Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Vývoj webové aplikace pro evidenci smluv

Bc. Oleksandr Kutsyn

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Oleksandr Kutsyn

Systémové inženýrství a informatika
Informatika

Název práce

Vývoj webové aplikace pro evidenci smluv

Název anglicky

Development of a contract management web application

Cíle práce

Hlavním cílem této diplomové práce je provést návrh a implementaci webové aplikace pro evidenci smluv. Aplikace bude poskytovat funkce potřebné pro evidovaní smluv v podniku. Pro implementaci této aplikace budou využity aktuální technologie pro vývoj softwaru.

V teoretické části práce bude cílem popsat proces evidence smluv a sestavit specifikaci zadavatele. Následujícím krokem je popsaní vybrané existující aplikace umožnující provedeni evidence smluv a popsat technologie které budou využity pro implementaci.

Metodika

Znalosti a informace potřebné pro splnění stanovených cílů budou čerpány z odborné literatury a dokumentace použitých jazyků/frameworků, také z technických publikací na webových portálech. Tyto zdroje poslouží jako podklady pro tvorbu teoretické časti práce.

V praktické častí bude provedena analýza zadavatelské specifikace ze které následně bude formulovány požadavky na funkcionalitu aplikace. Z těchto požadavků vyplyne prvotní návrh aplikace, na jehož základě bude provedena samotná implementace. Pro realizaci budou využity podklady z teoretické časti. Backend bude vyvíjen pomoci programovacího jazyka PHP a frameworku Laravel, frontendová část bude rozšířena o Bootstrap a DataTables, kde jako spojující prvek bude využit jazyk JavaScript.

Aplikace bude nasazena a otestována. Poznatky z jejího vývoje a zpětná vazba z testování budou shrnuty a budou navrženy případné další možnosti budoucího rozvoje.

Doporučený rozsah práce

60-80 stran

Klíčová slova

PHP, Laravel, JavaScript, Bootstrap, DataTables, Framework

Doporučené zdroje informací

Bootstrap [online]. [cit. 2021-05-19]. Dostupné z: https://getbootstrap.com/docs/4.0/
DataTables [online]. [cit. 2021-05-19]. Dostupné z: https://datatables.net/manual/
JavaScript [online]. [cit. 2021-05-19]. Dostupné z: https://devdocs.io/javascript/
Laravel [online]. [cit. 2021-05-19]. Dostupné z: https://laravel.com/docs/5.7/
PHP [online]. [cit. 2021-05-19]. Dostupné z: https://www.php.net/manual/en/

Předběžný termín obhajoby

2021/22 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 30. 03. 2022

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Vývoj webové aplikace pro evidenci smluv“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.03.2022 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) Ing. Jiřímu Brožkovi, Ph.D., za odborné vedení při tvorbě této diplomové práce.

Vývoj webové aplikace pro evidenci smluv

Abstrakt

Tato diplomová práce se věnuje vývoji aplikace pro evidenci smluv, implementace byla provedena ve formě webové aplikace. Diplomová práce se také věnuje tvorbě výchozí analýzy a následné tvorbě zadání aplikace.

Architektura této webové aplikace byla navržena a vyvinuta s využitím aktuálních technologií pro vývoj softwaru. Vizuální část aplikace byla implementovaná pomoci HTML, CSS, Bootstrap a JavaScript. Logická část byla implementovaná pomocí PHP frameworku Laravel. Pro ukládání souboru byla využita SOAP spisová služba, veškerá data, se kterými pracuje aplikace, jsou uložena v ORACLE SŘBD.

Klíčová slova: PHP, Laravel, JavaScript, Bootstrap, DataTables, Framework

Vývoj webové aplikace pro evidenci smluv

Abstract

This diploma thesis deals with the development of a contract management web application, the implementation was carried out in the form of a web application. The diploma thesis also deals with the creation of the initial analysis and the subsequent creation of the application assignment.

The architecture of this web application was designed and developed using current software development technologies. The visual part of the application was implemented using HTML, CSS, Bootstrap and JavaScript. The logical part was implemented using the Laravel PHP framework. The SOAP file service was used to store the files, all data with which the application works are stored in ORACLE DBMS.

Keywords: PHP, Laravel, JavaScript, Bootstrap, DataTables, Framework

Obsah

1 Úvod

Nedílnou součástí administrativy všech právních subjektů jsou smlouvy. K jejich efektivní evidenci a archivaci je třeba vytvořit interní systém správy, například ve formě softwaru – webové aplikace – pro evidenci smluv. Software pro evidenci (správu) smluv zjednodušuje, digitalizuje a zefektivňuje způsob, jakým jsou smlouvy spravovány. Software pro správu smluv je používán k vytváření, podepisování, obnovování a shromažďování dat o smlouvách. Je navržen tak, aby nahradil manuální správu smluv.

Web je unikátní platformou pro vývoj softwaru. Výhodou webových aplikací je fakt, že se mohou dostat v podstatě ke komukoli, kdekoli a na jakémkoli zařízení. „Při realizaci formou webové aplikace se zbavíte nutnosti software pro evidenci smluv instalovat na jednotlivé počítače. K jeho spuštění vám postačí internetový prohlížeč, například Google Chrome, a připojení k internetu.“ (1) Požadované informace pak budou vždy k dispozici. V současné době je nezbytností elektronická evidence smluv a jejich příloh s jednoznačnou identifikací i veškerými detailními evidenčními údaji, s možností zveřejňování smluv v Registru smluv MVČR prostřednictvím datových schránek, evidencí vlastních položek u smluv, s filtry ve vyhledávání a možností řazení smluv do skupin, upozorňováním na termíny, možností nastavení různých úrovní oprávnění a generování úkolů ke smlouvám, možností přidávání elektronických příloh a poznámek ke smlouvám, přidání elektronického podpisu apod. Webová aplikace poskytuje přehled ve smlouvách a dokumentech včetně jejich dodatků a příloh. Aplikace je vhodná pro evidenci a správu různých typů smluv a dokumentů. Aby byla webová aplikace efektivním nástrojem pro evidenci smluv, je pro ni stěžejní jednoduchost, přehlednost a možnost rychlého a snadného vyhledávání.

2 Cíl práce a metodika

2.1. Cíl práce

Hlavním cílem této diplomové práce je provést návrh a implementaci webové aplikace pro evidenci smluv. Aplikace bude poskytovat funkce potřebné pro evidování smluv v podniku. Pro implementaci této aplikace budou využity aktuální technologie pro vývoj softwaru.

V teoretické části práce bude cílem popsat proces evidence smluv a sestavit specifikaci zadavatele. Následujícím krokem je popsání vybrané existující aplikace umožnující provedení evidence smluv a popsat technologie, které budou využity pro implementaci.

2.2. Metodika

Znalosti a informace potřebné pro splnění stanovených cílů budou čerpány z odborné literatury a dokumentace použitých jazyků/frameworků, také z technických publikací na webových portálech. Tyto zdroje poslouží jako podklady pro tvorbu teoretické části práce.

V praktické části bude provedena analýza zadavatelské specifikace, ze které následně budou formulovány požadavky na funkcionalitu aplikace. Z těchto požadavků vyplyne prvotní návrh aplikace, na jehož základě bude provedena samotná implementace. Pro realizaci budou využity podklady z teoretické části. Backend bude vyvíjen pomocí programovacího jazyka PHP a frameworku Laravel, frontendová část bude rozšířena o Bootstrap a DataTables, kde jako spojující prvek bude využit jazyk JavaScript.

Aplikace bude nasazena a otestována. Poznatky z jejího vývoje a zpětná vazba z testování budou shrnuty a budou navrženy případné další možnosti budoucího rozvoje.

3 Teoretická východiska

Co se týká struktury teoretické části diplomové práce, první kapitola se věnuje problematice webových aplikací s konkrétním zaměřením na značkovací jazyk HTML a kaskádové styly CSS, skriptovací jazyky, frameworky, frontend (klientská strana aplikace) a backend. Následující kapitola teoretické části je věnována technologiím použitým v praktické části: PHP (Hypertext Preprocessor, hypertextový preprocesor – skriptovací programovací jazyk určený primárně pro programování dynamických webových stránek a aplikací), Laravel (serverový PHP framework, s jehož pomocí je možné vytvářet kompletní aplikace), architektura MVC (Model-View-Controller jako architektonický vzor v softwarovém designu běžně používaný k implementaci uživatelských rozhraní, dat a řídící logiky), SQL (Structured Query Language – standardizovaný dotazovací jazyk používaný ke správě relačních databází a provádění různých operací s daty v relačních databázích, které jsou založeny na relačním modelu sdružujícím data do tabulek – relací), Bootstrap (frontend framework pro rychlé a snadné vytváření webových šablon), DataTables (zásuvný modul, plugin, pro javascriptovou knihovnu jQuery), OracleSQL (integrované vývojové prostředí, které zjednodušuje vývoj a správu databáze Oracle v tradičním i cloudovém nasazení), JavaScript (objektově orientovaný, multiplatformní a událostmi řízený skriptovací jazyk), AJAX (Asynchronous JavaScript and XML), SOAP (Simple Object Access Protocol pro posílání zpráv ve formátu XML mezi dvěma aplikacemi (princip peer-to-peer)), Document Object Model (DOM, objektový model dokumentu, který umožňuje přistupovat k jednotlivým objektům XML – XHTML – dokumentu a pracovat s nimi) a virtuální DOM, WSDL (Web Services Description Language) a principy vývoje YAGNI, KISS a DRY.

3.1. Webová aplikace

Webová aplikace je místo na webu obsahující stránky s částečně nebo úplně neurčeným obsahem. Finální obsah stránky je dostupný až v okamžiku, kdy její návštěvník požádá o stránku z webového serveru. Finální obsah stránky je závislý na akcích návštěvníka a liší se pro různé požadavky – jde o tzv. dynamickou webovou stránku. U dynamických webových stránek mění obsažené informace uživatel disponující přístupovými právy do systému, přičemž změny jsou jednoduché a probíhají rychle. Naproti tomu statické webové stránky jsou vhodné v případě, že nebudou často aktualizovány, úpravy a změny obsahu těchto webových stránek totiž trvají delší dobu. (1)

Základním účelem webových aplikací je řešit problémy a úlohy. Webová aplikace je software, který využívá webové prohlížeče a webovou technologii k provádění úkolů a řešení problémů přes internet. Internet je nákladově efektivní komunikační kanál, který umožňuje vyměňovat informace a provádět rychlé a bezpečné transakce. Efektivní zapojení nicméně souvisí se schopností zachytit a uložit všechna potřebná data a aplikovat vhodné prostředky ke zpracování těchto informací a k prezentaci výsledků uživateli.

Webové aplikace používají kombinaci skriptů na straně serveru (PHP a ASP) k ukládání a získávání informací a skriptů na straně klienta (JavaScript a HTML) k prezentaci informací uživatelům. To umožňuje uživatelům komunikovat prostřednictvím online formulářů, systémů pro správu obsahu, nákupních košíků apod. Kromě toho aplikace umožňují vytvářet dokumenty, sdílet informace, spolupracovat na projektech a na sdílených dokumentech bez ohledu na jejich umístění či použité zařízení. (2)

Webová aplikace je poskytována prostřednictvím internetu, případně intranetu (vnitropodniková obdoba), přičemž funkci klienta zastává webový prohlížeč. Webové aplikace jsou implementovány jako informační systémy, internetové obchody, diskusní fóra, freemaily apod. U aplikací postavených na standardních funkcích prohlížeče je výhodou jejich multiplatformita – tzn. aplikaci lze spustit na kterémkoli operačním systému a kterékoli jeho verzi, výhodou pro uživatele jsou též minimální náklady a údržba. Naproti tomu jejich nevýhodou je závislost na poskytovateli aplikace, který může omezovat její využívání apod. Co se týká její struktury, první vrstvou webové aplikace je tzv. prezentační vrstva – webový prohlížeč, střední (logickou) vrstvou jsou nástroje pro dynamické generování stránek, třetí (datovou) vrstvou jsou databáze. Prohlížeč vysílá

požadavky do střední vrstvy, kde jsou zpracovány. Vrstvy mohou být na různých místech, případně na jediném webovém serveru. (3)

Tok webové aplikace probíhá následujícím způsobem: uživatel odešle požadavek na webový server přes internet, buď prostřednictvím webového prohlížeče, nebo uživatelského rozhraní aplikace, webový server předá tento požadavek příslušnému webovému aplikačnímu serveru, webový aplikační server provede požadovaný úkol (například dotaz na databázi nebo zpracování dat) a následně vygeneruje a odešle na webový server výsledky s požadovanými informacemi nebo zpracovanými daty. Webový server odpoví klientovi požadovanou informací, která se poté zobrazí na displeji uživatele. Nejnovějším trendem v oblasti tvorby internetových stránek jsou tzv. progresivní webové aplikace (PWA), které jsou vytvářeny prostřednictvím moderních rozhraní API. „*Prostřednictvím push notifikací, přístupu k hardwaru zařízení a možnosti práce offline se do jisté míry smývají hranice mezi webovou a nativní mobilní aplikací. Kombinují tak v sobě to nejlepší z obou světů, což je důvodem, proč k tomuto řešení přechází stále více poskytovatelů služeb.*“ (4)

Webové aplikace jsou internetové aplikace, které jsou přístupné prostřednictvím webového prohlížeče mobilního telefonu. Není třeba je stahovat ani instalovat. K webovým aplikacím má uživatel časově neomezený přístup z jakéhokoli zařízení. Pro přístup k požadovaným datům lze použít počítač nebo mobilní zařízení. Webové aplikace lze používat na jakékoli platformě – Windows, Linux či Mac. Naproti tomu desktopová aplikace je softwarový program, který může uživatel spustit na počítači a jeho prostřednictvím provést konkrétní úkol. Desktopové aplikace jsou vyvinuty vždy pro konkrétní operační systém – Windows, Mac nebo Linux. Jde o program, který uživatel instaluje na pevný disk. Aktualizace desktopových aplikací instalují ručně koncoví uživatelé, případně mohou být publikovány prostřednictvím internetu. Desktopové aplikace jsou navrženy tak, aby běžely v izolovaném prostředí. Schopnost pracovat bez připojení k internetu je jednou z vlastností desktopových aplikací, které je odlišují od webových aplikací. Nevýhodou desktopových aplikací je nutnost instalace nových verzí programu a bezpečnostních aktualizací, provádění ukládání a zálohování dat uživatelem (možným řešením je využití cloudového řešení). (5)

3.1.1. HTML, CSS

HTML (HyperText Markup Language) a CSS (Cascading Style Sheets, tzv. kaskádové styly, kdy označení kaskádové souvisí s možností vrstvit definice stylu, přičemž styl aplikovaný na nadřazený prvek se použije na všechny jemu podřízené prvky) jsou základními jazyky, které se primárně používají k vytváření webových stránek a webových aplikací (HTML je značkovací jazyk používaný pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy, zatímco CSS je jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML – Extensible HyperText Markup Language nebo XML). Zásadní rozdíl mezi nimi je v tom, že HTML se používá k vytváření webových stránek a CSS se používá k ovládání stylu a rozvržení webových stránek. V HTML je nejprve napsán text a poté jsou do něj přidány prvky nebo značky, které se následně objeví na stránce. Prohlížeč tak pozná záhlaví stránky, začátek a konec odstavce apod. – tzn. HTML určuje strukturu informací v dokumentu. Problematické jsou různé interpretace popisu obsahu na různých zařízeních či prohlížečích a také dodržování zpětné kompatibility. Vytváření webové aplikace pomocí HTML a CSS vypadá následovně: (6)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Colored Lists | <!-- Do Something Smart Here --></title>
    <link rel="stylesheet" href="style.css" type="text/css" />
    <link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />
    <script type='text/javascript'
src='//ajax.googleapis.com/ajax/libs/jquery/1.3.2/jquery.min.js?ver=1.3.2'></sc
ript>
</head>
<body>
    <div id="page-wrap">
        <div id="header">
            <h1><a href="/">Colored Lists</a></h1>
            <div id="control">
<!-- IF LOGGED IN -->
```

```
                <p><a href="/logout.php" class="button">Log out</a> <a
href="/account.php" class="button">Your Account</a></p>
<!-- IF LOGGED OUT -->
                <p><a class="button" href="/signup.php">Sign up</a>   <a
class="button" href="/login.php">Log in</a></p>
<!-- END OF IF STATEMENT -->
            </div>
        </div>
```

CSS umožňuje oddělit obsah a strukturu (sémantiku) dokumentu od jeho grafické podoby. V CSS jsou pravidla aplikována prostřednictvím vlastností CSS. Vlastnosti CSS se obecně dělí do dvou širších kategorií. První kategorií je prezentace, která specifikuje barvu textu, typ písma, velikost písma, barvy pozadí, obrázky na pozadí atd. Druhou kategorií je rozvržení, které definuje polohu různých prvků na obrazovce. Styly mohou být umístěny buď v samostatném souboru (import pomocí elementu style v hlavičce), přímo v elementu (pomocí atributu style), nebo v hlavičce HTML dokumentu. Pomocí HTML a CSS je vytvořeno kompletní rozhraní webové stránky. (7) Výhodami kaskádových stylů jsou širší možnosti formátování, jednodušší zpracování obsahu, možnost pokročilých úprav s využitím JavaScriptu, rychlejší načítání stránek, rychlejší úprava obsahu a aktualizace, prostor pro responzivní weby apod. Naopak nevýhodou CSS byla různá úroveň podpory webových prohlížečů.

3.1.2. Skriptovací jazyky

Skriptovací jazyk je série příkazů, které lze provádět bez nutnosti kompilace. Zatímco všechny skriptovací jazyky jsou programovací jazyky, ne všechny programovací jazyky jsou skriptovací jazyky (JavaScript, Lua, PHP, Perl, Ruby, PL/SQL a Python jsou příklady skriptovacích jazyků). Skript je program, který je zapsán ve skriptovacím jazyce. Skriptovací jazyky jsou jazyky používané k propojení či rozšíření různých aplikací a komponent (naopak nejsou používány ke komplikovaným výpočtům či k práci se složitými datovými strukturami). Jsou obvykle interpretované (s možností měnit části programu za běhu, není vyžadován samostatný překlad), slabě typované či netypované (proměnné mohou mít libovolný obsah, automatické konverze typů), typické jsou vestavěné složitější typy a operátory (vyhledávací tabulky, seznamy apod.). (8)

Skriptovací jazyky se používají pro správu systému (start a ukončení činnosti systému, provádění dávkových operací, základní systémové operace jako například archivace apod.), automatizaci tvorby programů (často se opakující činnosti), přizpůsobení zařízení či aplikací (Windows Scripting Host, rozšíření souboru funkcí, makra v textových editorech) atd. Jejich výhodou je jednoduchá instalace aplikací, kdy v řadě případů stačí jen zkopírovat zdrojové soubory, jednoduché použití a učení, integrace s dostupnými technologiemi, rychlý vývoj aplikací a dynamické vlastnosti (konverze, typování, rozsahy polí apod.). Naopak k nevýhodám skriptovacích jazyků patří zaměření na konkrétní oblast (například PHP u dynamických webových stránek), neúplnost (předpokládá se kooperace s dalšími jazyky), nesoulad s pravidly dobrého návrhu (objektově orientované programování, strukturování programu apod.). Hlavními oblastmi využití jsou internet (PHP, JavaScript, Perl), GUI (grafická uživatelská rozhraní), komponentní technologie. (9)

3.1.3. Frameworky

V počítačovém programování je framework neboli aplikační rámec softwarová struktura (soubor do sebe pasujících knihoven), ve které může být software poskytující obecnou funkčnost selektivně měněn dodatečně napsaným kódem. Slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Poskytuje standardní způsob vytváření aplikací a je univerzálním, opakovaně použitelným softwarovým prostředím, které poskytuje konkrétní funkce jako součást větší softwarové platformy pro usnadnění vývoje softwarových aplikací, produktů a řešení. Softwarové frameworky zahrnují podpůrné programy, sady nástrojů, knihovny kódů, kompilátory a aplikační programovací rozhraní (API), která propojují různé komponenty. Framework se skládá z hot spots, které vytvářejí zcela specifickou funkcionalitu, a frozen spots, které jsou neměnné a definují celkovou architekturu softwarové struktury, její základní komponenty a vztahy mezi nimi. (10)

Hlavní výhodou používání frameworků je šetření nákladů, využívání osvědčených technologií, zefektivnění vývoje a s tím související růst ziskovosti a konkurenceschopnosti, minimalizace bezpečnostních rizik atd. Výhodou pro programátory je zvýšení jejich hodnoty na trhu práce, získání dovednosti vytváření kvalitních webových aplikací a možnost být součástí širší komunity programátorů. K nejpoužívanějším frameworkům patří Nette framework a Zend framework. V tuzemsku oblíbený Nette framework nabízí

výborný výkon, ladicí nástroje, strmější křivku učení, zabezpečení proti bezpečnostním dírám, čistý a vyzrálý objektový návrh využívající interfaces, podporu moderních technologií, rozdělení práce mezi HTML kodéra a programátora i možnost doplnění se Zend frameworkem. Zend Framework je sbírka více než šedesáti balíčků pro profesionální vývoj PHP aplikací. Každý balíček je dostupný na GitHubu a lze jej nainstalovat přes Composer. Zend Framework je open source, objektově orientovaný webový aplikační rámec pro PHP. Je označován jako tzv. knihovna komponent, neboť disponuje řadou volně propojených komponent, které lze používat víceméně nezávisle. Zend Framework nabízí též pokročilou implementaci Model-View-Controller (MVC), kterou lze použít k vytvoření základní struktury pro aplikace. K nejčastěji používaným komponentám Zend Framework patří Zend_Controller, Zend_Config, Zend_Layout, Zend_Db, Zend_Registry a Zend_Db_Table. Prostřednictvím těchto komponent lze poměrně rychle vytvořit jednoduchou aplikaci. (11)

3.1.4. Frontend

Frontend a backend jsou dva nejpoužívanější termíny v počítačovém průmyslu, které určují typ práce, kterou softwarový vývojář dělá, a technologie, které používá. Frontend je část webu, se kterou uživatel přímo komunikuje. Označuje se také jako klientská strana aplikace. Zahrnuje vše, s čím uživatelé přímo pracují – barvy a styly textu, obrázky, grafy a tabulky, tlačítka, navigační nabídku apod. HTML, CSS a JavaScript jsou základní jazyky používané pro vývoj frontendu. Strukturu, design, obsah a chování všeho, co je vidět na obrazovkách prohlížeče při otevření webových stránek, webových aplikací nebo mobilních aplikací, implementují právě vývojáři frontendu. Dvěma hlavními cíli frontendu jsou odezva a výkon. Vývojář musí zajistit, aby byl web responzivní, tzn. aby se správně zobrazoval na zařízeních všech velikostí, přičemž žádná část webu by se neměla chovat abnormálně bez ohledu na velikost obrazovky. (12)

Frontend je vytvářen pomocí jazyku HTML (Hypertext Markup Language), který se používá k návrhu přední části webových stránek pomocí značkovacího jazyka. HTML je kombinací hypertextového a značkovacího jazyka, kdy hypertext definuje vazby mezi webovými stránkami. Kaskádové styly, často označované jako CSS, jsou jazykem, jehož cílem je zjednodušit proces vytváření prezentovatelných webových stránek. CSS umožňuje aplikovat styly na webové stránky. JavaScript je skriptovací jazyk používaný k vytváření

interaktivních stránek. Používá se ke zlepšení funkčnosti webových stránek při spouštění her a webového softwaru. Vykreslování obsahu se děje v počítači uživatele namísto na vzdáleném webovém serveru pomocí JavaScriptu. V praxi to znamená, že server je potřeba pouze k základní obsluze webové aplikace a prohlížeč bude mít na starosti vykreslování této aplikace v konečné podobě. Prezentační logika spojená s vytvářením webové stránky, která má na starosti řešení toho, jak jsou věci prezentovány uživateli na obrazovce, je řešena na straně klienta. (13)

K hlavním frontend frameworkům a knihovnám patří:

- AngularJS – JavaScript open-source front-end framework, který se používá především k vývoji jednostránkových webových aplikací (SPA). Je to neustále rostoucí a rozšiřující se framework, který zkvalitňuje vývoj webových aplikací (například mění statický HTML na dynamický HTML) – je používán k tvorbě dynamických webů a webových aplikací. Je založen na principu znovupoužitelných komponent, čímž usnadňuje vývoj a odděluje design od funkcionality. Disponuje unikátní funkcí oboustranného vázání dat, což výrazně zvyšuje výkon aplikací založených na prohlížeči. Lze jej používat pro tvorbu progresivních webových i vícestránkových aplikací.
- React.js je deklarativní, efektivní a flexibilní javascriptová knihovna pro vytváření uživatelských rozhraní (spravuje ji Facebook). Soustředí se primárně na vrstvy pohledu (views) a je vhodná pro práci s rychle se měnícími daty, tudíž může být využita jako základ pro tvorbu jednostránkových či mobilních aplikací. Její základ tvoří zapouzdřené komponenty. Framework využívá virtuální DOM, který zefektivňuje výkon a jenž může být poskytnut buď na straně klienta, nebo v kombinaci s Node.js na straně serveru.
- Bootstrap je bezplatná kolekce nástrojů s otevřeným zdrojovým kódem pro vytváření responzivních webů a webových aplikací. Jde o nejpopulárnější framework pro vývoj responzivních webových stránek zaměřených na mobilní zařízení.
- jQuery je open-source javascriptová knihovna, která zjednodušuje interakce mezi HTML/CSS dokumentem, přesněji Document Object Model (DOM), a JavaScriptem. jQuery zjednodušuje procházení a manipulaci s HTML dokumenty,

zpracování událostí prohlížeče, animace DOM, interakce Ajax a vývoj JavaScriptu pro různé prohlížeče.

- SASS (Syntactically Awesome Style Sheets) je kompilovaný jazyk rozšiřující syntaxi CSS o proměnné, podmínky, cykly, funkce, mixiny apod. Šetří čas, množství napsaného kódu, je přehlednější a snadněji se udržuje. Používá se ke snadnému rozšíření funkčnosti existujícího CSS webu.
- Flutter je open-source SDK pro vývoj uživatelského rozhraní spravované společností Google. Využívá programovací jazyk Dart. Flutter je framework s vlastní vykreslovací vrstvou, což ho přibližuje frameworkům (engine) pro vývoj her. Využívá reaktivní přístup k programování. Používá základní UI prvky, které je možné doplnit o vlastní, přičemž prvky vzhledově odpovídají tomu, jak mají na dané platformě vypadat v nativním podání. (14)

3.1.5. Backend

Renderování stránek znamená generování nebo vykreslování HTML výstupu. HTML je značkovací jazyk, který weboví vývojáři používají k vytváření webových stránek. Vykreslování webu může probíhat jak na straně serveru, tak na straně klienta. Zatímco frontend souvisí s klientskou stranou, backend je renderování na straně serveru. Backend vývoj byl základním způsobem vytváření webových stránek a webových aplikací. Uživatel navštívil stránku, odeslal požadavek na obsah, server tento požadavek zpracoval a vytvořil odpověď, která byla zaslána zpět do uživatelova prohlížeče. Jestliže je web vykreslován na straně serveru, všechny procesy spojené s vytvářením stránky HTML se zpracovávají na vzdáleném serveru hostujícím webovou stránku nebo webovou aplikaci. V okamžiku, kdy vzdálený server zpracovává požadavek, uživatelův webový prohlížeč je nečinný a čeká, až server dokončí zpracování požadavku a odešle odpověď. Když je odpověď přijata, webové prohlížeče ji interpretují a zobrazí obsah na obrazovce. (15)

Jak bylo zmíněno výše, backend je serverová strana webu. Ukládá a uspořádává data a také zajišťuje, že vše na klientské straně webu funguje správně. Je to část webu, kterou uživatel nevidí a nemůže s ní pracovat, a část softwaru, která nepřichází do přímého kontaktu s uživatelem. K vlastnostem vyvinutým vývojáři backendu mají uživatelé pouze nepřímý přístup prostřednictvím frontendové aplikace. Činnosti, jako je psaní API, vytváření knihoven a práce se systémovými komponentami jsou součástí backendu. Back-

end je ovladačem pro webovou aplikaci, a proto by měl být vyvinut nejdříve. Backend je jádrem aplikace, propojuje ji s dalšími systémy, ukládá informace do databází, je zodpovědný za výkon, bezpečnost a další klíčové aspekty aplikace. Naproti tomu frontend je uživatelské rozhraní pro aplikaci – to, co uživatelé vidí. Je též důležitou součástí aplikace, nicméně může být vyvinut až po vytvoření základních funkcí. (15)

K nejpoužívanějším programovacím jazykům pro backend webu patří:

- PHP – skriptovací jazyk na straně serveru navržený speciálně pro vývoj webových aplikací. Vzhledem k tomu, že kód PHP se spouští na straně serveru, je označován jako skriptovací jazyk na straně serveru. PHP je interpretovaný jazyk, který nevyžaduje překladač a může běžet v podstatě na všech hlavních operačních systémech (Windows, Unix, Linux, macOS apod.). Výhodou PHP je podpora různých standardních systémů správy databází jako SQLite, MySQL atd., kompatibilita napříč platformami, podpora komunity a snadná naučitelnost. Výhodou je také jeho vysoká bezpečnost, kdy k šifrování uživatelských dat je k dispozici řada hashovacích funkcí.
- C++ je univerzální programovací jazyk a v dnešní době široce používaný backendový jazyk. C++ je blízký jazyku C – koncept objektově orientovaného programování kombinuje se syntaxí a výkonem jazyka C.
- Java je jedním z nejpopulárnějších a nejrozšířenějších programovacích jazyků a platforem. Java je objektově orientovaný programovací jazyk běžně používaný pro vývoj webových aplikací. Funguje na principu Write-Once-Run-Anywhere, tzn. zkompilovaný kód lze spustit na jakékoli platformě, která podporuje Javu, bez nutnosti rekompilace. Dále podporuje tzv. multithreading umožňující souběžné spouštění dvou či více vláken pro maximální využití CPU.
- JavaScript lze použít jako frontend i backend programovací jazyk. Node.js je open-source a multiplatformní běhové prostředí pro spouštění kódu JavaScript mimo prohlížeč. NodeJS není framework ani programovací jazyk, je používán k vytváření backendových služeb, jako jsou API, nebo webové aplikace. V praxi ho používají velké firmy jako Uber, Walmart, Paypal, Netflix apod.
- Python se svou syntaxí podobá anglickému jazyku. Jeho jednoduchá syntaxe je srozumitelná i začínajícím programátorům. Nespornou výhodou Pythonu je též jeho snadná čitelnost a snadné psaní, udržování a pochopení kódu. Pro vývoj webových

aplikací jsou využívány především dva následující Python frameworky: Django integrované funkce zjednodušující vytváření webu a umožňující opětovné použití kódu. Lze ho synchronizovat mimo jiné s databázemi Oracle SQL a MySQL; framework Pyramid je používán v případech, kdy je třeba implementovat různá řešení. Je používán buď s rozšířenými pluginy, případně s různými databázemi.

K hlavním backendovým frameworkům patří Laravel, Express, Django, Rails, Spring atd. (16)

3.2. Použité technologie

3.2.1. PHP

PHP (Hypertext Preprocessor, hypertextový preprocesor, původně Personal Home Page) je skriptovací programovací jazyk určený primárně pro programování dynamických webových stránek a aplikací (například formáty WML, HTML, XHTML), který lze použít i k vytváření desktopových aplikací (kompilovaná forma PHP) a konzolových aplikací. Má syntaxi podobnou jazyku C/C++. Značná část jeho syntaxe je převzata z jazyků C, Java a Perl (například třídy a asociativní pole) s několika unikátními funkcemi specifickými pro PHP. Cílem jazyka je umožnit webovým vývojářům rychle vytvářet dynamicky generované stránky. Skripty jsou prováděny na straně serveru. Je volně k dispozici pro všechny operační systémy. PHP obsahuje řadu užitečných funkcí, nicméně má i některé nevýhody, jako například mezery v jeho standardních knihovnách. S řadou funkcí se totiž buď špatně pracuje, případně v něm úplně chybí. Použití kvalitních knihoven přitom zrychluje a zjednodušuje vývoj aplikace, která bude navíc snadno udržovatelná. PHP je v současnosti jedním z nejpopulárnějších programovacích jazyků, který je široce používaný jak v komunitě open source, tak v průmyslu k vytváření velkých webových aplikací a aplikačních rámců. PHP umožňuje rychlé vytváření kvalitních a zabezpečených webových aplikací. (17)

K funkcím PHP patří matematické funkce, funkce pro práci s textovými řetězci, síťové funkce (čtení záznamů DNS, sockety, převod mezi IP adresami a doménovými adresami), odesílání elektronické pošty a práce s elektronickou poštou prostřednictvím protokolu IMAP, čtení informací ze síťových zařízení pomocí protokolu SNMP, podpora regulárních výrazů, práce se soubory a adresáři, funkce pro práci s časem a daty, přístup do

adresářových služeb LDAP, podpora souborově orientovaných databází, funkce pro zpracování dat z webových formulářů, funkce pro spouštění externích programů a zpracování jejich výstupu apod. „*Namísto psaní programů, které ve výsledku vygenerují HTML, používá PHP opačný přístup. Do HTML stránky můžeme zapsat příkazy PHP, jejichž výsledek se po interpretaci zkombinuje s okolním HTML kódem.*“ (18)

3.2.2. Laravel

Laravel je serverový PHP framework, s jehož pomocí je možné vytvářet kompletní aplikace (aplikace s funkcemi, které obvykle vyžadují backend). Laravel je nejrozšířenější framework pro PHP, který zjednodušuje vytváření full-stack webových aplikací. Je pro něj typický jednoduchý a přehledný kód, optimalizovaný pro reálné webové aplikace – jednotlivé komponenty je možné generovat prostřednictvím příkazů v terminálu, což výrazně urychluje tvorbu webových stránek. Laravel je kompletní open-source PHP framework navržený tak, aby usnadnil a urychlil vývoj webových aplikací pomocí vestavěných funkcí. Právě tyto funkce jsou hlavním důvodem, proč weboví vývojáři Laravel používají. Základem Laravelu je framework Symfony – pravděpodobně nejúspěšnější framework pro PHP, který v tuzemsku konkuruje Nette frameworku. Kvůli popularitě Laravelu k němu existuje obsáhlá dokumentace i řada návodů. Laravel je používán v menších i velkých projektech. Pro používání Laravelu jsou třeba pokročilé znalosti PHP, stejně jako znalost objektově orientovaného programování a MVC architektury (tzn. programování v Laravelu je programování v PHP s využitím objektově navržených znovupoužitelných komponent postavených na principech MVC architektury). (19) „*Laravel nabízí moderní a rychlé řešení pro vývoj webových aplikací. Je postaven na návrhových vzorech a dodržuje pricipy jako jsou například DRY, KISS, SOLID atd. Snahou Laravelu je dodržovat jednoduchost a čitelnost kódu. Framework poskytuje opravdu jednou práci s databází a nabízí vlastní ORM Eloquent či migrace, které pomáhají práci v týmu, aby byla databáze jednoduše synchronizována bez nutnosti složitých mechanismů. Laravel také zcela podporuje RESTful architekturu a umožňuje tak snadno vytvářet cesty (routes) či API.*“ (20)

Důležitou součástí Laravelu je Laravel Forge, který je určen pro management serveru – PHP a Laravel deploy webových aplikací. Pro Laravel Forge je typická jednoduchost a rychlá pomoc. Službu lze využít primárně u menších a středně velkých

PHP aplikací, které nevyžadují speciální infrastrukturu. Její fungování se liší od systému platform-as-a-service (PaaS) – místo přímého hostování PHP aplikace využívá SSH pro přihlášení na server a instaluje veškerý software potřebný pro spuštění PHP aplikací. Další výhodou je možnost přesunu – změny hostingového poskytovatele a pokračování v nastavení serveru. Služba umožňuje též instalaci balíčků a bezpečné nastavení serverového zabezpečení. Ke klíčovým vlastnostem patří možnost instalace SSL certifikátů a HTTPS přesměrování, přidání několika stránek na jeden server, deploy aplikace přímo z Githubu přes SSH, správa cronu a démon procesů, přidávání a spouštění vlastních bash skriptů na serveru, jednoduchá správa MySQL databáze a jednoduchá integrace s DigitalOcean, AWS nebo linode. (21)

Nastavení databáze při vytvoření nového projektu Laravel může vypadat následovně:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Případně lze vytvořit novou databázi pro aplikaci links: (22)

```
# Connect via the mysql CLI
mysql -u root -p
mysql> create database links_development;
mysql> exit
# Or use the -e flag to run the create command
mysql -u root -e'create database links_development'
```

3.2.3. MVC

MVC neboli Model-View-Controller je architektonický vzor v softwarovém designu běžně používaný k implementaci uživatelských rozhraní, dat a řídicí logiky. Původně vznikl na desktopech. Je založen na oddělení logiky a výstupu, což zajišťuje lepší dělbu práce a lepší údržbu (viz například problém špagetového kódu, kdy logické operace a renderování výstupu jsou v jediném souboru a zamotané do sebe). Některé další vzory jsou založeny na MVC, například MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter) a MVW (Model-View-Whatever). Aplikace je složena ze součástí tří typů, které vykonávají tři hlavní role – řízení, logiku a výstup. První částí jsou kontrolery (Controllers). „*Kontroler je část, se kterou komunikuje uživatel. Předá jí parametry a ona mu vrátí data (např. HTML stránku). Kontroler typicky parametry předá modelům, od kterých získá data. Tato data předá pohledům (šablonám), které data začlení do nějakého HTML kódu. Tento HTML kód pošle kontroler uživateli do prohlížeče. Funguje tedy jako takový prostředník.*“ (23) Druhou částí jsou modely (Models), které obsahují logiku aplikace (například výpočty či práce s databází). Třetí částí jsou pohledy (Views), které obsahují Blade šablony s HTML kódem. „*Blade je šablonovací systém, který na rozdíl od ostatních systémů nezakazuje používání PHP kódu v pohledu, tudíž neslouží pouze pro vkládání proměnných do HTML kódu. Blade používá speciální syntaxi, jež je poté převedena na normální PHP kód a zároveň tento vygenerovaný pohled je následně uložen do mezipaměti pro příští použití kvůli rychlejšímu načítání (pokud není daný pohled modifikován).*“ (23) Model-View-Controller se stal standardem v designu moderních webových aplikací. Většina kódu webových aplikací spadá do jedné z následujících tří kategorií: prezentace, logika a přístup k datům, a právě vzor MVC toto oddělení dobře modeluje. Konečným výsledkem je, že prezentační kód lze konsolidovat v jedné části aplikace s logikou v jiné a kód pro přístup k datům v další. Toto oddělení je přínosné pro udržení organizace kódu, obzvláště v případech, kdy na stejné aplikaci pracuje více vývojářů.

3.2.4. SQL

SQL (Structured Query Language) jako nástupce jazyka SEQUEL (Structured English Query Language) je standardizovaný dotazovací jazyk, který se používá ke správě relačních databází a provádění různých operací s daty v relačních databázích, které jsou založeny na relačním modelu sdružujícím data do tabulek (relací). Relační systémy, známé také jako databáze SQL, zahrnují sadu tabulek obsahujících data v řádcích a sloupcích. Každý sloupec v tabulce odpovídá kategorii dat (například jméno zákazníka), zatímco každý řádek obsahuje datovou hodnotu pro protínající se sloupec. SQL, původně vytvořený v sedmdesátých letech minulého století, je používán nejen správci databází, ale také vývojáři vytvářejícími skripty i datovými analytiky. Použití SQL zahrnuje úpravu databázových tabulek a indexových struktur, přidávání, aktualizace a mazání řádků dat a získávání podmnožin informací z databází. Dotazy a další operace SQL mají formu příkazů – běžně používané příkazy SQL zahrnují výběr, přidání, vložení, aktualizaci, odstranění, vytvoření, změnu a zkrácení. Od sedmdesátých a osmdesátých let se stal SQL standardním programovacím jazykem pro relační databáze. (24)

U jazyka SQL lze rozlišit následující skupiny příkazů: příkazy DML (data manipulation language) pro získávání, ukládání a odstraňování dat v databázi (DELETE, INSERT, UPDATE, SELECT); příkazy DDL (data definition language) pro tvorbu a úpravy struktury databáze (CREATE, DROP, ALTER); příkazy TCL (transactional control language) pro správu databázových transakcí (COMMIT, BEGIN, ROLLBACK); příkazy DCL (data control language) pro správu uživatelských práv a rolí (REVOKE, GRANT). V následující ukázce je nejprve vytvořena a smazána tabulka zaměstnanců a poté je v tabulce přidán a smazán sloupec s datem narození: (25)

```
CREATE TABLE zamestnanci
(
    cislo UNSIGNED INT NOT NULL PRIMARY KEY,
    jmeno VARCHAR(100) NOT NULL,
    prijmeni VARCHAR(100) NOT NULL,
    mesto VARCHAR(100) NOT NULL
)
DROP TABLE zamestnanci
ALTER TABLE zamestnanci ADD narozen DATETIME NOT NULL
ALTER TABLE zamestnanci DROP narozen
```

Transakce probíhají jako posloupnosti příkazů splňujících podmínky ACID. Vlastnosti ACID na serveru SQL zajišťují integritu dat během transakce. SQL ACID je zkratka pro atomicity (A), consistency (C), independence (I), durability (D). Atomicity znamená, že buď proběhnou všechny operace (insert, update, delete – vložení, aktualizace, smazání) v rámci jedné transakce, nebo neproběhnou žádné. Případně všechny příkazy v rámci transakce jsou buď dokončeny, nebo vráceny zpět. Consistency zajišťuje konzistenci databáze, tzn. že cokoliv se stane během transakce, data musí být ponechána v konzistentním stavu. Pokud je transakce úspěšně dokončena, jsou provedeny všechny změny v databázi. Jestliže dojde během transakce k chybě nebo dojde k selhání systému, všechny již provedené změny budou automaticky vráceny zpět (tzn. databáze se obnoví do stavu, ve kterém byla před zahájením transakce). Independence znamená, že každá transakce je individuální a žádná jiná transakce nemá přístup k výsledkům ostatních transakcí, dokud není daná transakce dokončena. Nelze také provést stejnou operaci pomocí více transakcí současně. Jakmile je transakce dokončena, změny, které provedla v databázi, budou trvalé (durability). I když dojde k selhání systému nebo k jakýmkoli abnormálním změnám, veškerá potvrzená data budou trvale uložena. (26)

3.2.5. Bootstrap

Kaskádové styly zlepšují vzhled webové stránky, nicméně u nových stránek je potřeba neustále dokola nastavovat základní vzhledové parametry. Zjednodušení nabízí předpřipravené komplexní knihovny – frameworky – s vlastnostmi CSS. Jedním z těchto frameworků je volně dostupný (open-source) Bootstrap vyvinutý společností Twitter v roce 2011 (Bootstrap byl původně určen pro eliminaci nekonzistence nástrojů používaných programátory Twitteru). Bootstrap je frontend framework pro rychlé a snadné vytváření webových šablon. Jeho základní vlastností je responzivita (od verze dva vydané v roce 2012) – responzivita znamená, že pokud uživatel otevře webovou stránku na PC, na notebooku, v tabletu nebo na mobilu, velikost obrazovky se automaticky přizpůsobí a obsah webu se zobrazí správně. Framework je přizpůsoben pro mobilní zařízení, která jsou používána ve stále větší míře na úkor stolních počítačů. Lze jej spustit na jakékoli platformě (Windows, Linux, Mac). Dalšími vlastnostmi jsou flat design, dvanáctisloupcový grid systém a mobile-first přístup. Díky tomu je kód kompaktnější a podporuje dobré praktiky. Výhodou Bootstrapu je snadné zpracování uživatelského

rozhraní ve webové aplikaci bez ohledu na to, zda jde o rozhraní v administraci frontendových či backendových aplikací. Další výhodou Bootstrapu je jeho kompatibilita s nejnovější verzí nejpoužívanějších prohlížečů, a podpora responzivního designu (rozložení stránky se dynamicky přizpůsobuje s ohledem na typ zařízení). (27)

Bootstrap nabízí značné množství komponent – menu, tabulky, tlačítka, formuláře atd. – které jsou na stránku vkládány prostřednictvím HTML. Obsahuje též javascriptové komponenty využívající knihovnu jQuery – dropdown menu, dialogová okna apod. Zdrojový kód stránky bude v Bootstrapu totožný jako zdrojový kód stránky s HTML a CSS. Rozdíl spočívá v nutnosti načtení kaskádových stylů pro Bootstrap, šablony a souboru s javascriptovými skripty. Soubory lze buď stáhnout z příslušných stránek a vložit do projektu, případně lze využít Bootstrap CDN a soubory načíst ze serverů Bootstrapu (problémem mohou být výpadky datových úložišť, které mohou znemožnit funkčnost webové stránky). Postup načtení je uveden v následující ukázce: (28)

```
<linkrel="stylesheet"href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/c
ss/bootstrap.min.css"integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
<script                    src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link                                                    rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script               src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
  </body>
</html>
```

Bootstrap zkracuje čas potřebný k vývoji webových stránek. Webová stránka by měla být k dispozici v takové formě, aby si každý uživatel mohl bez problémů číst a prohlížet její obsah. V Bootstrapu je webová stránka vytvořena pouze jednou a následně prohlížeč zobrazí webovou stránku ve správné podobě na jakémkoli monitoru či displeji. To znamená úsporu času, což je největší výhoda Bootstrapu – jeho použití zvyšuje rychlost vývoje.

Další výhodou je snadné použití – s Bootstrapem může pracovat vývojář disponující základními znalosti HTML a CSS. Bootstrap je kompatibilní s různými prohlížeči (je navržen tak, aby webová stránka vypadala stejně ve všech moderních prohlížečích, jako je Firefox, Chrome, Opera, Internet Explorer / Microsoft Edge apod.). Bootstrap je k dispozici zdarma a lze ho snadno přizpůsobit – tzn., pokud vývojář nechce využívat všechny funkce Bootstrapu, může použít pouze několik vybraných funkcí, které si zvolí a stáhne na webových stránkách Bootstrapu. (29)

3.2.6. DataTables

DataTables je zásuvný modul (plugin) pro javascriptovou knihovnu jQuery. Jedná se o vysoce flexibilní nástroj, založený na pravidlech progresivního vylepšování, který přidává všechny tyto pokročilé funkce do jakékoli HTML tabulky. Cílem DataTables je zlepšit dostupnost dat v tabulkách HTML. DataTables rozlišuje dvě kategorie uživatelů, kteří komunikují se softwarovými rozhraními: koncoví uživatelé jsou ti, kteří používají rozhraní v prohlížeči, a vývojáři pracující s DataTables na vytváření aplikací, webů, služeb apod. Koncoví uživatelé musí mít možnost získat požadované informace z tabulky co nejrychleji, a proto DataTables disponuje vestavěnými funkcemi umožňujícími vyhledávání, objednávání, stránkování apod. Obdobně má DataTables dva různé režimy zpracování dat: zpracování na straně klienta – celý soubor dat je načten předem a zpracování dat se provádí v prohlížeči – a zpracování dat na straně serveru. Při zpracování dat na straně klienta je kompletní sada dat načtena předem. Data lze číst z DOM, zdroje dat JS nebo přes Ajax a zpracování dat probíhá v prohlížeči. Každý režim má své výhody a nevýhody, ale klíčový ukazatel, který režim zvolit, je založen na počtu řádků v tabulce. Obecně platí, že při práci s méně než 10 000 řádky je použito zpracování na straně klienta, pro více než 100 000 řádků je použito zpracování na straně serveru, případně je potřeba rozhodnout se na základě povahy aplikace a dat, která je třeba zobrazit. „*V administraci,*

kde jsou tisíce až desetitisíce záznamů, už není vhodné vypisovat vše. Lepší je provést vyhledání a seřazení na serveru a odeslat jen zlomek záznamů. Hodnota z integrovaného vyhledávaní je automaticky obsažena v XHR requestu, který se posílá na server.“ (30)

Tyto dva režimy zpracování se přitom vzájemně vylučují – tzn. nelze je používat současně ani není možné dynamicky přecházet z jednoho režimu do druhého. Zpracování na straně klienta je výchozím provozním režimem DataTables, protože se snadno používá a nevyžaduje psaní dalšího kódu. V režimu zpracování na straně klienta se řazení dat v tabulce, vyhledávání, stránkování a všechny další operace zpracování dat, které DataTables zajišťuje, provádí v prohlížeči samotným DataTables. Při zpracování na straně serveru se načtou pouze potřebná data a server provádí jejich zpracování (toto řešení je vhodné pro velké soubory dat). Problém nastává v okamžiku, kdy server není schopen zvládnout nápor požadavků. Zpracování na straně serveru je vhodné u velkého množství dat, která je třeba zobrazit v tabulce (miliony řádků dat). Na takové úrovni může odeslání dat klientovi a následné zpracování dat Javascriptem znamenat značnou režii a důsledkem může být špatný výkon koncové aplikace. V režimu zpracování na straně serveru je veškeré objednávání, vyhledávání apod. dat předáno serveru, který může využívat zde dostupné databázové zdroje, které jsou na tento druh operací vyladěny. Každá stránka dat zahrnuje odeslání požadavku Ajax na server. Přestože dokončení každého požadavku Ajaxu může trvat zlomek sekundy, tento přístup může být vhodnější než dlouhé čekání, až se všechna data načtou. (31)

3.2.7. Oracle SQL

Oracle SQL Developer je bezplatné integrované vývojové prostředí, které zjednodušuje vývoj a správu databáze Oracle v tradičním i cloudovém nasazení. SQL Developer nabízí kompletní end-to-end vývoj aplikací PL/SQL, pracovní list pro spouštění dotazů a skriptů, konzolu DBA pro správu databáze, rozhraní sestav, kompletní řešení pro modelování dat a migrační platformu pro přesun databáze třetích stran společnosti Oracle. Structured Query Language (SQL) je sada příkazů, pomocí kterých programy a uživatelé přistupují k datům v databázi Oracle. Aplikační programy a nástroje Oracle v řadě případů umožňují uživatelům přístup k databázi bez přímého použití SQL, nicméně tyto aplikace musí používat SQL při provádění požadavků uživatele. Existuje řada různých systémů pro správu relačních databází, například Sybase, Microsoft Access a MySQL, ale dvěma

nejoblíbenějšími a nejpoužívanějšími jsou Oracle a MS SQL Server. Přestože existuje řada podobností mezi těmito dvěma platformami, lze identifikovat též řadu rozdílů. Zřejmě nejviditelnějším rozdílem je jazyk, který používají. Ačkoli oba systémy používají verzi strukturovaného dotazovacího jazyka nebo SQL, MS SQL Server používá Transact SQL nebo T-SQL, což je rozšíření SQL původně vyvinuté společností Sybase a používané společností Microsoft. Oracle používá PL/SQL nebo procedurální jazyk/SQL. Oba jazyky mají odlišnou syntaxi a schopnosti. Hlavním rozdílem mezi nimi je způsob, jakým zpracovávají proměnné, uložené procedury a vestavěné funkce (například PL/SQL v Oracle může seskupovat procedury do balíčků, což na MS SQL Server nelze provést). PL/SQL je komplexní a potenciálně výkonnější, zatímco T-SQL je jednodušší a snáze se používá. „*PL/SQL je rozšíření jazyka SQL o procedurální rysy. Je specifické pro produkty firmy Oracle, procedurální rozšíření SQL produktů jiných firem se zpravidla navzájem liší. Výjimkou je SŘBD DB2 společnosti IBM, který podporuje jak vlastní procedurální jazyk SQL PL, tak je plně kompatibilní s jazykem PL/SQL včetně datových typů. Základním stavebním kamenem PL/SQL je tzv. PL/SQL blok, který může být buď tělem triggeru, procedury a funkce, nebo samostatný.*“ (32) Kromě datových typů z SQL (CHAR, NUMBER, DATE, BLOB) lze v PL/SQL použít některé speciální datové typy, například PLS_INTEGER, BINARY_INTEGER, BOOLEAN, BINARY_FLOAT a BINARY DOUBLE, případně složené datové typy RECORD (typ záznam) a VARRAY (pole proměnné délky).

3.2.8. Procedury

Pomocí příkazu CREATE PROCEDURE lze vytvořit proceduru nebo specifikaci volání. Procedura je skupina příkazů PL/SQL, které lze volat podle jména. Specifikace volání sděluje Oracle Database, kterou metodu Java má vyvolat při volání, a též sděluje databázi, jaké typy převodů má provést. Uložené procedury (tzn. procedury uložené v databázi) nabízejí výhody v oblastech vývoje, integrity, zabezpečení, výkonu a alokace paměti. Oproti tradičnímu přístupu, kdy s daty pracuje přímo aplikace přistupující k databázi prostřednictvím SQL, uložené procedury jsou vykonávány databázovým serverem. To přináší řadu výhod například v oblasti sdílení kódu, kdy komplexní kód uložené procedury může být sdílen a využíván více aplikacemi, které přistupují k databázi, přičemž činnost procedury nemusí být implementována zvlášť v každé aplikaci (tento

přístup je užitečný především v případě, kdy jsou aplikace programovány v různých jazycích). Další výhodou je zjednodušení rozhraní mezi databází a aplikací. Jestliže aplikace k databázi přistupuje výhradně prostřednictvím volání uložených procedur, pak lze do značné míry měnit schéma databáze, aniž by to vyžadovalo měnit kód samotné aplikace. Podobně chyby v kódu uložené procedury lze opravovat bez nutnosti distribuovat a instalovat nové verze aplikace přistupující k databázi. Výhodné jsou i menší přesuny dat mezi aplikací a databázovým serverem, což je výhodné především v případech, kdy je aplikace spuštěna na jiném zařízení než databáze. Naproti tomu nevýhodou může být nekompatibilita jazyků uložených procedur, případně chybějící nástroje pro ladění kódu či podpora uložených procedur. (33)

Blok PL/SQL má následující strukturu:

DECLARE – deklarace konstant a proměnných;

BEGIN – výkonné příkazy bloku;

EXCEPTION – výkonné příkazy obsluh přerušení;

END – ukončeni procedury.

3.2.9. JavaScript

JavaScript je objektově orientovaný, multiplatformní a událostmi řízený skriptovací jazyk. Skriptovací jazyky prokázaly svou užitečnost v různých aplikačních oblastech. Syntaxe JavaScriptu náleží do rodiny jazyků C, C++ a Java, nicméně sémanticky se od nich liší. JavaScript je používán mimo jiné pro vytváření webových aplikací, je jedním ze zásadních jazyků pro vývojáře webových aplikací. Sofistikované webové aplikace navržené jako náhrada aplikací pro stolní počítače vyžadují značné množství javascriptového kódu, který je nutné spouštět přiměřenou rychlostí. Navíc při tvorbě webových aplikací je třeba brát v úvahu fakt, že uživatelé mají současně otevřeno několik oken se spuštěnými webovými aplikacemi, jako je e-mail, kalendář, prohlížení obrázků a manipulace s nimi apod. Podpora JavaScriptu v prohlížeči tedy může do značné míry zlepšit či zhoršit uživatelský dojem. JavaScript je důležitou součástí každého velkého webového prohlížeče. Je uznáván jako plnohodnotný programovací jazyk, umožňující provádět složité výpočty a interakce. Je používán při tvorbě webových stránek jako součást HTML kódu (obvykle je javascriptový program spouštěn na straně klienta až po stažení stránky, na rozdíl od PHP). Je využíván též na straně serveru při tvorbě serverových

aplikací fungujících v reálném čase, k tvorbě dynamických a interaktivních webových stránek i k tvorbě rozšíření pro řadu aplikací. (34)

Kompletní implementace JavaScriptu se skládá z tří odlišných částí: jádro (ECMAScript jako standardizovaná verze JacaScriptu, z níž jsou odvozeny ostatní implementace, ECMA – European Computer Manufacturers Association neboli Evropská asociace výrobců počítačů), objektový model dokumentu (DOM) a objektový model prohlížeče. Webové prohlížeče jsou pouze jedním hostitelským prostředím, ve kterém může existovat implementace ECMAScript. Hostitelské prostředí poskytuje základní implementaci ECMAScriptu a implementačních rozšíření navržených pro rozhraní se samotným prostředím. DOM je aplikační programovací rozhraní pro XML, které bylo rozšířeno pro použití v HTML. (35)

3.2.10. AJAX

Význam webových aplikací v posledních letech roste, neboť lidé mají stále více tendenci využívat internet při provádění svých každodenních aktivit (nakupování, hraní, práce, socializace apod.). Testování je velmi důležitou součástí tvorby webových aplikací, které představuje řadu nových výzev. Webové aplikace jsou totiž velmi dynamické a právě dynamické funkce ztěžují jejich testování. Skript na straně serveru by mohl generovat velký počet stránek na straně klienta, které se mohou jevit odlišně a vykonávat různé funkce v závislosti na uživatelských vstupech na straně klienta a vnitřních stavech na straně serveru. AJAX (Asynchronous JavaScript and XML) umožňuje aktualizace stránek za běhu interakcí se serverovými skripty. Takové aktualizace stránky nevyžadují načtení nové stránky, ale spíše úpravu části aktuální stránky (například změnu velikosti rámečku, přemístění tlačítka, nahrazení části stránky obrázkem apod.). Tyto aktualizace mění jak rozhraní, tak funkčnost aktuální stránky klienta. (36)

AJAX je technologie, jejímž prostřednictvím lze z JavaScriptu provádět HTTP požadavky. Odešle HTTP požadavek a stáhne data na dané adrese (například HTML kód webové stránky). AJAX umožňuje tyto operace provést přímo v javascriptovém kódu a na pozadí, aniž by se stránka, na které se uživatel nachází, musela znovu načíst. Je využíván při tvorbě SPA (Single Page Aplication). Běžný web se skládá z několika HTML stránek, na které vedou odkazy. „*Při kliknutí na odkaz prohlížeč smaže vše, co na stránce vidíme, a načte novou stránku. V single page aplikaci však takové odkazy nejsou. Místo toho se*

spouští AJAX, který stahuje nové stránky a data na pozadí. Poté JavaScript do části stránky, která má být měněna, vloží nový obsah. To může působit příjemněji než refreshování celého okna, a podobá se to více uživatelskému rozhraní desktopových aplikací. Příkladem SPA je např. Gmail.“ (37)

Výhodou technologie AJAX je přiblížení k desktopovým aplikacím, eliminace potřeby znovunačtení a překreslení webové stránky při akci uživatele, snížení zátěže webového serveru a větší plynulost práce. Díky technologii AJAX může webová aplikace zpracovávat uživatelské interakce s webovou stránkou způsobem, který eliminuje potřebu obnovovat stránku nebo znovu načítat celou stránku při každé interakci uživatele. AJAX interakce jsou zpracovávány asynchronně na pozadí, a uživatel tak může pokračovat v práci se stránkou. Interakce jsou iniciovány javascriptovým kódem. Když je interakce dokončena, JavaScript aktualizuje zdrojový kód HTML stránky. Změny jsou provedeny okamžitě bez nutnosti obnovení stránky. Uživatel spustí událost, například uvolněním klávesy při zadávání jména. Výsledkem je volání funkce JavaScript, která inicializuje objekt XMLHttpRequest. Objekt XMLHttpRequest je nakonfigurován s parametrem požadavku, který obsahuje ID komponenty, které událost spustila, a hodnotu, kterou uživatel zadal. Objekt XMLHttpRequest poté odešle asynchronní požadavek na webový server. Na webovém serveru požadavek zpracovává objekt servlet nebo listener. Data jsou načtena z datového úložiště a je připravena odpověď obsahující data ve formě XML dokumentu. Nakonec objekt XMLHttpRequest přijme data XML pomocí funkce zpětného volání, zpracuje je a aktualizuje HTML DOM (Document Object Model), aby zobrazil stránku obsahující aktualizovaná data. Níže je uvedena ukázka jednoduché webové aplikace AJAX. (38)

```
<html>
<head>
	<title>An Ajax Web Application</title>
</head>
<body>
<h1 id="page-title"></h1>
<button id="load-data">Click Here to Load the Data</button>
<script>
	document.getElementById("load-data").addEventListener("click",function(){
		var xmlhttp = new XMLHttpRequest();
		xmlhttp.open("GET", "data.txt");
```

```
            xmlhttp.onreadystatechange = function() {
                  document.getElementById("page-title").innerHTML            =
this.responseText;
            };
            xmlhttp.send();
      });
</script>
</body>
</html>
```

3.2.11. SOAP

Základ infrastruktury webových služeb tvoří tři základní technologie: WSDL (Web Services Description Language) – standardní formát pro popis rozhraní webové služby, SOAP (Simple Object Access Protocol) – protokol používaný pro komunikaci a UDDI (Universal Description, Discovery and Integration) – standardní mechanismus umožňující registraci a vyhledávání webových služeb. Základem webových služeb je protokol SOAP (Simple Object Access Protocol) pro posílání zpráv ve formátu XML mezi dvěma aplikacemi (princip peer-to-peer) – aplikace pošle zprávu v XML formátu jiné aplikaci, požadavek je obsloužen, výsledek zaslán jako druhá zpráva zpět (s kořenovým elementem Envelope – obálkou, v níž jsou vloženy dva elementy – hlavička Header, která je nepovinná, a tělo Body, ve kterém se přenáší informace o volané službě a předávaných parametrech). Zpráva funguje jako jednosměrný přenos informace od odesílatele k příjemci, nicméně kombinací několika zpráv lze implementovat běžné komunikační scénáře. K přenosu jsou využívány internetové protokoly HTTP nebo HTTPS. Podobně jako HTTP je SOAP bezestavový protokol. Používá se jako náhrada remote procedure call – vzdáleného volání procedur (RPC). SOAP je základní vrstva komunikace mezi webovými službami, která poskytuje prostředí pro složitější komunikaci. Webová služba je vyvolána serverem, který čeká na požadavky klientů a ve chvíli, kdy přes HTTP přijde SOAP zpráva, spustí webovou službu a předá jí požadavek. Výsledek je následně předán zpět klientovi jako odpověď. Další standardy WSDL a UDDI dále rozšiřují možnosti a jednoduchost použití SOAP. (39) V současnosti je SOAP nejčastěji používaným protokolem používaným pro RPC webové služby (komunikace probíhá formou požadavek – odpověď ve formátu XML, přičemž odpověď je zasílána vždy). Protokol určuje, jak volat

metody webové služby, jak metodám zadat parametry i jak bude vrácena návratová hodnota.

3.2.12. Document Object Model (DOM), virtuální DOM

K vytváření dynamických webových aplikací je často používán JavaScript a je tedy důležité, aby uměl manipulovat s jednotlivými komponenty webových stránek. Webová stránka se obvykle skládá z jednoho či více dokumentů typu HTML. Aby skriptovací jazyky dokázaly manipulovat s komponentami webové stránky, v HTML je k dispozici rozhraní označované jako Document Object Model (DOM). DOM je objektový model dokumentu, který umožňuje přistupovat k jednotlivým objektům XML (XHTML) dokumentu a pracovat s nimi. DOM je abstraktní reprezentací dokumentu HTML a slouží jako model pro strukturální změny stránek prostřednictvím skriptovacích jazyků. Webová stránka se po načtení stane objektem, ke kterému lze přistupovat přes vlastnost document nebo window.document. DOM Document Object je rodičovským elementem dokumentu HTML a vlastní všechny prvky aktuální webové stránky. Nad modelem dokumentu nebo kterýmkoli jeho potomkem lze manipulovat se stromem elementů, měnit a přidávat obsah elementů a jejich atributů, vyhledávat elementy na základě jejich vlastností (identifikátor, typ apod.) a registrovat události. (40)

DOM je jazykově neutrální rozhraní, které umožňuje programům a skriptům dynamicky přistupovat a aktualizovat více objektů, jako je obsah, struktura a styl dokumentu. DOM představuje dokument jako uzly a objekty. Jedná se o objektově orientovanou reprezentaci webové stránky upravenou skriptovacím jazykem, jako je JavaScript. Virtuální objekt DOM je reprezentace objektu DOM. Virtuální DOM je ideální pro využití v mobilních aplikacích. Virtuální DOM (VDOM) je koncept programování, kde je ideální nebo virtuální reprezentace uživatelského rozhraní uchovávána v paměti a synchronizována se skutečným DOM pomocí knihovny (například ReactDOM – virtuální DOM byl jedním z hlavních odlišujících prvků a výhod React ve srovnání s předchozími frameworky a novější knihovny začaly používat stejný přístup – například Vue.js). Tento proces je označován jako reconciliation. Virtuální DOM je odlehčená javascriptová reprezentace DOM používaná ve webových frameworcích, jako jsou React, Vue.js a Elm. Aktualizace virtuálního DOM je rychlejší než aktualizace DOM. Framework hledá rozdíly

mezi předchozím virtuálním DOM a aktuálním a na základě toho provede pouze nezbytné změny skutečného DOM. (41)

DOM, zkratka pro Document Object Model, je datová reprezentace struktury a obsahu dokumentu v prohlížeči. Skládá se z objektů. Protože reprezentuje dokument prohlížeče a skládá se z objektů, lze s ním manipulovat pomocí skriptovacích jazyků, jako je JavaScript. DOM může vypadat následovně:

```
<html>
   <body>
      <div class="welcome-page">
        <h1>Welcome!</h1>
        <p>Hello!</p>
      </div>
   </body>
</html>
Po přidání další značky "p".
<html>
   <body>
      <div class="welcome-page">
        <h1>Welcome!</h1>
        <p>Hello!</p>
        <p>Bye!</p>
      </div>
   </body>
</html>
```

Pro vyšší efektivitu je používán virtuální DOM s ReactJS. Při aktualizaci nebo úpravě kódu, který bude vykreslen do (skutečného) DOM, ReactJS analyzuje oba DOM a porovná je. Když ReactJS vidí, že se změnil pouze některý obsah, implementuje do DOM pouze tyto změny. Porovnáním DOM je eliminována nutnost provádět vykreslování veškerého obsahu. ReactJS uvidí tuto změnu a překreslí každý jednotlivý prvek do virtuálního DOM, ať už je nový, nebo ne. Jakmile je virtuální DOM plně aktualizován, ReactJS jej porovná s DOM. ReactJS pak vykreslí pouze identifikované změny, tzn. nevykreslí celý strom DOM. (42)

Ve webových aplikacích se stále více využívá principu jednostránkových aplikací (výhodou jednostránkových aplikací je jednoduché uživatelské rozhraní, rychlejší načítání

a snadná orientace – veškerý obsah je shromážděn na jedné stránce), které manipulují s rozhraním DOM. Na jednu stranu je stromová struktura rozhraní DOM nenáročná na procházení, na druhou stranu její změny jsou pro potřeby jednostránkových aplikací pomalé a závislé na konkrétním typu webového prohlížeče. Souvisí to s procesem renderování stránky, který probíhá v několika fázích – vytváření stromu DOM z analýzy dokumentu HTML (načtením a analýzou dokumentu HTML jsou elementy postupně konvertovány na DOM uzly a shora dolů postupně vzniká strom). Další fází je aplikování stylů z analýzy pravidel CSS, kdy jsou ke stromu postupně připojována CSS pravidla a vzniká strom pro renderování, který obsahuje vizuální vlastnosti (například velikost, barva pozadí apod.). Další fází je rozmístění elementů, kdy je strom rozšířen o přesné souřadnice, kde na obrazovce mají být umístěny. Následuje vykreslení stránky na obrazovku webového prohlížeče. Problematické je, že při neustálých změnách grafického rozhraní je často aktualizováno více uzlů a proces renderování se opakuje pro každý uzel. (43)

3.2.13. WSDL

WSDL (Web Services Description Language) je jazyk na bázi XML (Extensible Markup Language) určený pro popis webových služeb. Je výsledkem spolupráce firem IBM a Microsoft. Vznikl z potřeby sjednocení jazyka používaného pro popis rozhraní webových služeb. Webové služby umožňují „*jednoduchou komunikaci mezi aplikacemi ve velmi heterogenním prostředí, protože komunikace je založena na platformě nezávislých standardů – především na jazyce XML a protokolu HTTP. Aplikace si mezi sebou posílají XML zprávy, které přenášejí dotazy a odpovědi jednotlivých aplikací.*“ (44) Aby bylo možné volat protokolem SOAP, je nezbytné vědět co a jak volat. K tomu je určen právě jazyk WSDL pro popis rozhraní služby. WSDL popisuje jména dostupných operací, typy jejich parametrů a návratových hodnot, a kde a jak je služba dostupná (port, URL, HTTP/HTTPS/SMTP). Naopak nepopisuje sémantiku operací, pouze syntaxi jejich volání. WSDL je založen na XML, a obsahuje následující hlavní tagy:

<types> definice datových typů

<portType> souhrn operací – odpovídá interface (Java) nebo header souboru (C)

<message> komunikační zpráva – odpovídá zavolání nebo návratu z funkce

<binding> definuje možný způsob přístupu různými protokoly

<service> a <port> pro každý <binding> definují adresu, na kterou se má příslušný protokol spojit

<operation> odpovídá metodě (Java) nebo funkci (C)

WSDL je dokument, který popisuje rozhraní služby definující jména a typy vstupních parametrů, operací a návratové hodnoty. Poskytuje též informace o způsobech, jak službu volat (jaký protokol použít, na kterém portu a URL bude dostupná atd.). Operace je definována vstupní a výstupní zprávou – klient posílá vstupní a server výstupní zprávy (které se skládají z jedné nebo více logických částí a nesou vstupní či výstupní informace). Lze rozlišit operace jednosměrné (one-way), kdy klient posílá zprávu, na kterou server nereaguje; operace typu žádost-odpověď (solicit-response), kdy server pošle zprávu a klient odpoví; operace typu požadavek-odpověď (request-response), kdy klient posílá zprávu a server odpoví; a operace typu notifikace (notification), kdy server sdělí zprávu klientovi. Funkce WSDL je tedy podobná hlavičkovým souborům v jazyce C, případně rozhraní v Javě. Jestliže je WSDL správně specifikován, obsahuje všechny informace potřebné pro využívání služby. „*Jazyk WSDL slouží k popisu síťových služeb jako množiny koncových bodů zpracovávajících zprávy. Operace a zprávy jsou popisovány na abstraktní úrovni a teprve poté jsou svázány s konkrétním síťovým protokolem a datovým formátem. To umožňuje snadné vytvoření popisu rozhraní, které nabízí jednu službu několika způsoby. V praxi WSDL popisy nejčastěji popisují služby, které si posílají zprávy pomocí formátu SOAP a protokolu HTTP.*“ (45)

WSDL byl prezentován World Wide Web Consortium (W3C) jako průmyslový standard pro popis webových služeb. Dokument WSDL definuje služby jako soubor síťových koncových bodů nebo portů. Ve WSDL jsou abstraktní definice koncových bodů a zpráv odděleny od jejich síťové implementace či vazeb datových formátů. Toto oddělení podporuje opětovné použití abstraktních definic – Zpráv, což jsou abstraktní popisy vyměňovaných dat, a typů Port, které představují abstraktní kolekce operací. Konkrétní specifikace protokolu a formátu dat pro konkrétní typ Port vytváří opakovaně použitelnou vazbu. Port je definován přidružením síťové adresy k opakovaně použitelné vazbě a kolekce portů definuje službu. Proto se dokument WSDL skládá z několika prvků: typ (kontejner pro definice datových typů pomocí nějakého systému typů), operace jako abstraktní popis akce podporované službou, zpráva jako abstraktní typová definice dat, port jako unikátní koncový bod definovaný jako kombinace síťové adresy a vazby, vazba

(obvykle SOAP) jako specifikace konkrétního datového formátu a protokolu pro konkrétní typ Port, služba jako kolekce souvisejících koncových bodů a typ Port jako abstraktní sada operací podporovaná alespoň jedním koncovým bodem. (46)

3.3. Principy vývoje

Psaní dobrého a kvalitního kódu je náročné. Proto je vhodné uplatňovat některé základní principy softwarového inženýrství. Tyto principy jsou souborem pokynů, které programátorovi pomáhají vytvářet lepší, robustnější a čistý kód. Výsledkem je vytváření vysoce kvalitního softwaru. K základním principům patří principy DRY, KISS a YAGNI. DRY pochází z Don't Repeat Yourself (neopakujte se) – tzn. programátor nepíše totožný kód na více míst, nebo ho bude muset synchronizovat při každé změně kódu, což by bylo zdlouhavé. KISS pochází z Keep It Simple, Silly (Stupid) – tento princip říká, že pokud se programátor vyvaruje zbytečné složitosti, bude jeho program robustnější. Podobný princip je YAGNI (You Aren't Gonna Need It – nebudeš to potřebovat).

3.3.1. YAGNI

Platí, že při vývoji softwaru by měla vždy existovat rovnováha mezi pokusem a omylem a plánováním (přičemž plánování dlouho dopředu může projekt spíše zkomplikovat než usnadnit). Princip YAGNI (You ain‘t gonna need it nebo You Aren't Gonna Need It, Nebudeš to potřebovat) odkazuje na předvídání budoucího vývoje, přílišnou analýzu a implementaci prvků, které mohou, ale nemusí být potřeba. Princip YAGNI řeší tyto problémy tím, že odmítá tradiční, plánovanou metodu vývoje softwaru. Tento princip souvisí s programovacím konceptem inkrementálního vývoje – tzn. vytvářet jeden prvek po druhém a průběžně testovat jejich funkčnost. Pochopení definice YAGNI je snadné, jeho implementace v praxi je náročnější. YAGNI je jedním z klíčových principů extrémního programování. Extrémní programování zlepšuje softwarový projekt několika způsoby – usnadňuje komunikaci, zvyšuje jednoduchost, zajišťuje zpětnou vazbu, respekt a odvahu. Programátoři neustále komunikují se svými zákazníky i s ostatními programátory. Udržují kód jednoduchý a čistý. Získávají zpětnou vazbu testováním svého softwaru. Dodávají produkt zákazníkům co nejdříve a implementují změny podle návrhu. Každý malý úspěch prohlubuje jejich respekt k práci každého člena týmu a s tímto základem jsou programátoři schopni odvážně reagovat na měnící se požadavky a technologie. Princip YAGNI tedy radí vždy implementovat další prvky, až když je programátor skutečně

potřebuje, nikoliv když předvídá či předpokládá, že je bude potřebovat. Tento přístup minimalizuje množství zbytečné práce, která zůstane nevyužitá a nedokončená, což zvyšuje produktivitu vývojářů i jednoduchost produktu. YAGNI funguje nejlépe v prostředích, ve kterých je kód udržován v tvárném a snadno měnitelném stavu. Tento princip úzce souvisí s principem Keep It Simple Stupid. Tím, že se programátor vyhne přidávání funkcí a složitosti do kódu, dokud to není skutečně potřeba, může být celkový návrh systému jednodušší. (47)

3.3.2. KISS

KISS je zkratka pro Keep It Simple Stupid. Tento princip vyjadřuje snahu udržovat kód co nejjednodušší. Čím jednodušší je kód, tím snazší je ho pochopit a udržovat. Klíčem k dodržování tohoto principu je naučit se rozdělit řešený problém do co nejjednodušších a nejmenších kroků a následně vytvořit co nejjednodušší kód k jejich implementaci. Princip KISS tedy odkazuje na skutečnost, že jednoduché systémy se snáze spravují. Na druhou stranu udržovat věci jednoduché neznamená nutně méně práce, neboť takový přístup vyžaduje více znalostí k implementaci. Programátoři uplatňují tento princip ve dvou rovinách – při návrhu aplikace (platí, že čím více funkcí bude aplikace mít, tím náročnější bude údržba kódu). Jednou ze základních zásad je nesnažit se napsat dokonalou aplikaci. Při návrhu webové aplikace je důležité zjistit, co od ní očekávají různé cílové skupiny a jakým způsobem ji budou používat – první verze aplikace pak bude mít jen základní funkce (viz koncept minimálního životaschopného produktu) a postupně jsou v případě potřeby přidávány další. Druhou úrovní, ve které programátoři využívají princip KISS, je při psaní aplikace, která je rozdělena na menší části a pro ty je psán co nejjednodušší a snadno měnitelný zdrojový kód. V opačném případě by byl kód komplexní a složitý (špagetový kód) a byla by nutná jeho refaktorizace (zpřehlednění a zjednodušení kódu při zachování jeho funkčnosti). Jestliže se vývojář potýká s nějakým problémem, řešením je tedy rozdělit ho na menší a srozumitelné části, o nichž si myslí, že jim rozumí, a poté se pokusit implementovat řešení v kódu. V opačném případě vznikají komplikované implementace i těch nejjednodušších problémů a dalším nežádoucím vedlejším efektem je špagetový kód. Aplikací principu KISS bude vývojář schopen vyřešit rychleji více problémů, vytvářet kvalitnější, snadno udržovatelný a jednodušší kód (s méně řádky) i pro řešení složitých problémů. (48)

3.3.3. DRY

Zkratka DRY(Don't Repeat Yourself) doslova znamená Neopakuj se. Tento princip (někdy označovaný jako Single Source of Truth, SSoT) se zaměřuje na omezení opakovaného používání stejného kódu. To znamená, že při změně určitého jednotlivého prvku by neměla být vyžadována změna v logicky nesouvisejících komponentách. Platí, že méně kódu má za následek jeho lepší udržovatelnost. Nejběžnějším příkladem jsou funkce. Obecným pravidlem je, že pokud programátor píše totožnou logiku více než třikrát, je vhodné vytvořit funkci a v kódu ji opakovaně použít. Cílem tohoto principu je eliminace duplicitních informací a vytváření jednoduššího a kvalitnějšího kódu. Princip DRY zahrnuje veškeré zdroje informací v systému „*od zdrojového kódu přes datové schéma po dokumentaci*“. (49)

Každý řádek kódu v aplikaci je možným zdrojem chyb. Při nedodržení principu DRY je zvýšené riziko výskytu chyb v programu (obzvláště při kopírování částí kódu na různá místa hrozí, že při následné opravě chyb programátor opomene některý z překopírovaných řádků chybného kódu). S tím souvisí náročnější údržba kódu, kdy při změně kódu je nutné změnit i další příslušná místa. Riziko je vyšší v případě, že změny provádí jiná osoba než autor původního kódu, případně změna proběhla již před delší dobou. Protikladem principu DRY je WET (Write Everything Twice, We Enjoy Typing nebo We Edit Too much). Na druhou stranu ne všechny totožné úseky kódu znamenají porušení principu DRY – kritériem pro posouzení je fakt, zda příslušné úseky spolu logicky souvisejí (tzn. vyjadřují-li shodnou informaci, či nikoliv). Nežádoucím efektem principu DRY je rostoucí provázanost, kdy dochází ke spojení úseků kódu, které spolu nesouvisí. Je tedy vhodné princip aplikovat jen do té míry, která usnadňuje psaní jednoduššího kódu (například při tvorbě HTML šablon je nedodržení principu tolerováno, neboť jeho dodržení by nebylo efektivní). Podmnožinou principu DRY je princip Once and only once, podle kterého musí být vše ve zdrojovém kódu vyjádřeno pouze jednou. S tím souvisí princip abstrakce, „*pokud se v kódu vyskytuje stejná nebo podobná implementace stejné funkcionality v různých částech, mělo by dojít ke vzniku abstrakce těchto částí*“. (49)

4 Vlastní práce

4.1. Výchozí analýza

V rámci této analýzy byla zjištěna současná situace ve společnosti ve vztahu k evidenci smluv, kterou je potřeba změnit dle požadavku vycházejících ze zadání, které bude připraveno.

Tato společnost aktuálně disponuje aplikací, která je ale zastaralá a nevyhovující svou funkcionalitou. Velká část smluv jednotlivých úseků společnosti je kvůli této skutečnosti evidovaná v tabulkách v MS Excelu.

Tento model práce se smlouvami byl vyhovující při menším počtu smluv, ale se zvyšováním počtu smluv se tento model stal nevyhovujícím jak kvůli nepřehlednosti, tak kvůli časové náročnosti zaevidování jednotlivých dokumentů.

Cílem této diplomové práce je provést návrh a implementaci webové aplikace pro evidenci smluv za pomoci webových technologií.

4.2. Tvorba zadání

Evidované údaje:

- Číslo smlouvy;
- Název smlouvy;
- Dodatky ke smlouvě (datum platnosti, č. dodatku);
- Subjekt smlouvy (organizační útvar);
- Název partnera (kategorie/druh/smluvní partner);
- Garant smlouvy;
- Datum platnosti smlouvy;
- Kategorie smlouvy;
- Účinnost (ano/ne);
- Osobní údaje;

Parametry pro setřídění výsledků vyhledávání a omezení platnosti: zachovat dle stávající evidence smluv.

4.2.1. Vkládání smluv, vyhledávání, zobrazování výsledků

Smluvní dokumentaci bude do Evidence smluv vkládat a dále spravovat Editor (viz níže bod 4.5.2).

- V evidenci smluv bude možné vyhledávat smlouvy dle všech výše uvedených evidovaných údajů (jednotlivě i v kombinaci);
- Výsledek vyhledávání (přehled smluv) bude možné exportovat (např. do formátu MS Office);
- Výsledek vyhledávání (přehled smluv) se bude zobrazovat všem bez ohledu na typ přístupového oprávnění;
- Konkrétní evidovaná smluvní dokumentace se zobrazí dle typu přístupového oprávnění (viz bod 4.5.2).

4.2.2. Role a přístupová oprávnění

Editor:

Asistentka a náhradník: právník na smluvní agendu.

Role:

- vkládat nové smlouvy, vč. vyplnění evidovaných údajů (ad 1);
- úprava evidovaných údajů, vč. přidělování přístupových práv dle pokynu garanta; archivace smluv (vyznačením);
- zobrazení všech smluv, vč. archivovaných smluv.

Vedení:

VŘ, asistentka VŘ, tajemnice VŘ, vedoucí oddělení, ředitel IT služeb, poradce VŘ, provozní ředitel.

- rozhodnutí o přiřazování garantů a uživatelů do skupin odpovídajících Kategorii smlouvy;
- zobrazení veškerých (i neúčinných) smluv ve své kategorii.

Garant:

Garantem je role přirazená ke každé Kategorii smluv – osoba, která smlouvu sjednává a věcně odpovídá za agendu v příslušné Kategorii smluv, kterou smlouva upravuje.

Role:

- určí evidované údaje na formuláři ke smlouvě;
- zobrazení všech smluv v Kategoriích smluv, u kterých je garantem.

Uživatel:

Uživatelem je role přirazená ke každé Kategorii smluv – zaměstnanci příslušných oddělení, kteří zobrazení smlouvy potřebují pro svou práci.

Nadefinování Uživatelů do skupin pro jednotlivé Kategorie smluv provedou Garanti s předstihem před spuštěním Evidence smluv.

Role:

- zobrazení účinných smluv v Kategoriích smluv, u kterých je uživatel ve skupině.

Administrátor:

Role:

- přidělování a správa přístupových práv dle pokynů Garanta.

4.2.3. Logování

Evidence smluv bude logovat informaci o veškerých akcích, které v aplikaci byly provedeny v rozsahu:

- datum;
- čas;
- akce;
- osoba.

4.2.4. Zadání nové smlouvy

Písemné vyhotovení smlouvy bude Garantem bezprostředně po jejím podepsání poslední smluvní stranou předáno Editorovi, který do Evidence smluv zadá údaje evidované o smlouvě. Okamžikem uložení se smlouva zobrazí všem Uživatelům, kteří jsou zařazeni ve skupině odpovídající Kategorii smlouvy.

4.2.5. Aplikace automaticky provede:

- provádí výpočet dní a automaticky upravuje údaj o účinnosti smlouvy dle platnosti smlouvy;
- zasílá info o blížícím se konci účinnosti smlouvy;
- provede archivaci po ukončení účinnosti.

4.2.6. Archivace smluv

Smlouvy budou archivovány v aplikaci evidence smluv Editorem/Automatizovaně po pozbytí jejich účinnosti.

V případě potřeby Editor vyhledá archivovanou smlouvu a její obsah vhodným způsobem zpřístupní žadateli. Skartovány budou smlouvy ve lhůtách dle Skartačního plánu.

4.3. Analýza řešení dostupných na trhu

Na základě sdělených priorit společnosti byla stanovena tato kritéria a váhy:

- Modularita systému – aby produkt obsahoval takovou strukturu modulů, která bude nejlépe vyhovovat požadavkům na aplikaci a pokrývat požadavky na podnikové procesy,
  váha přiřazená tomuto kritériu: 45,
- Servisní dostupnost – aby byl dodavatel po ruce vždy, když je potřeba,
  váha přiřazená tomuto kritériu: 10,
- Cena,
  váha přiřazená tomuto kritériu: 20,
- Možnosti do budoucna – aby nebyla aplikace v budoucnu firmě „malá“, ale mohla být nadále rozšiřována,
  váha přiřazená tomuto kritériu: 25.

Naši potencionální dodavatelé software jsou společnosti INSIO software, s. r. o., a Aptien Labs, s. r. o., anebo interní vývoj aplikace. Dle zjištěných informací na webech společnosti můžeme přistoupit rovnou k bodovému hodnocení jednotlivých nabídek podle stanovených kritérií:

Jednotlivé nabídky členové řešitelského týmu podle stanovených kritérií hodnotili takto:

1. hodnotitel:

Kritérium / Produkty	interní vývoj	INSIO software, s. r. o.	Aptien Labs, s. r. o.
modularita systému	5	1	1
servisní dostupnost	3	3	1
cena	4	2	3
možnosti do budoucna	3	2	3

Tabulka 1 – Hodnocení nabídky od 1. hodnotitele

2. hodnotitel:

Kritérium / Produkty	interní vývoj	INSIO software, s. r. o.	Aptien Labs, s. r. o.
modularita systému	4	2	2
servisní dostupnost	3	3	2
cena	5	2	3
možnosti do budoucna	5	2	3

Tabulka 2 – Hodnocení nabídky od 2. hodnotitele

Poté, co jsme stanovili bodové hodnocení nabídek, je třeba vypočítat jejich aritmetický průměr podle každého kritéria pro každou společnost:

$$\bar{x}_{ij} = \frac{\sum_{i=1,\,j=1}^{n,\,m} x_{ij}}{n}$$

Obrázek 1 – Vzorec pro výpočet (Zdroj: Vlastní)

Kde *i* je kritérium, *j* je produkt, *n* je počet kritérií a *m* je počet produktů. Výsledky jsou zobrazeny v následující tabulce:

Kritérium	Průměrné bodové hodnocení		
	interní	INSIO	Aptien
modularita systému	4,5	1,5	1,5
servisní dostupnost	3	3	1,5
cena	4,5	2	2
možnosti do budoucna	4	2	2

Tabulka 3 – Hodnocení nabídky od společností

V poslední fázi bylo stanoveno pořadí jednotlivých nabídek jako součet součinů vah a průměrného bodového hodnocení u jednotlivých kritérií pro každou z nabídek. Výpočet byl proveden opět pro všechny varianty řešení.

4.3.1. Produkt interní vývoj:

y1 = 0,45 * 4,5 + 0,1 * 3 + 0,2 * 4,5 + 0,25 * 4 = 4.225
vyjadřuje, že vážené bodové hodnocení nabídky produktu interní vývoj podle zvolených kritérií je rovno 4.225.

4.3.2. Produkt INSIO software, s. r. o.:

y2 = 0,45 * 1,5 + 0,1 * 3 + 0,2 * 2 + 0,25 * 2 = 1.875.
vyjadřuje, že vážené bodové hodnocení nabídky produktu INSIO software, s. r. o., podle zvolených kritérií je rovno 1.875.

4.3.3. Produkt Aptien Labs, s. r. o.:

y3 = 0,45 * 1,5 + 0,1 * 1,5 + 0,2 * 2 + 0,25 * 2 = 1.725.
vyjadřuje, že vážené bodové hodnocení nabídky produktu Helios podle zvolených kritérií je rovno 1.725.

Z toho vyplývá, že nejvhodnější nabídka pro společnost je varianta y1, tedy nabídka produktu interní vývoj. Bylo tedy rozhodnuto provést vývoj aplikace pro evidenci smluv interně ve společnosti. Bylo tedy rozhodnuto provést vývoj aplikace pro evidenci smluv interně ve společnosti.

Toto rozhodnutí bylo především ovlivněno tím, že společnosti, které nabízejí balíčkové řešení, mají pevně daný rozsah evidovaných funkcí a modulu, ani jedna z variant nesplňovala předpoklady společnosti svou nabídkou. Společnosti nabízely úpravu aplikace „Na míru“, ale pokud by této možnosti bylo využito, tak cena aplikace razantně stoupá.

Další nevýhodou je to že dodávající společnosti mají pevně danou cenu za jednu licenci pro uživatele.

4.4. Návrh řešení

Na základě analýzy výše uvedených požadavků od společnosti bylo vytvořeno zadání a byly vybrány technologie, které budou použity pro tvorbu webové aplikace, a jaké moduly budou nejvhodnější pro splnění požadavku společnosti.

Jelikož konkrétní technologie společnost nedefinovala, byly vybrány technologie dle osobních zkušeností vývojáře a to jsou konkrétně tyto:

- PHP – jako jazyk programování
- Laravel – jako hlavní framework
- JavaScript – jako skriptovací jazyk,
- Bootstrap – jako frontend framework,
- DataTables – jako prlug-in pro práci s tabulkami,
- Dropzone – jako knihovna pro práci se soubory,
- Oracle Database – jako databázový systém.

4.5. Popis prostředí

Framework Laravel má několik systémových požadavků. Všechny tyto požadavky jsou splněny virtuálním strojem s operačním systémem Windows Server 2022, následovně byla provedena konfigurace Apache serveru, kde byla zajištěna podpora následujících požadavků:

- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- BCMath PHP Extension

4.6. Instalace Laravel

Instalace samotného frameworku byla provedena následujícím příkazem:

```
composer global require laravel/installer
```

Nasledně bylo provedeno vytvoření nového projektu pomocí následujícího příkazu:

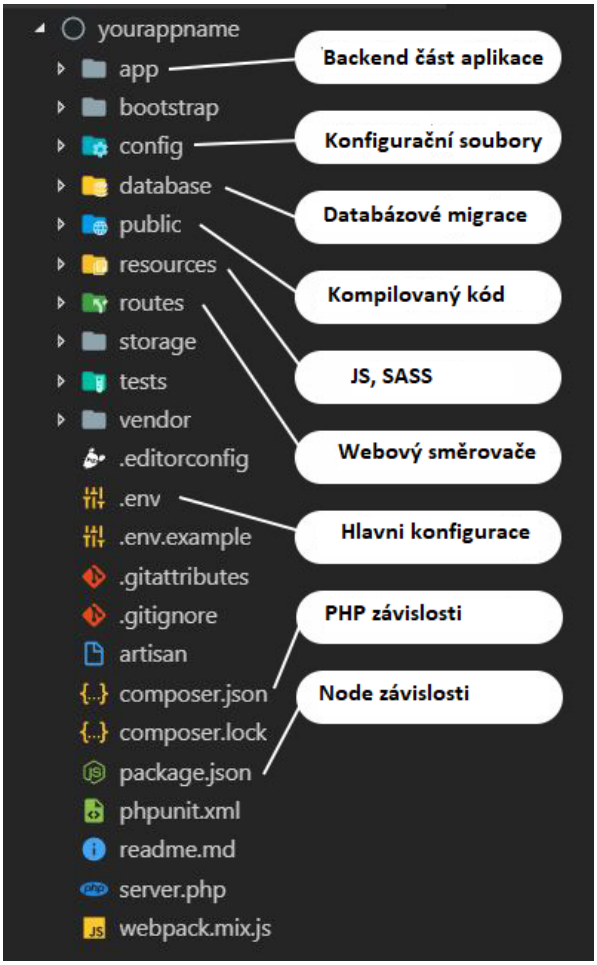
```
laravel new název_projektu
```

Dále bylo potřeba vytvořit aplikační klíč:

```
php artisan key:generate
```

Tento řetězec je obvykle 32 znaků dlouhý. Pokud není klíč aplikace nastaven, nebudou uživatelské relace a další šifrovaná data bezpečné.

Pro provedení veškerých příkazů je generovaná struktura projektu, která je vyobrazena na obrázku níže:



Obrázek 2 - Struktura Laravel (Zdroj: Vlastní)

Laravel obsahuje soubor public/.htaccess, který slouží k poskytování adres URL bez předního řadiče index.php v cestě, který byl upraven následovně:

```
Options +FollowSymLinks –Indexes
RewriteEngine On
RewriteCond %{HTTP:Authorization} .
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [L]
```

4.7. Konfigurace plug-in DataTables

Plug-In DataTables neobsahuje speciální znaky z českého jazyka, proto základní skript byl upraven následovně:

```
$.extend( $.fn.dataTableExt.oSort, {
   "czech-pre": function ( a ) {
        var special_letters = {
            "A": "Aa", "a": "aa", "Á": "Ab", "á": "ab",
            "C": "Ca", "c": "ca", "Č": "Cb", "č": "cb",
            "D": "Da", "d": "da", "Ď": "db", "ď": "db",
            "E": "Ea", "e": "ea", "É": "eb", "é": "eb", "Ě": "Ec", "ě": "ec",
            "I": "Ia", "i": "ia", "Í": "Ib", "í": "ib",
            "N": "Na", "n": "na", "Ň": "Nb", "ň": "nb",
            "O": "Oa", "o": "oa", "Ó": "Ob", "ó": "ob",
            "R": "Ra", "r": "ra", "Ř": "Rb", "ř": "rb",
            "S": "Sa", "s": "sa", "Š": "Sb", "š": "šb",
            "T": "Ta", "t": "ta", "Ť": "Tb", "ť": "tb",
            "U": "Ua", "u": "ua", "Ú": "Ub", "ú": "ub", "Ů": "Uc", "ů": "uc",
            "Y": "Ya", "y": "ya", "Ý": "Yb", "ý": "yb",
            "Z": "Za", "z": "za", "Ž": "Zb", "ž": "zb"
        };
        for (var val in special_letters)
            a = a.split(val).join(special_letters[val]).toLowerCase();
        return a;
    },
    "czech-asc": function ( a, b ) {
        return ((a < b) ? -1 : ((a > b) ? 1 : 0));
    },
    "czech-desc": function ( a, b ) {
        return ((a < b) ? 1 : ((a > b) ? -1 : 0));
    }
} );
```

4.8. Konfigurace plug-in DropZone

Jelikož jako komponenta pro práci se soubory byla vybraná knihovna DropZone, bylo ji potřeba konfigurovat pro účely nahrávání souboru do spisové služby následovně:

```
   <script type="text/javascript">
           Dropzone.options.dropzone =
               {
```

```
                    maxFilesize: 48,
                    renameFile: function (file) {
                        var dt = new Date();
   var time = dt.getTime();
   return time + file.name;
                    },
                    acceptedFiles: ".jpeg,.jpg,.png,.gif,.pdf",
                    addRemoveLinks: true,
                    timeout: 50000,
                    removedfile: function (file) {
                        var name = file.upload.filename;
                        console.log(file);
                        $.ajax({
                            headers: {
                                'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')
                            },
                            type: 'POST',
                            url: '{{ url("delete") }}',
                            data: {filename: name},
   success: function (data) {
                                // console.log("Soubor uspěšně smazan!!");
   if (window.formFiles != null) {
                                    delete window.formFiles[name];
                                }
                            },
                            error: function (e) {
                                console.log(e);
                            }
                        });
                        var fileRef;
   return (fileRef = file.previewElement) != null ?
                            fileRef.parentNode.removeChild(file.previewElement)
: void 0;
                    },
                    success: function (file, response) {
                        console.log(response);
                        if (window.formFiles == null) {
```

```
                        window.formFiles = {};
                    }
26.03.2022 10/14
                    var file = response.success;
                    window.formFiles[file.filename] = true;
                        document.getElementById('smlouvy'
).value = JSON.stringify( Object.keys(window.formFiles || {}));
                    console.log(Object.keys(window.formFiles || {}));
                },
                error: function (file, response) {
                    return false;
                }
            };
    </script>
```

4.9. Konfigurace SOAP modulu spisové služby

Jedním z požadavků zadavatelské společnosti bylo, že veškeré nahrané soubory nebudou přístupné nikde na serveru, ale budou přenesené do spisové služby. Spisová služba ve společnosti, pro kterou je aplikace určená, má své SOAP rozhraní, pomocí kterého dokáže přenášet soubory ve formátu base64.

Z tohoto důvodu byla provedena konfigurace a implementace ve vyvíjené aplikaci pro tento modul, kód pro implementaci nahrávání a stahování souboru je vyobrazen níže:

Modul nahrávání do spisové služby:

```
$auth = array(
                "login" => config('wsdlauth.wsdllogin'),
                "password" => config('wsdlauth.wsdlpassword')
            );
            $fileName ="Nahrane PDF";
            $fileId="";
            $title= $rok.$nazev.$skupina.$utvar;
            $extension="pdf";
            $contentType="application/pdf";
            $search_query = new StdClass();
            $search_query->array =[];
            foreach ($jfile as $file)
            {
                $file = "images/$file";
```

```
                    $handle = fopen($file,'rb');
                    $file_content = fread($handle,filesize($file));
                    fclose($handle);
                    $fileDef = new StdClass();
                    $fileDef->fileId =$fileId;
                    $fileDef->title = $title;
                    $fileDef->extension = $extension;
                    $fileDef->contentType = $contentType;
                    $fileDef->data = $file_content;
                    $search_query->array[]=($fileDef);
                }
                $numberContract=$cislo;
                $titleContract=$nazev;
                $createUserDT = date("Y-m-d H:i:s");
                $createBy=$n;
                $client = new soapclient(config('wsdlauth.wsdl'),$auth);
                $response = $client->contractCreateOrUpdate(array(
                    "id",
                    "numberContract"=>$numberContract,
                    "titleContract"=>$titleContract,
                    "subjectContract"=>$skupina,
                    "partnerName"=>$kategoriepartnera,
                    "categoryContract"=>$kategorie,
                    "createUserDT"=>$createUserDT,
                    "createBy"=>$createBy,
                    "files"=>$search_query,
                ));
                $dokument="";
                if (is_array($response->files->array))
                {
                    foreach ($response->files->array as $file)
                    {
                        $dokument.= '<Doklad Spisovka="' . $file->fileId .
'"/>';
                    }
                }
                else
                {
```

```
                    foreach ((array)$response->files->array->fileId as $file)
                    {
                        $dokument.= '<Doklad Spisovka="' . $file . '"/>';
                    }
                }
            }
```

Modul stahovani souboru ze SPSL

```
@if (count($doklady)===1)
        @foreach ($id_spisovka as $file )
            <button id="<?php echo $file;?>">Stáhnout soubor Č<?php echo
$file;?></button>
            <script>
                document.getElementById('<?php echo $file;?>').onclick =
function() {
                    var phpadd= atob(`<?php echo base64_encode(
getsfile($file));?>`);
                    let a = document.createElement("a");
                    a.href = "data:application/octet-stream;base64,"+phpadd;
                    a.download = "documentName.pdf"
                    a.click();
                };
            </script>
        @endforeach
        <hr>
    @endif
```

4.10. Implementace rozhraní SŘBD

- Seznam smluv

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetSeznam Akce="S" Stav="A"/> <!-- A = Aktivní, R = Archivní  -->
</Dotaz>
<Odpoved>
     <GetSeznam Akce="S" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Smlouvy>
               <Smlouva Id="" Rok="" Cislo="" Skupina="" Utvar="" Partner=""
Nazev=""
PlatnostOd="" PlatnostDo="" Kategorie="" OsobniUdaje="" ="" Stav=""
TlZobrazit="" TlUpravit=""
TlDodatek=""/>
          </Smlouvy>
```

```
          <Error Text=""/>
     </GetSeznam>
</Odpoved>
Seznam číselníků
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetCiselniky Akce="V"/>
</Dotaz>
<Odpoved>
     <GetCiselniky Akce="V" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Slozky>
               <Slozka Nazev="">
                    <Utvary>
                         <Utvar Nazev=""/>
                    <Utvary/>
               </Slozka>
          </Slozky>
          <PartnerKategorie>
               <Kategorie Nazev=""/>
          </PartnerKategorie>
          <PartnerDruh>
               <Druh Nazev=""/>
          </PartnerDruh>
          <SmlouvaKategorie>
               <Kategorie Nazev=""/>
          </SmlouvaKategorie>
          <SmlouvaStav>
               <Stav Nazev=""/>
          </SmlouvaStav>
          <OsobniUdaje>
               <Udaje Nazev=""/>
          </OsobniUdaje>
          <Tlacitka TlNovy="" TlArchiv="" TlGarant="" TlUzivatel=""/>
          <Error Text=""/>
     </GetCiselniky>
</Odpoved>
```

- Nová smlouva

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetSmlouva Akce="N">
          <Smlouva Rok="" Nazev="" Skupina="" Utvar="" PlatnostOd=""
PlatnostDo="" Kategorie="
" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
     </SetSmlouva>
</Dotaz>
<Odpoved>
     <SetSmlouva Akce="N" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
```

```
          <Error Text=""/>
     </SetSmlouva>
</Odpoved>
Načtení smlouvy
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetSmlouva Akce="E">
          <Smlouva Id=""/>
     </GetSmlouva>
</Dotaz>
<Odpoved>
     <GetSmlouva Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Smlouva Id="" Rok="" Cislo="" Nazev="" Skupina="" Utvar=""
PlatnostOd="" PlatnostDo
="" Kategorie="" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
          <Dodatky>
               <Dodatek Id="" Rok="" Cislo="" Nazev="" />
          </Dodatky>
          <Error Text=""/>
     </GetSmlouva>
</Odpoved>
```

- Uložení smlouvy

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetSmlouva Akce="E">
          <Smlouva Id="" Rok="" Cislo="" Nazev="" Skupina="" Utvar=""
PlatnostOd="" PlatnostDo
="" Kategorie="" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
     </SetSmlouva>
</Dotaz>
<Odpoved>
     <SetSmlouva Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Error Text=""/>
     </SetSmlouva>
</Odpoved>
```

- Nový dodatek smlouvy

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetDodatek Akce="N">
          <Dodatek IdSmlouvy="" Rok="" Nazev="" Skupina="" Utvar=""
PlatnostOd="" PlatnostDo="
```

```
" Kategorie="" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
     </SetDodatek>
</Dotaz>
<Odpoved>
     <SetDodatek Akce="N" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetDodatek>
</Odpoved>
```

- Načtení dodatku smouvy

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetDodatek Akce="E">
          <Dodatek Id="" />
     </GetDodatek>
</Dotaz>
<Odpoved>
     <GetDodatek Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Dodatek Id="" Rok="" Cislo="" Nazev="" Skupina="" Utvar=""
PlatnostOd="" PlatnostDo
="" Kategorie="" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
          <Error Text=""/>
     </GetDodatek>
</Odpoved>
```

- Uložení dodatku smouvy

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetDodatek Akce="E">
          <Dodatek Id="" Rok="" Cislo="" Nazev="" Skupina="" Utvar=""
PlatnostOd="" PlatnostDo
="" Kategorie="" OsobniUdaje="" Stav="" IdSpisovka=""/>
          <Partner Jmeno="" Prijmeni="" Rc="" Firma="" FirmaForma="" Ic=""
Dic="" Druh=""
Kategorie="" Ulice="" Psc="" Psc="" Obec="" Stat="" />
          <Doklady>
               <Doklad Spisovka=""/>
          </Doklady>
     </SetDodatek>
</Dotaz>
<Odpoved>
     <SetDodatek Akce="E" Zpracovano="A">
```

```
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Error Text=""/>
     </SetDodatek>
</Odpoved>
```

- Načtení seznamu garantů

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetGarant Akce="S" />
</Dotaz>
<Odpoved>
     <GetGarant Akce="S" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Garanti>
               <Garant Id="" Uzivatel="" Kategorie=""/>
          </Garanti>
          <Error Text=""/>
     </GetGarant>
</Odpoved>
```

- Nový garant

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetGarant Akce="N" />
</Dotaz>
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetGarant Akce="N">
          <Garant Uzivatel="" Kategorie=""/>
     </SetGarant>
</Dotaz>
<Odpoved>
     <SetGarant Akce="N" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetGarant>
</Odpoved>
```

- Editace garanta

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetGarant Akce="E" GarantId=""/>
</Dotaz>
<Odpoved>
     <GetGarant Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
          <SmlouvaKategorie>
               <Kategorie Nazev=""/>
          </SmlouvaKategorie>
          <Uzivatele>
               <Uzivatel PrijmeniJmeno=""/>
          </Uzivatele>
          </Editovany GarantId="" Uzivatel="" Kategorie="" />
     </GetGarant>
</Odpoved>
```

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetGarant Akce="E">
          <Garant Id="" Uzivatel="" Kategorie=""/>
     </SetGarant>
</Dotaz>
<Odpoved>
     <SetGarant Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetGarant>
</Odpoved>
```

- Smazání garanta

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetGarant Akce="D">
          <Garant Id=""/>
     </SetGarant>
</Dotaz>
<Odpoved>
     <SetGarant Akce="D" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetGarant>
</Odpoved>
```

- Načtení seznamu pracovniků

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetPracovnik Akce="S" />
</Dotaz>
<Odpoved>
     <GetPracovnik Akce="S" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error -->
          <Pracovnici>
               <Pracovnik Id="" Pracovnik="" Kategorie=""/>
          </Pracovnici>
          <Error Text=""/>
     </GetPracovnik>
</Odpoved>
Nový pracovnik
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetPracovnik Akce="N" />
</Dotaz>
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetPracovnik Akce="N">
          <Pracovnik Pracovnik="" Kategorie=""/>
     </SetPracovnik>
</Dotaz>
<Odpoved>
     <SetPracovnik Akce="N" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
```

```
          <Error Text=""/>
     </SetPracovnik>
</Odpoved>
```

- Editace pracovníka

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <GetPracovnik Akce="E" PracovnikId=""/>
</Dotaz>
<Odpoved>
     <GetPracovnik Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
          <SmlouvaKategorie>
               <Kategorie Nazev=""/>
          </SmlouvaKategorie>
          <Uzivatele>
               <Uzivatel PrijmeniJmeno=""/>
          </Uzivatele>
          </Editovany PracovnikId="" Pracovnik="" Kategorie="" />
     </GetPracovnik>
</Odpoved>
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetPracovnik Akce="E">
          <Pracovnik Id="" Pracovnik="" Kategorie=""/>
     </SetPracovnik>
</Dotaz>
<Odpoved>
     <SetPracovnik Akce="E" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetPracovnik>
</Odpoved>
```

- Smazání pracovníka

```
<Dotaz>
     <Uzivatel Login="" PrijmeniJmeno=""/>
     <SetPracovnik Akce="D">
          <Pracovnik Id=""/>
     </SetPracovnik>
</Dotaz>
<Odpoved>
     <SetPracovnik Akce="D" Zpracovano="A">
<!--Zpracovano="A" chybí element Error Zpracovano="N" zobrazí se jen element
Error   -->
          <Error Text=""/>
     </SetPracovnik>
</Odpoved>
```

Jelikož rozhraní pro komunikaci mezi SŘBD a webovou aplikací je velmi rozsáhlé, byla zde pro ukázku použita pouze jeho část.

4.11. Implementace Webové Aplikace

Z důvodu obsáhlosti tohoto projektu jako příklad bude použita jedna z funkcí aplikace. Jako příklad bude uvedena funkce vytváření smlouvy v aplikaci, která je zobrazena níže:

```
@extends('layouts.app')
@section('main')
    <div class="col-sm-12 ">
        <h1 class="display-3">Přidat smlouvu </h1>
<form method="post" action="{{url('smlouvy/store')}}" enctype="multipart/form-
data" class="dropzone" id="dropzone" style="display: none;" >
@csrf
</form>
<div>
@if ($errors->any())
<div class="alert alert-danger">
<ul>
@foreach ($errors->all() as $error)
<li>{{ $error }}</li>
@endforeach
</ul>
</div><br />
@endif
<form method="post" action="{{ route('smlouvy.store') }}" >
@csrf
<?php
$l=Auth::user()->username;
$n=Auth::user()->name ;
$val1 = <<<XML
<Dotaz>
<Uzivatel Login="$l" PrijmeniJmeno="$n"/>
<GetCiselniky Akce="V"/>
</Dotaz>
XML;
$pdo = DB::getPdo();
$stmt = $pdo->prepare("BEGIN EvidenceSmluv.prc_AppDotaz(:val1, :val2);END;");
$stmt->bindParam(':val1', $val1, PDO::PARAM_STR, strlen($val1));
$stmt->bindParam(':val2', $val2, PDO::PARAM_STR, 40000);
$stmt->execute();
$xml = simplexml_load_string($val2);
?>
<div class="form-row" style="margin-left: 10px ">
<div class="form-group col-sm-2">
<label for="rok">Rok:</label>
<input type="number" class="form-control"name="rok" value="{{ old('rok') }}"/>
</div>
<div class="form-group col-sm-2">
<label for="cislo">Cislo Smlouvy</label>
<input type="number" class="form-control" name="cislo" readonly />
</div>
<div class="form-group col-sm-4">
<label for="skupinautvar">Skupina:</label>
<select name="skupinautvar" class="form-control" name="skupinautvar" value="{{
old('skupinautvar') }}"  >
```

```
<option selected  hidden ></option>
@foreach($xml->xpath('//Slozky/Slozka/@Nazev') as $key  => $value)
<option value="{{ $value }}" {{ (old("skupinautvar") == $value ? "selected":"")
}}>{{ $value }}</option>
@endforeach
</select>
</div>
<div class="form-group col-sm-4">
<label for="utvar">Organizační útvar:</label>
<select disabled="true" name="utvar" class="form-control" value="{{
old('utvar') }}">
</select>
</div>
<div class="form-group col-sm-12">
<label for="nazevsmlouvy">Nazev:</label>
<input type="text" class="form-control " name="nazevsmlouvy" value="{{
old('nazevsmlouvy') }}"/>
</div>
</div>
<div class="form-row" style="margin-left: 10px ">
<div class="form-group col-sm-2">
<label for="datumplatnostismlouvyod">Datum od</label>
<input type="date" class="form-control" name="datumplatnostismlouvyod"
value="{{ old('datumplatnostismlouvyod') }}"/>
</div>
<div class="form-group col-sm-2">
<label for="datumplatnostismlouvydo">Datum do</label>
<input id="datumdo" type="date" class="form-control"
name="datumplatnostismlouvydo" value="{{ old('datumplatnostismlouvydo') }}"/>
</div>
<div class="form-group col-sm-2">
<label for="smlouvakategorie">Smlouva Kategorie</label>
<select name="smlouvakategorie" class="form-control" name="smlouvakategorie">
<option selected  hidden ></option>
@foreach ($xml->xpath('//SmlouvaKategorie/Kategorie/@Nazev') as $key  =>
$value)
<option value="{{ $value }}" {{ (old("smlouvakategorie") == $value ?
"selected":"") }}>{{ $value }}
</option>
@endforeach
</select>
</div>
<div class="form-group col-sm-2">
<label for="osobniudaje">Osobni Udaje</label>
<select name="osobniudaje" class="form-control" name="osobniudaje" >
<option selected  hidden ></option>
@foreach ($xml->xpath('//OsobniUdaje/Udaje/@Nazev') as $key  => $value)
<option value="{{ $value }}" {{ (old("osobniudaje") == $value ? "selected":"")
}}>{{ $value }}
</option>
@endforeach
</select>
</div>
<div class="form-group col-sm-2">
<label for="smlouvastav">Smlouva stav</label>
<select name="smlouvastav" class="form-control" name="smlouvastav">
<option selected  hidden ></option>
```

```
@foreach ($xml->xpath('//SmlouvaStav/Stav/@Nazev') as $key  => $value)
<option value="{{ $value }}" {{ (old("smlouvastav") == $value ? "selected":"")
}}>{{ $value }}
</option>
@endforeach
</select>
</div>
<div class="form-group col-sm-2">
<label for="uzivatel">Uživatel</label>
<input type="text" class="form-control" style="margin-left: 3px"
name="uzivatel" value="{{ Auth::user()->name }}" readonly/>
</div>
</div>
<h1 class="display-3">Informace o partnerovi</h1>
<div style="margin-left: 10%">
<table border="0" class="col-sm-4 float-left ">
<tr>
<td>IČO:</td>
<td><input class="prvek_fakturacni_ico form-control col-sm-6" type="text"
name="fakturacni_ico" id="fakturacni_ico" value="{{ old('fakturacni_ico')
}}"/></td>
</tr>
<tr>
<td>Údaje z ARES:</td>
<td>
<span class="tl_ares form-control col-sm-6">Zjistit údaje z ARES</span>
<span id="ares_okno"></span>
<div class="ajax_loader_ares"><div class="ajax_loader_ares_in"></div></div>
</td>
</tr>
<tr>
<td>Firma:</td>
<td><input class="prvek_fakturacni_firma form-control col-sm-6" type="text"
name="fakturacni_firma" value="{{ old('fakturacni_firma') }}"/></td>
</tr>
<tr>
<td>DIČ:</td>
<td><input class="prvek_fakturacni_dic form-control col-sm-6" type="text"
name="fakturacni_dic" value="{{ old('fakturacni_dic') }}"/></td>
</tr>
<tr>
<td>Právní forma:</td>
<td><input class="prvek_fakturacni_f form-control col-sm-6" type="text"
name="ares_pravni_f" value="{{ old('ares_pravni_f') }}"/></td>
</tr>
</table>
<table border="0" class="col-sm-4 float-left ">
<tr>
<td>RČ:</td>
<td><input class="rc form-control col-sm-6" type="text" name="rc" id="rc"
value="{{ old('rc') }}"/></td>
</tr>
<tr>
<td>Jmeno:</td>
<td><input class="jmeno form-control col-sm-6" type="text" name="jmeno"
value="{{ old('jmeno') }}" /></td>
</tr>
```

```
<tr>
<td>Příjmení: </td>
<td><input class="prijmeni form-control col-sm-6" type="text" name="prijmeni"
value="{{ old('prijmeni') }}"/></td>
</tr>
</table>
<table border="0" class="col-sm-4 float-left">
<tr>
<td>Adresa: </td>
<td><input class="prvek_fakturacni_adresa form-control col-sm-6" type="text"
name="fakturacni_adresa" value="{{ old('fakturacni_adresa') }}"/></td>
</tr>
<tr>
<td>Město: </td>
<td><input class="prvek_fakturacni_mesto form-control col-sm-6" type="text"
name="fakturacni_mesto" value="{{ old('fakturacni_mesto') }}"/></td>
</tr>
<tr>
<td>PSČ: </td>
<td><input class="prvek_fakturacni_psc form-control col-sm-6" type="text"
name="fakturacni_psc" value="{{ old('fakturacni_psc') }}"/></td>
</tr>
</table>
</div>
<div>
<div class="form-group" >
<label for="kategoriepartnera" >Kategorie Partnera:</label>
<select name="kategoriepartnera"  class="form-control" name="smlouvakategorie">
<option selected  hidden ></option>
@foreach ($xml->xpath('//PartnerKategorie/Kategorie/@Nazev') as $key  =>$value)
<option value="{{ $value }}" {{ (old("kategoriepartnera") == $value ?
"selected":"") }}>{{ $value }}</option>
@endforeach
</select>
</div>
<div class="form-group">
<label for="DruhPartnera">Nazev:</label>
<select name="DruhPartnera" class="form-control" name="DruhPartnera">
<option selected  hidden ></option
@foreach ($xml->xpath('//PartnerDruh/Druh/@Nazev') as $key  => $value)
<option value="{{ $value }}" {{ (old("DruhPartnera") == $value ? "selected":"")
}}>{{ $value }}</option>
@endforeach
</select>
</div>
</div>
<input type="hidden" id="smlouvy" name="smlouvy" value="" />
<button type="submit" class="btn btn-primary"style="float:left">Přidat</button>
</form>
<button class="btn btn-primary"style="float: left;margin-left: 10px;">Přidat
soubor</button>
</div>
</div>
@endsection
```

Tato část kódu je takzvaný „View“ pro stránku vytváření smlouvy, pomocí tohoto kódu jsou načítány základní číselníky z rozhraní SŘBD a je uživateli zobrazovaná webová stránka vytváření smlouvy, která vypadá následovně:

Obrázek 3 - Vytváření smlouvy (Zdroj: Vlastní)

Tato část kódu ale neobsahuje žádnou logiku zpracování vstupních dat, která byla zadána uživatelem. Po stisknutí tlačítka „Přidat“ jsou data pomoci POST požadavku předána do backendu aplikace, kde následně probíhá zpracování. Aby data byla nasměrována do správné části, byl použit následující „Route“ :

```
Route::post('smlouvy/store', 'SmlouvyController@store');
```

Následně jsou data zpracovávána v backendové části aplikace a předána do rozhraní SŘBD, příklad zpracování je vyobrazen níže:

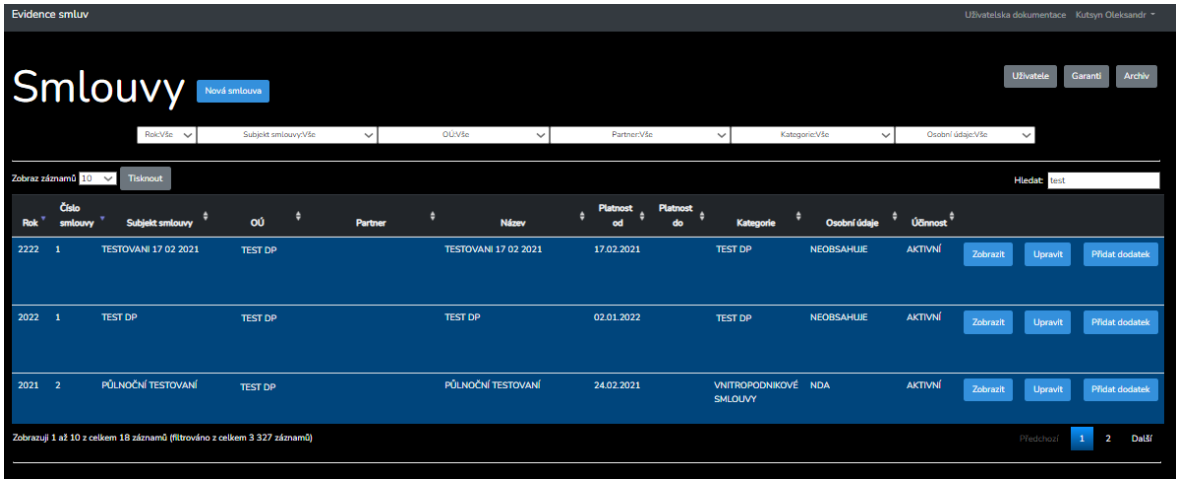
```
    public function store(Request $request)
    {
        $request->validate([

            'rok'=>'required',
            'nazevsmlouvy'=>'required',
            'skupinautvar'=>'required',
            'utvar'=>'required',
            'datumplatnostismlouvyod'=>'required',
            'smlouvakategorie'=>'required',
            'osobniudaje'=>'required',
            'smlouvastav'=>'required',
            'kategoriepartnera'=>'required',
            'DruhPartnera'=>'required',
            'smlouvy'=>'required'
        ]);
```

```
$rok=['rok'=> $request->get('rok')]['rok'];
$nazev=['nazevsmlouvy'=> $request->get('nazevsmlouvy')]['nazevsmlouvy'];
$skupina=['skupinautvar'=> $request->get('skupinautvar')]['skupinautvar'];
$utvar=['utvar'=> $request->get( 'utvar')]['utvar'];
$platnostOD=['datumplatnostismlouvyod'=> $request->get(
'datumplatnostismlouvyod')]['datumplatnostismlouvyod'];
$platnostODk = date("d.m.Y", strtotime($platnostOD));
$platnostDO=['datumplatnostismlouvydo'=> $request-
>get('datumplatnostismlouvydo')]['datumplatnostismlouvydo'];
$platnostDOk = date("d.m.Y", strtotime($platnostDO));
$kategorie=[ 'smlouvakategorie'=> $request-
>get('smlouvakategorie')]['smlouvakategorie'];
$osud=['osobniudaje'=> $request->get( 'osobniudaje')]['osobniudaje'];
$smlouvastav=['smlouvastav'=> $request->get('smlouvastav')]['smlouvastav'];
$ico=['fakturacni_ico'=> $request->get('fakturacni_ico')]['fakturacni_ico'];
$firma= ['fakturacni_firma'=> $request-
>get('fakturacni_firma')]['fakturacni_firma'];
$adresa= ['fakturacni_adresa'=> $request-
>get('fakturacni_adresa')]['fakturacni_adresa'];
$mesto=['fakturacni_mesto'=> $request-
>get('fakturacni_mesto')]['fakturacni_mesto'];
$psc=['fakturacni_psc'=> $request->get('fakturacni_psc')]['fakturacni_psc'];
$dic=['fakturacni_dic'=> $request->get('fakturacni_dic')]['fakturacni_dic'];
$DruhPartnera=['DruhPartnera'=> $request->get('DruhPartnera')]['DruhPartnera'];
$kategoriepartnera=['kategoriepartnera'=> $request-
>get('kategoriepartnera')]['kategoriepartnera'];
$pravnif=['ares_pravni_f'=> $request->get('ares_pravni_f')]['ares_pravni_f'];
$rc=['rc'=> $request->get('rc')]['rc'];
$jmeno=['jmeno'=> $request->get('jmeno')]['jmeno'];
$prijmeni=['prijmeni'=> $request->get('prijmeni')]['prijmeni'];
       $file=['smlouvy'=>$request->get('smlouvy')]['smlouvy'];
       $jfile=json_decode($file);
        $l=Auth::user()->username;
        $n=Auth::user()->name ;
        $auth = array(
            "login" => config('wsdlauth.wsdllogin'),
            "password" => config('wsdlauth.wsdlpassword')
        );
        $fileId="";
        $title= $rok.$nazev.$skupina.$utvar;
        $extension="pdf";
        $contentType="application/pdf";
        $search_query = new StdClass();
        $search_query->array =[];
        foreach ($jfile as $file)
        {
            $file = "images/$file";
            $handle = fopen($file,'rb');
            $file_content = fread($handle,filesize($file));
            fclose($handle);
            $fileDef = new StdClass();
            $fileDef->fileId =$fileId;
            $fileDef->title = $title;
            $fileDef->extension = $extension;
            $fileDef->contentType = $contentType;
            $fileDef->data = $file_content;
            $search_query->array[]=($fileDef);
```

```
        }
        $numberContract="16003";
        $titleContract=$nazev;
        $subjectContrac="Předmět";
        $partnerName="Jmeno partnera";
        $categoryContract="Kategorie";
        $createUserDT = date("Y-m-d H:i:s");
        $createBy="Kutsyn Oleksandr";
        $client = new soapclient(config('wsdlauth.wsdl'),$auth);
        $response = $client->contractCreateOrUpdate(array(
            "id",
            "numberContract"=>$numberContract,
            "titleContract"=>$titleContract,
            "subjectContract"=>$skupina,
            "partnerName"=>$kategoriepartnera,
            "categoryContract"=>$kategorie,
            "createUserDT"=>$createUserDT,
            "createBy"=>$createBy,
            "files"=>$search_query,
        ));
$dokument="";
$id_spisu=$response->id;
if (is_array($response->files->array))
{
    foreach ($response->files->array as $file)
    {
        $dokument.= '<Doklad Spisovka="' . $file->fileId . '"/>';
    }
}
    else
    {
        foreach ((array)$response->files->array->fileId as $file)
        {
            $dokument.= '<Doklad Spisovka="' . $file . '"/>';
        }
    }
        $smlouva =  <<<XML
<Dotaz>
     <Uzivatel Login="$l" PrijmeniJmeno="$n"/>
     <SetSmlouva Akce="N">
          <Smlouva Rok="$rok" Nazev="$nazev" Skupina="$skupina" Utvar="$utvar"
PlatnostOd="$platnostODk" PlatnostDo="$platnostDOk" Kategorie="$kategorie"
OsobniUdaje="$osud" Stav="$smlouvastav"
          IdSpisovka="$id_spisu" />
        <Partner Jmeno="$jmeno" Prijmeni="$prijmeni" Rc="$rc" Firma="$firma"
FirmaForma="$pravnif" Ic="$ico" Dic="$dic" Ulice="$adresa" Psc="$psc"
Obec="$mesto" Stat="" Druh="$DruhPartnera" Kategorie="$kategoriepartnera"/>
      <Doklady>
              $dokument
      </Doklady>
     </SetSmlouva>
</Dotaz>
XML;
        $pdo = DB::getPdo();
        $stmt = $pdo->prepare("BEGIN EvidenceSmluv.prc_AppDotaz(:val1,
:val2);END;");
        $stmt->bindParam(':val1', $smlouva, $pdo::PARAM_STR, strlen($smlouva));
```

```
        $stmt->bindParam(':val2', $val2, $pdo::PARAM_STR, 32768);
        $stmt->execute();
        $file = new Filesystem;
        $file->cleanDirectory('images/');
        return redirect('/smlouvy')->with('Uspěch', 'Smlouva uložena!');
    }
```

Po provedení této části kódu byl soubor uložen do spisové služby a data uložena v SŘBD pomocí rozhraní, následně do logu aplikace byl zapsán řádek „Úspěch, Smlouva uložena“, uživatel je přesměrován na hlavní stránku aplikace, kde může vidět seznam smluv. Stránka se seznamem smluv vypadá následovně (některé položky na obrázku byly upraveny z důvodu obsahu osobních údajů):



Obrázek 4 – Seznam smluv (Zdroj: Vlastní)

5 Výsledky a diskuse

Tématem této závěrečné práce je vývoj webové aplikace pro evidenci smluv, úspěšnost této práce závisela na mnoha faktorech, jako jsou například: správný výběr materiálu k nastudování, správné pochopení problematiky nebo správná interpretace požadavku zadavatelské společnosti.

Jelikož se nejedná o aplikaci, která slouží pouze jako prototyp pro tuto závěrečnou práci, ale jedná se o aplikaci, která je reálně používaná, některé úlohy byly složitější, než se očekávalo, ale veškeré požadavky od zadavatele byly splněny a aplikace byla úspěšně předaná do pilotního provozu.

Dle zpětné vazby od zadavatele aplikace je zadavatel s aplikaci maximálně spokojen, a také, že aplikace naplnila jeho očekávání. Dalším pozitivním faktorem je informace od společnosti zadavatele, že aplikace bude v budoucnu rozšířena o evidenci právních stanovisek a společnost zadavatele se také přiklání k variantě interního vývoje. Což pro mě jakožto pro vývojáře je jedním z faktorů signalizujících úspěšnost projektu jako celku.

Jedním z negativních faktorů byla nedostatečná podrobnost zadání, tato skutečnost měla za následek několik fází změn v projektu, které byly odhaleny během testovacího provozu a následně upravovány.

Práce na tomto projektu byla pro zhotovitele přínosnou, jelikož díky praktické implementaci projektu a jeho následnému nasazení získal zkušenosti, i přestože vývoj webové aplikace tohoto typu není ničím novým.

6 Závěr

Závěrečná práce byla zaměřena na téma vývoje webové aplikace pro evidenci smluv. Jako výsledek realizace této diplomové práce byla vytvořena plně funkční webová aplikace splňující veškeré požadavky zadavatelské společnosti, byla provedena výchozí analýza pro tvorbu zadání a byla zohledněna již existující podobná řešení.

Hlavní prioritou při vývoji bylo vytvořit aplikaci s co nejpřesněji splněnými požadavky a aby byla uživatelsky přívětivá. Z toho důvodu bylo učiněno rozhodnutí implementovat aplikaci jako webovou aplikaci pomocí frameworku Laravel.
Architektura aplikace byla navržena na základě vzoru MVC, databáze byla vytvořena v ORACLE SŘBD. V důsledku toho byla vyvinuta frontedová část webové aplikace, která odpovídá za vizuální uspořádání aplikace, a také backendová část odpovědná za zpracování požadavků a implementaci funkčnosti.

Byly implementovány veškeré funkce požadované zadavatelem, jako například jsou: správa uživatele, vytvoření smlouvy, uložení souboru do spisové služby. Veškeré informace jsou uloženy v databázi ORACLE.

V závěrečné fázi vývoje byl proveden test provozuschopnosti a správnost vyřizování chyb souvisejících s prací se smlouvami a oprávněními uživatele. Byla provedena kontrola správnosti zpracování a uložení dat smluv a následné zobrazení v seznamu. Výsledkem bylo, že webová aplikace je snadno škálovatelná a otevřená pro další rozvoj. Tyto výhody vyplynuly z výběru architektury MVC a použití frameworku Laravel.

Zkušenosti získané v oblasti vývoje mě obohatily o spoustu nových poznatků v oboru, ve kterém se nadále plánuji rozvíjet a implementovat tyto poznatky v praxi.

7 Seznam použitých zdrojů

1. **Mansoor, Halah.** Design and Implementation Dynamic Website for Electronic Library. *IARJSET*. 2017, Sv. 4, 2, stránky 133–138.

2. **Dissanayake, Nalaka Ruwan.** Web-based Applications: Extending the General Perspective of the Service of Web. *RiWAArch style: An abstract hybrid style for Rich Web-based Applications.* 2017.

3. **Brada, Přemysl.** *Webové aplikace.* Plzeň : ZČU.

4. **Koďousková, Barbora.** Co jsou progresivní webové aplikace (PWA) a jaké mají výhody. *Rascasone.* [Online] 18. 10. 2021. [Citace: 10. 12. 2021.] https://www.rascasone.com/cs/blog/progresivni-webova-aplikace-vyhody.

5. **Koďousková, Barbora.** Co je webová a desktopová aplikace a jaký je mezi nimi rozdíl? . *Rascasone.* [Online] 17. 11. 2021. [Citace: 25. 12. 2021.] https://www.rascasone.com/cs/blog/desktop-web-aplikace.

6. **Coyier, Chris.** Creating a Web App from Scratch – Part 4 of 8: HTML & CSS . *CSS-Tricks.* [Online] 9. 4. 2019. [Citace: 28. 12. 2021.] https://css-tricks.com/app-from-scratch-4-html-css/.

7. **Duckett, Jon.** *HTML & CSS. Design and Build Websites.* Indianapolis : John Wiley & Sons, Inc., 2011. 978-1-118-00818-8.

8. **Beneš, Miroslav.** *Skriptovací jazyky.* Ostrava : FEI VŠB-TUO.

9. **Ousterhout, John K.** Scripting: Higher Level Programming for the 21st Century. *IEEE Computer magazine.* 1998.

10. **Riehle, Dirk.** *Framework Design: A Role Modeling Approach.* Zürich : Swiss Federal Institute of Technology, 2000.

11. Zend Framework 3. *Zend Framework.* [Online] 2021. [Citace: 19. 12. 2021.] https://framework.zend.com/learn.

12. **Thapliyal, Vimal.** Difference Between Frontend and Backend MVC . *joomlatuts.net.* 2016.

13. **Pirsa, Andrea.** Front-end solution for enhancing web sites accessibility. *13th International Conference on Telecommunications (ConTEL).* 2015.

14. Nejoblíbenější front-end frameworky. *JobsContact.* [Online] 2021. [Citace: 11. 12. 2021.] https://www.jobscontact.cz/blog/nejoblibenejsi-minus-front-minus-minus-minus-end-minus-frameworky.

15. Backend Development. *George Mason University.* [Online] 2016. [Citace: 10. 12. 2021.] https://cs.gmu.edu/~tlatoza/teaching/swe432f16/Lecture%2013%20-%20Backend%20Development.pdf.

16. **Šafařík, Daniel.** Nejpoužívanější programovací jazyky pro backend webu. *ZonerCloud.* [Online] 29. 9. 2021. [Citace: 12. 12. 2021.] https://www.zonercloud.cz/magazin/nejpouzivanejsi-programovaci-jazyky-pro-backend-webu.

17. PHP Manual: General Information. *PHP.* [Online] 2021. [Citace: 20. 12. 2021.] https://www.php.net/manual/en/faq.general.php.

18. **Kosek, Jiří.** PHP - nejen dynamicky generované stránky. *Kosek.* [Online] [Citace: 20. 12. 2021.] https://www.kosek.cz/php/php.html.

19. Laravel Documentation. *Laravel.* [Online] 2021. [Citace: 21. 12. 2021.] https://laravel.com/docs/8.x.

20. Co je Laravel? *Laravel Blog.* [Online] 2021. [Citace: 21. 12. 2021.] https://laravelblog.cz/co-je-laravel.

21. **Klimčík, Jaroslav.** Laravel Forge. *Laravel Blog.* [Online] 7. 2. 2021. [Citace: 21. 12. 2021.] https://laravelblog.cz/clanek/laravel-forge.

22. **Barnes, Eric L.** Laravel Tutorial: Step by Step Guide to Building Your First Laravel Application. *Laravel News.* [Online] 5. 3. 2021. [Citace: 29. 12. 2021.] https://laravel-news.com/your-first-laravel-application.

23. **Lupčík, Jan.** Úvod do Laravel frameworku pro PHP. *ITNetwork.* [Online] 4. 11. 2019. [Citace: 19. 12. 2021.] https://www.itnetwork.cz/php/laravel/uvod-do-laravel-frameworku-pro-php.

24. **Sirkin, Jessica.** SQL (Structured Query Language) . *TechTarget.* [Online] 2020. [Citace: 20. 12. 2021.] https://searchdatamanagement.techtarget.com/definition/SQL.

25. **Hordějčuk, Vojtěch.** Jazyk SQL. *VOHO.* [Online] 2021. [Citace: 28. 12. 2021.] http://voho.eu/wiki/sql/.

26. ACID Properties in SQL Server. *Tutorial Gateway.* [Online] 2021. [Citace: 28. 12. 2021.] https://www.tutorialgateway.org/acid-properties-in-sql-server/.

27. Build fast, responsive sites with Bootstrap. *Bootstrap.* [Online] 2021. [Citace: 20. 12. 2021.] https://getbootstrap.com/.

28. Introduction. *Bootstrap.* [Online] 2021. [Citace: 29. 12. 2021.] https://getbootstrap.com/docs/4.4/getting-started/introduction/.

29. Browsers and devices. *Bootstrap.* [Online] 2021. [Citace: 29. 12. 2021.] https://getbootstrap.com/docs/4.0/getting-started/browsers-devices/.

30. **Kutáč, Pavel.** Vlastní filtry s DataTables a AJAXem. *Kutáč.* [Online] 9. 2. 2021. [Citace: 19. 12. 2021.] https://www.kutac.cz/weby-a-vse-okolo/vlastni-filtry-s-datatables-a-ajaxem.

31. DataTables. *DataTables.* [Online] 2021. [Citace: 18. 12. 2021.] https://datatables.net/manual/data/.

32. Základy PL/SQL. *Katedra informatiky a výpočetní techniky ZČU.* [Online] 2021. [Citace: 20. 12. 2021.] https://www.kiv.zcu.cz/studies/predmety/db2/cviceni-plsql-zaklady.html.

33. Database PL/SQL Language Reference. *ORACLE.* [Online] 2021. [Citace: 20. 12. 2021.] https://docs.oracle.com/cd/B28359_01/appdev.111/b28370/toc.htm.

34. **Haverbeke, Marijn.** *Eloquent Javascript, 3rd Edition: A Modern Introduction to Programming.* San Francisco : No Starch Press, 2018. 978-1593279509.

35. ECMAScript. *TC39.* [Online] 17. 12. 2021. [Citace: 21. 12. 2021.] https://tc39.es/ecma262/#sec-overview.

36. **Resig, John.** *Pro JavaScript Techniques.* New York : Apress, 2015. 9781430263920.

37. **Veverka, Radek.** AJAX v JavaScriptu - Základní dotazy. *ITnetwork.* [Online] 2020. [Citace: 10. 12. 2021.] https://www.itnetwork.cz/javascript/oop/ajax-v-javascriptu-zakladni-dotazy.

38. How to Develop a Web Application with Ajax. *Webucator.* [Online] 2021. [Citace: 25. 12. 2021.] https://www.webucator.com/article/how-to-develop-a-web-application-with-ajax/.

39. **Pirnau, Mironela.** The SOAP Protocol Used for Building and Testing Web Services. *Proceedings of the World Congress on Engineering.* 2011, Sv. 1.

40. **Antani, Ved.** *Mastering JavaScript: Explore and master modern JavaScript techniques in order to build large-scale web applications.* Birmingham : Packt Publishing, 2016. 978-1-78528-134-1.

41. **Zou, Yunxiao.** Virtual DOM coverage for effective testing of dynamic web applications. *the 2014 International Symposium.* 2014.

42. **Sheth, Kishan.** What is Virtual DOM? How Virtual DOM works ? What is Reconciliation ? What is diffing algorithm? What makes React so fast ? . *DEV.* [Online] 27. 4. 2021. [Citace: 30. 12. 2021.] https://dev.to/koolkishan/what-is-virtual-dom-how-virtual-dom-works-what-is-reconciliation-what-is-diffing-algorithm-what-makes-react-so-fast-327a.

43. **Koďousková, Barbora.** Proč k vývoji single-page aplikace zvolit knihovnu React? *Rascasone.* [Online] 12. 12. 2021. [Citace: 30. 12. 2021.] https://www.rascasone.com/cs/blog/jednostrankova-web-aplikace-spa-react.

44. **Kosek, Jiří.** Inteligentní podpora navigace na WWW s využitím XML. *Kosek.cz.* [Online] 2021. [Citace: 8. 12. 2021.] https://www.kosek.cz/diplomka/html/index.html.

45. **Chinnici, Roberto.** Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. *W3C.* [Online] 26. 6. 2007. [Citace: 9. 12. 2021.] https://www.w3.org/TR/wsdl/.

46. **Modeler, Rational Software.** Odkaz na jazyk WSDL (Web Services Description Language). *IBM.* [Online] 2021. [Citace: 25. 12. 2021.] https://www.ibm.com/docs/cs/rsm/7.5.0?topic=wsdl-web-services-description-language-reference.

47. **Lindstrom, Lowell.** *Extreme Programming and Agile Methods.* Berlin : Springer, 2004. 3-540-22839-X.

48. **Štráfelda, Jan.** Pravidlo KISS. [Online] [Citace: 30. 12. 2021.] https://www.strafelda.cz/kiss.

49. **Jonáš, Martin.** Návrhové principy: DRY. *Zdroják.cz.* [Online] 30. 5. 2012. [Citace: 22. 12. 2021.] https://zdrojak.cz/clanky/navrhove-principy-dry/.

50. **Koďousková, Barbora.** Contract Management System aneb software pro evidenci smluv. *Rascasone.* [Online] 13. 4. 2021. [Citace: 6. 12. 2021.] https://www.rascasone.com/cs/blog/contract-management-system-sprava-smluv.

8 Seznam obrázků