Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Systém pro správu dat zobrazovaných online a v mobilní aplikaci

Bc. Robert Michálek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Robert Michálek

Systémové inženýrství a informatika
Informatika

Název práce

Systém pro správu dat zobrazovaných online a v mobilní aplikaci

Název anglicky

System for managing data displayed online and in mobile application

---

Cíle práce

Cílem této diplomové práce je navrhnout, realizovat a nasadit systém pro správu dat z oblasti cestovního ruchu uložených v lokální databázi, vytvoření webové prezentace a mobilní aplikace pro zobrazování těchto dat uživatelům.

Metodika

Práce sestává ze dvou hlavních částí – přehledu teoretických východisek a praktické části.

Metodika zpracování teoretické části je založena na studiu odborných informačních zdrojů. Na základě syntézy zjištěných poznatků bude popsána problematika vývoje aplikací se zaměřením na jazyk C#, vývoje webových stránek, vývoje mobilních aplikací a napojení na databázové systémy.

V praktické části práce je provedena analýza a implementace backendové administrační aplikace a k ní náležející webové prezentace a mobilní aplikace. Pro implementaci administrační a mobilní aplikace bude použit jazyk C#, webová prezentace bude vytvořena s využitím jazyka PHP. Při návrhu a implementaci bude využito standardních metod a nástrojů softwarového inženýrství. Výsledné aplikace budou nasazeny, otestovány a na základě poznatků z nasazení a testů budou formulovány možnosti dalšího případného budoucího rozvoje.

Doporučený rozsah práce

60-80 stran

Klíčová slova

C#, .NET, MySQL, SQL, Android, Xamarin, PHP, jQuery, HTML, SEO, CSS, design

---

Doporučené zdroje informací

GILMORE, W J. Velká kniha PHP a MySQL 5 : kompendium znalostí. Brno: Zoner Press, 2006. ISBN 80-86815-53-6.

HERMES, Dan. Xamarin Mobile Application Development. Berlin: Springer. 2015. ISBN 9781484202159.

LACKO, Ľuboslav a Martin HERODEK. Vývoj aplikací pro Android. 1. vyd. Brno: Computer Press, 2015. ISBN 9788025143476.

ROBINSON, S. Kompletní průvodce programováním aplikací pro .NET platformu v jazyce C#. Brno: Computer Press, a.s., 2003. První vydání. ISBN 80-251-0085-5.

SCHNEIDER, R.D. MySQL, Oficiální průvodce tvorbou, správou a laděním databází. Praha: Grada Publishing, a.s., 2006. První vydání. ISBN 80-247-1516-3.

ŠÍMA, F. Microsoft Visual Studio .NET. Praha: Grada Publishing, a.s., 2006. První vydání. ISBN 80-247-1418-3.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 19. 2. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 19. 2. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 06. 04. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Systém pro správu dat zobrazovaných online a v mobilní aplikaci“ jsem vypracoval 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 uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 22. 3. 2020 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce panu Ing. Jiří Brožkovi, Ph.D., za ochotnou pomoc a odborné vedení při zpracování práce. Dále děkuji své ženě za podporu a pomoc, kterou se mnou v těžkých chvílích sdílela.

Systém pro správu dat zobrazovaných online a v mobilní aplikaci

Souhrn

Tato práce se zabývá problematikou vývoje aplikací ze dvou pohledů: back-end pro zaměstnance, kteří budou spravovat data, a front-end pro klienty, kteří budou data prohlížet online na webové stránce a mobilní aplikaci. Jsou zde popsány technologie pro tvorbu aplikací na platformě .NET Framework, využívání MySQL databáze, technologie pro tvorbu dynamických webových stránek s napojením na databázi a vývoj mobilních aplikací s nástrojem Xamarin. Dále je uveden návrh a implementace se všemi použitými technologiemi (C#, .NET, SQL, Xamarin, PHP, HTML, CSS, jQuery), způsob komunikace mezi aplikacemi a databází, návrh designu včetně UI specifikace pro back-end aplikaci a následně principy SEO optimalizace pro webové stránky a responzivní verze.

Klíčová slova: C#, .NET, MySQL, SQL, Android, Xamarin, PHP, jQuery, HTML, SEO, CSS, design

System for managing data displayed online and in mobile application

Summary

This thesis deals with application development from two perspectives: back-end for employees who will manage data; and front-end for clients who view data online on a website and mobile app. There are described technologies for creating applications on .NET Framework platform, using MySQL database, technologies for creating dynamic web pages with connection to database and development of mobile applications with Xamarin. It also includes design and implementation with all used technologies (C#, .NET, SQL, Xamarin, PHP, HTML, CSS, jQuery), communication between applications and database, design including UI specification for back-end application and subsequently SEO principles optimization for websites and responsive versions.

Keywords: C#, .NET, MySQL, SQL, Android, Xamarin, PHP, jQuery, HTML, SEO, CSS, design

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

S každodenním exponenciálním nárůstem nových technologií a platností Moorova zákona o výpočetním výkonu obvodů a jim odpovídajících uživatelských požadavků na informační systémy se v dnešní uspěchané době nesmí zapomínat i na vývoj řešení a aplikací, které by z druhé strany pomohly uspokojit potřeby zákazníků a zároveň chránit životní prostředí. Každá agentura rozdává svým klientům na dovolených či zájezdech často zbytečné tištěné materiály, ve kterých najdou umístění směnáren, poboček a dalších informací. Právě díky absenci novodobých systémů a finanční náročnosti na vývoj končí papírové materiály nevyužité v koši. Tyto podniky již mají své systémy pro evidenci destinací, resortů a pokojů, ale chybí zde rozvoj uživatelských technologií, které by klientům přinesly využití něčeho, co mají neustále u sebe – mobilní telefon či chytré hodinky.

Práce se zaměřuje na vývoj systému, který by většinu papírových administračních letáků a materiálů omezil a integroval informace do chytrých přenosných zařízení, která má v dnešní době každý cestoval u sebe. Práce pojednává o vytvoření jak administrační části pro správu dat firmou, tak prezentační části – webové responzivní stránce a mobilní aplikaci.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je vytvořit funkční informační systém, jehož data budou spravována zaměstnanci dané firmy a který bude poskytovat klientům (zákazníkům cestovní kanceláře) lokální informace během jejich pobytu v dané destinaci – např. zájmová místa, jako jsou restaurace, lékárny, nemocnice, směnárny, pošty či pobočky cestovní kanceláře.

Pro správu dat je pro zaměstnance vytvořena back-end aplikace, která bude nainstalována pouze na lokálních počítačích dané pobočky. Všechna data jsou uložena v centrální zabezpečené databázi. Pro klienty jsou k dispozici webové stránky včetně responzivní verze pro chytrá mobilní zařízení, ale také přímo mobilní aplikace, která může být stažená například z Google Play či Apple Store. Webová prezentace splňuje náležitosti SEO, je optimalizovaná pro plnou funkčnost na mobilních zařízeních a vyhovuje normám přístupnosti. Mobilní aplikace získává data z databáze přes vytvořené API rozhraní a je kompatibilní na hlavních operačních systémech pro mobilní zařízení – Android a Apple iOS.

2.2 Metodika

Práce sestává ze dvou hlavních částí – přehledu teoretických východisek a praktické části.

Metodika zpracování teoretické části je založena na studiu odborných informačních zdrojů. Na základě syntézy zjištěných poznatků bude popsána problematika vývoje aplikací se zaměřením na jazyk C#, vývoje webových stránek, vývoje mobilních aplikací a napojení na databázové systémy.

V praktické části práce je provedena analýza a implementace back-endové administrační aplikace a k ní náležící webové prezentace a mobilní aplikace. Pro implementaci administrační a mobilní aplikace byl použit jazyk C#, webová prezentace byla vytvořena s využitím jazyka PHP. Při návrhu a implementaci bylo využito standardních metod a nástrojů softwarového inženýrství. Výsledné aplikace byly nasazeny, otestovány a na základě poznatků z nasazení a testů budou formulovány možnosti dalšího případného budoucího rozvoje.

3 Teoretická východiska

3.1 Programovací jazyky

3.1.1 C#

C# (C-Sharp, česky vyslovovaný jako sí-šarp) je programovací jazyk poprvé zveřejněný v roce 2000. Vývojářský tým byl veden předním dánským softwarovým inženýrem Andersnem Hejlsbergem. Původní název měl být COOL (C-like Object Oriented Language), ale před zveřejněním na konferenci profesionálních vývojářů společnosti Microsoft v červenci 2000 byl změněn na C# (Strauss, 2016).

C# je objektově orientovaný programovací jazyk, který splňuje následující předpisy objektově orientovaného programování:

- Dědičnost – máme-li třídu X, můžeme vytvořit novou třídu Y, která bude z třídy X dědit všechny vlastnosti a funkce, a navíc bude mít své vlastní funkce. Třída, která dědí, se nazývá derivovaná třída.
- Abstrakce – chceme-li vytvořit základní funkčnost, kterou budou mít všechny derivované třídy k dispozici, vytvoří se její elementární abstrakce v rodičovské třídě. Na rozdíl od dědičnosti se musí v derivované třídě implementovat metody, které chceme použít z abstraktní třídy.
- Zapouzdření – zjednodušeně se jedná o skrývání vnitřní funkcionality třídy, která není potřeba pro implementaci třídy. Třída je tedy jako black box – víme, že třída bude konzistentní svými funkcemi (včetně parametrů), a pakliže dostane správný vstup, nezajímá nás, jak daná třída dojde k danému správnému výstupu.
- Polymorfismus – umožňuje objektům volání jedné metody se stejným jménem, ale s jinou implementací (s různými parametry). Uvnitř funkce jsou na základě získaných parametrů volány různé funkce (Strauss, 2016).

C# je jazyk s virtuálním strojem, což je dle ITnetwork (2018) nejvyužívanější podoba jazyka pro vývoj většiny aplikací. Zdrojový kód právě spuštěné aplikace je nejprve přeložen do tzv. mezikódu známého jako CIL (Common Intermediate Language), který funguje jako ve

strojovém kódu (obsahuje jednodušší instrukční sadu), a následně je rychleji interpretovatelný virtuálním strojem pro různé typy procesorů (Purdum, 2007).

3.1.2 PHP

PHP zastává v překladu hypertextový preprocesor (původně Personal Home Page, což bylo rekurzivně změněno, ale zkratka prvních písmen zůstala). Tento open-source skriptovací serverový jazyk byl vytvořen v roce 1994 Rasmusem Lerdorfem a dalšími talentovanými lidmi, kteří jej z velké části přepsali (Welling a kol., 2017). Poslední vydaná verze jazyka PHP je dle PHP.net (2020) verze 7.4.3 ze dne 20. února 2020.

Jedná se o skriptovací jazyk, který je přímo podporován značkovacím jazykem HTML a je-li jazyk PHP nasazen na webovém serveru, kód PHP může být vkládán přímo do HTML kódu webové stránky – jeho zpracování je interpretováno na webovém serveru, který následně vygeneruje výsledný HTML dokument (Welling a kol., 2017).

PHP v současné době podporuje a rozšiřuje mnoho knihoven (např. pro práci se soubory, zpracování grafiky a textů), internetové protokoly (FTP, HTTP apod.) a je přímo využitelný většinou novodobých databázových systému, jako je MySQL, MSSQL či Oracle. Tím, že se využívá hlavně u dynamických aplikací a webových stránek, kdy je zpracován na straně serveru a následně webový server navrátí výsledek, se dle Gilmora (2007) jedná o nejrozšířenější skriptovací jazyk pro webové stránky. Podle W3Techs (2020) je PHP využito v 78,8 % všech webových stránek.

3.1.3 SQL

Structured Query Language, ve zkratce SQL, je standardizovaný strukturovaný dotazovací jazyk, který je používán pro práci s databázemi. Pomocí SQL lze databáze vytvářet, upravovat a přidávat, spravovat a získávat z nich data. Dle Taylora (2013) je tento jazyk ze 70. let 20. století nejrozšířenějším jazykem pro práci s databázemi.

Reálně se jazyk SQL skládá z následujících podjazyků:

- DQL (Data Query Language) – jazyk pro vytváření dotazů,

- DDL (Data Definition Language) – jazyk pro definování datových struktur (databázových schémat),
- DCL (Data Control Language) – jazyk pro kontrolování přístupů k datům,
- DML (Data Manipulation Language) – pro přidávání, odebírání a upravování dat (Techopedia, 2018).

V nejnižší formě obsahuje SQL pouze pár příkazů pro práci s daty:

- select – pro výběr dat,
- insert – pro vložení dat,
- update – pro aktualizaci dat,
- delete – pro mazání dat (Techopedia, 2018).

3.2 Framework

Softwarový framework je koncepční platforma, kde mohou vývojáři nebo uživatelé selektivně upravovat nebo přepisovat společný kód. Frameworky mají podobu knihoven, kde je dobře definované rozhraní aplikačního programu (API) znovu použitelné kdekoli v rámci vyvíjeného softwaru (Techopedia, 2018).

Funkce odlišující frameworky od jiných knihovních forem jsou:

- výchozí chování: Před přizpůsobením frameworku se chová způsobem specifickým pro akci uživatele;
- inverze kontroly: Na rozdíl od jiných knihoven, globální tok ovládání je využíván samotným frameworkem než volajícím;
- rozšiřitelnost: Uživatel může framework rozšířit selektivním nahrazením výchozího kódu uživatelským kódem;
- nemodifikovatelný kód rámce: Uživatel může rámec rozšířit, ale nemůže jej upravit.

Účelem softwarového frameworku je tedy zjednodušit vývojové prostředí, což vývojářům umožňuje věnovat své úsilí více požadavkům projektu, než se zabývat běžnými, opakujícími se funkcemi a knihovnami tohoto frameworku (Techopedia, 2018).

3.2.1 .NET framework

.NET je označení pro softwarový framework od firmy Microsoft, který je součástí operačních systémů Microsoft Windows. Vývoj začal již v 90. letech 20. století, nicméně první vydání beta verze 1.0 proběhlo až v roce 2002 (Šíma a kol., 2006). Dle Microsoftu (2020) je nejnovější verzí 4.8 vydaná 18. dubna 2019.

Ačkoli je .NET framework součástí operačních systémů Windows, není automaticky aktualizovaný – novější verze obsahují funkce, které starší typy neznají. Proto je potřeba dbát na výběr správné verze při vývoji programu v .NET, jelikož u klienta nemusí být automaticky nejnovější verze, čímž by daný program nefungoval správně (Šíma a kol., 2006).

.NET framework v sobě obsahuje dvě sady:

- sadu obecného jazykového prostředí CLR (Common Language Runtime), která zajišťuje správu paměti, spouštění a ověřování kódu, správu vláknových činností, kompilaci a další systémové služby,
- sadu knihoven tříd, které jsou organizovány v hierarchii tzv. namespaces (Šíma a kol., 2006).

3.2.2 Xamarin framework

Xamarin je vývojový framework, který umožňuje kódovat nativní, multiplatformní iOS, Android a Windows mobilní aplikace v programovacím jazyku C#. Je portem .NET frameworku pro operační systémy iOS a Android s podporou Windows Phone (viz obrázek 1). Základem Xamarin.Android je Mono pro Android a pod Xamarin.iOS je MonoTouch. Jedná se o vazby C# k nativním rozhraním API pro Android a iOS pro vývoj v mobilních zařízeních a tabletech. Každé nové vydání operačních systémů Android a iOS je doplněno novým vydáním Xamarinu, které zahrnuje vazby na jejich nová API. Xamarin.Forms je vrstva nad ostatními vazbami uživatelského rozhraní a rozhraním Windows Phone API, které poskytuje knihovnu uživatelského rozhraní plně napříč platformami (Hermes, 2015).

Obrázek 1: Xamarin framework

(zdroj: Hermes, 2015)

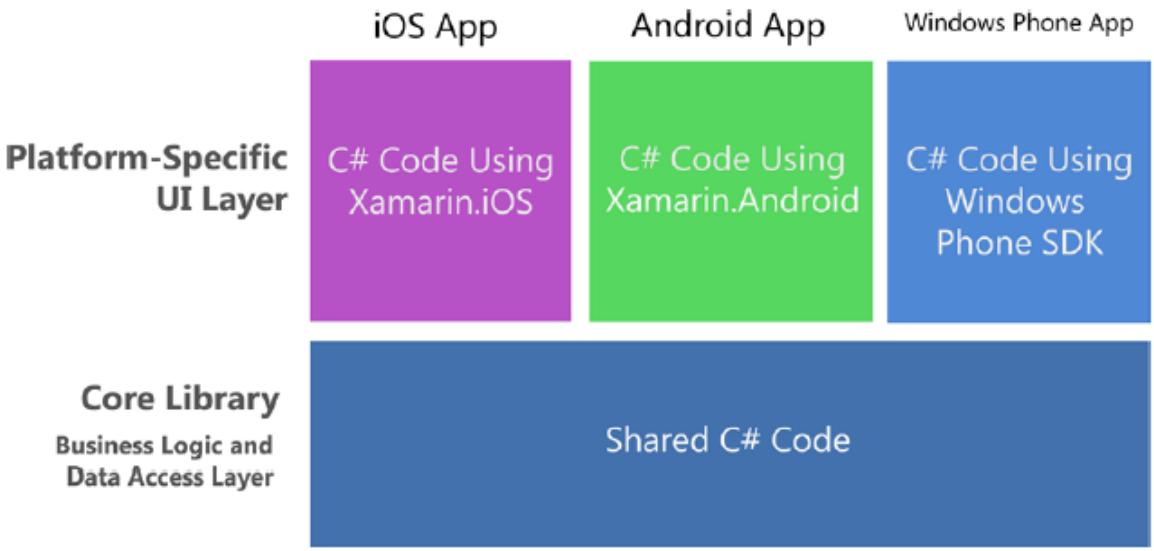
Existují dva hlavní přístupy k vývoji mobilního uživatelského rozhraní pomocí C#, které můžeme použít samostatně, zaměnitelně nebo v tandemu:

- Xamarin.Forms – multiplatformní UI nástroj pro Android, iOS, and Windows Phone.
- Specifické UI – využívají Xamarin.Android, Xamarin.iOS a Windows Phone SDK.

Xamarin.Forms obsahují plně multiplatformní sadu nástrojů poskytující jednu sadu ovládacích prvků, rozvržení a stránek uživatelského rozhraní, které chytře mapují příslušné vazby nativního uživatelského rozhraní v systémech iOS, Android a Windows Phone (viz obrázek 2). Protože Xamarin.Forms je novější než knihovny specifické pro platformu, je také méně plnohodnotný (Hermes, 2015).

Obrázek 2: Architektura Xamarin.Forms
(zdroj: Hermes, 2015)

Přístup založený na specifické platformě (viz obrázek 3) je starší a ustálenější, a proto poměrně podrobný a plně vybavený. To zahrnuje knihovny, které se přímo váží na rozhraní API specifická pro platformu: Xamarin.Android pro Android a Xamarin.iOS pro iOS. Pro Windows Phone používáme Windows Phone SDK, nativní API, které nevyžaduje žádné vazby Xamarin. Tyto knihovny specifické pro platformu nám poskytují hluboký přístup k nativním uživatelským rozhraním, které poskytují vizuálně ohromující a interaktivně bohaté uživatelské prostředí (Hermes, 2015).



Obrázek 3: Architektura platform-specific UI vrstvy

(zdroj: Hermes, 2015)

3.2.3 API

API je zkratka pro aplikační programové rozhraní. Jeho cílem je poskytnout rozhraní, na které by mohly ostatní programy posílat příkazy, které spustí nějaký proces uvnitř aplikace a případně vrátit nějaký výstup. Koncept se může zdát trochu abstraktní, ale ve skutečnosti existují API ve všem, co nějak souvisí s počítači. Pár příkladů ze skutečného života dle Lopeze (2016):

- Při používání jakékoli aplikace v počítači musí daná aplikace určitým způsobem komunikovat s operačním systémem (například vyžádání si určitého souboru, odeslání zvuku do reproduktorů apod.). Všechny tyto interakce mezi aplikací a systémem jsou možné díky API, které operační systém poskytuje.
- Pro interakci s uživatelem poskytuje mobilní aplikace grafické rozhraní (GUI). Rozhraní zachycuje všechny události, které uživatel spouští, jako je kliknutí na display nebo psaní na klávesnici, aby je mohl odeslat na server. GUI komunikuje se serverem pomocí API stejným způsobem, jakým program komunikuje s operačním systémem.
- Když se vytvoří web, který má zobrazovat tweety z uživatelského účtu Twitter, musí určitým způsobem komunikovat s Twitterem. Twitter poskytuje API, ke kterému lze přistupovat přes HTTP. Po ověření lze zasláním správných požadavků HTTP aktualizovat a/nebo načíst data z aplikace.

Specifickým typem API je REST API (REpresentational State Transfer API). Pro komunikaci používá protokol HTTP, takže je webovými aplikacemi nejpoužívanější. Ve skutečnosti se příliš neliší od funkce webových stránek, protože klient odešle požadavek HTTP a server odpoví pomocí odpovědi HTTP. Rozdíl je v tom, že rozhraní REST API intenzivně využívá stavové kódy HTTP k pochopení toho, co je odpověď, a namísto vrácení zdrojů HTML s CSS a JS používá odpověď JSON, XML nebo jakýkoli jiný formát dokumentu s pouhými informacemi, a nikoli grafické uživatelské rozhraní (Lopez, 2016).

3.2.4 Návrhové vzory

V moderním světě je architekt, který umí navrhnout efektivní oddělený systém, považován za neocenitelný přínos pro společnost. Je to proto, že oddělený systém sníží náklady na údržbu a pomůže vývojářům provádět rychlé změny programu, aniž by to ovlivnilo celý systém. Oddělený systém podporuje paralelní vývoj, což znamená, že lze práci rozdělit mezi různé vývojáře, což výrazně zkrátí dobu vývoje (CodeProject, b.r.).

3.2.4.1 MVC

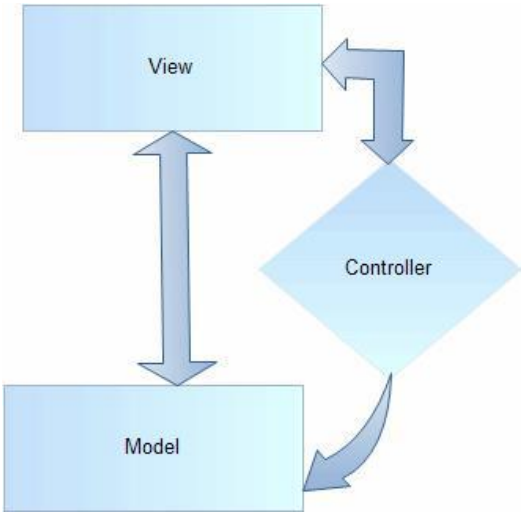
MVC (Model – View – Controller) je oblíbeným a nejstarším vzorem pro vývojáře webu. Pomáhá oddělit různé části webové aplikace a ponechává kód snadno pochopitelný. Tyto součásti jsou známé jako modely (model), pohledy (view) a řadiče (controller):

- Model – model spravuje data a/nebo logiku, která jsou vyžadována, a informuje přes řadič pohled událostmi.
- Pohled – pohledy obsahují šablony odpovědí, které jsou předávány řadiči pro další manipulaci, tzn. zprostředkovává uživateli interakci s aplikací.
- Řadič – řadič řídí požadavky a rozhoduje, jaká data použít a jak vykreslit příslušnou šablonu, tzn. zajišťuje koordinaci mezi modelem a pohledem (Lopez, 2016).

Dle CodeProject (b.r.) lze návrhový vzor MVC popsat následujícími kroky (viz obrázek 4):

1. uživatel klikne na tlačítko v uživatelském rozhraní,
2. řadič obdrží oznámení z daného objektu,
3. řadič přistoupí k modelu a provede určitou akci,
4. model na základě logiky propočte všechny parametry se změněnými daty,
5. komponenta získá přímo z modelu informaci o změnách,

6. komponenta vykreslí způsobenou akci.

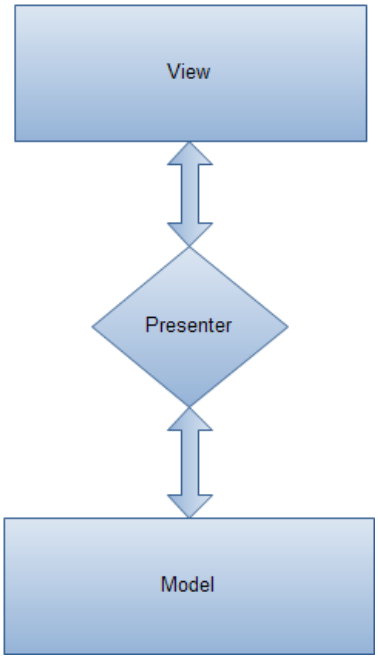


Obrázek 4: Návrhový vzor Model – View – Controller
(zdroj: Differences between MVC and MVP for Beginners – CodeProject, b.r.)

3.2.4.2 MVP

MVP (Model – View – Presenter) rozvíjí model MVC tím, že mezi model a pohled vkládá tzv. moderátora (Presenter). Události jsou přijímány pohledem a ten případně volá moderátora, který je následně odpovědný za aktualizaci pohledu upravenými daty (viz obrázek 5).

- Model (Model) – zprostředkovává datové rozhraní – když jsou potřeba data, jdou vždy přes toto rozhraní.
- Pohled (View) – zprostředkovává různé pohledy, které jsou předkládány uživateli, a ten s nimi interaguje (vyvolává určité události).
- Moderátor (Presenter) – slouží jako zprostředkovatel, ve kterém je naprogramována logika mezi uživatelem a programem – události volají funkce, které obsahuje moderátor. Jelikož zde není přímá komunikace mezi kódem, zaměřuje se vývoj spíše na grafické rozhraní, které nevyvolá poškození kódu (CodeProject, b.r.).

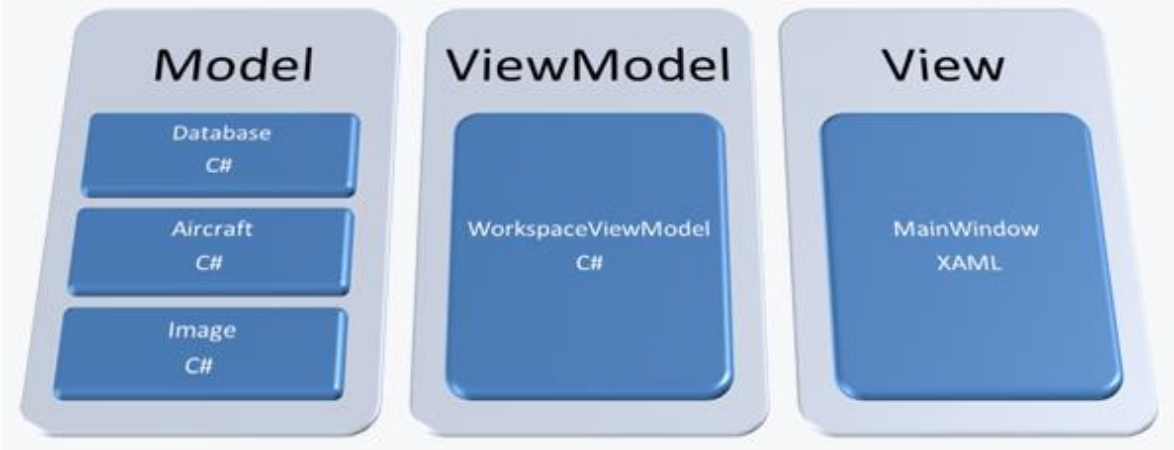


Obrázek 5: Návrhový vzor Model – View – Presenter
(zdroj: Differences between MVC and MVP for Beginners – CodeProject, b.r.)

3.2.4.3 MVVM

Model – View – ViewModel je návrhový vzor pro WPF (Windows Presentation Foundation) aplikace. Nabízí řešení, jak oddělit logiku aplikace od uživatelského rozhraní. Kódu je pak méně, vše je přehlednější a případné změny nejsou implementační složitostí. MVVM odděluje data, stav aplikace a uživatelské rozhraní. Samotné WPF bylo vytvořeno tak, aby se v něm MVVM používal pohodlně. Proto se v něm využívá binding a command – náhrada za uživatelské rozhraní řízené událostmi (viz obrázek 6).

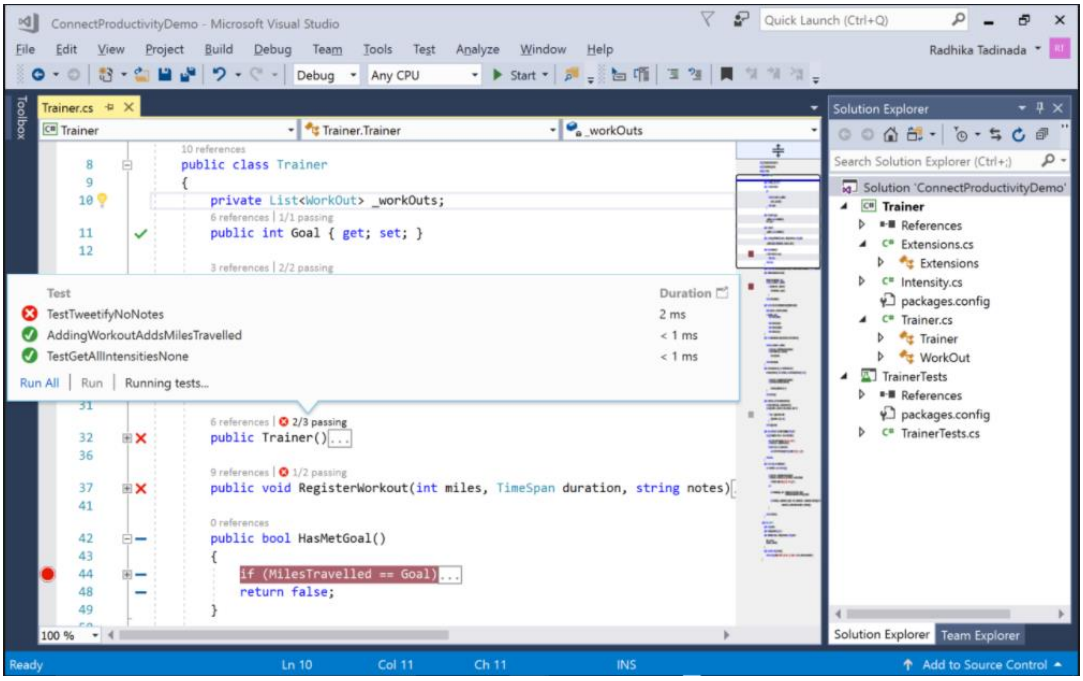
- Model popisuje data, se kterými aplikace pracuje. Například třídy, které se mapují na tabulky databáze, jsou modely. Model nesmí o stavu ovládacích prvků nic vědět.
- Pohled (View) reprezentuje uživatelské rozhraní v jazyce XAML. Může se jednat o okno aplikace, stránku, nebo ovládací prvek. Označuje se také jako UI, formulář nebo prezentační vrstva.
- Pohled-Model (ViewModel) spojuje Model a Pohled a drží si stav aplikace. Ovládací prvky jsou pomocí bindingu propojeny s ViewModelem a čerpají z něj svůj obsah. Provádí se v něm filtrování dat v závislosti na stavu aplikace (Dotnetportal, b.r.).



Obrázek 6: Návrhový vzor Model – View – ViewModel
(zdroj: MVVM: Model-View-ViewModel – Dotnetportal, b.r.)

3.2.5 Microsoft Visual Studio

Microsoft Visual Studio je z anglické zkratky IDE (Integrated Development Environment) vývojové prostředí vytvořené firmou Microsoft (viz obrázek 7). Toto vývojové prostředí umožňuje vytvářet jak tzv. konzolové aplikace, ve kterých není typické grafické rozhraní, ale pouze černé okno (terminál), tak i nativní aplikace, které využívají balíčky Windows Forms pro vytvoření grafického rozhraní se stejným vzhledem jako v operačním systému Windows. Jedná se o čisté prostředí, které neobsahuje žádné programovací jazyky – jednotlivé balíčky (VSPackage) lze do Visual Studia nainstalovat, čímž se dle Petzolda (2016) stává univerzálním prostředím napříč libovolnými programovacími jazyky.



Obrázek 7: Vývojové prostředí Microsoft Visual Studio
(zdroj: Visual Studio IDE, Code Editor, VSTS, & App Center, b.r.)

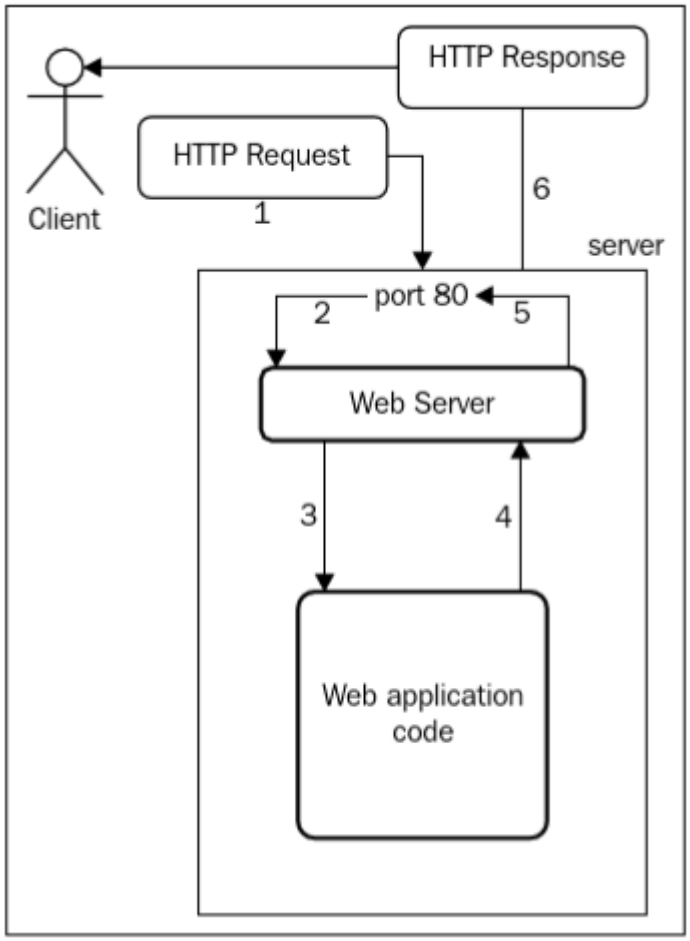
3.3 Web server

Web server je část softwaru, který je nainstalovaný a spuštěn na nějaké stanici a očekává (naslouchá) požadavky na specifickém portu. Nejčastěji se jedná o port 80, případně šifrovaný port 443 (Lopez, 2016).

Funkcí webového serveru je směrovat externí žádosti ke správné aplikaci ke zpracování. Jakmile aplikace navrátí odpověď, přepošle webový server odpověď klientovi. Detailnější popis k obrázku 8 podle Lopeze (2016) lze popsat následovně:

1. Klient (například prohlížeč) odešle žádost typu GET nebo POST.
2. Server obdrží žádost a předá jí na konkrétní port. Jestliže na daném portu naslouchá webový server, převezme kontrolu.
3. Webový server rozhodne, o jakou se jedná aplikaci – nejčastěji se jedná o soubor v souborovém systému. Pro rozhodnutí musí webový server zvážit část URL – například pro adresu „http://myserver.com/app1/hi“ by se pokusil vyhovět žádosti „app1“.

4. Webová aplikace po obdržení žádosti od webového serveru vygeneruje odpověď a předá ji zpět na webový server.
5. Webový server navrátí odpověď na patřičný port.
6. Odpověď je doručena klientovi.



Obrázek 8: Datový tok žádost-odpověď u webserveru
(zdroj: Lopez, 2016)

Podle Adaptic (2001) patří webový server Apache, programovací jazyk PHP a systém řízení báze dat MySQL k tzv. triádě (jedná se o tři komponenty nejčastěji využívané k vytváření dynamických webových stránek).

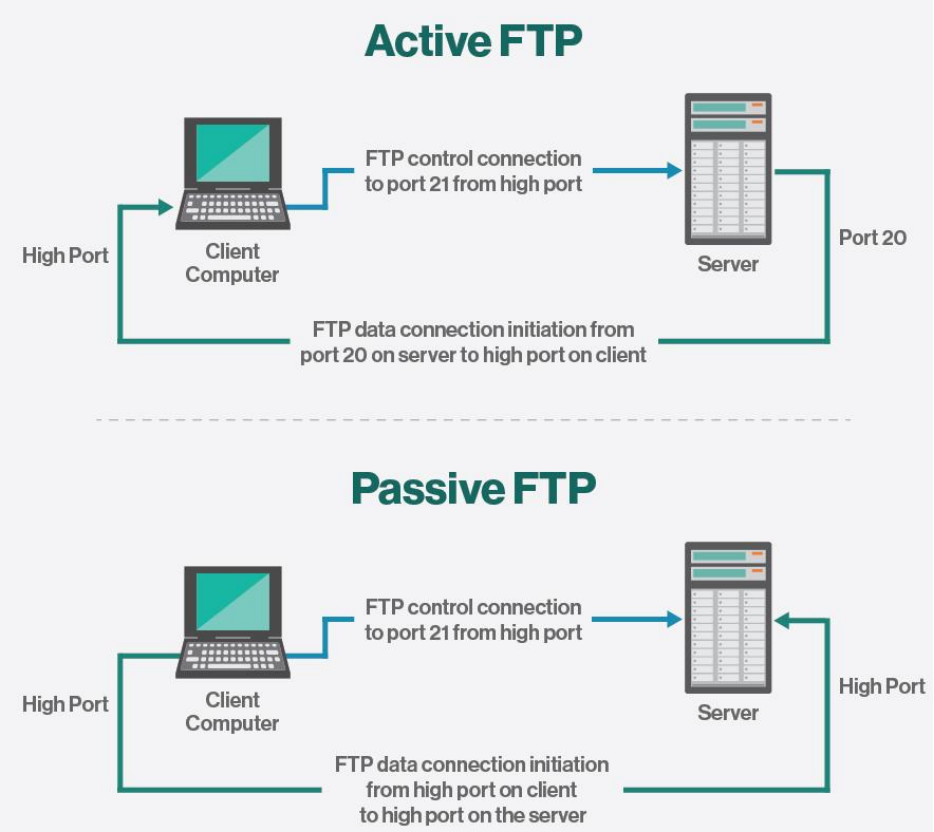
3.4 FTP

Z anglické zkratky FTP (File Transfer Protocol) se jedná o jeden ze standardizovaných internetových protokolů používaných k přenosu mezi klientem a serverem v počítačové síti,

který byl vyvinut v ranných 70. letech 20. století Abahyem Bhushanem v rámci jeho studia. Během let se FTP nadále vyvíjelo, proběhlo mnoho revizí, kde se s ohledem na zvyšující se nároky na bezpečnost měnil hlavně způsob přenosu (Whoishostingthis, b.r.).

Protokol pracuje ve dvou módech (viz obrázek 9):

- aktivní – kdy klient naslouchá na příslušném portu odpověď od serveru,
- pasivní – kdy klient nedostává odpověď na příslušný port (například z důvodu blokování na firewallu, překladu adres apod.) (Techtarget, b.r.).



Obrázek 9: Diagram pasivního a aktivního FTP připojení
(zdroj: What is File Transfer Protocol (FTP)? – Definition from WhatIs.com)

Zahájení komunikace je prováděno klientem na standardním portu 21, kde je poskytován nešifrovaný přenos dat (data jsou přenášena v tzv. plaintextu, tudíž v případě odchycení komunikace lze získat konkrétní údaje jako hesla, přihlašovací jména apod.). Vývojáři FTP protokolu drží krok s nejnovějšími bezpečnostními zásadami a byla vyvinuta dvě hlavní rozšíření – FTPS a SFTP (Techtarget, b.r.).

První a nejpoužívanější rozšíření má zkratku FTPS (File Transfer Protocol Secure), které vychází ze stejné metodologie, ale přidává k přenosu protokol SSL pro zajištění šifrované komunikace. Druhé rozšíření je novější a má zkratku SFTP (SSH File Transfer Protocol), které se již od základních verzí liší tím, že komunikace není zabezpečená SSL certifikátem, ale pro přenos je využíván SSH (Secure Shell) protokol. Ten zajišťuje přenos dat pomocí paketů (nikoli proudově), což zrychluje přenos a umožňuje posílat větší soubory (Whoishostingthis, b.r.).

3.5 Databáze

Data jsou pravděpodobně základním kamenem většiny webových aplikací. Samozřejmě, že aplikace musí být hezká, rychlá, bez chyb a tak dále, ale pokud je pro uživatele něco důležité, tak jaká data pro ně spravovat. Z toho lze vyvodit, že správa dat je jednou z nejdůležitějších věcí, která se musí zvážit při navrhování aplikace. Správa dat zahrnuje nejen ukládání souborů určených pouze pro čtení, ale také přidávání, načítání, aktualizace a odstraňování jednotlivých informací. K tomu je potřeba nástroj, který kategorizuje data a usnadňuje tyto úkoly – a tím je databáze (Lopez, 2016).

Databáze jsou nástroje pro správu dat. Mezi základní funkce databáze patří vkládání, vyhledávání, aktualizace a mazání dat, i když většina databázových systémů dělá víc než to. Databáze jsou rozděleny do dvou různých kategorií podle toho, jak ukládají data: relační a nerelační databáze. Relační databáze strukturují data velmi podrobným způsobem, což nutí uživatele, aby používal definovaný formát a umožňoval vytvoření spojení – tj. vztahů mezi různými částmi informací. Nerelační databáze jsou systémy, které ukládají data uvolněněji, jako by neexistovala žádná zjevná struktura (Lopez, 2016).

Hlavní jednotkou jsou tabulky, kde se každá tabulka skládá ze sloupců a řádků. V dnešní době se nejvíce využívají relační databázové systémy (DBS), kde platí pravidlo SŘBD (systému řízení báze dat) – všechna data jsou spravována pouze pomocí relačních operací. Tudíž musí platit následující rovnice (1) (Vostrovský, 2004):

$$DBS = DB + SŘBD \tag{1}$$

3.5.1 Datové modelování

Při vytváření a modelování databáze se v dnešní době podle Vostrovského (2004) používají hlavní dva postupy – postup zdola nahoru a postup shora dolů. Pro postup zdola nahoru se začíná s univerzální relací U, ve které jsou všechny atributy, a až následně se specifikují jejich vzájemné závislosti. Jedná se o komplexní a delší postup, a proto je vhodný pro model s méně než 25 atributy. Jelikož jsou databáze většinou velké komplexity, používá se spíše postup shora dolů v následujících krocích:

1. Specifikace jednotlivých entitních množin,
2. specifikace vztahů – 1 : 1, 1 : N, M : N,
3. přiřazení primárních klíčů,
4. transformace do logické struktury spolu s kardinalitou vztahů,
5. celý model se prověří z hlediska datové normalizace (Vostrovský, 2004).

Výše zmíněná datová normalizace prověřuje strukturální správnost a konzistenci datového modelu. Existuje celkem pět normálních forem (dále jako NF), ale postačuje využívání do třetí NF. Dle první NF nesmí relace obsahovat vícenásobná data (tzn. stejná data uložena ve více tabulkách). Ve druhé NF se ověřuje, zda data závisí na celém klíči (tzn. k jedné hodnotě A náleží přesně jedna hodnota B) a ve třetí NF se ověřuje, že neklíčová data závisí jen na klíči a nikoli mezi sebou. Všechny NF se začleňují (viz obrázek 10), tj. pakliže je splněna třetí NF, je zároveň splněna první a druhá NF (Vostrovský, 2004).

Obrázek 10: Začleňování normálních forem
(zdroj: Vostrovský, 2004)

S vytvářením databází a jejich datových modelů je nutné zajistit i tzv. integritu dat. Integrita dat znamená, že v databázi musí existovat určitý stroj, který obsahuje mechanismy

k zabezpečení báze dat před neúmyslným (či úmyslným) poškozením a zneužitím. Nejčastěji je realizována jako zajištění integrity dat, což znamená, že se vytvářejí záchytná pravidla již při realizaci modelu.

Tato pravidla se nazývají tzv. integritní omezení IO a dělí se na tři typy:

- Entitní integrita – jednoznačně identifikuje každý řádek dotyčné relační tabulky primárním klíčem.
- Doménová integrita – definuje množinu všech přípustných hodnot určitého daného atributu (sloupce) relační tabulky.
- Referenční integrita – garantuje korektnost vztahů mezi logicky souvisejícími tabulkami pomocí cizích klíčů (Vostrovský, 2004).

3.5.2 MySQL

Podle Lopeze (2016) je MySQL oblíbenou volbou vývojářů PHP již poměrně dlouhou dobu a podle Wellinga (2017) je v dnešní době nejvíce využívaným SŘDB vytvořeným švédskou firmou MySQL; nyní vlastněná společností Sun (Oracle). Je to relační databázový systém, který používá jazyk SQL jako jazyk pro komunikaci se systémem. MySQL zajišťuje, aby k datům mohlo přistupovat více uživatelů současně, nabízí rychlý přístup a dohlíží na to, aby přístup získali pouze oprávnění uživatelé (Welling, 2017).

3.6 World Wide Web

World Wide Web (zkráceně www) je v doslovném překladu světově rozsáhlá pavučina pro systém prohlížení, ukládání a odkazování mezi hypertextovými dokumenty. Historie sahá až do roku 1946, kdy byl napsán krátký článek o možném fungování počítačů v budoucnosti americkým spisovatelem Murrayem Leinsterem. První prototyp byl vytvořen v roce 1980 britským tvůrcem Timem Bernersem Leem při práci na projektu „Enquire“ v evropské organizaci pro jaderný výzkum CERN. Zmíněný projekt „Enquire“ zahrnoval obyčejný software s databází lidí, ve kterém Tim Berners Lee hojně využíval právě hypertextové odkazy, které byly navzájem provázané.

První formální návrh projektu „WorldWideWeb“ jako „web“ hypertextových odkazů byl publikován 12. listopadu 1990 Timem Bernersem Leem a jeho spolupracovníkem Robertem

Cailliauem. Jedním z požadavků bylo zobrazení na koncových zařízeních (prohlížečích), a proto bylo www postaveno na modelu klient-server a na třech hlavních technologiích:

- HTML,
- HTTP,
- webové servery a webové prohlížeče (Webfoundation, b.r.).

3.6.1 HTML

HyperText Markup Language (zkráceně HTML) je značkovací jazyk, který slouží k tvorbě webových stránek a tvoření dokumentového objektového modelu (DOM – Document Object Model), kde jsou jednotlivé hypertextové dokumenty navzájem propojeny. Byl vytvořen v roce 1990 Timem Bernersem Leem pro jednodušší práci při tvorbě www (W3schools, b.r.).

Použitím standardní definice typu dokument (DTD – Document Type Definition) popisuje HTML striktně definovanou strukturu webových stránek (respektive jejích elementů). Každá verze HTML obsahuje definovanou množinu elementů (tagů), které ve výsledku generují a poskládají vzhled webové prezentace. Základní struktura musí obsahovat určité značky, viz tabulka 1 (Jakpsatweb, b.r.).

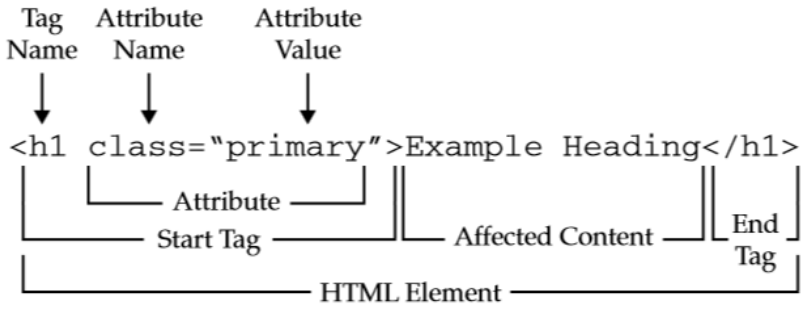
Tabulka 1: Základní struktura HTML dokumentu

Značka (tag)	Význam	Párový tag	Výskyt
Doctype	specifikace struktury DTD	ne	začátek souboru
Html	začátek HTML dokumentu	ano	ihned po Doctype
Head	hlavička stránky	ano	ihned po Html
Body	tělo stránky	ano	ihned po Head

(zdroj: Jakpsatweb, b.r.)

Jednotlivé elementy (značky nebo tagy) jsou definovány pomocí hranatých závorek, přičemž se dělí na tzv. párové a nepárové značky. U párových značek je potřeba mít počáteční i koncovou značku, zatímco u nepárových stačí pouze počáteční. Každá značka je nazvána tzv. názvem tagu (Tag Name) a následně upřesněna jménem atributu (Atribute Name) a jeho hodnotou (Attribute Value) – struktura viz obrázek 11. V dřívějších verzích HTML se

využíval atribut „style“ pro formátování tagu, ale byl nahrazen (aktuálně je nevalidní), a využívá se atributů třídy (class) a id, dle kterých se následně provádí formátování přes CSS styly (W3schools, b.r.).

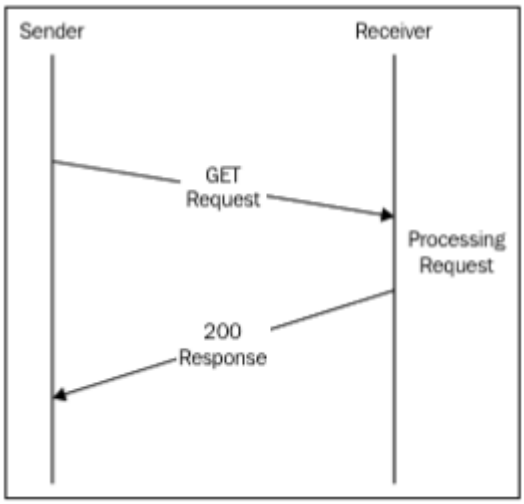


Obrázek 11: Grafické znázornění HTML syntaxe
(zdroj: W3schools, b.r.)

Značkovací jazyk prošel celkem pěti vývojovými větvemi – aktuální verze je HTML5, která nadále definuje budoucí vývoj stránek a je využívána novodobými webovými stránkami. Mezi největší změny patří odstranění starých a pro dnešní a budoucí dobu nezbytných značek a definování nových sémantických značek, které strukturovaněji rozloží strukturu webových stránek. Nebylo opomenuto ani na multimediální a grafické značky, které byly taktéž vytvořeny pro podporu mediálních souborů a rozšířených formulářových prvků (Dummies, b.r.).

3.6.2 HTTP

HTTP znamená Hypertext Transfer Protocol. Jedná se o protokol, jehož hlavním cílem je umožnit vzájemnou komunikaci mezi dvěma prvky nebo uzly. Pro dosáhnutí cíle musí být zpráva formátována stylem, aby mu obě strany rozuměly, a zároveň musí obě strany dodržovat předem stanovená pravidla. V tomto modelu (viz obrázek 12) jsou dvě entity – odesílatel a příjemce. Odesílatel odesílá zprávu příjemci – tato zpráva se nazývá žádost (request). Příjemce získá žádost, zpracuje ji a vygeneruje druhou zprávu – odpověď (response) (Lopez, 2016).



Obrázek 12: Jednoduchý HTTP GET požadavek
(zdroj: Lopez, 2016)

HTTP zpráva obsahuje několik částí:

- URL (Uniform Resource Locator) reprezentuje cíl dané zprávy.
- HTTP metoda je sloveso zprávy. Identifikuje, jakou akci chce odesílatel provést. Nejčastější je GET a POST. GET se používá, když odesílatel něco chce (vyžaduje informaci nazpět), nejčastěji HTML kód webové stránky. POST se používá, když chce odesílatel provést určitou akci, která změní data u příjemce – například uložení názvu profilu.
- Tělo zprávy je k dispozici většinou v odpovědi, nicméně požadavek ho může obsahovat také. Může obsahovat text v jakémkoli formátu – HTML, obyčejný text, JSON apod.
- Hlavička zprávy jsou metadata, která příjemce potřebuje k pochopení obsahu zprávy.
- Návratový kód reprezentuje stav odpovědi pomocí číselného kódu, dle kterého prohlížeče a ostatní nástroje reagují. Mezi nejčastější kódy se řadí 200 pro úspěšný požadavek, 401 pro nepovolení z důvodu oprávnění, 404 pro nenalezení stránky a 500 pro interní serverový problém (chyba na straně serveru, která nelze být obnovena) (Lopez, 2016).

3.6.3 HTTPS

HTTPS (Hypertext Transfer Protocol Secure) je komunikační protokol, který chrání integritu a důvěrnost dat mezi počítačem uživatele a webovými stránkami. Uživatelé při používání

webu očekávají bezpečí a soukromí. Bez ohledu na obsah webové stránky se doporučuje chránit připojení uživatelů pomocí protokolu HTTPS. Data odeslaná pomocí protokolu HTTPS jsou zabezpečena prostřednictvím protokolu TLS (Transport Layer Security), který poskytuje tři hlavní vrstvy ochrany:

- šifrování – šifruje přenesená data tak, aby byla ochráněna před odposlechem (tzn. když uživatel prohlíží webové stránky, nikdo nemůže „odposlouchávat“ jeho konverzaci, sledovat aktivitu na stránkách ani ukrást jeho údaje),
- integrita dat – data nelze během přenosu pozměnit ani poškodit (ať záměrně, nebo jinak),
- ověření – potvrzuje, že uživatelé komunikují s požadovaným webovým serverem, poskytuje ochranu proti útokům typu man-in-the-middle a posiluje důvěru uživatelů, což přináší další výhody (Google, b.r.).

Podle Google (b.r.) je v dnešní době vyvíjen tlak na využívání pouze TLS verze 1.2 spolu s bezpečnostním certifikátem vydaným jednou z certifikačních autorit a certifikačním klíčem o 2 048 bitech, jelikož tato verze využívá zabezpečenější hashovací algoritmus SHA-256 a odstraňuje podporu dalších kryptografických klíčů, které se již podařilo úspěšně napadnout či rozluštit.

3.6.4 URI

URI (Uniform Resource Identifier) slouží k identifikaci určitého zdroje, kdy univerzální syntaxe umožňuje přístup k dostupným objektům pomocí existujících protokolů. Syntaxe je:

schéma:[//[uživatel[:heslo]@]host[:port]][/cesta][?dotaz][#fragment]

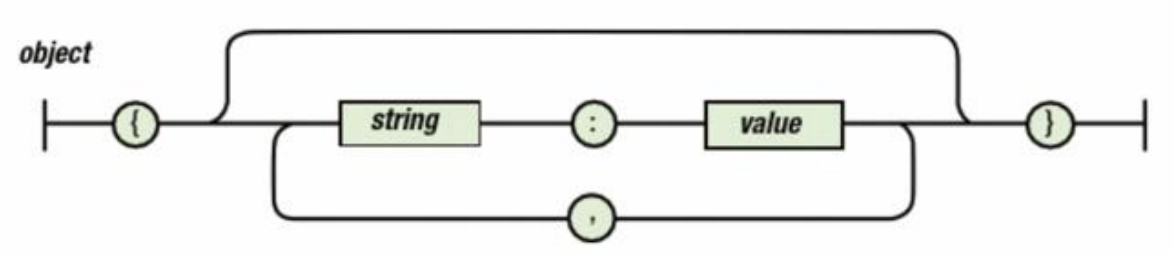
- schéma – specifikuje přenosový protokol (např. http, ftp, https apod.),
- uživatel a heslo – slouží k autentizaci pro přístup k objektu,
- host – specifikuje doménové jméno nebo IP adresu objektu,
- port – specifikuje port pro komunikaci (https využívá port 443 nebo 8443),
- cesta – specifikuje umístění určitého dokumentu,
- dotaz – umožňuje specifikovat parametry, které upravují žádost pro objekt,

- fragment – umožňuje vytvářet hypertextový odkaz v rámci jedné stránky (World Wide Web Consortium, b.r.).

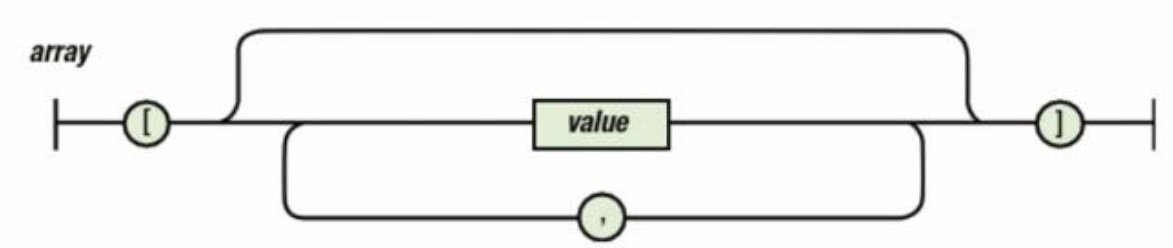
3.6.5 JSON

Javascript Object Notation datový formát (zkráceně JSON) je derivovaný formát ze skriptovacího jazyka Javascript splňující stejná pravidla, která definuje ECMA-262 standardizace. JSON je znám jako standard pro výměnu dat, použitelný kdekoli dochází k výměně dat. Dle Smithe (2015) jde v kostce o textovou reprezentaci definovanou malou sadou řídicích pravidel, ve kterých jsou data strukturována. Specifikace uvádí, že data mohou být strukturována ve kterémkoli ze dvou následujících složení:

- kolekce dvojic název/hodnota (viz obrázek 13),
- uspořádaný seznam hodnot (viz obrázek 14).



Obrázek 13: JSON – kolekce dvojic název/hodnota
(zdroj: Smith, 2015)



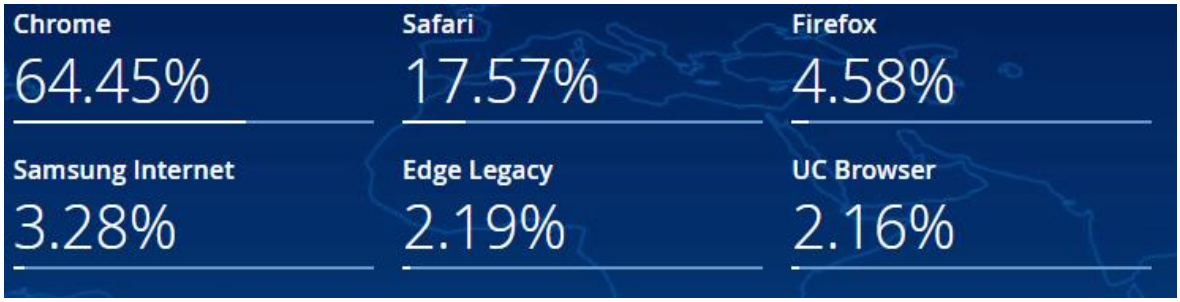
Obrázek 14: JSON – uspořádaný seznam hodnot
(zdroj: Smith, 2015)

3.6.6 Webové prohlížeče

Softwarový koncový klient neboli webový prohlížeč vyvinul Tim Berners Lee již v roce 1990, spolu se značkovacím jazykem HTML, webovými adresami (URI) a protokolem HTTP. Je to program, který umožňuje vygenerovat ze zdrojového kódu (konkrétně ze značek

HTML elementů) vizuální webovou stránku s obsahem v lidem čitelné formě (Webfoundation, b.r.).

Během času bylo vyvinuto mnoho různých webových prohlížečů, mezi nejznámější a nejvyužívanější se řadí Google Chrome od firmy Google, Microsoft Edge od firmy Microsoft (náhrada Internet Exploreru), Safari od firmy Apple a UC/Samsung Browser od firmy Android (nativní prohlížeč na Android zařízeních). Statistiky využití prohlížečů jsou zobrazené na obrázku 15 publikované firmou StatCounter (2020) v únoru 2020.



Obrázek 15: Procentuální podíl využití prohlížečů k únoru 2020
(zdroj: StatCounter Global Stats – Browser, OS, Search Engine including Mobile Usage Share, 2020)

3.7 Design

Kvalitní grafický návrh nebo řešení ať aplikace, nebo webové stránky buduje značku a výrazně zvyšuje její hodnotu. Začínající firmy sázejí na výrazný až extravagantní design produktu a využívají ho pro marketingový tah k dohnání případné existující konkurence, nebo k plnění business plánu. Celkovou hodnotu podniku také navyšují jednotlivé prvky designu (logo, styl či design produktů), čímž se stávají firemními aktivy (Mckay, 2013).

3.7.1 Interakční design

Základní věc, která propojuje uživatele a technologii produktu, je uživatelské rozhraní (UI – User Interface). Právě přes rozhraní zadávají uživatelé pokyny produktu (pro lepší pochopení si lze představit, že jde v podstatě o rozhovor mezi uživatelem a produktem), čímž program provádí úkoly a uživatelé dosahují svých cílů. Správný design má minimalizovat tuto komunikaci způsobem přirozeným, profesionálním a snadno srozumitelným. Špatný design nutí uživatele k zamyšlení, hledání a občas k experimentování pro získání svých cílů (Mckay, 2013).

Rozmístění jednotlivých prvků podléhá tzv. myšlenkovému modelu (MindModel), kdy je potřeba brát ohled na:

- minimální počet uživatelských potvrzení,
- minimální počet klikání myší a dalších kláves,
- minimální počet dotazových dialogů,
- zachování rozmístění ovládacích prvků,
- umístění aktivních prvků vždy vpředu,
- předsouvání aktivních prvků zprava doleva,
- umisťování pasivních prvků na pravé straně (Mckay, 2013).

Před zahájením vytváření rozložení uživatelského rozhraní je třeba vytvořit uživatelskou specifikaci, která sdružuje všechny uživatelské požadavky na systém z pohledu uživatelů. Za zmíněné uživatele je potřeba považovat všechny uživatele/osoby, které budou s daným rozhraním interagovat a které budou očekávat určitou podporu v práci z jeho strany (Mckay, 2013).

3.8 SEO

Optimalizace pro vyhledávače (zkráceně SEO z anglického Search Engine Optimization) jsou určité techniky, díky kterým se mohou webové stránky umístit na lepších pozicích při vyhledávání. Z pohledu SEO se jedná o dva faktory – on-page faktory a off-page faktory. On-page faktory zahrnují optimalizaci stránek jako takových (tj. zdrojový kód, meta značky, texty, URL adresy, popisky obrázků apod.), které můžeme ovlivnit. Off-page faktory nelze zcela ovlivnit – jedná se o faktory v síti Internet (tj. počet zpětných odkazů, síla odkazů, sociální signály, konkurence apod.) (Kubíček, 2008).

Nejdůležitějším aspektem je zahrnutí (indexace) webových stránek u tzv. vyhledávačů/pavouků. Právě pavouci stránky procházejí (anglicky crawling) a jejich obsah je evidován u webových vyhledávačů, které je na základě určitých algoritmů umisťují ve výsledcích hledání. Takové procházení začíná odkazem, kdy pavouk navštíví stránku a její obsah si stáhne do skladiště na vlastním serveru. Po analýze zdrojového kódu postupně přeskakuje mezi jednotlivými odkazy (tagy <a>) a proces stále opakuje (Kubíček, 2008).

4 Vlastní práce

4.1 Analýza a požadavky

Hlavním cílem tohoto systému je usnadnit uživatelům (klientům cestovní kanceláře) orientaci v dané lokalitě, kde na základě určitých specifikací zaměstnanci vkládají informace o dané zemi, destinaci, oblasti, ale především informace o restauracích a zájmových místech (tzv. POI) – tj. lékárny, nemocnice, směnárny, pošta či pobočky kanceláře. Celý systém sestává ze tří částí:

- Administrační část (back-end) pro zaměstnance pro vkládání a úpravu míst.
- Webová prezentace pro online zobrazování a hledání míst přes webový prohlížeč.
- Mobilní aplikace, která může být stažena do chytrého mobilního telefonu a využívána nezávisle na webovém prohlížeči.

Cílem zavedení tohoto systému je předání všech důležitých i pomocných informací klientům dané kanceláře v novodobém způsobu, a tím zároveň omezit plýtvání papírem (tištěné materiály, které jsou předávány klientům v dané destinaci).

Přímé požadavky na daný systém jsou:

- administrativa vložených dat:
    - země,
    - destinace,
    - restaurace,
    - zájmová místa (POI);
- zobrazování dat:
    - webová prezentace,
    - mobilní aplikace.

4.2 Návrh běhového prostředí

4.2.1 Aplikační jazyk

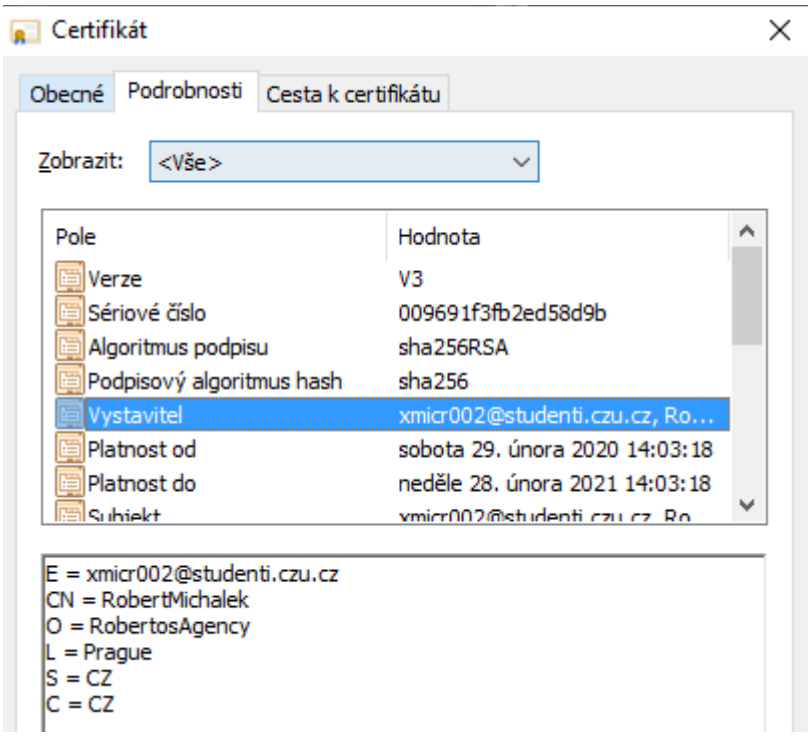
Celý systém je rozdělen na tři části – back-end administrativní část, front-end prezentační část a mobilní aplikaci. Back-end aplikace slouží pro administrativní účely a je nainstalovaná na lokálních počítačích dané firmy, čímž je izolována od volného přístupu přes síť Internet. Vyvinutí separátní administrativní části, přes kterou jako jedinou lze data vkládat/upravovat/mazat, má důvod z pohledu bezpečnosti. Umístěním administrativní části na volně dostupné prostranství, čímž je doména či veřejná IP adresa, by se celý systém vystavoval možnému riziku útoku počítačových útočníků, buď za cílem omezit fungování služeb (například DDoS útokem), či neoprávněného vniknutí možnou slabinou do vnitřní části aplikace, přes kterou by se teoreticky mohl útočník dostat až k datům v databázi. Back-end část byla vyvinuta programovacím jazykem C#, který byl zvolen z důvodu integrace s vývojovým prostředím Visual Studio, které je mimo jiné využité i k vývoji mobilní aplikace v programovacím jazyku C# a frameworku Xamarin.

Front-end část je webová stránka vytvořená značkovacím jazykem HTML, kaskádovými styly CSS, jazykem PHP a skriptovacími jazyky JavaScript a jQuery. Jak front-end část, tak mobilní část využívá pro komunikaci s databází programovací jazyk PHP. U webové prezentace je integrace s databází funkcionálně o něco snadnější, kdy samotný programovací jazyk PHP již obsahuje funkce pro přímou komunikaci s databázovým rozhraním, zatímco u mobilní aplikace neexistuje přímé rozhraní mezi kódem psaným v C# a databázovým rozhraním. Pro tuto komunikaci bylo vyvinuto v programovacím jazyku PHP rozhraní (Application Programming Interface, dále jako API), přes které si kód v C# nejprve vyžádá informaci a následně získaný výsledek ve formátu JSON zpracuje.

4.2.2 Konfigurace Apache

Apache neboli softwarový webový server slouží jako úložiště pro webovou prezentaci a API rozhraní, kde je k webovému serveru přiřazena určitá IP adresa (případně doména). Na vývoj tohoto systému byl využit Apache HTTP server, který je součástí aplikace Xampp. V tomto případě byl nakonfigurován na lokální IP adresu (localhost) a byl vytvořen SSL certifikát, který rozšiřuje protokol HTTP o šifrování, tj. HTTPS. Před zahájením komunikace mezi

koncovým zařízením (prohlížeč) a webserverem se navzájem autorizují (SSL Handshake), následně si vyměňují šifry a celá komunikace (získávání stránky/načítání) je zabezpečena. SSL certifikát byl vytvořen utilitou makecert (viz obrázek 16), která je taktéž součástí aplikace Xampp.



Obrázek 16: Vygenerovaný SSL soukromý certifikát

(zdroj: vlastní zpracování)

Po vygenerování certifikátu a jeho nasazení na webserver bylo nastaveno v konfiguračním souboru httpd-xampp.conf přesměrování všech požadavků přes protokol HTTP na protokol HTTPS pomocí modulu mod_rewrite (viz obrázek 17).

```
# Redirect /robertos folder to https
RewriteCond %{HTTPS} !=on
RewriteCond %{REQUEST_URI} robertos
RewriteRule ^(.*) https://%{SERVER_NAME}$1 [R,L]
```

Obrázek 17: Nastavení přesměrování http na https

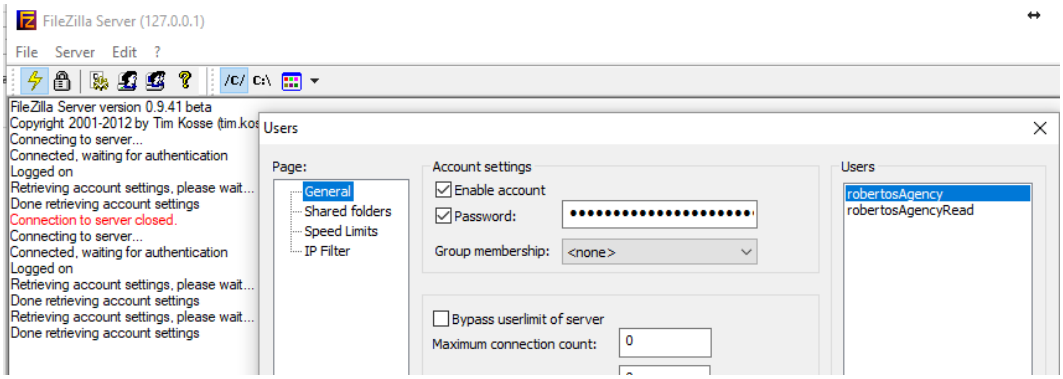
(zdroj: vlastní zpracování)

4.2.3 Konfigurace databáze

Databázové rozhraní je využíváno v rámci programu Xampp. Xampp disponuje nejpoužívanějším systémem řízení databáze dat – MySQL s rozhraním PhpMyAdmin. Všechny součásti systému komunikují s databází skriptovacím jazykem SQL. U back-end aplikace je využita knihovna MySQL Connector Net 6.9.9 od firmy Oracle, která rozšiřuje jazyk C# o možnost připojení k lokální MySQL databázi. U front-end části již programovací jazyk PHP obsahuje nezbytné funkce pro připojení k databázi a u mobilní aplikace bylo vytvořeno API rozhraní, které předává požadavky a navrací hodnoty ve formátu JSON přes protokol HTTP. Kódování databáze je utf8mb4_czech_ci, které podporuje českou diakritiku.

4.2.4 Konfigurace FTP

FTP slouží pro nahrávání obrázků k jednotlivým položkám přes administrativní rozhraní. Do lokálního programu Xampp byl přidán doplněk FileZilla Server, který emuluje funkcionalitu webového serveru, na který se může jak zapisovat (nahrávat dané obrázky), tak i číst (načítat obrázky). Pro daný systém byly vytvořeny dva účty (viz obrázek 18) – první je hlavní účet „robertosAgency“, který má právo do dané složky zapisovat (využitý v back-end aplikaci); druhý účet „robertosAgencyRead“ je pouze pro čtení.



Obrázek 18: Nastavení účtů ve FileZilla Serveru

(zdroj: vlastní zpracování)

4.2.5 Vývojové prostředí

Jednotlivé součásti systému byly vyvíjeny na operačním systému Microsoft Windows 10, ale díky využití vývojového prostředí Visual Studio a platformy .NET je zajištěna kompatibilita i případných starších operačních systémů (minimálně operační systém

Microsoft Windows 8, který je nadále podporován společností Microsoft; s operačním systémem Windows 7 se již nepočítá, neboť byl jeho vývoj ukončen dne 14. 1. 2020[1]). Systém není vyvinut pro operační systém firmy Apple.

Back-end část byla vyvinuta ve vývojovém prostředí Microsoft Visual Studio Community 2019 verze 16.3., zahrnující platformu .NET 4.5.2. Pro komunikaci mezi jazykem C# a databázovým rozhraním MySQL byl do Visual Studia připojen rozšiřující modul MySQL Connector Net 4.9.9. od firmy Oracle.

Front-end část byla vyvíjena ve vývojovém prostředí NetBeans IDE 8.0.2, které firma Oracle Corporation nabízí zdarma. Pro testování a renderování obsahu webových stránek byly primárně využívány webové prohlížeč Google Chrome verze 80.0.3987 a nový Microsoft Edge verze 80.0.361.62. Finální verze byla testována i na dalších novodobých webových prohlížečích – Internet Explorer 11, Mozilla Firefox ESR a Safari 13. Rozložení a design stránek je plně responzivní, což bylo taktéž testováno na novodobých prohlížečích (viz výše), včetně testování při různých šířkách zobrazovacího pole, které bylo určeno ze statistiky nejrozšířenějších rozlišení k lednu 2020 (viz obrázek 19).

Obrázek 19: Nejpoužívanější rozlišení k lednu 2020
(zdroj: Screen Resolution Stats Worldwide | StatCounter Global Stats, b.r.)

[1] https://support.microsoft.com/en-us/help/4057281/windows-7-support-ended-on-january-14-2020

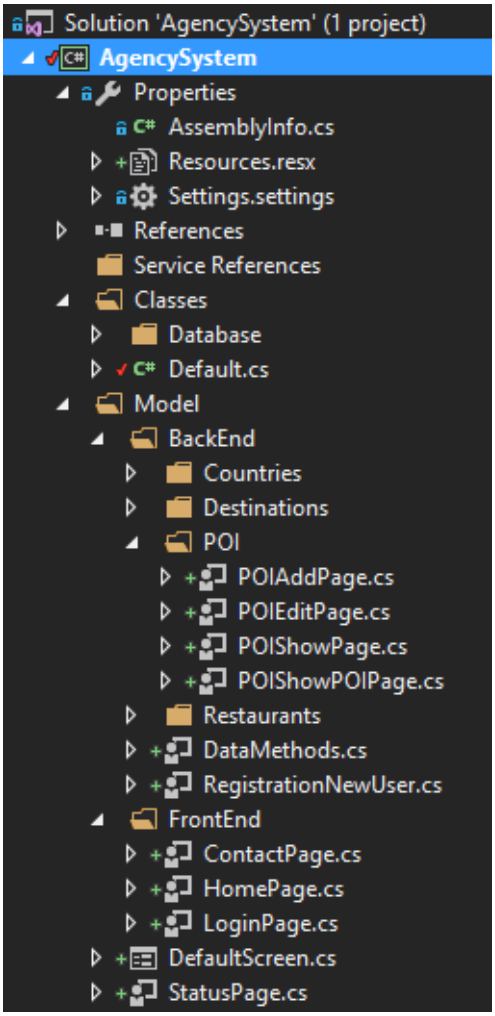
4.3 Vývoj back-end aplikace

4.3.1 Návrh objektového modelu

Objektový model u back-end aplikace využívá návrhový vzor MVP (Model – View – Presenter), který rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. Hierarchie kódu je rozdělena na tři části (viz obrázek 20):

- classes (třídy) – které slouží pouze pro práci s databází,
- resources (zdroje) – slouží pro jazykovou mutaci textů,
- model – rozdělen na FrontEnd (uživatel nepřihlášen) a BackEnd (uživatel přihlášen).

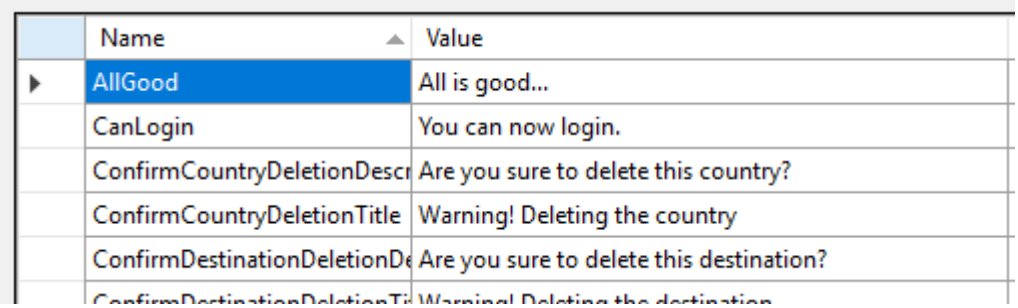
První část, Classes, obsahuje funkce pro práci s databází. Každý panel aplikace má třídu se svými funkcemi a dotazy do databáze. Pro vykonávání a získávání výsledků z databáze je nejdůležitější třída DatabaseLayout.cs, která obsahuje funkce pro připojení k databázi přes MySQL ovladač, práci s databází a kontrolu dat. Pro další zpracování dat, konkrétně zpracování obrázků přes FTP, je vytvořena třída FTPLayout.cs, která definuje funkci k připojení a manipulaci/nahrávání/změna obrázků na lokálním Apache serveru.



Obrázek 20: Adresářová struktura projektu ve Visual Studiu

(zdroj: vlastní zpracování)

Druhá část, Resources, obsahuje jednotlivé jazykové texty/mutace (Value), které se na základě volání názvu proměnné (Name) navrátí a vypíšou na jednotlivých pozicích. Příkladem je zápis volání „Properties.Resources.ConfirmCountryDeletionDescription“, kdy řetězec proměnné (Name) „ConfirmCountryDeletionDescription“ navrátí hodnotu (Value) „Are you sure to delete this country?“ (viz obrázek 21).



	Name	Value
▶	AllGood	All is good...
	CanLogin	You can now login.
	ConfirmCountryDeletionDescr	Are you sure to delete this country?
	ConfirmCountryDeletionTitle	Warning! Deleting the country
	ConfirmDestinationDeletionDe	Are you sure to delete this destination?

Obrázek 21: Nastavení překladů přes Resources

(zdroj: vlastní zpracování)

Třetí část, Model, definuje pozici a umístění jednotlivých panelů a umístění jednotlivých grafických prvků – tlačítka, pole, popisky apod. Pro lepší přehlednost souborů jsou jednotlivé panely rozděleny do dvou částí – BackEnd a FrontEnd. Panely ve složce FrontEnd obsahují kódy pro výchozí stránky bez přihlášení – tj. pouze hlavní stránka, která se načte po spuštění, a stránka s kontaktními údaji. Panely ve složce BackEnd obsahují kódy pro stránky, které jsou dostupné až po úspěšném přihlášení – tj. přidávání/úpravy/mazání zemí, destinací, restaurací, POI apod.

Všechny stránky dědí vlastnosti z hlavního formuláře (DefaultScreen.cs), kde je definováno umístění a funkce hlavního menu a patičky s logy a časovými údaji. Tyto stránky jsou realizovány jako panely (User Control) a reakce na jednotlivé provedené akce je čistě přepínání těchto panelů (viz obrázek 22) a předávání informací přes tzv. delegáty.

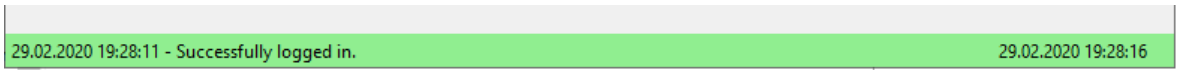
```
/// <summary>
/// Function shows edit destination page
/// </summary>
private void ShowEditDestinationsPage()
{
    if (checkLoggedIn() == true)
    {
        if (!Panel.Controls.Contains(DestinationsEditPage.Instance))
        {
            Panel.Controls.Add(DestinationsEditPage.Instance);
            DestinationsEditPage.Instance.Dock = DockStyle.Fill;
            DestinationsEditPage.Instance.BringToFront();
        }
        else
            DestinationsEditPage.Instance.BringToFront();
    }
}
```

Obrázek 22: Přepínání panelů (User Control)

(zdroj: vlastní zpracování)

Výše zmíněný hlavní formulář při přepínání panelů nastavuje danému panelu výšku a šířku, následně umisťuje pozici nadpisu tak, aby byl vždy na stejném místě, a v neposlední řadě i tlačítka pro provedení akce uložení a tlačítka zpět.

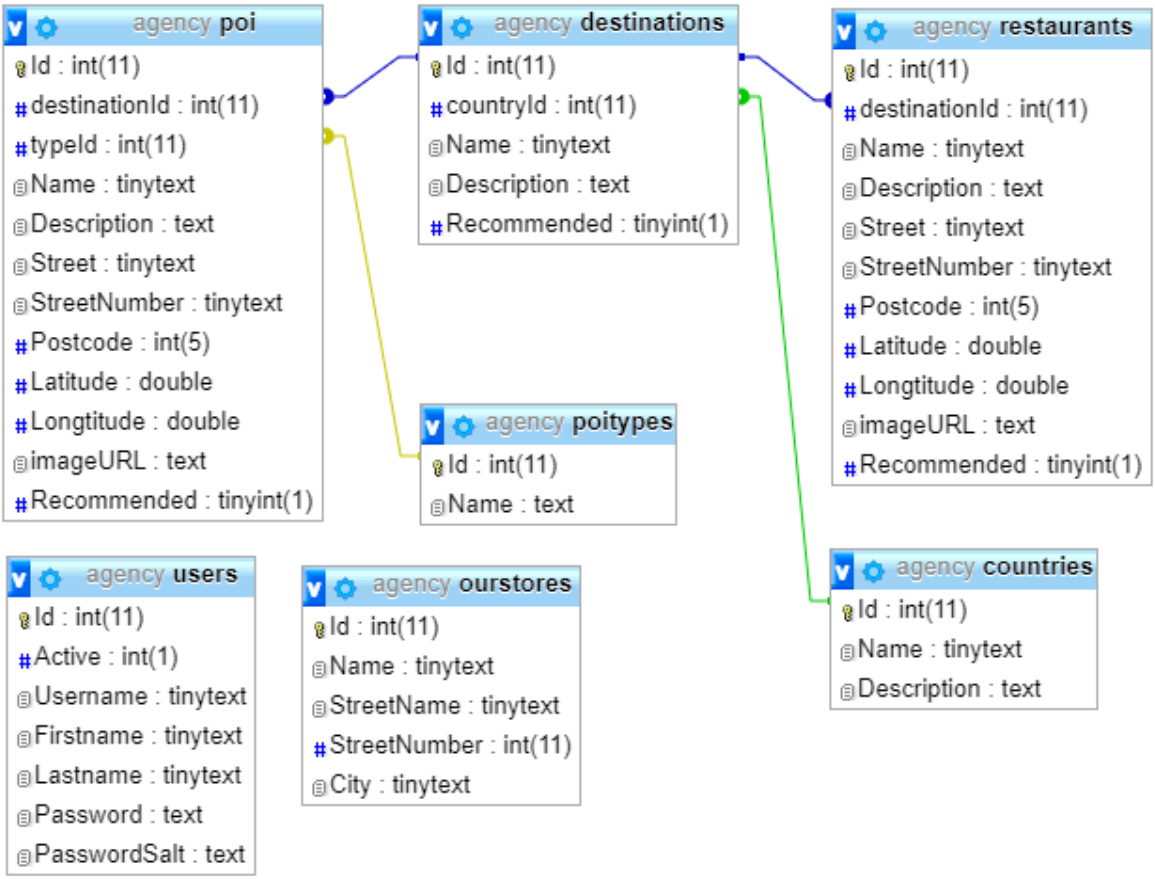
V případě, že dojde při běhu aplikace k problémům jak bezpečnostním (špatné přihlášení), tak auditovým (úprava informací určitých uživatelem), je jejich záznam vypsán ve stavové liště patičky hlavního formuláře (viz obrázek 23), a zároveň je zapsán na lokální disk do souboru logs.txt.



Obrázek 23: Stavová lišta back-end aplikace
(zdroj: vlastní zpracování)

4.3.2 Návrh databázové struktury

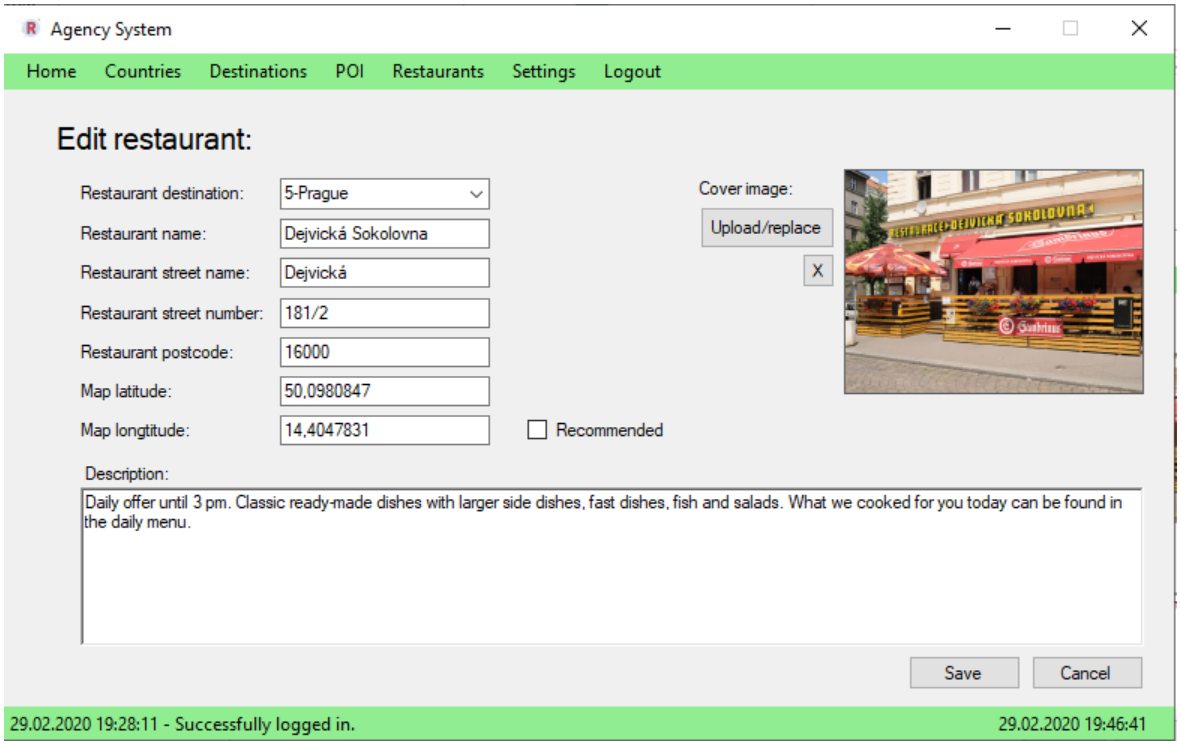
Před vytvořením databáze jako takové bylo ručně načrtnuto orientační schéma, ze kterého se postupně odvozovaly jednotlivé tabulky. Následně se metodou brainstormingu vymýšlelo, co všechno bude možné v aplikaci provádět, byl zohledněn budoucí možný vývoj (hlavně webové prezentace a mobilní aplikace) a nakonec vytvořena databáze v základním rozložení. Základním rozložením je myšleno, že obsahuje tabulky čistě nutné pro práci s daty, nikoli tabulky pro logování akcí a úprav, tabulky pro reporting a další možné budoucí sociální indikátory (počet zobrazení apod.). Databáze včetně vazeb a klíčů mezi tabulkami byla vytvořena postupem shora dolů a splňuje normální formy třetí úrovně. Diagram struktury a jednotlivých vazeb je znázorněný na obrázku 24.



Obrázek 24: Diagram struktury databáze a vazeb mezi tabulkami
(zdroj: vlastní zpracování)

4.3.3 Návrh rozložení prvků

Vzhled back-end aplikace byl zpracován dle zkušenosti autora práce v oboru interakčního designu (viz obrázek 25). Jednotlivé prvky a akce byly navrženy tak, aby byly pro uživatele přirozené a svým umístěním uživatele vedly k požadované akci. Před samotným programováním a převáděním designu do kódu by standardně mělo být provedeno testování cílovou skupinou, ale tento krok nebyl součástí této diplomové práce. Zpracovaná UI (User Interface) specifikace je přiložena v příloze č. 2.



Obrázek 25: Vzhled back-end stránky pro úpravu restaurace

(zdroj: vlastní zpracování)

4.3.4 Postup prací a milníky

Vytváření back-end části by se dalo rozdělit do pěti hlavních milníků. Prvním bylo stanovení hlavních požadavků (cílů) aplikace, tj. co má aplikace umět, jaký problém řeší, shrnutí možného budoucího vývoje tak, aby se všechny možné cíle specifikovaly ihned na začátku (případné budoucí změny hlavních funkcí by mohly následně zapříčinit zpoždění ve vývoji a další finanční náklady). Z jednotlivých požadavků se vytvořila struktura databáze, respektive se odvozovaly jednotlivé tabulky a jejich vazby. Po návrhu databáze byly struktury a vazby otestovány standardy a normálními formami, po jejichž splnění byl splněn druhý milník – funkční databáze.

Třetím milníkem bylo vyvinutí aplikačního základu, respektive vytvoření základů aplikace a naprogramování funkčních tzv. core funkcí, které jsou využívány všemi částmi aplikace a musí být plně odladěné pro stabilní běh aplikace. Mezi core funkce patří například práce s databází, funkce hlavního menu, stavového řádku, přepínání panelů, technika vykreslování apod. Následně byly vytvořeny jednotlivé stránky (panely), na kterých bylo vytvořeno plně

funkční rozhraní pro daný účel (například přidání země, smazání destinace, úprava destinace, přihlášení apod.). Jakmile byla celá funkční část otestována, předposledním milníkem bylo nasadit prototyp zpracovaný v přiložené UI specifikaci na funkční vrstvu, a vytvořit tím kompletní aplikaci. Posledním pátým milníkem bylo plné otestování funkčnosti, testování náročnosti aplikace na výpočetní výkon, využití dočasné lokální paměti pro práci, bezpečnostní verifikace a další parametry.

4.4 Vývoj webové prezentace

4.4.1 Návrh objektového modelu

Webová prezentace slouží pouze k zobrazování dat z databáze klientům. Dokumentový objektový model je vytvořen značkovacím jazykem HTML verze 5, kterému se pomocí kaskádových stylů CSS aplikují grafické prvky a pomocí skriptovacího jazyka JavaScript, konkrétně javascriptové knihovny jQuery, jsou vytvořeny animace a akce na webové stránce. Pro zobrazování dat z databáze je naprogramován objektovým programovacím jazykem PHP funkční model, který navíc využívá strukturovaný dotazovací jazyk SQL pro vytváření dotazů do databáze. Z výše popsaného je datový objekt rozdělen na tři části – první částí je objektový model HTML, druhou částí je design a vzhled stránky a třetí částí je napojení webové stránky na databázi.

První část, objektový model, je vytvořen značkovacím jazykem HTML verze 5, kdy je využíváno novodobých standardů pro tvorbu webových prezentací dle best-practice autora práce. Struktura objektového modelu z pohledu velikostí rozložení a šablony využívá webový framework Bootstrap, který tvoří základní funkce responzivity. Bootstrap je plně kompatibilní s HTML 5 a upravuje nové sémantické prvky header, footer, article a section (viz obrázek 26).

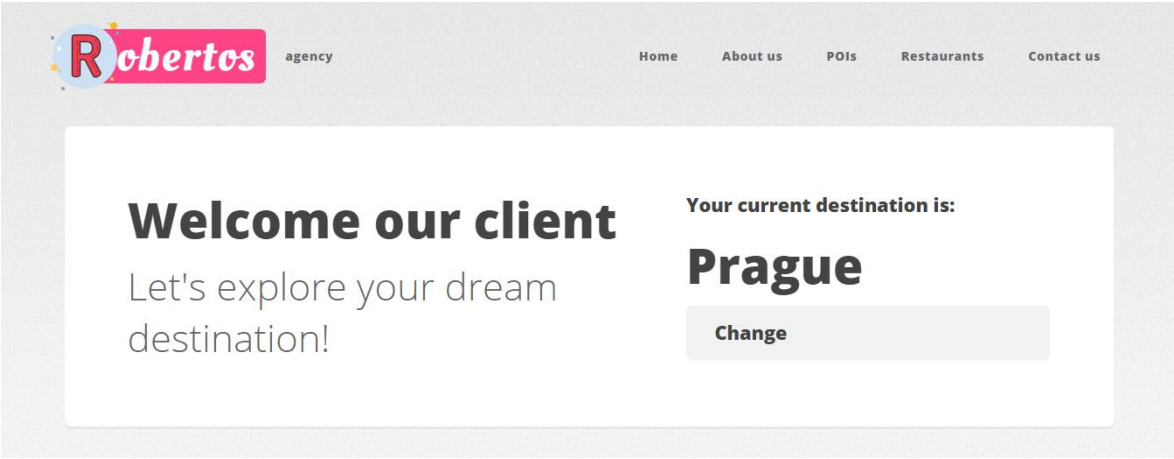
Obrázek 26: HTML struktura webové prezentace

(zdroj: vlastní zpracování)

Webová stránka je plně responzivní, tj. obsah stránky se přizpůsobuje šířce a výšce zobrazovacího okna tak, aby byl vždy správně čitelný bez narušení tzv. přetékání zobrazovacího okna. Responzivita byla verifikována na stolním počítači se zobrazovacím oknem 1920 x 1080, na notebooku se zobrazovacím oknem 1366 x 768, na iPadu se zobrazovacím oknem 1024 x 768 a na chytrém telefonu se zobrazovacím oknem 1080 x 2280. Dále byla responzivita a kompatibilita testována na novodobých prohlížečích

– Google Chrome, Microsoft Edge, Internet Explorer a Mozilla Firefox – kde se prováděla emulovaná změna zobrazovacího okna až na šířku 300 a výšku 600. Výše zmíněnou kompatibilitou nebyla myšlena pouze verifikace správnosti zobrazení obsahu, ale také přizpůsobení stránek pro osoby s určitými zdravotními dysbalancemi, jako je například barvoslepost či slepost. Webová prezentace splňuje specifikace Blind Friendly Web 2.3. a WCAG 2.1.

Druhá část, grafický layout, je nasazena s využitím kaskádových stylů CSS a javascriptových akcí pomocí knihovny jQuery. Všechny elementy v objektovém modelu obsahují atributy class nebo id, na které se z kaskádových stylů aplikují určité vlastnosti (není využit atribut style, který není již vhodné používat). Základní layout a rozložení stránek byly nejprve načrtnuty na papír (tzv. wireframe), čímž se definovala struktura objektového modelu a následně se při vývoji dolaďovala pomocí barev a stylů dle best-practice autora. Layout úvodní stránky je zobrazen níže na obrázku č. 27.



Obrázek 27: Layout úvodní stránky webové prezentace
(zdroj: vlastní zpracování)

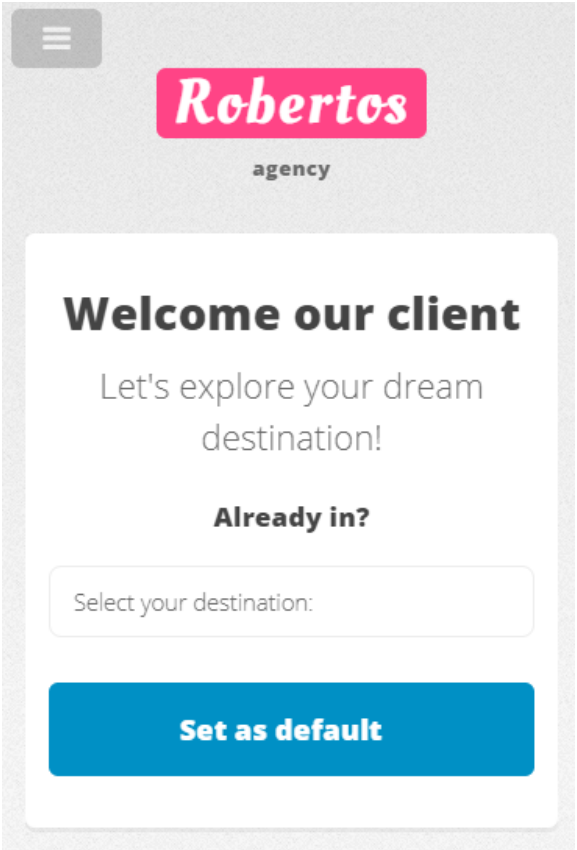
Mimo kaskádové styly frameworku bootstrap, které definují základní chování responzivní verze, obsahuje webová stránka hlavní kaskádový soubor se styly main.min.css, kde se definují základní textové, grafické a poziční elementy, ale také se přepisují výchozí styly specifikované v bootstrap knihovnách. V tomto souboru je také řešena responzivita nových elementů pomocí @media typu (viz obrázek 28), který nastavuje určité vlastnosti na základě velikosti zobrazovacího okna.

```css
@media screen and (max-width: 1280px) {
    body {
        font-size: 11pt;
    }
    section, article {
        margin-bottom: 3em !important;
    }
}
```

Obrázek 28: CSS specifikace @media typu

(zdroj: vlastní zpracování)

CSS, spolu s jQuery javascriptovou knihovnou, slouží k zobrazení responzivního menu v případě zmenšení rozlišení pod 980px – to znamená, že namísto standardního menu v řádku se zobrazí ikonka tzv. burger menu, ve které je navigace schovaná. Kliknutím na ikonku dojde k zobrazení menu vyjetím z levé strany. Od výše zmíněné šířky 980px se všechny elementy upravují na verzi pro tablety a mobilní telefony (viz obrázek 29).



Obrázek 29: Mobilní verze webové aplikace

(zdroj: vlastní zpracování)

Třetí část, práce s databází, sestává z naprogramovaného rozhraní v jazyku PHP pro komunikaci s databázovým rozhraním a získávání výsledků kladením SQL dotazů. Jednotlivé předávané parametry na stránce jsou důkladně ověřovány (viz obrázek 30), aby nedošlo k útokům známým jako SQL Injection, kdy se do parametrů podsune určitý řetězec, který změní výsledný dotaz do databáze.

```php
public function quote($value) {
    $connection = $this->connect();
    return "'" . $connection->real_escape_string($value) . "'";
}
```

Obrázek 30: Ochrana získaných parametrů
(zdroj: vlastní zpracování)

Příklad tohoto útoku je zobrazen na obrázku č. 31, kde se do pole vložil řetězec „105 OR 1=1“, a tím se změnil dotaz do databáze, která by namísto jednoho uživatele vypsala všechny uživatele (tzv. dump celé tabulky Users).

UserId: 105 OR 1=1

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;

SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Obrázek 31: Příklad ukázky útoku SQL Injection
(zdroj: SQL Injection, b.r.)

4.4.2 Postup prací

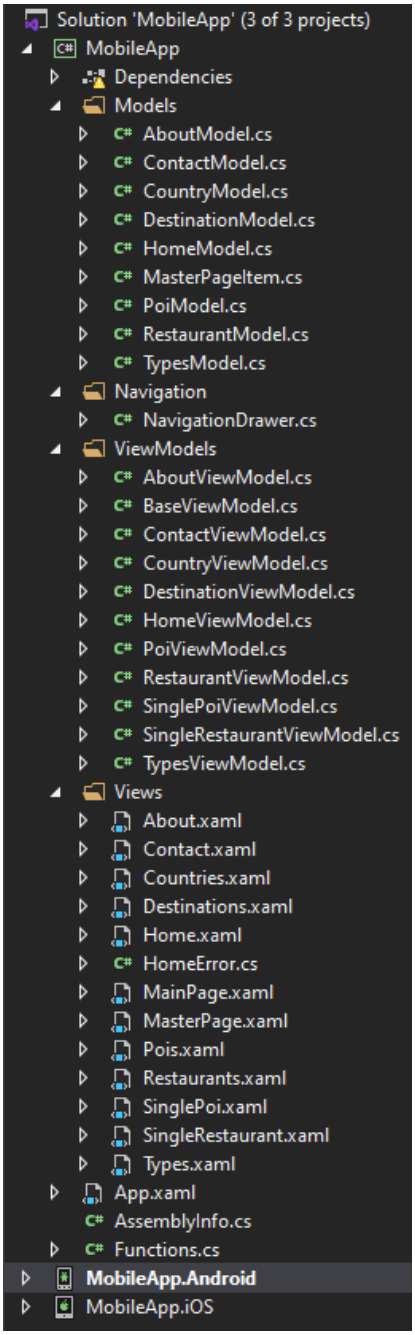
S ohledem na celkový vývoj systému byla webová prezentace vytvořena jako druhá v pořadí, ihned po vytvoření back-end aplikace pro správu dat. Důvodem bylo prvotní vytvoření databáze a aplikace, přes kterou se daná data mohou nejprve i upravovat a až následně zobrazovat. Jelikož se po vyvinutí back-end části již vědělo, jaké informace jsou uchovávány, byl navržen wireframe webové prezentace s využitím zásad interakčního designu, na jehož základě byl vytvořen datový objektový model stránky. Následně bylo vytvořeno rozhraní pro získávání dat z databáze a jejich vypisování v modelu. Jakmile byla ověřena plná funkčnost stránky, byly doladěny poslední grafické prvky a začalo se

s testováním. Stránka byla testována jak z pohledu zabezpečení neoprávněného přístupu do databáze, tak z pohledu optimalizace výkonu a responzivity. Jako poslední krok bylo provedení SEO analýzy a optimalizace dle novodobých standardů tak, aby se stránky při nasazení správně a rychle zaindexovaly do webových algoritmů a byly následně nabízeny jako výsledky hledání dalším potencionálním klientům v síti Internet.

4.5 Vývoj mobilní aplikace

4.5.1 Návrh objektového modelu

Objektový model mobilní aplikace využívá návrhový vzor MVVM (Model – View – ViewModel), který odděluje logiku aplikace (data a stav aplikace) od uživatelského rozhraní (viz obrázek 32).



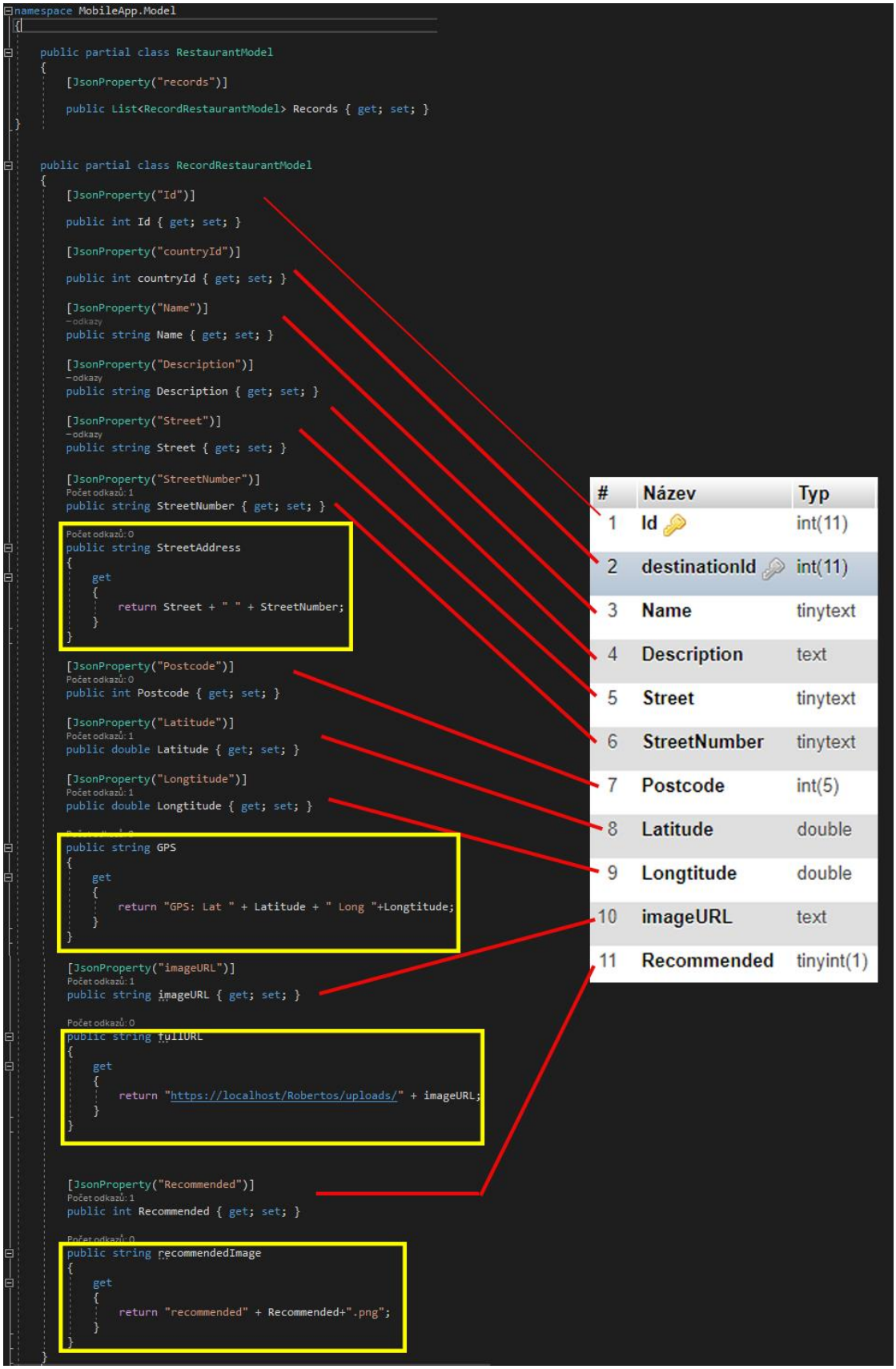
Obrázek 32: Hierarchická struktura kódu mobilní aplikace

(zdroj: vlastní zpracování)

Hierarchická struktura je tedy rozdělena na tři části:

- Models – popisuje data, se kterými aplikace pracuje (mapuje třídy na tabulky databáze),
- Views – reprezentují prezentační vrstvu aplikace (uživatelské rozhraní v jazyce XAML),
- ViewModels – spojuje Model a View (drží si stav aplikace).

První část, Models, obsahuje všechny datové třídy, které mají stejné vlastnosti a minimálně stejný počet objektů jako patřičná tabulka sloupců v databázi. Praktickým příkladem je obrázek č. 33 – datová třída RestaurantModel.cs (v levé části) obsahuje patnáct deklarovaných objektů (jedenáct namapovaných objektů spojených červenou čarou s databázovými sloupci a čtyři separátní funkce označených žlutými obdélníky vracející složené tvary objektů) – na pravé straně je tabulka restaurants z databáze, která obsahuje 11 identických sloupců stejných vlastností (spojené červenými čarami s objekty aplikace).



Obrázek 33: Namapované objekty C# a sloupce tabulky databáze

(zdroj: vlastní zpracování)

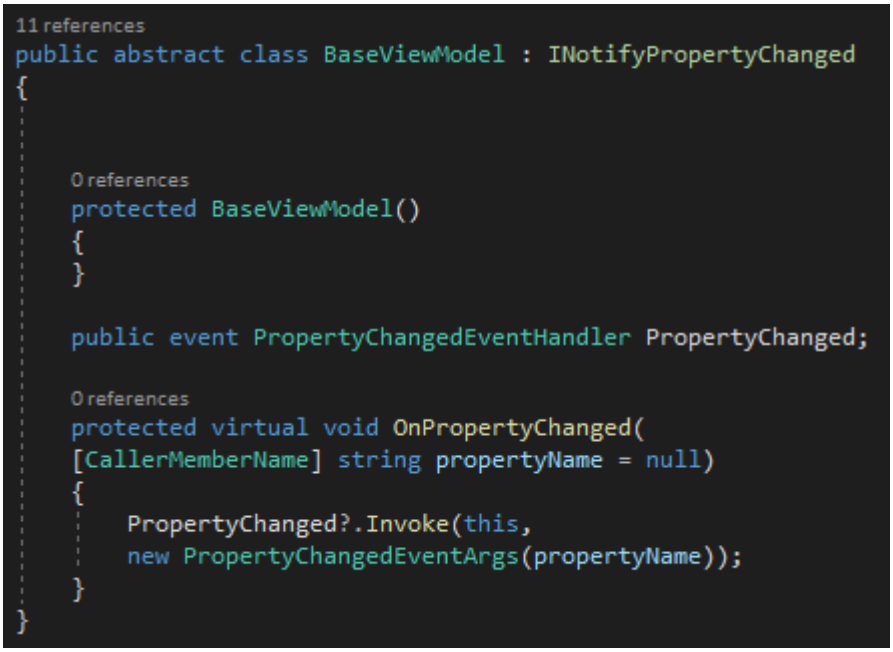
Druhá část, Views, reprezentuje uživatelské rozhraní v jazyce XAML. Jedná se o jednotlivá okna/stránky/ovládací prvky, které se zobrazují koncovému uživateli. Každá stránka má svůj obsahový soubor, který definuje rozložení prvků (mřížka, plovoucí objekty, seznamy apod.), ve kterém jsou jednotlivé texty a hodnoty reprezentovány přes napojení (binding) na ViewModel (viz obrázek 34).

```
<ListView x:Name="ListViewRestaurants">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="50"/>
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="50"/>
                    </Grid.ColumnDefinitions>

                    <Image Source="restaurant.png"  WidthRequest="50"
                           HeightRequest="50" Grid.Column="0"/>
                    <Button Text="{Binding Name}"
                            Clicked="RestaurantButtonClicked"
                               Command="{Binding ClickCommand}"
                            CommandParameter="{Binding Id}" Grid.Column="1" />
                    <Image Source="{Binding recommendedImage}"
                           WidthRequest="50" HeightRequest="50" Grid.Column="2"/>
                </Grid>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
```

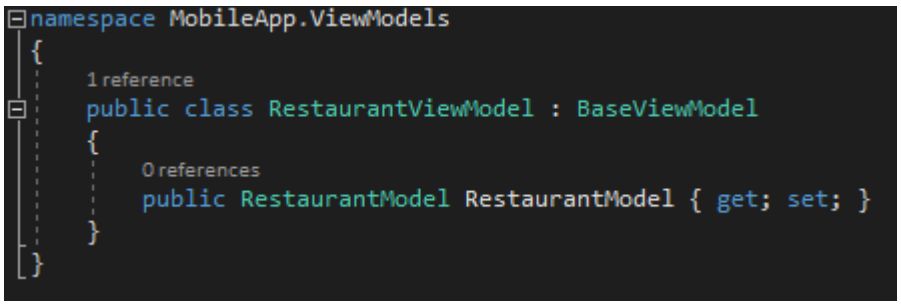
Obrázek 34: Definovaná struktura mobilní stránky seznamu restaurací
(zdroj: vlastní zpracování)

Třetí část, ViewModels, obsahuje jednotlivé třídy k objektovým modelům a propojuje pomocí bidingu ovládací prvky (ovládací prvky z ní čerpají svůj obsah). ViewModels třídy provádějí filtrovaní dat v závislosti na stavu aplikace, nicméně aby se změny a vlastnosti propagovaly až do uživatelského rozhraní (Views), je v základní třídě BaseViewModel.cs naimplementováno rozhraní INotifyPropertyChanged (viz obrázek 35), přes které jsou dědičností jednotlivé modely informovány o dané změně hodnoty vlastnosti (viz obrázek 36).



Obrázek 35: Implementace rozhraní INotifyPropertyChanged

(zdroj: vlastní zpracování)



Obrázek 36: Dědičnost rozhraní přes třídu BaseViewModel

(zdroj: vlastní zpracování)

4.5.2 Návrh databázového napojení

Pro zachování jedinečnosti využívá mobilní aplikace stejnou databázi jako back-end aplikace a webová prezentace. Před vývojem byla původní databáze analyzována a s ohledem na možné nové požadavky v mobilní aplikaci upravena. Po upravení databáze bylo nutné vyřešit problém napojení mobilní aplikace na databázi. Narozdíl od back-end aplikace nelze využít modul MySQL Net Connector, jelikož není kompatibilní a vhodný pro použití v mobilních zařízeních, a ani nelze využití přímé příkazy SQL jako u webové prezentace, jelikož je jazyk C# ani platforma Xamarin nezná. Pro řešení tohoto problému

jsem vybral vyvinutí aplikačního rozhraní (API), které slouží pro výměnu dat ve formátu JSON přes HTTP protokol (viz následující kapitola 4.5.3.).

Při načítání stránky v mobilní aplikaci, která vyžaduje určitá data z databáze, je nejprve z prezentační vrstvy volána funkce DataAPI s požadovaným parametrem (viz obrázek 37), na jehož základě se vybere API URL adresa pro získání JSON výsledku (viz obrázek 38). Výsledek je následně na stránce převeden (deserializován) zpět do standardní struktury.

```
if (RefDestinationId != null && RefDestinationId !="" )
{
    returnArray = (await client.GetAsync(
        Functions._DataAPI("AllRestaurantsByDestinationID"))).Content.ReadAsStringAsync().Result;
}
else
{
    returnArray = (await client.GetAsync(
        Functions._DataAPI("AllRestaurants"))).Content.ReadAsStringAsync().Result;
}
RestaurantModel resultParse = new RestaurantModel();
if (returnArray != "")
{
    resultParse = JsonConvert.DeserializeObject<RestaurantModel>(returnArray);
}

ListViewRestaurants.ItemsSource = resultParse.Records;
```

Obrázek 37: Volání funkce DataAPI a zpracování výsledku

(zdroj: vlastní zpracování)

```
public static string _DataAPI(string parameter)
{
    string RefCountryId = App._RefCountryId;
    string RefDestinationId = App._RefDestinationId;
    string RefRestaurantId = App._RefRestaurantId;
    string RefPoiId = App._RefPoiId;

    string API = "";
        switch (parameter)
        {
            case "OneCountry":
                API = "https://localhost/Robertos/api/country/read_one.php?Id=" + RefCountryId;
            case "AllCountries":
                API = "https://localhost/Robertos/api/country/read.php"; break;

            case "OneDestination":
                API = "https://localhost/Robertos/api/destination/read_one.php?Id=" + RefCountr
            case "AllDestinations":
                API = "https://localhost/Robertos/api/destination/read.php"; break;
            case "AllDestinationsByCountryID":
                API = "https://localhost/Robertos/api/destination/read_byCountryId.php?countryI

            case "OneRestaurant":
                API = "https://localhost/Robertos/api/restaurant/read_one.php?Id=" + RefRestaur
            case "AllRestaurants":
                API = "https://localhost/Robertos/api/restaurant/read.php"; break;
            case "AllRestaurantsByDestinationID":
                API = "https://localhost/Robertos/api/restaurant/read_byDestinationId.php?desti

            case "OnePoi":
                API = "https://localhost/Robertos/api/poi/read_one.php?Id=" + RefPoiId; break;
            case "AllPois":
                API = "https://localhost/Robertos/api/poi/read.php"; break;
            case "AllPoisByDestinationID":
                API = "https://localhost/Robertos/api/poi/read_byDestinationId.php?destinationI
    }
```

Obrázek 38: Realizace funkce DataAPI

(zdroj: vlastní zpracování)

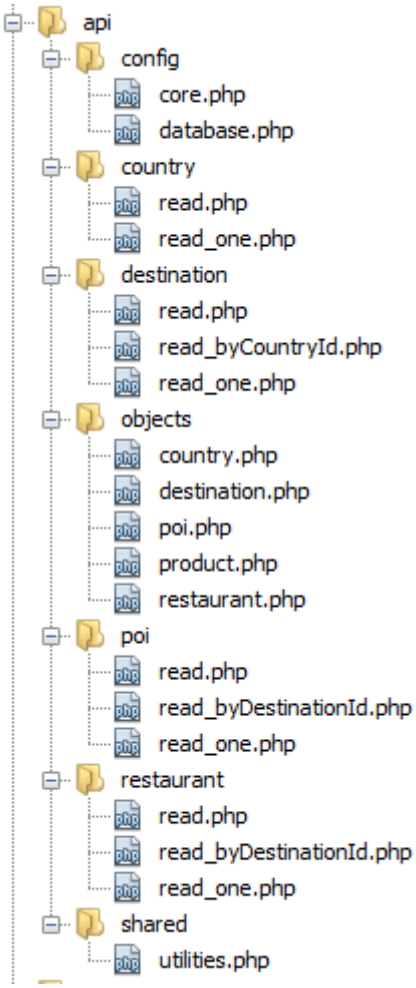
4.5.3 Návrh API rozhraní

Aplikační rozhraní (API) bylo vytvořeno jako komunikační kanál mezi mobilní aplikací a databází. API využívá pro přenos dat JSON zápis, který umožňuje přenášet data v polích a objektech. Samotné rozhraní bylo naprogramováno jazykem PHP a je součástí webového serveru Apache, na kterém běží webová prezentace.

Hierarchická struktura je rozdělena na dvě části (viz obrázek 39):

- Objekty (objects) – reprezentují PHP třídy pro jednotlivé objekty, které obsahují vlastnosti a metody a přes které se získávají data z databáze,

- Typy (restaurant, country, ...) – reprezentují PHP soubory využívající tříd objektů pro zpracování výstupu JSON a navrácení hodnot protokolem HTTP přes definovanou URL adresu.



Obrázek 39: Hierarchická struktura kódů API rozhraní
(zdroj: vlastní zpracování)

V první části, Objects, jsou definovány objekty dané tabulky (kolik má tabulka sloupců, tolik je vytvořených objektů) a jednotlivé funkce pro získání dat z databáze. Každá funkce obsahuje daný SQL dotaz, do kterého se případně dosazují parametry a který se následně předává PHP databázovým funkcím k zpracování (viz obrázek 40).

```php
public function readByDestinationId() {

    $query = "SELECT
                Id, destinationId, Name, Description, Street, StreetNumber,
                Postcode, Latitude, Longtitude, imageURL, Recommended
            FROM
                " . $this->table_name . "
        WHERE
            destinationId = ?
        ORDER BY
            Name";

    $stmt = $this->conn->prepare($query);

    $stmt->bindParam(1, $this->destinationId);

    $stmt->execute();

    return $stmt;
}
```

Obrázek 40: Funkce API pro získání destinace dle parametru ID

(zdroj: vlastní zpracování)

V druhé části jsou vytvořeny typy dle počtu tabulek, které se využívají, a v rámci jednoho typu jsou definovány návratové funkce, které jsou nakonfigurovány. Právě jednotlivé typy jsou volány externě a ty se starají o kompletní zpracování požadavku (viz obrázek 41) v následující sekvenci:

- nastavení správné HTTP hlavičky,
- napojení na databázi,
- získání parametrů pro filtrování,
- volání funkce pro zpracování dotazu,
- zpracování odpovědi a zakódování do formátu JSON,
- vypsání/navrácení řetězce.

```php
<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Allow-Credentials: true");
header('Content-Type: application/json');

include_once '../config/database.php';
include_once '../objects/destination.php';

$database = new Database();
$db = $database->getConnection();

$Destination = new Destination($db);

$Destination->Id = isset($_GET['Id']) ? $_GET['Id'] : die();

$Destination->readOne();

if($Destination->Name!=null){
    // create array
    $Destination_arr = array(
        "Id" =>  $Destination->Id,
        "countryId" =>  $Destination->countryId,
        "Name" => $Destination->Name,
        "Description" => $Destination->Description,
        "Recommended" => $Destination->Recommended
    );

    http_response_code(200);
    echo json_encode($Destination_arr);
}
else{
    http_response_code(404);
    echo json_encode(array("message" => "Destination does not exist."));
}
?>
```

Obrázek 41: Zpracování požadavku a navrácení řetězce JSON

(zdroj: vlastní zpracování)

Po zavolání výše uvedené funkce read_one.php s parametrem Id=5 (celá adresa níže):

https://localhost/Robertos/api/destination/read_one.php?Id=5

navrátí přes HTTP daný JSON řetězec – na obrázku 42 je zobrazen čistý výstup z prohlížeče a na obrázku 43 je zobrazen navrácený JSON řetězec v uživateli přívětivější formě aplikací Postman, která má skvělé využití v případě delších a složitějších JSON řetězců. Právě navrácený řetězec se následně v kódu mobilní aplikace dekóduje zpět z JSON formátu a využívá se jako pole, které se dosazuje do jednotlivých pohledů (Views).

```
{"Id":"5","countryId":"2","Name":"Prague","Description":"Prague is
the capital and largest city of the Czech Republic. It is the 14th
largest city in the European Union. It is also the historical
capital of Bohemia. Situated in the north-west of the country on
the Vltava river, the city is home to about 1.4 million people,
while its larger urban zone is estimated to have a population of
2.2 million. The city has a temperate climate, with warm summers
and chilly winters.","Recommended":"1"}
```


Obrázek 42: Navrácený JSON řetězec v prohlížeči

(zdroj: vlastní zpracování)


```
{
    "Id": "5",
    "countryId": "2",
    "Name": "Prague",
    "Description": "Prague is the capital and largest city of the Czech
        Republic. It is the 14th largest city in the European Union. It
        is also the historical capital of Bohemia. Situated in the
        north-west of the country on the Vltava river, the city is home
        to about 1.4 million people, while its larger urban zone is
        estimated to have a population of 2.2 million. The city has a
        temperate climate, with warm summers and chilly winters.",
    "Recommended": "1"
}
```

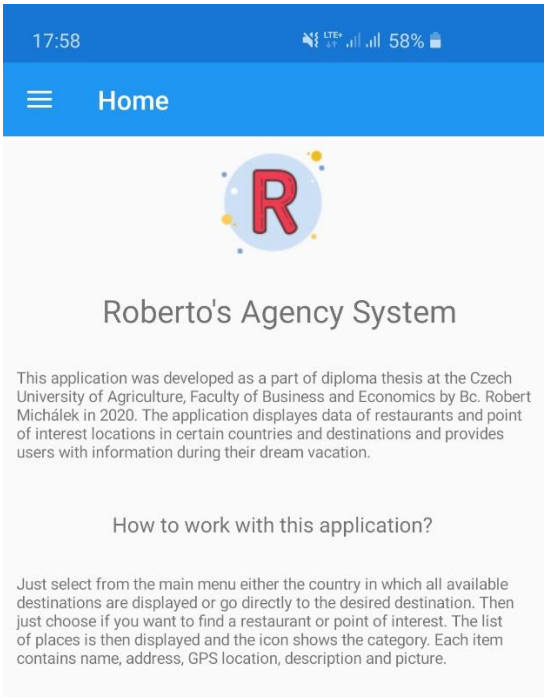

Obrázek 43: Navrácený JSON řetězec aplikací Postman

(zdroj: vlastní zpracování v aplikaci Postman)

4.5.4 Návrh rozložení aplikace

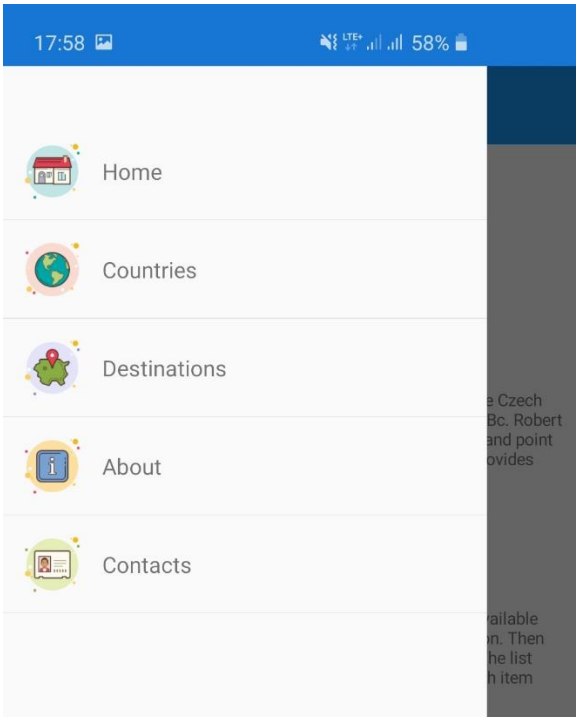
Mobilní aplikace vychází z datového modelu webové prezentace, kde již bylo definováno, jaké informace se zobrazí klientům, a tudíž byl layout mobilní aplikace téměř přímo odvozen. Pro stručnost byl před kódováním vytvořen wireframe, který obsahoval základní rozložení menu, obsah menu, zobrazení jednotlivých kategorií a rozdělení prvků na detailech položky.

Na obrázku 44 je zobrazena úvodní strana mobilní aplikace, na obrázku 45 je zobrazeno menu v mobilní aplikaci, na obrázku 46 je zobrazen seznam POI v dané destinaci a na obrázku 47 je zobrazena stránka s jednou restaurací.



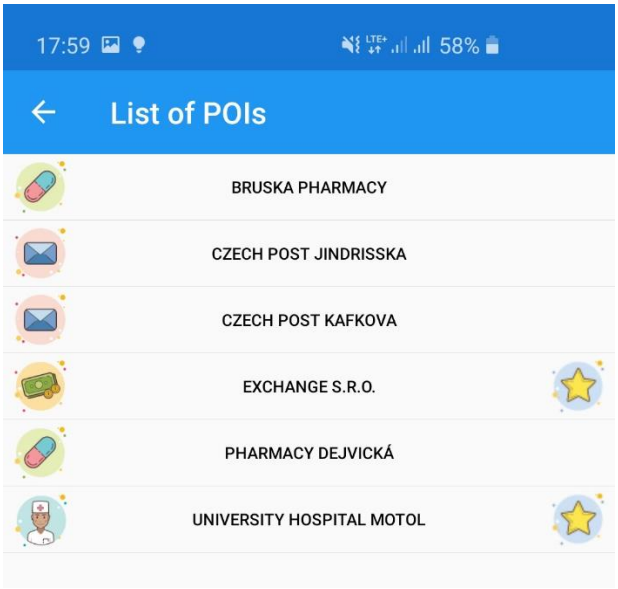
Obrázek 44: Mobilní aplikace – úvodní stránka

(zdroj: vlastní zpracování)



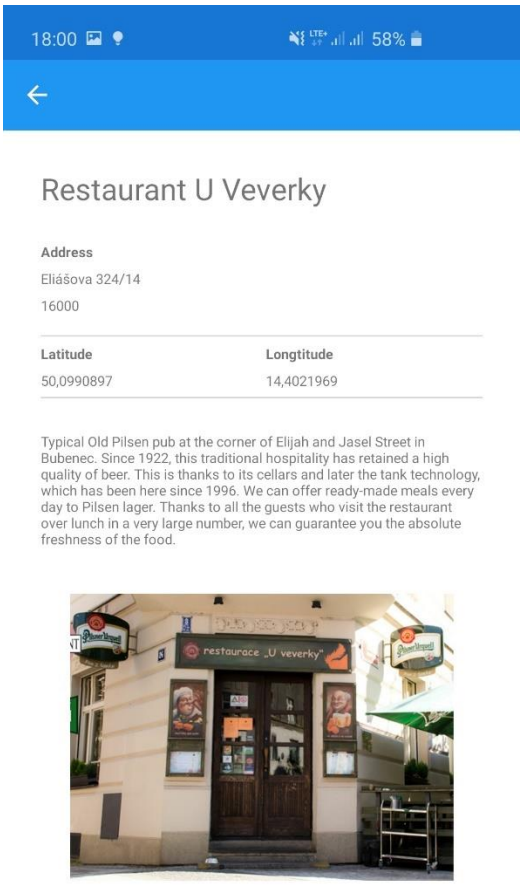
Obrázek 45: Mobilní aplikace – zobrazení menu

(zdroj: vlastní zpracování)



Obrázek 46: Mobilní aplikace – seznam POI

(zdroj: vlastní zpracování)



Obrázek 47: Mobilní aplikace – stránka restaurace

(zdroj: vlastní zpracování)

4.5.5 Postup prací

Prvním krokem před vývojem mobilní aplikace bylo vytvoření wireframe layoutu na základě stávající webové prezentace, ze kterého se odvodilo, jaká data bude nutné do aplikace přenášet. Po identifikaci a stanovení dat bylo vytvořeno API rozhraní, které získává data z databáze ve formátu JSON vhodném k přenosu. Jakmile bylo API rozhraní připraveno a otestováno, začal vývoj mobilní aplikace jako takové – nejprve byly vytvořeny věci jádra aplikace, konkrétně umístění a naprogramování menu, definování seznamů a pozic, hlavní stránka a způsob přepínání stránek. Jako první se naprogramovaly stránky základní „About us“ a „Kontakt“, které nejsou napojené na databázi a texty se řeší v rámci ViewModelu. Následně se vytvořily stránky s elementárním napojením na databázi, tj. země a destinace, jelikož jsou stejných typů a jedná se o jednodušší přiřazení řetězce JSON do Xamarin seznamu. Poté se programovaly restaurace, které jsou také stejných typů, ale již obsahují více dat včetně složených objektů – bylo specifikováno pozicování prvků na stránce a čtení/zobrazení obrázků přímo z webové prezentace. Po otestování funkčnosti a stanovení definic pro vzhled se naprogramovala poslední část – seznam POI, kde bylo potřeba pracovat s různými typy a těm přiřadit rozdílné ikony. Jako poslední byla naprogramována jednotlivá zobrazení POI stejným stylem jako stránka s restaurací.

Po otestování funkčnosti byla mobilní aplikace revidována tak, aby neobsahovala zbytečný kód, který by zpomaloval běh aplikace a zároveň byly testovány a verifikovány elementární funkce, jako je zachycení chyby při čtení z API z důvodu neexistujícího připojení k síti Internet – v tomto případě vypsání chybové hlášky a zastavení zpracování.

4.6 Testování

4.6.1 Aplikace

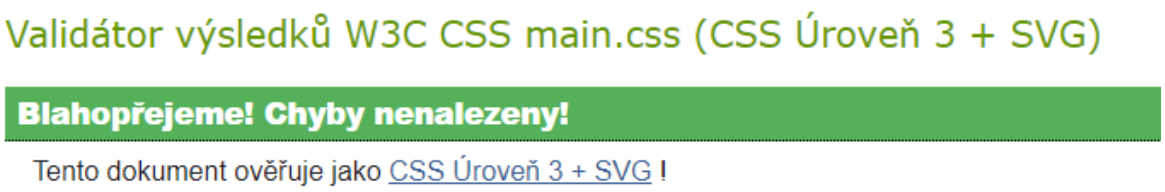
Back-end administrační část byla testována z mnoha úhlů. Byly testovány všechny vstupní parametry, jejich změna na hodnotu, která má být ošetřena, test uložení dat do databáze, které obsahovaly nestandardní znaky, zadání jiné hodnoty, než bylo typově specifikováno a dalších testů. Okrajově bylo také využit zabudovaný testovací nástroj Unit Testing, který má nadefinované určité testovací scénáře a ty se po změně v aplikaci mohou provést

najednou. Jedná se o testy kritických funkcí, které nesmí být ovlivněny jinými změnami kódu.

4.6.2 Webová prezentace

Oproti back-end části, kde byl kladený důraz na funkcionalitu a zabezpečení dat a funkcí, u webové prezentace je důraz kladen převážně na vzhled. Webovou stránku navštěvují stávající či budoucí klienti a je nutné zaujmout klienta na první pohled – například rozhozený formát či nečitelné texty jsou neomluvitelné a ihned odrazují klienta. Stránky byly testovány na novodobých prohlížečích a na celé řadě rozlišeních od širokoúhlých monitorů až po mobilní zařízení.

Byla provedena SEO analýza zdrojového kódu a test on-page faktorů a mnohých doporučeních dle best-practice autora. Pro validitu byl použit lokálně nainstalovaný program WebSite Auditor verze 4.43.9, který nezjistil závažnější chyby. Zdrojový kód a kaskádové styly CSS byly také podroben testu v online validátoru W3C, který potvrdil správného nastavení (viz obrázek 48).



Obrázek 48: Validní CSS kód
(zdroj: vlastní zpracování)

Uživatelská přívětivost, použitelnost a responzivita webové prezentace byla testována pomocí online nástroje Wave pro test zdrojového kódu, Color Contract Accessibility Validatoru pro test kontrastu barev na stránce dle standardu WCAG 2.1 AA, byla provedena ruční kontrola dle WCAG 2.1 a dle pravidel na Blind Friendly webu – žádné problémy nebyly nalezeny. Rychlost načítání stránek je průměrně do 2 sekund.

Přestože webová prezentace přímo neumožňuje ukládat data do databáze, byly všechny funkce pro práci s databází testovány, všechny proměnné a data jsou izolována, ošetřena a filtrována, aby se chránil kód před možným útokem SQL Injection.

4.6.3 Mobilní aplikace

Mobilní aplikace byla funkčně testována na mobilním zařízení Samsung J5 a Samsung S10e, které byly lokálně připojeny a spojeny s Visual Studio projektem přes knihovnu Android Emulator. Byla testována rychlost odezvy a stahování obrázků – nebyly nalezeny problémy. Otestování na různých platformách nebylo provedeno, jelikož se jedná o izolovaný projekt běžící na lokálním notebooku, a tudíž není možné aplikace aktuálně sdílet s dalšími testery.

Aplikace byla taktéž například testována na spuštění bez připojení k internetu, čímž by mohlo dojít k zamrznutí aplikace a neočekávaným problémům se zpracováním, ale zachycení je správně nastaveno a aplikace ihned zobrazí uživateli stránku s informací, že aplikaci je možné používat pouze s řádným internetovým připojením.

5 Výsledky a diskuse

Vytváření robustních a komplexních informačních systémů a aplikací, kdy mezi sebou jednotlivé moduly určitým způsobem komunikují, vyžaduje zkušenosti s vedením projektů a znalosti výpočetních systémů, aby byla ohlídána všechna kritéria, správně naplánována posloupnost prací a dosažení úspěšného projektu a splnění cílů. Programování a vývoj dílčích modulů zlehčují a urychlují novodobé nástroje, jako jsou různé frameworky (bootstrap či Xamarin) a vývojová prostředí (Microsoft Visual Studio či NetBeans), které obsahují již předdefinovanou sadu standardizovaných prvků a struktur. Mimo jiné umožňují kontrolu použitého jazyka za chodu, nabízejí a popisují funkce a parametry, dokončují příkazy, odkazují na deklarace patřičných funkcí a dalších vlastností.

Výsledkem této diplomové práce je komplexní, více platformní systém, který může být realizován jako nadstavba stávajícího informačního systému firmy. Výhodou je, že celý systém není potřeba vyvíjet od začátku, u všech částí stačí upravit napojení na databázi, změnu základních databázových pravidel a spustit jej. Zmíněná databázová pravidla jsou standardizována a lehce modifikovatelná databázovými specialisty, kteří znají novou strukturu tabulek databáze. Zdrojové kódy jsou čisté, popsané a lze je znovu použít pro budoucí vývoj či úpravy – pro změnu designu není třeba měnit základní funkce aplikace, a naopak pro přidání funkcionality není třeba přepisovat a upravovat design fungující části.

Celý systém obsahuje plně funkční základní model s nejnutnějšími funkcemi, který poskytuje představu a ukázku plně funkčního systému, který lze následně po diskusi se zájemcem škálovat, přidávat nové funkce a dále rozvíjet, čímž se může navázat spolupráce možného budoucího vývoje a zájmu o služby vývojáře.

6 Závěr

Hlavním cílem práce bylo vytvoření funkčního systému, který sestává ze tří hlavních částí – back-end aplikace pro správu dat vytvořenou v programovacím jazyce C# s využitím .NET frameworku; front-end dynamické webové stránky zobrazující data z databáze vytvořenou pomocí značkovacího jazyka HTML a programovacího jazyka PHP a mobilní aplikace pro operační systém Android a iOS vytvořenou v programovacím jazyce C# s využitím frameworku Xamarin a rozhraním REST API. Dále byl systém otestován a demonstrován pro nasazení a implementaci do reálného provozu.

Jako první cíl bylo navržení logického rozložení celého systému, struktura dat, rozhraní, funkčnost a stanovení cílů a požadavků celého systému. Jako druhý cíl bylo stanoveno vytvoření programového rozhraní pro běh jednotlivých částí systému, vytvoření databáze a vytvoření back-end aplikace pro správu dat. Otestováním komunikace a funkčnosti byl splněn druhý cíl a následoval třetí cíl – zpracování a naprogramování webové stránky, která zobrazuje data z databáze uživatelům v síti Internet. Po nastavení, ověření a testování požadovaných vlastností stránky (responzivní design a zabezpečení) byl splněn třetí cíl, po kterém následovalo vytvoření a naprogramování mobilní aplikace pro operační systém Android a iOS, spolu s realizací programového rozhraní API pro komunikaci s databází. Výsledným pátým cílem bylo otestování a analýza funkčnosti systému jako celku, včetně shrnutí výsledků a možností budoucího vývoje a expanze.

Systém by se mohl rozvíjet hned několika směry. Prvním směrem by bylo rozšíření back-end části a databáze o další typy dat a informací, například doplnění o obchody, hřiště, parky apod., o možnost rozšíření aplikace o další specifická data firmy a možnosti spolupráce se třetími stranami pro správu dat.

Druhým směrem by bylo rozšíření mobilní aplikace o možnost stáhnutí potřebných dat definovaných uživatelem offline a integrace s GPS modulem tak, aby se nemuselo využívat na cestách mobilních dat a zároveň byla funkční aplikace s polohovatelnými údaji pro získání aktuální polohy a vzdálenosti od určitých bodů.

Stanovené cíle diplomové práce byly dosaženy, výsledkem je plně funkční systém, který sestává ze tří částí – interní back-end aplikace pro správu dat, online webová prezentace pro zobrazení dat v síti Internet a mobilní aplikace pro operační systémy Android a iOS.

7 Seznam použitých zdrojů

GILMORE, W.J., 2007. *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály.* Vyd. 1. [i.e. 2. vyd.]. Brno: Zoner Press. Encyklopedie webdesignera. ISBN 80-868-1553-6.

HERMES, D., 2015. *Xamarin mobile application development: cross-platform C# and Xamarin.Forms fundamentals*. New York, NY: Apress. ISBN 978-1-4842-0214-2.

KUBÍČEK, M., 2008. *Velký průvodce SEO: jak dosáhnout nejlepších pozic ve vyhledávačích*. Brno: Computer Press. ISBN 978-80-251-2195-5.

LOPEZ, A., 2016. Learning PHP 7. Birmingham: Packt Publishing Limited. ISBN 9781785883415.

MCKAY, E.N., 2013. *UI is communication: how to design intuitive, user centered interfaces by focusing on effective communication*. Boston: Elsevier, Morgan Kaufmann. ISBN 978-012-3969-804.

PETZOLD, CH., 2006. *Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]*. Brno: Computer Press. ISBN 80-251-1058-3.

PURDUM, J.J., 2007. *Beginning C# 3.0: an introduction to object oriented programming*. Indianapolis Wiley Pub. ISBN 978-0-470-26129-3.

SMITH, B., 2015. *Beginning JSON*. New York, NY: Apress. ISBN 9781484202036.

STRAUSS, D., 2016. *C# Programming Cookbook*. Birmingham: Packt Publishing Limited. ISBN 9781786463968.

ŠÍMA, F. a D. VILÍMEK, 2006. Microsoft Visual Studio .NET: praktické programování krok za krokem. Praha: Grada. Průvodce (Grada). ISBN 80-247-1418-3.

TAYLOR, A.G., 2013. *SQL for dummies. 8th edition. Hoboken, New Jersey: Wiley, John & Sons --For dummies*. ISBN 978-1-118-60796-1.

VOSTROVSKÝ, V., 2004. *Vytváření databází v ORACLE*. Praha: Česká zemědělská univerzita v Praze, Provozně ekonomická fakulta. ISBN 978-80-213-1191-6.

WELLING, L. a L. THOMSON, 2017. *Mistrovství PHP a MySQL*. Brno: Computer Press. ISBN 978-80-251-4892-1.

1. díl – Úvod do C# a .NET frameworku, 2018. In: Itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. [cit. 15. 2. 2020]. Dostupné z: https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-uvod-do-jazyka-a-dot-net-framework

Co je Apache Server | Adaptic, 2001. In: Tvorba webu | Adaptic [online]. [cit. 22. 1. 2020]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/apache-server/

Differences between MVC and MVP for Beginners – CodeProject, 2018. In: CodeProject – For those who code [online]. [cit. 19. 12. 2019]. Dostupné z: https://www.codeproject.com/Articles/288928/Differences-between-MVC-and-MVP-for-Beginners

Download .NET Framework | Free official downloads. .NET | Free. Cross-platform. Open Source. [online]. [cit. 10. 2. 2020]. Dostupné z: https://dotnet.microsoft.com/download/dotnet-framework

History of the Web – World Wide Web Foundation. *World Wide Web Foundation – Founded by Tim Berners-Lee, inventor of the Web, the World Wide Web Foundation empowers people to bring about positive change.* [online]. Copyright ©2008 [cit. 10. 3. 2020]. Dostupné z: https://webfoundation.org/about/vision/history-of-the-web/

HTML příručka, přehled HTML tagů. *Jak psát web, návod na html stránky* [online]. [cit. 8. 1. 2020]. Dostupné z: https://www.jakpsatweb.cz/html/

Introduction to HTML. *W3Schools Online Web Tutorials* [online]. [cit. 8. 1. 2020]. Dostupné z: https://www.w3schools.com/html/html_intro.asp

Notable Changes in HTML5 - dummies. *dummies - Learning Made Easy* [online]. [cit. 8. 1. 2020]. Dostupné z: https://www.dummies.com/web-design-development/html/notable-changes-in-html5/

MVVM: Model-View-ViewModel. *Největší český web zaměřený na .NET framework* [online]. Copyright © 2020 [cit. 8. 3. 2020]. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

PHP: News Archive - 2020. *PHP: Hypertext Preprocessor* [online]. Copyright © 2001 [cit. 29. 2. 2020]. Dostupné z: https://www.php.net/archive/2020.php#2020-02-20-3

StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share. *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share* [online]. Copyright © StatCounter 1999 [cit. 10. 3. 2020]. Dostupné z: https://gs.statcounter.com/

Usage Statistics and Market Share of PHP for Websites, March 2020. *W3Techs - extensive and reliable web technology surveys* [online]. Copyright © 2009 [cit. 8. 3. 2020]. Dostupné z: https://w3techs.com/technologies/details/pl-php

What is FTP (File Transfer Protocol)? A definition from WhatIs.com. Networking information, news and tips - SearchNetworking [online]. [cit. 6. 11. 2020]. Dostupné z: https://searchnetworking.techtarget.com/definition/File-Transfer-Protocol-FTP

What is Software Framework? - Definition from Techopedia. Techopedia - Where IT and Business Meet [online]. Copyright © 2020 Techopedia Inc. [cit. 05. 03. 2020]. Dostupné z: https://www.techopedia.com/definition/14384/software-framework

What is Structured Query Language (SQL) – Definition from Techopedia, 2018. In: Techopedia - Where Information Technology and Business Meet [online]. [cit. 7. 11. 2019]. Dostupné z: https://www.techopedia.com/definition/1245/structured-query-language-sql

World Wide Web Consortium (W3C) [online]. [cit. 09. 11. 2019]. Dostupné z: https://www.w3.org/Addressing/URL/uri-spec.html

Zabezpečení webu protokolem HTTPS - Nápověda Search Console. *Google Help* [online]. Copyright © 2020 Google [cit. 09. 03. 2020]. Dostupné z: https://support.google.com/webmasters/answer/6073543?hl=cs

8 Přílohy

Příloha 1 CD s přiloženými soubory:

- export databáze ve formátu .sql,
- zdrojový kód back-end aplikace,
- zdrojový kód webové prezentace včetně API rozhraní,
- zdrojový kód mobilní aplikace.

Příloha 2 UI specifikace back-end aplikace

Příloha 2

UI specifikace back-end aplikace

Název

Systém pro evidenci a správu dat

Motivace

Navrhnout a vytvořit uživatelsky přívětivé rozhraní, ve které budou moci pověřené osoby přidávat, upravovat a mazat data z databáze, které se následně zobrazují jak na webové prezentaci, tak v mobilní aplikaci.

Seznam cílů:

- přihlášení uživatele,
- evidence a úprava zemí,
- evidence a úprava destinací,
- evidence a úprava restaurací,
- evidence a úprava zájmových bodů (POI).

Persony

Hlavní persona 1

Jméno	Josef Bubák
Věk	35
Pohlaví	Mužské
Koníčky	Cestování, rodina, jídlo mobilní telefon, sociální sítě
Typický den	Vstane, vysprchuje se, jde do práce, po cestě se nasnídá, večer s přáteli do kavárny.
Stručná historie	Josef Bubák vystudoval business vysokou školu. Během pár let procestoval přes třicet destinací a vyzkoušel různorodé kultury a jídla.

Hlavní persona 2

Jméno	Lucie Nová
Věk	25
Pohlaví	Ženské
Koníčky	Cestování, jóga, vaření, studium, film
Typický den	Vstane, upraví se, nasnídá se, jde do školy. Odpoledne pracuje na částečný úvazek v cestovní agentuře. Po práci se navečeří s partnerem a koukají se na filmy.
Stručná historie	Lucie Nová je výborná studentka, která studuje a pracuje zároveň. Ráda cestuje a prozkoumává nové lokality.

Vedlejší persona

Jméno	Ota Majer
Věk	47
Pohlaví	Mužské
Koníčky	Cestování, podnikání, rodina, „bastlíř“
Typický den	Vstane, dá si kávu, nasnídá se na terase a jede autem do práce. Následně se schází s rodinou u večeře.
Stručná historie	Ota Majer vystudoval cestovní ruch a je ředitelem pobočky cestovní kanceláře. Osobně navštěvuje nabízené destinace.

Use Cases

Přihlášení

- Uživatel očekává zadání přihlašovacího jména.
- Uživatel očekává zadání přihlašovacího hesla.
- Uživatel očekává při správném zadání přihlášení do aplikace.
- Uživatel očekává při nesprávném zadání zobrazení chybové hlášky.

Stránka po přihlášení

- Uživatel očekává zobrazení stránky s informací o přihlášení.
- Uživatel očekává zobrazení krátkých informací, jak pracovat s aplikací.

Stránka zemí

- Uživatel očekává zobrazení seznamu zemí seřazených podle jména od A do Z.
- Uživatel očekává zobrazení stránky pro úpravu určité země po dané akci.
    - Uživatel očekává zobrazení stránky pro úpravu dané země.
    - Uživatel očekává zobrazení aktuálního nastavení země – název.
    - Uživatel očekává možnost upravení a uložení všech změn dané země.
    - Uživatel očekává možnost návratu na výpis všech zemí bez uložení změn.
- Uživatel očekává možnost smazání jednotlivých zemí po dané akci.
    - Uživatel očekává neumožnění smazání země v případě, že obsahuje nějaké destinace.

Stránka destinací

- Uživatel očekává zobrazení seznamu všech destinací vč. jejich země seřazených dle názvu destinací od A do Z.
- Uživatel očekává možnost úpravy jednotlivých destinací po dané akci.
    - Uživatel očekává zobrazení stránky pro úpravu dané destinace.
    - Uživatel očekává zobrazení aktuálního nastavení destinace – země, název.
    - Uživatel očekává možnost upravení a uložení každého nastavení dané destinace – země, název.
    - Uživatel očekává možnost návratu na výpis všech destinací bez uložení změn.
- Uživatel očekává možnost smazání jednotlivých destinací.
    - Uživatel očekává neumožnění smazání destinace v případě, že obsahuje nějaké nemovitosti.

Stránka restaurací

- Uživatel očekává zobrazení seznamu všech restaurací vč. jejich země a destinace seřazených dle názvu restaurace od A do Z.
- Uživatel očekává možnost filtrovat restaurace dle země a destinace.

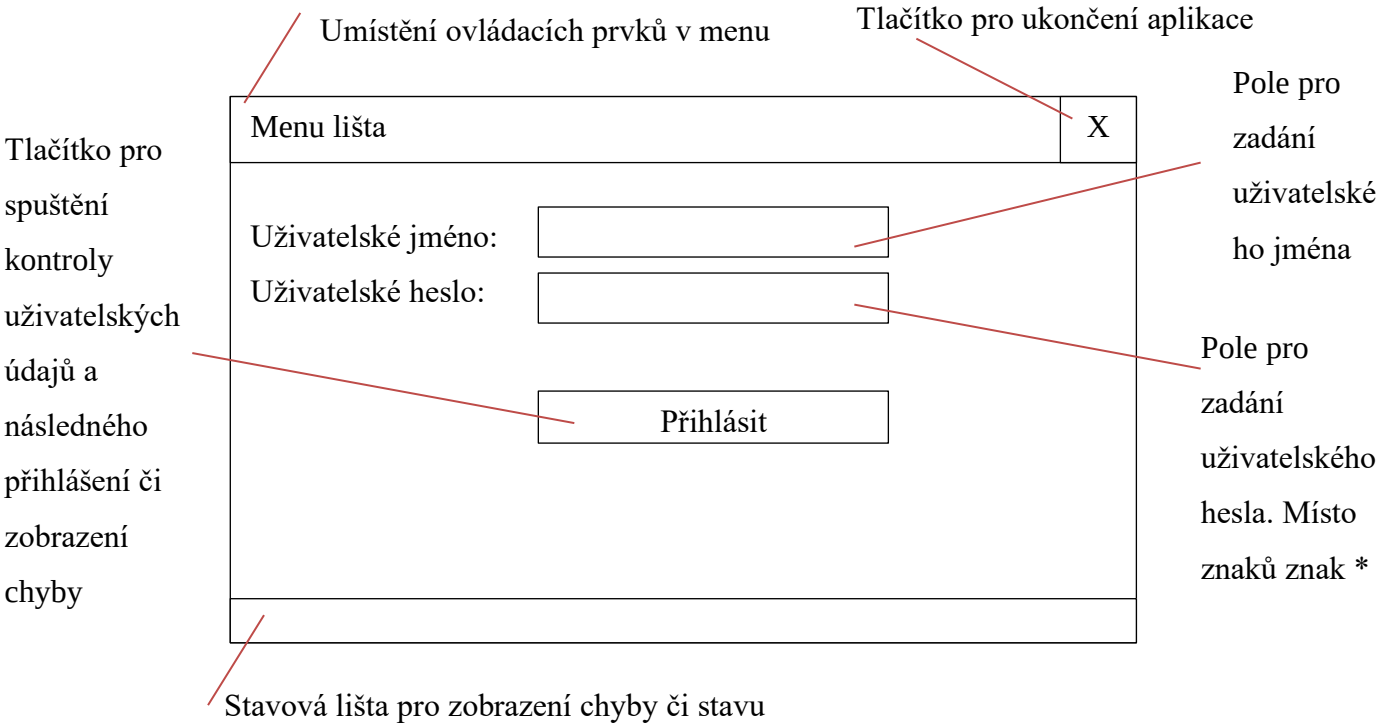
- Uživatel očekává možnost úpravy jednotlivých restaurací po dané akci.
    - Uživatel očekává zobrazení stránky pro úpravu dané restaurace.
    - Uživatel očekává zobrazení aktuálního nastavení restaurace vč. mapy zobrazující umístění.
    - Uživatel očekává možnost upravení a uložení každého nastavení dané restaurace.
    - Uživatel očekává možnost návratu na výpis všech restaurací bez uložení změn.
- Uživatel očekává možnost smazání jednotlivých restaurací.
    - Uživatel očekává neumožnění smazání destinace v případě, že obsahuje nějaké restaurace.

Stránka zájmových míst (POI)

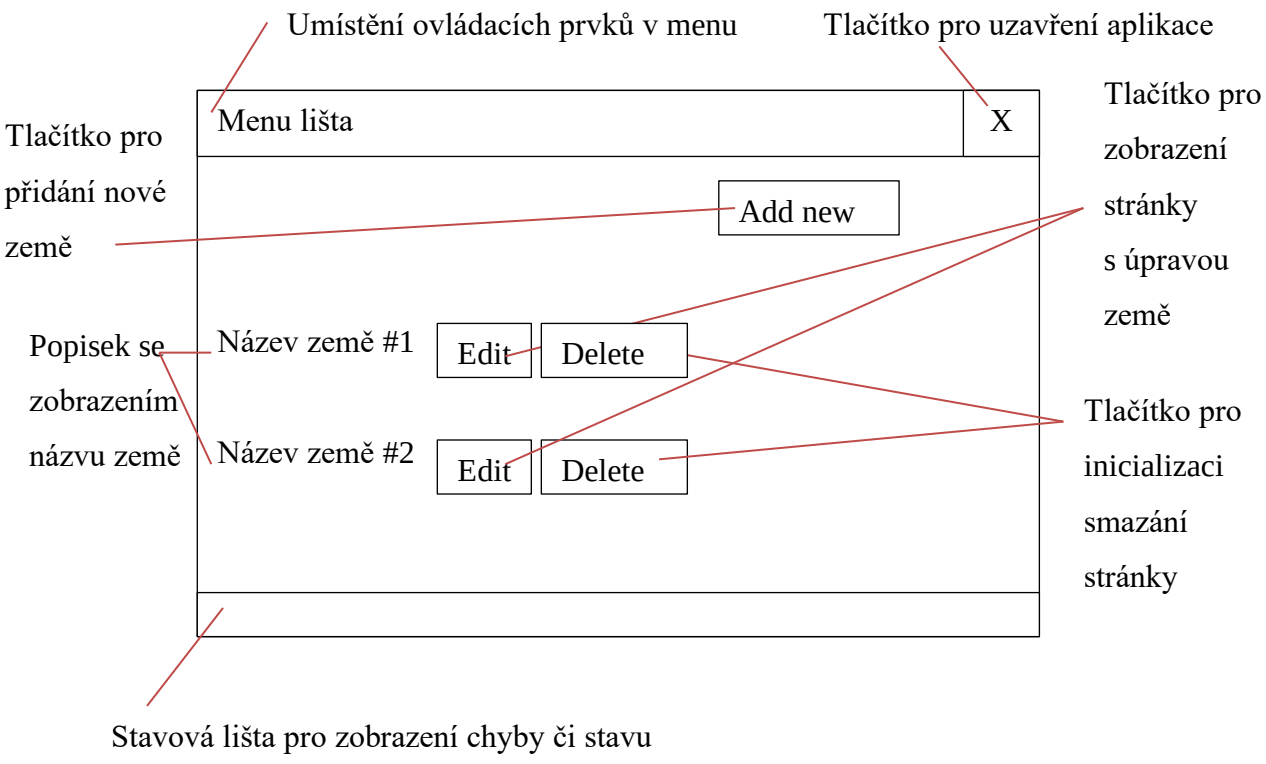
- Uživatel očekává zobrazení seznamu všech zájmových míst vč. jejich země a destinace seřazených dle názvu od A do Z.
- Uživatel očekává možnost filtrovat zájmová místa dle země a destinace.
- Uživatel očekává možnost úpravy jednotlivých restaurací po dané akci.
    - Uživatel očekává zobrazení stránky pro úpravy daného místa.
    - Uživatel očekává zobrazení aktuálního nastavení místa vč. mapy zobrazující umístění.
    - Uživatel očekává možnost upravení a uložení každého nastavení daného místa.
    - Uživatel očekává možnost návratu na výpis všech míst bez uložení změn.
- Uživatel očekává možnost smazání jednotlivých zájmových míst.
    - Uživatel očekává neumožnění smazání destinace v případě, že obsahuje nějaká zájmová místa.

Logické návrhy

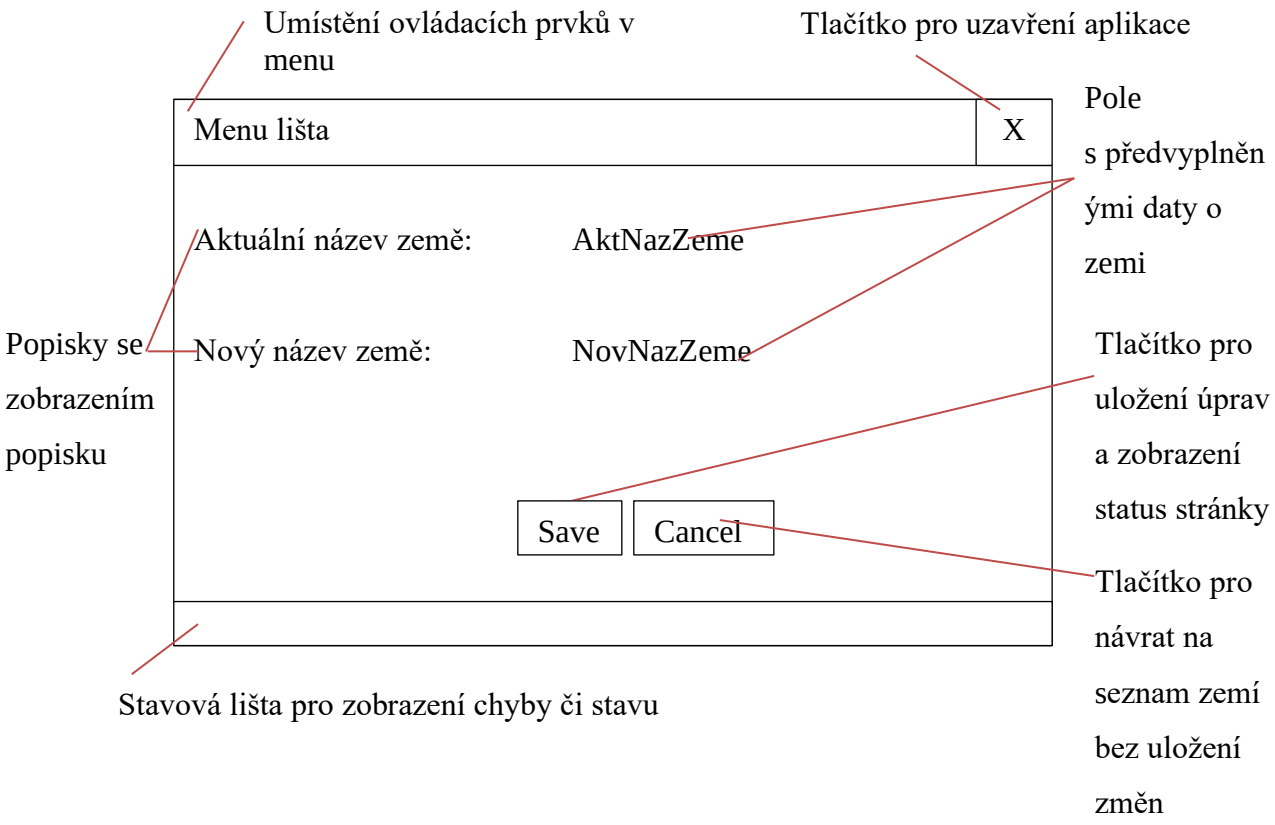
Stránka přihlášení



Stránka zobrazení zemí



Stránka úpravy země



Scénáře

Přihlášení

- Systém zobrazí stránku pro přihlášení.
- Systém zobrazí grafický prvek popisek pro pole uživatelské jméno.
- Systém zobrazí grafický prvek pole pro uživatelské jméno.
- Systém zobrazí grafický prvek popisek pro pole uživatelské heslo.
- Systém zobrazí grafický prvek pole pro uživatelské heslo a popisek.
- Systém zobrazí tlačítko Přihlásit.
- Po kliknutí na tlačítko Přihlásit systém provede kontrolu zadaných údajů.
    - V případě správného zadání zobrazí stránku po přihlášení.
    - V případě nesprávného zadání vypíše do stavové řádky chybovou hlášku.

Stránka po přihlášení

- Systém zobrazí stránku po přihlášení.
- Systém zobrazí informační hlášku o úspěšnosti přihlášení.
- Systém zobrazí grafický prvek s textovým obsahem.

Stránka zobrazení zemí

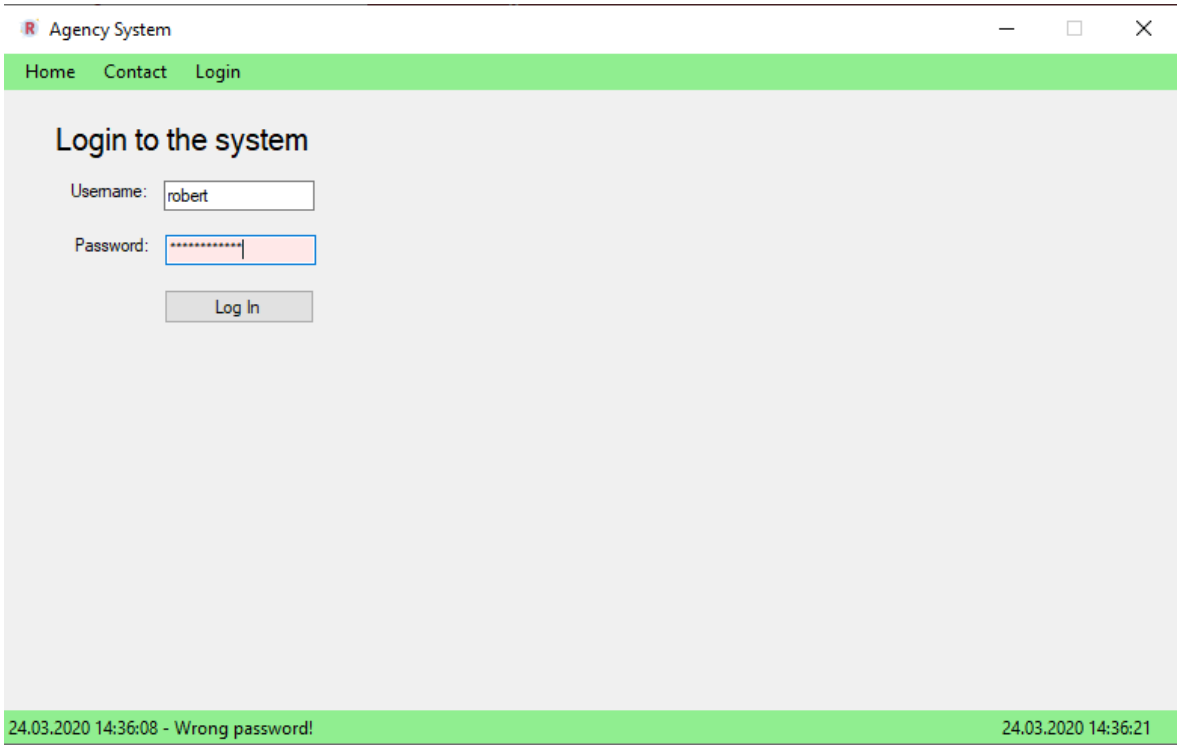
- Systém zobrazí stránku s přehledem všech zemí.
- Systém načte data a vygeneruje grafický prvek popisek a 2 tlačítka pro každou zemi v jednom řádku.
- Systém očekává kliknutí na tlačítko Edit.
    - Po kliknutí na tlačítko Edit systém zobrazí editační stránku dané země.
- Systém očekává kliknutí na tlačítko Delete.
    - Po kliknutí na tlačítko Delete systém inicializuje akci pro smazání země.

Stránka úpravy země

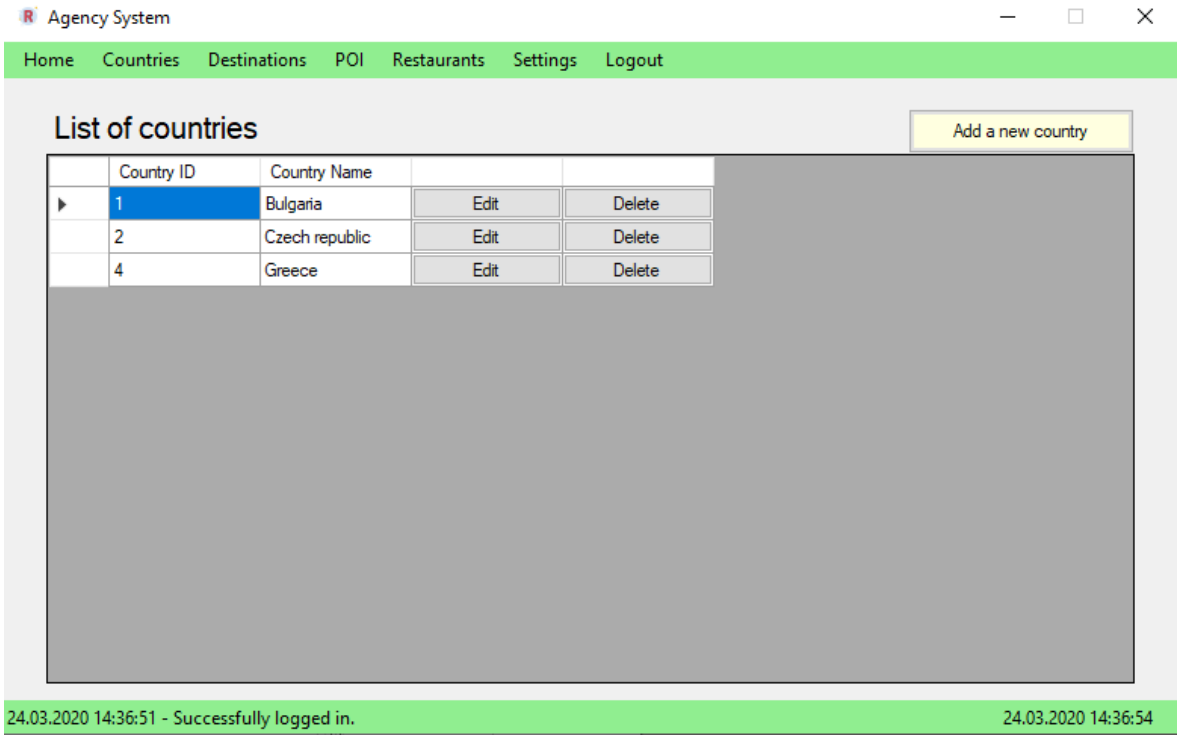
- Systém zobrazí stránku s úpravou dané země.
- Systém zobrazí grafický prvek popisek s titulkem pro aktuální název země.
- Systém zobrazí grafický prvek popisek s vyplněným názvem aktuální země.
- Systém zobrazí grafický prvek popisek s titulkem pro nový název země.
- Systém zobrazí grafický prvek popisek s předvyplněným názvem aktuální země.
- Systém očekává kliknutí na tlačítko Save.
    - Po kliknutí na tlačítko Save systém uloží nová data a zobrazí status stránku s informací o uložení.
- Systém očekává kliknutí na tlačítko Cancel.
    - Po kliknutí na tlačítko Cancel systém zobrazí stránku se seznamem všech zemí.

Grafické návrhy

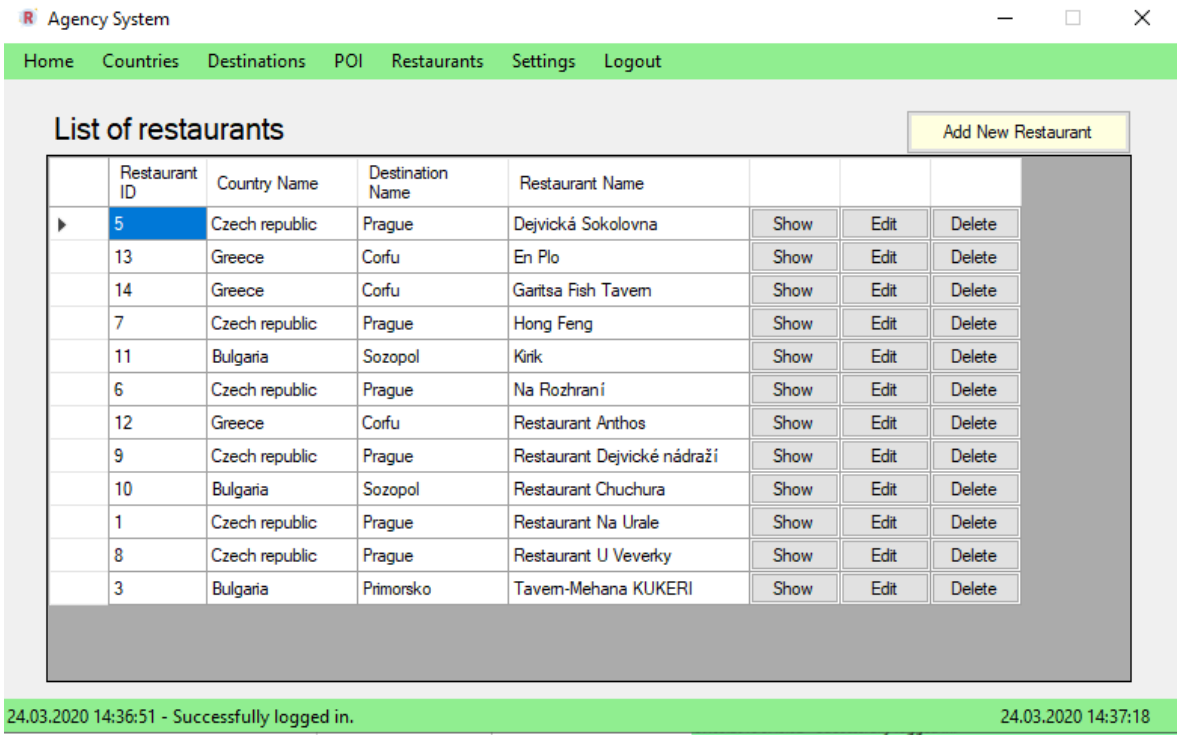
Stránka přihlášení



Stránka zobrazení zemí



Stránka zobrazení restaurací



Stránka upravení restaurace