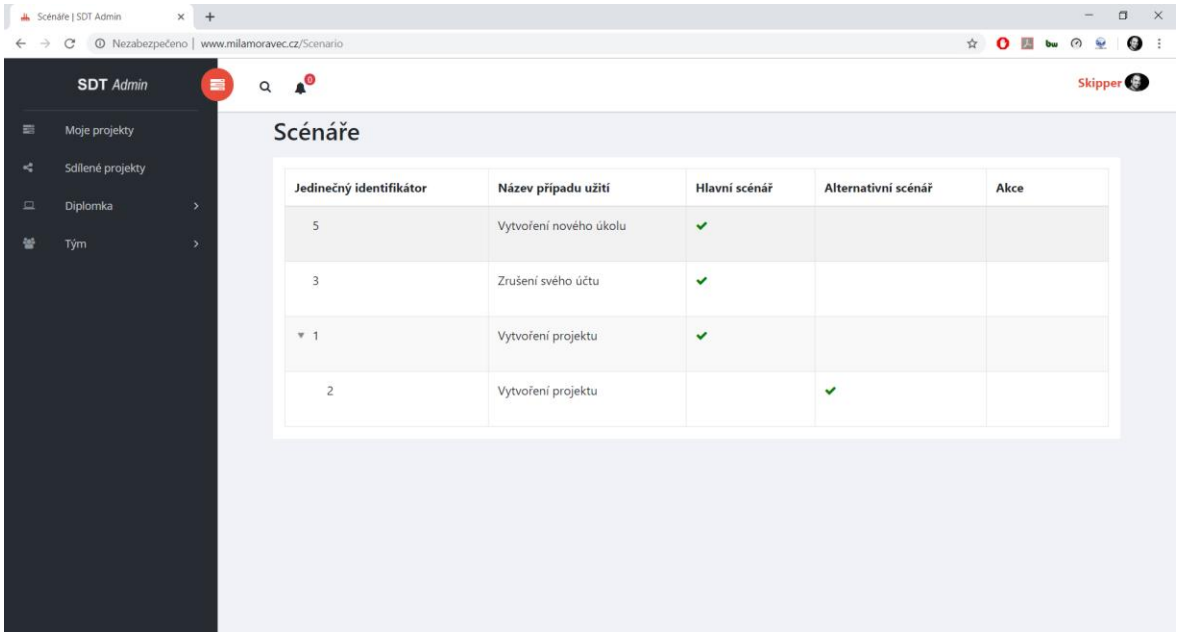
Jak bylo naznačeno výše, scénáře se vytváří v rámci případů užití. Ke každému případu užití existuje právě jeden hlavní scénář, který může mít více alternativních scénářů. Jak je zvykem, alternativní scénář již žádný alternativní scénář mít nemůže. K vytvoření nového scénáře se používá tlačítka „+“, které vyvolá následující formulář (Obrázek 51) vycházející ze standardního předpisu pro scénáře.

Jedinečný identifikátor scénáře se generuje automaticky dle počtu vytvořených scénářů v projektu. Následuje popis vysvětlující chování scénáře. Další položkou je výběr hlavních a vedlejších aktérů. Hlavní aktéři jsou přebráni z aktérů přiřazených k samotnému případu užití, vedlejší aktéři jsou všichni ostatní aktéři z projektu. Vstupní podmínka je další položkou formuláře, která definuje, co se musí stát, aby mohl začít hlavní scénář. Výstupní podmínka pak definuje stav, který nastane po úspěšném dokončení běhu hlavního scénáře.




**Obrázek 51: Tvorba scénáře**

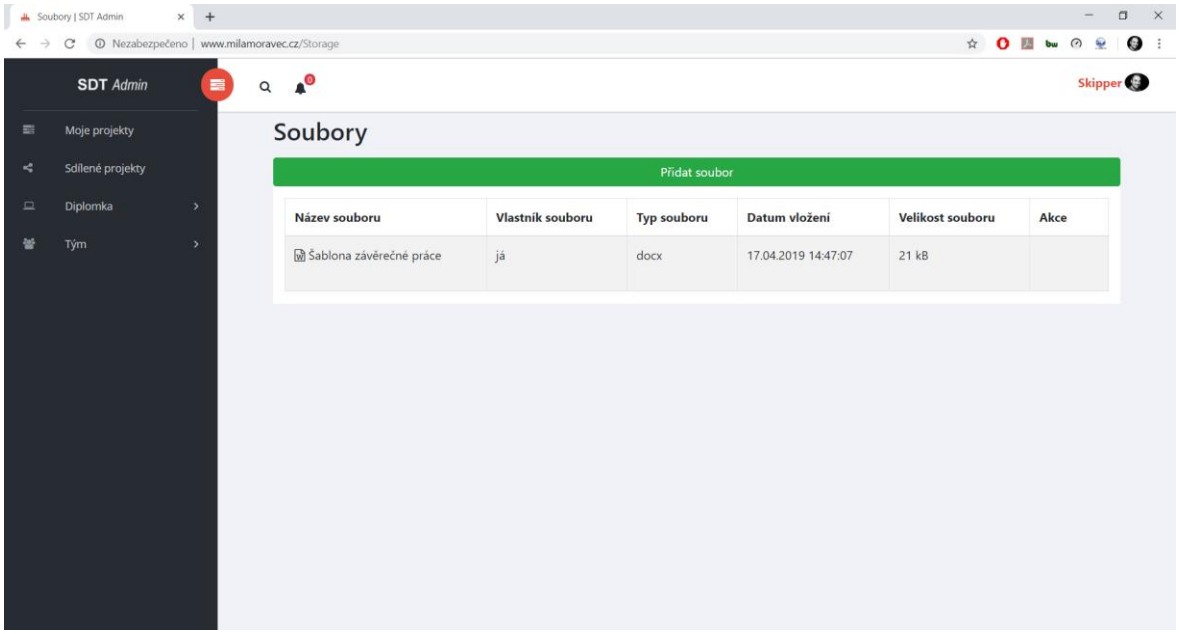
Takto vytvořený scénář se zařadí do seznamu scénářů. Po zobrazení scénáře ke čtení je možné obdobným způsobem vytvořit alternativní scénáře, které se, jak je vidět na následujícím obrázku (Obrázku 52), projeví v seznamu hlavních scénářů do stromové struktury.




**Obrázek 52: Seznam scénářů**

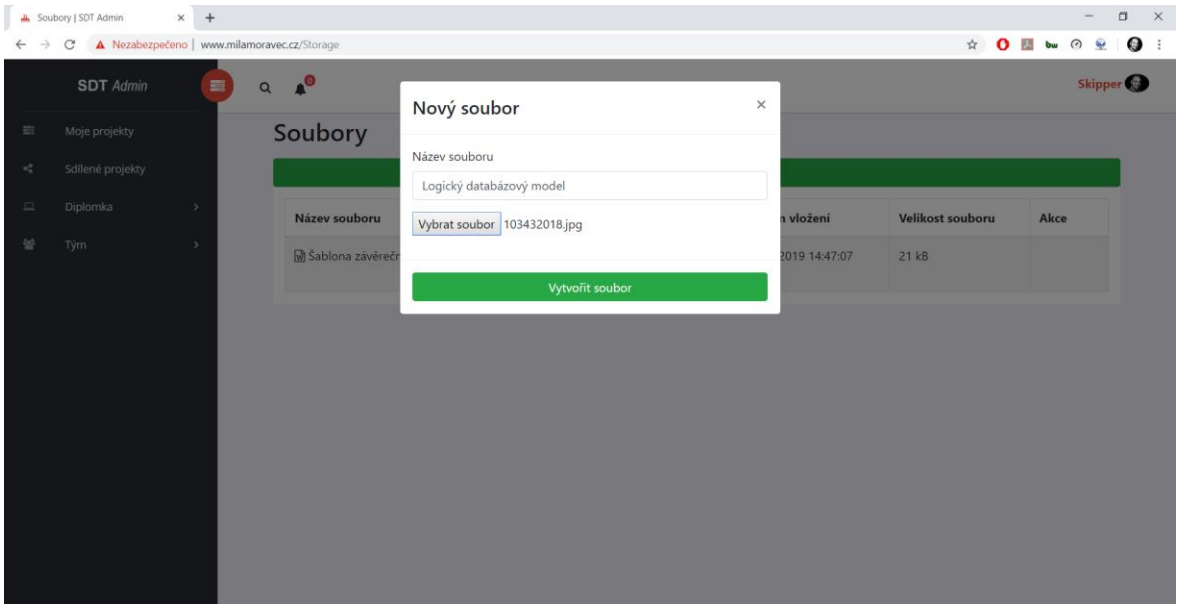
### 7.4.6 Správa souborů

Poslední položkou v rámci projektu jsou soubory. Seznam souborů (Obrázek 53) uchovává u každého souboru jeho název, jméno člena projektu, který soubor nahrál, typ souboru, datum, kdy byl soubor nahrán a jeho velikost. Právo na odstranění tohoto souboru má pouze jeho vlastník. Ostatní členové projektu mohou daný soubor libovolně stahovat.




**Obrázek 53: Seznam souborů**

K vytvoření nového souboru je zapotřebí zadat jeho název a cestu k jeho aktuálnímu umístění. Opět se zde tedy využívá pouze jednoduchého dialogu (Obrázek 54) namísto samostatného formuláře pro vytvoření souboru.




**Obrázek 54: Dialog pro vytvoření nového souboru**

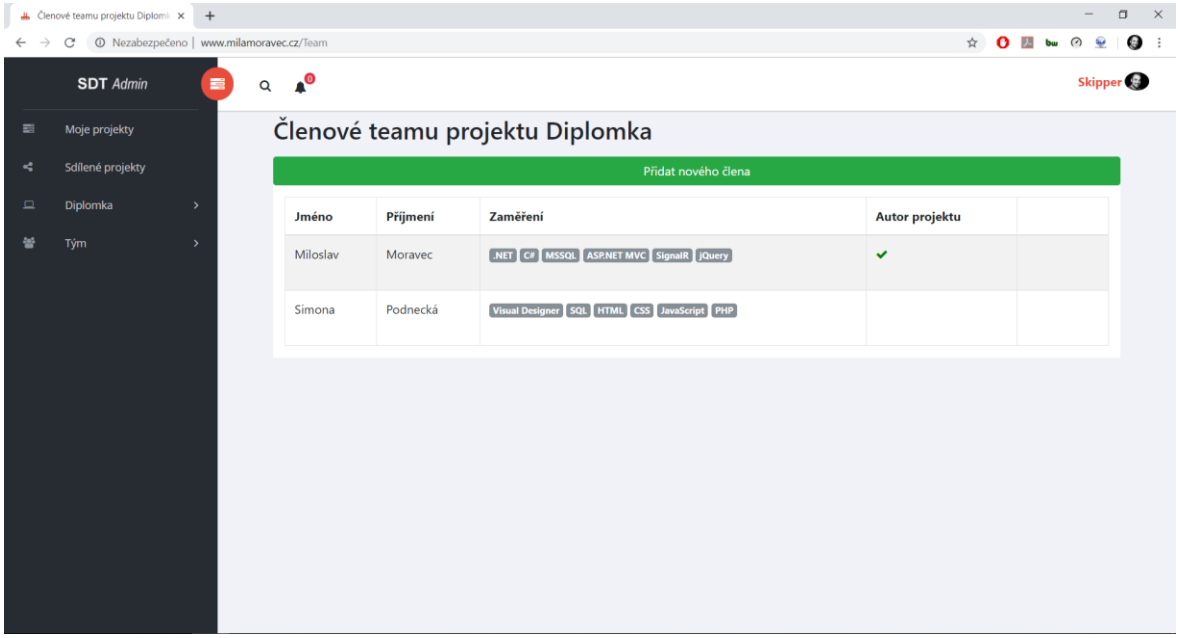
## 7.5 Správa týmu

Všechny úkony, které byly představeny v předcházející podkapitole se tvoří v rámci týmu. Jak bylo zmíněno výše, tým se skládá z kontaktů autora projektu či jednotlivých členů projektu. Po otevření nějakého projektu se zpřístupní záložka „Tým“, která obsahuje tři části:

- Členové týmu;
- Úkoly;
- Sledování úkolů;

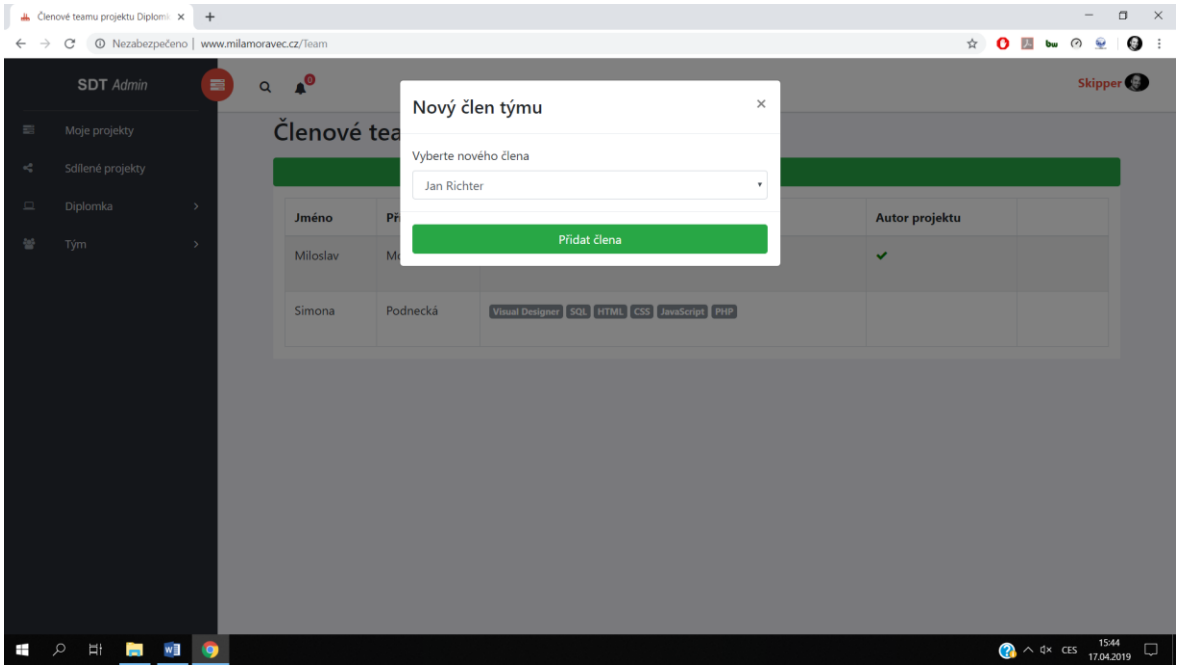
### 7.5.1 Členové týmu

V seznamu členů týmu (Obrázek 55) lze u každého člena vidět jeho jméno, příjmení a seznam technologií, na které se zaměřuje. Dále se odlišuje, kdo z členů projekt založil a je tedy jeho autorem.




**Obrázek 55: Členové týmu**

Přidání nového člena projektu probíhá prostřednictvím dialogu (Obrázek 56), ve kterém se ze seznamu kontaktů vybírá, kdo se má stát novým členem projektu. Seznam kontaktů je seřazen dle dovedností uživatele korespondujících s technologiemi projektu. Nově přidaný uživatel dostává o této akci zprávu prostřednictvím upozornění a projekt, do kterého byl přidán, se mu objeví v záložce „Sdílené projekty“. Od té doby má plné právo na modifikování tohoto projektu a přidávání dalších lidí do projektu ze svých kontaktů. Odebírat členy z projektu může pouze autor projektu.



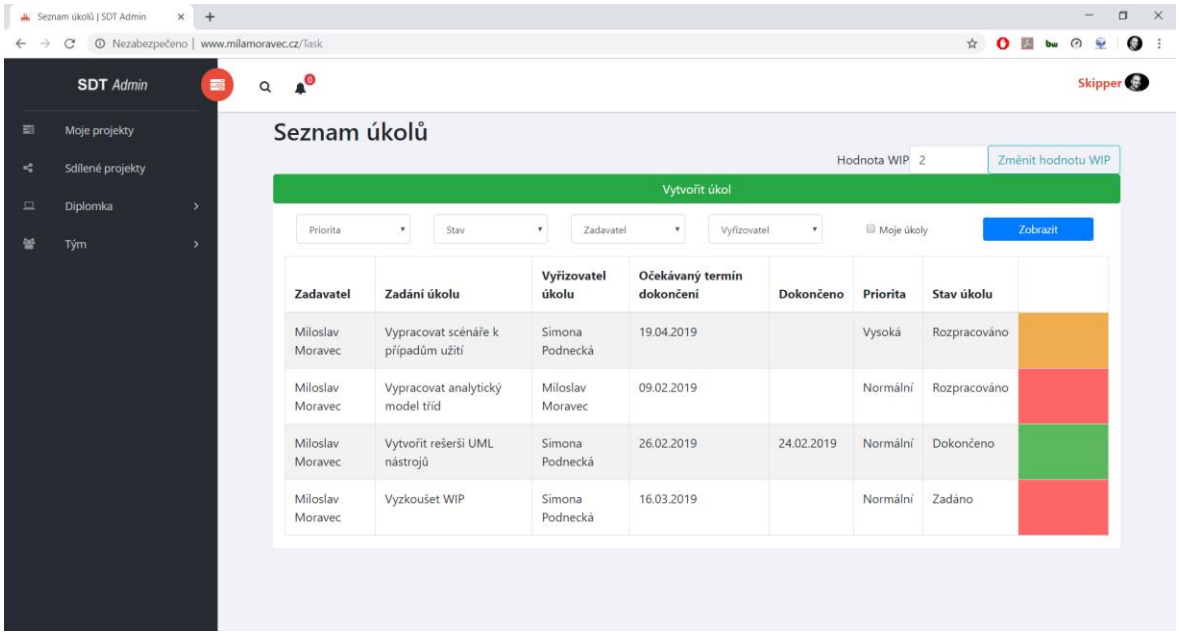

**Obrázek 56: Nový člen projektu**

### 7.5.2 Úkoly

Aby byla práce v týmu efektivní, umožňuje tato aplikace zadávání úkolů napříč týmem. V seznamu úkolů (Obrázek 57) se nachází všechny zadané úkoly udržující následující informace:

- zadavatel úkolu,
  - Jméno uživatele, který úkol vytvořil a má tak právo tento úkol odstranit či jinak modifikovat.
- zadání úkolu,
- vyřizovatel úkolu,
  - Jméno uživatele, který má úkol splnit. Tento uživatel může být totožný se zadavatelem úkolu.
- očekávaný termín dokončení,
- datum kdy byl úkol skutečně dokončen,
  - Datum, kdy byl úkolu změněn stav na dokončený.
- priorita úkolu,
  - Může být nízká, normální či vysoká.
- stav úkolu.
  - Stavy úkolu jsou: zadaný, rozpracovaný a dokončený.

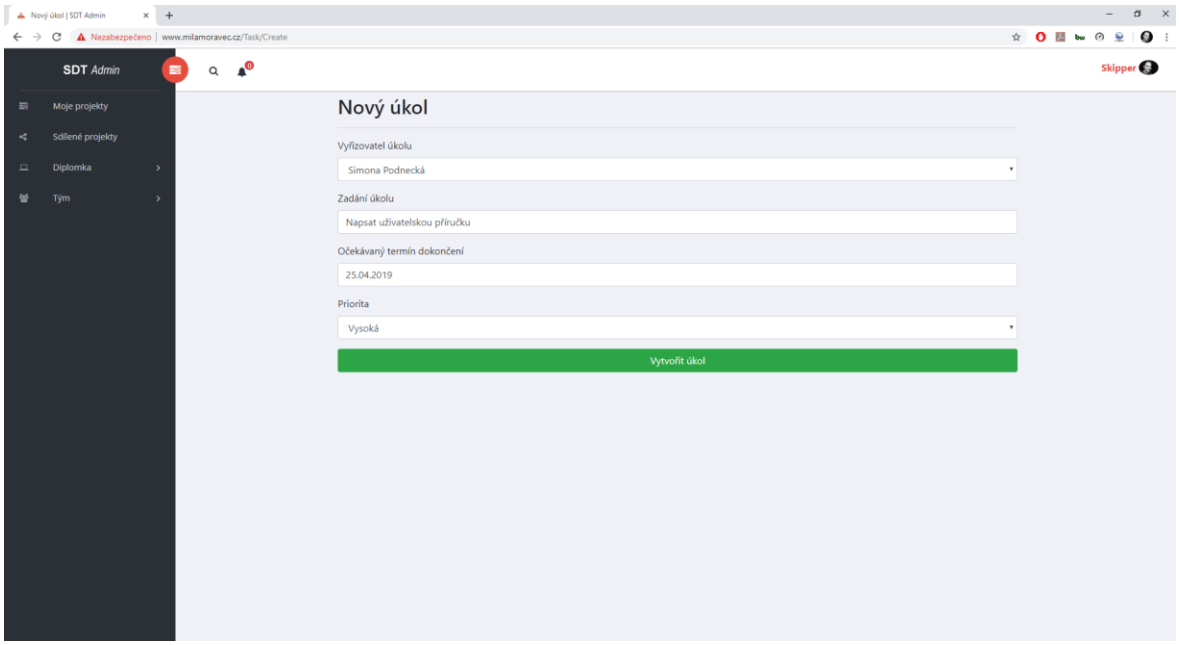
Dále je pro přehlednost každý úkol opatřen barevným označením. Modrá barva znamená, že je úkol zadaný. Oranžová barva znamená, že je úkol rozpracovaný. Zelená barva znamená, že byl úkol dokončen v zadaném limitu. Červená barva znamená, že úkol nebyl (nebo stále není) dokončený v zadaném limitu.




**Obrázek 57: Seznam úkolů**

Výpis seznamu úkolu lze filtrovat dle atributů úkolu či lze vypsat pouze úkoly, které má přihlášený uživatel přiřazené k vyřízení. V pravém horním rohu okna se nachází textové pole sloužící k modifikaci hodnoty WIP, která udává kolik úkolů může být v jeden čas ve stavu „Rozpracováno“. Tuto hodnotu může modifikovat pouze autor projektu. V případě, že chce uživatel změnit stav nějakého úkolu ze „Zadáno“ na „Rozpracováno“ a překročil by tak povolenou hodnotu WIP, systém mu tuto změnu nedovolí.

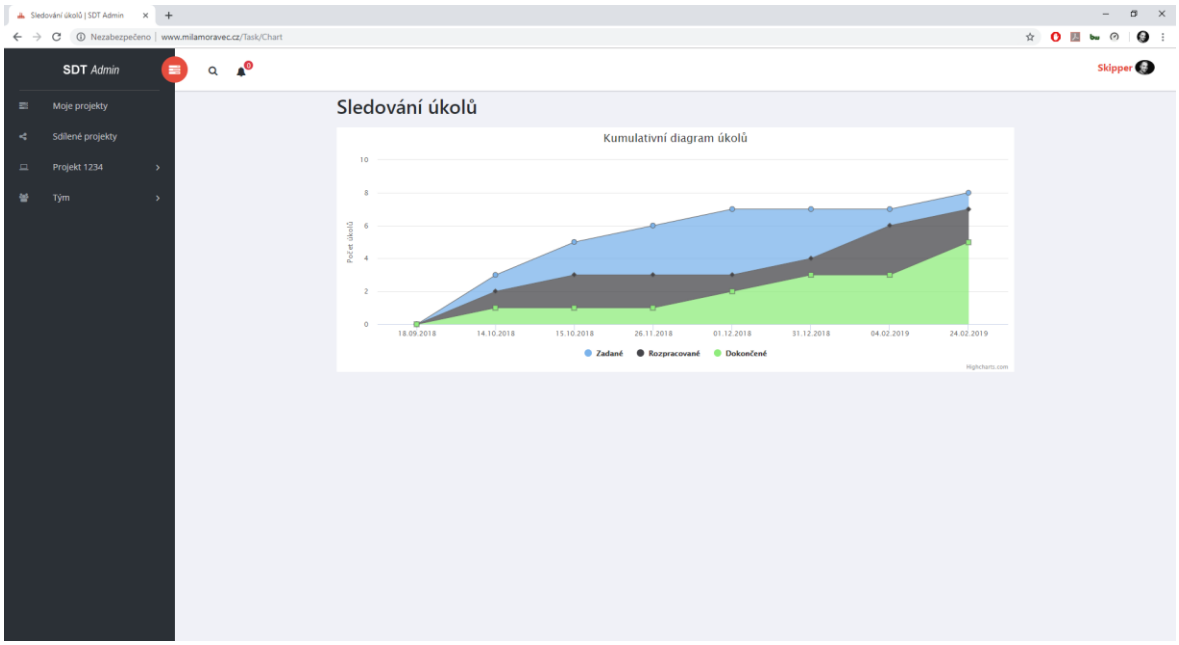
Vytvoření nového úkolu se provádí pomocí tlačítka „Vytvořit úkol“, které zobrazí formulář (Obrázek 58), na kterém se nachází všechny důležité atributy zmíněné výše.




**Obrázek 58: Nový úkol**

### 7.5.3 Sledování úkolů

V poslední záložce týkající se sledování úkolů jsou zaznamenány všechny provedené úkoly v rámci projektu. Tyto úkony jsou reprezentovány kumulativním grafem (Obrázek 59), který definuje pro každé datum aktuální stavy úkolů v systémů. Kolik úkolů bylo zadaných, kolik rozpracovaných a kolik dokončených v daný časový okamžik. Díky tomuto lze sledovat poměry počtů úkolů v jednotlivých stavech a upravovat tak hodnotu WIP pro zamezení, či naopak povolení více rozpracovaných úkolů.




**Obrázek 59: Kumulativní diagram úkolů**

# ZÁVĚR

V rámci této diplomové práce bylo využito znalostí ze studia magisterského oboru, zejména pak z předmětů zabývajících se projektováním softwarových systémů. Práce s pokročilými webovými technologiemi (ASP.NET, Entity Framework, …) byla nastudována samostatně.

Výsledná aplikace by se dala uplatnit v menších týmech jako pomůcka pro vývojáře, kteří potřebují snadnou a rychlou dostupnost analýzy jimi vyvíjeného projektu. Dokumentace, která lze z aplikace získat, může sloužit jako skvělé vodítko při implementaci nejasných zákoutí systému. Vývojář si může například vytisknout potřebné scénáře, položit si je vedle sebe na stůl a podle nich psát požadovaný kód.

Aplikace by se dala dále rozšířit o uživatelskou roli administrátora, který by měl jasný přehled nad uživateli v systému, včetně jejich chování a projektů. V této chvíli si aplikace žije tzv. „vlastním životem“ bez jasného řízení třetí osobou. Dále by se dalo upravit přihlašování do systému, které zatím obsahuje pouze uživatelské jméno a heslo. Nové přihlašování by mohlo počítat se zadáním emailové adresy sloužící pro ověření nově registrovaného uživatele, obnovu zapomenutého hesla či zasílání notifikací ze systému. Aktuálně se uživatel dozví o zaslané zprávě pouze po přihlášení do systému, což může být někdy pozdě.

Úskalí, které provázelo tuto diplomovou práci, představovalo její testování. V práci se sice píše, že testování je jednou z nejdůležitějších fází vývoje softwaru a že by mělo obsahovat například jednotkové či integrační testy, ale vzhledem k objemné velikosti výsledné aplikace a neznalosti přesného postupu testování webových aplikací se na tyto nedostalo. Jednotkové a integrační testy alespoň trochu nahradily testy uživatelského rozhraní, které i přes možnou nedůvěru odhalily několik implementačních nedostatků a splnily tak účel testování.

Vývoj webových aplikací je obor, který mě vždy velice zajímal. Ačkoliv jsem začínal s programovacím jazykem PHP a v rámci magisterského oboru jsem se dozvěděl i něco o jazyku JSP, tak framework ASP.NET MVC je něco, co mě opravdu uchvátilo a jsem rád, že jsem si v něm mohl vyzkoušet vyvinout nějaký větší projekt.

# POUŽITÁ LITERATURA


[1] Defining and Using Project Management Process Groups. WYSOCKI, Robert K. *Effective project management: traditional, agile, extreme*. Fifth edition. Indianapolis, IN: Wiley Publishing, [2009], s. 6. ISBN 978-0-470-42367-7.

[2] NDEGWA, Amos. What is a Web Application? *MaxCDN* [online]. 31 May 2016 [cit. 2019-02-04]. Dostupné z: https://www.maxcdn.com/one/visual-glossary/web-application/

[3] CHRISTENSSON, Per. DBMS Definition. *TechTerms* [online]. Sharpened Productions, 2006 [cit. 2019-02-04]. Dostupné z: https://techterms.com/definition/dbms

[4] CHRISTENSSON, Per. RDBMS Definition. *TechTerms* [online]. Sharpened Productions, 2017 [cit. 2019-02-04]. Dostupné z: https://techterms.com/definition/rdbms

[5] CREATE TRIGGER (Transact-SQL). *Microsoft Docs* [online]. 2017 [cit. 2019-02-04]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql?view=sql-server-2017

[6] SWERSKY, Dave. The SDLC: 7 phases, popular models, benefits & more. *Raygun* [online]. 31 May 2018 [cit. 2019-01-30]. Dostupné z: https://raygun.com/blog/software-development-life-cycle/

[7] Waterfall Software Development Model. *Oxagile* [online]. 5 February 2014 [cit. 2019-01-31]. Dostupné z: https://www.oxagile.com/company/blog/the-waterfall-model/

[8] POWELL-MORSE, Andrew. Iterative Model: What Is It And When Should You Use It? *Airbrake* [online]. 15 December 2016 [cit. 2019-02-01]. Dostupné z: https://airbrake.io/blog/sdlc/iterative-model

[9] PAL, Sayan Kumar. Software Engineering: Spiral Model. *Geeks for Geeks: A computer science portal for geeks* [online]. [cit. 2019-01-31]. Dostupné z: https://www.geeksforgeeks.org/software-engineering-spiral-model/

[10] POWELL-MORSE, Andrew. Spiral Model: Software Development For Critical Projects. In: *Airbrake* [online]. 29 September 2016 [cit. 2019-01-31]. Dostupné z: https://airbrake.io/blog/sdlc/spiral-model

[11] SUTHERLAND, Jeff. Introduction to Scrum. *The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework* [online]. Version 1.1. Cambridge, 2012, s. 14 [cit. 2019-02-01]. Dostupné z: https://web.archive.org/web/20150814201800/http://jeffsutherland.com/ScrumPapers.pdf

[12] *What Is Kanban? An Introduction to Kanban Methodology* [online]. [cit. 2019-02-01]. Dostupné z: https://resources.collab.net/agile-101/what-is-kanban

[13] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

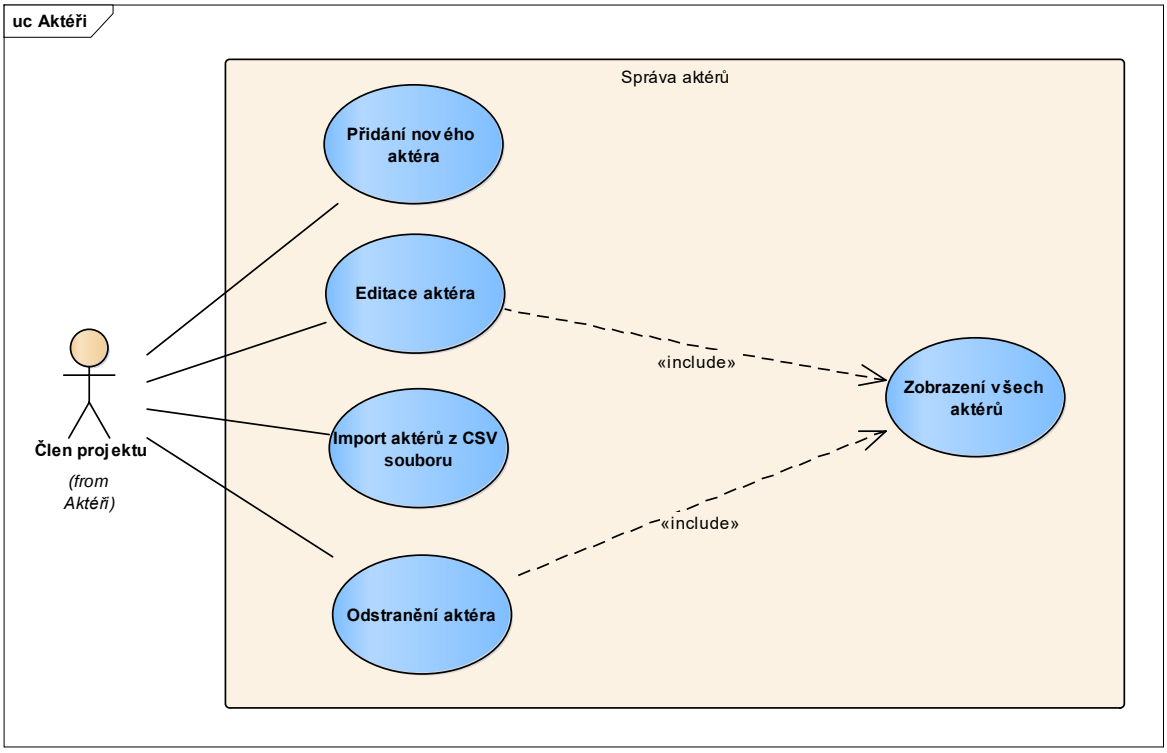
[14] SPARKS, Geoffrey. *Enterprise Architect: User Guide* [online]. 2014. [cit. 2019-01-12]. Dostupné z: https://www.sparxsystems.com.au/bin/EAUserGuide.pdf
[15] *StarUML documentation: Introduction* [online]. 2018 [cit. 2019-01-12]. Dostupné z: https://docs.staruml.io/
[16] *UML Designer: Overview* [online]. [cit. 2019-01-14]. Dostupné z: http://www.umldesigner.org/overview/
[17] Atlassian Documentation: What is a Project. *Jira Software Support* [online]. 2015 [cit. 2019-01-16]. Dostupné z: https://confluence.atlassian.com/jira064/what-is-a-project-720416135.html
[18] Atlassian Documentation: What is a Workflow. *Jira Software Support* [online]. 2015 [cit. 2019-01-16]. Dostupné z: https://confluence.atlassian.com/jira064/what-is-workflow-720416127.html
[19] Atlassian Documentation: Generating Reports. *Jira Software Support* [online]. 2018 [cit. 2019-01-16]. Dostupné z: https://confluence.atlassian.com/jira064/generating-reports-720416012.html
[20] O Trellu: Co je Trello? *Trello* [online]. 2018 [cit. 2019-01-16]. Dostupné z: https://trello.com/about
[21] Trello Tour. *Trello* [online]. 2019 [cit. 2019-01-16]. Dostupné z: https://trello.com/tour?truid=tr5354c1-61ae-6284-2f04-936937e66fca
[22] Getting Started: What is monday.com? *Monday* [online]. 2019 [cit. 2019-01-16]. Dostupné z: https://support.monday.com/hc/en-us/articles/115005310945-What-is-monday-com-
[23] Tutorials: How to get started with monday.com - First user. *Monday* [online]. 2019 [cit. 2019-01-16]. Dostupné z: https://support.monday.com/hc/en-us/articles/115005305649-How-to-get-started-with-monday-com-First-user
[24] Getting Started: What is the difference between Boards, Shareable Boards and Private Boards? *Monday* [online]. 2019 [cit. 2019-01-16]. Dostupné z: https://support.monday.com/hc/en-us/articles/115005311105-What-is-the-difference-between-Boards-Shareable-Boards-and-Private-Boards-
[25] RANKINS, Ray, Paul BERTUCCI, Chris GALLELLI a Alex T. SILVERSTEIN. *Microsoft SQL server 2014 unleashed*. Indianapolis, Indiana: Sams, 2015. ISBN 978-0672337291.
[26] GALLOWAY, Jon, Brad WILSON, K. Scott ALLEN a David MATSON. *Professional ASP.NET MVC 5*. Indianapolis, IN: Wrox, a Wiley brand, 2014. Wrox professional guides. ISBN 978-1-118-79475-3.
[27] Code-First vs Model-First vs Database-First: Pros and Cons. *Ryadel* [online]. 2018 [cit. 2019-01-20]. Dostupné z: https://www.ryadel.com/en/code-first-model-first-database-first-vs-comparison-orm-asp-net-core-entity-framework-ef-data/
[28] How To Install SQL Server Developer Edition: Install SQL Server. *SQLServerTutorial.net* [online]. [cit. 2019-04-05]. Dostupné z: http://www.sqlservertutorial.net/install-sql-server/

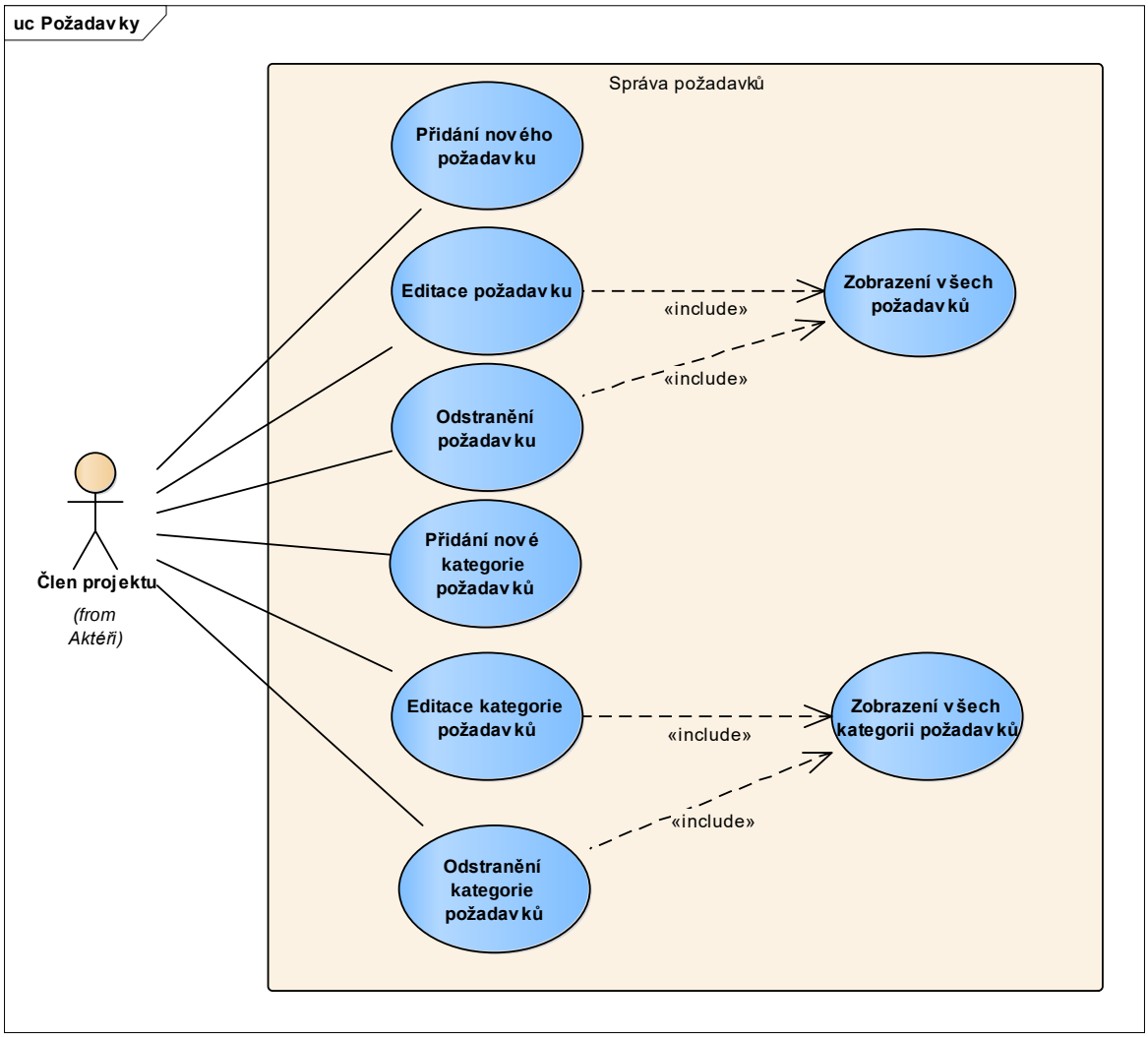
[29] What is Selenium? Introduction to Selenium Automation Testing. *Guru99* [online]. [cit. 2019-04-20]. Dostupné z: https://www.guru99.com/introduction-to-selenium.html

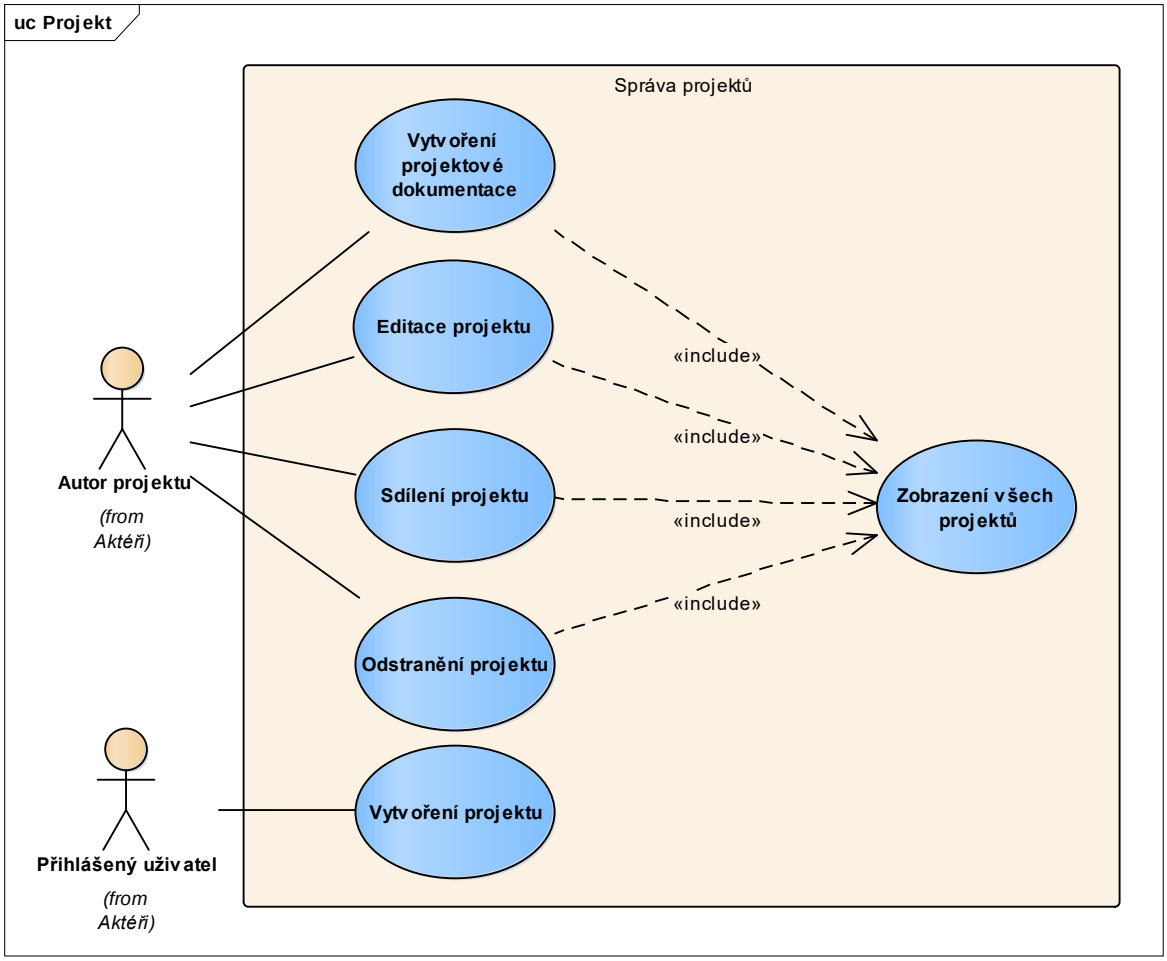
[30] What is Selenium WebDriver? Difference with RC. *Guru99* [online]. [cit. 2019-04-20]. Dostupné z: https://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html

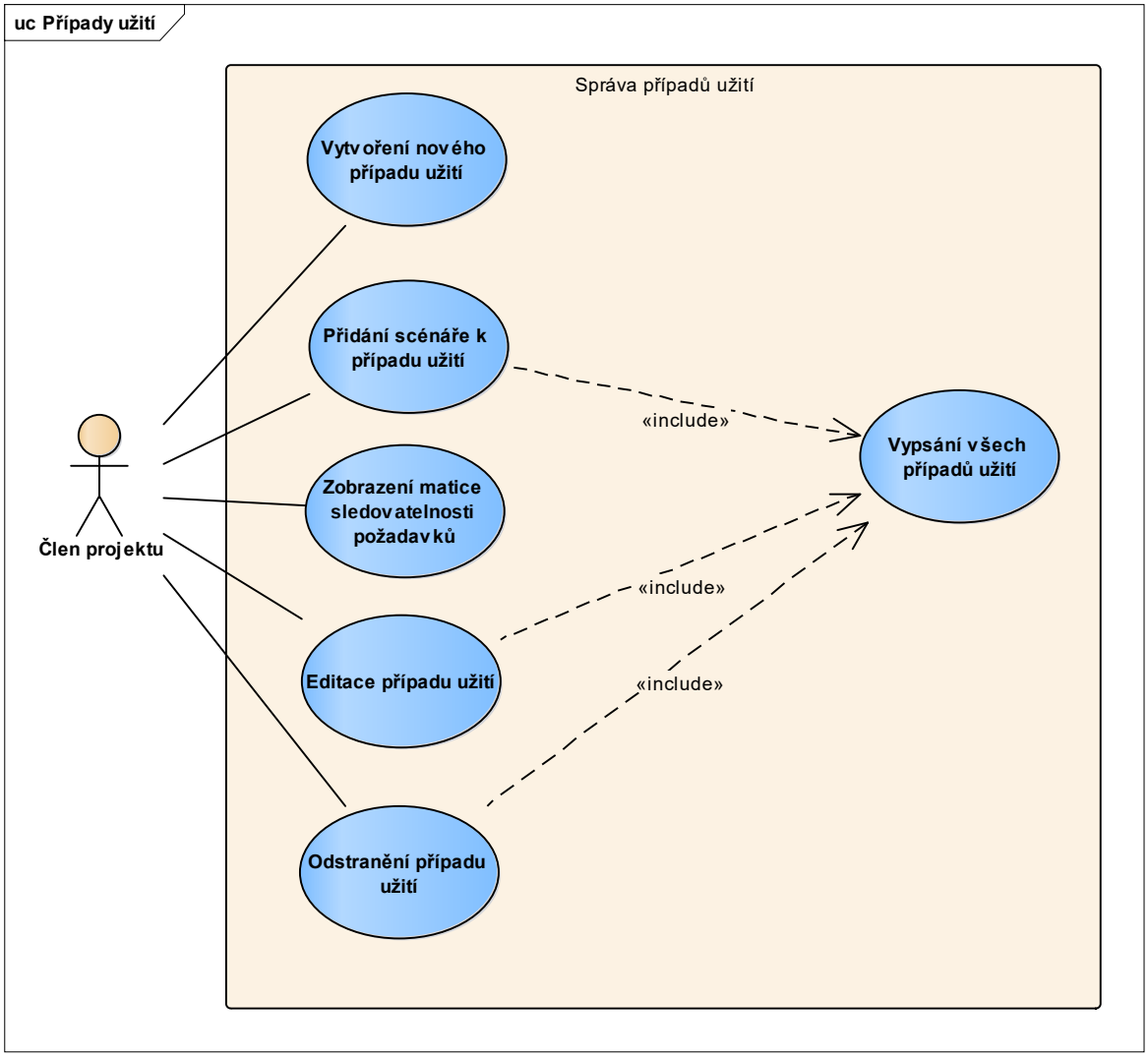
# PŘÍLOHY

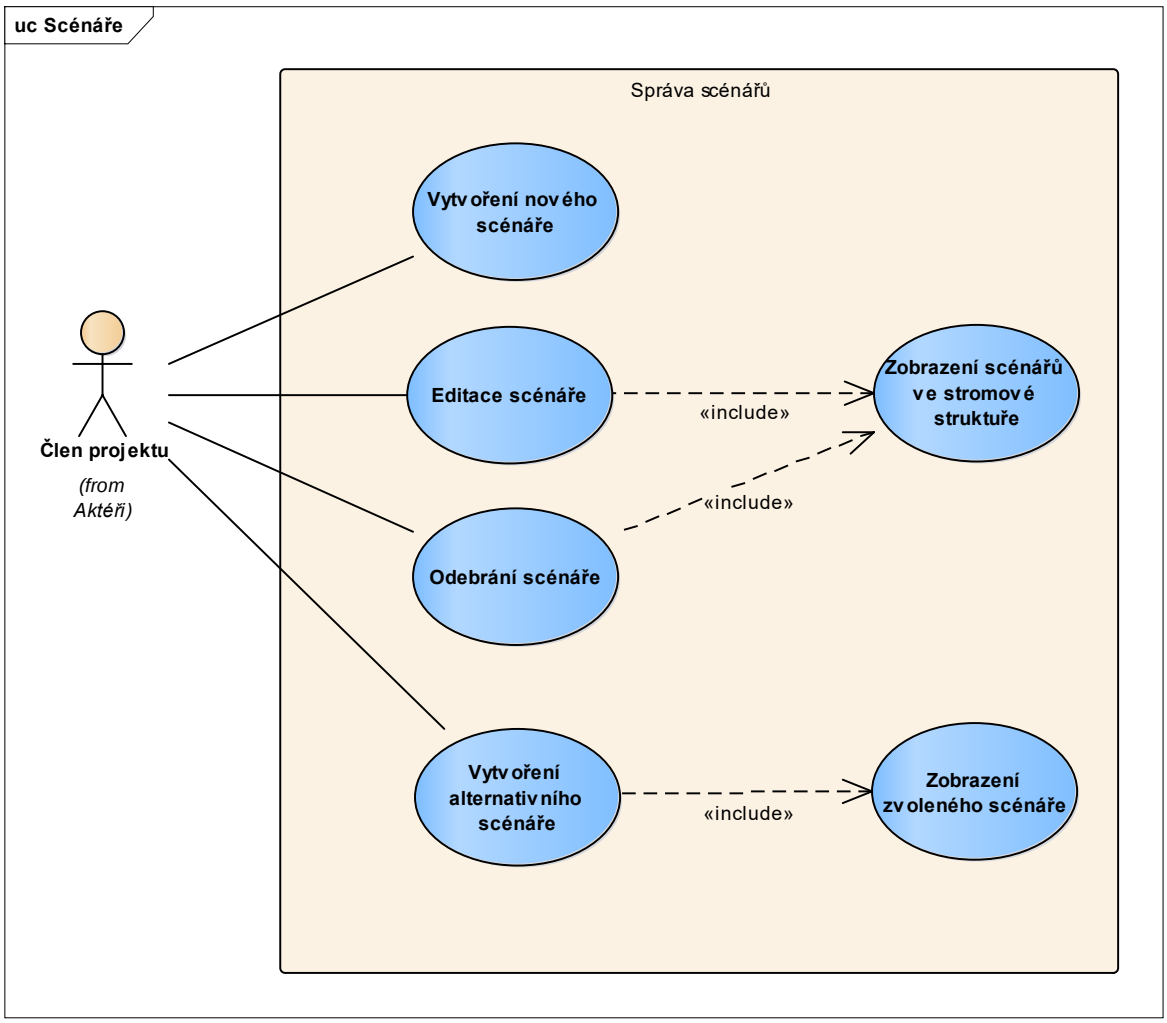
# Příloha A – Diagramy případů užití

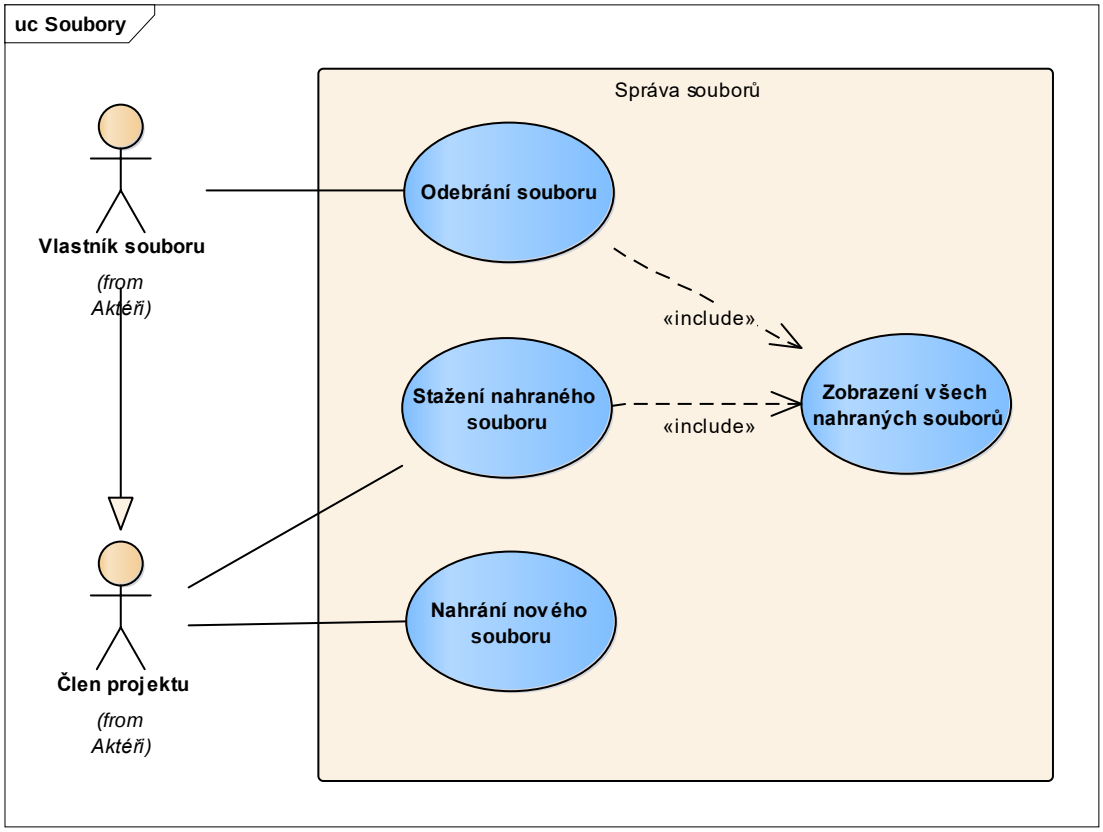


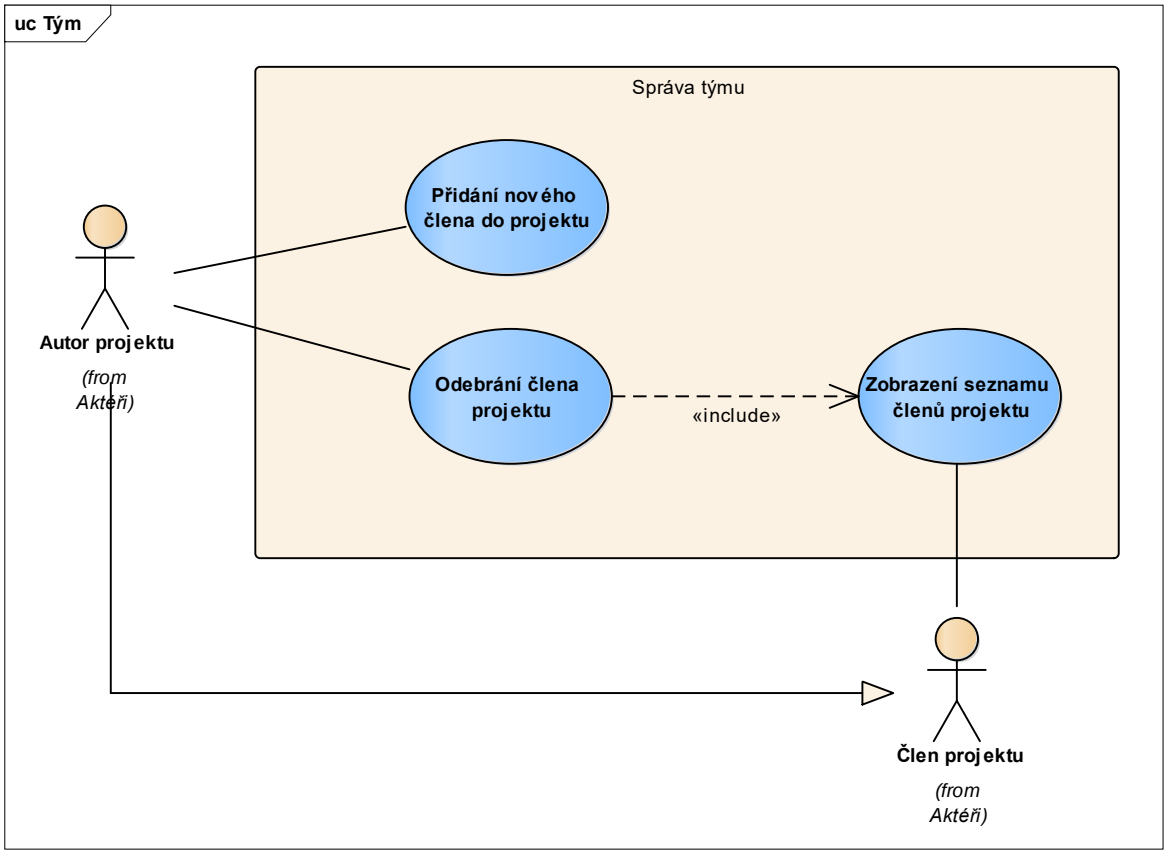

uc Požadavky
Správa požadavků
Přidání nového požadavku
Editace požadavku
Odstranění požadavku
Přidání nové kategorie požadavků
Editace kategorie požadavků
Odstranění kategorie požadavků
Zobrazení všech požadavků
Zobrazení všech kategorii požadavků
«include»
«include»
«include»
«include»
Člen projektu
(from Aktéři)

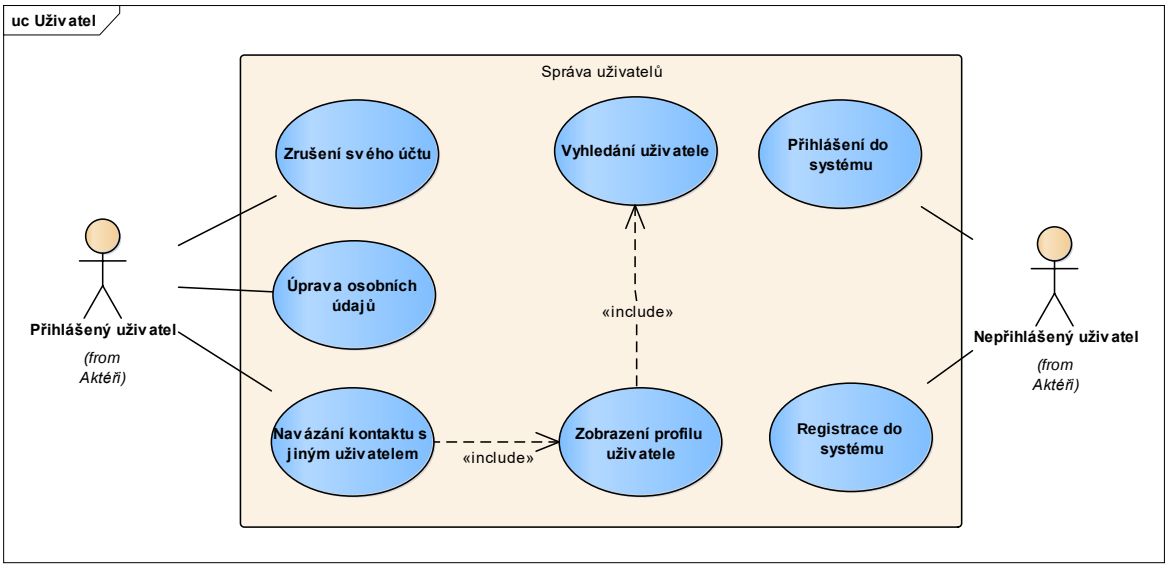

uc Projekt
Správa projektů
Vytvoření projektové dokumentace
Editace projektu
Sdílení projektu
Odstranění projektu
Vytvoření projektu
Zobrazení všech projektů
«include»
«include»
«include»
«include»
Autor projektu
(from Aktéři)
Přihlášený uživatel
(from Aktéři)


uc Případy užití
Správa případů užití
Vytvoření nového případu užití
Přidání scénáře k případu užití
Zobrazení matice sledovatelnosti požadavků
Editace případu užití
Odstranění případu užití
Vypsání všech případů užití
«include»
«include»
«include»
Člen projektu


uc Scénáře
Správa scénářů
Vytvoření nového scénáře
Editace scénáře
Odebrání scénáře
Vytvoření alternativního scénáře
Zobrazení scénářů ve stromové struktuře
Zobrazení zvoleného scénáře
«include»
«include»
«include»
Člen projektu
(from Aktéři)

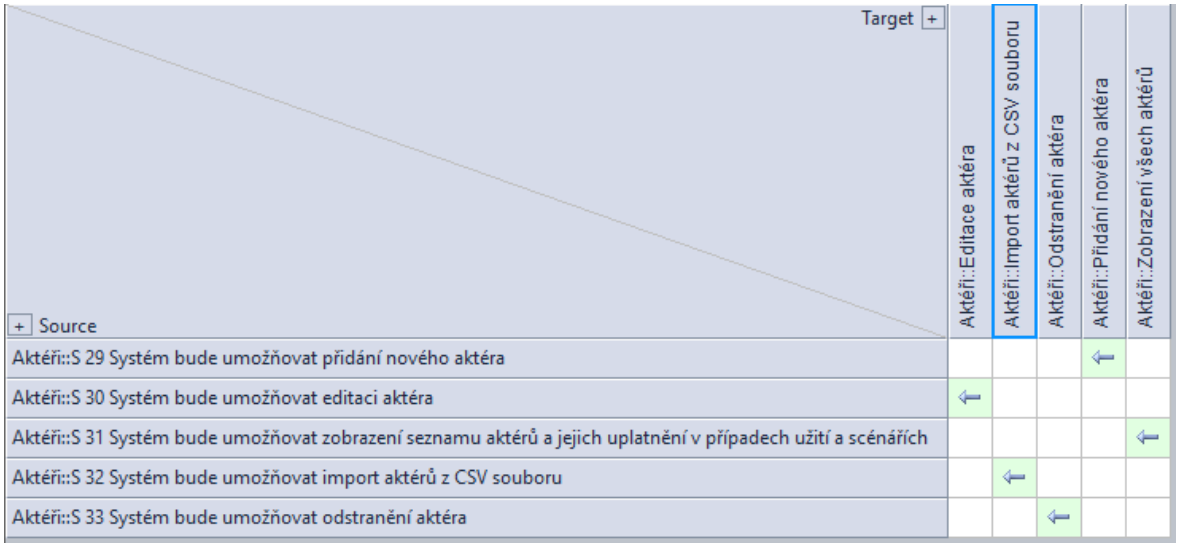

uc Soubory
Správa souborů
Vlastník souboru
(from Aktéři)
Odebrání souboru
«include»
Stažení nahraného souboru
«include»
Zobrazení všech nahraných souborů
Nahrání nového souboru
Člen projektu
(from Aktéři)


uc Tým
Správa týmu
Přidání nového člena do projektu
Autor projektu
(from Aktéři)
Odebrání člena projektu
«include»
Zobrazení seznamu členů projektu
Člen projektu
(from Aktéři)


uc Uživatel
Správa uživatelů
Zrušení svého účtu
Vyhledání uživatele
Přihlášení do systému
Úprava osobních údajů
«include»
Navázání kontaktu s jiným uživatelem
«include»
Zobrazení profilu uživatele
Registrace do systému
Přihlášený uživatel
(from Aktéři)
Nepřihlášený uživatel
(from Aktéři)

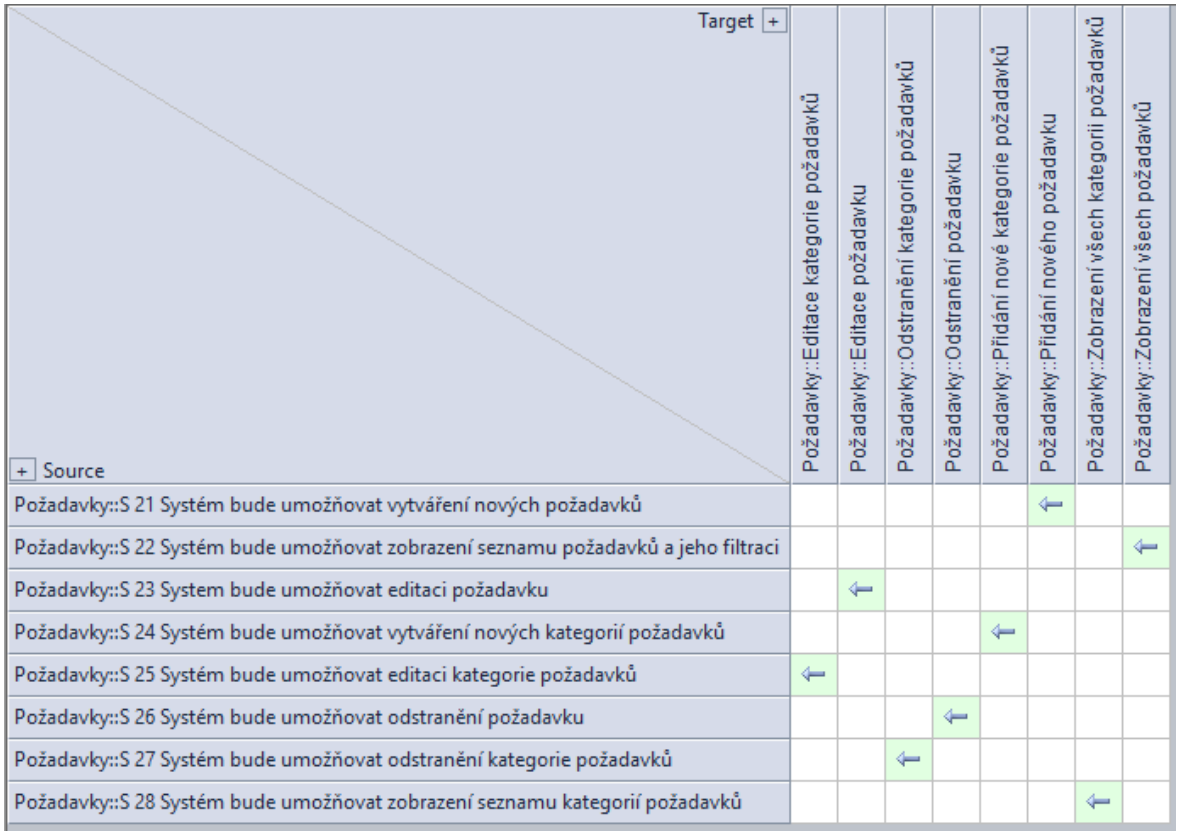
# Příloha B – Matice sledovatelnosti požadavků

Kategorie Aktéři



| Source \ Target | Aktéři::Editace aktéra | Aktéři::Import aktérů z CSV souboru | Aktéři::Odstranění aktéra | Aktéři::Přidání nového aktéra | Aktéři::Zobrazení všech aktérů |
|---|---|---|---|---|---|
| Aktéři::S 29 Systém bude umožňovat přidání nového aktéra | | | | ← | |
| Aktéři::S 30 Systém bude umožňovat editaci aktéra | ← | | | | |
| Aktéři::S 31 Systém bude umožňovat zobrazení seznamu aktérů a jejich uplatnění v případech užití a scénářích | | | | | ← |
| Aktéři::S 32 Systém bude umožňovat import aktérů z CSV souboru | | ← | | | |
| Aktéři::S 33 Systém bude umožňovat odstranění aktéra | | | ← | | |

Kategorie Požadavky



| Source \ Target | Požadavky::Editace kategorie požadavků | Požadavky::Editace požadavku | Požadavky::Odstranění kategorie požadavků | Požadavky::Odstranění požadavku | Požadavky::Přidání nové kategorie požadavků | Požadavky::Přidání nového požadavku | Požadavky::Zobrazení všech kategorií požadavků | Požadavky::Zobrazení všech požadavků |
|---|---|---|---|---|---|---|---|---|
| Požadavky::S 21 Systém bude umožňovat vytváření nových požadavků | | | | | | ← | | |
| Požadavky::S 22 Systém bude umožňovat zobrazení seznamu požadavků a jeho filtraci | | | | | | | | ← |
| Požadavky::S 23 System bude umožňovat editaci požadavku | | ← | | | | | | |
| Požadavky::S 24 Systém bude umožňovat vytváření nových kategorií požadavků | | | | | ← | | | |
| Požadavky::S 25 Systém bude umožňovat editaci kategorie požadavků | ← | | | | | | | |
| Požadavky::S 26 Systém bude umožňovat odstranění požadavku | | | | ← | | | | |
| Požadavky::S 27 Systém bude umožňovat odstranění kategorie požadavků | | | ← | | | | | |
| Požadavky::S 28 Systém bude umožňovat zobrazení seznamu kategorií požadavků | | | | | | | ← | |

Kategorie Případy užití

| Source \ Target | Případy užití::Editace případu užití | Případy užití::Odstranění případu užití | Případy užití::Přidání scénáře k případu užití | Případy užití::Vypsání všech případů užití | Případy užití::Vytvoření nového případu užití | Případy užití::Zobrazení matice sledovatelnosti požadavků |
|---|---|---|---|---|---|---|
| Případy užití::S 34 Systém bude umožňovat přidání nového případu užití | | | | | ← | |
| Případy užití::S 35 Systém bude umožňovat editaci případu užití | ← | | | | | |
| Případy užití::S 36 Systém bude umožňovat vypsat seznam případů užití | | | | ← | | |
| Případy užití::S 37 Systém bude umožňovat přidání scénáře k případu užití | | | ← | | | |
| Případy užití::S 38 Systém bude umožňovat odebrání případu užití | | ← | | | | |
| Případy užití::S 39 Systém bude umožňovat zobrazení matice sledovatelnosti požadavků | | | | | | ← |

Kategorie Projekt

| Source \ Target | Projekt::Editace projektu | Projekt::Odstranění projektu | Projekt::Sdílení projektu | Projekt::Vytvoření projektové dokumentace | Projekt::Vytvoření projektu | Projekt::Zobrazení všech projektů |
|---|---|---|---|---|---|---|
| Projekt::S 01 Systém bude umožňovat vytvoření nového projektu | | | | | ← | |
| Projekt::S 02 Systém bude umožňovat editaci projektu | ← | | | | | |
| Projekt::S 03 Systém bude umožňovat sdílení projektu s vybranými uživateli | | | ← | | | |
| Projekt::S 04 Systém bude umožňovat export informací o projektu - dokumentace ve formátu PDF | | | | ← | | |
| Projekt::S 05 Systém bude umožňovat odstranění projektu | | ← | | | | |

Kategorie Scénáře

| Source \ Target | Scénáře::Editace scénáře | Scénáře::Odebrání scénáře | Scénáře::Vytvoření alternativního scénáře | Scénáře::Vytvoření nového scénáře | Scénáře::Zobrazení scénářů ve stromové struktuře | Scénáře::Zobrazení zvoleného scénáře |
|---|---|---|---|---|---|---|
| Scénáře::S 40 Systém bude umožňovat vytvoření scénáře k případu užití | | | | ← | | |
| Scénáře::S 41 Systém bude umožňovat editaci scénáře | ← | | | | | |
| Scénáře::S 42 Systém bude umožňovat vytvářet alternativní scénář k hlavnímu | | | ← | | | |
| Scénáře::S 43 Systém bude umožňovat zobrazení scénářů ve stromové struktuře | | | | | ← | |
| Scénáře::S 44 Systém bude umožňovat odebrání scénáře | | ← | | | | |

Kategorie Soubory

| Source \ Target | Soubory::Nahrání nového souboru | Soubory::Odebrání souboru | Soubory::Stažení nahraného souboru | Soubory::Zobrazení všech nahraných souborů |
|---|---|---|---|---|
| Soubory::S 45 Systém bude umožňovat nahrávání souborů k projektu | ← | | | |
| Soubory::S 46 Systém bude umožňovat stažení nahraného souboru | | | ← | |
| Soubory::S 47 Systém bude umožňovat zobrazení seznamu souborů v projektu | | | | ← |
| Soubory::S 48 Systém bude umožňovat odebrání souboru | | ← | | |

Kategorie Tým

| Source \ Target | Tým::Odebrání člena projektu | Tým::Přidání nového člena do projektu | Tým::Zobrazení seznamu členů projektu |
|---|---|---|---|
| Tým::S 49 Systém bude umožňovat přidávání nových členů do projektu | | ← | |
| Tým::S 50 Systém bude umožňovat odebírání členů z projektu | ← | | |
| Tým::S 51 Systém bude umožňovat zobrazit seznam členů projektu | | | ← |

Kategorie Uživatelé

| Source \ Target | Uživatel::Editace osobních údajů | Uživatel::Navázání kontaktu s jiným uživatelem | Uživatel::Přihlášení do systému | Uživatel::Registrace do systému | Uživatel::Úprava osobních údajů | Uživatel::Vyhledání uživatele | Uživatel::Zobrazení profilu uživatele | Uživatel::Zrušení svého účtu |
|---|---|---|---|---|---|---|---|---|
| Uživatel::S 06 Systém bude umožňovat registraci nového uživatele | | | | ← | | | | |
| Uživatel::S 07 Systém bude umožňovat přihlášení uživatele | | | ← | | | | | |
| Uživatel::S 08 Systém bude umožňovat úpravu osobních údajů uživatele | ← | | | | ← | | | |
| Uživatel::S 09 Systém bude umožňovat vyhledání jiného uživatele podle jména | | | | | | ← | | |
| Uživatel::S 10 Systém bude umožňovat navazování kontaktů s ostatními uživateli | | ← | | | | | | |
| Uživatel::S 11 Systém bude umožňovat zrušení účtu uživatele | | | | | | | | ← |
| Uživatel::S 12 Systém bude umožňovat zobrazení profilu uživatele | | | | | | | ← | |

## Příloha C – Scénáře

| Případ užití: Vytvoření nového úkolu |
|---|
| ID: 2 |
| Stručný popis:<br>Člen projektu vytváří nový úkol. |
| Hlavní aktéři:<br>Člen projektu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Člen projektu je přihlášen v systému a má otevřený projekt. |
| Hlavní scénář:<br>1. Případ je Členem projektu spuštěn příkazem „vytvořit nový úkol“.<br>2. Dokud jsou údaje o úkolu neplatné:<br>2.1. Systém žádá Člena projektu, aby zadal všechny údaje včetně znění úkolu, vyřizovatele úkolu, priority úkolu a datumu očekávaného splnění úkolu.<br>2.2. Systém ověří zadané údaje.<br>3. Systém vytvoří nový úkol.<br>4. Systém odešle upozornění členovi projektu, který dostal úkol na starost. |
| Výstupní podmínky:<br>Úkol byl úspěšně vytvořen. |
| Alternativní scénáře:<br>Neplatné datum. |

| Alternativní scénář: Vytvoření nového úkolu: Neplatné datum |
|---|
| ID: 2.1 |
| Stručný popis:<br>Systém informuje Člena projektu, že zadal neplatné datum očekávaného splnění úkolu. |
| Hlavní aktéři:<br>Člen projektu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Člen projektu zadal neplatné datum splnění úkolu. |
| Alternativní scénář:<br>1. Alternativní scénář začíná krokem 2.2 hlavního scénáře.<br>2. Systém informuje Člena projektu, že zadal neplatné datum splnění úkolu. |
| Výstupní podmínky:<br>Žádné. |

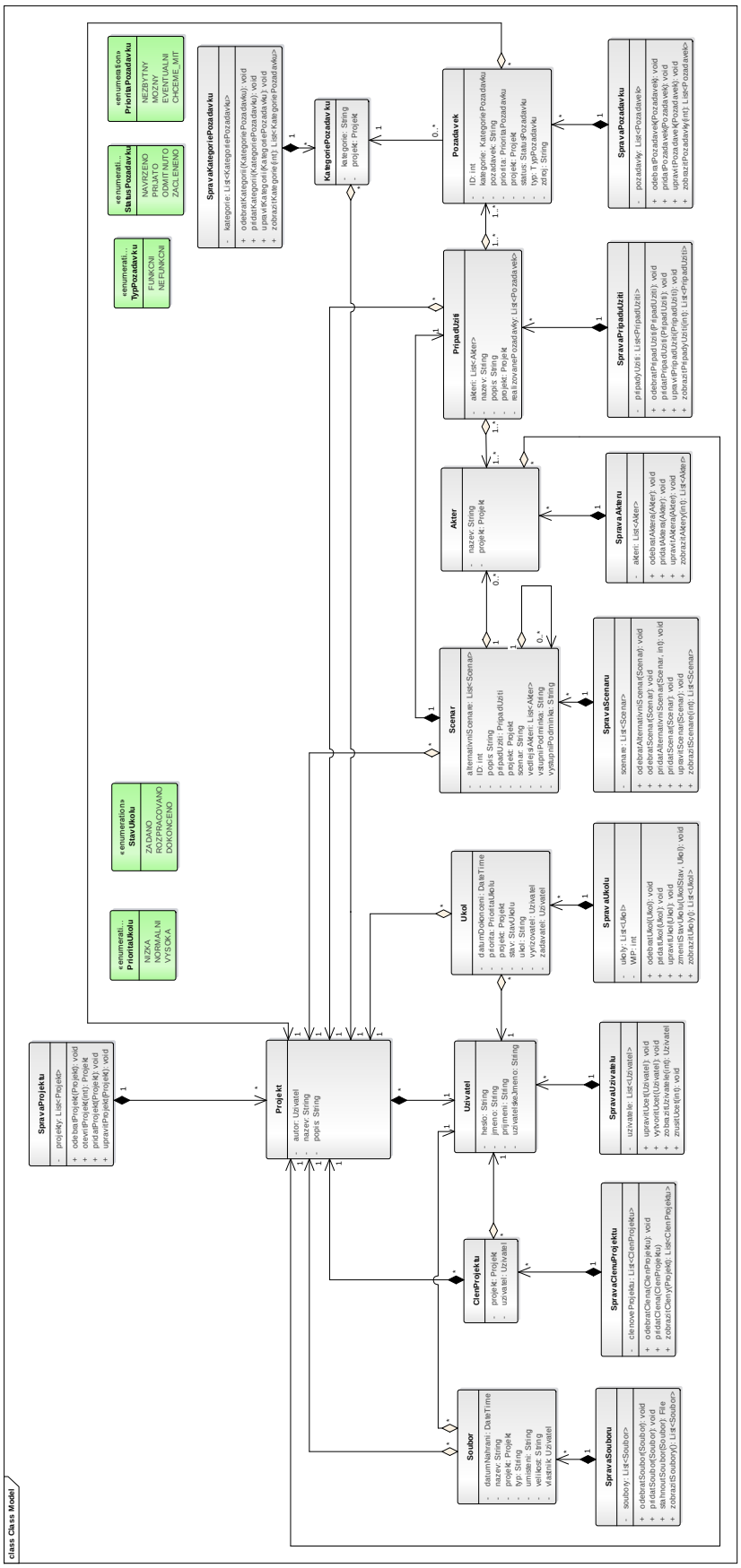
| Případ užití: Editace úkolu |
| --- |
| ID: 3 |
| Stručný popis:<br>Zadavatel úkolu upraví informace o úkolu |
| Hlavní aktéři:<br>Zadavatel úkolu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Zadavatel úkolu je přihlášen v systému a má otevřený projekt.<br>2. Zadavatel úkolu nalezl úkol, který vytvořil. |
| Hlavní scénář:<br>1. Případ je Zadavatelem úkolu spuštěn příkazem „upravit úkol“.<br>2. Dokud jsou údaje o úkolu neplatné:<br>2.1. Systém žádá Zadavatele úkolu, aby zadal všechny údaje včetně znění úkolu, vyřizovatele úkolu, priority úkolu a datumu očekávaného splnění úkolu.<br>2.2. Systém ověří zadané údaje.<br>3. Systém upraví úkol.<br>4. Systém odešle upozornění Vyřizovateli úkolu o provedené úpravě. |
| Výstupní podmínky:<br>Úkol byl úspěšně upraven. |
| Alternativní scénáře:<br>Neplatné datum. |

| Případ užití: Odstranění úkolu |
| --- |
| ID: 4 |
| Stručný popis:<br>Zadavatel úkolu odstraní úkol, který vytvořil |
| Hlavní aktéři:<br>Zadavatel úkolu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Zadavatel úkolu je přihlášen v systému a má otevřený projekt.<br>2. Zadavatel úkolu nalezl úkol, který vytvořil. |
| Hlavní scénář:<br>1. Případ je Zadavatelem úkolu spuštěn příkazem „odstranit úkol“.<br>2. Systém odstraní úkol.<br>3. Systém odešle upozornění Vyřizovateli úkolu o odstranění úkolu. |
| Výstupní podmínky:<br>Úkol byl úspěšně odstraněn. |
| Alternativní scénáře:<br>Žádné. |

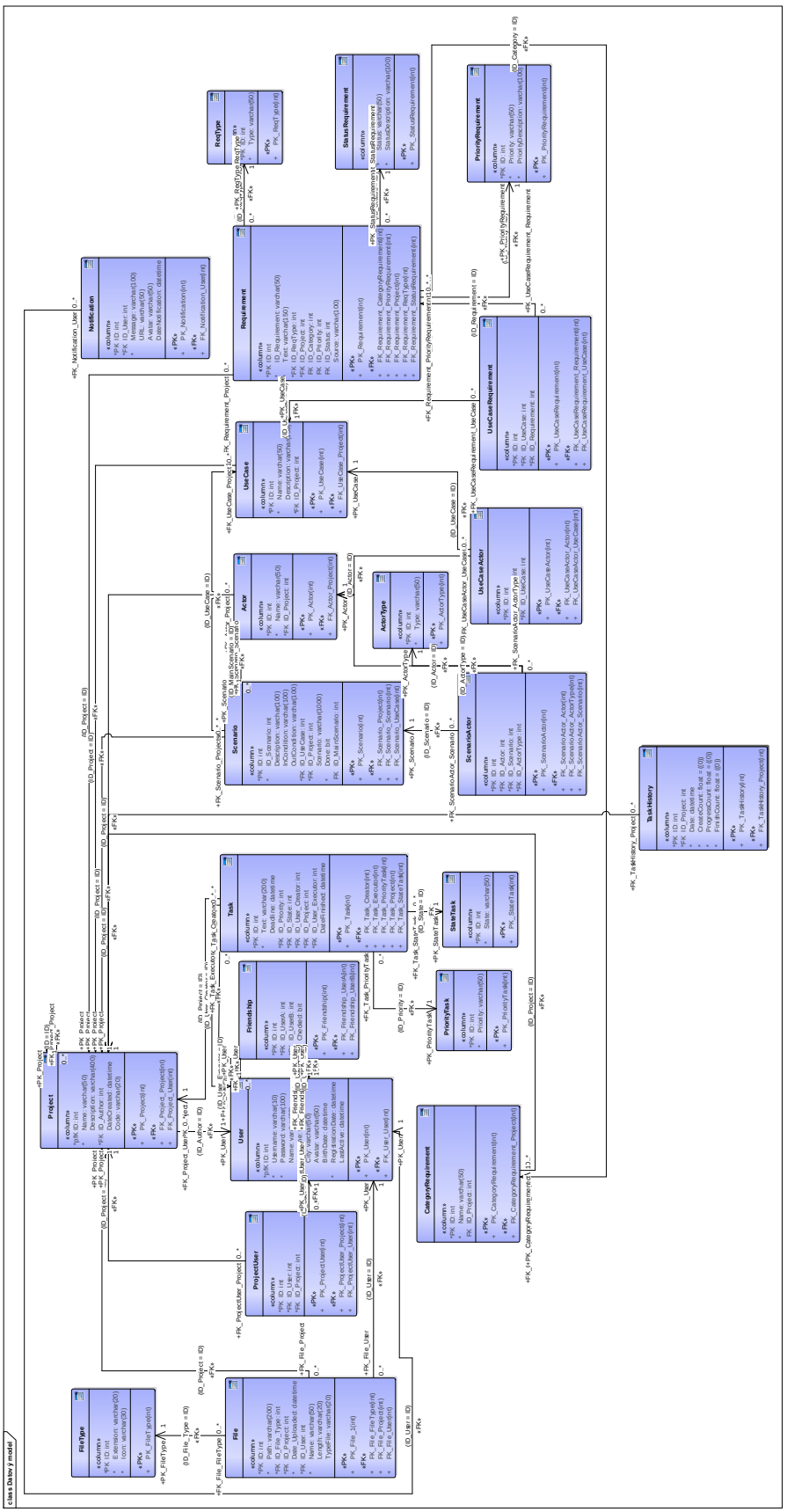
| Případ užití: Upravení limity WIP |
| --- |
| ID: 5 |
| Stručný popis:<br>Autor projektu upraví hodnotu limity WIP. |
| Hlavní aktéři:<br>Autor projektu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Autor projektu je přihlášen v systému a má otevřený projekt. |
| Hlavní scénář:<br>1. Případ je Autorem projektu spuštěn příkazem „upravit hodnotu WIP“.<br>2. Dokud je hodnota WIP neplatná:<br>2.1. Systém žádá Autora projektu, aby zadal správnou hodnotu.<br>2.2. Systém ověří správnost hodnoty WIP.<br>3. Systém upraví hodnotu limity WIP. |
| Výstupní podmínky:<br>Hodnota WIP byla úspěšně změněna. |
| Alternativní scénáře:<br>Neplatná hodnota |

| Alternativní scénář: Upravení limity WIP: Neplatná hodnota |
| --- |
| ID: 5.1 |
| Stručný popis:<br>Systém informuje Autora projektu, že zadal neplatnou hodnotu WIP. |
| Hlavní aktéři:<br>Autor projektu |
| Vedlejší aktéři:<br>Žádní. |
| Vstupní podmínky:<br>1. Zákazník zadal hodnotu WIP nekorespondující s aktuálním počtem rozpracovaných úkolů. |
| Alternativní scénář:<br>1. Alternativní scénář začíná krokem 2.2 hlavního scénáře.<br>2. Systém informuje Autora projektu, že zadal neplatnou hodnotu limity WIP. |
| Výstupní podmínky:<br>Žádné. |

# Příloha D – Návrhový model tříd



# Příloha E – Datový model



## Příloha F – Přiložené DVD

Obsah přiloženého DVD:

- zdrojové kódy aplikace (projekt z vývojového prostředí Microsoft Visual Studia),
- analýza v programu Enterprise Architect,
- soubor představující zálohu databáze,
- soubor udržující internetový odkaz na spustitelnou aplikaci,
- elektronická podoba teoretické části práce ve formátu PDF.