Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Informační systém znalce v oblasti informační technologií

Bakalářská práce

Vilém Havel
Školitel: Ing. Jaroslav Kothánek, Ph.D.

České Budějovice 2017

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Vilém Havel B14135
Student: .....
(jméno, příjmení, tituly)

Aplikovaná informatika
Kriminalisticko-technická činnost v IT
Obor – zaměření studia: .....

Ústav aplikované informatiky
oddělení Forenzních věd a kriminalistiky
Katedra: .....

Ing. Jaroslav Kothánek, Ph.D.,
jkothanek@prf.jcu.cz
Školitel: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)
Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Informační systém znalce v oblasti informačních technologií

Úkoly a cíle práce :

1. Seznamte se s problematikou zásad znalecké činnosti v IT
2. Proveďte analýzu zákonných potřeb znalce v oblasti IT
3. Vytvořte informační systém pokrývající administrativní a pomocné činnost znalce v oblasti IT, včetně komunikace s datovou schránkou a další komunikace s orgány činnými v trestním řízení
4. Aplikujte informační systém a ověřte jeho funkčnost
5. Prověřte IS ve vztahu k bezpečnosti ukládaných dat

Základní doporučená literatura :

1. Zákon o znalcích a tlumočnících č. 36/1967 Sb., ve znění pozdějších předpisů
2. Prováděcí vyhláška k zákonu o znalcích a tlumočnících č. 37/1967 Sb. ve znění pozdějších předpisů
3. Dále viz pokyn vedoucího v průběhu vytváření IS

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných Přírodovědeckou fakultou] elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Budějovicích dne …….. Podpis …….

Bibliografické údaje

Havel V., 2017: Informační systém znalce v oblasti informační technologií [Information system for experts in areas of Information Technologies Bc. Thesis, in Czech] – 40 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace

Cílem této práce je vytvořit informační systém pro znaleckou činnost. V práci je popsána znalecká činnost a návrh aplikace až po její výslednou implementaci. Výsledkem praktické části práce je funkční aplikace, která dokáže archivovat a vytvářet znalecké posudky.

Klíčová slova

Informační systém, znalecká činnost, webová aplikace, ASP.NET Core, API, framework

Annotation

Main objective of this thesis is to create information system for expert activity. Thesis describes from expert activity and application design, to its final implementation. Output of practical part is working application which can archive and create expert opinion.

Keywords

Information systém, expert activity, web application, ASP.NET Core, API, framework

Poděkování

Rád bych poděkoval svému vedoucímu práce panu doktorovi Jaroslavovi Kothánkovi za jeho odborné rady a připomínky v průběhu práce. Díky patří také mé rodině a přátelům za podporu při studiu.

Seznam použitých zkratek

Zkratka	Význam
IS	Informační Systém
IT	Informační Technologie
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
MVC	Model View Controller
AJAX	Asynchronous JavaScript and XML
XML	Extensible Markup Language
GUI	Graphical User Interface
JSON	JavaScript Object Notation
API	Application Programming Interface
IIS	Internet Information Services
ORM	Objektově Relační Mapování
CRUD	Create Read Update Delete

Obsah

1 Úvod

V dnešní době moderních technologií a rozšířenosti internetu je vysoký zájem o informační systémy, které nám usnadňují práci a řeší úlohy, které by byly bez počítače proveditelné velmi těžko. Informační systémy se nejčastěji vyskytují ve formě webové aplikace. Hlavně díky její snadné rozšiřitelnosti a přístupu. Při použití moderních webových frameworků můžeme klientskou část udělat více přívětivou a intuitivní. Jako je tomu u těžkých klientů. V mém průzkumu nebylo nalezeno žádné open-source řešení zabývající se tímto problémem. S velkou pravděpodobností, ale existují řešení, která nejsou veřejně dostupná.

1.1 Cíle

Cílem práce je navrhnout a naprogramovat informační systém, který bude vyhovovat potřebám znalce. Systém bude podléhat bezpečnostním kritériím na ukládání dat. Dále bude umět pracovat se znaleckými daty a provádět s nimi příslušné operace.

Dílčí cíle:

- Seznámit se s problematikou zásad znalecké činnosti v IT. Dále zjistit jaké jsou náležitosti posudku.
- Provést analýzu a návrh informačního systému, aby vyhovoval pracovní náplni znalce a učinil jí jednoduší. Dalším krokem bude zvolit a navrhnout vhodnou databázi.
- Provést implementaci informačního systému. Dále napojit systém na databázi a ověřit správné ukládání dat.
- Propojit systém s datovou schránkou, se kterou bude umět komunikovat, a s jinými webovými službami, které usnadňují práci znalce.
- Zajistit správu ukládaných dokumentů s ohledem na jejich bezpečnost.
- Aplikovat informační systém a ověřit jeho funkčnost.

2 Znalecká činnost

Znalec je osoba s příslušnou kvalifikací a praxí, kterou stanovuje zákon č.36/67 Sbírky o znalcích a tlumočnících [1] a podle vyhlášky ministerstva č.37/67 Sbírky o provedení zákona o znalcích a tlumočnících [2]. Znalec není jediný, kdo může vydávat znalecké posudky, ale můžou být vyžádány také od znaleckých a vědeckých ústavů, vysokých škol a jiných institucí. Znalecký posudek si můžou vyžádat státní orgány, ostatní právnické osoby, fyzické osoby, podnikatelé vykonávající samostatně výdělečnou činnost a občané.

2.1 Jmenování znalců

Pro stání se znalcem musí být podán návrh od orgánů veřejné moci, vědeckých institucí, vysokých škol a organizací u nichž pracují osoby přicházející v úvahu např. občanská sdružení, obecně prospěšné společnosti a nadace, pokud to vyplívá z předmětu jejich činnosti. Znalcem může být také jmenován ten, kdo o jmenování požádá. Orgány a organizace uvedené výše na žádost ministra spravedlnosti nebo předsedy krajského soudu vyjadřují, zda navrhovaný znalec splňuje podmínky pro jmenování [2].

2.2 Zodpovědnost znalce

Znalec je povinen vykonávat činnost osobně, řádně a ve stanovené lhůtě. Znalec, má možnost vykonávat činnost mino obvod krajského soudu, u něhož jsou zapsáni. Do povinností znalce spadá zachovávání mlčenlivosti o skutečnostech, které se dozvěděl v průběhu vykonávání činnosti a i po jejím ukončení pokud nejsou přiměřeným způsobem použity pro vědecké nebo vzdělávací způsoby. Také je povinen osobně stvrdit, doplnit nebo blíže vysvětlit jeho obsah pokud si to od něj státní orgán vyžádá.

Pokud povaha věci vyžaduje vyšší odbornost je znalec oprávněn přibrat konzultanta na posouzení dílčích otázek. Jestliže se tak stane, v posudku musí být uvedeny důvody a okolnosti, které vedly k vybrání konzultanta.

V případě, že znalec potřebuje k provedení znaleckého úkonu přístroje a materiály, kterými organizace nedisponuje, tak je povinna vydat o tom znalci potvrzení, podle něhož zažádá jinou organizaci o potřebnou pomoc v rozsahu stanoveném zákonem. V tomto případě se v posudku uvádí náklady, které znalec uhradil za použití jejich majetku.

Povinnost znalců je vést si znalecký deník, do něhož zaznamenávají provedení všech posudků, jejich předmět, pro koho byla činnost provedena, výše odměny, výloh a den jejich proplacení. Znalecký deník je opatřen pečetí od krajského soudu, u kterého je uveden počet listů deníku. Kvůli řádnému vedení deníku se provádí občasné kontroly [2].

2.3 Náležitosti posudku

Příslušný orgán, který ustanovil znalce, vymezí ve svém opatření jeho úkol, aby se znalec zabýval jen takovými skutečnostmi, k jejichž posouzení je třeba jeho odborných znalostí.

Znalecký posudek je sešit s očíslovanými stranami a jeho šňůrka je připevněna k poslední straně posudku a přetištěna znaleckou pečetí. V sešitu se nachází popis zkoumaného materiálu, souhrn skutečností, k nimž při úkonu přihlížel a výčet otázek na které má odpovědět. Na poslední straně dokumentu se nachází znalecká doložka, která obsahuje označení seznamu, v němž je znalec zapsán, označení oboru, v němž je oprávněn podávat posudky, a číslo položky pod kterou je úkon zapsán ve znaleckém deníku. Na požádání státního orgánu je povinnost znalce písemný posudek stvrdit, doplnit nebo jeho obsah blíže vysvětlit [2].

2.4 Analýza potřeb znalce

Hlavním cílem analýzy bylo seznámení s potřebami znalce. Analýza určuje, co má systém umět, ale ne jakým způsobem toho dosáhnout. Podle úsudku autora práce, byla zvolena jako informační systém webová aplikace. Hlavně kvůli snadnému přístupu odkudkoliv za pomoci internetového prohlížeče. V dnešní době není problém napsat aplikaci, která by

byla na podobné úrovni jako těžký klient. Toto řešení nám také usnadňuje případné aktualizace systému. Aplikace bude přizpůsobená speciálně pro činnost znalce, aby zjednodušila jeho jednotlivé úkony.

Webová aplikace bude obsahovat přihlášení pro jediného uživatele, kterým je soudní znalec, pro kterého je systém napsán. Z tohoto důvodu nebude třeba řešit v aplikaci práva. Přihlásit se bude moci administrátor aplikace, který bude moci přidávat, upravovat a mazat jednotlivé znalecké posudky. Také bude moci upravovat příslušná nastavení aplikace. Informace o jednotlivých opatřeních nesmí být veřejně přístupné. Tak bude kladen velký důraz na bezpečnost. Jedná se o důkazy v kriminálních případech, jejichž odhalení by mohlo narušit průběh vyšetřování.

Po přijetí nového opatření bude založen nový posudek založený na získaných údajích. U vybraných údajů bude také možnost editace a mazání. V informačním systému bude implementováno fulltextové vyhledávání. Záznamy půjde řadit podle data založení a vypracování posudku. Aplikace bude upozorňovat znalce na vypršení splatnosti faktury. Další funkčnost bude komunikace s webem ARES, u kterého jsou zaregistrované ekonomické subjekty. Tento web bude využívat na ověření identity a doplnění informací o dožadateli. IS bude propojený s datovou schránkou. Ze získaných údajů půjde vygenerovat úvodní část znaleckého posudku.

Další důležitou součástí systému bude ukládání stop. V této části se bude přiřazovat popis jednotlivých stop a bude k nim možné přiřadit dokumenty.

3 Návrh

3.1 Architektura

Před implementací samotného řešení bylo nutné pro usnadnění vývoje zvolit návrh aplikace. Na základě těchto modelů byla navržena aplikace a zvoleny technologie, podle kterých proběhla implementace aplikace.

3.1.1 Single-page Application

V tradičních webových aplikacích se komunikace se serverem řeší načtením celé stránky do prohlížeče. Server zpracuje požadavek, najde požadovaná data a pošle je s příslušnou HTML šablonou webovému prohlížeči. Tento způsob není tolik efektivní. Vždy musíme znovu stahovat celou stránku, i když by nám stačila obnovit pouze data nebo změnit jenom část stránky.

V Single-page aplikacích [3] je celá webová stránka nahrána při prvním požadavku a routování mezi jednotlivými stránkami je řízeno na straně klienta. Tímto způsobem je server méně zatížen. Pomocí tohoto způsobu nám odpadá protivné problikávání stránek, protože obnovujeme pouze tu část, kterou potřebujeme. Uživateli poté přijde aplikace svižnější.

3.1.2 Webová služba

Webová služba je aplikace, která nám dovoluje komunikaci dvou nezávislých strojů na síti. Tato metoda nám umožnuje vyvíjet serverovou a klientskou část aplikace nezávisle na sobě. Také nás nezajímá, na jaké zařízení data posíláme. Záleží zcela na klientovi, jak je bude zpracovávat. Jediné co potřebujeme vědět je, jak bylo navrženo aplikační rozhraní webové aplikace, přes které bude server a klient komunikovat.

3.1.3 REST

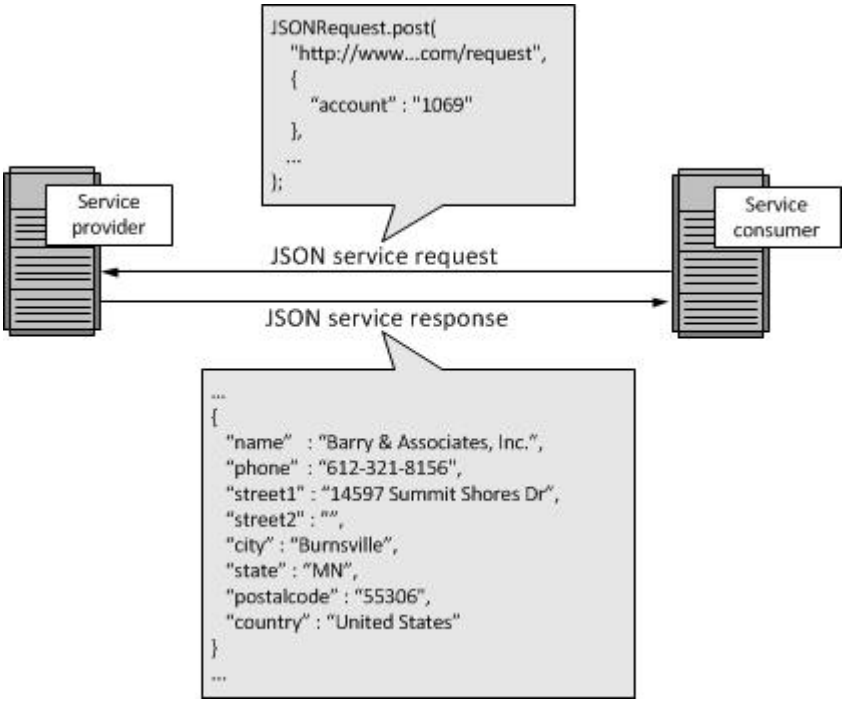
Representational state transfer [4] je druh protokolu, který používají webové služby. Je založený na jednotném a snadném přístupu ke zdrojům. Pro komunikaci přes HTTP využívá čtyř metod pomocí filosofie CRUD.

Operace	Metoda	Popis
Create	POST	Vytvoří nový záznam

Read	GET	Načte data
Update	PUT	Upraví stávající záznam
Delete	DELETE	Smaže záznam

Tabulka 1 CRUD operace – převzato z [4]

K tomuto protokolu byl zvolen formát pro výměnu dat JSON. Tento formát byl vybrán pro jeho jednoduchost, obecnost a čitelnost pro člověka. Hodnoty se ukládají ve formátu klíč hodnota. Tento způsob je velmi jednoduchý na pochopení a díky rozsáhlé podpoře nám umožňuje snadné parsování dat v rozličných programovacích jazycích.



Obrázek 1 Komunikace přes JSON – převzato z [4]

3.2 Výběr technologií

Technologie byly vybrány na základě zkušeností autora a také pro vyzkoušení nových technologií. Byl kladen důraz na podporu výrobce a možnosti rozšířitelnosti do budoucna.

3.2.1 ASP.NET Core

ASP.NET Core [5] je nový open-sourcový scriptovací jazyk bežící na straně serveru. Je napsaný v jazyce C# a vyvíjen společností Microsoft. Vychází ze staršího frameworku ASP.NET [6]. Je od základu přepsaný, aby byl rychlejší a mohl být spuštěn na více zařízeních. Podporovanými operačními systémy jsou Windows, Mac a vybrané linuxové distribuce. Microsoft oznámil, že starší verze ASP.NET nebude dále vyvíjena a bude se věnovat jen ASP.NET Core. Z tohoto důvodu bylo výhodnější použít verzi Core, která není závislá na technologiích od Microsoftu. Jeho nejnovější verze v době psaní Bakalářské práce je 2.0. Byla vydaná 14.8.2017 a je volně přístupná na uložišti Github. vyznačuje se lepším výkonem, zjednodušením vývoje a menším zabíráním prostoru na disku.

ASP.NET Core	ASP.NET
Běží na Windows, macOS a Linuxu	Běží na Windows
Více verzí na jednom zařízení	Jedna verze na zařízení
Vyvíjet můžeme ve Visual Studiu, Visual Studiu pro Mac nebo Visual Studiu Code pomocí C# a F#	Vyvíjet můžeme ve Visual Studiu pomocí C#, VB a F#
Větší výkon než ASP.NET	Dobrý výkon
Používá .NET framework nebo .NET Core	Používá .NET Framework

Tabulka 2 Porovnání ASP.NET frameworků - převzato z [5]

3.2.2 Entity Framework Core

Pro rychlejší práci s databází byl použit framework, který zajištuje objektově relační mapování. Tato technika nám umožňuje automatickou konverzi dat mezi relační databází a objektově orientovaným jazykem. Také nám redukuje kód, který je programátor nucen napsat. Jako nejvhodnější ORM systém byl vybrán Entity Framework

Core [7], který je odlehčenou verzí klasického Entity Frameworku a je přímo vytvořen pro spolupráci s ASP.NET Core [5].

3.2.3 AngularJS

Pro práci na klientské straně byl vybrán javascriptový framework AngularJS [8]. Tento framework byl napsaný společností Google a je vhodný pro tvorbu single-page aplikací. Také nám řeší rozdíly implementace mezi jednotlivými prohlížeči. Oproti javascriptové knihovně JQuery, která míchá aplikační logiku, zpracování událostí a manipulaci s DOM. AngularJS nám rozděluje psaní kódu pomocí architektury MVC.

3.2.4 AngularJS Material

Angular Material [9] patří do skupiny GUI frameworků, které nám mají zajistit rychlejší vytváření vzhledu stránky. Po grafické stránce vypadá tento framework uživatelsky přívětivě a proto nemusel být nijak výrazně upravován. Další vlastností je tvorba layoutu, aby se stránka chovala responzivně a byla čitelná i na mobilních zařízeních. Byl vybrán hlavně kvůli jeho úzké spolupráci s AngularemJS [8]. Je napsaný Googlem a založený na jejich Material desingu.

3.2.5 Bower

S přibývajícím nárůstem javascriptových knihoven a frameworků je dobré využít nějakého nástroje pro automatické řazení a aktualizaci komponent. Pro tento účel byl vybrán manažer balíčků Bower [10]. Tento nástroj nám drží informace o nainstalovaných komponentách v jednom JSONovém souboru. Nové balíčky můžeme jednoduše přidat pomocí příkazu bower install názevBalíčku.

3.2.6 GitLab

Z důvodu zálohování zdrojových kódů a práce z více míst, bylo nutné zvolit uložiště uzpůsobené pro tyto účely. Autor práce proto zvolil gitové uložiště GitLab [11], které se oproti podobnému řešení GitHub vyznačuje výhodou, že ve verzi zdarma můžeme mít zdrojové kódy v soukromém režimu. Dále podporuje základní operace jako commitování kódu a jeho následné stahování. Také umožňuje verzování, tvoření větví a jejich následné slučování.

3.2.7 IIS

Neboli Internet Information Services [12] je webový server vytvořený společností Microsoft. Je spustitelný pouze pod operačním systémem Microsoft Windows. V aplikaci je použita jeho bezplatná a zjednodušená verze Express.

3.2.8 MS-SQL

Jako databázový server byl zvolen MS-SQL [13]. Tento server velmi dobře spolupracuje s technologiemi, které byly zvoleny pro vývoj aplikace. Byl vyvinutý společností Microsoft. Pro informační systém byla použita jeho bezplatná verze Express. Tato verze je omezená datovým prostorem na 10GB a omezením výkonu. Pro menší aplikace, jako tento systém, je jeho výkon a velikost dostatečný.

3.3 Uživatelské role

Při návrhu uživatelských rolí nebylo nutné vymýšlet složité rozdělení práv u uživatelů. Informační systém bude jednouživatelský. Nepůjde zakládat nové uživatele. Z toho vyplývá, že byl řešen pouze rozdíl mezi přihlášeným a nepřihlášeným uživatelem. Žádné přidělování práv nebylo nutné řešit.

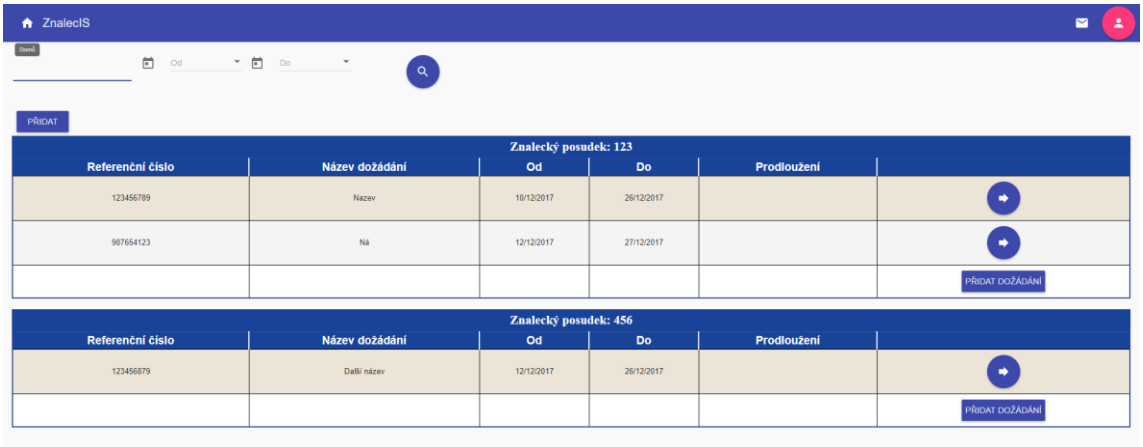
Přihlášený uživatel bude mít administrátorská práva. Bude moci provádět všechny operace v systému a v databázi. Bez přihlášení nebude uživatel schopen s webem nijak komunikovat, kromě zaslání přihlašovacích údajů. Při pokusu načtení jiné webové stránky aplikace bude přesměrován na přihlašovací obrazovku.

3.4 Návrh uživatelského rozhraní

Při tvorbě návrhu obrazovek, byl kladen nárok na co největší uživatelský komfort. Orientace v aplikaci by měla být co nejvíce intuitivní a přehledná. Také by měla usnadnit uživateli zadávání nového posudku. Návrh můžeme rozdělit na pět hlavních stavů.

3.4.1 Úvodní obrazovka

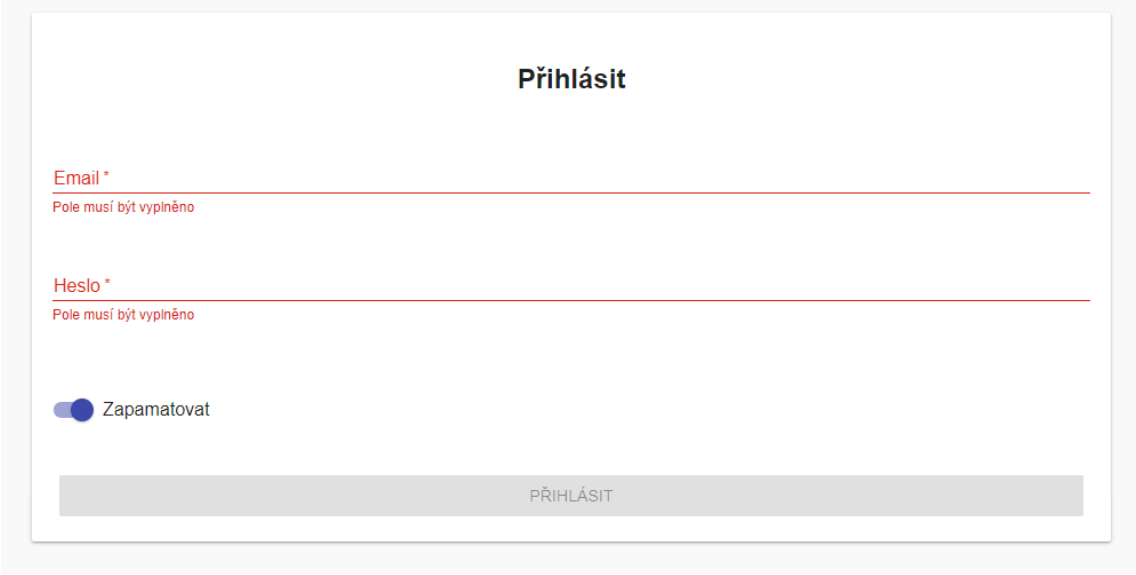
Při spuštění aplikace jsme přesměrováni na úvodní obrazovku. Zde se nachází přehled všech znaleckých posudků a jednotlivých dožádání, která jsou na ně navázaná. Dožádání budou barevně odlišená podle stavu, ve kterém se nacházejí. Obrazovka obsahuje akce na založení nového znaleckého deníku a dožádání. Také si můžeme otevřít detail dožádání. Výsledky bude možné filtrovat pomocí čísla jednacího, názvu a data dožádání. Pokud se obrazovka zobrazí nepřihlášenému uživateli, bude automaticky přesměrován na přihlašovací obrazovku.



Obrázek 2 Úvodní obrazovka

3.4.2 Přihlášení

Pro nepřihlášené uživatele se nejprve zobrazí přihlašovací obrazovka. Bude se zde nacházet formulář pro zadání jména a hesla. Také se zde bude nacházet zaškrtávací tlačítko pro zapamatování hesla a tlačítko pro samotné odeslání formuláře. Po neúspěšných pokusech o přihlášení se zde objeví údaj o uzamknutí účtu.



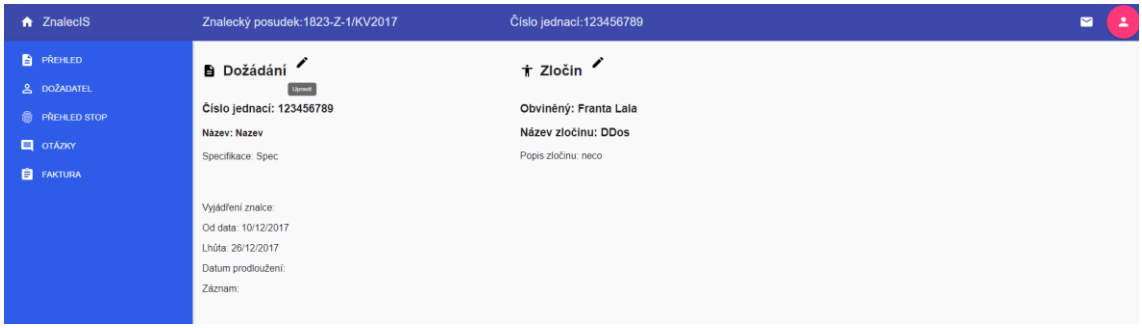
Obrázek 3 Přihlašovací obrazovka

3.4.3 Založení nového dožádání

Obrazovka bude obsahovat několik formulářů na založení nového dožádání. Nejprve založíme úvodní informace, poté vyplníme dožadatele a jeho adresu, nebo vybereme nějakého uloženého. Následně uložíme příslušné stopy a znalecké otázky, které mají být zodpovězeny. Po vyplnění všech údajů a odeslání formulářů budeme přesměrováni do přehledu dožádání.

3.4.4 Přehled dožádání

V přehledu budeme moci nahlédnout do detailů vybraného dožádání. Bude se skládat z několika podkategorií.



Obrázek 4 Obrazovka Přehled

Přehled

Na této podkategorii se bude nalézat detail dožádání a zločinu. Oba bude možné editovat.

Dožadatel

Na této obrazovce bude detail dožadatele a jeho adresa. Také zde bude možná editace.

Přehled stop

V tomto detailu se bude nacházet list stop. Stopy bude možné přidávat, editovat a mazat. Při editaci se dostaneme na novou obrazovku, kde bude možné přidat dokumenty a

soubory. Ty půjdou stáhnout a mazat. Obrázkové soubory se automaticky budou řadit do galerie, která půjde prohlížet.

Obrázek 5 Obrazovka Přehled stop

Otázky

Obrazovka se znaleckými otázkami bude vypadat jako list, do kterého půjde přidávat. Existující otázky půjdou mazat a editovat.

Faktura

V detailu faktury budou vidět její detaily. Také zde budeme moci generovat znalecký posudek.

3.4.5 Nastavení

V obrazovce nastavení budeme moci přidávat a editovat nové vzory. Také zde budeme moci upravovat nastavení aplikace.

3.5 Návrh aplikačního rozhraní

Při návrhu aplikačního rozhraní byl kladen důraz na jednoduchost a přehlednost jednotlivých URL adres. API bylo sestaveno na základě funkčních požadavků. Návrh http byl proveden podle zaběhlých zvyklostí RESTu [4]. Získávání zdrojů pomocí metody

GET. Vytváření a úprava záznamů probíhá pomocí metod POST a PUT. A pro mazání byla použita metoda DELETE.

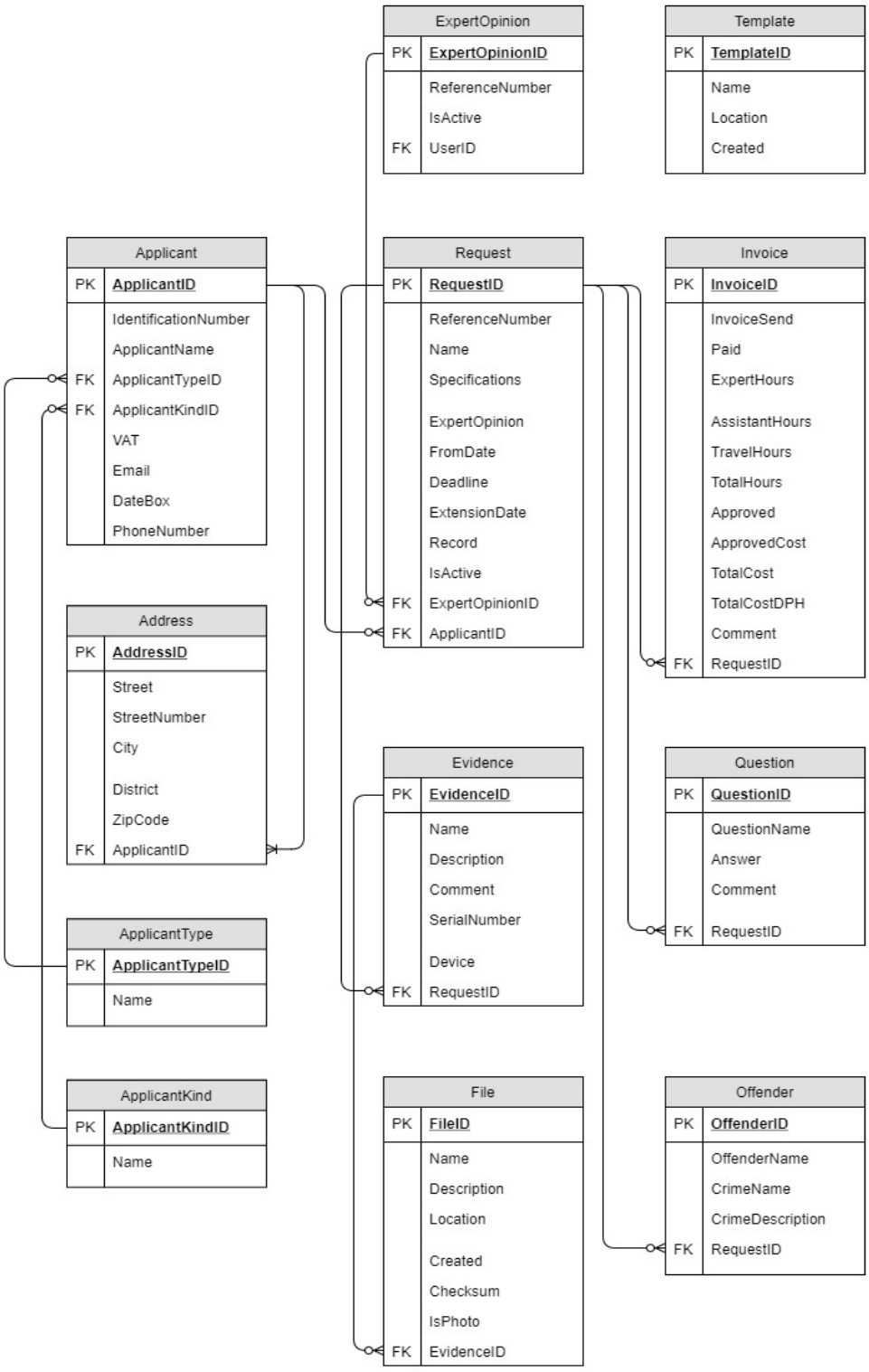
URL	Metoda	Vstupní data	Výstupní data
Api/Login	POST	LoginViewModel	LoginViewModel
Api/Logout	POST		
Api/Applicant/{id}	GET		Applicant
Api/Applicant	POST	Applicant	Applicant
Api/Applicant/{id}	PUT	Applicant	Applicant
Api/Request/{id}	GET		Request
Api/Request	POST	Request	Request
Api/Request/{id}	PUT	Request	Request
Api/GetFile/{id}	GET		List File
Api/UploadFile/{id}	POST		File
Api/CreateFile/{id}	GET		File
Api/DeleteFile/{id}	DELETE		File
Api/DownloadFile/{id}	GET		
Api/Question/{id}	GET		List Question
Api/Question	POST	Question	Question
Api/Question/{id}	PUT	Question	Question
Api/Question/{id}	DELETE		Question
Api/Address/{id}	GET		Address
Api/Address	POST	Address	Address
Api/Address/{id}	PUT	Address	Address
Api/Evidence/{id}	GET		EvidenceList
Api/Evidence	POST	Evidence	Evidence
Api/Evidence/{id}	PUT	Evidence	Evidence
Api/Evidence/{id}	DELETE		Evidence
Api/GenerateOpinion{id}	GET		

Api/Invoice/{id}	GET		Invoice
Api/Invoice	POST	Invoice	Invoice
Api/Invoice/{id}	PUT	Invoice	Invoice
Api/Ares/{id}	GET		Applicant

Tabulka 3 Aplikační rozhraní

3.6 Model databáze

K vyjádření databáze slouží Entitně relační model, který ilustruje vztah mezi jednotlivými entitami v databázi. Rozlišujeme tři typy vztahů: 1:1, 1:M a M:N. Tento model je zejména vhodný pro návrháře nebo správce databáze. Slouží k lepšímu pochopení struktury databáze. Na základě zkušeností autora práce byla vybrána pro ukládání dat relační databáze. Tento druh databáze bezpochyby patří k nejčastějším. Je založena na databázových tabulkách, které mají mezi sebou relační závislost. Pro manipulaci s databází a výběru dat se využívá jazyk SQL.



Obrázek 6 Entitně relační model

4 Implementace

Pro programovací jazyk C# je nejvhodnější vývojové prostředí Visual Studio od Microsoftu. Umožňuje nám psát programy pro různé platformy. Od konzolových a mobilních aplikacích, až po ty webové. Pro vývoj Bakalářské práce byla použita plná verze Visual Studio Enterprise 2017 pro Microsoft Windows.

4.1 Adresářová struktura

Pro větší přehlednost bylo řešení aplikace rozděleno do čtyř projektů Common, Data, Logic a nakonec Web. Projektové řešení bylo uspořádáno tak, aby bylo seřazeno od datové vrstvy po aplikační. Rozdělení na menší projekty nám zjednodušuje orientaci v aplikaci.

4.1.1 Common

V projektu Common se nachází rozšíření datových typů, datové konstanty a statické metody usnadňující nám práci v aplikaci. Projekt by měl být znovu využitelný v budoucnu pro jiné aplikace.

4.1.2 Data

Projekt Data obsahuje datové modely databáze, které jsou vygenerovány pomocí Entity Frameworku. Také se zde nachází objekty určené pro posílání dat na klienta. Tyto objekty jsou serializovány do formátu JSON. Jako poslední věc se tu nachází adresář Sql. V tomto adresáři se nachází inicializační script pro vytvoření databáze a v budoucnu se tu budou nacházet případné skripty na úpravu databáze, pokud dojde k její změně.

4.1.3 Logic

V logické části aplikace se nachází třídy se specifickými výjimkami použitými v aplikaci. Také se tu nalézá třída použitá pro parsování XML služby a třída, která byla využita na generování znaleckého posudku.

4.1.4 Web

Projekt web obsahuje složku wwwroot ve které se nachází statické soubory. Tyto soubory jsou volně ke stažení i pro nepřihlášeného uživatele. Mezi tyto položky patří Css, javascripty a obrázky použité v aplikaci. Dále tu jsou zdroje, ze kterých se načítají texty do obrazovek. Vzory používané k tvorbě znalecké posudku a nakonec konkrétní obrazovky.

Jako další složkou v projektu web jsou kontrolery sloužící k příjmu a odesílání dat z klientské části. Dále tu jsou uloženy dokumenty a soubory k jednotlivým stopám. Také se tu nalézá složka appsettings.json ve které jsou uložená nastavení pro server. V souboru bower.json jsou uložené odkazy na nainstalované javascriptové knihovny. Další soubor se jmenuje bundleconfig.json. Obsahuje css a javascriptové soubory napsané autorem práce. Poté tu máme třídu Program.cs. Zde se nachází funkce pro spuštění serveru. Nakonec projekt obsahuje třídu Startup.cs, ve které se nachází nastavení aplikace a služby, které bude aplikace využívat.

4.2 Vývoj aplikačního rozhraní

Na začátku vývoje bylo nutné vytvořit databázi podle modelu z předchozí kapitoly. Pro vytvoření databáze byl použit program Microsoft SQL Server Management Studio. Tento software nám dovoluje připojit se k příslušnému MS-SQL [13] serveru a pomocí designeru v něm vytvořit novou databázi s jednotlivými tabulkami, prakticky bez pomoci SQL jazyka. Tabulky byly posléze propojeny cizími klíči v databázovém diagramu.

Tato databáze byla využita k vytvoření modelů tříd za pomoci Entity Frameworku [7]. V konzolovém okně Visual Studia pustíme následující příkaz. Zde nastavíme cestu k databázovému serveru a složku, do které se mají modely vytvořit. Pro vygenerování jenom některých tabulek přidáme příkaz –Table a vypíšeme konkrétní tabulky.

```
Scaffold-DbContext
"Server=localhost\sqlexpress;Database=ExpertDB;Trusted_Connection=True
;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

Poté byl projekt napojen na použitou databázi pomocí Connection Stringu, který se nachází ve složce appsettings.json.

4.2.1 Kontroléry server

Serverová část využívá asynchronních kontrolerů z důvodu většího výkonu při zpracování požadavků. Takový kontroler musí v hlavičce metody obsahovat klíčové slovo async. Poté v něm můžeme volat asynchronní metody, které běží ve svém vlákně. Tyto metody obsahuje i Entity Framework, takže můžeme urychlit načítání dat z databáze. Na vrácení výsledku poté počkáme pomocí klíčového await. Kód napsaný pod slovem await se nevykoná, dokud se nevrátí výsledek z předchozí metody.

```
public async Task<IActionResult> GetEvidence([FromRoute] int id)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var evidence = await _context.Evidence
                                  .SingleOrDefaultAsync(m => m.EvidenceId == id);

            if (evidence == null)
            {
                return NotFound();
            }

            return Ok(evidence);
        }
```

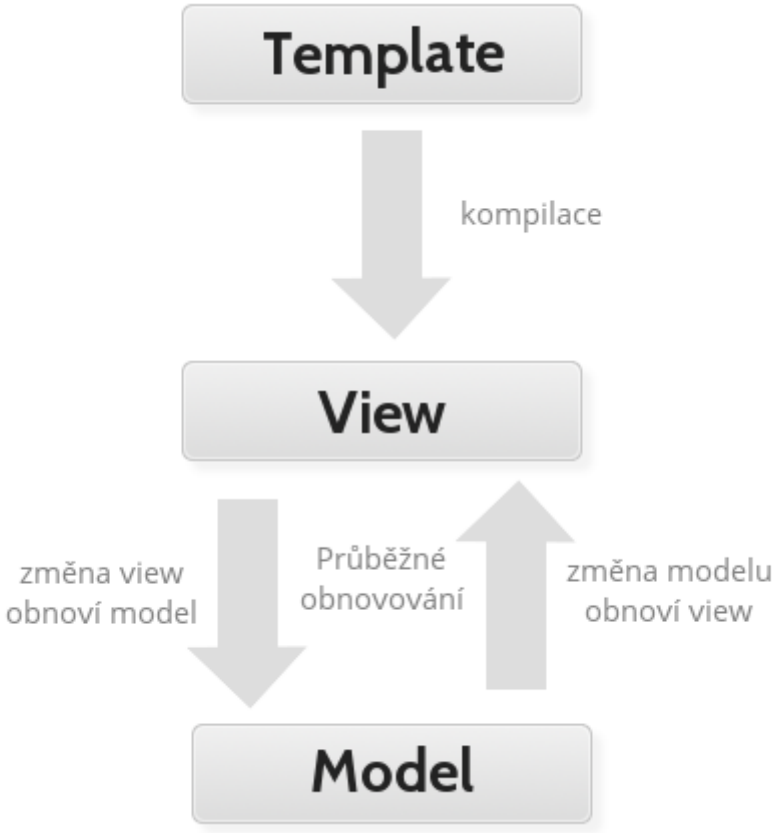
4.3 Vývoj klienta

Pro vývoj klientské části aplikace byl použit framework AngularJS. Tento soubor knihoven můžeme považovat za komplexní aplikaci běžící na straně klienta. Pro inicializaci aplikace musíme nejprve nadefinovat příslušné moduly, které bude aplikace využívat [8].

```
var app = angular.module('AcountApp', [
    'ngRoute',
    'ngAnimate',
    'ngAria',
    'ngMessages',
    'ngMaterial',
    'ngFileUpload',
    'thatisuday.ng-image-gallery',
    'pascalprecht.translate'
]);
```

4.3.1 Kontroléry klient

Aplikace se vždy skládá z jednoho nebo více kontrolérů. Pro přehlednost aplikace má každý vzor nadefinován svůj vlastní kontrolér. Ten je definován jako konstruktor funkce, která je použita jako argument pro scope. Z našeho hlediska je scope prostor kam můžeme vložit informace. Scope je v AngularuJS používán pro oddělení modelu, kontroleru a obrazovky, ale zároveň je udržuje synchronizované [8]. Z toho vyplývá, že po změně dat ve scope se promítnou změny i na obrazovce a obráceně. Tato vlastnost se nazývá Two-way data binding [14].



Obrázek 7 Two-way data binding – převzato z [14]

4.3.2 Routování

Z důvodu menší zátěže serveru je routování řešené na straně klienta. K tomuto účelu je použita knihovna AngularJS Route, která toto umožňuje [8]. Abychom mohli provést routování na straně klienta musíme to nejprve umožnit v nastavení javascriptovém souboru aplikace App.js. Toto nastavení provedeme ve funkci app.config, jak je vidět v následující ukázce.

```
$routeProvider.when('/:requestId/applicant', {
            templateUrl: "../view/applicant.html",
            controller: "applicantCtrl",
            resolve: {
            applicantList: function ($rootScope, $location, applicantService)
              {
                    return applicantService.getApplicants();
```

V podmínce funkce when nastavíme url, při které se má stránka zobrazit. Také můžeme do url přidávat jednotlivé parametry. Ve funkci dále určíme jaký vzor se má zobrazit a jaký kontrolér ho bude ovládat. V našem případě je také určen slib, který musí být splněn než se nám zobrazí nová stránka.

Aby to celé fungovalo, musíme vložit do html kódu speciální tag ng-view. Ten nám určuje místo kde se má stránka změnit, pokud uživatel provede přesměrování na jinou stránku.

```
<div ng-view></div>
```

4.3.3 Service

Services jsou použity pro volání Ajaxových dotazů na stranu serveru.

```
this.getApplicants = function () {
        return $http.get("/api/applicant/" + $rootScope.requestID)
              .then(function (response) {
                    return response.data;
              }, function (response) {
                    return response.statusText;
              });
  }
```

4.3.4 Validace formulářů

Validace formulářů je nejprve řešena na straně klienta pomocí Framewoku AngularJS. To nám zajišťuje rychlou kontrolu dat vložených uživatelem, ještě předtím než odešleme data na server. Data jsou poté na serveru kontrolována ještě jednou, protože zkušený uživatel může validaci na straně klienta odstranit a zaslat nám nesprávná data, která by aplikaci mohla narušit.

Ve formuláři jsou data svázána pomocí direktivy ng-model, která nám zajišťuje zaslání dat do kontroleru, abychom s nimi mohli dále pracovat. Dále je zde využita direktiva ng-submit, pro potvrzení formuláře a zavolání příslušné funkce. Tato funkce lze zavolat jen pokud jsou splněny podmínky validace. V ukázce můžeme vidět nestandardní

html tagy, které jsou součástí frameworku AngularJS Material a zajišťují nám tvorbu GUI aplikace. Pro zobrazení validačních hlášek je použita direktiva ng-messages [9].

```
<form name="loginForm" ng-submit="logIn()" novalidate layout="column">
                <md-input-container>
                    <label for="emailInput" translate="Email">Email</label>
                    <input id="emailInput" type="text" name="email"
                           ng-model="login.email" ng-required="true" />
                    <div ng-messages="loginForm.email.$error">
                          <div ng-message="required"
            translate="Error.Required">Pole musi byt vyplneno</div>
                    </div>
                </md-input-container>
                <md-input-container>
                    <label for="passwordInput"
                    translate="Password">Password</label>
                    <input id="passwordInput" type="password" name="password" ng-
                    model="login.password" ng-required="true" />
                    <div ng-messages="loginForm.password.$error">
                            <div ng-message="required"
                    translate="Error.Required">Pole musi byt vyplneno</div>
                    </div>
                </md-input-container>
                <md-input-container>
                      <md-switch class="md-primary" ng-
            model="login.rememberMe"><span
            translate="RememberMe">Zapamatovat</span></md-switch>
                </md-input-container>
                       <md-button class="md-raised md-primary" ng-
            disabled="loginForm.$invalid" type="submit"
            translate="Button.Login">Přihlásit</md-button>
        </form>
```

4.3.5 Multijazyčnost

Požadavek na jazyk v aplikaci byla čeština, ale pro případné rozšíření byla přidána i podpora anglického jazyka. V moderních aplikacích se texty ukládají na jednom místě, kde jsou uloženy v jednom nebo více zdrojových souborů. Toto řešení utváří větší přehled v textech a redukuje jejich opakování. Hlavní výhoda vzniká v případě opakujících se textů. Tyto texty, můžeme jednoduše změnit centrálně aniž bychom je museli po jednom dohledávat.

Z důvodu menší zátěže serveru to za nás řeší javascriptová knihovna Angular-translate. Při prvním zobrazení aplikace knihovna vyhodnotí uložený nebo defaultně

nastavený jazyk. Poté se stáhnou ze serveru příslušné zdroje ve formátu JSON a zobrazí se v prohlížeči. Jazyky můžeme libovolně měnit v nastavení aplikace.

4.4 Bezpečnost

Zabezpečení je u webového serveru, který je volně přístupný z internetu a obsahuje citlivé záznamy, kritická věc. Z tohoto důvodu bylo použito hotové řešení od Microsoftu, které by mělo odpovídat kladeným požadavkům na bezpečnost. V ostrém provozu musí být samozřejmě komunikace se serverem zašifrovaná, abychom předešli odposlechu dat.

4.4.1 ASP.NET Identity

ASP.NET Identity je bezpečnostní řešení od Microsoftu pro autentizaci a autorizaci uživatelů [15]. Zajišťuje nám několik způsobů přihlášení. Základem je vytvoření účtů v databázi, ale také můžeme využít přihlášení přes účet od Microsoftu. Také je možnost přihlásit se přes účty třetích stran jako je Facebook, Google a Twitter. Toto řešení nebylo potřeba použít. Na základě uživatelského jména nebo uživatelské role nám přidělí práva pro používání systému. Pro použití Microsoft Identity musíme doinstalovat příslušné knihovny a připsat do třídy Startup.cs následují kód.

```
services.AddIdentity<ApplicationUser, IdentityRole>()
                .AddEntityFrameworkStores<ApplicationDbContext>()
                .AddDefaultTokenProviders();
```

Po prvním přihlášení do aplikace nám vygeneruje databázi nebo najde už existující databázi, na kterou má odkaz. Tato databáze je tvořena několika tabulkami, ze kterých je nejdůležitější tabulka uživatele, ve které jsou za hešovaná hesla pomocí algoritmu HMACSHA256 [15]. Tento algoritmus je v současné době považován za neprolomený. Pokud chceme po serveru, aby po uživateli požadoval autentizaci, musíme nad kontrolerem napsat klíčové slovo Autorize. Server poté ověří uživatelovo session id a pokud souhlasí, zpřístupní mu požadovaná data.

Při třech neúspěšných pokusech o přihlášení bude uživatelský účet na nějakou dobu zamknut na úrovni databáze. V tomto případě je uživatel přesměrován na obrazovku pro uzamčení uživatele. V nastavení, také máme možnost vnutit uživateli požadovanou sílu hesla.

4.5 Správa souborů

V sekci Stopy bylo třeba vyřešit kromě uložení údajů do databáze také nahrávání a uložení souborů na serveru. Existují dva způsoby. V prvním ukládáme celý soubor do databáze a v druhém do databáze uložíme jenom jeho odkaz na soubor uložený na disku. Byl vybrán druhý způsob, i za předpokladu, že můžou vznikat nekonzistence v datech, pokud smažeme pouze jednu část. V porovnání s první metodou je tento způsob rychlejší a zabírá o dost méně místa v databázi. To nám přijde vhod, protože budeme mít omezenou velikost databáze.

Na serveru jsou dokumenty uloženy ve složce, která není z internetu přístupná. Jdou stáhnout pouze přes aplikační rozhraní jako přihlášený uživatel.

4.5.1 Nahrání na server

Pro nahrání souborů na server bylo použito asynchronních požadavků. Uživatel vybere soubor z disků a poté ho může nahrát na server. Pro usnadnění práce na straně klienta byla použita knihovna ng-upload, která nám poskytuje vícero možností pro úpravu souborů a má vlastní service na nahrávání souborů.

```
Upload.upload({
            url: "/api/uploadfile/" + $routeParams.evidenceId,
            data: { file: files }
        })
```

Na straně serveru poté soubor přečteme přes třídu StreamReader, vytvoříme z něj hash sha256 a uložíme údaje o souboru do databáze. Poté soubor uložíme na disk.

4.5.2 Stažení ze serveru

Pro stažení souboru z databáze nejprve podle id zkontrolujeme, jestli soubor v databázi existuje a poté zavoláme metodu, která soubor načte z disku a odešle na klienta.

```
public FileResult DownloadFile(string filePath, string pathDetail, string
fileName)
        {
            IFileProvider provider = new PhysicalFileProvider(filePath +
       pathDetail);
            IFileInfo fileInfo = provider.GetFileInfo(fileName);
            if (!fileInfo.Exists)
            {
                throw new Exception("File not found");
            }
            var types = this.GetMimeTypes();
            var mimeType = Path.GetExtension(fileName).ToLowerInvariant();
            var readStream = fileInfo.CreateReadStream();
            return File(readStream, types[mimeType], fileName);
        }
```

4.6 Generování znaleckého posudku

Jednou z vlastností informačního systému je generování znaleckého posudku ze získaných dat. Posudek bude vygenerovaný ve formátu Microsoft Word. Jazyk C# je úzce spjatý s technologiemi od Microsoftu jako je Microsoft Office. Proto existuje oficiální knihovna na generování dokumentů OpenXML [16].

4.6.1 Knihovna OpenXML

V roce 2007 změnil Microsoft formátování všech Office dokumentů na ukládání souborů v XML. U jakéhokoliv souboru můžeme přepsat jeho koncovku na .zip a bude otevřen jako archiv, kde jsou uloženy jednotlivé části dokumentu ve formátu XML. Hlavním cílem pro zavedení nového formátu byla zpětná kompatibilita a splnění podmínek standardizačního procesu [16]. Pro informační systém byl nejdůležitější formát Word, ve kterém bude vygenerován výsledný posudek. Tělo dokumentu je uspořádáno podle

schématu body-paragraph-run. Do tagu run poté vkládáme text, obrázek nebo tabulku. V paragraph a run také nastavujeme formátování textu.

4.6.2 Řešení vzorů

V našem případě bylo nutné vyřešit, že znalecké posudky se mohou lišit. Bylo navrženo následující řešení. Na server nahrajeme vzorový dokument, který bude obsahovat klíčová slova. Tato slova aplikace vyhledá a nahradí je za data z databáze. Tento postup nám dovoluje libovolný počet dokumentů obsahující klíčová slova. Jednotlivé vzory můžeme libovolně mazat a nahrazovat za jiné. Názvy klíčových slov jsou popsány v příloze.

```
foreach (var paragraph in element.Descendants<Paragraph>())
            {
                Regex regex = new Regex(searchText);
                Match match = regex.Match(paragraph.InnerText);
                if (match.Success)
                {
                    foreach (var question in questionList)
                    {
                        paragraph.InsertAfterSelf(new Paragraph(new Run(new
Text(question.Question))));
                        paragraph.InsertAfterSelf(new Paragraph(new Run(new
Text(question.Answer))));
                    }
                    paragraph.Remove();
                }
            }
```

4.7 Komunikace s ARES

Pro usnadnění přístupu nového uživatele, byla navrhnuta komunikace s portálem Ministerstva vnitra ARES [17]. Tento systém zajišťuje zpřístupnění údajů o vedení registrů a evidencí veřejné správy o ekonomických subjektech. Služba je napsaná v rozhraní XML a je veřejně přístupná bez nutnosti registrace. Také obsahuje určitá omezení:

- odeslání více než 1000 dotazů v době od 8:00 do 18:00
- odeslání více než 5000 dotazů v době od 18:00 do 8:00 rána následujícího dne

- pokus o porušení ochrany Ministerstva financí
- opakované posílání stejných nebo nesprávně vyplněných dotazů
- větší počet současně zadaných dotazů
- obcházení omezení dotazů z většího množství IP adres
- automatizované propátrávání databáze

V mém řešení uživatel odešle dotaz, do kterého zadá IČO nebo název příslušného orgánu, který chce najít. Tento požadavek se odešle na server, který vytvoří dotaz na informační server ARES. Pro získání informací které potřebujeme, nám stačí základní dotaz na výpis informačních údajů pomocí metody GET [17].

Údaje, které obdržíme ze serveru jsou ve formátů XML a musíme je nejprve ručně rozparsovat. S tímto nám pomůže .netová třída XmlReader, která umí číst XML soubory. Poté co vytvoříme z XML objekty, můžeme je poslat ve formátu JSON na klientskou část. Data nám poté pomůžou doplnit formulář. Tato metoda nám řeší problém s Cross-Origin Resource Sharing, který nám z prohlížeče zabraňuje posílání dotazů na jinou doménu.

4.9 Komunikace s datovou schránkou

Díky datovým schránkám se nám zásadně změnil způsob doručování úředních dokumentů. Nyní můžeme zasílat a přijímat dokumenty v elektronické podobě orgánům veřejné moci. Datové schránky nám mohou plně nahradit klasický způsob zasílání dokumentů v listinné podobě, protože zákon o datových schránkách zrovnoprávňuje papírovou a elektronickou verzi zasílaného dokumentu. Z toho vyplívá, že posílání dokumentů přes datové schránky je rychlejší, spolehlivější a levnější veřejná správa. Datovou schránku lze bezplatně zřídit na kontaktním místě veřejné správy Czech POINT [18].

Aplikační rozhraní datových schránek může využívat každý registrovaný uživatel. Toto rozhraní je popsáno v provozním řádu datových schránek.

Komunikaci s datovou schránkou se nepodařilo navázat kvůli špatné chybě při přihlašování. Chyba byla pravděpodobně způsobená špatným certifikátem. Také to mohlo být způsobeno špatnou přehledností a složitostí aplikačního rozhraní webové služby. Chybu se zatím nepodařilo vyřešit.

Ještě existuje komerční řešení, které by mohlo komunikaci s datovou schránkou zjednodušit, ale z finančních důvodů nebylo pro aplikaci vhodné.

4.10 Nasazení

Použitý framework ASP.NET Core byl navržen pro multiplatformní použití, ale z důvodu větší podpory od strany Microsoftu bylo zvoleno řešení založené na Microsoft technologiích. Jako webový server byl použit IIS 10.0 Express. K tomu byl přibrán databázový server MS-SQL Express 2017. Verze Express byla vybrána z důvodu bezplatné licence. Obsahuje určitá omezení, ale pro naše účely by měl její výkon být dostačující. Servery běží na operačním systému Microsoft Windows 10.

Nejprve byl na operační systém nainstalován příslušný software, který budeme využívat. Také nesmí chybět nainstalované SDK .NET Core. Po instalaci databázového serveru musíme spustit inicializační script, který nám vytvoří prázdnou databázi. Dále musíme změnit v souboru appsetting.json Connection string na námi vytvořenou databázi. Poté nahrajeme aplikaci na webový server.

Funkčnost serveru byla ověřena, v prohlížečích Chrome, Firefox a Edge. Internet Explorer nebude podporován.

5 Závěr

Hlavním cílem této bakalářské práce byla tvorba Informačního systému pro soudního znalce.

V teoretické části byla rozebrána problematika zásad znalecké činnosti. Bylo zjištěno, jaké náležitosti musí obsahovat znalecký posudek. Také jak funguje znalecký deník. Nakonec byla provedena analýza požadavků na Informační systém.

V další části byl proveden návrh aplikace. Správný návrh nám může zjednodušit samotnou implementaci. Nejprve byla zvolena architektura, podle které se bude aplikace řídit. Poté byly, zvoleny technologie, které budou využity pro vývoj aplikace. Výběr byl proveden podle nároků na aplikaci a znalostí autora s vybranými technologiemi. Podle analýzy bylo vytvořeno aplikační rozhraní s jednotlivými metodami a následně byl vytvořen návrh Entitně relačního modelu databáze.

Samotná implementace vycházela z výsledků analýzy a návrhů z předchozích částí Bakalářské práce. Nejprve byla implementována serverová část aplikace. Pomocí Microsoft SQL Server Management Studio byla vytvořena databáze, která byla pomocí Entity Frameworku převedena na objekty. Poté byly implementovány kontroléry pomocí návrhu aplikačního rozhraní z předchozí kapitoly. Klientská část aplikace byla napsána v javascriptovém frameworku AngularJS. Tato technologie byla zvolena na základě dobré podpory při tvorbě Single-page aplikací. Pro tvorbu GUI byl zvolen Framework AngularJS Material, pro svou dobrou interakci s AngularJS.

Bezpečnost aplikace byla zajištěna pomocí frameworku ASP.NET Identity. Toto oficiální řešení od Microsoftu nám poskytuje autentizaci a autorizaci uživatelů. Také se stará o hešování hesel za pomoci algoritmu HMACSHA256.

Přenos souborů na server, byl zajištěn pomocí asynchronních požadavků. Soubory jsou poté uloženy do složek, a podrobné údaje jsou uloženy v databázi. Přístup k souborům byl omezen pouze pro přihlášené uživatele. Dále byla zajištěna komunikace

s webem státní správy ARES. Také byla snaha o navázání komunikace s Datovou schránkou. Bohužel toho nebylo dosaženo z důvodu problémů s aplikačním rozhraním.

Na závěr bylo provedeno aplikování serveru a ověření jeho funkčnosti. Webový server IIS běží na operačním systému Windows 10 a je napojen na databázi MS-SQL Express.

Literatura

[1] ČESKÁ REPUBLIKA. *Zákon o znalcích a tlumočnících*. In: . Praha, 1967, ročník 1967, číslo 36. Dostupné také z: https://www.zakonyprolidi.cz/cs/1967-36#oddil2

[2] ČESKÁ REPUBLIKA. *Vyhláška ministerstva spravedlnosti k provedení zákona o znalcích a tlumočnících*. In: . Praha, 1967, ročník 1967, číslo 37. Dostupné také z: https://www.zakonyprolidi.cz/cs/1967-37

[3] Single-page Application. *Codeschool* [online]. Orlando: Code school, 2017 [cit. 2017-12-09]. Dostupné z: https://www.codeschool.com/beginners-guide-to-web-development/single-page-applications

[4] REST. *TechTarget* [online]. Newton: TechTarget, 2016 [cit. 2017-12-09]. Dostupné z: http://searchmicroservices.techtarget.com/definition/RESTful-API

[5] ASP.NET Core. *Microsoft* [online]. Redmond: Microsoft, 2017 [cit. 2017-12-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/

[6] *ASP.NET* [online]. Redmond: Microsoft, 2007 [cit. 2017-12-09]. Dostupné z: https://www.asp.net/

[7] Entity Framework Core. *Microsoft* [online]. Redmond: Microsoft, 2017 [cit. 2017-12-09]. Dostupné z: https://docs.microsoft.com/en-us/ef/core/

[8] *AngularJS* [online]. Mountain View: Google, 2009 [cit. 2017-11-08]. Dostupné z: https://angularjs.org/

[9] *AngularJS Material* [online]. Mountain View: Google, 2014 [cit. 2017-12-09]. Dostupné z: https://material.angularjs.org/latest/

[10] *Bower* [online]. San Francisco: Twitter, 2012 [cit. 2017-12-09]. Dostupné z: https://bower.io/

[11] *GitLab* [online]. San Francisco: GitLab, 2011 [cit. 2017-11-07]. Dostupné z: https://about.gitlab.com/

[12] *IIS* [online]. Redmond: Microsoft, 2013 [cit. 2017-11-12]. Dostupné z: https://www.iis.net/

[13] MS-SQL. *Microsoft* [online]. Redmond: Microsoft, 2016 [cit. 2017-12-09]. Dostupné z: https://docs.microsoft.com/cs-cz/sql/sql-server/editions-and-components-of-sql-server-2017

[14] Itnetwork. *Itnetwork* [online]. Praha: Itnetwork, 2017 [cit. 2017-12-09]. Dostupné z: https://www.itnetwork.cz/javascript/angularjs/javascript-tutorial-uvod-do-angularjs

[15] ASP.NET Identity. *Asp.net* [online]. Redmond: Microsoft, 2013 [cit. 2017-12-09]. Dostupné z: https://www.asp.net/identity

[16] Open XML. *Microsoft* [online]. Redmond: Microsoft, 2007 [cit. 2017-12-09]. Dostupné z: https://msdn.microsoft.com/cs-cz/library/office/bb448854.aspx

[17] *ARES* [online]. Praha: Ministerstvo financí ČR, 2013 [cit. 2017-11-15]. Dostupné z: http://wwwinfo.mfcr.cz/ares/

[18] *Datová schránka* [online]. Praha: Ministerstvo Vnitra ČR, 2009 [cit. 2017-12-05]. Dostupné z: https://www.datoveschranky.info/

Seznam obrázků

Seznam tabulek

Přílohy

Přiložené CD

Na přiloženém CD se nachází soubory v následující adresářové struktuře.

- Havel_BP_2017.pdf – Plné znění Bakalářské práce ve formátu PDF
- Informační systém
    - ZnalecIS.zip – Výsledná aplikace spustitelná ve Visual Studiu
    - InitScript.sql - Inicializační script pro vytvoření databáze

Použitý software

Microsoft Word - https://products.office.com/cs-cz/word

Microsoft Visual Studio - https://www.visualstudio.com/cs/

Microsoft SQL Server Management Studio - https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms

IIS Express - https://www.microsoft.com/cs-cz/download/details.aspx?id=48264

MS-SQL Express - https://www.microsoft.com/cs-cz/sql-server/sql-server-editions-express

.NET Core - https://github.com/dotnet/core

ASP.NET Core - https://github.com/aspnet/Home

Entity Framework Core - https://github.com/aspnet/EntityFrameworkCore

Microsoft Identity - https://github.com/aspnet/Identity

AngularJS - https://angularjs.org/

AngularJS Material - https://material.angularjs.org/latest/

Nuget Packages - https://www.nuget.org/

Bower - https://bower.io/

Spuštění aplikace

Aplikace se nachází na přiloženém CD nebo na gitovém uložišti GitLab. Pro stažení z GitLabu přes příkazovou řádku můžeme využít aplikaci git nebo projekt stáhnout z webové stránky v zazipovaném souboru. Ke spuštění aplikace nejprve potřebujeme mít nainstalované Visual Studio, IIS a MS-SQL server. Za prvé otevřeme script a změníme cestu kam se má databáze vytvořit. Poté v MS-SQL script spustíme. To nám vytvoří novou znaleckou databázi v adresáři, který jsme určili. Dále otevřeme soubor ve Visual Studiu. Po načtení projektu musíme přepsat connection string na námi vytvořenou databázi v souboru appsetting.json. Tento soubor se nachází v projektu ZnalecIS.Web. Na závěr můžeme pustit projekt přímo ve Visual studiu nebo ho vydat na IIS server.

Klíčová slova pro generování posudku

Klíčová slova můžeme rozdělit na dvě kategorie:

Jednotlivá slova

- ReferenceNumberEIS
- NameEIS
- SpecificationsEIS
- ExpertOpinionEIS
- FromDateEIS
- DeadlineEIS
- ExtensionDateEIS
- RecordEIS
- OffenderNameEIS

- CrimeNameEIS
- CrimeDescriptionEIS
- IdentificationNumberEIS
- ApplicantNameEIS
- VatEIS
- EmailEIS
- DateBoxEIS
- PhoneNumberEIS
- StreetEIS
- CityEIS
- DistrictEIS
- ZipCodeEIS

Listy

- EvidenceListEIS
- QuestionListEIS