

Návrh aplikace pro evidenci smluv

Marek Scheibinger



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Marek Scheibinger**
Osobní číslo: **A17566**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Softwarové inženýrství**
Forma studia: **Prezenční**
Téma práce: **Návrh aplikace pro evidenci smluv**
Téma práce anglicky: **An Application Design for Contract Records**

Zásady pro vypracování

1. Proveďte rešerši existujících řešení.
2. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
3. Proveďte rozbor a analýzu požadavků na zvolené řešení.
4. Realizujte navrženou aplikaci.
5. Věnujte pozornost zabezpečení aplikace.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam doporučené literatury:

1. NEUSTADT, Ila; ARLOW, Jim. *UML 2 a unifikovaný proces vývoje aplikací*. Computer Press, Albatros Media as, 2016.
2. JOHNSON, Glenn. *Programming in HTML5 with JavaScript and CSS3: training guide*. Redmond, Wash.: Microsoft, 2013. ISBN 978-0735674387.
3. UNHELKAR, Bhuvan. *Software engineering with uml*. Auerbach Publications, 2017.
4. FREEMAN, Adam. *Pro Asp. net Core Mvc*. Apress, 2016.
5. JAKOBUS, Benjamin. *Mastering Bootstrap 4: Master the latest version of Bootstrap 4 to build highly customized responsive web apps*. Packt Publishing Ltd, 2018.
6. BEN-GAN, Itzik; DAVIDSON, Louis; VARGA, Stacia. *MCSA SQL Server 2016 Database Development Exam Ref 2-pack: Exam Refs 70-761 and 70-762*. Microsoft Press, 2017.

Vedoucí bakalářské práce:

doc. Ing. Petr Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:
Termín odevzdání bakalářské práce:

28. listopadu 2019
15. května 2020



doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 3.8.2020

Marek Scheibinger, v.r.

.....
podpis diplomanta

ABSTRAKT

Bakalářská práce se zabývá návrhem a implementací systému pro správu smluv. V teoretické části je rozebráno několik již existujících řešení, jsou popsány a je zdůvodněno použití technologií, které byly použity k implementaci této webové aplikace. Praktická část se věnuje návrhu aplikace a implementaci. Návrh aplikace je realizován prostřednictvím sběru požadavků a analýzy systému. Implementace je uskutečněna pomocí frameworku ASP.NET Core MVC s využitím databáze MS SQL Server.

Klíčová slova: Systémy pro správu smluv, Webová aplikace, ASP.NET Core, MVC, C#, Entity Framework

ABSTRACT

Bachelor's thesis is dedicated to the design and implementation of a contract management system. The theoretical part discusses several already existing solutions as well as describes and justifies the utilization of technologies that have been used for implementation of this web application. The practical part deals with the application design and implementation. The application design is realized via requirements gathering and system analysis. The implementation is carried out using ASP.NET Core MVC Framework and MS SQL Server database.

Keywords: Contract management systems, Web applications, ASP.NET Core, MVC, C#, Entity Framework

Tímto bych chtěl poděkovat panu vedoucímu práce doc. Ing. Petrovi Šilhavému, Ph.D. za odborné vedení a konzultace bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

OBSAH	6
ÚVOD.....	7
I. TEORETICKÁ ČÁST	8
1 REŠERŠE EXISTUJÍCÍCH ŘEŠENÍ.....	9
1.1 AGILOFT	9
1.2 ONESOFT CONNECT.....	10
1.3 CONCORD.....	11
1.4 ODLIŠNOSTI VYVÍJENÉ APLIKACE	12
2 POUŽITÉ TECHNOLOGIE.....	13
2.1 ASP.NET CORE FRAMEWORK	13
2.2 NÁVRHOVÝ VZOR MVC.....	14
2.3 PROGRAMOVACÍ JAZYK C#	15
2.4 ENTITY FRAMEWORK	15
2.5 MICROSOFT SQL SERVER	16
II. PRAKTICKÁ ČÁST	17
3 NÁVRH APLIKACE	18
3.1 FUNKČNÍ POŽADAVKY	18
3.1.1 Uživatelské požadavky.....	18
3.1.2 Systémové požadavky	21
3.2 NEFUNKČNÍ POŽADAVKY	28
3.3 ANALÝZA	29
3.3.1 Modelování případů užití	29
3.3.2 E-R diagram	44
3.3.3 Diagram tříd	44
3.3.4 Drátěné modely	46
4 IMPLEMENTACE	51
4.1 VYTVOŘENÍ ARCHITEKTURY	51
4.2 IMPLEMENTACE FUNKCÍ	53
4.3 IMPLEMENTACE ZABEZPEČENÍ.....	56
4.4 TESTOVÁNÍ.....	61
4.5 GRAFICKÉ ZPRACOVÁNÍ	64
ZÁVĚR	66
SEZNAM POUŽITÉ LITERATURY.....	67
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	70
SEZNAM OBRÁZKŮ	71
SEZNAM TABULEK.....	73
SEZNAM PŘÍLOH.....	75
PŘÍLOHA P I: E-R DIAGRAM.....	76

ÚVOD

Teoretická část této práce je zaměřena na rešerši několika již existujících řešení, jejich porovnání a závěry. Dle provedené rešerše jsou vybrány funkce odlišující vyvíjenou aplikaci. Jsou popsány technologie, které byly následně použity při implementaci aplikace, zejména pak ASP.NET Core MVC Framework a programovací jazyk C#. Je zdůvodněno jejich použití a popsány jejich výhody při práci na tomto typu webových aplikací.

Praktická část práce obsahuje návrh softwaru, jehož součástí je sběr požadavků a analýza systému. Funkční a nefunkční požadavky vycházejí z rešerše již existujících řešení. V analytické části je vytvořen model případů užití, diagram tříd, E-R diagram a návrh front-endové části aplikace pomocí drátěných modelů. Implementační část obsahuje vytvoření architektury, implementaci funkcí, implementaci zabezpečení systému (proti útokům typu SQL Injection a CSFR, šifrování hesel pomocí kryptografické funkce, rozdělení uživatelů do rolí a zajištění šifrované komunikace klient-server pomocí protokolu HTTPS), manuální penetrační a integrační testovací část a způsob grafického zpracování aplikace. Na základě návrhu aplikace a jeho implementace je vytvořen vlastní systém pro správu smluv.

I. TEORETICKÁ ČÁST

1 REŠERŠE EXISTUJÍCÍCH ŘEŠENÍ

V této kapitole je probráno, porovnáno a vyhodnoceno několik vybraných systémů pro správu smluv. Při shrnutí hodnocení je vycházeno ze zveřejněných dokumentací a veřejně dostupných instruktážních videí daných systémů. Důraz je kladen na způsob integrace evidence smluv v aplikaci a na funkce, kterými může uživatel s danou evidencí interagovat. Také je kladen důraz na funkce těchto aplikací, které jednotlivé hodnocené programy odlišují.

1.1 Agiloft

Firma Agiloft, Inc. je americká firma zaměřující se na řízení obchodních procesů. Jejich stejnojmenný software se zaměřuje právě na správu smluv ve webovém prostředí.

Důvodem, proč je tento software zmiňován, je fakt, že již pátým rokem byl vyhodnocen zámořským magazínem PC MAG [1] jako nejlepší systém pro správu smluv a nejvhodnější systém pro firmy všech velikostí pro rok 2020.

Aplikace obsahuje domovskou stránku, kde jsou odkazy na profil uživatele, vytvoření nové smlouvy, čekající žádosti a zobrazení smluv ve formě „všechny smlouvy“ a „moje smlouvy“. Tato domovská stránka je vysoce přizpůsobitelná, to znamená, že je možno měnit její rozložení a barvy. Obsah, tedy jednotlivé odkazy na funkce programu, jsou také přizpůsobitelné.

Při vytváření nové smlouvy je uživatel tázán na vyplnění informací o smlouvě ve formě formuláře. Tyto informace jsou – typ smlouvy, informace o žadateli, informace o smlouvě, jako možnost obnovy smlouvy, název smlouvy, začátek a konec smlouvy a další. Je možnost také nahrání samotné smlouvy ve formátu DOCX, která je předvyplněna pomocí šablony.

Po vytvoření smlouvy je potřeba, aby smlouvu schválil či neschválil nadřízený. Poté smlouva „spadne“ do životního cyklu smlouvy.

Aplikace také umožňuje zobrazení několika grafů, například procentuální zobrazení momentálního stavu smluv.

Struktura tohoto programu a workflow velmi přehledná a intuitivní. Obsahuje všechny důležité funkce pro evidenci smluv.



Obrázek 1: Domovská stránka systému Agiloft [2]

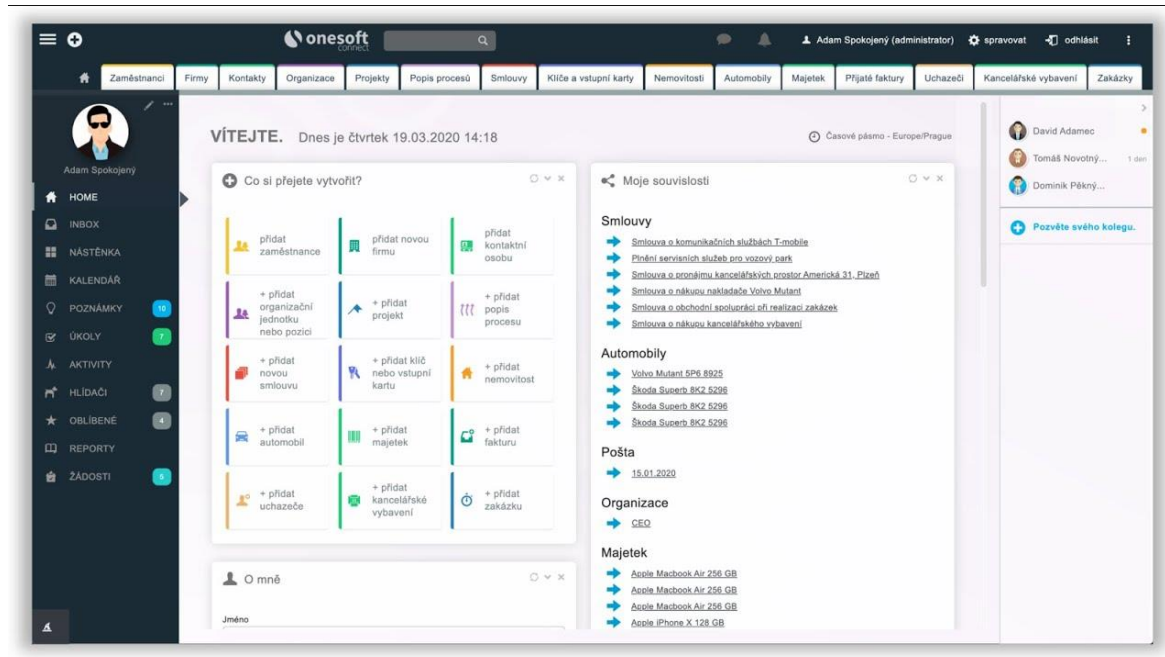
1.2 Onesoft Connect

Onesoft Connect, Inc. je Čechy založená firma sídlící v Kalifornii, USA. Stejnojmenný software je zařazen právě z důvodu, že jeho autory jsou Češi.

Samotný software není pouze evidence smluv. Uživatel si může vybrat z několika evidencí, či si vytvořit svoji vlastní. Aplikace také funguje jako systém pro správu úkolů, obsahuje chatovací místnost a vysokou úroveň přizpůsobivosti.

Tím, že tento software defacto funguje jako systém pro řízení firmy, je velmi rozsáhlý, a to může v konečném důsledku uškodit. Uživatel si může jednotlivé smlouvy, či samotné atributy smlouvy hlídat pomocí „hlídače“, a tak může být upozorněn, pokud se smlouva například blíží jejímu vypršení. Po například půl roce používání softwaru, kdy má uživatel nastavené hlídače na atributy a samotné smlouvy, kterých mohou být stovky – a nejen smlouvy, ale další vytvořené evidence, kterých mohou být desítky, tak by se snadno tato aplikace mohla stát velmi nepřehlednou a tudíž neefektivní. Aplikace neobsahuje životní cykly položek v jednotlivých evidencích.

Onesoft Connect je vhodný pro malé až středně velké firmy, kde je potřeba evidovat více prvků najednou a mít ke všemu přístup na jednom místě. Ve velkých firmách by mohl velmi rychle nastat případ, který je zmíněn v předchozím odstavci.



Obrázek 2: Domovská stránka systému Onesoft Connect [3]

1.3 Concord

Firma Concord, Inc. je zahraniční firma sídlící v San Franciscu, USA s pobočkou v Paříži ve Francii. Její stejnojmenný software se specializuje právě na správu smluv. Tento software byl vyhodnocen magazínem PC MAG jako třetí nejlepší systém pro správu smluv pro rok 2020. [2]

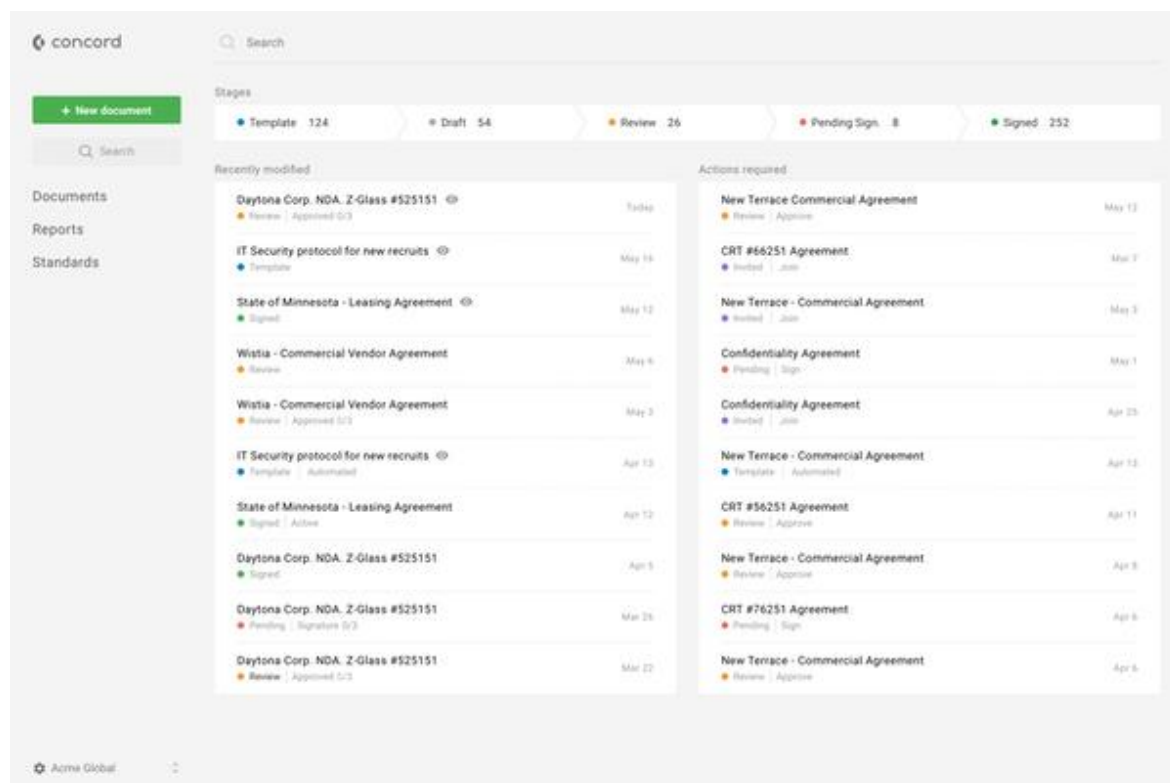
Po přihlášení do aplikace se uživatel dostane na domovskou stránku, kde nalezne informace o nedávných změnách ve smlouvách a čekající žádosti. Také je k dispozici přehled o všech smlouvách ve formě zobrazení celkového počtu smluv v jednotlivých fázích jejich životního cyklu. Je zde i tlačítko pro vytvoření nového dokumentu/smlouvy.

Při vytváření nové smlouvy se uživateli zobrazí napůl rozdělená obrazovka, kdy na levé straně je editovatelný dokument vytvořen podle výchozí šablony, a na pravé straně je formulář tážající uživatele klíčové informace o smlouvě.

Po vytvoření smlouvy je možnost smlouvu dále upravovat a připomínkovat. Dokument se může exportovat do PDF nebo jiných formátů. Při schválení tvaru smlouvy přejde smlouva do stavu, kdy se čeká na podpisy zainteresovaných stran.

Po podpisu smlouvy je smlouva aktivní dle začátku a konce smlouvy. Dále je také možnost zobrazení několika grafů znázorňujících například počet smluv podepsaných každý měsíc za uplynulý rok.

Aplikace Concord je velmi přehledná a efektivní. Je možno ji využít pro firmy všech velikostí. Nedostatkem oproti zmíněným programům může být míra přizpůsobitelnosti uživatelem. Aplikace je však graficky provedena přehledně a její navigace je intuitivní.



Obrázek 3: Domovská stránka systému Concord [4]

1.4 Odlišnosti vyvíjené aplikace

Hodnocené zmíněné aplikace jeví známky velmi dobrého zpracování, pokud jde o evidenční část, což je samozřejmě jejich hlavní účel.

S ohledem na zmíněná řešení bude v tomto projektu kromě vlastní evidence smluv kladen důraz i na řízení výkonu zaměstnanců budoucího uživatele. Proto pro členy statutárního orgánu firmy (nebo jimi pověřené osoby – dále „podepisovatelé“) je vytvořen přehled o výkonu jednotlivých osob podílejících se na vytvoření smlouvy, pokud jde o počet uzavřených smluv za zvolené časové období. V aplikaci jsou definovány role jednotlivých uživatelů a jejich hierarchie.

2 POUŽITÉ TECHNOLOGIE

Server builtwith.com zpracovává statistiky nejpoužívanějších technologií napříč celým internetem. Žebříček [5] pro rok 2020 je následující: 1. PHP (31%), 2. ASP.NET (30%), 3. OpenResty (5%) a ostatní (34%).

Tento žebříček jasně představuje dva giganty webových technologií současnosti. Důležité je však zmínit, že PHP je programovací jazyk, zatímco ASP.NET je framework používající programovací jazyky C# nebo F#. Nejpoužívanějšími PHP frameworky [6] jsou pak Laravel, Codeigniter nebo Symfony.

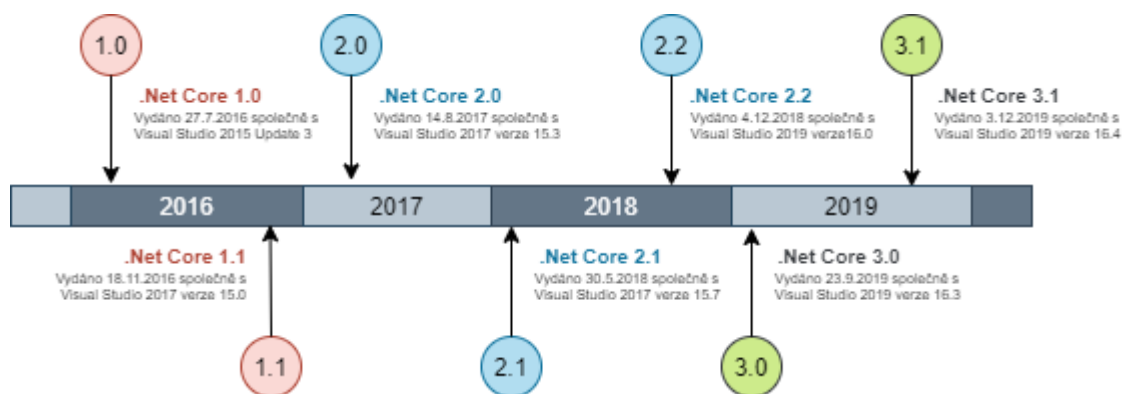
Pro tento projekt je využit ASP.NET Core MVC framework z důvodů popsanych v dalších kapitolách.

2.1 ASP.NET Core Framework

ASP.NET Core [7] open-source framework je přepracování ASP.NET 4.x. Využívá .NET Framework i .NET Core, čímž umožňuje cross-platformní nasazení, tedy kromě sestavení (build) pro Windows podporuje i sestavení pro platformy macOS nebo Linux. Oproti ASP.NET 4.x je ASP.NET Core [8] výkonnější a uživatel může zvolit pro svůj vývoj kromě IDE Visual Studio i IDE Visual Studio pro Mac nebo Visual Studio Code, s využitím programovacího jazyku C# nebo F#.

Co se týče rychlosti překladu a kvality kódu, ASP.NET [10] je přesnější, rychlejší a systematictější než zmíněné PHP. ASP.NET nabízí kvalitní a uživatelsky přívětivé balíčky zabezpečení, například `Microsoft.AspNet.Identity`, nebo `Microsoft.AspNet.Authorization`, které jsou v tomto projektu využity.

ASP.NET Core [7] také používá takzvané Razor Pages, které jsou doporučeným přístupem k vytvoření webového uživatelského rozhraní. Razor View v tomto případě umožňuje kombinovat syntaxi HTML a C#, s možností využití dalších technologií jako jsou CSS, Javascript nebo jQuery.

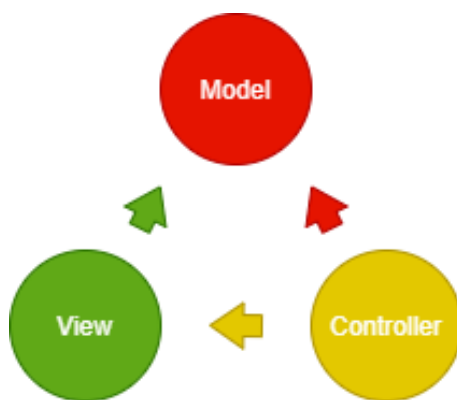


Obrázek 4: Historie ASP.NET Core [9]

Pro tento projekt je použit konkrétně ASP.NET Core ve verzi 2.1. Použitým IDE je Visual Studio 2017. Tato verze je v době realizace bakalářské práce stále v podpoře Microsoftu, podpora končí k 21. srpnu 2021 [11]. Je však možnost projekt aktualizovat na verzi 2.2, následně na 3.0 a konečně na nejnovější verzi 3.1 [12].

2.2 Návrhový vzor MVC

Návrhový vzor architektury MVC [13] (Model-View-Controller) odděluje aplikaci do tří hlavních skupin komponent: modelů, zobrazení a kontrolerů. Pomocí tohoto modelu jsou požadavky uživatelů směřovány na kontroler, který je zodpovědný za práci s modelem pro provádění akcí uživatele nebo načtení výsledků dotazů. Kontroler zvolí zobrazení, které se zobrazí uživateli, a poskytne mu veškerá data modelu, která vyžaduje.



Obrázek 5: Diagram architektury MVC [13]

Výhodou využití tohoto návrhového vzoru je tedy oddělení jednotlivých částí aplikace do jasně daných komponent. Ulehčuje tak vývojovou fázi aplikace, kdy umožňuje práci více vývojářů na více komponentech současně, bez vzájemného rušení. Také ulehčuje práci v provozní části životního cyklu aplikace. Pro zavádění nových funkcí či změnu již stávajících funkcí totiž není třeba brát v úvahu celý software, nýbrž pouze komponentu či

komponenty, které s danou funkcí pracují. Náklady na opravu či rozvoj aplikace se tak zásadně snižují [14].

2.3 Programovací jazyk C#

C# [15] je vysokoúrovňový, silně-typovaný, typově-bezpečný, objektově-orientovaný programovací jazyk. Je multiplatformní, a tedy dokáže pracovat s několika platformově-specifickými překladači a frameworky, obzvlášť však právě s Microsoft .NET Frameworkem pro Windows.

Tento jazyk [15] bohatě implementuje paradigma objektové orientace, obsahuje zapouzdření, dědičnost a polymorfismus.

Často používanou funkcí v tomto projektu, která je pro tento projekt klíčová, a kterou programovací jazyk C# nabízí, je Language Integrated Query (LINQ). Jedná se [15] o dotazovací jazyk, který je součástí jazyka C# od verze 3.0 a .NET Frameworku od verze 3.5. LINQ poskytuje unifikovaný způsob, kterým se lze dotazovat na data v databázi, XML souboru nebo i na prvek v poli, typu `IEnumerable<T>`.

Důvodem, proč je využití LINQ pro tento projekt klíčové, je jeho odolnost vůči útokům typu SQL Injection [16]. SQL Injection [17] je typ útoku, ve kterém je škodlivý kód vložen do řetězců, které jsou poté předány instanci SQL Serveru pro parsování a provedení. Použitím LINQ se však tento problém vyřeší. LINQ dotazy [16] totiž nejsou skládány z manipulace řetězců. LINQ posílá všechny data do databáze ve formě SQL parametrů a tím pomáhá zastavit SQL Injection.

2.4 Entity Framework

Cílem EF [18] (Entity Framework) je umožnit uživateli interagovat s daty z relační databáze za použití objektového modelu, který je přímo propojen s doménovými objekty v aplikaci. Jinými slovy, místo toho, aby se přistupovalo k datům z databáze jako ke sbírce řádků a sloupců, je k nim přistupováno skrze silně typované objekty, kterým se říká entity. Na tyto entity je možno se dotazovat právě skrze LINQ.

EF [19] umožňuje dva přístupy: Code-First a Database-First. Code-First API vytvoří databázi za chodu, na základě vytvořených entitních tříd a konfiguraci. Také dokáže aktualizovat databázi za použití funkce zvané migrace. Database-First přístup se využívá tehdy, když je

již databáze připravena. EF následně podle již vytvořené databáze vytvoří odpovídající entity. Databáze se poté může manuálně měnit a dle ní aktualizovat zasažené entity.

Pro tento projekt je využit přístup Code-First.

2.5 Microsoft SQL Server

Zvoleným systémem řízení báze dat pro projekt je Microsoft SQL Server. Je to [20] relační databázový a analytický systém. Je postaven na standardizovaném dotazovacím jazyku SQL, také však zavádí vlastní obdobu zvanou Transact-SQL (T-SQL).

SQL Server [20] je primárně založen na řádkové a sloupcové tabulkové struktuře, která propojuje související datové elementy v různých tabulkách, což zabraňuje možné redundanci v případě, že by stejná data byly uloženy v databázi na více místech.

Relační model [20] také poskytuje referenční integritu a další integritní omezení pro udržení přesnosti dat.

Využití Microsoft SQL Server pro Windows umožňuje vývojáři pracovat v programu zvaném SQL Server Management Studio. Tento program je pro tento projekt využit v závislosti se zobrazením dat v databázi a testováním ukládaných dat na serveru.

II. PRAKTICKÁ ČÁST

3 NÁVRH APLIKACE

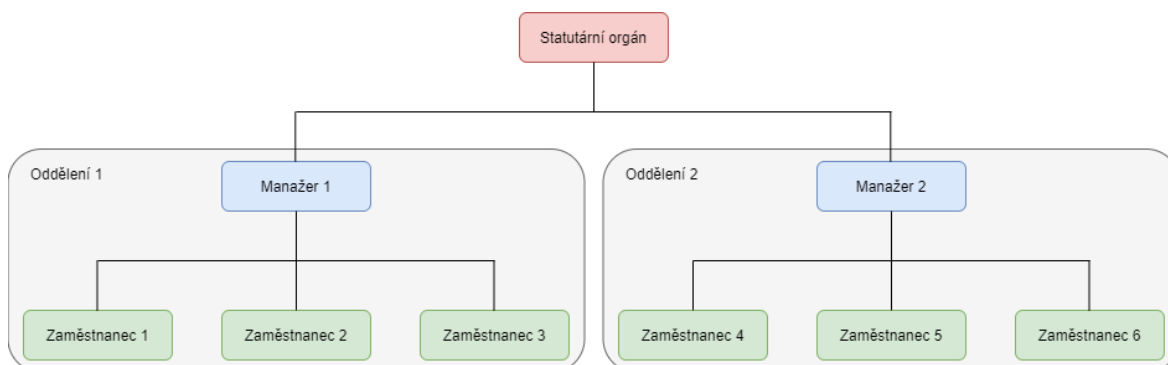
Návrh aplikace zahrnuje sběr funkčních a nefunkčních požadavků, a analýzu – tedy modelování případů užití, diagram tříd, E-R diagram a drátěné modely.

3.1 Funkční požadavky

Funkční požadavky popisují chtěné funkcionality systému. Obecně se rozdělují [21] na tzv. uživatelské a systémové požadavky.

3.1.1 Uživatelské požadavky

Vyvíjená aplikace pro evidenci smluv je určena k využití organizací či firmě používající organizační strukturu. Pro tento projekt je uvažována liniová řídicí struktura podniku, kde vystupuje statutární orgán (dále řečeni pověřeni podepisovatelé), manažeři a zaměstnanci, respektive obchodníci. (Obrázek 6) Je tedy třeba implementovat systém registrace a přihlášení, systém rolí a oddělení.



Obrázek 6: Liniová organizační struktura firmy

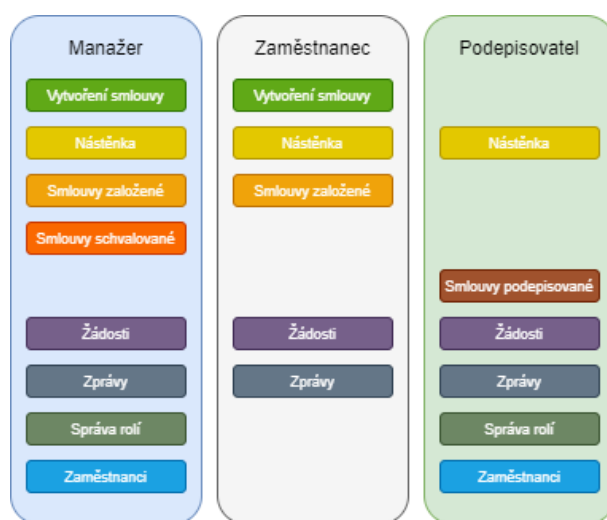
V aplikaci bude existovat také funkce „Zaměstnanci“, do které budou mít přístup pouze Podepisovatelé a Manažeři. Podepisovatelům tato funkce zobrazí všechny Manažery a Zaměstnance s údaji o jejich výsledcích – počet aktivních smluv, počet smluv čekajících na podpis, celkový počet smluv. Manažerům se zobrazí stejné informace, avšak pouze o zaměstnancích na jejich oddělení. Podepisovatelé i Manažeři budou moci upravovat přiřazené oddělení relevantních Manažerů nebo Zaměstnanců.

Po přihlášení bude uživateli zobrazena domovská stránka programu – Nástěnka. Nástěnka bude obsahovat relevantní informace pro přihlášeného uživatele v závislosti na jeho roli. Budou to informace o jeho smlouvách – počty smluv v jednotlivých fázích životního cyklu a počty smluv v jednotlivých typech. Dále také bude obsahovat informace jako je počet

nových zpráv a počet nových žádostí. Aplikace tedy bude obsahovat i funkce pro zasílání zpráv a žádostí.

Na všech zobrazeních v aplikaci se budou nacházet tlačítka pro navigaci v programu, což bude dosaženo využitím vertikální navigační lišty. Konkrétně to budou tlačítka „Nástěnka, Smlouvy (Založené, Podepisované, Schvalované – dle role přihlášeného uživatele), Žádosti, Zprávy, Správa rolí a Zaměstnanci“. V záhlaví stránky bude tlačítko pro založení nové smlouvy, dále tlačítko pro zobrazení informací o přihlášeném uživateli a tlačítko pro odhlášení uživatele.

Jak již bylo zmíněno, v systému bude použit systém rolí. Konkrétně to budou role „Manažer“, „Zaměstnanec“ a „Podepisovatel“. Každá z těchto rolí bude mít odlišný přístup k jednotlivým funkcím aplikace. Jednotlivé přístupy k funkcím jsou předvedeny na následujícím obrázku. (Obrázek 7)



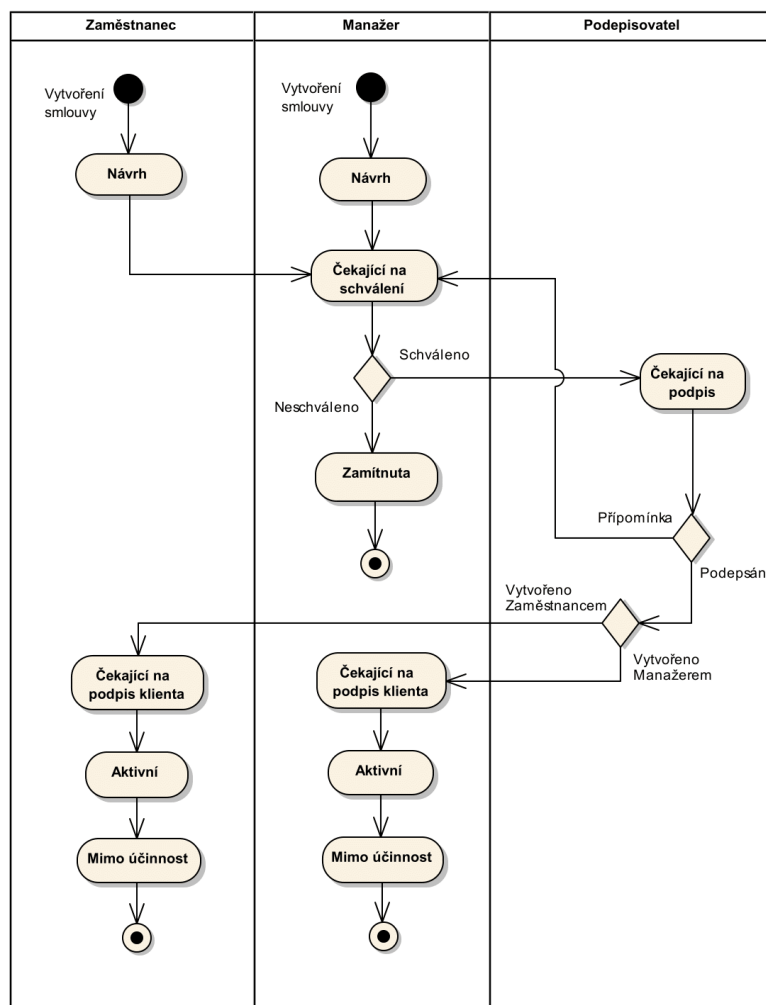
Obrázek 7: Přístupy k funkcím aplikace dle rolí uživatele

Po registraci uživatel nenabývá žádné role. Roli mu může přiřadit pouze Manažer nebo Podepisovatel, a to jakoukoliv ze zmíněných třech rolí. Manažer může přiřazovat uživatele do role Zaměstnanec. Podepisovatel může přiřazovat uživatele do rolí Podepisovatel, Manažer, Zaměstnanec.

V aplikaci bude možno vytvořit smlouvu podle šablony. Zobrazení pro vytvoření smlouvy bude obsahovat formulář s možnostmi vyplnění atributů smlouvy, jako Typ smlouvy, Název smlouvy, Popis smlouvy, Způsob platby, Počet měsíců účinnosti a informace o fyzické nebo právnické osobě zákazníka. V případě fyzické osoby to budou následující informace: Jméno a příjmení, Bydliště, Číslo občanského průkazu a Datum narození. V případě právnické

osoby to budou: Název společnosti, Sídlo, Identifikační číslo a Daňové identifikační číslo. Atributy smlouvy budou uloženy v databázovém systému.

Nově vytvořená smlouva je zařazena do životního cyklu smlouvy, který bude představovat následující fáze – Návrh, Čekající na schválení, Čekající na podpis, Čekající na podpis klienta, Zamítnuta, Aktivní a Mimo účinnost. (Obrázek 8) Možnost upravovat, či exportovat smlouvu bude záviset na fázi, ve které se smlouva momentálně nachází.



Obrázek 8: Životní cyklus smlouvy

Po vytvoření smlouvy se smlouva nachází ve fázi „Návrh“. V této fázi má Zaměstnanec možnost stažení smlouvy ve formátu DOCX, který bude mít určitý předvyplněný vzor dle svých zadaných atributů smlouvy. Smlouvu Zaměstnanec může doplnit, uložit, odeslat, a přesunout do stavu „Čekající na schválení“. Před odesláním však ještě bude muset nahrát ke smlouvě prokázání totožnosti. Pro fyzickou osobu je to občanský průkaz, pro právnickou osobu to jsou výpis z obchodního rejstříku a osvědčení o DPH. Ve stavu „Návrh“ také může

zaměstnanec atributy smlouvy dodatečně měnit. Ve stavu „Čekající na schválení“ může atributy dodatečně měnit i Manažer.

Po přesunutí do stavu „Čekající na schválení“ se odešle žádost Manažerovi dané smlouvy, a ten smlouvu může schválit nebo zamítnout. Také má přístup ke smlouvě ve formátu DOCX a k nahrání upravené smlouvy. Manažer tedy může případně smlouvu měnit. Pokud smlouvu schválí, přejde do stavu „Čekající na podpis“ a Podepisovateli smlouvy se odešle žádost o podpis. Pokud smlouvu Manažer zamítne, smlouva přejde do stavu „Zamítnuta“ a Zaměstnanci, který smlouvu vytvořil, se pošle zpráva s daným oznámením.

Každý typ smlouvy má svého výchozího Podepisovatele, přičemž všechny typy smlouvy však mohou mít stejného Podepisovatele. Výchozí Podepisovatel může smlouvu určenou k podpisu stáhnout ve formátu PDF, podepsat (tedy nahrát podepsanou smlouvu), přidat další podepisovatele, nebo vrátit smlouvu zpět Manažerovi s odůvodněním – zprávou. Tento výchozí Podepisovatel má pak právo odeslat podepsanou smlouvu Zaměstnanci. Takto tedy smlouva přejde do stavu „Čekající na podpis klienta“ a danému Zaměstnanci přijde zpráva s daným oznámením.

Zaměstnanec může ve stavu smlouvy „Čekající na podpis klienta“ nahrát podpis klienta a předat smlouvu do stavu „Aktivní“.

3.1.2 Systémové požadavky

Pro další zpracování [21] je potřeba dané uživatelské požadavky přeložit do formy tzv. systémových požadavků. Tyto se liší tím, že obsahují podrobný popis dané funkcionality, dále se berou v úvahu určitá omezení a definuje se také celkový rozsah realizace dané funkce.

Na dalších stranách přikládám tabulkové zpracování zmíněných uživatelských požadavků přeložených do systémových požadavků.

Tabulka 1: Požadavek Registrace

Zkratka:	P001
Název:	Registrace
Popis:	Po zapnutí aplikace bude uživatel vyzván k registraci. Registrace uživatele bude zahrnovat zapsání jeho e-mailové adresy, jména, příjmení, uživatelského jména a hesla. Po

	stisknutí tlačítka „Registrovat“ budou tyto atributy budou uloženy do databáze. Heslo bude uloženo šifrovaně.
Omezení:	E-mail bude potřeba zadat v daném formátu x@x.x. Je potřeba pro heslo zvolit vhodné šifrování. Validace vstupu.
Očekávaný výstup:	Úspěšná registrace: Uživatel bude registrován v systému. Neúspěšná registrace: Uživateli se zobrazí chybová hláška o špatném zadání dat v registraci.

Tabulka 2: Požadavek Přihlášení

Zkratka:	P002
Název:	Přihlášení
Popis:	Po zapnutí aplikace bude uživatel vyzván k přihlášení. Přihlášení bude složeno z dvou polí pro text – uživatelské jméno a heslo, přičemž heslo bude mít nastaveno pole na „password“, tedy text hesla nebude při zapisování uživatel vidět. Dále bude existovat tlačítko „Přihlásit se“, které na back-endu ověří, zda daný uživatel v systému existuje, a pokud ano, ověří zadané heslo, a přihlásí uživatele do systému.
Omezení:	-
Očekávaný výstup:	Úspěšné přihlášení: Uživatel bude přihlášen v systému, zobrazí se mu stránka „Nástěnka“. Neúspěšné přihlášení: Uživateli se zobrazí chybová hláška o špatném zadání dat v přihlášení.

Tabulka 3: Požadavek Navigace

Zkratka:	P003
Název:	Navigace

Popis:	Aplikace bude obsahovat navigaci mezi jednotlivými funkcemi. Konkrétně jsou to funkce „Vytvoření smlouvy, Nastěnka, Smlouvy (Založené, Schvalované, Podepisované), Žádosti, Zprávy, Správa rolí, Zaměstnanci, Informace o uživateli“.
Omezení:	Přístup k funkcím bude omezen v závislosti na roli uživatele.
Očekávaný výstup:	Uživatel se může pohybovat mezi danými funkcemi aplikace specifických pro jeho roli.

Tabulka 4: Požadavek Vytvoření smlouvy

Zkratka:	P004
Název:	Vytvoření smlouvy
Popis:	Tato stránka bude obsahovat formulář, jehož data se odešlou k uložení na server. Po odeslání se také danými daty naplní šablona smlouvy.
Omezení:	Je potřeba zvolit vhodný formát dokumentu. (DOCX) Validace dat.
Očekávaný výstup:	Uživatel bude schopen vytvořit smlouvu v databázi.

Tabulka 5: Požadavek Detail smlouvy

Zkratka:	P005
Název:	Detail smlouvy
Popis:	Tato stránka bude obsahovat informace o smlouvě, možnost stáhnutí smlouvy ze serveru ve formátu DOCX nebo PDF v závislosti na její fázi životního cyklu a roli uživatele. Také bude umožňovat odkazy na nahrání prokázání totožnosti pro fyzickou či právnickou osobu. Uživateli v roli podepisovatele pak budou zobrazeny tlačítka pro přidání či odebrání dalších podepisovatelů.

Omezení:	-
Očekávaný výstup:	Uživatel je schopen si zobrazit detail smlouvy.

Tabulka 6: Požadavek Nahrání souborů

Zkratka:	P006
Název:	Nahrání souborů
Popis:	Nahrání souborů formátů DOCX či PDF. Jedná se o nahrání upravené smlouvy, nahrání prokázání totožnosti a nahrání podpisů ke smlouvě.
Omezení:	Je potřeba uvažovat nahrání pouze správného formátu a omezení velikosti nahrávaného souboru.
Očekávaný výstup:	Uživatel je schopen nahrávat relevantní soubory DOCX a PDF na server.

Tabulka 7: Požadavek Nástěnka

Zkratka:	P007
Název:	Nástěnka
Popis:	Tato stránka bude obsahovat informace o smlouvách relevantních pro daného uživatele. Bude obsahovat informaci o nových zprávách a žádostech.
Omezení:	Nástěnka bude mít více zobrazení, každé pro jistou roli uživatele.
Očekávaný výstup:	Uživatel je schopen si zobrazit svoji relevantní nástěnku.

Tabulka 8: Požadavek Zobrazení smluv

Zkratka:	P008
Název:	Zobrazení smluv

Popis:	Zobrazení relevantních smluv pro uživatele daných rolí uživatele.
Omezení:	Smlouvy budou mít více zobrazení, konkrétně „Založené, Schvalované a Podepisované“.
Očekávaný výstup:	Uživatel je schopen si zobrazit své relevantní smlouvy.

Tabulka 9: Požadavek Žádosti

Zkratka:	P009
Název:	Žádosti
Popis:	Uživatelům budou chodit žádosti ohledně podpisům ke smlouvám či ke schválení smluv.
Omezení:	-
Očekávaný výstup:	Uživatel je schopen si zobrazit své relevantní žádosti.

Tabulka 10: Požadavek Zprávy

Zkratka:	P010
Název:	Zprávy
Popis:	Funkce určená pro avíza ohledně smluv.
Omezení:	-
Očekávaný výstup:	Uživatel je schopen si zobrazit své relevantní zprávy.

Tabulka 11: Požadavek Správa rolí

Zkratka:	P011
Název:	Správa rolí
Popis:	Uživatelé v rolích „Manažer“ či „Podepisovatel“ mohou měnit role svým podřízeným.

Omezení:	Zobrazení funkce pouze pro uživatele v rolích „Zaměstnanec“ a „Podepisovatel“.
Očekávaný výstup:	Uživatel je schopen spravovat role uživatelů aplikace.

Tabulka 12: Požadavek Přehled zaměstnanců

Zkratka:	P012
Název:	Přehled zaměstnanců
Popis:	Uživatelé v rolích „Manažer“ či „Podepisovatel“ mají přístup k přehledu výkonnosti zaměstnanců ve formě počtů smluv v jednotlivých fázích jejich životního cyklu.
Omezení:	Manažer má přístup pouze k Zaměstnancům svého oddělení. Podepisovatel má přístup ke všem Manažerům i Zaměstnancům.
Očekávaný výstup:	Uživatel je schopen si zobrazit výkonnost zaměstnanců na základě jejich role.

Tabulka 13: Požadavek Úprava oddělení

Zkratka:	P013
Název:	Úprava oddělení
Popis:	Uživatelé v rolích „Manažer“ mohou měnit oddělení svých Zaměstnanců. Uživatelé v rolích „Podepisovatel“ mohou měnit oddělení Zaměstnanců i Manažerů.
Omezení:	Zobrazení pouze pro role „Manažer, Podepisovatel“
Očekávaný výstup:	Uživatel je schopen měnit oddělení uživatelů.

Tabulka 14: Požadavek Životní cyklus smlouvy

Zkratka:	P014
----------	------

Název:	Životní cyklus smlouvy
Popis:	<p>Každá smlouva bude mít přiřazen atribut, který bude značit v jaké fázi životního cyklu se smlouva nachází. Úprava či mazání smlouvy a její exportování do formátu PDF bude na těchto fázích záležet.</p> <p>Tyto fáze jsou: „Návrh, Čekající na schválení, Čekající na podpis, Čekající na podpis klienta, Aktivní, Zamítnuta, Mimo účinnost.</p>
Omezení:	-
Očekávaný výstup:	Smlouvám bude přiřazena fáze životního cyklu.

Tabulka 15: Požadavek Správa smlouvy

Zkratka:	P015
Název:	Správa smlouvy
Popis:	<p>Ve fázi Návrh:</p> <p>Zaměstnanec může měnit atributy smlouvy, nahrávat soubory ke smlouvě a předat smlouvu do stavu „Čekající na schválení“.</p> <p>Ve fázi Čekající na schválení:</p> <p>Manažer může smlouvu zamítnout, předat k podpisu a upravit.</p> <p>Ve fázi Čekající k podpisu:</p> <p>Podpisovatel může smlouvu podepsat, přiřadit další podepisovatele, vrátit smlouvu zpět manažerovi s odůvodněním, předat smlouvu do stavu „Čekající na podpis klienta“.</p> <p>Ve fázi Čekající na podpis klienta:</p> <p>Zaměstnanec může nahrát podpis klienta a předat smlouvu do fáze „Aktivní“.</p>

Omezení:	-
Očekávaný výstup:	Smlouvy je možno spravovat v závislosti na rolích uživatele.

Tabulka 16: Požadavek Systém rolí

Zkratka:	P016
Název:	Systém rolí
Popis:	Systém bude obsahovat systém rolí – Zaměstnanec, Manažer a Podepisovatel.
Omezení:	-
Očekávaný výstup:	Smlouvy je možno spravovat v závislosti na rolích uživatele.

Tabulka 17: Požadavek Zobrazení informací o přihlášeném uživateli

Zkratka:	P017
Název:	Zobrazení informací o přihlášeném uživateli
Popis:	Tato stránka zobrazí přihlášenému uživateli jeho informace, tj. jméno a příjmení, přihlašovací jméno, email, oddělení, role.
Omezení:	-
Očekávaný výstup:	Uživateli je umožněno zobrazit si své informace.

3.2 Nefunkční požadavky

Nefunkční požadavek [22] je omezující podmínka uvalená na daný systém. Mohou to tedy být například výkonové, kapacitní, dostupnostní či zabezpečující požadavky.

Aplikace bude evidovat smlouvy v databázovém systému. Tento systém musí být spolehlivý, vzhledem k tomu, že na něm stojí celá premise programu.

Aplikace bude nabývat určité úrovně grafického zpracování pro správnou navigaci mezi stránkami a pro uživatelsky přívětivý pocit z používání programu. Je tedy potřeba výběru správné ikonografie, použití barev a rozmístění funkcí v programu pro lehkou navigaci.

Dále je potřeba aby byl systém udržitelný, tedy v případě, že v provozní fázi projektu bude potřeba v programu něco změnit, aby se mohla tato změna provést právě v komponentě, kde se nachází, a nebyl kvůli ní třeba brát v úvahu celý software. To má za cíl snížení nákladů na opravu.

Bude také kladen důraz na zabezpečení aplikace, to znamená, že bude k dispozici autentizace a autorizace uživatelů, hesla budou šifrována pomocí hashovací funkce a smlouvy nebude možno v průběhu jejich účinnosti měnit. Nahrávání souborů na server bude omezeno na požadovaný formát a velikost souboru do 5 MB. Komunikace klient-server bude šifrována pomocí protokolu HTTPS s možností nahrání certifikátu SSL po nasazení systému ve firmě.

3.3 Analýza

Záměrem analýzy [22] je tvorba analytického modelu. Tento model se zaměřuje na to, co systém musí udělat, avšak nezabývá se detaily týkajícími se způsobu, jakým to udělá.

V této kapitole je znázorněn model případů užití, diagram tříd, E-R diagram a drátěné modely.

3.3.1 Modelování případů užití

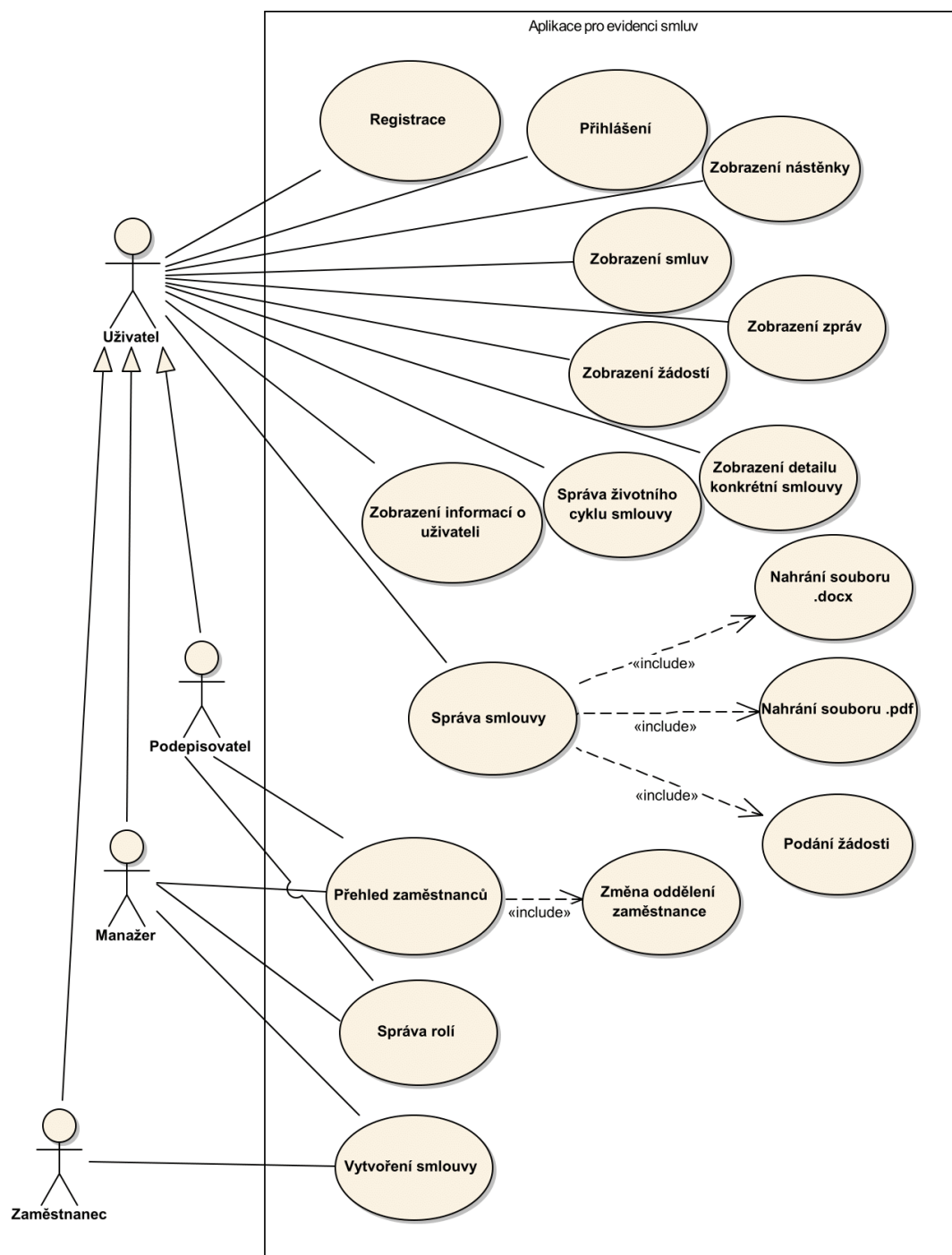
Modelování případů užití [22] je jednou z forem inženýrství požadavků. Má za úkol najít hranice systému, vyhledat aktéry, případy užití a relace mezi aktéry.

V diagramu případů užití (Obrázek 9) figurují čtyři aktéři vycházející z požadavků na software. Jsou jimi Uživatel, Manažer, Zaměstnanec a Podepisovatel.

Uživatel zahrnuje případy užití Registrace, Přihlášení, Zobrazení nástěnky, Zobrazení smluv, Zobrazení zpráv, Zobrazení žádostí, Zobrazení detailu konkrétní smlouvy, Správa životního cyklu smlouvy, Zobrazení informací o uživateli a případ užití Správa smlouvy, který dále obsahuje případy užití Nahrání souboru DOCX, Nahrání souboru PDF a Podání žádosti. Případy užití aktéra Uživatel pak dědí všichni ostatní aktéři.

Aktér Podepisovatel pak zahrnuje případy užití Správa rolí a Přehled zaměstnanců, který obsahuje i případ užití Změna oddělení zaměstnance. Aktér Manažer zahrnuje případy užití

Správa rolí a Vytvoření smlouvy. Aktér Zaměstnanec pak pouze případ užití Vytvoření smlouvy.



Obrázek 9: UML diagram případů užití

Ke každému případu užití je uveden také jeho takzvaný scénář případu užití. Tyto scénáře představují seznam kroků, které podrobně definují interakci mezi aktéry a systémem. Jsou přiloženy na dalších stranách v tabulkovém zpracování.

Tabulka 18: Scénář případu užití Registrace

ID: UC001		
Název: Registrace		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
Vstupní podmínky:		
Žádné		
Výstupní podmínky:		
Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Systém	Zobrazí zobrazení pro registraci s formulářem obsahujícím pole pro: Email, Jméno, Příjmení, Přihlašovací jméno a Heslo.
2	Uživatel	Vyplní formulář pro registraci. Poté stiskne tlačítko pro registraci.
3	Systém	Po validní registraci je uživateli zobrazena hláška o úspěšné registraci.
Rozšiřující kroky:		
2a	Uživatel	Odešle neplatný formulář.
2a1	Systém	Zobrazí uživateli hlášku o neplatné registraci.
Zahrnuje požadavky:		
P001	Registrace	

Tabulka 19: Scénář případu užití Přihlášení

ID: UC002

Název: Přihlášení		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
Vstupní podmínky: Žádné		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Systém	Zobrazí zobrazení pro přihlášení s formulářem obsahujícím pole pro přihlašovací jméno a heslo.
2	Uživatel	Vyplní formulář pro přihlášení. Poté stiskne tlačítko pro přihlášení.
3	Systém	Po autentifikaci uživatele je uživateli přihlášen v systému a je přesměrován na stránku „Nástěnka“.
Rozšiřující kroky:		
2a	Uživatel	Odešle neplatný formulář.
2a1	Systém	Zobrazí uživateli hlášku o neplatném přihlášení.
Zahrnuje požadavky:		
P002	Přihlášení	

Tabulka 20: Scénář případu užití Zobrazení nástěnky

ID: UC003	
Název: Zobrazení nástěnky	
Primární aktér: Uživatel	
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec	

Vstupní podmínky:		
Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.		
Výstupní podmínky:		
Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Nástěnka.
2	Systém	Zobrazí uživateli zobrazení Nástěnka. Toto zobrazení se liší v závislosti na roli uživatele.
Zahrnuje požadavky:		
P003	Navigace	
P007	Nástěnka	
P016	Systém rolí	

Tabulka 21: Scénář případu užití Zobrazení smluv

ID: UC004
Název: Zobrazení smluv
Primární aktér: Uživatel
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec
Vstupní podmínky:
Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.
Výstupní podmínky:
Žádné
Hlavní scénář:

Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Smlouvy a vybere požadované smlouvy k zobrazení. (Založené, Schvalované, Podepisované)
2	Systém	Zobrazí uživateli zobrazení pro seznam relevantních smluv.
Zahrnuje požadavky:		
P003	Navigace	
P008	Zobrazení smluv	
P016	Systém rolí	

Tabulka 22: Scénář případu užití Zobrazení žádostí

ID: UC005		
Název: Zobrazení žádostí		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
Vstupní podmínky: Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Žádosti.
2	Systém	Zobrazí uživateli zobrazení pro seznam relevantních žádostí.

Zahrnuje požadavky:	
P003	Navigace
P009	Žádosti
P016	Systém rolí

Tabulka 23: Scénář případu užití Zobrazení zpráv

ID: UC006		
Název: Zobrazení zpráv		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
Vstupní podmínky: Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Zprávy.
2	Systém	Zobrazí uživateli zobrazení pro seznam relevantních zpráv.
Zahrnuje požadavky:		
P003	Navigace	
P010	Zprávy	
P016	Systém rolí	

Tabulka 24: Scénář případu užití Zobrazení konkrétní smlouvy

ID: UC007		
Název: Zobrazení detailu konkrétní smlouvy		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
<p>Vstupní podmínky:</p> <p>Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.</p>		
<p>Výstupní podmínky:</p> <p>Prvky detailu smlouvy jsou zobrazeny v závislosti na fázi životního cyklus smlouvy a roli uživatele.</p>		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Smlouvy a vybere požadované smlouvy k zobrazení. (Založené, Schvalované, Podepisované)
2	Systém	Zobrazí uživateli zobrazení pro seznam relevantních smluv.
3	Uživatel	Vybere smlouvu k zobrazení jejího detailu, může k tomu využít filtr. Klikne na danou smlouvu v seznamu.
4	Systém	Zobrazí zobrazení detailu vybrané smlouvy.
Zahrnuje požadavky:		
P003	Navigace	
P005	Detail smlouvy	
P016	Systém rolí	

Tabulka 25: Scénář případu užití Správa životního cyklu smlouvy

ID: UC008		
Název: Správa životního cyklu smlouvy		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
<p>Vstupní podmínky:</p> <p>Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec.</p>		
<p>Výstupní podmínky:</p> <p>Prvky detailu smlouvy jsou zobrazeny v závislosti na fázi životního cyklu smlouvy a roli uživatele.</p>		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Smlouvy a vybere požadované smlouvy k zobrazení. (Založené, Schvalované, Podepisované)
2	Systém	Zobrazí uživateli zobrazení pro seznam relevantních smluv.
3	Uživatel	Vybere smlouvu k zobrazení jejího detailu, může k tomu využít filtr. Klikne na danou smlouvu v seznamu.
4	Systém	Zobrazí zobrazení detailu vybrané smlouvy.
5	Uživatel	Má možnost v závislosti na fázi životního cyklu smlouvy a role uživatele měnit fázi životního cyklu smlouvy.
Zahrnuje požadavky:		

P003	Navigace
P014	Životní cyklus smlouvy
P016	Systém rolí

Tabulka 26: Scénář případu užití Zobrazení informací o přihlášeném uživateli

ID: UC009		
Název: Zobrazení informací o přihlášeném uživateli		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
Vstupní podmínky: Uživatel je přihlášen.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko s textem svého přihlašovacího jména.
2	Systém	Zobrazí uživateli zobrazení informací o přihlášeném uživateli.
Zahrnuje požadavky:		
P003	Navigace	
P017	Zobrazení informací o přihlášeném uživateli	

Tabulka 27: Scénář případu užití Správa smlouvy

ID: UC010

Název: Správa smlouvy		
Primární aktér: Uživatel		
Vedlejší aktéři: Podepisovatel, Manažer, Zaměstnanec		
<p>Vstupní podmínky:</p> <p>Uživatel je přihlášen a nabývá jednu z rolí Podepisovatel, Manažer nebo Zaměstnanec. Uživatel se nachází na zobrazení detailu smlouvy.</p>		
<p>Výstupní podmínky:</p> <p>Žádné</p>		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Systém	Dle role uživatele a fáze životního cyklu jsou uživateli zobrazeny možnosti pro správu smlouvy.
2	Uživatel	Klikne na jednu z možností správy smlouvy.
Rozšiřující kroky:		
2a	Uživatel	Klikne na možnost Nahrát soubor DOCX
2a1	Systém	Zobrazí zobrazení pro nahrání souboru DOCX
2a2	Uživatel	Nahraje soubor DOCX
2a3	Systém	Zobrazí hlášku o úspěšném/neúspěšném nahrání souboru.
2b	Uživatel	Klikne na možnost Nahrát soubor PDF
2b1	Systém	Zobrazí zobrazení pro nahrání souboru PDF
2b2	Uživatel	Nahraje soubor PDF
2b3	Systém	Zobrazí hlášku o úspěšném/neúspěšném nahrání souboru.
2c	Uživatel	Klikne na možnost Úpravy smlouvy.

2c1	Systém	Zobrazí zobrazení s formulářem pro úpravu existující smlouvy.
2c2	Uživatel	Vyplní a odešle formulář.
2c3	Systém	Zobrazí hlášku o úspěšném/neúspěšném uložení úprav smlouvy.
Zahrnuje požadavky:		
P005	Detail smlouvy	
P015	Správa smlouvy	
P016	Systém rolí	

Tabulka 28: Scénář případu užití Přehled zaměstnanců

ID: UC011		
Název: Přehled zaměstnanců		
Primární aktér: Podepisovatel, Manažer		
Vedlejší aktéři: Žádní		
Vstupní podmínky: Uživatel je přihlášen a je v jedné z rolí Podepisovatel či Manažer.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Zaměstnanci.
2	Systém	Zobrazí uživateli zobrazení Zaměstnanci s přehledem všech relevantních zaměstnanců pro danou roli uživatele.
Rozšiřující kroky:		

2a	Uživatel	Klikne na libovolného zaměstnance v přehledu zaměstnanců.
2a1	Systém	Zobrazí uživateli detail daného zaměstnance.
Zahrnuje požadavky:		
P003	Navigace	
P012	Přehled zaměstnanců	
P016	Systém rolí	

Tabulka 29: Scénář případu užití Změna oddělení zaměstnance

ID: UC012		
Název: Změna oddělení zaměstnance		
Primární aktér: Podepisovatel, Manažer		
Vedlejší aktéři: Žádní		
Vstupní podmínky: Uživatel je přihlášen a je v jedné z rolí Podepisovatel či Manažer. Uživatel se nachází na detailu Zaměstnance.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne na tlačítko Upravit oddělení.
2	Systém	Zobrazí uživateli zobrazení s formulářem pro úpravu oddělení.
3	Uživatel	Vybere oddělení a odešle formulář.
4	Systém	Zobrazí hlášku o úspěšné změně oddělení.
Zahrnuje požadavky:		

P012	Přehled zaměstnanců
P013	Úprava oddělení
P016	Systém rolí

Tabulka 30: Scénář případu užití Správa rolí

ID: UC013		
Název: Správa rolí		
Primární aktér: Podepisovatel, Manažer		
Vedlejší aktéři: Žádní		
Vstupní podmínky: Uživatel je přihlášen a je v jedné z rolí Podepisovatel či Manažer.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne v navigační liště na tlačítko Správa rolí.
2	Systém	Zobrazí uživateli zobrazení pro Správu rolí s obsahem relevantním pro danou roli uživatele.
3	Uživatel	Vybere roli, do které/ze které chce přidat/odebrat uživatele. Stiskne tlačítko Upravit.
4	Systém	Zobrazí detail vybrané role.
5	Uživatel	Stiskne tlačítko pro přidání/odebrání uživatelů.
6	Systém	Zobrazí zobrazení pro přidání/odebrání uživatelů.
7	Uživatel	Vybere uživatele, kterého roli chce upravit, má možnost ji upravit a uložit tlačítkem Uložit.
8	Systém	Vypíše hlášku o úspěšném uložení role.

Zahrnuje požadavky:	
P003	Navigace
P011	Správa rolí
P016	Systém rolí

Tabulka 31: Scénář případu užití Vytvoření smlouvy

ID: UC014		
Název: Vytvoření smlouvy		
Primární aktér: Podepisovatel, Manažer		
Vedlejší aktéři: Žádní		
Vstupní podmínky: Uživatel je přihlášen a je v jedné z rolí Zaměstnanec či Manažer.		
Výstupní podmínky: Žádné		
Hlavní scénář:		
Krok	Aktér/Systém	Popis
1	Uživatel	Klikne na tlačítko Nová smlouva
2	Systém	Zobrazí uživateli zobrazení s formulářem pro vytvoření nové smlouvy.
3	Uživatel	Vyplní a odešle formulář.
4	Systém	Zobrazí hlášku o úspěšném/neúspěšném vytvoření smlouvy.
Zahrnuje požadavky:		
P003	Navigace	
P004	Vytvoření smlouvy	

3.3.2 E-R diagram

Návrh databáze následuje po sběru a analýze požadavků. Datový model z nich totiž přímo vychází. Pro realizaci databáze je třeba vytvořit E-R, tedy entitně-relační, diagram. Tento diagram slouží pro jasnou identifikaci jednotlivých entit a rozložení jejich atributů.

Je vytvořeno celkem čtrnáct entit, které se vážou na diagram tříd, obsahující relevantní atributy. Jsou jimi: Contract, Lifecycle, Request, Request_type, Contract_type, PaymentOptions, Signatures, Changes, Changes_type, AspNetUsers, AspnetUserRoles, AspNetRoles, Department, a Message.

Tabulka Contract shrnuje všechny potřebné informace o dané smlouvě, existují relace mezi touto tabulkou a tabulkami Lifecycle, Request, Contract_type, PaymentOptions, Signatures ve vztahu 1:N.

Vazební tabulkou je pak AspNetUserRoles pro zajištění vztahu M:N mezi tabulkami AspNetUsers a AspNetRoles. Tabulka Signatures pak také hraje roli vazební tabulky pro entity AspNetUsers a Contract. Zajišťuje tak možnost mít na více smlouvách více podepisovatelů. Stejně tak je vazební tabulkou i tabulka Message, která má relaci 1:N vůči tabulce Contract a relaci 1:0..N vůči tabulce AspNetUsers. Takto zajišťuje možnost zaslání zprávy všem, nebo pouze několika uživatelům v závislosti na druhu zprávy a jasnou identifikaci smlouvy, které se tato zpráva týká. Vazebními tabulkami jsou také tabulky Request a Changes umožňující mít více druhů změn a více druhů žádostí na jedné smlouvě.

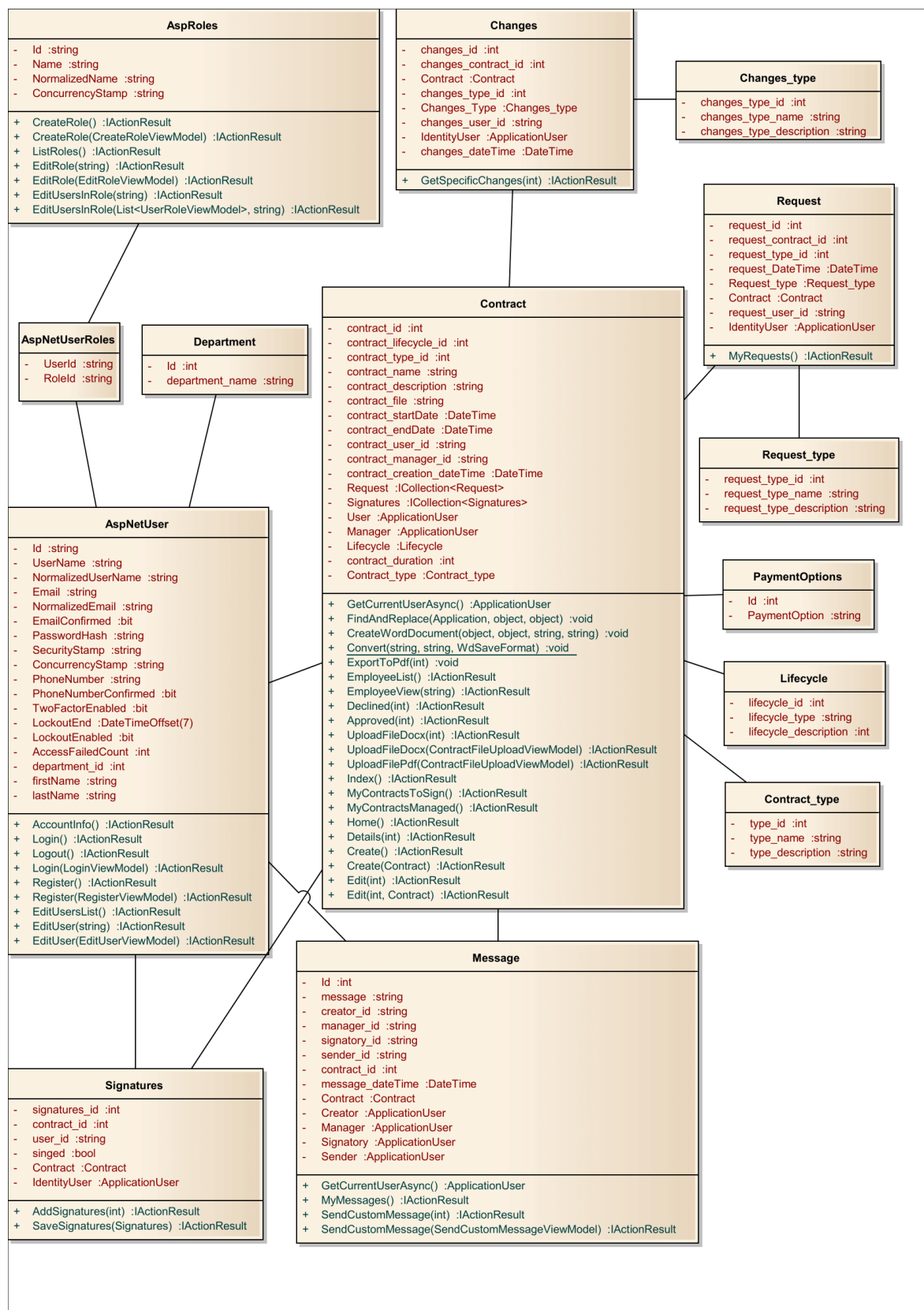
Diagram viz Příloha P I: E-R diagram.

3.3.3 Diagram tříd

Třída [23] se dá definovat jako deskriptor množiny objektů, které sdílejí stejné atributy, operace, metody, relace a chování.

Diagram tříd (Obrázek 11) přímo vychází z navrženého E-R diagramu. Obsahuje navržené atributy a funkce každé entity. Navržení atributů dané entity je zároveň navržení Modelu v návrhovém vzoru MVC. Funkce entity jsou pak vlastnosti Controlleru daného Modelu v návrhovém vzoru MVC.

Jednotlivé funkce jsou navrženy s atributem [HttpGet] a [HttpPost], proto na diagramu tříd jsou některé funkce napsány dvakrát, s odlišnými parametry. Jedna funkce má totiž atribut [HttpGet] a vrací view, další s atributem [HttpPost] se stará o zpracování dat získaných z daného view.



Obrázek 10: UML diagram tříd

3.3.4 Drátěné modely

Při navrhování webové aplikace je důležité si rozmyslet také její vizuální podobu z důvodu pozdější rychlé implementace. Je využito drátěných modelů zachycujících návrh jednotlivých zobrazení.

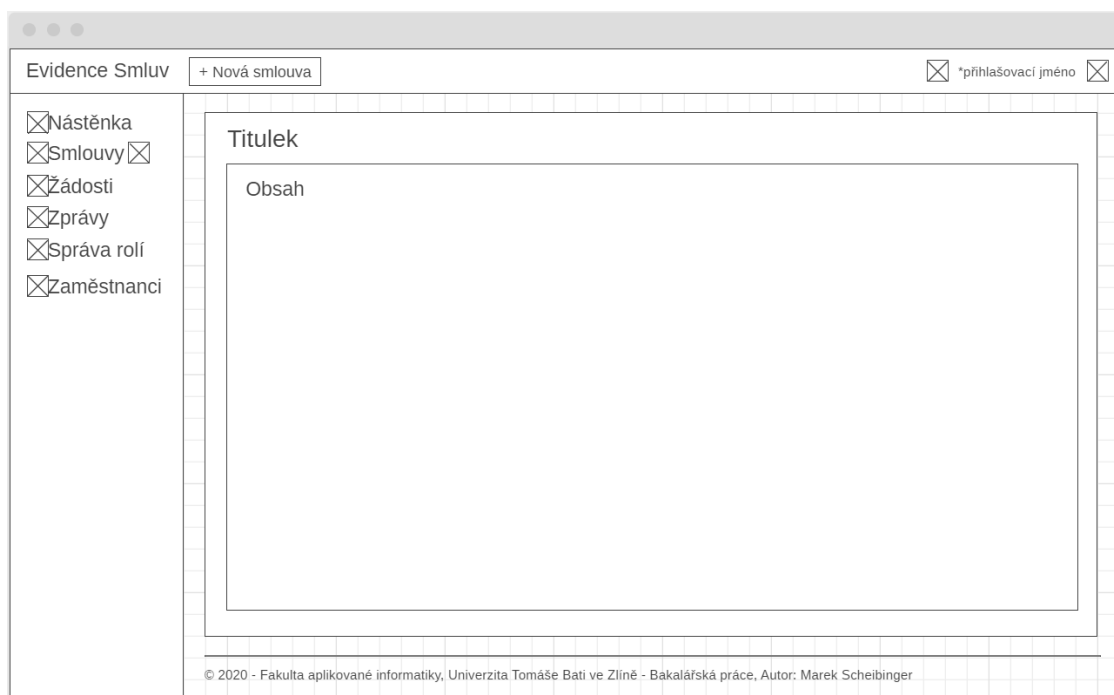
Po zapnutí aplikace je uživateli zobrazena stránka pro přihlášení s možností registrace. (Obrázek 12)

The image shows a wireframe of a web form titled "Evidence smluv". The form is centered on a grid background. It contains two input fields: "Přihlašovací jméno" (Login name) and "Heslo" (Password). Below these fields are two buttons: "Přihlásit" (Login) and "Registrovat" (Register). At the bottom of the form, there is a copyright notice: "©2020 Marek Scheibinger, FAI UTB".

Obrázek 11: Zobrazení "Přihlášení"

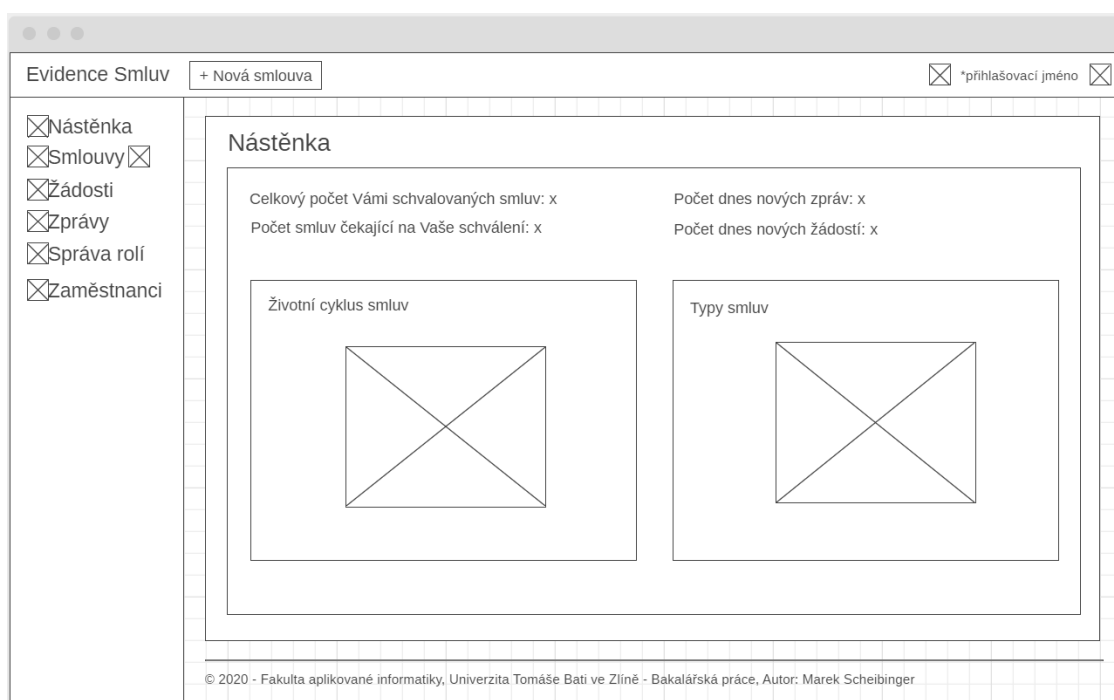
Zobrazení Registrace je stejné jako Přihlášení s výjimkou přidání dalších třech polí pro zadání emailu, jména a příjmení.

Po úspěšném přihlášení je uživatel přesměrován do aplikace, které obsah v navigační liště se liší v závislosti na rolích uživatele. Pro ukázkou je přiloženo několik drátěných modelů uživatele v roli Manažer, z důvodu jeho přístupu ke většině funkcí aplikace. (Obrázek 13)



Obrázek 12: Zobrazení aplikace pro uživatele v roli Manažer

Z přihlášení je uživatel defaultně odkázán na zobrazení „Nástěnka“, které obsahuje informace o celkovém počtu smluv schvalovaných daným manažerem, počet smluv čekající na schválení, počet nových zpráv a počet nových žádostí. Zároveň obsahuje grafy pro vizuální zobrazení počtu smluv v jednotlivých fázích životního cyklu a v jednotlivých smluvních typech. (Obrázek 14)



Obrázek 13: Zobrazení "Nástěnka"

Dalším zobrazením je zobrazení pro vytvoření smlouvy. Obsahuje formulář pro vyplnění požadovaných atributů smlouvy. Uživatel si může vybrat mezi fyzickou a právnickou osobou. (Obrázek 15)

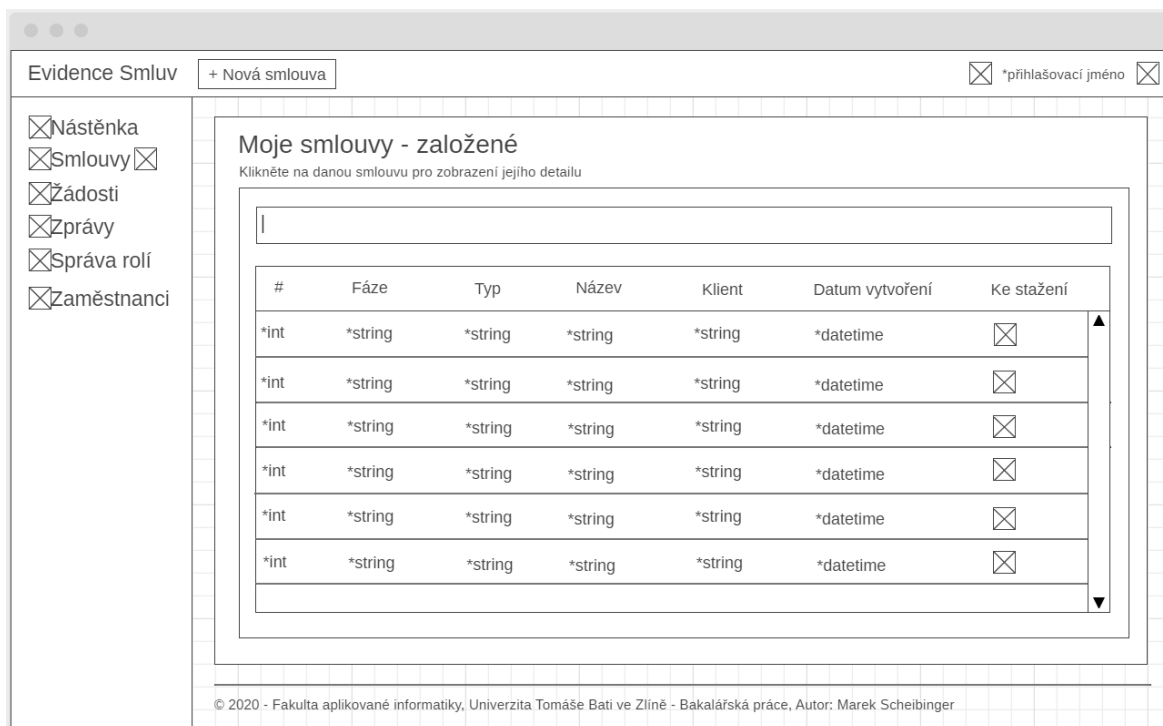
The screenshot shows a web application window titled 'Evidence Smluv'. On the left is a sidebar with navigation links: 'Nástěnka', 'Smlouvy', 'Žádosti', 'Zprávy', 'Správa rolí', and 'Zaměstnanci'. The main area is titled 'Vytvoření smlouvy'. It contains the following fields and controls:

- 'Typ smlouvy': A dropdown menu.
- 'Způsob platby': A dropdown menu.
- Two radio buttons for entity selection: 'Fyzická osoba' and 'Právnická osoba'.
- Under 'Fyzická osoba':
 - 'Jméno a příjmení': Text input.
 - 'Bydliště': Text input.
 - 'Číslo OP': Text input.
 - 'Datum narození': Text input.
- Under 'Právnická osoba':
 - 'Název společnosti': Text input.
 - 'Sídlo': Text input.
 - 'IČ': Text input.
 - 'DIČ': Text input.
- 'Název smlouvy': Text input.
- 'Popis smlouvy': Text input.
- 'Vytvořit': A button at the bottom.

At the bottom of the window, a footer reads: '© 2020 - Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně - Bakalářská práce, Autor: Marek Scheibinger'.

Obrázek 14: Zobrazení Vytvoření smlouvy

Smlouvy si uživatel může zobrazit v tabulkové formě. Toto zobrazení je stejné pro všechny varianty smluv s výjimkou nadpisu. Jsou to tedy zobrazení Moje smlouvy – založené, schvalované a podepisované. (Obrázek 16) Tato obsahují pole pro vstup pro filtr smluv a samotnou tabulku relevantních smluv. Zobrazení Žádosti a Zprávy jsou pak na stejném principu. Tedy nadpis a tabulka s relevantními daty s rozdílem v obsahu sloupců, jmenovitě jsou to: název smlouvy, datum a čas podání žádosti, typ žádosti pro zobrazení Žádosti, a název smlouvy, zpráva, odesílatel a datum odeslání zprávy pro zobrazení Zprávy. Implicitní drátěné modely pro tato zobrazení z důvodu redundance nejsou přiloženy.



Obrázek 15: Zobrazení "Moje smlouvy – založené"

Po kliknutí uživatelem na libovolnou smlouvu se zobrazí zobrazení detailu dané smlouvy. Prvky tohoto zobrazení se liší v závislosti na fázi životního cyklu, ve které se momentálně daná smlouva nachází a na roli uživatele, který si smlouvu zobrazil. V příkladu (Obrázek 17) je zobrazení detailu právě vytvořené smlouvy. Tvůrce smlouvy má možnost stáhnout šablonu smlouvy, nahrát prokázání totožnosti, upravit atributy smlouvy, nahrát upravenou smlouvu a předat smlouvu ke schválení manažerovi, respektive předat smlouvu do druhé fáze jejího životního cyklu.

The screenshot shows a web application titled "Evidence Smluv". In the top right corner, there is a login field labeled "*přihlašovací jméno" with a password icon. The left sidebar contains a menu with the following items, each with a checkbox icon: "Nástěnka", "Smlouvy", "Žádosti", "Zprávy", "Správa rolí", and "Zaměstnanci". The main content area is titled "Detail smlouvy" and is divided into two columns of form fields. The left column contains: "Název smlouvy" (*string), "Popis smlouvy" (*string), "Typ smlouvy" (*string), "Fáze smlouvy" (*string), "Smlouva byla vytvořena" (*string dne *datetime), and "Ke stažení" (with a download icon). The right column contains: "Jméno a příjmení" (*string), "Bydliště" (*string), "Číslo OP" (*string), "Datum narození" (*date), "Způsob platby" (*string), and "Smlouvu schvaluje" (*string). Below these columns are two buttons: "Nahrát .docx" and "Nahrát prokázání totožnosti". At the bottom of the main area is a button labeled "Historie změn". A separate section at the very bottom is titled "Chybí prokázání totožnosti" and contains an "Upravit" button. The footer of the application reads: "© 2020 - Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně - Bakalářská práce, Autor: Marek Scheibinger".

Obrázek 16: Zobrazení detailu smlouvy po jejím vytvoření

Z detailu smlouvy vede více cest k dalším zobrazením. Tlačítka pro nahrání dokumentů zobrazí uživateli zobrazení obsahující formuláře pro nahrání daných dokumentů. Tlačítko „Historie změn“ převede uživatele na stejnojmenné zobrazení, kde je zobrazen seznam změn na smlouvě v tabulkové formě. Tato tabulka zahrnuje sloupce: provedeno uživatelem, typ změny, datum a čas změny. Tlačítko „Upravit“ převede uživatele na zobrazení pro upravení smlouvy obsahující stejný formulář jako na zobrazení Vytvoření smlouvy. Tento formulář je však již předvyplněn existujícími daty, uživatel může data změnit a uložit či se vrátit na detail prostřednictvím tlačítka Zpět na detail. Zobrazení pro správu rolí je specifické pro uživatelské role Manažer a Podepisovatel. Z tohoto zobrazení může uživatel přejít na zobrazení pro editaci role, respektive přidání/odebrání uživatelů do/z role. Z důvodu redundance nejsou tyto implicitní drátěné modely přiloženy.

4 IMPLEMENTACE

4.1 Vytvoření architektury

Vytvářením architektury je myšleno vytvoření databáze, napojení projektu na databázi a vytvoření relevantních tříd/entit/modelů s využitím návrhového vzoru MVC.

Jak bylo již v této práci zmíněno, při vytváření databáze je využito přístupu EF Code-First. Jsou vytvořeny třídy, respektive entity, které jsou zároveň Modelem v návrhovém vzoru MVC.

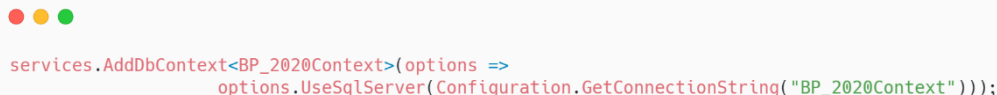


```
public class Contract
{
    [Key]
    public int contract_id { get; set; }
```

Obrázek 17: Ukázka vytvoření modelu Contract

Modely jsou naplněny atributy navrženými v E-R diagramu. Jsou vytvořeny všechny entity, kromě entit AspNetUser, AspNetRole a AspNetUserRoles. Důvod je vysvětlen později. Následně je potřeba definovat využívanou databázi, kam se provede migrace. Tato konfigurace se provádí v součásti appsettings.json, v sekci „ConnectionStrings“, kde se vytvoří DbContext.

Následně je potřeba přidat tento DbContext do projektu – to se provede v součásti Startup.cs, kde se ve funkci ConfigureServices(IServiceCollection services) přidá následující kód. (Obrázek 19) Název „BP_2020Context“ je použit z důvodu názvu aplikace v systému – BP_2020.



```
services.AddDbContext<BP_2020Context>(options =>
    options.UseSqlServer(Configuration.GetConnectionString("BP_2020Context"));
```

Obrázek 18: Přidání DbContext do projektu

Nakonec je ještě potřeba vytvoření třídy spravující vytvořený DbContext. (Obrázek 20)



```
public class BP_2020Context : IdentityDbContext
{
    public BP_2020Context (DbContextOptions<BP_2020Context> options)
        : base(options)
    {
    }
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
    }
}
```

Obrázek 19: Vytvoření třídy spravující DbContext

Tato třída může dědit ze třídy „DbContext“ nebo také „IdentityDbContext“. Výhodou využití druhé zmíněné třídy je automatické vytvoření tříd pro správu uživatelů a rolí v databázi. Automaticky se tak využívá již vytvořených funkcí definovaných v knihovně Microsoft.AspNet.Identity, jako je například šifrování a verifikace hesel ve třídě AspNetUser. Jsou tak vytvořeny entity AspNetUser, AspNetRole a AspNetUserRoles.

Následně je potřeba vytvořit kontextové třídy pro každou entitu. (Obrázek 21)



```
public DbSet<BP_2020.Models.Contract> Contract { get; set; }
public DbSet<BP_2020.Models.Lifecycle> Lifecycle { get; set; }
public DbSet<BP_2020.Models.Contract_type> Contract_type { get; set; }
public DbSet<BP_2020.Models.Request> Request { get; set; }
public DbSet<BP_2020.Models.Request_type> Request_type { get; set; }
public DbSet<BP_2020.Models.Signatures> Signatures { get; set; }
public DbSet<BP_2020.Models.Changes> Changes { get; set; }
public DbSet<BP_2020.Models.Changes_type> Changes_type { get; set; }
public DbSet<BP_2020.Models.PaymentOptions> PaymentOptions { get; set; }
public DbSet<BP_2020.Models.Message> Message { get; set; }
public DbSet<BP_2020.Models.Department> Department { get; set; }
```

Obrázek 20: Vytvoření kontextových tříd

Nyní již jen zbývá převést vytvořenou databázi na Microsoft SQL Server. K tomu je využito funkce nabízenou Code-First EF – migrace. Migrace se vytvoří pomocí zadání příkazu „Add-Migration“ do „Package Manager Console“, což je modul nabízený Visual Studiemi. Tím se vygeneruje SQL kód, kterým se následně aktualizuje relevantní databáze na serveru. (Obrázek 22) Pro aplikaci migrace se zadá příkaz „Update-Database“.

```
migrationBuilder.CreateTable(
    name: "Contract",
    columns: table => new
    {
        contract_id = table.Column<int>(nullable: false)
            .Annotation("SqlServer:ValueGenerationStrategy",
                SqlServerValueGenerationStrategy.IdentityColumn),
        contract_lifecycle_id = table.Column<int>(nullable: false),
        contract_type_id = table.Column<int>(nullable: false),
        contract_name = table.Column<string>(nullable: true),
        contract_description = table.Column<string>(nullable: true),
        contract_file = table.Column<string>(nullable: true),
        contract_startDate = table.Column<DateTime>(nullable: false),
        contract_endDate = table.Column<DateTime>(nullable: false),
        contract_user_id = table.Column<string>(nullable: true),
        contract_manager_id = table.Column<string>(nullable: true),
        contract_creation_dateTime = table.Column<DateTime>(nullable: false)
    },
```

Obrázek 21: Ukázka vygenerované migrace

Výsledkem je vytvořená relační databáze na Microsoft SQL Serveru. Pro následné zobrazení správnosti provedené migrace je použit program SQL Server Management Studio.

4.2 Implementace funkcí

Po vytvoření entit, respektive modelů, a připojení na databázi je započata implementace funkcí jednotlivých kontrolerů a zobrazení. Každý model má přiřazený kontroler, skrze kontroler je pak zobrazeno zobrazení uživateli.

```
public class ContractsController : Controller
{
    private readonly BP_2020Context _context;

    private UserManager<IdentityUser> _userManager { get; }
    public IHostingEnvironment HostingEnvironment { get; }
    public Microsoft.Office.Interop.Word.Document wordDocument { get; set; }

    public ContractsController(BP_2020Context context, UserManager<IdentityUser> userManager,
        IHostingEnvironment hostingEnvironment)
    {
        _context = context;
        _userManager = userManager;
        HostingEnvironment = hostingEnvironment;
    }
}
```

Obrázek 22: Ukázka kontroleru ContractsController

Důležité je zmínit, že pro generování dokumentu DOCX po vytvoření smlouvy a export do formátu PDF používám knihovnu „Aspose.Words“, která po nasazení ve firmě požaduje zakoupení licence.

V kontroleru ContractsController je pro ukázkou práce s návrhovým vzorem MVC uvedena funkce UploadFileDocx(), která se stará o nahrání DOCX souboru na server. Tato funkce, jako několik dalších, je rozdělena na [HttpGet] a [HttpPost] funkce.



```
[HttpGet]
public IActionResult UploadFileDocx(int? id)
{
    Contract contract = _context.Contract.Find(id);

    ContractFileUploadViewModel contractFileUploadViewModel = new ContractFileUploadViewModel
    {
        contractId = contract.contract_id
    };
    return View(contractFileUploadViewModel);
}
```

Obrázek 23: Implementace funkce UploadFileDocx(int id) s atributem [HttpGet]

Tato funkce (Obrázek 24) nejdříve najde požadovanou smlouvu, které id je posláno v parametru funkce, následně naplní View Model speciálně vytvořen pro nahrání souboru, vepíše do něj id smlouvy, a daný View Model pošle do zobrazení. Zmiňovaný View Model navíc obsahuje proměnnou contractFile, do které se v zobrazení bude nahrávat soubor.

Zobrazení (view) potom vypadá následovně. (Obrázek 25)

```

@model ContractFileUploadViewModel

@{
    ViewBag.Title = "Nahrát soubor .docx";
}

<div class="card mb-3 shadow p-3 bg-white rounded">
    <div class="card-header">
        <h2>Nahrání upravené smlouvy</h2>
    </div>
    <form enctype="multipart/form-data" asp-controller="Contracts" asp-action="UploadFileDocx">
        <div class="card-body">
            <div class="form-group row">
                <label asp-for="contractId" class="col-sm-2 col-form-label"></label>
                <div class="col-sm-10">
                    <input asp-for="contractId" class="form-control" readonly="readonly" />
                </div>
            </div>
            <div class="form-group row">
                <label asp-for="contractFile" class="col-sm-2 col-form-label"></label>
                <div class="custom-file">
                    <input asp-for="contractFile" class="form-control custom-file-input" accept=".docx"
id="input" />
                    <label class="custom-file-label">Vyberte soubor...</label>
                </div>
            </div>
            @section Scripts {
                <script>
                    $(document).ready(function () {
                        $('custom-file-input').on("change", function () {
                            var fileName = $(this).val().split("\\").pop();
                            $(this).next('custom-file-label').html(fileName);
                        });
                    });
                    const input = document.getElementById('input')

                    input.addEventListener('change', (event) => {
                        const target = event.target
                        if (target.files && target.files[0]) {
                            const maxAllowedSize = 5 * 1024 * 1024;
                            if (target.files[0].size > maxAllowedSize) {
                                target.value = 'Prosím nahrajte dokument do velikosti 5 MB'
                            }
                        }
                    })
                </script>
            }
        </div>
        <div class="card-footer">
            <button type="submit" asp-route-id="@Model.contractId" class="btn btn-
primary">Nahrát</button>
            <a asp-action="Details" asp-route-id="@Model.contractId" class="btn btn-primary">Zpět na
detail</a>
        </div>
        @Html.AntiForgeryToken()
    </form>
</div>

```

Obrázek 24: Zobrazení UploadFileDocx.cshtml

Toto zobrazení je Razor View obsahující HTML jazyk společně s jazykem C#. Speciálně toto zobrazení ještě obsahuje kód Javascriptu, respektive jQuery, pro manipulaci vstupního prvku formuláře, kterým se načítá soubor a pro omezení velikosti nahrávaného souboru.

Po nahrání souboru DOCX a kliknutí uživatele na tlačítko „Nahrát“, se odešlou zadané informace do varianty funkce UploadFileDocx() v kontroleru s atributem [HttpPost]. Kód této funkce je přiložen níže. (Obrázek 26)



```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> UploadFileDocx(ContractFileUploadViewModel model)
{
    var routeValue = model.contractId;
    if (ModelState.IsValid)
    {
        string fileNameDocx = null;
        if (model.contractFile != null)
        {
            string uploadsFolder = Path.Combine(HostingEnvironment.WebRootPath, "files");
            fileNameDocx = model.contractId + ".docx";
            string filePath = Path.Combine(uploadsFolder, fileNameDocx);
            using (var stream = new FileStream(filePath, FileMode.Create))
            {
                model.contractFile.CopyTo(stream);
            }
            ExportToPdfAspose(model.contractId);

            var user = await GetCurrentUserAsync();
            var userId = user?.Id;
            var newFileChangeType = 10; // Odkaz na desátý záznam v databázi v tabulce
            Changes_type, "Nahrán nový soubor .docx"
            Changes change = new Changes
            {
                changes_contract_id = model.contractId,
                changes_user_id = userId,
                changes_type_id = newFileChangeType,
                changes_dateTime = DateTime.Now
            };
            _context.Add(change);
            await _context.SaveChangesAsync();
        }
        return RedirectToAction("Details", new { id = routeValue });
    }
}

```

Obrázek 25: Ukázka funkce UploadFileDocx() s atributem [HttpPost]

Data získaná ze zobrazení se díky ContractFileUploadViewModelu pošlou do této funkce, kdy se nahraje soubor na server, exportuje se do formátu PDF díky další vytvořené funkci ExportToPdfAspose(), nahraje se do databáze informace o provedené změně na smlouvě, a uživatele přesměruje zpět na detail smlouvy, kterou právě upravil.

Stejným způsobem, tedy využitím návrhového vzoru MVC, jsou realizovány i ostatní funkce v kontroleru i ve všech ostatních kontrolerech jednotlivých modelů.

4.3 Implementace zabezpečení

Aplikace je zabezpečena vůči útokům SQL Injection a CSFR, přístup do aplikace je omezen pouze na přihlášené uživatele, uživatelská hesla jsou šifrována a je využito HTTPS šifrování komunikace klient-server.

Jak je uvedeno v teoretické části práce, útoku typu SQL Injection lze zabránit použitím dotazovacího jazyka LINQ, který je součástí programovacího jazyka C# a EF.

Jako příklad využití dotazovacího jazyka LINQ je přiložena následující ukázka funkce pro výpis schvalovaných smluv manažerovi z databáze. (Obrázek 27)



```
public async Task<IActionResult> MyContractsManaged()
{
    var user = await GetCurrentUserAsync();
    var userId = user?.Id;

    var bP_2020Context = _context.Contract
        .Include(c => c.Contract_type)
        .Include(l => l.Lifecycle)
        .Include(u => u.ManagerId)
        .Where(m => m.ManagerId.Id == userId);

    return View(await bP_2020Context.ToListAsync());
}
```

Obrázek 26: Ukázka LINQ

Místo manipulace s řetězci, která je zranitelná vůči útoku SQL Injection, jsou v jazyku LINQ využívány parametry. Tím LINQ zastavuje útok SQL Injection.

Dalším tématem náchylným ke zneužití je ukládání uživatelských hesel. Díky využití knihovny Microsoft.AspNet.Identity jsou všechna hesla před uložením šifrována. Je k tomu využito následující hashovací funkce. (Obrázek 28)



```
public static string HashPassword(string password)
{
    if (password == null)
    {
        throw new ArgumentNullException("password");
    }

    byte[] salt;
    byte[] subkey;
    using (var deriveBytes = new Rfc2898DeriveBytes(password, SaltSize, PBKDF2IterCount))
    {
        salt = deriveBytes.Salt;
        subkey = deriveBytes.GetBytes(PBKDF2SubkeyLength);
    }

    var outputBytes = new byte[1 + SaltSize + PBKDF2SubkeyLength];
    Buffer.BlockCopy(salt, 0, outputBytes, 1, SaltSize);
    Buffer.BlockCopy(subkey, 0, outputBytes, 1 + SaltSize, PBKDF2SubkeyLength);
    return Convert.ToBase64String(outputBytes);
}
```

Obrázek 27: Hashovací funkce knihovny Microsoft.AspNet.Identity

Verifikace hashovaného hesla pak prochází funkcí VerifyHashedPassword(string hashedPassword, string password) následovně. (Obrázek 29)

```
public static bool VerifyHashedPassword(string hashedPassword, string password)
{
    if (hashedPassword == null)
    {
        return false;
    }
    if (password == null)
    {
        throw new ArgumentNullException("password");
    }

    var hashedPasswordBytes = Convert.FromBase64String(hashedPassword);

    if (hashedPasswordBytes.Length != (1 + SaltSize + PBKDF2SubkeyLength) || hashedPasswordBytes[0] !=
    0x00)
    {
        return false;
    }

    var salt = new byte[SaltSize];
    Buffer.BlockCopy(hashedPasswordBytes, 1, salt, 0, SaltSize);
    var storedSubkey = new byte[PBKDF2SubkeyLength];
    Buffer.BlockCopy(hashedPasswordBytes, 1 + SaltSize, storedSubkey, 0, PBKDF2SubkeyLength);

    byte[] generatedSubkey;
    using (var deriveBytes = new Rfc2898DeriveBytes(password, salt, PBKDF2IterCount))
    {
        generatedSubkey = deriveBytes.GetBytes(PBKDF2SubkeyLength);
    }
    return ByteArraysEqual(storedSubkey, generatedSubkey);
}
```

Obrázek 28: Funkce pro verifikaci hashovaného hesla

Využívá se kryptografické funkce PBKDF2 [24], která generuje klíč o požadované délce z hesla zadaného uživatelem a soli libovolné délky, a to opakovaným voláním pseudonáhodné funkce, jejímž výstupem je hash o určité délce.

V tomto projektu se klíč generuje o délce 256 bitů, počet iterací je 1000 a sůl se používá o délce 128 bitů.

O registraci a přihlášení se stará kontroler AccountController. (Obrázek 30) Tento kontroler obsahuje funkce AccountInfo(), Logout(), [HttpGet] Login(), [HttpPost] Login(LoginViewModel login), [HttpGet] Register() a [HttpPost] Register(RegisterViewModel register).

```
public class AccountController : Controller
{
    private UserManager<IdentityUser> UserMgr { get; }
    private SignInManager<IdentityUser> SignInMgr { get; }

    public AccountController(UserManager<IdentityUser> userManager,
        SignInManager<IdentityUser> signInManager)
    {
        UserMgr = userManager;
        SignInMgr = signInManager;
    }

    private Task<IdentityUser> GetCurrentUserAsync() => UserMgr.GetUserAsync(HttpContext.User);
}
```

Obrázek 29: Ukázka kontroleru AccountController

Funkce pro přihlášení (Obrázek 31) využívá funkce `SignInManager.PasswordSignInAsync()`, což je asynchronní funkce sloužící k autentizaci uživatele.

```
[HttpPost]
[AllowAnonymous]
public async Task<IActionResult> Login(LoginViewModel login, string command)
{
    var result = await SignInMgr.PasswordSignInAsync(login.username, login.password, false, false);

    if (result.Succeeded)
    {
        return RedirectToAction("Home", "Contracts");
    }
    else
    {
        ViewBag.Result = "Výsledek: " + result.ToString();
    }
    return View();
}
```

Obrázek 30: Funkce pro přihlášení uživatele

Funkce pro registraci (Obrázek 32) využívá funkce `UserManager CreateAsync()`, která vytvoří v databázi uživatele a heslo pošle k hashovací funkci a uložení.

```
[HttpPost]
[AllowAnonymous]
public async Task<IActionResult> Register(RegisterViewModel register)
{
    try
    {
        ViewBag.Message = "Uživatel je již registrován";
        IdentityUser user = await UserMgr.FindByNameAsync(register.username);
        if (user == null)
        {
            user = new IdentityUser();
            user.UserName = register.username;
            user.Email = register.email;

            IdentityResult result = await UserMgr.CreateAsync(user, register.password);

            if (result.Succeeded)
            {
                ViewBag.Message = "Uživatel byl vytvořen!";
            }
            else
            {
                ViewBag.Message = "Chyba: " + result.ToString();
            }
        }
    }
    catch (Exception ex)
    {
        ViewBag.Message = ex.Message;
    }
    return View();
}
```

Obrázek 31: Funkce pro registraci uživatele

Součástí zabezpečení aplikace je také systém rolí. Nemí žádoucí, aby měl každý uživatel přístup ke všem funkcím aplikace, a tak jsou rozřazeni do rolí – Zaměstnanec, Manažer a

Podpisovatel. O role se stará RolesController. (Obrázek 33) Tento kontroler má funkce pro přidání a úpravu rolí, a také pro přiřazení či odebrání uživatelů do/z role.

```
public class RolesController : Controller
{
    private RoleManager<IdentityRole> roleManager;
    private UserManager<IdentityUser> userManager { get; }

    public RolesController(RoleManager<IdentityRole> roleMgr, UserManager<IdentityUser> userMgr)
    {
        roleManager = roleMgr;
        userManager = userMgr;
    }
}
```

Obrázek 32: Ukázka kontroleru RolesController

Funkce EditUsersInRole(List<UserRoleViewModel> model, string roleId) se stará právě o přiřazení či odebrání uživatelů z/do rolí. (Obrázek 34)

```
[HttpPost]
public async Task<IActionResult> EditUsersInRole(List<UserRoleViewModel> model, string roleId)
{
    var role = await roleManager.FindByIdAsync(roleId);

    if(role == null)
    {
        ViewBag.ErrorMessage = $"Role s Id = {roleId} nebyla nalezena";
        return View("NotFound");
    }

    for (int i = 0; i < model.Count; i++)
    {
        var user = await userManager.FindByIdAsync(model[i].UserId);

        IdentityResult result = null;

        if (model[i].IsSelected && !(await userManager.IsInRoleAsync(user, role.Name)))
        {
            result = await userManager.AddToRoleAsync(user, role.Name);
        }
        else if (!model[i].IsSelected && await userManager.IsInRoleAsync(user, role.Name))
        {
            result = await userManager.RemoveFromRoleAsync(user, role.Name);
        }
        else
        {
            continue;
        }

        if (result.Succeeded)
        {
            if (i < (model.Count - 1))
            {
                continue;
            }
            else
            {
                return RedirectToAction("EditRole", new { Id = roleId });
            }
        }
    }
    return RedirectToAction("EditRole", new { Id = roleId });
}
```

Obrázek 33: Funkce EditUsersInRole

Dalším prvkem je zabezpečení vůči útokům CSFR (Cross-site Request Forgery). ASP.NET Core nabízí elegantní způsob zabezpečení pro všechny funkce zpracovávající výsledky formulářů. Je jím užití atributu `[ValidateAntiForgeryToken]` na funkci typu `[HttpPost]` a vložení příkazu `@Html.AntiForgeryToken()` do formuláře daného View. Příklad využití viz Obrázek 29 a Obrázek 30 v kapitole 4.2 Implementace funkcí.

Aplikace používá pro komunikaci klient-server protokol HTTPS pomocí užití příkazu `app.UseHttpsRedirection()` v součásti `Startup.cs` ve funkci `Configure()`.

4.4 Testování

Pro testovací část projektu jsou vybrány přístupy penetračního a integračního testování z pohledu white-box. Je vytvořeno několik testovacích případů, dle kterých je systém kontrolován. Cílem těchto testů je detekce a prevence bugů pro zajištění nejvyšší možné kvality softwaru. Pro tento účel je program publikován prostřednictvím cloudové platformy Microsoft Azure a testován v prohlížečích Chrome, Firefox, Safari, Edge a Opera v různých rozlišeních. Několik ukázkových testovacích případů pro otestování nejdůležitějších funkcí systému je přiloženo níže v tabulkové formě.

Tabulka 32: Penetrační testovací případ SQL Injection

ID případu:	001
Kategorie:	Penetrační testy
Popis:	SQL Injection na formuláři Vytvoření smlouvy
Prerekvizity:	Uživatel je přihlášen v systému.
Testovací kroky:	
Krok:	Popis:
1.	Zobrazení obrazovky pro vytvoření smlouvy.
2.	Zadání příkazu SQL injection (např. <code>nazev smlouvy; DROP TABLE Contracts</code>) do volného vstupu formuláře.
3.	Odeslání formuláře.
Očekávaný výsledek:	Je vytvořena smlouva s atributy odpovídající zadanému vstupu. Tabulka <code>Contracts</code> nebyla smazána.

Tabulka 33: Penetrační testovací případ Vstup do aplikace bez přihlášení

ID případu:	002
Kategorie:	Penetrační testy
Popis:	Vstup do aplikace bez přihlášení
Prerekvizity:	Žádné
Testovací kroky:	
Krok:	Popis:
1.	Zadání adresy <code>https://*název stránky*/Contracts/Home</code> do vyhledávače.
2.	Odeslání požadavku.
Očekávaný výsledek:	Uživateli je zobrazena obrazovka „Přístup odepřen“.

Tabulka 34: Penetrační testovací případ Uložení hashovaného hesla

ID případu:	003
Kategorie:	Penetrační testy
Popis:	Uložení hashovaného hesla
Prerekvizity:	Žádné
Testovací kroky:	
Krok:	Popis:
1.	Registrace nového uživatele do systému.
2.	Kontrola uloženého záznamu v databázi.
Očekávaný výsledek:	Záznam je uložen, heslo je šifrováno.

Tabulka 35: Penetrační testovací případ Omezený přístup k funkcím v závislosti na roli uživatele

ID případu:	004
-------------	-----

Kategorie:	Penetrační testy
Popis:	Omezený přístup k funkcím v závislosti na roli uživatele
Prerekvizity:	Žádné
Testovací kroky:	
Krok:	Popis:
1.	Přihlášení uživatele v roli Zaměstnanec.
2.	Uživatel nemá přístup k funkcím „Správa rolí“ a „Zaměstnanci“.
Očekávaný výsledek:	Uživatel nemá přístup k daným funkcím.

Velikost testovací specifikace vychází z úrovně komplexnosti testovaného systému. Pro cíle bakalářské práce přikládám ukázkou integračního testování ve formě dvou testovacích případů testující funkčnost formuláře pro vytvoření smlouvy.

Tabulka 36: Integrační testovací případ Vytvoření smlouvy – pozitivní

ID případu:	005
Kategorie:	Integrační testy
Popis:	Vytvoření smlouvy – pozitivní
Prerekvizity:	Uživatel je přihlášen v systému v roli Zaměstnanec nebo Manažer
Testovací kroky:	
Krok:	Popis:
1.	Kliknutí na tlačítko „Nová smlouva“.
2.	Vyplnění formuláře validními daty.
3.	Odeslání formuláře.
Očekávaný výsledek:	Smlouva je úspěšně uložena v databázi.

Tabulka 37: Integrační testovací případ Vytvoření smlouvy – negativní

ID případu:	007
-------------	-----

Kategorie:	Integrační testy
Popis:	Vytvoření smlouvy – negativní
Prerekvizity:	Uživatel je přihlášen v systému v roli Zaměstnanec nebo Manažer
Testovací kroky:	
Krok:	Popis:
1.	Kliknutí na tlačítko „Nová smlouva“.
2.	Vyplnění formuláře ne-validními daty.
3.	Odeslání formuláře.
Očekávaný výsledek:	Uživateli je zobrazena chybová hláška.

4.5 Grafické zpracování

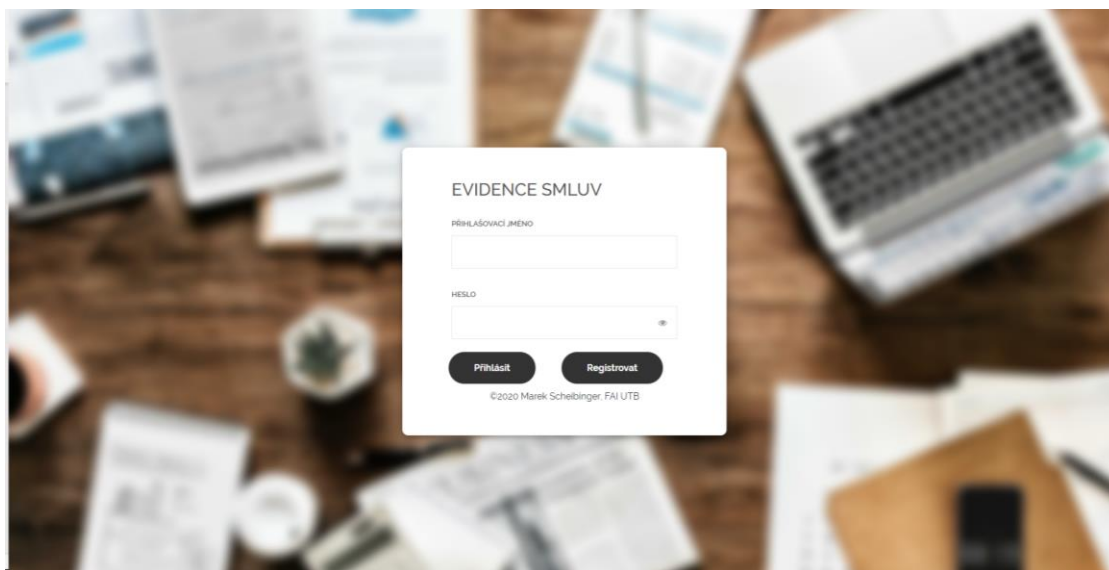
Grafické zpracování aplikace vychází z návrhu drátěných modelů. Používanými technologiemi pro jejich implementaci jsou HTML, CSS, Javascript a jQuery. Jakožto návrhářskou šablonu pro prvky aplikace je vybrána sada nástrojů Bootstrap ve verzi 4.5. Je vybrána také vhodná ikonografie, která je čerpána z balíčku Font Awesome [25].



Obrázek 34: Použitá ikonografie

Je vybrána následující paleta barev: #c7ccd1, #343a40, #28a745, #007bff.

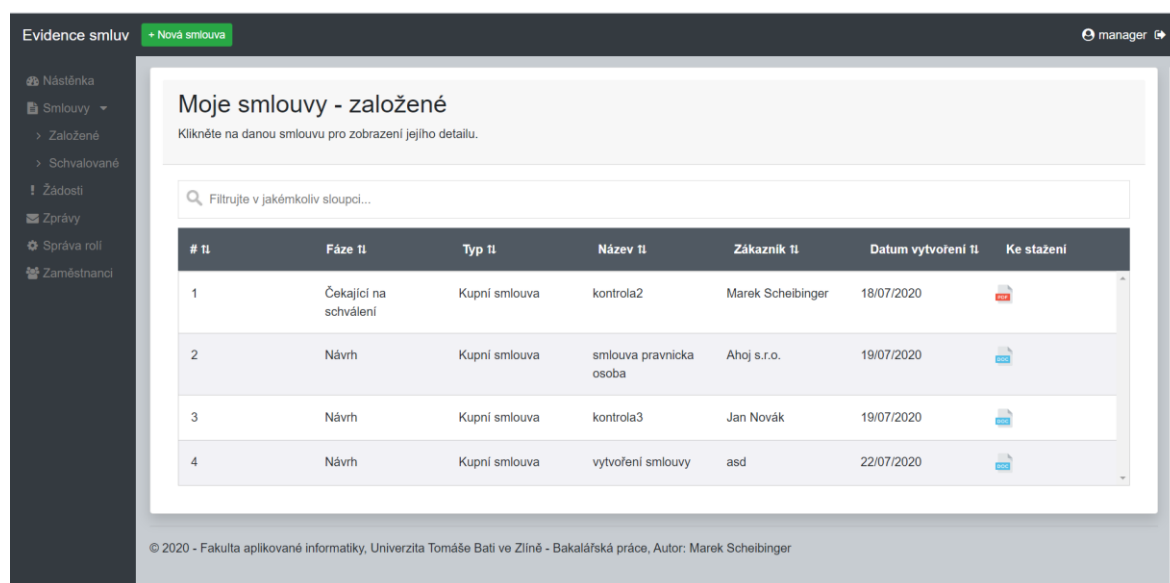
Pro přihlašovací a registrační obrazovku (Obrázek 36) je inspirací již existující šablona. Tato šablona [26] je použita a upravena. Konkrétně je přidáno pozadí a stínování karty. Pro výběr pozadí těchto obrazovek [27] je čerpáno ze serveru Wallpaper.dog.



Obrázek 35: Zobrazení k přihlášení uživatele

Pro realizaci grafického zobrazení uvnitř aplikace jsou inspirací již existující řešení, které byly zmíněny v teoretické části bakalářské práce. Při implementaci však již není používáno šablon, je čerpáno pouze z dokumentace Bootstrap [28] a edukační webové stránky zaměřené na webové technologie W3Schools [29].

Pro ukázkou grafického zpracování uvnitř aplikace je přiložen následující obrázek. (Obrázek 37) V tomto případě je přihlášen uživatel v roli Manažer, má tak přístup ke většině funkcí aplikace a jeho zobrazení je tedy relevantní pro prezentaci grafického zpracování.



Obrázek 36: Ukázkou grafického zpracování aplikace

ZÁVĚR

Výsledkem teoretické části této bakalářské práce je popis technologií využitých ve vývoji projektu. Je popsán framework ASP.NET Core a návrhový vzor MVC, spolu s programovacím jazykem C# a jeho výhody při vytváření tohoto typu webových aplikací. Dalším výstupem jsou také závěry uvedené k jednotlivým již existujícím řešením.

Výsledkem praktické části bakalářské práce je pak návrh aplikace pro evidenci smluv a na jejím základě vyvinutá plně funkční webová aplikace. Byly vybrány přístupy k vývoji této aplikace a byla zajištěna bezpečnostní opatření vůči útokům typu SQL Injection a CSFR, ukládání citlivých dat jako jsou uživatelská hesla, zajištění šifrované komunikace klient-server pomocí protokolu HTTPS a omezený přístup k jednotlivým funkcím aplikace na základě uživatelských rolí.

Systém byl vytvořen na základě požadavků získaných rešerší již existujících řešení. Na základě těchto požadavků byl vytvořen model případů užití, E-R diagram a diagram tříd. Podle tohoto návrhu byl systém vyvinut. Součástí návrhu tohoto systému je také možnost v budoucnosti tuto aplikaci upravovat a rozvíjet, bez nutnosti většího zásahu do již existující architektury. Vytvořená aplikace je plně funkčním systémem pro správu smluv, jehož cílem je poskytovat podporu firmám k řízení smluv a řízení jejich životního cyklu. Pro nasazení systému ve firmě je doporučeno využít certifikátu SSL a zrcadlení databáze například prostřednictvím cloudové platformy Microsoft Azure.

SEZNAM POUŽITÉ LITERATURY

- [1] The Best Contract Management Software | PCMag. The Latest Technology Product Reviews, News, Tips, and Deals | PCMag [online]. Copyright © 1996 [cit. 13.07.2020]. Dostupné z: <https://www.pcmag.com/picks/the-best-contract-management-software>
- [2] Agiloft Service Desk Review | PCMag. The Latest Technology Product Reviews, News, Tips, and Deals | PCMag [online]. Copyright © 1996 [cit. 13.07.2020]. Dostupné z: <https://www.pcmag.com/reviews/agiloft-service-desk>
- [3] YouTube. YouTube [online]. Copyright © 2020 Google LLC [cit. 13.07.2020]. Dostupné z: <https://www.youtube.com/watch?v=6UkbNmhzsF8>
- [4] Concord Software - 2020 Reviews, Pricing & Demo. Business Software Reviews from Software Advice® [online]. Copyright © 2006 [cit. 13.07.2020]. Dostupné z: <https://www.softwareadvice.com/nz/scm/concord-contract-management-profile/>
- [5] Framework technologies Web Usage Distribution on the Entire Internet. BuiltWith Web Technology Usage Statistics [online]. Dostupné z: <https://trends.builtwith.com/framework/traffic/Entire-Internet>
- [6] 8 Best PHP Frameworks for Web Developers. Hosting Platform - Go Online With Hostinger For Only \$0.99 Now [online]. Copyright © 2004 [cit. 27.07.2020]. Dostupné z: <https://www.hostinger.com/tutorials/best-php-framework>
- [7] Vyberte ASP.NET 4. x a ASP.NET Core | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 13.07.2020]. Dostupné z: <https://docs.microsoft.com/cs-cz/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-3.1>
- [8] ALBAHARI, Joseph; ALBAHARI, Ben. C# 7.0 in a nutshell: The definitive reference. " O'Reilly Media, Inc.", 2017.
- [9] Introduction with Updated ASP.NET Core 3.1 | by Valiant Technosoft | Medium. Medium – Get smarter about what matters to you. [online]. Dostupné z: <https://medium.com/@valianttechnosoft/introduction-with-updated-asp-net-core-3-1-434b80f0facf>
- [10] ASP.NET vs PHP: Which is Better for Web Development? | Existek Blog. Offshore Software Development Company | Offshore Software Development Services [online]. Copyright © Existek 2012 [cit. 27.07.2020]. Dostupné z: <https://existek.com/blog/aspnet-vs-php-which-is-better-for-web-development/>

- [11] .NET Core official support policy. .NET | Free. Cross-platform. Open Source. [online]. Dostupné z: <https://dotnet.microsoft.com/platform/support/policy/dotnet-core>
- [12] Migrace z ASP.NET Core 2,2 na 3,0 | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 13.07.2020]. Dostupné z: <https://docs.microsoft.com/cs-cz/aspnet/core/migration/22-to-30?view=aspnetcore-3.1&tabs=visual-studio>
- [13] Přehled ASP.NET Core MVC | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 13.07.2020]. Dostupné z: <https://docs.microsoft.com/cs-cz/aspnet/core/mvc/overview?view=aspnetcore-3.1>
- [14] 6 Advantages of using MVC model for effective web application development. Software Development Company - IT Consulting Firm | Brainvire [online]. Copyright © 2020 Brainvire Infotech Inc. [cit. 27.07.2020]. Dostupné z: <https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/>
- [15] Lekce 7 - LINQ v C# .NET - Revoluce v dotazování. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2020 itnetwork.cz. Veškerý obsah webu [cit. 14.07.2020]. Dostupné z: <https://www.itnetwork.cz/csharp/kolekce-a-linq/c-sharp-tutorial-linq-dotazy>
- [16] LINQ & SQL Injection . Entity Framework Entity Framework | Entity Framework 6 Tutorial and Documentation [online]. Dostupné z: <https://entityframework.net/linq-prevent-sql-injection>
- [17] SQL Injection - SQL Server | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 14.07.2020]. Dostupné z: <https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver15>
- [18] TROELSEN, Andrew; JAPIKSE, Philip. Pro C# 7: With. net and. net Core. Apress, 2017.
- [19] Code First vs Database First . Entity Framework Entity Framework | Entity Framework 6 Tutorial and Documentation [online]. Dostupné z: <https://entityframework.net/code-first-vs-database-first>

- [20] What is Microsoft SQL Server? A definition from WhatIs.com. SQL Server: Covering today's SQL Server topics [online]. Dostupné z: <https://searchsqlserver.techtarget.com/definition/SQL-Server>
- [21] WIEGERS, Karl; BEATTY, Joy. Software requirements. Pearson Education, 2013
- [22] 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.
- [23] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. The unified modeling language reference manual. 1999.
- [24] Základy kryptografie pro manažery: PBKDF2 - CleverAndSmart Management Consulting. CleverAndSmart Management Consulting [online]. Copyright © 2008 [cit. 16.07.2020]. Dostupné z: <https://www.cleverandsmart.cz/zaklady-kryptografie-pro-manazery-pbkdf2/>
- [25] Font Awesome. Font Awesome [online]. Dostupné z: <https://fontawesome.com/>
- [26] Login Form 14 by Colorlib - Free HTML Login Form 2020 - Colorlib. How To Start A Blog From Scratch Using WordPress - Colorlib [online]. Dostupné z: <https://colorlib.com/wp/template/login-form-v14/>
- [27] Business Training Wallpapers on WallpaperDog. Cool Wallpapers - WallpaperDog [online]. Copyright © 2020 [cit. 27.07.2020]. Dostupné z: <https://wallpaper.dog/business-training-wallpapers>
- [28] Introduction · Bootstrap v4.5. Bootstrap · The most popular HTML, CSS, and JS library in the world. [online]. Dostupné z: <https://getbootstrap.com/docs/4.5/getting-started/introduction/>
- [29] W3Schools Online Web Tutorials. W3Schools Online Web Tutorials [online]. Dostupné z: <https://www.w3schools.com/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PHP	Hypertext Preprocessor
IDE	Integrated Development Environment
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
MVC	Model View Controller
LINQ	Language Integrated Query
SQL	Structured Query Language
XML	Extensible Markup Language
EF	Entity Framework
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secured
SSL	Secure Sockets Layer
DOCX	Office Open XML document
PDF	Portable Document Format
UML	Unified Modeling Language
PBKDF2	Password-Based Key Derivation Function 2
CSFR	Cross-site Request Forgery

SEZNAM OBRÁZKŮ

Obrázek 1: Domovská stránka systému Agiloft [2].....	10
Obrázek 2: Domovská stránka systému Onesoft Connect [3]	11
Obrázek 3: Domovská stránka systému Concord [4]	12
Obrázek 4: Historie ASP.NET Core [9]	14
Obrázek 5: Diagram architektury MVC [13].....	14
Obrázek 6: Liniová organizační struktura firmy.....	18
Obrázek 7: Přístupy k funkcím aplikace dle rolí uživatele	19
Obrázek 8: Životní cyklus smlouvy	20
Obrázek 9: UML diagram případů užití	30
Obrázek 11: UML diagram tříd	45
Obrázek 12: Zobrazení "Přihlášení"	46
Obrázek 13: Zobrazení aplikace pro uživatele v roli Manažer	47
Obrázek 14: Zobrazení "Nástěnka“	47
Obrázek 15: Zobrazení Vytvoření smlouvy.....	48
Obrázek 16: Zobrazení "Moje smlouvy – založené"	49
Obrázek 17: Zobrazení detailu smlouvy po jejím vytvoření	50
Obrázek 18: Ukázka vytvoření modelu Contract	51
Obrázek 19: Přidání DbContext do projektu	51
Obrázek 20: Vytvoření třídy spravující DbContext.....	52
Obrázek 21: Vytvoření kontextových tříd	52
Obrázek 22: Ukázka vygenerované migrace	53
Obrázek 23: Ukázka kontroleru ContractsController	53
Obrázek 24: Implementace funkce UploadFileDocx(int id) s atributem [HttpGet] ...	54
Obrázek 25: Zobrazení UploadFileDocx.cshtml	55
Obrázek 26: Ukázka funkce UploadFileDocx() s atributem [HttpPost]	56
Obrázek 27: Ukázka LINQ	57
Obrázek 28: Hashovací funkce knihovny Microsoft.AspNet.Identity	57
Obrázek 29: Funkce pro verifikaci hashovaného hesla	58
Obrázek 30: Ukázka kontroleru AccountController	58
Obrázek 31: Funkce pro přihlášení uživatele	59
Obrázek 32: Funkce pro registraci uživatele	59
Obrázek 33: Ukázka kontroleru RolesController	60

Obrázek 34: Funkce EditUsersInRole	60
Obrázek 35: Použitá ikonografie	64
Obrázek 36: Zobrazení k přihlášení uživatele	65
Obrázek 37: Ukázka grafického zpracování aplikace.....	65

SEZNAM TABULEK

Tabulka 1: Požadavek Registrace	21
Tabulka 2: Požadavek Přihlášení	22
Tabulka 3: Požadavek Navigace	22
Tabulka 4: Požadavek Vytvoření smlouvy	23
Tabulka 5: Požadavek Detail smlouvy	23
Tabulka 6: Požadavek Nahrání souborů	24
Tabulka 7: Požadavek Nástěnka	24
Tabulka 8: Požadavek Zobrazení smluv	24
Tabulka 9: Požadavek Žádosti	25
Tabulka 10: Požadavek Zprávy	25
Tabulka 11: Požadavek Správa rolí	25
Tabulka 12: Požadavek Přehled zaměstnanců	26
Tabulka 13: Požadavek Úprava oddělení	26
Tabulka 14: Požadavek Životní cyklus smlouvy	26
Tabulka 15: Požadavek Správa smlouvy	27
Tabulka 16: Požadavek Systém rolí.....	28
Tabulka 17: Požadavek Zobrazení informací o přihlášeném uživateli	28
Tabulka 18: Scénář případu užití Registrace	31
Tabulka 19: Scénář případu užití Přihlášení	31
Tabulka 20: Scénář případu užití Zobrazení nástěnky.....	32
Tabulka 21: Scénář případu užití Zobrazení smluv	33
Tabulka 22: Scénář případu užití Zobrazení žádostí	34
Tabulka 23: Scénář případu užití Zobrazení zpráv	35
Tabulka 24: Scénář případu užití Zobrazení konkrétní smlouvy.....	36
Tabulka 25: Scénář případu užití Správa životního cyklu smlouvy	37
Tabulka 26: Scénář případu užití Zobrazení informací o přihlášeném uživateli	38
Tabulka 27: Scénář případu užití Správa smlouvy	38
Tabulka 28: Scénář případu užití Přehled zaměstnanců	40
Tabulka 29: Scénář případu užití Změna oddělení zaměstnance.....	41
Tabulka 30: Scénář případu užití Správa rolí	42
Tabulka 31: Scénář případu užití Vytvoření smlouvy	43
Tabulka 32: Penetrační testovací případ SQL Injection	61

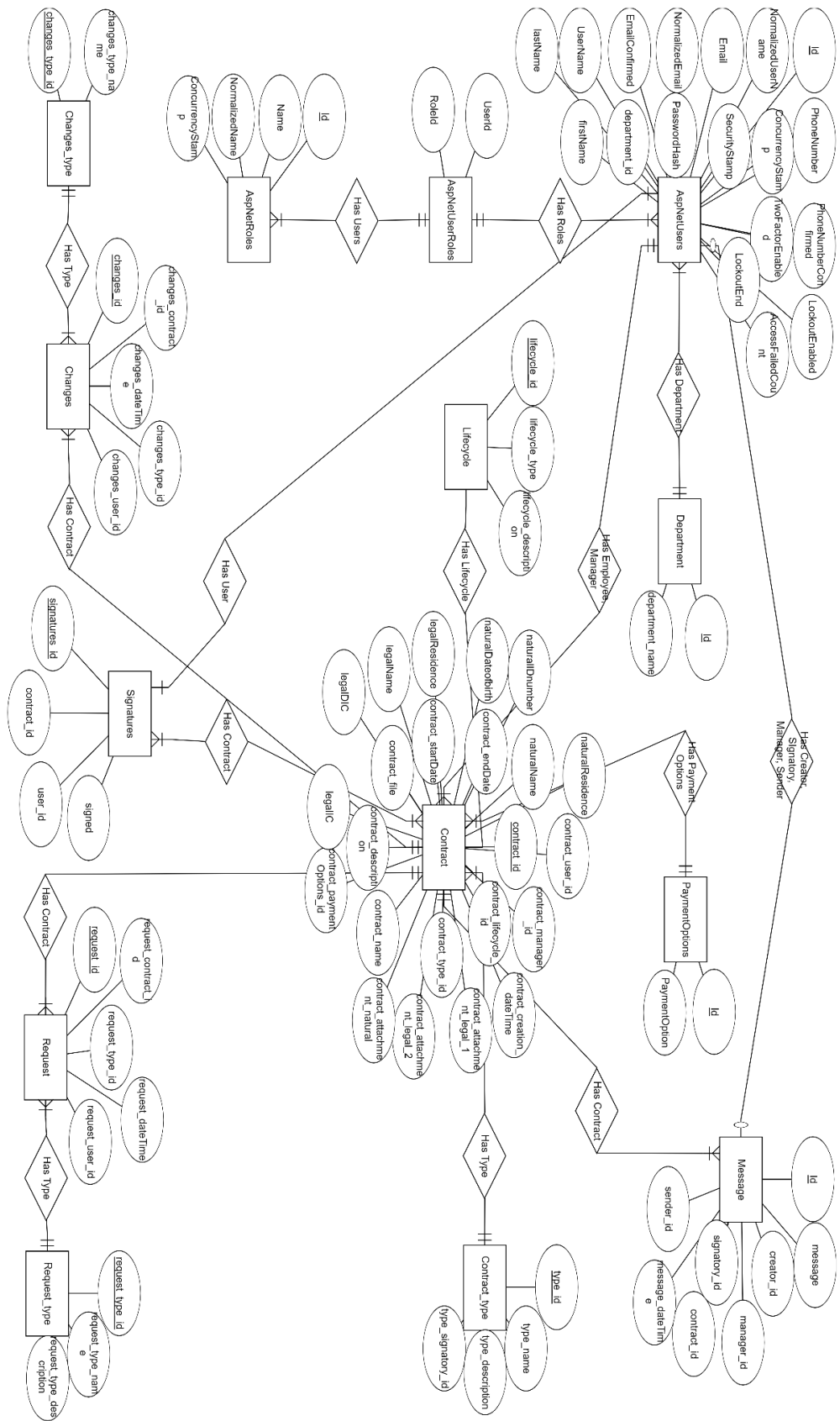
Tabulka 33: Penetrační testovací případ Vstup do aplikace bez přihlášení.....	62
Tabulka 34: Penetrační testovací případ Uložení hashovaného hesla	62
Tabulka 35: Penetrační testovací případ Omezený přístup k funkcím v závislosti na roli uživatele	62
Tabulka 36: Integrační testovací případ Vytvoření smlouvy – pozitivní	63
Tabulka 37: Integrační testovací případ Vytvoření smlouvy – negativní.....	63

SEZNAM PŘÍLOH

P I E-R diagram

P II CD disk s bakalářskou prací a zdrojovým kódem

PŘÍLOHA P I: E-R DIAGRAM



Obrázek 1: E-R diagram