Návrh aplikace pro realitní portál

Kryštof Raiskub

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Kryštof Raiskub**
Osobní číslo:	**A16083**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**prezenční**
Téma práce:	**Návrh aplikace pro realitní portál**
Téma anglicky:	**A Real Estate Web-Portal**

Zásady pro vypracování:

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

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **LAMBERT, Matt. Learning Bootstrap 4. 2nd Revised edition. Birmingham: Packt Publishing, 2016. ISBN 978-1785881008.**
2. **BUONANNO, Enrico. Functional programming in C#. Shelter Island, NY: Manning Publications, 2017. ISBN 16-172-9395-4.**
3. **FREEMAN, Adam. Pro Asp.net core MVC 2. 7th edition. New York, NY: Springer Science Business Media, 2017. ISBN 978-148-4231-494.**
4. **DE OLIVEIRA, Jason. Learning ASP.NET Core 2.0:: Build modern web apps with ASP.NET Core 2.0, MVC, and EF Core 2. Birmingham, UK: Packt Publishing, 2017. ISBN 1788476638.**
5. **VERMA, Rishabh a Neha SHRIVASTAVA. .NET Core 2.0 By Example. GB: Packt Publishing, 2018. ISBN 1788395093.**
6. **AQUINO, Chris a Todd GANDEE. Front-end web development: the Big Nerd Ranch guide. Atlanta, GA: Big Nerd Ranch, 2016. ISBN 978-0134433943.**

Vedoucí bakalářské práce: **Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **3. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 7. prosince 2018

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ Fakulta aplikované informatiky

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne Kryštof Raiskub v.r.

ABSTRAKT

Bakalářská práce se zabývá návrhem webové aplikace pro realitní portál, která bude sloužit pro správu a nabídku nemovitostí. V první části této bakalářské práce je provedena rešerše existujících řešení, a to jak na českém, tak i zahraničním (americkém) trhu. V další části je proveden rozbor technologií použitých na straně klienta i na straně serveru, které byly použity při návrhu aplikace. Následně je zpracován rozbor a analýza požadavků na funkcionalitu aplikace, od kterých se odvíjí praktická část, zabývající se samotnou realizací aplikace a vypracováním návodu použití pro uživatele.

Klíčová slova: Webová aplikace, realitní portál, webové technologie, analýza požadavků.

ABSTRACT

The thesis deals with the design of web application for real estate portal. The first part of the thesis analyzes existing solutions both on the Czech and foreign (United States) markets. The next part is focused on technologies used on the client and also server side during the realization process. Then takes a place analysis of funcionality requirements for the application, which also take a place in the practical part focused solely on the realization of the application and creation of a manual for the users.

Keywords: Web application, real estate portal, web technologies, requirements analysis

Děkuji svému vedoucímu práce Ing. Petru Šilhavému, Ph. D. za odborné vedení, za cenné rady a připomínky a také za neocenitelné zkušenosti získané realizací této bakalářské práce.
Dále čestně prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická, nahraná do IS/STAG jsou totožné ve znění:

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

OBSAH

ÚVOD

V dnešní době, kdy stále roste poptávka po bydlení, se stávají systémy pro nabídku realit stále relevantnějšími. K prvnímu kontaktu zákazníka s trhem nemovitostí dochází většinou na internetu a při tak velkém množství inzertních portálů je nutné hledat nové a inovační způsoby, jak zákazníka přimět využívat náš systém. Realitní kanceláře a majitelé nemovitostí jsou ale stejně tak důležití, jako zákazníci, a proto je nutné vytvořit takové prostředí, které bude vyhovovat oběma stranám.

Návrh takové aplikace obnáší nejenom samotné programování, ale také úvodní studii a analýzu, ve které dojde k ucelení požadavků a představy o tom, co by měl systém svým uživatelům nabízet a tyto poznatky v další fázi aplikovat na vývoj samotného softwaru.

Co dělá aplikaci úspěšnou aplikací, je obecně těžké říci, někdy ale může vyhrát aplikace s intuitivnějším ovládáním a příjemnějším grafickým rozhraním nad aplikací s rozšířenější funkcionalitou, která ale vzhledem k cílové skupině postrádá smysl. Systémy pro nabídku realit jsou produktem využívaným všemi věkovými kategoriemi, vyjma těch nejmladších, a proto je povinností vývojáře na tuto skutečnost při realizaci myslet.

Úkolem této bakalářské práce je navržení a realizace tohoto typu aplikace od počáteční fáze studií až po konečnou realizaci za použití standartních technologií HTML, CSS a Javascript pro tvorbu webových prezentací a moderních technologií pro vytvoření rychlé a spolehlivé implementace serveru, která bude oporou celého systému.

I. TEORETICKÁ ČÁST

1 REŠERŠE EXISTUJÍCÍCH ŘEŠENÍ

Inzertních portálu pro správu a nabídku nemovitostí je v dnešní době na internetu opravdu mnoho, a to jak na českém, tak i zahraničním trhu. Nejenom, že máme na trhu veřejné portály, jejichž obsah tvoří samotní uživatelé, ale mimo to máme také soukromé portály, které slouží konkrétním realitním kancelářím pro jejich potřeby podnikání.

1.1 Český trh s inzertními portály

1.1.1 Sreality.cz

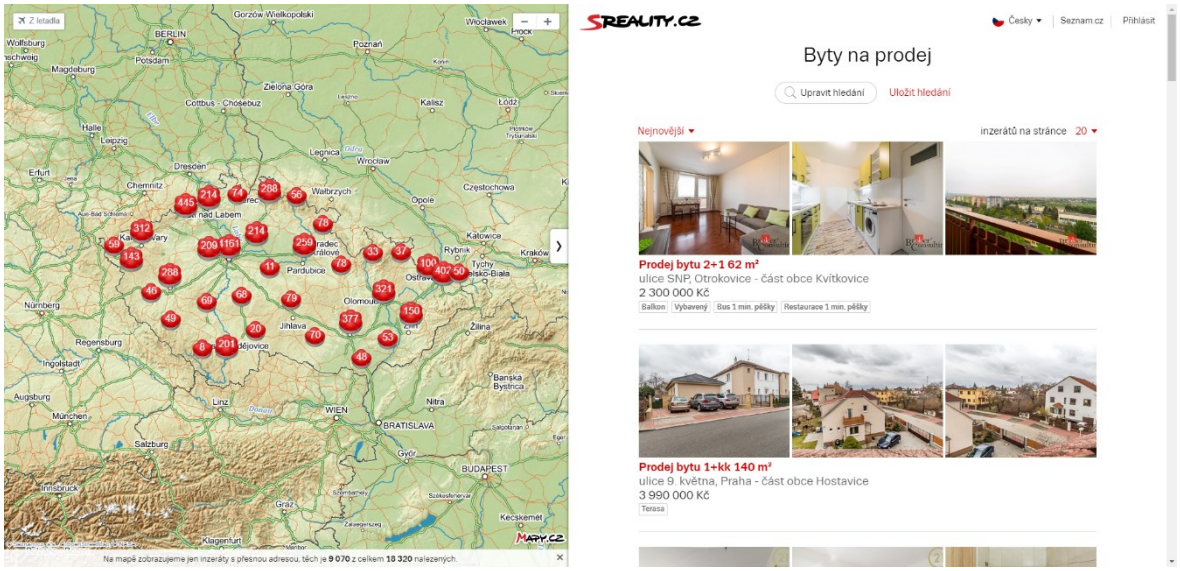
Asi největším a nejznámějším portálem na českém trhu jsou Sreality.cz, které zapadají do ekosystému různých služeb a dalších inzertních portálů pod značkou Seznam.cz. Tento inzertní portál je velmi strohý a zaměřuje se pouze na správu a nabídku nemovitostí, nesetkáme se zde s žádnou formou personalizace, na druhou stranu ale přesně plní požadavky na systém tohoto typu, čili business modelu zákazník (resp. realitní kancelář, majitel nemovitosti) – zákazník.

Za portálem Sreality.cz stojí určitě velký tým vývojářů, protože pro správu nabídek s nemovitostmi lze využít jednak administraci přímo na webu, ale také speciální software pro správu nemovitostí a mimo to je vytvořena pomocí moderního frontend frameworku Angular.

Inzerce na tomto portálu je placená jak pro realitní kanceláře, tak pro soukromé osoby a z tohoto důvodu jsem nemohl projít procesem přidávání nabídky s nemovitostí.

Výhody

Hlavní výhodou Srealit je určitě postavení na trhu a s tím spojená návštěvnosti webu. Podle samotných Srealit je měsíční návštěvnost více než 1 000 000 reálných uživatelů. Jednoduchý design, který je intuitivní a přehledný, určitě skvěle pokrývá celé spektrum návštěvníků webu, od nejstarších až po nejmladší a skvělé je také vyhledávání nemovitostí skrze mapu.



Obr. 1. Vyhledávání podle mapy na Srealitách

Nevýhody

Nevýhodou se zdá být složitost procesu přidávání nabídky s nemovitostí. Na první pohled se zdá, že soukromá osoba, která chce přidat novou nemovitost musí kontaktovat infolinku. Další nevýhodou je placená inzerce, jejíž cena je cca 70,- Kč/den.

1.1.2 M&M Reality

Opakem Srealit může být například inzertní webová stránka realitní kanceláře M&M Reality, která je soukromá a slouží pouze pro potřeby této realitní kanceláře. Na první pohled je web typickou webovou stránkou sloužící pro účely soukromého podnikání a oproti Srealitám zde mimo nabídku nemovitostí narážíme na velkou formu personalizace. Můžeme si tedy přečíst novinky z blogu, prohlídnout, jaké služby realitní kancelář nabízí, v sekci „O nás“ se můžeme dočíst více informací o realitní kanceláři nebo se podívat na možnosti kariéry.

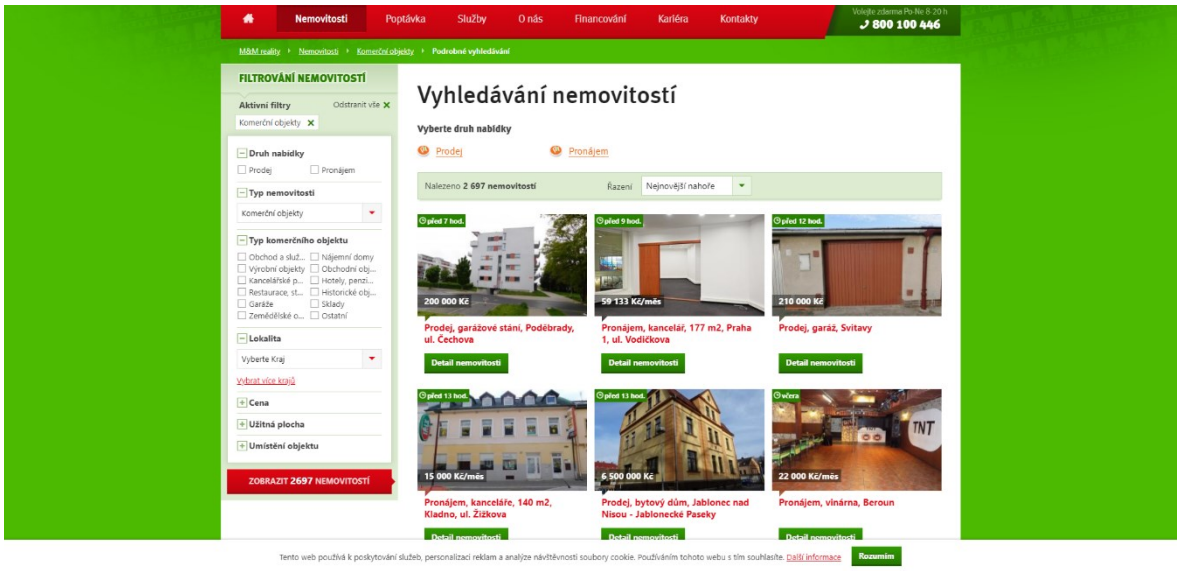
Výhody

Soukromý inzertní portál je určitě vhodnější a smysluplnější pro budování klientské základny a v rámci možností v sektoru realit také budování opakujících se klientů a návštěvníků webu. Ačkoliv má například M&M Reality uživatelský účet i na Srealitách (pravděpodobně z důvodu návštěvnosti), soukromý web působí profesionálně a realitní kancelář není závislá na cizí platformě, která může kdykoliv skončit (i když je to nepravděpodobné) nebo například změnit podmínky užití, které nebudou realitní kanceláři vyhovovat.

Výhodou jsou také rozsáhle profily makléřů, které obsahují informace o makléři, všechny jeho nabídky, ale také například recenze.

Nevýhody

Nevýhodou webu jsou modely pro ukládání nemovitostí do databáze. Z komplexnosti vyhledávání, kde můžeme filtrovat pouze podle pěti kategorií a také podle parametrů nemovitosti na detailní stránce konkrétní nabídky se zdá, že modely jsou koncipovány pouze pro opravdové minimum základních informací o nabídce a ostatní informace je nutno popsat v textovém popisu nemovitosti.



Obr. 2. Vyhledávání nemovitostí na M&M reality

1.2 Zahraniční trh s inzertními portály

Na zahraničním trhu jsem se zaměřil zejména na trh americký, a to z důvodu jazykové bariéry.

1.2.1 Realtor.com

Americký inzertní portál realtor.com je něco mezi Srealitama (veřejný portál) a M&M Reality (soukromý portál). Na portál může přidávat nabídky kdokoliv, kdo má uživatelský účet, mimo to ale realtor.com poskytuje několik jiných, zajímavých služeb pro návštěvníky tohoto webu, například pomoc a předschválení při vyřizování hypotéky, sledování vývoje ceny vlastní nemovitosti, nebo také rady a tipy při kupování nemovitosti v různých lokalitách, blog apod. Ačkoliv tento typ obsahu nedávají na portál inzerenti, ale samotná firma

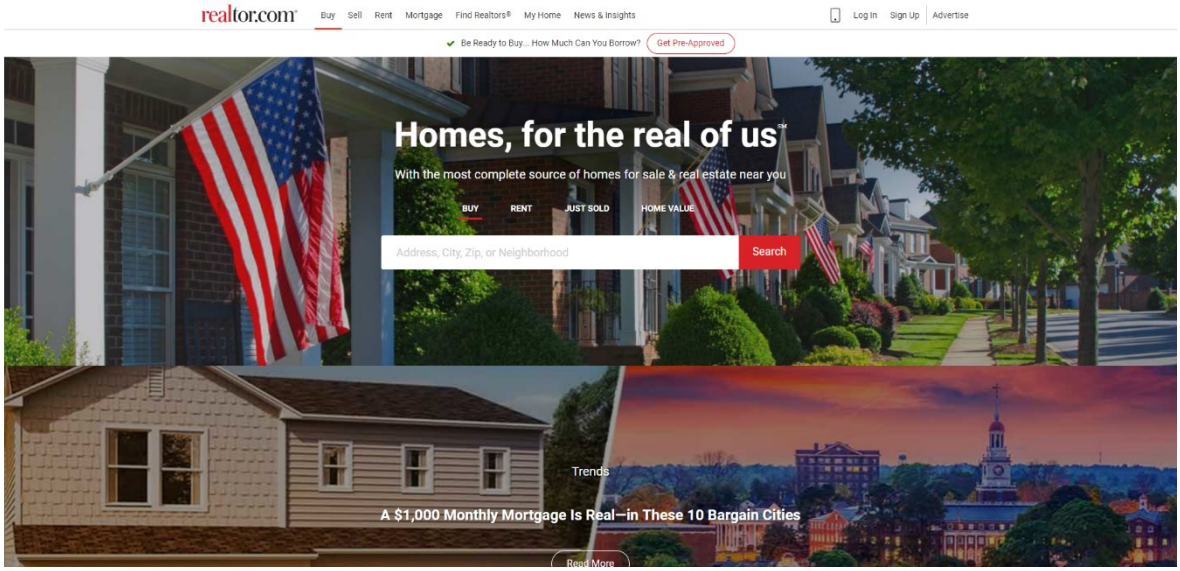
realtor.com, setkáváme se zde s určitou formou personalizace, web není tak strohý, jako například Sreality a návštěvník webu má více důvodů, proč web opakovaně navštěvovat. Tento obsah je zdrojem návštěvnosti, což pomáhá i samotným inzerentům, kteří z toho také profitují a mají vyšší návštěvnost na svých nabídkách.

Realtor.com také disponuje dvěma mobilními aplikacemi na operační systémy Android a iOS, z nichž jedna se specializuje pouze na pronájmy a druhá je zaměřena na nabídku a prodej nemovitostí.

Výhody

Podobně jako u Srealit, velkou výhodou portálu realtor.com je návštěvnost, která činí podle SimilarWeb.com průměrně 85 milionu návštěvníků za měsíc. K těmto číslům přispívají mimo samotných inzerátů s nemovitostmi také doplňkové služby a blog portálu realtor.com.

Zajímavé je také jednoduché vyhledávání nemovitostí za pomocí pouze jednoho kritéria - adresy, další detailní parametry je možné zadat až po vyhledání výsledků podle adresy. Navíc jsou tyto detailní parametry dostupné vždy v horní liště a návštěvník se tak nemusí vracet o krok zpět na vyhledávácí obrazovku. Z hlediska zdrojů na straně serveru to není nejefektivnější řešení, ale je to skvělá uživatelská zkušenost a velmi rychlý způsob vyhledávání. I když je takové vyhledávání možné i například na Srealitách, v momentě, kdy má návštěvník k dispozici velké množství vyhledávácích parametrů se obvykle se snaží vyplnit co nejvíce z nich a doba, než uvidí samotné inzeráty se značně protáhne.



Obr. 3.Vyhledávání nemovitostí na Realtor.com

Realtor.com také disponuje detailním profilem inzerenta včetně recenzí, stejně tak, jako tomu je u M&M Reality a na detailních stránkách jednotlivých nabídek zobrazuje spoustu užitečných informací, jako například daň z nemovitosti, kalkulačku hypotečních splátek apod.

Nevýhody

Při tak velké návštěvnosti je nevýhodou konkurence inzerentů, kde se může nabídka s nemovitostí velmi rychle ztratit mezi ostatními a „zapadnout“ ve výsledcích vyhledávání.

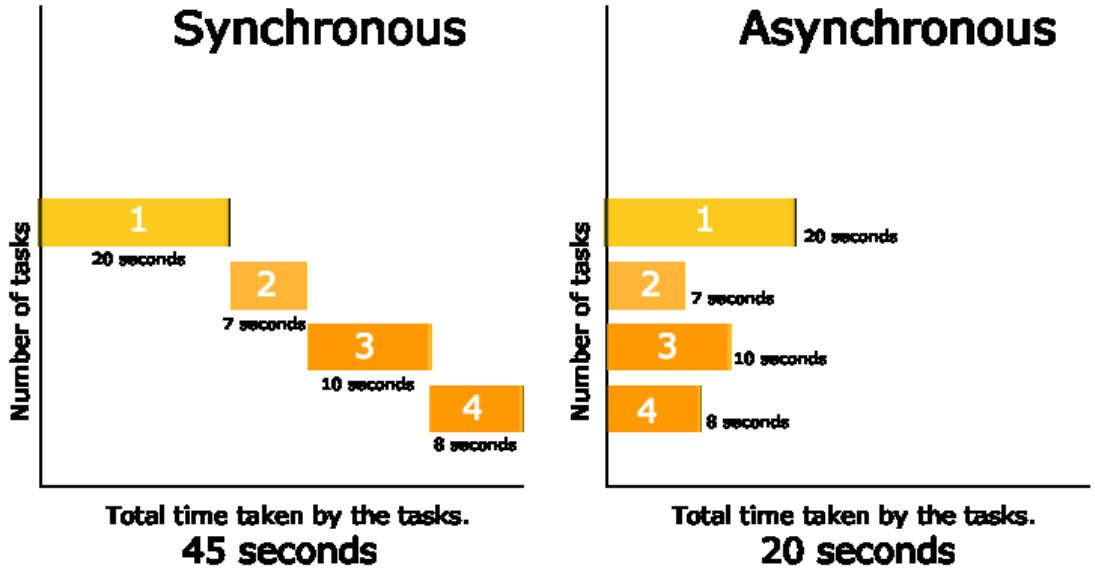
2 POUŽITÉ TECHNOLOGIE

Základními technologiemi pro tvorbu webových stránek, ať už statických, či dynamických, jsou HTML, CSS a Javascript. Tyto technologie slouží k vytváření toho, co vidí návštěvník webu, tak zvaného „frontendu“.

U dynamických stránek, které pracují s persistentními daty a databází, je potřeba také zvolit správnou technologii na straně serveru, jinak taky „backendu“. Možností pro programování serveru je spousta. Každý programovací jazyk, který je schopný zpracovávat HyperText Transer Protocol (HTTP) požadavky (HTTP request) a odeslat zpět odpověď (HTTP response) je vhodný pro naprogramování webové aplikace. Mezi takové programovací jazyky patří například Python, Javascript na enginu Node.js, Ruby nebo C#. Většina programovacích jazyků a všechny zmíněné disponují také frameworkama, což jsou šablony, které obsahují sady nástrojů a usnadňují vývoj požadované webové, ale i například desktopové aplikace.

Každý programovací jazyk má své případy užití a mimo to, zda jsou například objektově orientované se dělí také na blokující a neblokující architekturu.

- **Neblokující (asynchronní) architektura** – procesy při vykonávání IO operací (čtení/zápis do databáze, práce s filesystémem, atd.) neblokují procesor, jakmile je výsledek procesu k dispozici, zařadí se do fronty a procesor ve vhodný okamžik předá výsledek do programu, kde je zpracován tzv. callbackem
- **Blokující (synchronní) architektura** – procesy při vykonávání IO operací blokují procesor a žádný další proces nemůže být do ukončení vykonávání aktuálního procesu spuštěn [1]



Obr. 4. Synchronní vs. asynchronní architektura [2]

Typickým příkladem pro neblokující architektury je programovací jazyk Javascript běžící na enginu Node.js (pro webové aplikace). Javascript je programovací jazyk optimalizovaný pouze pro jedno jádrové procesory, jemuž neblokující architektura umožňuje maximální využití zdrojů procesoru a v rámci webových aplikací je vhodný pro aplikace závislé na IO operacích. Na druhou stranu se nehodí pro operace náročné na procesor, například pro zpracování obrázků, které by po dobu zpracování zablokovalo procesor a ostatní požadavky by musely čekat, než se obrázek zpracuje. [3]

Většina programovacích jazyků jsou postaveny na blokující architektuře, které jsou ale také zároveň optimalizovány pro více jádrové procesory. V případě webových aplikací, kdy je jedno jádro zablokováno procesem a přijde další požadavek, je tomuto požadavku přiřazeno další volné vlákno procesoru a dochází k tzv. multitaskingu. [1]

2.1 Frontend technologie

2.1.1 HyperText Markup Langugage

Základním stavebním kamenem každé webové stránky je HyperText Markup Language (HTML). Je to značkovací jazyk definující prvky na webové stránce. Aktuální verze HTML je HTML5.

Každý HTML dokument se skládá ze základní struktury obsahující typ dokumentu (DOCTYPE), který slouží prohlížečům k rozpoznání verze HTML, dále hlavičky stránky,

ve které se definují importy (kaskádové styly, scripty, knihovny, meta tagy) a těla stránky, kde jsou umístěny jednotlivé HTML prvky.

Jednotlivé prvky se definují pomocí párových a nepárových tagů, pomocí kterých můžeme definovat například:

- Video - <video>
- Nadpisy různé úrovně - <h1></h1> - <h6></h6>
- Paragraf -
- Logický blok -
- Vektorovou grafiku - <svg>
- Hypertextové odkazy - <a> - slouží k navigování mezi stránky

x.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Nadpis dokumentu</title>
</head>
<body>

    <div>
        <h1>Hlavní nadpis</h1>
        <p>Paragraf</p>
        <img src="img/foto.jpg">
    </div>

</body>
</html>
```

Obr. 5. Základní struktura HTML a definice prvků

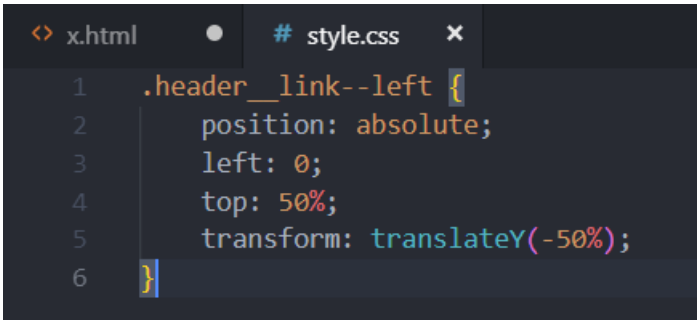
Mimo to ale také HTML5 definuje řadu užitečných API, díky kterým můžeme přistupovat k poloze uživatele, získat přístup k webkameře nebo detekovat změnu orientace displeje (zejména u mobilních zařízení). K těmto API se přistupuje pomocí Javascriptu. [4]

2.1.2 Kaskádové styly

Ačkoliv můžeme stylovat HTML prvky přímo v HTML pomocí inline stylů, tento způsob není nejlepším přístupem. Dochází tak k velké duplicitě kódu a velmi špatné udržovatelnosti kódu.

Lepším přístupem pro stylování HTML prvků je za pomocí kaskádových stylů (CSS). CSS umožňuje definování tříd a identifikátorů s příslušnými „key – value“ atributy (styly), které poté můžeme přiřadit HTML prvkům a vyvarovat se tak duplicitnímu kódu a oddělením logiky zvýšit jeho udržovatelnost. [5]

Problémem při vytváření CSS stylů může být při velkém počtu tříd jejich pojmenovávání. Pro jejich zjednodušení a zavedení řádu je možné zavést například konvenci pojmenovávání tříd Block – Element – Modifier (BEM). Tato konvence je skvělá pro lepší organizaci kódu a jeho jednotnou strukturu a klade si za cíl vytvářet nezávislé bloky, které jsou znovupoužitelné. Blokem může být například fotogalerie, elementem jedna konkrétní fotografie a modifikátorem zaoblení rohů fotografie. Další skvělou konvencí je přidávání předpony „js-“ k HTML prvkům, které slouží pouze k manipulaci skrze Javascript. [6]



Obr. 6. Aplikování stylů metodou BEM

CSS nám také umožňuje animování prvků za pomocí přechodů, a nebo také „keyframes“, které se hodí pro komplexnější animace. Animovat je možné velké množství atributů, ale pouze průhlednost a transformace jsou jediné atributy, které zaručují plynulý 60 snímků za sekundu, zejména při animaci velkého množství prvků v jeden okamžik. Při počátečním renderování prvků na webové stránce prochází každý prvek čtyřmi fázemi:

1. Stylování – aplikování CSS stylů
2. Rozložení – pozicování prvku, vypočítávání jeho velikosti
3. Vykreslení – vybarvení pixelů prvku (obrázky, stíny, barvy)
4. Kompozice – kompozice prvků na stránce, jejich pořadí při překrývání

S každou další animací prvku musí prvek projít určitými fázemi, podle toho, jaký atribut je animován. Nejefektivnějšími jsou atributy průhlednost (opacity), který vyvolá pouze fázi vykreslení a atribut transformace (transform), který vyvolá pouze fázi kompozice. [7]

2.1.3 Javascript

Javascript je programovací jazyk navržený v 90. letech k řízení logiky na webových stránkách. Ačkoliv název obsahuje slovo „Java“, s programovacím jazykem Java nemá nic společného a tento název měl pouze upoutat pozornost Java programátorů.

Javascript je „weakly typed“ programovací jazyk, nesetkáme se u něj tedy s datovými typy. Ať už pracujeme s celými čísly, řetězci nebo poli, můžeme pro deklaraci použít kterýkoliv z následujících tří typů:

- **var** – jeho „platnost“ je v rámci funkce
- **let** – od verze ES6, jeho „platnost“ je v rámci bloku
- **const** – po inicializaci není možné změnit hodnotu

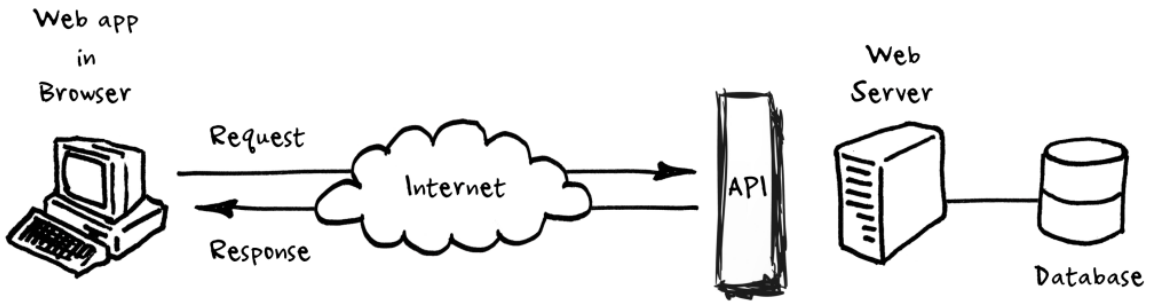
a je interpretován „za běhu“. S tím se pojí velký nedostatek Javascriptu, a tím je debugging. Jakákoliv chyba v kódu se projeví až při spuštění programu přímo v prohlížeči.

Dalším problémem, s kterým se Javascript potýká, je podpora prohlížečů. Jelikož je kód interpretován za běhu enginem umístěným přímo v prohlížeči klienta, je klient, ale také vývojář závislý na daném enginu a jeho implementaci jednotlivých specifikací jazyka. Javascript má několik verzí, do roku 2014 zde byla pouze jediná verze Javascriptu - ES5 (ECMAscript) a dá se o ní říci, že jsou její specifikace z velké části implementovány ve všech prohlížečích. V roce 2015 přišla nová verze – ES6 a od té doby vychází každý rok nová verze přinášející nové vlastnosti a specifikace.

Důležitou rolí Javascriptu je přístup do Document Object Modelu (DOM), díky němu je možné modifikovat a přistupovat k obsahu dokumentu (HTML stránky). Obsahuje všechny jednotlivé HTML prvky a jejich atributy, ke kterým můžeme přistupovat a měnit je pomocí DOM API. [8]

Pomocí asynchronního Javascriptu (AJAX) můžeme také navazovat spojení se službami třetích stran a využívat tak existujících služeb namísto vytváření vlastní implementace požadované služby. Komunikovat se vzdálenou službou můžeme za pomocí posílání požadavků na Application Programming Interface (API). API definuje, jak a jakým způsobem je možné s danou službou komunikovat. Definuje přístupové body aplikace, typ dat, které očekává při zaslání požadavku a v opačném případě typ dat, které vrací jako odpověď. Při komunikaci s externími servery se ve většině případů komunikuje pomocí datového typu Javascript Object Notation (JSON). Komunikace se službami třetích stran probíhá asynchronně a

neblokuje tedy procesor při čekání na odpověď dotazovaného serveru. V případě, že by toto spojení probíhalo synchronně a na odpověď bychom museli čekat například 3 sekundy, po celou tuto dobu by byl zablokován procesor a webová stránka by nereagovala na interakce uživatele, tzn. kliknutí na tlačítko nebo například odeslání formuláře.



Obr. 7. Komunikace s externími servery [9]

Na internetu je velká spousta veřejných API, se kterými můžeme pomocí Javascriptu navázat spojení. Velmi často používanými službami jsou například služby poskytující informace o počasí (OpenWeatherMap) a mapy (Google Maps, Mapbox), ale je možné narazit také na nějaké zajímavější API jako je například API pro majitele automobilů Tesla, přes které umožňuje ovládat automobil na dálku.

Další využití asynchronního Javascriptu je komunikace s vlastním serverem, například při potřebě stažení dodatečných dat. Můžeme posílat standartní požadavky HTTP GET pro získávání dat a požadavky POST pro posílání dat na server. Požadavek na server se odešle za pomocí Javascriptu a také se jím zpracuje odpověď. Celá operace se děje v pozadí bez jakéhokoliv přerušení a aktualizování prohlížeče, a zlepšuje tak „user-experience“.

2.1.4 Bootstrap

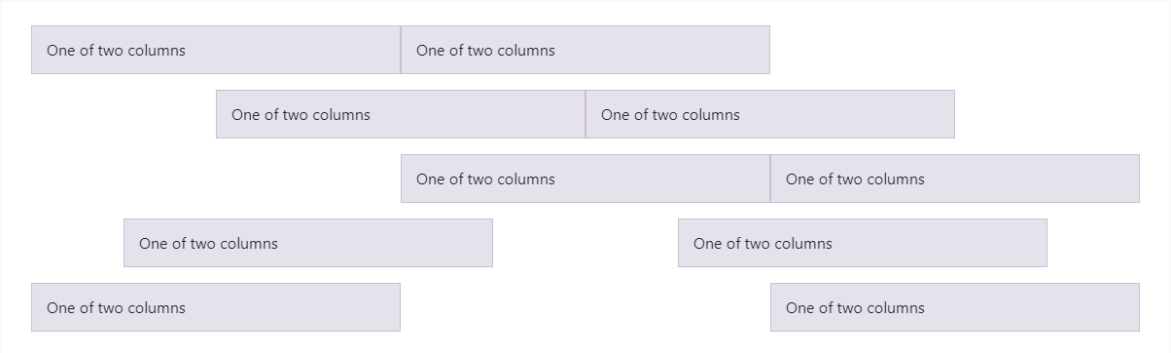
Bootstrap je frontend knihovna komponentů a dalších pomocných CSS tříd pro tvorbu responzivních, „mobile-first“ grafických rozhraní na webu.

„Mobile first“ znamená, že primárním a také defaultním cílem jsou přenosná zařízení, jako jsou telefony nebo tablety a Bootstrap je opravdu skvělým nástrojem pro vytváření responznivního designu. Pro tvorbu responsivního designu využívá čtyř základních předpon tříd, které pokrývají celé spektrum velikostí displejů:

- **Bez předpony** – displeje do 576px – odtud také „mobile first“
- **Sm** – malé displeje nad 576px

- **Md** – středně velké displeje nad 768
- **Lg** – velké displeje nad 992px
- **Xl** – extra velké displeje nad 1200px

Tyto předpony lze spojovat s velkou sadou tříd, které pak mění styly požadovaných HTML elementů při překročení určité velikosti displeje. Ve spojení tříd pro vytváření layoutů, nazývané „Grid layout“, a předpon pro responzivní design je snadné vytvořit velmi rychle layout a přizpůsobit ho na všechny velikosti displejů, od velkých desktopů až po telefony.



Obr. 8. Bootstrap grid systém [10]

Bootsrap obsahuje také velkou spoustu předpřipravených komponent, které je možné také modifikovat a rozšiřovat. Užitečnými jsou zejména komponenty pro tvorbu navigačních menu, formulářů, vyskakovacích oken a sliderů pro obrázky. Bootstrap má v základu svůj specifický styl a pokud je vyžadován vlastní design aplikace, je nutné Bootstrap rozšířit o své vlastní CSS třídy a buďto přepsat třídy Bootstrapu, nebo je kompletně nahradit.

Nevýhodou také je, že pro správné fungování některých komponent je zapotřebí dalších závislostí (Popper.js, jQuery), a tím se zvyšují požadavky na server a doba nahrávání webové stránky z důvodu stahování a parsování těchto závislostí. [11]

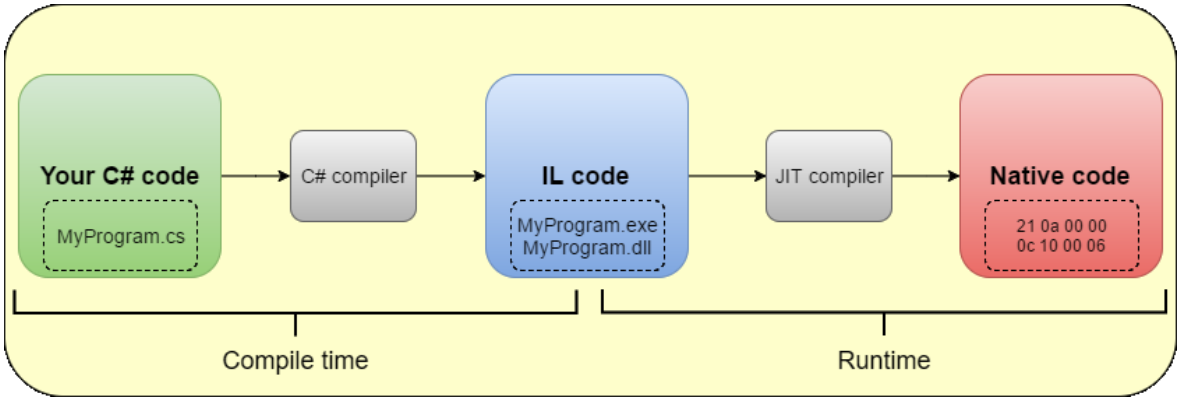
2.2 Backend technologie

Aplikace bylo naprogramována v programovací jazyku C# a frameworku ASP.NET Core, obě dvě jsou technologiemi firmy Microsoft.

2.2.1 Programovací jazyk C#

Programovací jazyk C# je moderní, objektově orientovaný programovací jazyk, se kterým lze programovat mobilní aplikace, desktopové aplikace, ale také webové aplikace a spoustu dalších.

C# má základy v programovacím jazyku C, ale oproti C je to programovací jazyk vyšší úrovně. C# je částečně kompilovaný jazyk a částečně také interpretovaný za „běhu“ programu. Po kompilaci C# kódu je kód přetransformován na Intermediate Language (IL) a je uložen ve spustitelném souboru (.exe). Common Language Runtime (CLR) je program sloužící pro spouštění a interpretování IL kódu kompilátorem Just In Time („právě v čas“), a při spuštění IL je tento kód transformován do nativního kódu, kterému „rozumí“ procesor a dokáže jej zpracovat. JIT kompilátor také při spuštění provádí optimalizaci programu na hardware, na kterém je spuštěný.



Obr. 9. Proces kompilace C# kódu [12]

Při kompilování C# kódu do IL je možné zachytit různé chyby syntaxe v kódu (např. chybějící středník), ale zásadní chyby, které mají vliv na funkčnost (přístup k neexistujícímu prvku v poli), jsou odhaleny až při kompilovaní JIT kompilátorem za „běhu“ programu.

C# disponuje oproti programovacím jazykům nižší úrovně „Garbage Collectorem“, který se stará o správu paměti programu a automaticky odstraňuje nedosažitelné a nepoužívané objekty. Oproti Javascriptu je C# „strongly typed“ programovací jazyk a pro deklaraci proměnných a návratových hodnot metod používáme datové typy (string, int, bool apod.). [13]

2.2.2 Framework ASP .NET CORE

ASP.NET Core je framework postavený na programovacím jazyku C# a je to multiplatformí open-source framework pro tvorbu různých typů aplikací. ASP.NET Core je momentálně ve stabilní verzi 2.2, současně je dostupný i „preview“ verze 3.0, která se zaměřuje na rozšíření palety aplikací, které je možné s tímto frameworkem vytvářet.

Tento framework má pevně danou adresářovou strukturu, názvy kontrolerů a pohledů. Je jakousi šablonou pro zvolený typ aplikace. urychluje její vývoj a obsahuje spoustu užitečných nástrojů a modulů. Jedním z nich je například modul „Identity“ pro správu

uživatelských účtů, nebo také předpřipravené layouty pro tvorbu Web API projektů s frontendem založeným na knihovně React.js nebo frameworku Angular.

Při programování webových aplikací s frontendem renderovaným na straně serveru obsahuje každý nově vytvořený projekt také knihovnu Bootstrap a knihovnu jQueryValidate pro validaci formulářů. [14]

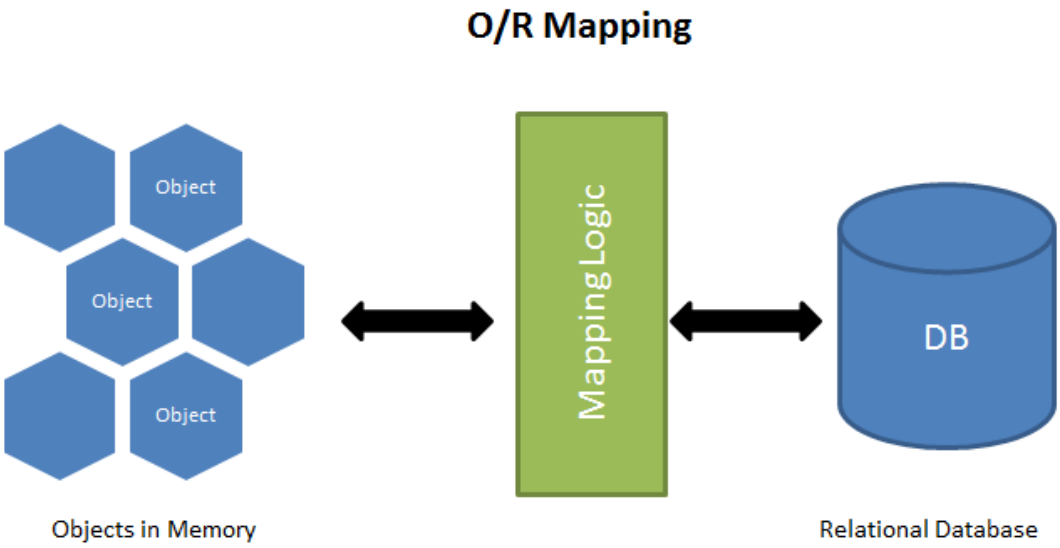
Co se týče architektury, framework ASP.NET Core je defaultně synchronní, má ale také podporu pro asynchronní programování a ve spojení s podporou pro více jádrové procesory by měla být teoreticky tato metoda implementace velmi svižná a schopná vyřídit velké množství příchozích požadavků za současného snížení čekací doby na odpověď na straně klienta.

K frameworku je k dispozici také Command Line Interface (CLI) obsahující základní příkazy pro zakládání a řízení projektů, jako jsou například generování migrací a aktualizování databáze, generování certifikátů pro podporu protokolu HTTPS, sestavení projektu, instalace NuGet balíčků apod. [15]

2.2.3 Entity Framework Core

Dalším frameworkem, který byl při realizaci bakalářské práce použit, je open-source, multiplatformní, objektově – relační mapovací (ORM) framework Entity Framework Core od společnosti Microsoft. Předchůdcem tohoto frameworku je Entity Framework poprvé představený v roce 2008 ve verzi 1.0, nejnovější verze vyšla v roce 2013 ve verzi 6.0 a v roce 2016 byl tento framework kompletně přepsán a dostal přívlastek Core, který je aktuálně ve verzi 2.0.

ORM systém je abstrakcí nad samotnou implementací databáze a umožňuje vývojářům přistupovat k databázi objektově orientovaným programovacím jazykem jejich volby (v případě Entity Frameworku Core jazyk C#), namísto jazyku SQL, což značně urychluje vývoj softwaru. ORM je v architektuře aplikace umístěn mezi business logikou a samotnou databází a zajišťuje mapování dat z databáze na objekty business logiky a naopak.



Obr. 10. Schéma ORM [15]

V případě Entity Frameworku je s databází komunikováno za pomocí .NET objektů, a mimo to umožňuje přístup do databáze syntaxí LINQ, která také chrání aplikaci před útokem typu SQL injection. [16]

2.3 Další použité technologie a nástroje

2.3.1 Vývojové prostředí Visual Studio

Visual Studio je vývojovým prostředím od společnosti Microsoft. Je to velmi komplexní vývojové prostředí vhodné pro programování nejen aplikací postavených na technologiích od Microsoftu. Aktuálně nejnovější verze je 2019 a jako všechny předchozí verze je dostupná také ve verzi Community, která je zdarma.

Typů aplikací, které je možné ve Visual Studiu programovat, je opravdu velká škála, od aplikací pro mobilní zařízení, přes hry v programovacím jazyku C++ nebo enginu Unity, až po aplikace pro datovou vědu (data science).

Visual Studio podporuje velké množství programovacích, ale i značkovacích jazyků a při programování webových aplikací má vývojář k dispozici podporu pro všechny potřebné technologie. Pro rozšíření funkcionality tohoto vývojového prostředí je možné instalovat různá rozšíření z open source repositáře, kde je možné najít placené i neplacené rozšíření.

Při vývoji webových aplikací s použitím frameworku ASP.NET Core je možné do Visual Studia stáhnout kompletní balíček, který obsahuje všechny závislosti, včetně webového

serveru a Entity Frameworku Core. Stejně jako u rozšíření pro samotné vývojové studio jsou k dispozici také rozšíření pro aplikace v podobě NuGet balíčků. Pro ASP.NET Core jsou k dispozici například balíčky pro tvoření fotogalerií, balíček pro podporu tvorby komplexních LINQ dotazů nebo balíček SQL server, který byl při realizaci této aplikace použit.

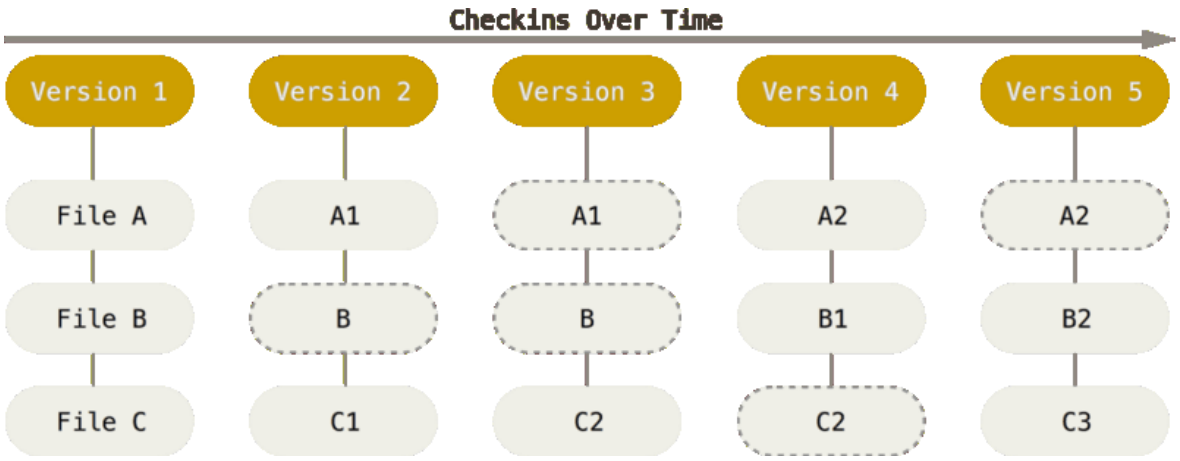
Visual Studio má také výborné debugovací nástroje, které pomáhají vývojářům v odhalování chyb v kódu, diagnostické nástroje pro monitorování běhu aplikace, Intellisense, neboli nápověda a automatické doplňování kódu a spoustu dalších užitečných nástrojů. [17]

2.3.2 Lokální verzovací systém GIT

Verzovací systémy jsou nástroje pro ukládání jednotlivých verzí dokumentů a dalších souborů. Jeden z nejznámějších a nejpoužívanějších lokálních verzovacích systémů je GIT vyvinutý zakladatelem Linuxu Linusem Torvaldsem. GIT kontroluje změny v dokumentech a je inicializován nad konkrétní složkou filesystému, nazývanou pracovní adresář. V každém dokumentu, který je v této složce, včetně zanořených dokumentů, jsou GITem sledovány provedené změny. Při práci s GITem je vytváření nové verze dokumentů třífázovým procesem.

1. **Pracovní adresář** – obsahuje dokumenty a soubory, ve kterých provádíme změny
2. **Staging fáze** – specifikování modifikovaných dokumentů, jejichž změny mají být součástí nové verze
3. **Nová verze** - vytvoření nové verze a uložení do GIT repositáře

GIT ukládá do každé nově vytvořené verze informace o všech sledovaných souborech ve složce. Dokumenty, ve kterých nebyli provedeny žádné změny, nebo neprošli staging fází, jsou do nové verze uloženy pouze jako reference na předchozí, nemodifikovaný soubor.



Obr. 11. Ukládání nových verzí v GITu [18]

Opakem způsobu udržování jednotlivých verzí dokumentů jsou tzv. „Delta“ verzovací systémy, které do každé nové verze ukládají pouze rozdíl mezi původním a modifikovaným dokumentem a nezměněné soubory nejsou součástí nové verze.

Nadstavbou GITu jsou také vzdálené repositáře, např. GitHub, které usnadňují práci v týmech a umožňují sdílení kódu. Většina vývojových prostředí mají integrovanou podporu pro verzovací systém GIT a také pro vzdálený repositář a umožňují tak správu jednotlivých verzí přímo z vývojového prostředí. [18]

3 HROZBY ÚTOKU NA WEBOVOU APLIKACI

3.1 SQL Injection

Velmi nebezpečný útok, který hrozí webovým aplikacím pracujícím s SQL databázemi, je SQL Injection. Při tomto útoku dochází k manipulaci a přerušení SQL dotazu a navázáním dalšího škodlivého dotazu může dojít k přidání nových dat, k modifikaci, smazání nebo vytažení potřebných dat z databáze.

Náchylné na tento typ útoku jsou aplikace, které konstruují SQL dotazy přímo ze vstupu uživatele (přes formulář), který není ošetřen. Jednotlivé parametry (vstupy) jsou v SQL dotazu zapsány v uvozovkách a pokud útočník do formulářového pole zapíše další uvozovku, je dotaz ukončen a může být vykonán další, škodlivý dotaz.

3.2 Cross Site Request Forgery (CSRF)

CSRF, neboli také útok podvržením požadavku, je typ útoku, u kterého útočník podstrčí uživateli škodlivý obsah (odkaz, formulář), kterému může uživatel důvěřovat a nevědomě provést požadovanou akci.

Útočník musí dobře znát napadenou aplikaci a vědět, kam má útok směřovat. Využívá se skutečnosti, že uživatel je přihlášený k nějaké webové službě a ve svém prohlížeči má uložený soubor cookie, který při opětovné návštěvě napadené aplikace zajistí, že je uživatel stále autentizován. Samotný útok už je poté realizován například posláním škodlivého odkazu uživateli přes e-mail, nebo v kombinaci s útokem Cross Site Scripting (XSS) může být takový odkaz, či formulář zobrazen na webové stránce, kterou uživatel dobře zná a důvěřuje jí. Pokud bychom se bavili například o nějakém blogu, kam by uživatel napsal komentář a potvrdil jeho odeslání, mohl by se při odeslání také aktivovat script, který by odeslal skrytý škodlivý formulář.

3.3 Cross Site Scripting (XSS)

Útoky XSS se týkají zejména neošetřených formulářových polí, u kterých má uživatel možnost zadávat prvky HTML. S takovým formulářem je možné odeslat také Javascript kód, který může být uložen do databáze, a pokud obsah databáze slouží také jako výstup při renderování jednotlivých stránek aplikace pro ostatní uživatele, tento script se může v prohlížeči spustit a modifikovat strukturu HTML elementů na stránce, znemožnit použití stránky nebo s kombinací CSFR útoku posloužit pro odeslání škodlivého formuláře. [19]

II. PRAKTICKÁ ČÁST

4 ROZBOR A ANALÝZA POŽADAVKŮ

V první fázi realizace praktické části bylo nutné definovat, jakou funkcionalitu systému uživatelé požadují, co by měl systém pro správu nemovitostí nabídnout potencionálnímu uživateli, a jak bude probíhat interakce uživatele se systémem.

Při rozboru a analýze požadavků byly výchozím bodem obecné předpoklady pro webovou aplikaci podobného typu a základní funkcionalita je víceméně totožná s tím, co nabízí ostatní inzertní portály.

4.1 Požadavky

Prvním krokem při navrhování softwaru je definice požadavků na systém, které se dělí obecně dělí na funkční a nefunkční požadavky.

4.1.1 Funkční požadavky

Pomocí funkčních požadavků je definována funkcionalitu systému. Funkční požadavky pro realitní portál byly pro přehlednost rozděleny do tří hlavních balíčků:

- **Správa nemovitostí**
- **Správa uživatelů**
- **Obecné požadavky**

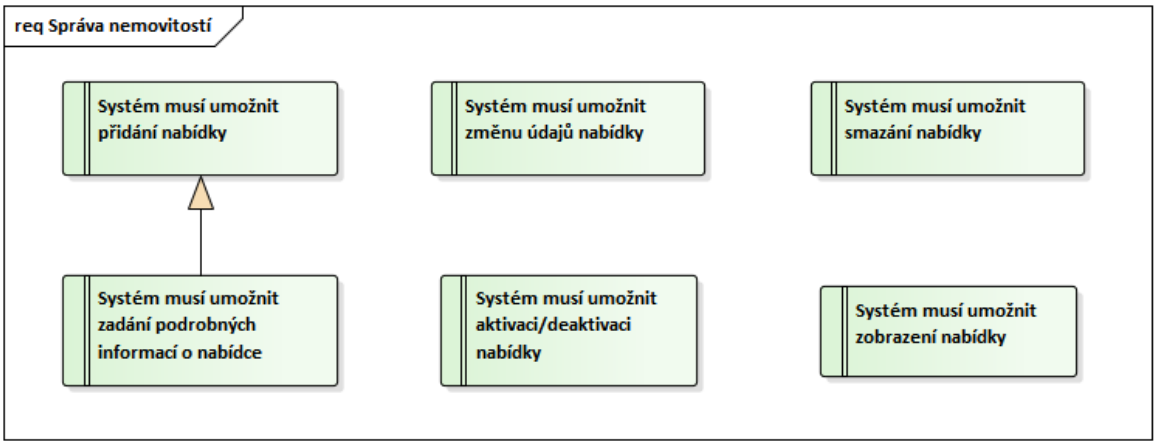
Správa nemovitostí

Balíček požadavků pro správu nemovitostí pokrývá základní požadavky pro vytvoření nové nabídky s nemovitostí, editaci, smazání a zobrazení nabídky s nemovitostí.

Při definování požadavku na vytvoření nové nabídky s nemovitostí bylo cílem, aby nebyl uživatel až tak moc závislý na implementaci databázového modelu pro nemovitost. Z tohoto důvodu byl tento požadavek rozšířen o další požadavek, který definuje, že „systém musí umožnit zadání podrobných informací o nabídce“. Znamená to, že při vytváření nové nabídky má uživatel k dispozici „key - value“ páry, do kterých může zadat informace o nabídce, které nejsou pokryty v základním databázovém modelu nemovitosti. Pokud bude chtít například k nabídce zadat informaci o tom, že na zahradě nemovitosti je k dispozici pergola, může to jednoduše zapsat jako „Pergola – dřevěná“. Tyto doplňující informace se také promítají v grafickém rozhraní aplikace a získávají tak větší pozornost než textový popis.

Požadavkem na změnu údajů nabídky bylo stanoveno, že uživatel musí mít možnost změnit libovolné údaje nabídky s nemovitostí, kterou on sám vytvořil, a také musí mít možnost odstranění nabídky ze systému.

Dalším požadavkem pro správu nemovitosti je požadavek na skrývání jednotlivých nabídek. Může jednoduše nastat situace, kdy potřebuje majitel nemovitosti skrýt nabídku, například z důvodu rezervace nemovitosti, jejíž prodej/pronájem ještě není stoprocentní.



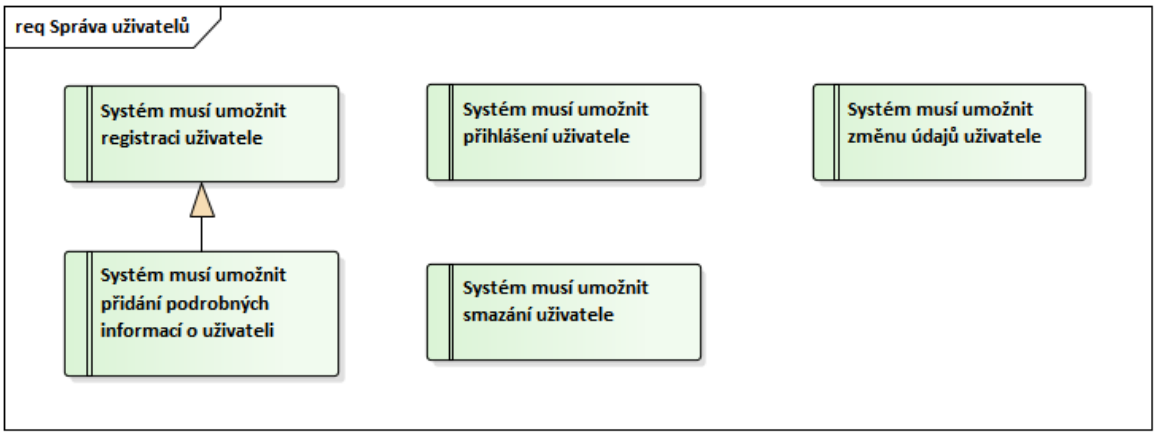
Obr. 12. Funkční požadavky na správu nemovitostí

Správa uživatelů

Uživatelské účty jsou aplikaci zaměřeny na uživatele, kteří budou na portálu inzerovat své nabídky s nemovitostmi. Pro ostatní uživatele nemá uživatelský účet co nabídnout.

Požadavky na správu uživatelů jsou obdobné, jako tomu bylo u požadavků na správu nemovitostí. Jsou zde požadavky na vytvoření, editaci, smazání, mimo to ale také na přihlášení a dohlášení uživatele.

Strategie pro registraci nového uživatele je taková, že se uživatel zaregistruje pouze pomocí nezbytných údajů pro jeho identifikaci – e-mailu a hesla a v případě, že má zájem o sobě zveřejnit více informací, si může doplnit tyto informace ve správě svého uživatelského účtu. Tímto způsobem má uživatel určitou „volnost“ a může se rozhodnout, které informace o sobě zveřejní, a které naopak ne.



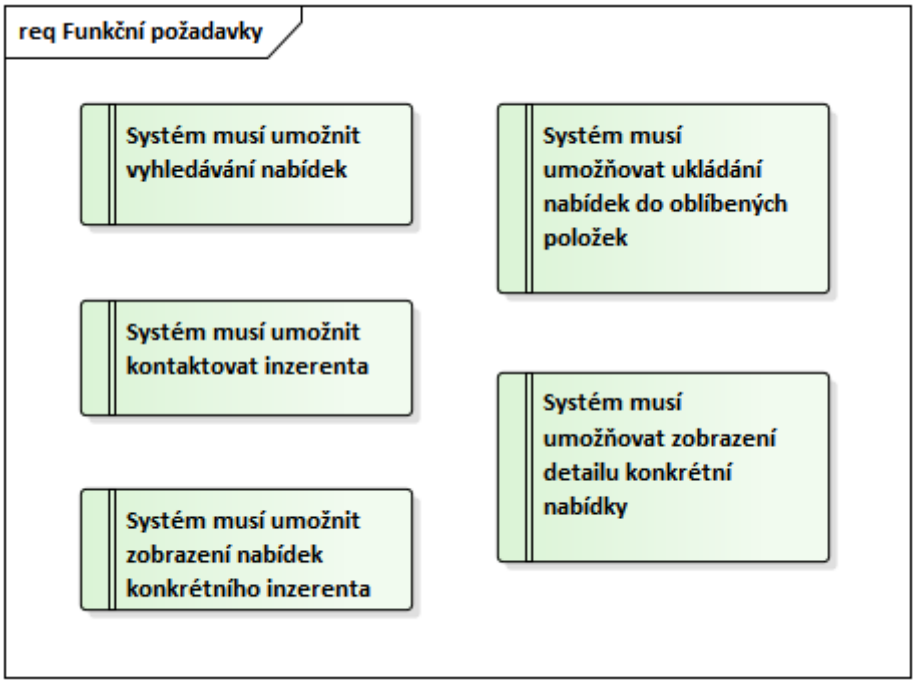
Obr. 13. Funkční požadavky na správu uživatelů

Obecné požadavky

Obecné požadavky cílí na uživatele v roli zákazníka, resp. uživatele, který si prohlíží nemovitosti, popřípadě má zájem nějakou nemovitost koupit nebo pronajmout.

Nejdůležitějším požadavkem pro zákazníka je snadná navigace mezi velkou spoustou nemovitostí a umožnit mu filtrování podle velké škály parametrů nebo podle konkrétního inzerenta. Požadavek na zobrazení nabídky musí zákazníkovi umožnit zobrazení detailní stránky konkrétní nemovitosti.

Dále byl ke zlepšení komunikace mezi zákazníkem a inzerentem zaveden požadavek na možnost kontaktování inzerenta přímo v rámci aplikace, typicky přes kontaktní formulář. Posledním obecným požadavkem je požadavek na možnost ukládání nemovitostí do osobního seznamu.

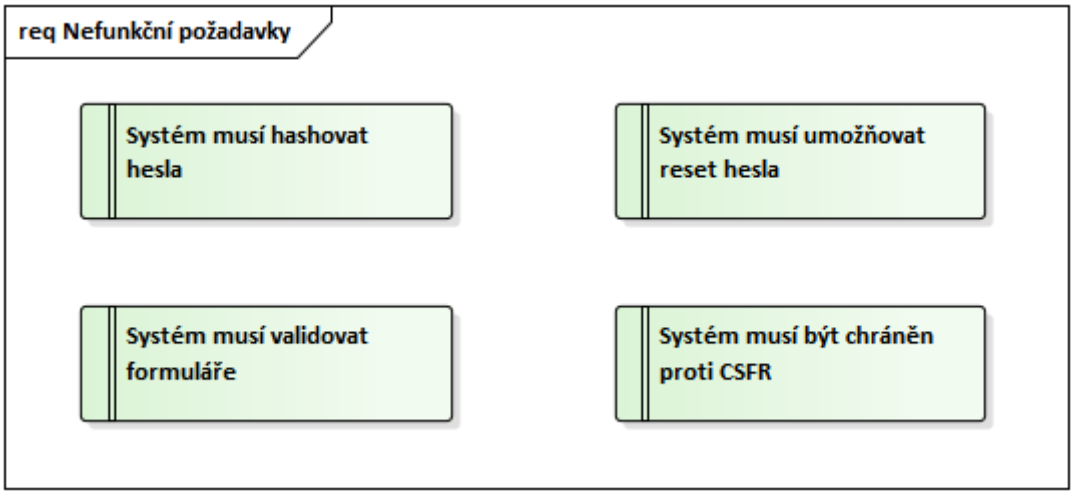


Obr. 14. Obecné funkční požadavky

4.1.2 Nefunkční požadavky

Nefunkční požadavky vznáší požadavky na systém z hlediska bezpečnosti, legislativy, efektivnosti systému apod.

Z hlediska bezpečnosti uživatelů byly implementovány bezpečnostní požadavky na ukládání hesel k uživatelským účtům v hashované podobě, umožnění uživatelům obnovit heslo v případě jeho zapomenutí a chránit je proti CSFR útokům. Pro ověření „správnosti“ dat odeslaných na server uživatelem systému se musí provádět validaci dat.



Obr. 15. Nefunkční požadavky

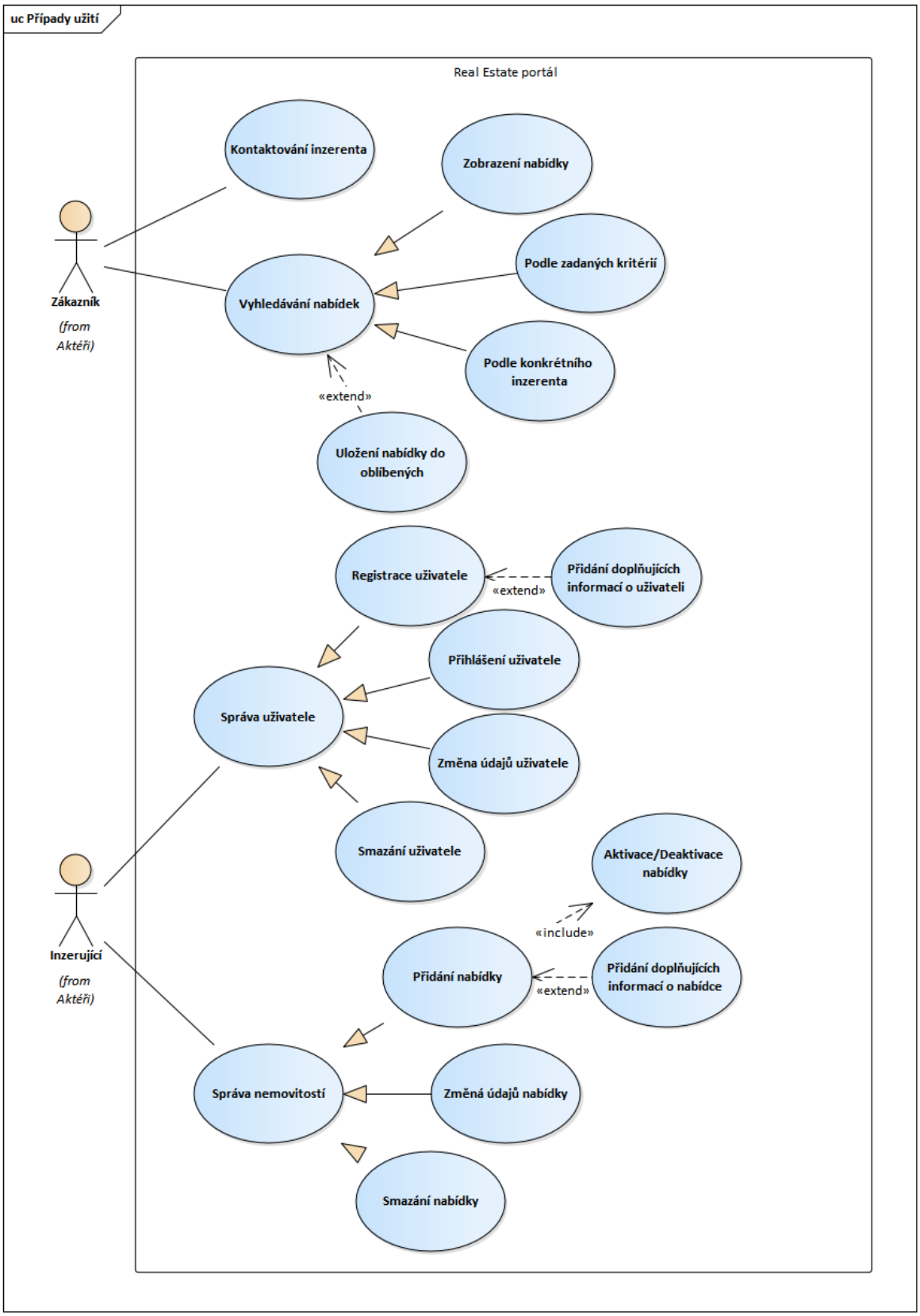
4.2 Use case model

Po vymezení funkcionality systému je důležité zodpovědět otázku, „Jak budou uživatelé systém využívat“. Pro znázornění jejich chování byl navržen model případů užití, který popisuje interakci uživatele se systémem a každou akci popisuje v přesných krocích pomocí scénářů, popřípadě alternativních scénářů, pokud je nutné ošetřit různé chyby nebo může uživatel zvolil více způsobů, jak se dostat k požadovanému výsledku.

Model případů užití úzce koresponduje s požadavky, a zatímco požadavky řeší, co musí systém umět, model případů užití se zaměřuje na to, jak a v jakých krocích je toho dosaženo.

V modelu případů užití je také důležité definovat aktéry, čili entity vystupující v systému. V této konkrétní aplikaci vystupují 2 aktéři:

- **Inzerent –** uživatel, který systém využívá k inzerování nemovitostí
- **Zákazník –** uživatel, který využívá systém z důvodu prohlížení, nakupování, pronajímání nemovitostí



Obr. 16. Use case model

Model případů užití byl opět z důvodu přehlednosti rozdělen do tří hlavních balíčků, podobně jako je tomu u požadavků. Ačkoliv to nejsou balíčky jako takové, jsou v modelu znázorněny pomocí generalizace, tzn. například případ užití „Správa nemovitosti“ je zobecněním případů užití „Přidání nabídky“, „Změna údajů nabídky“, „Zobrazení nabídky“, a „Smazání nabídky“.

K pochopení, jak bude uživatel se systémem interagovat, a jakými kroky musí projít, aby docílil požadovaného výsledku, byl ke každému případu užití napsán jednoduchý scénář, který obecně odráží jednotlivé kroky.

4.2.1 Scénář: Kontaktování inzerenta

Scénář popisuje kroky uživatele v roli zákazníka při kontaktování majitele nemovitosti.

Tab. 1. Scénář kontaktování inzerenta

Název scénáře	Kontaktování inzerenta
Účastníci scénáře	Zákazník, systém
Hlavní scénář	1. Zákazník vyplní kontaktní formulář u požadované nemovitosti 2. Systém provede validaci vstupních dat z formuláře 3. Systém vyřídí požadavek a přepošle zprávu na e-mail inzerenta
Alternativní scénář	2a. V případě nevalidních vstupních dat systém upozorní zákazníka chybovými hláškami

4.2.2 Scénář: Vyhledávání nabídek podle zadaných kritérií

Scénář popisuje zákazníka při vyhledávání nemovitostí podle požadovaných kritérií vyplněním vyhledávacího formuláře.

Tab. 2. Scénář vyhledávání nabídek podle zadaných kritérií

Název scénáře	Vyhledávání nabídek podle zadaných kritérií

Účastníci scénáře	Zákazník, systém
Hlavní scénář	1. Zákazník vyplní formulář určený k vyhledávání nabídek nemovitostí 2. Systém vyhledá v databázi nabídky vyhovující zadaným parametrům 3. Systém zobrazí všechny vyhovující nabídky
Alternativní scénář	4a. Systém upozorní zákazníka chybou hláškou, že nebyla nalezena žádná nemovitost vyhovující zadaným parametrům

4.2.3 Scénář: Vyhledávání nabídek podle konkrétního inzerenta

Scénář popisuje zákazníka při vyhledávání nemovitostí podle konkrétního uživatelského účtu inzerenta.

Tab. 3. Vyhledávání nabídek podle konkrétního inzerenta

Název scénáře	Vyhledávání nabídek podle konkrétního inzerenta
Účastníci scénáře	Zákazník, systém
Hlavní scénář	1. Zákazník rozklikne profil inzerenta 2. Systém zobrazí všechny veřejně dostupné informace o inzerentovi 3. Systém zobrazí všechny jeho aktivní nabídky s nemovitostmi
Alternativní scénář	3a. Systém upozorní zákazníka chybou hláškou, že inzerent nemá žádné aktivní nabídky

4.2.4 Scénář: Zobrazení nabídky

Scénář popisuje uživatele systému při prohlížení konkrétní nabídky s nemovitostí.

Tab. 4. Scénář zobrazení nabídky

Název scénáře	Zobrazení nabídky
Účastníci scénáře	Zákazník, systém
Hlavní scénář	1. Zákazník si po zobrazení výsledků vyhledávání nebo po rozkliknutí profilu inzerenta zvolí nemovitost, jejíž detail chce zobrazit 2. Klikne na odkaz „Zobrazit detail“ 3. Systém zobrazí zákazníkovi detail s požadovanou nemovitostí
Alternativní scénář	3a. V případě chyby (např. souběžné smazání nabídky) systém zobrazí zákazníkovi příslušnou chybovou hlášku

4.2.5 Scénář: Uložení nabídky do oblíbených

Scénář popisuje zákazníka při ukládání nabídky s nemovitostí do seznamu oblíbených nemovitostí.

Tab. 5. Scénář uložení nabídky do oblíbených

Název scénáře	Uložení nabídky do oblíbených
Účastníci scénáře	Zákazník, systém
Hlavní scénář	1. Zákazník si rozklikne konkrétní nabídku s nemovitostí 2. Systém zobrazí zákazníkovi požadovanou nabídku 3. Zákazník klikne na tlačítko "Přidat do oblíbených" 4. Systém zkontroluje, zda-li už nemá zákazník nabídku uloženou

	5. Systém uloží nabídku s nemovitostí do seznamu oblíbených položek
Alternativní scénář	4a. Systém upozorní zákazníka chybou hláškou, že má nabídku již uloženou

4.2.6 Scénář: Registrace uživatele

Scénář popisuje proces registrace nového uživatelského účtu.

Tab. 6. Scénář registrace uživatele

Název scénáře	Registrace uživatele
Účastníci scénáře	Inzerent, systém
Hlavní scénář	1. Inzerent klikne v hlavní nabídce na tlačítko "Registrovat" 2. Systém zobrazí inzerentovi registrační formulář 3. Inzerent se zaregistruje pouze za poskytnutí informací nezbytně nutných k identifikaci uživatele 4. Systém provede validaci vstupních dat 5. Systém zkontroluje, zda-li už neexistuje uživatel se stejným e-mailem 6. Zaregistruje uživatele
Alternativní scénář	4a. V případě nevalidních vstupních dat systém znovu zobrazí formulář s chybovými hláškami 5a. V případě, že existuje inzerent se stejným e-mailem systém znovu zobrazí formulář s chybovými hláškami

4.2.7 Scénář: Přidání doplňujících informací o uživateli

Scénář popisuje přidání dodatečných informací k uživatelskému účtu za účelem lepší komunikace se zákazníkem a větší důvěryhodnosti uživatelského účtu.

Tab. 7. Scénář přidání doplňujících informací o uživateli

Název scénáře	Přidání doplňujících informací o uživateli
Účastníci scénáře	Uživatel, systém
Hlavní scénář	1. Uživatel s uživatelským účtem se v administraci přepne na Správu účtu 2. Systém zobrazí předvyplněný formulář se všemi poskytnutými informacemi o daném inzerentovi 3. Inzerent vyplní doplňující informace o své osobě 4. Systém provede validaci vstupních dat formuláře 5. Systém uloží informace do databáze
Alternativní scénář	4a. V případě nevalidních vstupních dat systém znovu zobrazí inzerentovi formulář s chybovými hláškami

4.2.8 Scénář: Přihlášení uživatele

Scénář popisuje proces přihlášení uživatele s existujícím uživatelským účtem.

Tab. 8. Scénář přihlášení uživatele

Název scénáře	Přihlášení uživatele
Účastníci scénáře	Uživatel, systém
Hlavní scénář	1. Uživatel klikne v hlavní nabídce na tlačítko "Přihlásit se" 2. Systém zobrazí přihlašovací formulář 3. Uživatel se přihlásí pomocí e-mailové adresy a hesla

	4. Systém zkontroluje, zdali uživatelský účet existuje a je zadané správné heslo 5. Přihlásí uživatele do systému
Alternativní scénář	4a. V případě, že je některá z požadovaných informací nesprávná, systém zobrazí uživateli znovu přihlašovací formulář s obecnou chybovou hláškou "Nesprávný e-mail nebo heslo"

4.2.9 Scénář: Změna údajů uživatele

Scénář popisuje proces změny poskytnutých údajů na svém uživatelském účtu

Tab. 9. Scénář změna údajů uživatele

Název scénáře	Změna údajů uživatele
Účastníci scénáře	Uživatel, systém
Hlavní scénář	1. Uživatel s uživatelským účtem přejde v administraci na správu účtu 2. Systém zobrazí předvyplněný formulář s údaji daného uživatele 3. Inzerent upraví ve formuláři požadované informace 4. Systém provede validaci vstupních dat a uloží nová data do databáze
Alternativní scénář	4a. Při nevalidních vstupních datech systém zobrazí uživateli znovu formulář s chybovými hláškami

4.2.10 Scénář: Smazání uživatele

Scénář popisuje proces smazání existujícího uživatelského účtu a všech dat k němu přidružených, včetně nabídek s nemovitostmi.

Tab. 10. Scénář smazání uživatele

Název scénáře	Smazání uživatele
Účastníci scénáře	Uživatel, systém
Hlavní scénář	1. Uživatel s uživatelským účtem přejde v administraci na správu účtu 2. Zvolí "Smazání uživatelského účtu" 3. Systém vyzve uživatele k potvrzení smazání účtu heslem 4. Uživatel zadá heslo a potvrdí smazání 5. Systém smaže inzerenta a jeho nabídky z databáze
Alternativní scénář	5a. V případě špatně zadaného hesla systém vyzve inzerenta k opětovnému zadání

4.2.11 Scénář: Přidání nabídky

Scénář popisuje inzerenta při přidávání nové nabídky s nemovitostí.

Tab. 11. Scénář přidání nabídky

Název scénáře	Přidání nabídky
Účastníci scénáře	Inzerent, systém
Hlavní scénář	1. Inzerent zvolí v administraci "Vytvoření nové nabídky" 2. Vyplní všechna požadovaná pole formuláře a odešle formulář 3. Systém provede validaci vstupních dat 4. Systém uloží nabídku do databáze
Alternativní scénář	4a. V případě nevalidních vstupních dat je inzerentovi znovu zobrazen formulář s chybovými hláškami

4.2.12 Scénář: Změna údajů nabídky

Scénář popisuje inzerenta při editaci již existující nabídky s nemovitostí.

Tab. 12. Scénář změna údajů nabídky

Název scénáře	Změna údajů nabídky
Účastníci scénáře	Inzerent, systém
Hlavní scénář	1. Inzerent v administraci zvolí nabídku, kterou chce editovat 2. Systém zobrazí inzerentovi formulář s předvyplněnými políčky dané nabídky 3. Inzerent provede požadované změny a odešle formulář 4. Systém provede validaci vstupních dat formuláře 5. Systém uloží změny do databáze
Alternativní scénář	4a. V případě nevalidních vstupních dat je inzerentovi znovu zobrazen formulář s chybovými hláškami

4.2.13 Scénář: Smazání nabídky

Scénář popisuje proces mazání nabídky s nemovitostí.

Tab. 13. Scénář smazání nabídky

Název scénáře	Smazání nabídky
Účastníci scénáře	Inzerent, systém
Hlavní scénář	1. Inzerent si v administraci zvolí nabídku, kterou chce smazat 2. Systém zobrazí inzerentovi potvrzovací okno 3. Inzerent potvrdí smazání nabídky 4. Systém smaže požadovanou nabídku z databáze

Alternativní scénář	4a. V případě chyby (např. souběh) systém zobrazí inzerentovi příslušnou chybovou hlášku

5 IMPLEMENTACE APLIKACE PRO REALITNÍ PORTÁL

Webová aplikace tohoto typu vyžaduje velkou spoustu práce, jak na straně klienta (HTML, CSS, Javascript), tak na straně serveru (databáze, implementace logiky serveru). Z tohoto důvodu byla praktická část rozdělena do těchto dvou kategorií, které ale byly vyvíjeny paralelně.

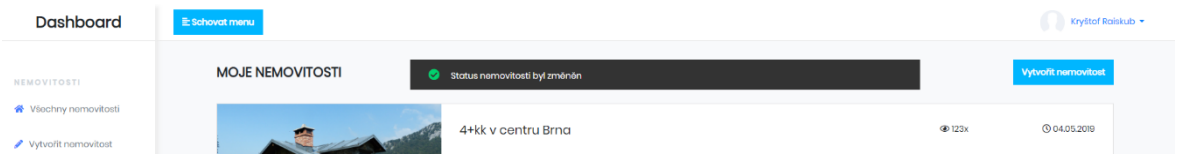
5.1 Klient (Frontend)

Na straně klienta byla snaha přiblížit se co nejvíce dnešním standardům v oblasti aplikací pro realitní portál a obecně v oboru webových aplikací, a proto v aplikaci bylo využito i služeb a nástrojů třetích stran. Jedná se o služby Mapbox pro zobrazování nabídek s nemovitostmi na mapách a geocaching, nebo open source nástroj noUISlider [20] pro tvorbu posouvátek pro zlepšení user experience při filtrování nemovitostí.

Na aplikaci se můžeme dívat ze dvou pohledů – z pohledu zákazníka, který aplikaci využívá z důvodu prohlížení a případného nákupu či pronájmu nemovitostí a na druhé straně z pohledu inzerujícího, který aplikaci využívá z důvodu inzerování nemovitostí.

Každý z těchto pohledů zahrnuje zcela odlišnou strukturu UI, ale obecně grafické zpracování zůstává totožné (barvy, hranatý styl prvků, …).

Co také zůstává pro oba pohledy stejné, je způsob zobrazování chybových hlášek uživateli. Pro zobrazování notifikačních zpráv (např. úspěšné vytvoření nové nabídky) a errorů 404, tedy že stránka nebyla nalezena, se uživateli zobrazují notifikace v podobě „toust messages“ a uživatel je přesměrován na logicky příslušnou stránku. Tímto způsobem je eliminován krok navíc, který musí uživatel provést, aby se ze samostatné stránky 404 dostal zpět, typicky například tlačítkem „Zpět na hlavní stránku“.



Obr. 17. Toast message

Framework ASP .NET Core disponuje funkcí pro tvorbu layoutů, které lze přiřadit k jednotlivým pohledům (view) aplikace nebo ke skupině pohledů. Layouty slouží pro definování základní HTML struktury pro skupiny pohledů a nedochází tak k duplicitě kódu.

Aplikace je rozdělena na dva základní layouty, jeden pro inzerenta a druhý pro zákazníka. Pro každý z těchto layoutů je důležité, aby měl uživatel možnost interagovat se svým uživatelským účtem, pokud jím disponuje. V hlavní nabídce každého layoutu má proto k dispozici parciální pohled, který obsahuje komponentu zobrazující odkazy na přihlášení a registraci v případě nepřihlášeného uživatele, pro přihlášené uživatele se zobrazuje profilová fotka s jménem uživatele a k dispozici má také dropdown menu s odkazy pro rychlou navigaci (zobrazení nemovitostí daného uživatele, správa účtu a odhlášení).

Obr. 18. Parciální pohled pro rychlou navigaci v uživatelském účtu

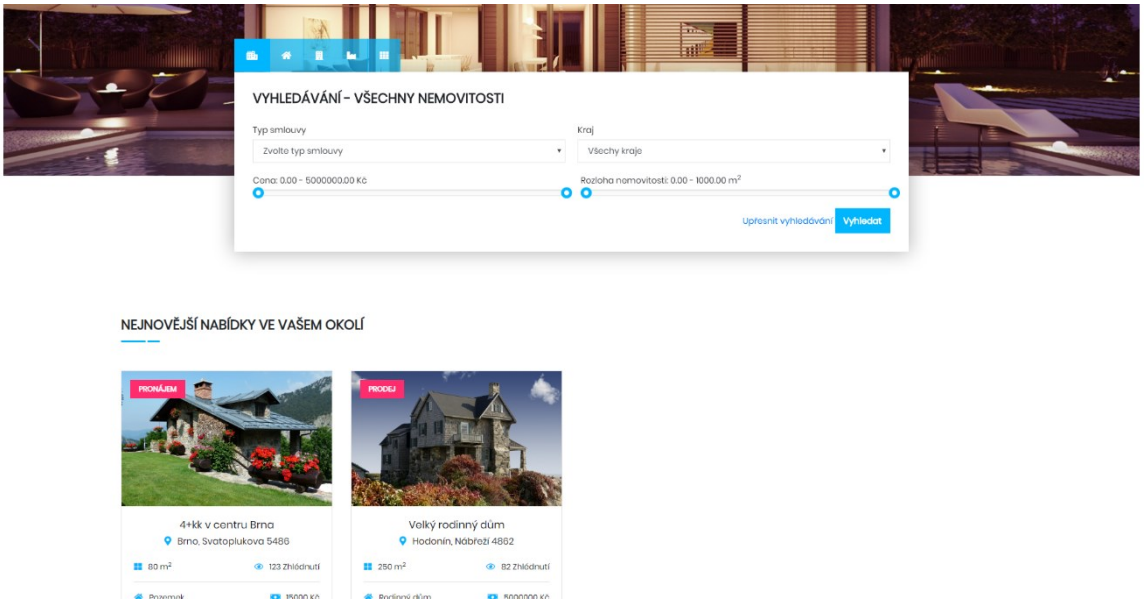
5.1.1 Aplikace z pohledu kupujícího

Layout z pohledu kupujícího obsahuje základní strukturu HTML, scripty a CSS styly, hlavní menu a patičku. Za veškeré další prvky umístěné v pohledech jsou zodpovědné jednotlivé pohledy.

Hlavní stránka

Na hlavní stránce aplikace bylo vytvořeno prostředí, které zákazníkovi, i když nejsou známy jeho preference, nabídne nějakou formu personalizace. Bylo využito HTML5 Navigator API pro získání souřadnic prohlížeče klienta, po zpracování jeho polohy jsou tyto souřadnice odeslány na geocaching API službu Mapboxu, jejíž odpověď obsahuje mimo rozsáhlých geografických informací také kraj, ve kterém se klient právě nachází. Tento údaj je následně za pomocí asynchronního Javascriptu odeslán na vlastní server, který vyfiltruje čtyři nejnovější nabídky v požadovaném kraji, odešle je zpět do prohlížeče a po zpracování odpovědi jsou nemovitosti zobrazeny zákazníkovi. V tomto složitém procesu vyžadujícím několik API požadavků může dojít k jakékoliv neočekávané chybě (např. při navazování spojení s Mapboxem), a proto na jsou na hlavní stránce defaultně zobrazeny čtyři nejnovější nemovitosti z jakéhokoliv kraje.

Zde má také uživatel k dispozici jednoduchý vyhledávací formulář, přes který lze vyhledávat pouze podle základních kritérií, ale může se zde přes tlačítko „Upřesnit vyhledávání“ přesunout na stránku vyhledávání, kde je rozsáhlý formulář pro vyhledávání.

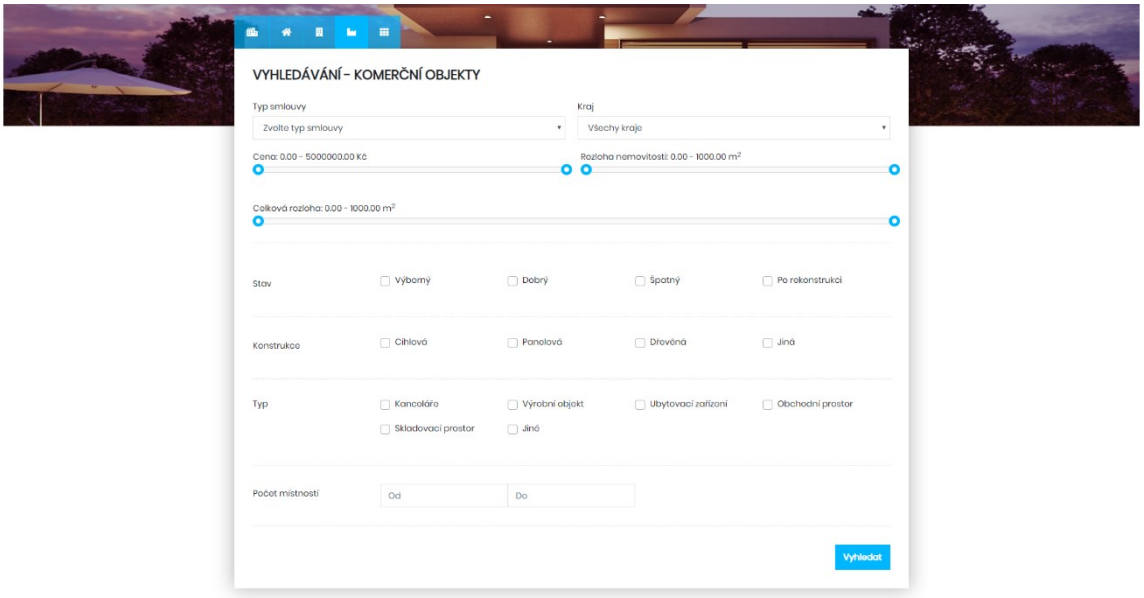


Obr. 19. Hlavní stránka aplikace

Vyhledávání

Stránka pro vyhledávání obsahuje rozsáhlý formulář umožňující zákazníkovi volit mezi velkým množstvím parametrů pro filtrování mezi nemovitostmi. Snažil jsem se zákazníkovi poskytnout co nejrelevantnější výsledky vyhledávání, a proto jsem zapracoval do parametrů vyhledávání skoro všechny atributy z databázového modelu nemovitosti.

Formulář je dynamicky generovaný za pomocí Javascriptu a zákazníkovi jsou tak předloženy k volbě pouze ty parametry, které jsou relevantní pro daný typ nemovitosti.



Obr. 20. Vyhledávací formulář na stránce vyhledávání

Základními parametry pro vyhledávání je typ smlouvy (pronájem, prodej), kraj, cena nemovitosti a její rozloha. Po zvolení konkrétního typu nemovitosti se vygenerují další políčka formuláře reflektující parametry daného typu nemovitosti. Například po zvolení typu nemovitosti „Domy“ se vygenerují políčka celková rozloha pozemku, dispozice nemovitosti, stav nemovitosti a další.

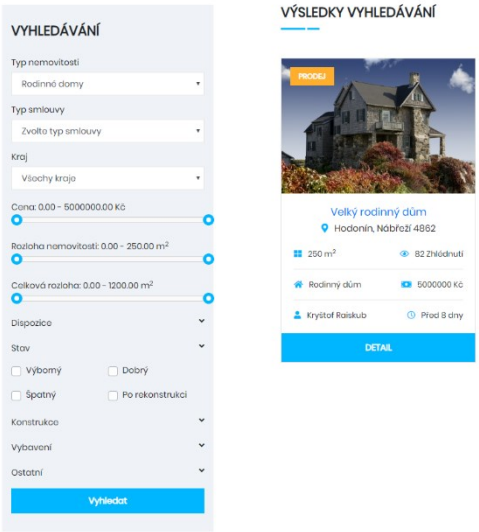
Posouvátka jsou taktéž dynamická a pro každý typ smlouvy a nemovitosti je jejich rozmezí aktualizováno na hodnoty odpovídající hodnotám v databázi.

Uživatel není nijak omezený tím, podle kolika parametrů může vyhledávat a nemusí také vyhledávat podle žádných parametrů a zobrazit si tak všechny dostupné nemovitosti.

Po odeslání formuláře je uživatel přesměrován na stránku s výsledky vyhledávání.

Výsledky vyhledávání

Stránka s výsledky vyhledávání zobrazuje všechny nemovitosti vyhovující zadaným parametrům. Pokud nebyla nalezena žádná vyhovující nemovitost, je uživateli vypsána odpovídající zpráva. Uživatel může kdykoliv změnit kritéria vyhledávání v postranním panelu a nemusí se tak vracet zpět na stránku vyhledávání. Postranní formulář obsahuje předvyplněná políčka již zvolených kritérií, takže uživatel nemusí vyplňovat všechna požadovaná kritéria znovu.



Obr. 21. Výsledky vyhledávání

Detail nabídky s nemovitostí

Po vyhledání nemovitostí si může uživatel zobrazit detailní výpis libovolné nemovitosti. Detailní výpis obsahuje veškeré informace, které inzerent o nemovitosti zveřejnil. Uživatel v roli zákazníka tak získá přehled o základních, ale i podrobných informací o nemovitosti, včetně informací o inzerentovi, občanské vybavenosti a galerie, pokud tyto informace inzerent zveřejnil.

Aby zákazník získal představu o tom, kde přesně se nemovitost nachází, za pomocí služeb třetích stran, konkrétně Mapboxu, bylo implementováno zobrazení nemovitosti na mapě. Zdrojem informace pro zobrazení nemovitosti na mapě je adresa nemovitosti, která ale musí být nejprve převedena na GPS (Global Positioning System) souřadnice. Zasláním API požadavku na geocaching (konkrétně zpětného geocachingu pro převod slovní adresy na souřadnice) službu Mapboxu jsou získány zpět tyto souřadnice, kterými je vycentrována mapa a přímo do středu je také vykreslena značka. Mapbox využívá jako podklad pro mapy open source mapy OpenStreetMap, které mají v České republice velmi solidní pokrytí a za předpokladu, že adresa nemovitosti existuje, by se nemělo stát, že se nepodaří místo na mapě zobrazit.

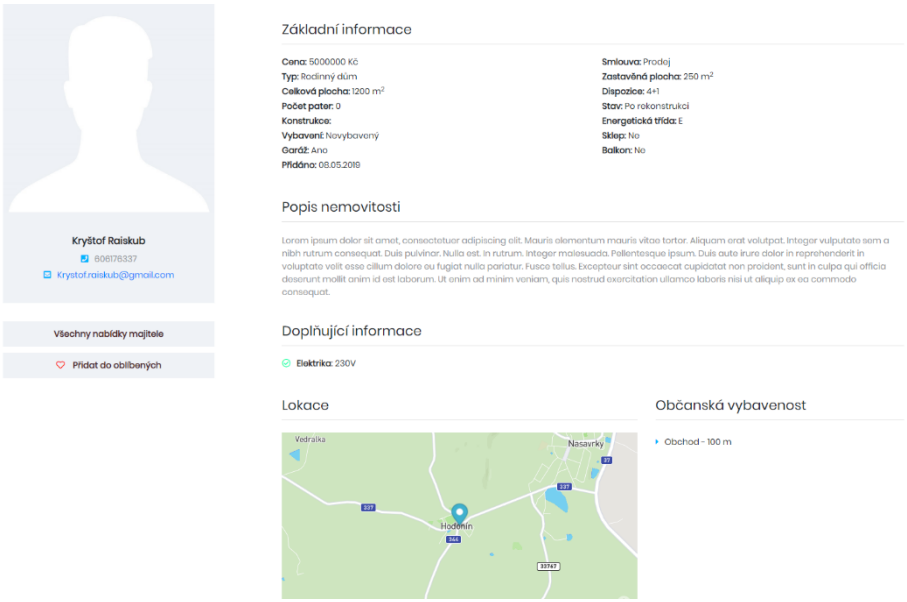
I když má zákazník k dispozici na detailní stránce nemovitosti informace o inzerentovi včetně jeho e-mailu, případně telefonního čísla, bylo by velmi uživatelsky nepřívětivé nedat zákazníkovi možnost projevit zájem o nemovitost. Pro tuto skutečnost je na detailní stránce nemovitosti k dispozici kontaktní formulář, skrze který má zákazník možnost navázat kontakt s inzerentem v rámci aplikace a odeslat mu e-mail. Pro odeslání e-mailu musí zákazník vyplnit své jméno, vlastní e-mail, který inzerentovi poslouží jako adresa pro následnou komunikaci a zprávu, kterou chce inzerentovi zaslat. Pro sjednocení a zorganizování e-mailů v e-mailové schránce inzerenta, které byly zaslány přes realitní portál, byl zaveden jednotný předmět pro všechny zprávy odeslané skrze tento konkrétní formulář. Tímto způsobem se dokáže inzerent lépe zorientovat ve své e-mailové schránce a determinovat, které e-maily jsou od zákazníků z realitního portálu.

Zákazník má také možnost ukládání nabídek do osobního listu oblíbených položek. Ukládání je implementováno přes HTML5 Local Storage API. Local Storage je paměť v prohlížeči, jejíž obsah se oproti cookies nepřenáší na server (větší bezpečnost) a také má daleko větší kapacitu, minimálně 5MB. [21] Každá metoda implementace této funkcionality, ať už pomocí uživatelských účtů, nebo naopak pomocí Local Storage, má své výhody i nevýhody.

Výhodou implementace pomocí Local Storage je, že nejsou uživatelé nuceni k zakládání uživatelských účtů, a že se nevyužívají zdroje serveru (správa uživatelského účtu, požadavky na ukládání, zobrazení, mazání položek, apod.), protože veškerá správa oblíbených položek se provádí přímo v prohlížeči uživatele za pomocí Javascriptu. Na druhou stranu, nevýhodou je vázanost paměti na konkrétní prohlížeč, a tudíž pro uživatele, kteří by si ukládali nemovitosti a poté navštěvovali aplikaci z různých zařízení, je tato metoda nepoužitelná. Z výhod a nevýhod implementace pomocí této metody vyplývají také výhody a nevýhody implementace pomocí uživatelských účtů, tzn. nevýhoda je využívání zdrojů serveru, naopak výhodou je vázanost na uživatelský účet v aplikaci a přístup k uloženým položkám z jakéhokoliv zařízení po přihlášení do systému.

K udržení aktualizovaného listu oblíbených položek je do Local Storage mimo informací o nemovitostech ukládán také záznam s datumem poslední kontroly uložených nemovitostí na serveru. Při každém nahrání stránky se porovná tento záznam s aktuálním časem, a pokud je prodleva mezi poslední aktualizací a aktuálním časem delší než 8 hodin, odešle se na server požadavek, který zkontroluje, zda jsou nemovitosti v seznamu oblíbených položek stále v databázi a inzerent je nesmazal. Do prohlížeče je poté odesláno pole s identifikátory již neexistujících nemovitostí, pomocí kterých je aktualizována paměť v prohlížeči. Ačkoliv se tato kontrola neobejde bez využití zdrojů serveru, v nejhorším případě dojde k odeslání 3 požadavků za den a pokud chceme minimalizovat náklady na server, toto řešení může být skvělou alternativou.

Poslední sekcí na detailní stránce nemovitosti je sekce „Podobné nemovitosti“, která má za úkol udržet pozornost uživatele a nabídnout mu další nemovitosti, které se nachází ve stejném kraji jako právě prohlížená nemovitost.

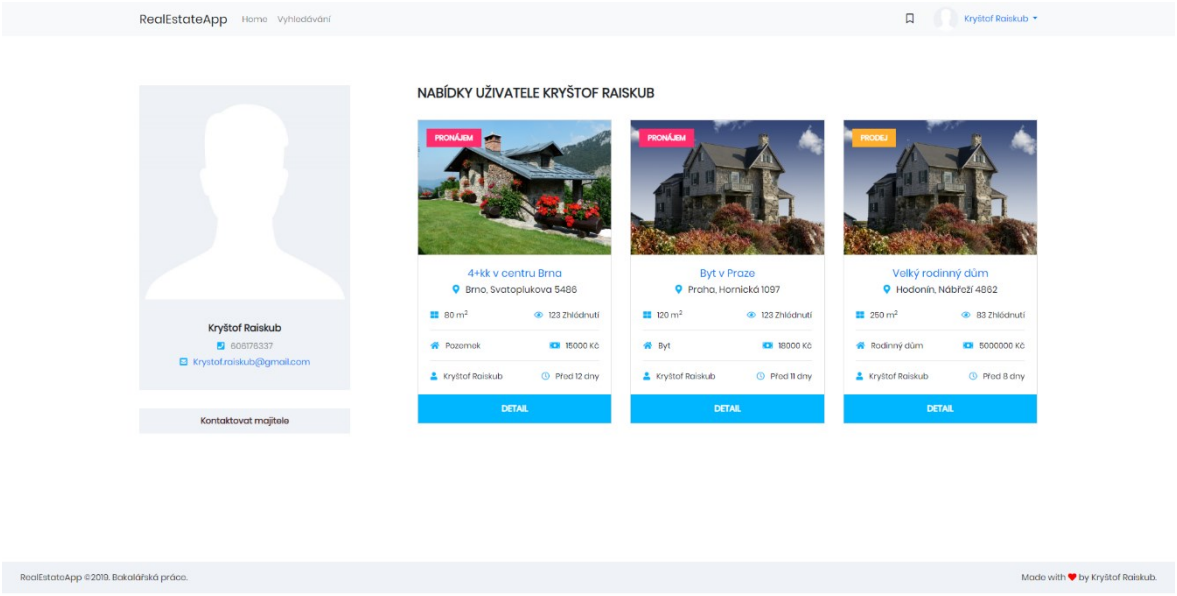


Obr. 22. Detail nabídky s nemovitostí

Detail inzerenta

Z detailní stránky nemovitosti si může uživatel po kliknutí na odkaz „Všechny nabídky uživatele“ zobrazit také všechny nemovitosti, které má inzerent v nabídce. Mimo nabídky tato stránka také slouží jako profil inzerenta a obsahuje všechny informace, které o sobě inzerent zveřejnil. Zákazník se zde může dostat jak k základním kontaktním údajům, tak také v dnešní době ke stále více využívanějším komunikačním kanálům jako jsou sociální sítě, přes které bývá komunikace obvykle rychlejší než přes e-mail.

Stejně jako z detailu nabídky má zákazník také možnost kontaktovat inzerenta skrze kontaktní formulář, který je založen na stejném principu jako u detailu nabídky. Zákazník tedy vyplní pouze své jméno, e-mail, zprávu a jako předmět zprávy je opět zvolen jednotný předmět pro lepší organizaci v e-mailové schránce inzerenta. Tento kontaktní formulář je vhodnější spíše pro obecnější dotazy, které nesměřují k žádné nemovitosti.



Obr. 23. Detail inzerenta

5.1.2 Aplikace z pohledu inzerenta

Inzerent má k dispozici kompletní dashboard, který slouží jako rozcestník při správě uživatelského účtu a nabídek s nemovitostmi. Inzerent může z dashboardu vytvářet, editovat, mazat nabídky, upravovat svůj uživatelský profil, měnit heslo a smazat svůj účet.

Základní struktura layoutu pro dashboard obsahuje postranní nabídku, která je pro účely responsivního designu zasouvací, aby na mobilních zařízeních s malým displejem nezakrývala hlavní obsah. Nabídka je rozdělena do sekcí „Nemovitosti“ obsahující odkazy na stránky pro správu nemovitostí, následně sekce „Profil“, z které se může inzerent navigovat na stránky pro správu účtu a v poslední řadě dvě navigační tlačítka pro opuštění dashboardu a odhlášení.

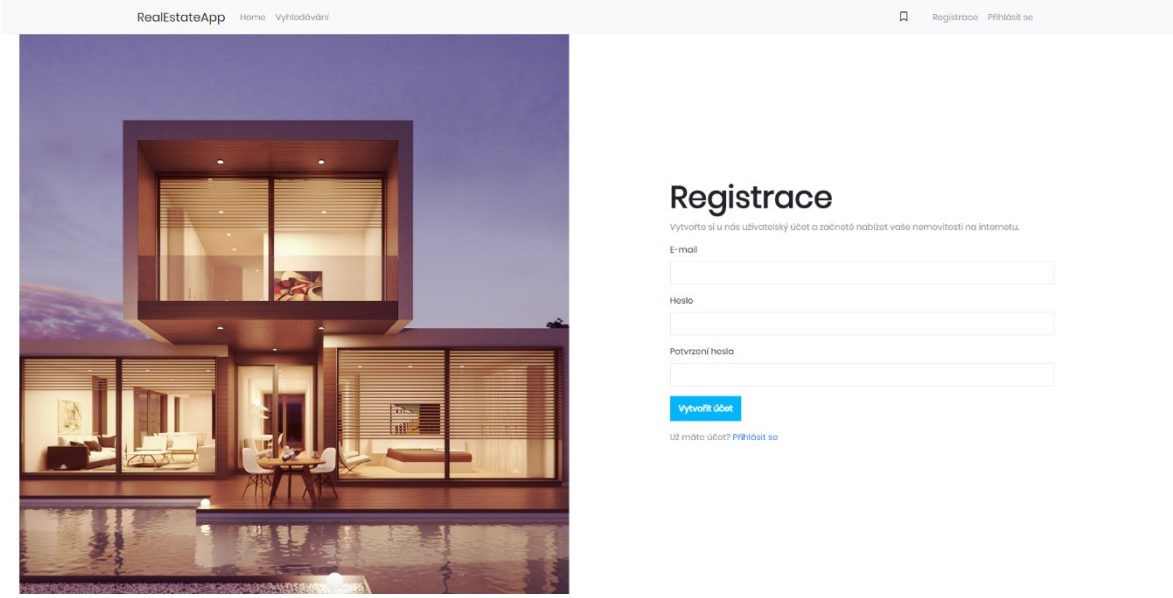
Registrace nového uživatelského účtu

Pro uživatelské účty byla implementována pouze lokální strategie a není tedy možné se přihlásit přes vzdálené poskytovatele, jako je například Google nebo Facebook. K přihlášení a registraci má uživatel k dispozici odkazy v hlavní nabídce layoutu.

Při registraci nového uživatele musí uživatel vyplnit unikátní e-mail, heslo a potvrzení hesla. Pokud již existuje uživatelský účet se stejným e-mailem nebo jsou vstupní data nevalidní, je uživatel upozorněn odpovídající chybovou hláškou. Pro volbu hesla je systém nakonfigurován benevolentněji a jediné omezení, které musí uživatel dodržet, je minimální počet šesti znaků hesla. Ačkoliv je vhodné uživatele chránit a vyžadovat po nich bezpečná hesla, každý

z uživatelů už má pravděpodobně vybudovaný svůj vlastní set hesel, které používá napříč různými službami a aplikacemi a vyžadování speciálních znaků, číslic a dalších znaků v hesle může vést pouze ke zvýšení míry zapomenutí hesla.

Aplikace nevyžaduje v procesu registrace po uživateli více informací, než je nezbytně nutné k jeho identifikaci a přihlašování, uživatel má ale následně možnost přidání dalších informací o svém účtu ve správě účtu.



Obr. 24. Registrační formulář

Přihlášení do aplikace

Grafické rozhraní a struktura pro stránku „Přihlášení“ je totožná se stránkou pro registraci, uživateli je ale namísto registračního formuláře předložen formulář pro přihlášení. Pro přihlášení do systému je uživatel vyzván k zadání přihlašovacího e-mailu a hesla. Má také možnost specifikovat, zda chce „být zapamatován“ a v tomto případě se cookies s autorizačním klíčem uloží do prohlížeče persistentně a přihlášení uživatele přetrvá i po zavření a znovuotevření prohlížeče.

Pokud zadá uživatel špatné přihlašovací jméno nebo heslo, je mu vypsána pouze obecná chybová hláška „Heslo nebo přihlašovací jméno nejsou správné“. V případě bruteforce ataků útočník tímto způsobem nezjistí, zda se mu podařilo najít e-mail nebo heslo, které patří k nějakému existujícímu uživatelskému účtu.

Resetování hesla k uživatelskému účtu

Uživatel má v případě, že zapomene heslo ke svému uživatelskému účtu možnost obnovit heslo pomocí e-mailu, který použil pro registraci. Ze stránky „Přihlásit se“ může kliknout na odkaz „Zapomněli jste heslo?“, který uživatele přesměruje na stránku s opět stejným grafickým rozhraním a strukturou, jako tomu bylo v případě přihlášení a registrace. Zde uživatel vyplní e-mail, který náleží jeho uživatelskému účtu, na který je mu následně vygenerován a poslán zašifrovaný token v podobě klasického hypertextového odkazu, obsahující jedinečné identifikátory jeho uživatelského účtu.

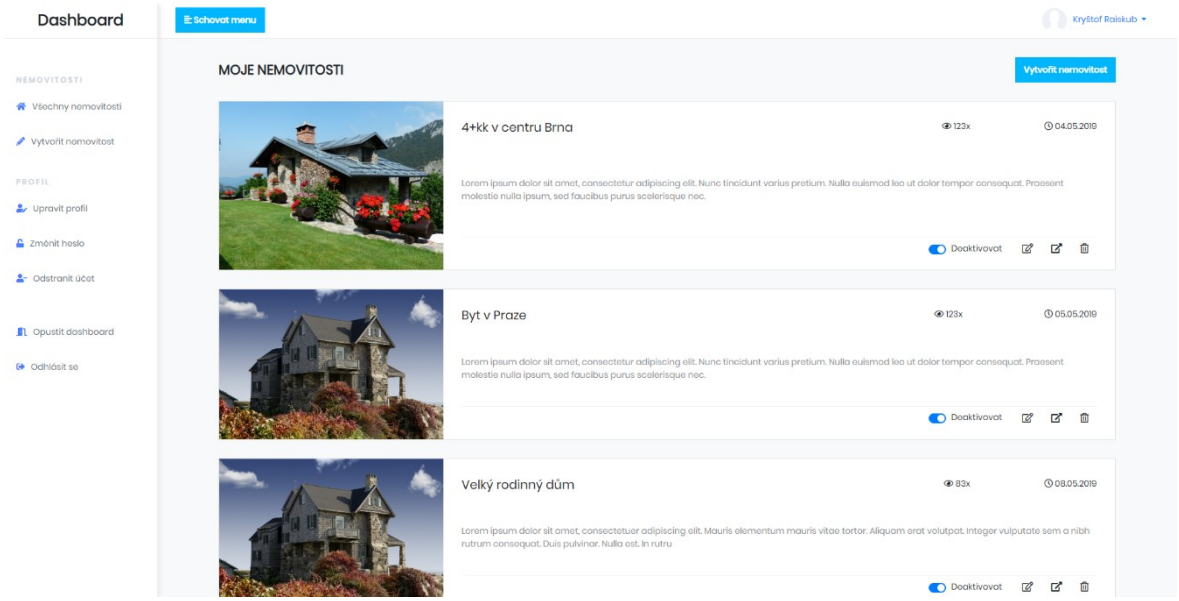
Po rozkliknutí odkazu v e-mailu je uživatel přesměrován zpět do aplikace, kde si za předpokladu, že má správný token, kterému ještě nevypršela platnost, může resetovat heslo.

Hlavní stránka dashboardu

Poté, co se uživatel přihlásí ke svému uživatelskému účtu je automaticky přesměrován do dashboardu na hlavní stránku.

Tato stránka obsahuje výpis všech nabídek daného uživatele a každá položka seznamu obsahuje základní informace pro identifikaci nabídky (nadpis, popis, fotografie, datum přidání) a také statistický údaj o počtu zobrazení nabídky. Ve spodní liště jsou dále umístěny tři navigační tlačítka sloužící pro editaci, smazání, zobrazení detailu nabídky a vedle toho také jednoduchý přepínač pro aktivaci a deaktivaci nabídky. Deaktivované nabídky jsou viditelné pouze z dashboardu vlastníka dané nabídky.

Smazání nabídky je dvoufázový proces, při kterém musí uživatel nejprve zvolit nabídku, kterou chce smazat a kliknout na tlačítko s ikonou koše a dále je pomocí vyskakovacího okna vyzván k potvrzení smazání. Tímto způsobem je zabráněno nechtěnému smazání nabídky.



Obr. 25. Hlavní stránka dashboardu

Stránka Nová nemovitost

Tato stránka slouží pro vytvoření nové nemovitosti a obsahuje dynamicky generovaný formulář. První část formuláře je statická a obsahuje políčka formuláře, představující atributy nemovitosti, které jsou pro všechny typy nemovitostí společné. Posledním políčkem v této sekci formuláře je výběr typu nemovitosti, na jehož základě se dynamicky vygeneruje další část formuláře se specifickými atributy pouze pro daný typ nemovitosti.

Poslední část formuláře je opět statická a zde může uživatel zadat libovolný počet objektů občanské vybavenosti, doplňující informace o nabídce, nahrát hlavní fotografii a další, doplňující fotografie.

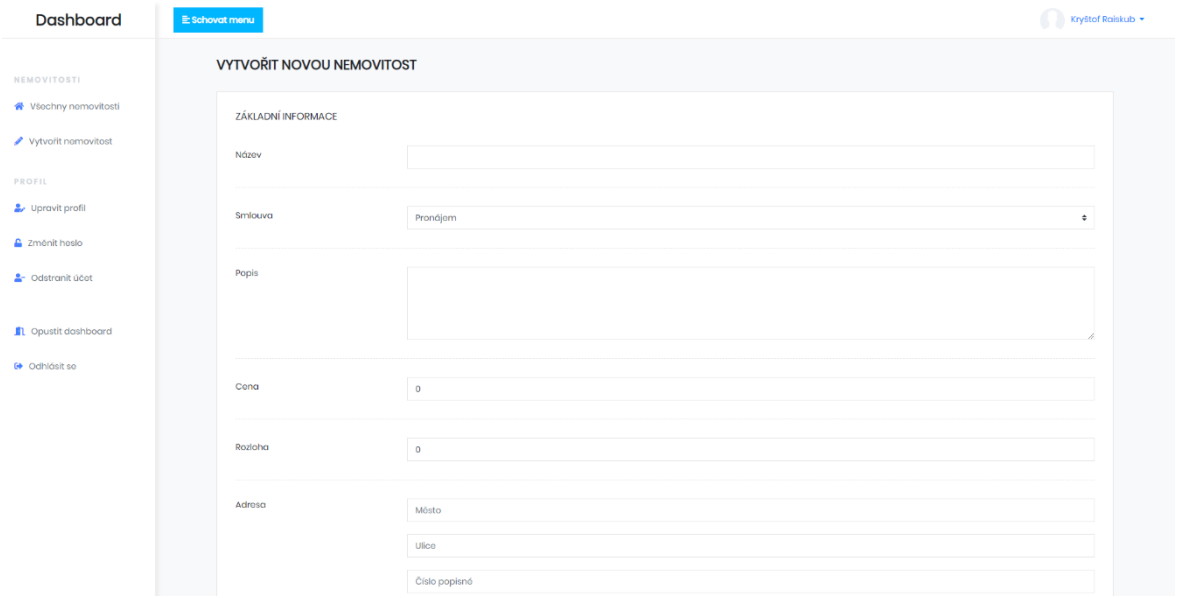
Objekty občanské vybavenosti i doplňujících informací jsou skupiny dvou formulářových polí, které fungují jako páry „klíčů“ a „hodnot“. Pro doplňující informace může být klíčem například „odpad“ a hodnota „městská kanalizace“. Pro objekty občanské vybavenosti může být klíčem například „lékárna“ a hodnotou „450“, čili vzdálenost lékárny od nemovitosti je 450 metrů. Defaultně se zobrazuje pouze jedna skupina políček pro každý typ, uživatel má ale možnosti po kliknutí na tlačítko „Přidat další“ vygenerovat další pár požadovaného typu informace.

Motivací pro doplňující informace bylo, aby nebyl uživatel příliš závislý na databázovém modelu nemovitosti a mohl si specifikovat další vlastní uživatelská pole a rozšířit tak základní model nemovitosti o další důležité informace, které by jinak musel specifikovat

v textovém popisu nemovitosti. Tyto uživatelská pole jsou na detailní stránce zobrazená podobně jako je tomu u základních atributů (v tabulce) a získávají tak větší pozornost.

Pro některé typy atributů (dispozice, stav, konstrukce apod.) vychází ze skutečnosti konečný počet hodnot, kterých mohou nabývat nebo byl tento konečný počet nadefinován autorem aplikace. Na jednu stranu to může být do jisté míry omezení inzerenta, na druhou stranu bylo docíleno jednolitosti hodnot ukládaných do databáze, protože inzerenti nemůžou zapsat jednu hodnotu vícero způsoby. Pokud by měli inzerenti volnou ruku k zapsání dispozice nemovitosti, jeden inzerent by mohl zapsat například 1+kk a druhý 1 + kk (s mezerami), což by značně zhoršilo další manipulaci s daty, například při vyhledávání nemovitostí.

Po odeslání formuláře se provede validace vstupních dat a v případě chybně vyplněných políček jsou uživateli zobrazeny příslušné chybové hlášky, v opačném případě je nemovitost uložena do databáze, uživateli je dána zpětná vazba v podobě úspěšné notifikace a je přesměrován na hlavní stránku dashboardu.



Obr. 26. Formulář pro vytvoření nové nemovitostí

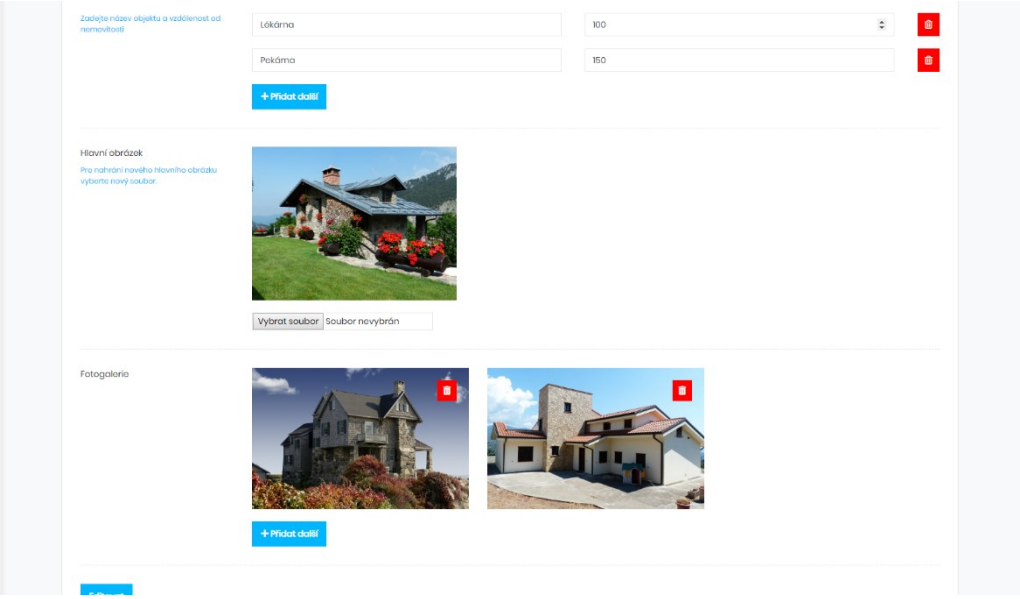
Stránka Úprava nemovitosti

Pokud uživatel potřebuje změnit údaje o nemovitosti, může se ze stránky „Všechny nemovitosti“ přesunout kliknutím na ikonku zápisníčku s tužkou přesunout na editační stránku konkrétní nemovitosti.

Tato stránka obsahuje stejný, dynamicky generovaný, formulář jako stránka „Nová nemovitost“, políčka formuláře jsou ale předvyplněná údaji, které inzerent o nemovitosti

specifikoval. Inzerent má možnost kompletní editace všech údajů. Může tak změnit základní informace, typ nemovitosti a s ní spojené specifické informace, může přidat, či odebrat objekty občanské vybavenosti a doplňujících informací nebo smazat a nahrát nové obrázky do galerie.

Při mazání obrázku z galerie má inzerent k dispozici u každého obrázku v levém horním rohu ikonku koše s červeným pozadím, která funguje pro přidání obrázku do fronty obrázků určených k odebrání z nabídky. Uživatel po kliknutí na tuto ikonku dostane zpětnou vazbu v podobě změny ikonky z koše na ikonku „fajfky“ a zbarvením pozadí do zelena. Po opětovném kliknutí na tuto ikonku se grafické zobrazení ikonky vrátí zpět do původního stavu a obrázek se odebere z fronty obrázků pro odebrání z nabídky.

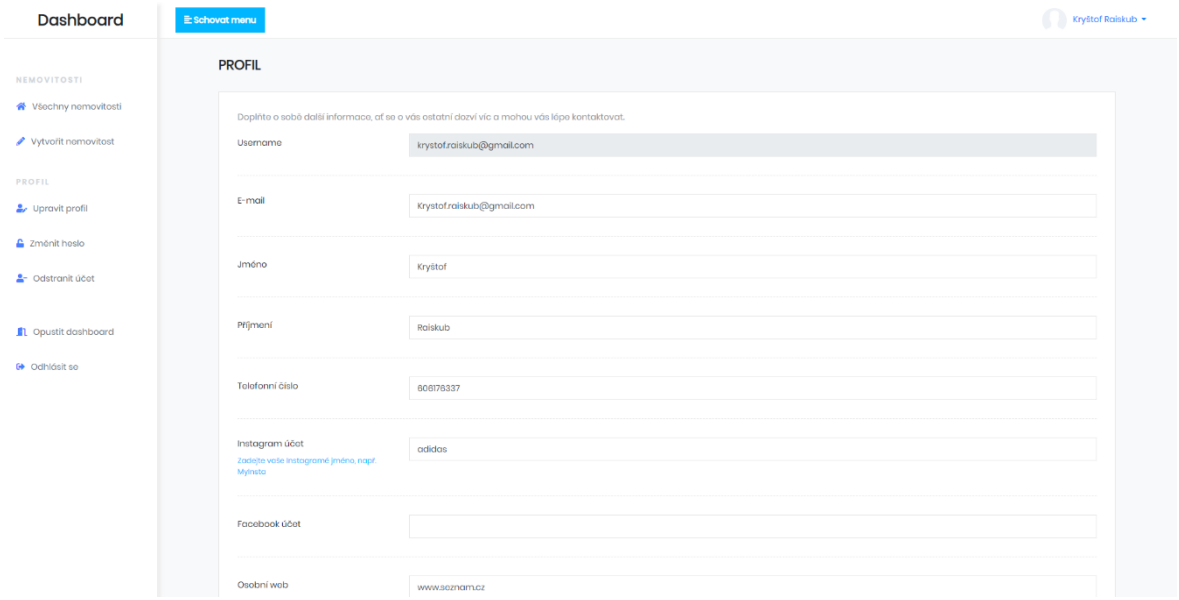


Obr. 27. Editace obrázků a doplňujících informací

Stránka Správa účtu

Po registraci nového uživatele obsahuje nově vytvořený účet pouze e-mailovou adresu, což zároveň funguje jako jediná kontaktní informace pro zákazníky. Pro poskytnutí více kontaktních informací má uživatel na stránce Správa účtu k dispozici formulář, kde může vyplnit další kontaktní údaje, stručný popis „o mně“ a přidat profilovou fotku.

Jako další kontaktní údaje může specifikovat telefonní číslo a z potřeb vycházejících z „dnešního“ obvyklého způsobu komunikace má také možnost zveřejnit odkazy na své profilové účty na sociálních sítích (Instagram, Facebook) a také odkaz na svůj osobní web, pokud jím disponuje.



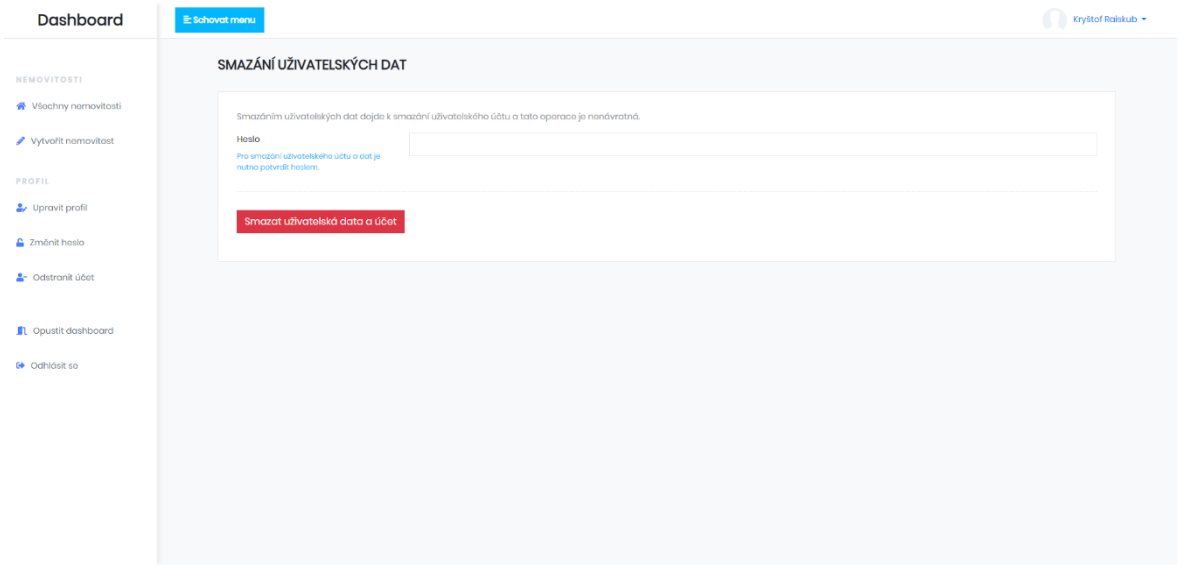
Obr. 28. Správa údajů uživatelského profilu

Stránka Změna hesla

Inzerent má také možnost si kdykoliv změnit heslo ke svému uživatelskému účtu, ať už z vlastní iniciativy, či při pozorovaní podezřelé aktivity na svém účtu. Ke změně hesla musí uživatel zadat aktuální heslo, nové heslo a potvrzení nového hesla.

Stránka Odstranit účet

Každý inzerent s uživatelským účtem má možnost smazání svého účtu. Toto smazání je nenávratná operace, a proto je nutné ji potvrdit svým aktuálním heslem. Po smazání uživatelského účtu se ze systému smažou všechny dostupné informace o daném účtu, včetně nabídek s nemovitostmi a nahrané fotografie.



Obr. 29. Smazání uživatelského účtu

5.2 Backend

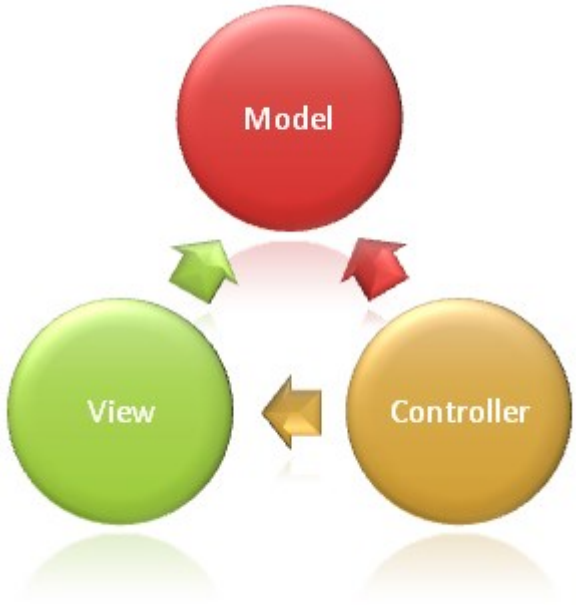
5.2.1 MVC architektura aplikace

Při implementaci logiky serveru bylo z velké části využito architektonického vzoru zvaného „Model – View - Controller“ (MVC). Tento architektonický vzor rozděluje aplikaci do tří komponent a tím zajišťuje oddělení logiky tak, že při změně v jedné z komponent má tato změna malý, ideálně žádný vliv na zbylé komponenty. Tento návrhový vzor zjednodušuje systém jako celek a člení ho na menší, dílčí „subsystémy“, a tím i zvyšuje udržovatelnost kódu.

Model – představuje modely dat, se kterými aplikace pracuje. Při použití objektově-relačního mapování reflektují databázové modely jednotlivých entit vystupujících v aplikaci.

View – pohled – představuje jedno konkrétní grafické rozhraní (např. hlavní stránka aplikace) a stará se o naplnění pohledu daty z modelu a zobrazení výstupu uživateli

Controller – zpracovává interakci uživatele, představuje prostředníka a propojuje komunikaci mezi pohledem a modelem, naplňuje model daty a vybírá příslušný pohled



Obr. 30. Architektura Model - View – Controller [22]

5.2.2 Razor Pages

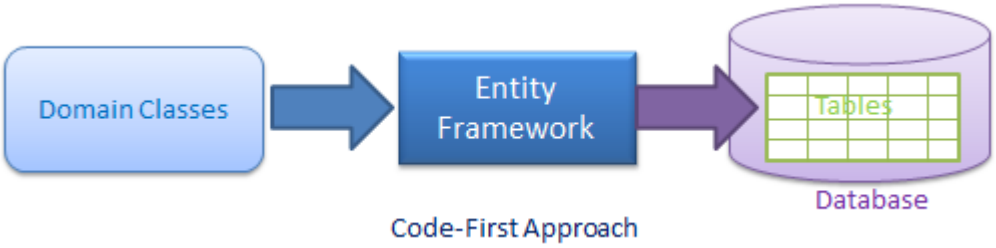
Zbylá část aplikace byla vytvořeno pomocí „Razor pages“, které jsou vhodné pro izolované procesy s jednoduchou manipulací s daty. Každá stránka Razor je skupinou dvou souborů, z nichž jeden je view a druhý je určený pro logiku daného procesu. Stránky Razor umožňují komunikaci s databází, tak jako MVC, ale rozdílem je, že logika a model dat je specifický právě pro jednu konkrétní Razor stránku a není možné je využít nikde jinde v aplikaci. Tímto způsobem je implementovaný celý modul Identity, který je součástí ASP.NET Core a byl v aplikaci použit pro správu uživatelských účtů.

5.2.3 Data Transfer Objects (DTO)

Při práci s daty bylo využíváno „Data transfer object“ (DTO) objektů. Tyto objekty se používají v případě, kdy máme odlišné modely dat pro data reprezentující tabulky databáze a pro data, kterými je naplněn view a zobrazen uživateli. Účelem DTO je minimalizování požadavků při komunikaci se serverem. V situacích, kdy potřebujeme získat komplexní data, složená například z více objektů, bychom museli provést vícero požadavků na získání kompletní informace. Při použití DTO můžeme transformovat komplexní data do jednoho modelu, který vyřídí požadavek najednou. V opačném případě můžeme mít komplexní model dat, ale potřebujeme pouze jeho část. Transformováním na příslušný DTO zjednodušíme tato data a pro vyřízení požadavku využijeme pouze jeho část, čímž dochází ke snížení velikosti přeposílaných dat. [23]

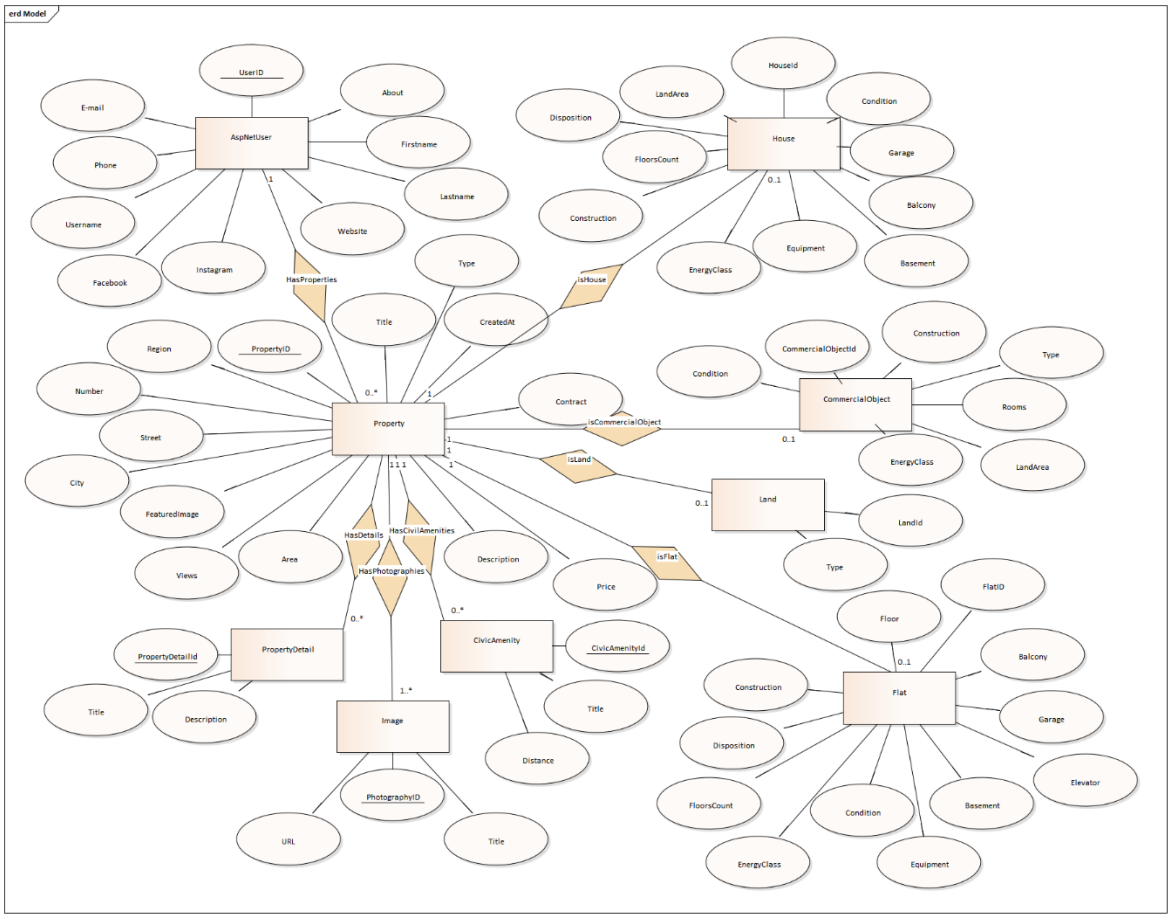
5.2.4 Databáze

Ačkoliv byla databáze vytvořena přístupem „code-first“, pro znázornění dat v aplikaci byl vytvořen také entitně – relační (ER) diagram. „Code-first“ přístup je opakem „database-first“ přístupu, kdy se namísto použití již existující databáze vytvoří schéma databáze s tabulkami a sloupci z korespondujících tříd a atributů, a po aplikování migrace je ze schématu vytvořena nová databáze. Tento přístup značně urychluje proces vývoje aplikace. Entity Framework si udržuje aktuální stav tabulek a sloupců v databázi a v případě změny ve třídách a atributech se po aplikování nové migrace provedou odpovídající změny v databázi.



Obr. 31. Code First přístup [24]

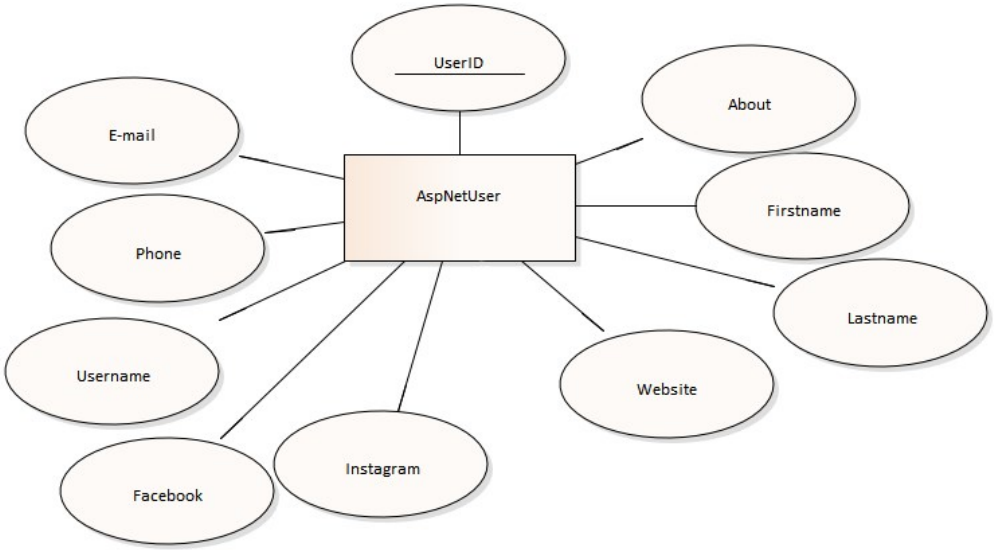
Každá entita v ER modelu představuje tabulku v databázi, která tedy obsahuje tabulky AspNetUsers, Properties, Houses, Flats, CommericalObjects, Lands, Images, CivicAmenites a PropertyDetails. Tabulka AspNetUsers byla spolu s dalšími tabulkami pro ukládání uživatelských rolí, tokenů a dalších vygenerována automaticky při nasazení modulu Identity do aplikace. Aplikace s těmito tabulkami nepracuje, a proto nejsou dále v práci zmiňovány.



Obr. 32. Entitně - relační diagram

Tabulka ASPNetUsers

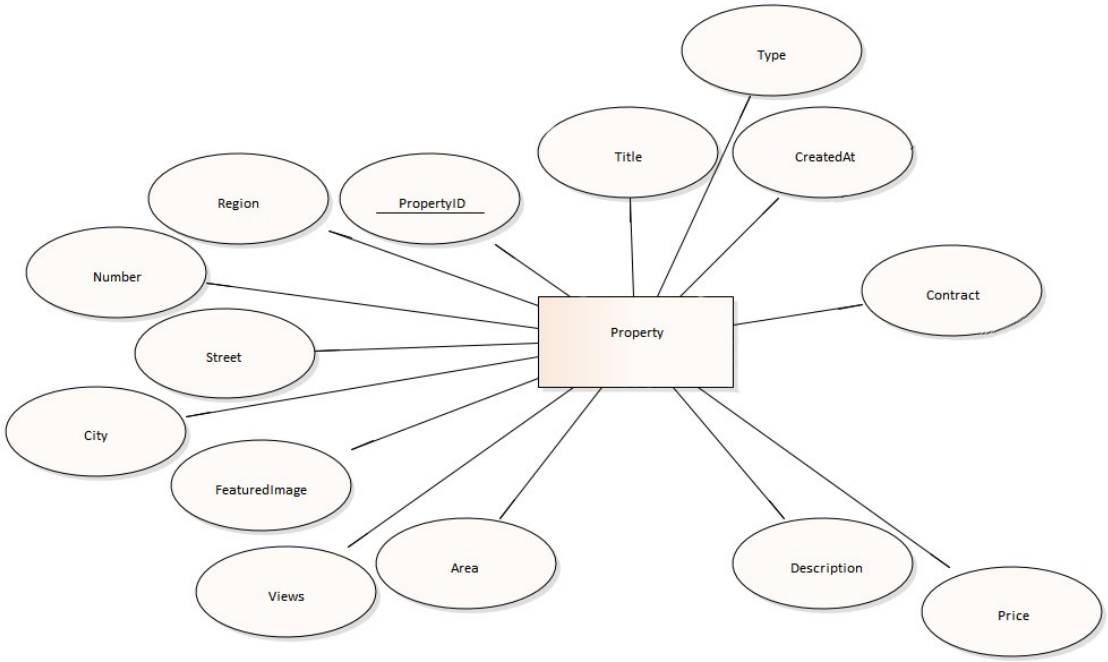
Tabulka reprezentuje uživatelský účet inzerenta a je automaticky vygenerována modulem Identity. Vygenerovaná tabulka obsahuje pouze základní informace o uživateli, jako je jméno účtu, e-mail, telefonní číslo, heslo v hashované podobě a další bezpečnostní atributy, a proto byla pro potřeby aplikace rozšířena o další atributy jméno a příjmení uživatele, krátký popis o uživateli a další kontaktní údaje na sociální sítě.



Obr. 33. Atributy entity AspNetUser

Tabulka Properties

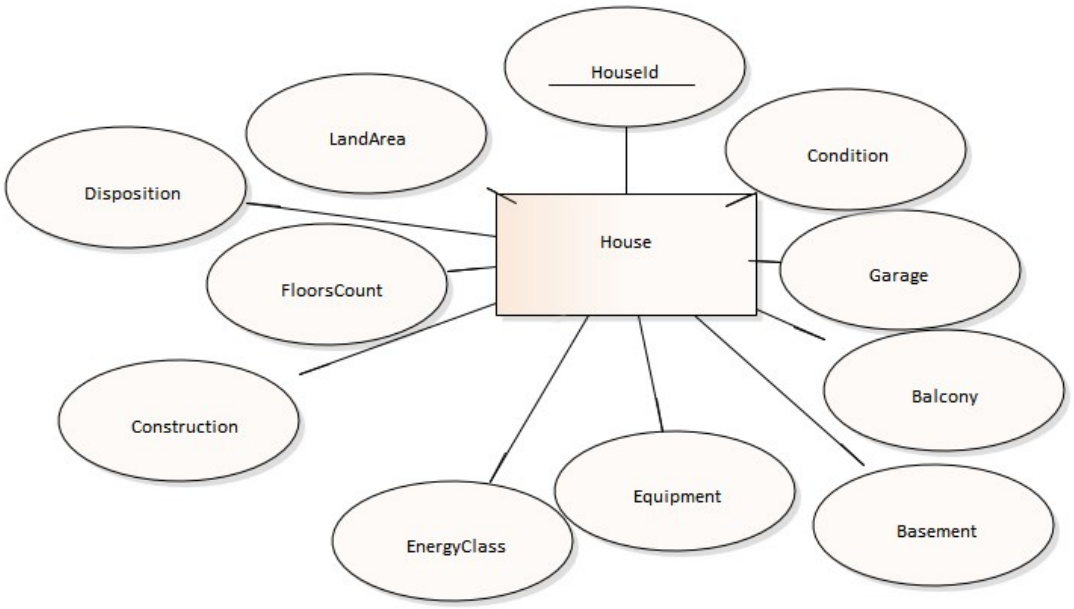
Tabulka slouží jako základní model pro nemovitost. Obsahuje atributy, které jsou pro všechny typy nemovitostí stejné a taky cizí klíč UserId pro propojení tabulek AspNetUsers a Properties.



Obr. 34.Atributy entity Property

Tabulka Houses

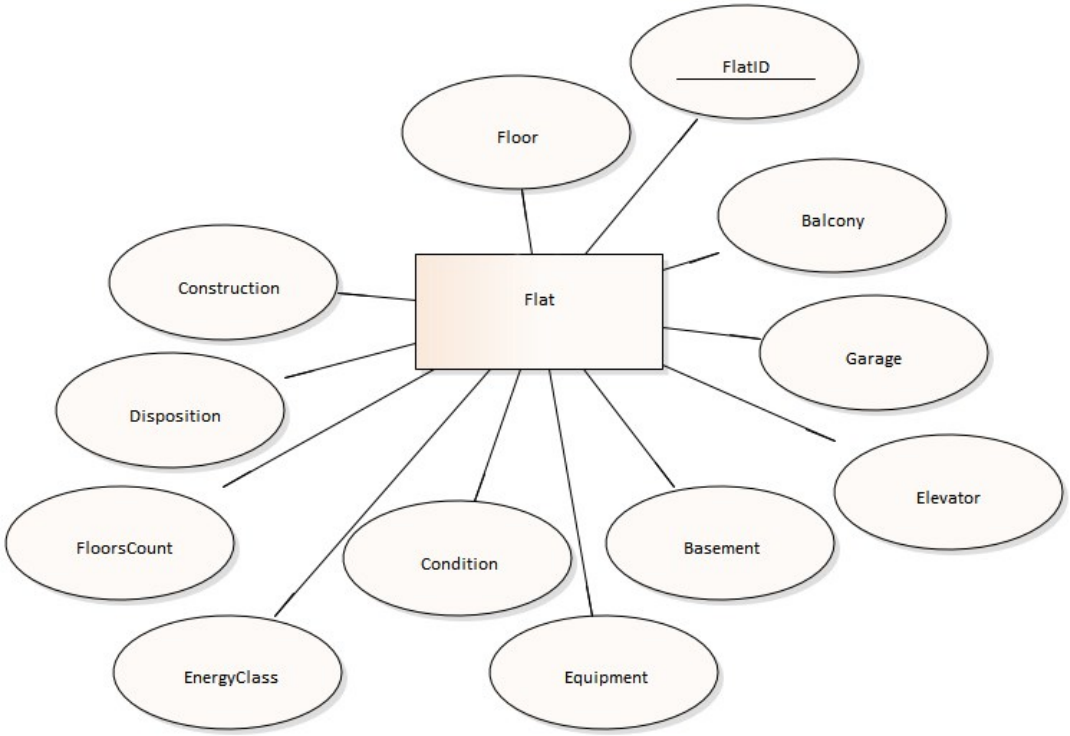
Tabulka obsahuje atributy specifické pro nemovitosti typu rodinný dům. Obsahuje cizí klíč PropertyId pro propojení s tabulkou Properties v případě, že je nemovitost typu rodinný dům.



Obr. 35. Atributy entity House

Tabulka Flats

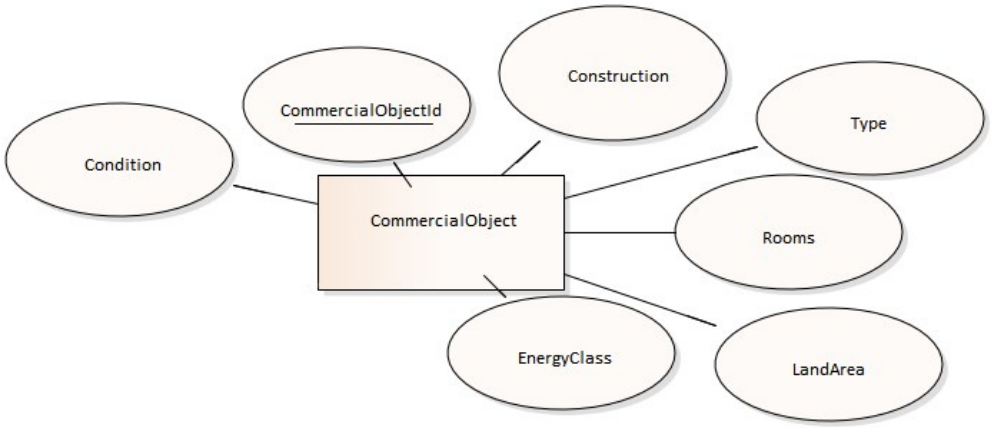
Tabulka obsahuje atributy specifické pro nemovitosti typu byt. Obsahuje cizí klíč PropertyId pro propojení s tabulkou Properties v případě, že je nemovitost typu byt.



Obr. 36. Atributy entity Flat

Tabulka CommercialObjects

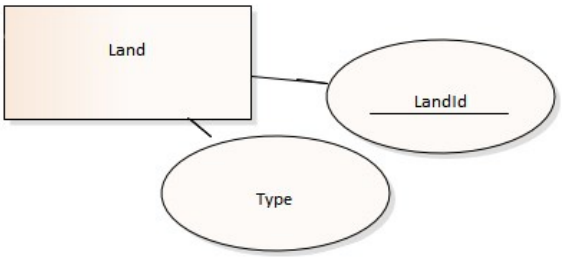
Tabulka obsahuje atributy specifické pro nemovitosti typu komerční objekt. Obsahuje cizí klíč PropertyId pro propojení s tabulkou Properties v případě, že je nemovitost typu komerční objekt.



Obr. 37. Atributy entity CommercialObject

Tabulka Lands

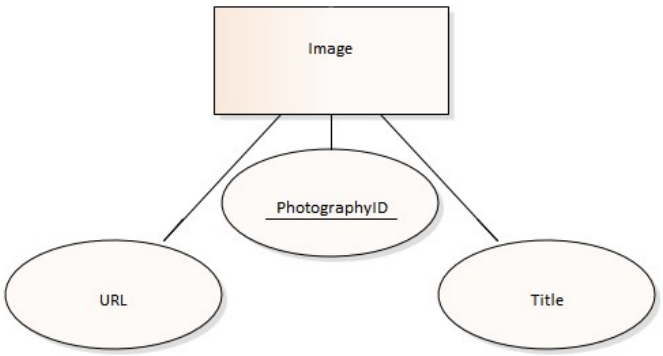
Tabulka obsahuje atributy specifické pro nemovitosti typu pozemek. Obsahuje pouze jediný atribut specifikující typ pozemku. Obsahuje cizí klíč PropertyId pro propojení s tabulkou Properties v případě, že je nemovitost typu pozemek.



Obr. 38. Atributy entity Land

Tabulka Images

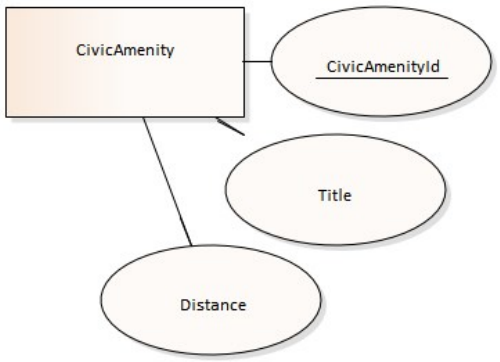
Tabulka reprezentuje detailní obrázky k nemovitostem. Samotné obrázky jsou ukládány do filesystému a do tabulky je ukládáno pouze URL obrázku. Obsahuje cizí klíč PropertyId pro propojení s tabulkou Properties pro uložení reference na konkrétní nemovitost.



Obr. 39. Atributy entity Image

Tabulka CivicAmenities

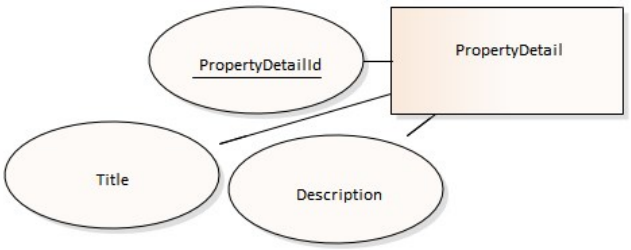
Tabulka reprezentuje objekty občanské vybavenosti. Každá položka občanské vybavenosti je skupinou atributů název a vzdálenost. Obsahuje cizí klíč PropertyId pro uložení reference na konkrétní nemovitost.



Obr 40. Atributy entity CivicAmenity

Tabulka PropertyDetails

Tabulka reprezentuje doplňující informace k nemovitosti, které může uživatel zadávat jako „key – value“ páry. Obsahuje cizí klíč PropertyId pro uložení reference na konkrétní nemovitost.



Obr. 41. Atributy entity PropertyDetail

Vztahy mezi tabulkami

- AspNetUsers – Properties – 1:N – Jeden uživatel může mít 0 až několik nemovitostí
- Properties – Houses – 1: 0..1 – Nemovitost může, ale nemusí být typu rodinný dům
- Properties – Flats – 1: 0..1 – Nemovitost může, ale nemusí být typu byt
- Properties – CommercialObject – 1: 0..1 – Nemovitost může, ale nemusí být typu komerční objekt
- Properties – Land – 1: 0..1 – Nemovitost může, ale nemusí být typu pozemek
- Properties – Images – 1:N – Nemovitost může mít 0 až několik obrázků
- Properties – CivicAmenities – 1:N – Nemovitost může mít 0 až několik objektů občanské vybavenosti

- Properties – PropertyDetails – 1:N – Nemovitost může mít 0 až několik detailních informací

Aplikace je naprogramována tak, aby každá vytvořená nemovitost byla jednoho z typů rodinný dům, byt, komerční objekt nebo pozemek, a i když uživatel později změní její typ při úpravě nemovitosti, jsou v databázi provedeny odpovídající změny.

6 IMPLEMENTACE ZABEZPEČENÍ APLIKACE

U aplikací pracují s uživatelské účty a umožňují uživatelům posílání dat přes formuláře na server je potřeba brát větší důraz na zabezpečení, aby nedošlo k úniku citlivých informací.

Prvním a zároveň také nejdůležitějším bezpečnostním opatřením je použití šifrované komunikace mezi serverem a uživatelem systému pomocí protokolu Hypertext Transfer Protocol Secure (HTTPS). Protokol HTTPS je nadstavbou standartního protokolu HTTP a zajišťuje přeposílání šifrovaných dat, jejich integritu a znemožňuje útočníkům odposlouchávání komunikace. Tato aplikace vyžaduje komunikaci s klientem přes HTTPS protokol, a i když uživatel implicitně zadá do URL adresy „http://“, je automaticky přesměrován na jiný port se zabezpečenou komunikací.

Použitím modulu Identity v této konkrétní aplikaci pro správu uživatelských účtů, který implementuje best practicies, bylo dosaženo standartního zabezpečení účtů. Hesla uživatelů jsou v databázi ukládány v hashované podobě, a tím pádem pro běžného uživatele nečitelné. K ještě větší bezpečnosti ukládá modul Identity k uživatelským účtům do databáze také „security stamp“, který reflektuje aktuální citlivé informace uživatelského účtu, a kdykoliv se tyto informace změní, je aktualizován taky „security stamp“. Na základě „security stampu“ jsou generovány autentizační cookies, které slouží uživateli k tomu, aby zůstal v aplikaci přihlášený, například při obnovení nebo načtení nové stránky. V případě útoku, kdy se útočníkovi podaří zjistit přihlašovací údaje k nějakému účtu a uživatel tuto skutečnost v časný okamžik rozpozná a změní si heslo, je vygenerován také nový „security stamp“ a s ním i nová autentizační cookies. Předchozí cookie se stane nevalidní a útočník je při dalším načtení stránky ze systému odhlášen a ztratí přístup k tomuto účtu.

Aplikace je proti SQL Injection útokům chráněna ne žádným speciálním bezpečnostním opatřením, ale způsobem, jakým probíhá komunikace s databází. Při použití syntaxe Language Integrated Query (LINQ) nejsou dotazy skládány pomocí manipulace a spojováním řetězců, ale jsou do databáze předávány jako SQL parametry, které zabraňují riziku vzniku útoku tohoto typu.

K ochraně uživatelů proti útokům typu Cross Site Request Forgery (CSFR) je aplikace chráněna implementací Anti Forgery tokenu. Díky této implementaci je ke každému formuláři vygenerovaný také speciální token, který je po odeslání formuláře ověřen a v případě nevalidního tokenu není požadavek vyřízen. Framework ASP.Net Core má velmi dobrou podporu proti útokům tohoto typu a po zapnutí této ochrany automaticky generuje tokeny pouze

k těm požadavkům, které jsou na tyto útoky citlivé, tzn. zejména požadavky typu POST, kdy dochází k modifikaci dat v aplikaci.

Aplikace vytvořené v ASP.NET Core defaultně posílají klientům (prohlížečům) HTTP hlavičku s údaji o webovém serveru, na kterém aplikace běží (v tomto případě webový server Kestrel). Pro útočníka, který by se pokusil nějakým způsobem poškodit tuto aplikaci, je každá informace vzácná a může zkrátit dobu, za kterou se útočníkovi podaří prolomit zabezpečení aplikace a napáchat škodu. Z toho důvodu má tato aplikace zakázáno odesílání této hlavičky.

7 DALŠÍ MOŽNÝ ROZVOJ APLIKACE

Aplikace v tomto stavu poskytuje uživatelům solidní základ pro inzerování a prohlížení nemovitostí. Pro další rozvoj aplikace je zde hodně prostoru jak z pohledu uživatelské funkcionality, tak z pohledu efektivnosti, výkonu a optimalizace serveru.

Z hlediska uživatelské funkcionality by se dala zlepšit komunikace mezi uživateli systému (inzerent - kupující) a dát například možnost kupujícímu si sjednat prohlídku zvolené nemovitosti. Inzerenti by měli k dispozici kalendář, který by mohl být synchronizovaný s často používaným Google kalendářem a kupující by mohli žádat o prohlídky ve volných časových slotech v kalendáři. Inzerent by mohl tyto žádosti vyřizovat přímo z dashboardu za pomocí rychlé nabídky a žádost buď potvrdit, zamítnout nebo navrhnout jiný čas prohlídky.

Pro usnadnění vyhledávání nabídek by mohl být systém rozšířen o vyhledávání nemovitostí podle mapy. Při kupování nemovitosti může být lokalita hlavním kritériem a tento způsob poskytuje kupujícím přehled o lokalitě nemovitosti, ale i dalších alternativách v dané lokalitě ještě před tím, než navštíví detailní stránku, kde je mapa se zakreslenou adresou nemovitosti.

Aplikace by také mohla kupujícím poskytnout hypoteční kalkulačku se srovnáním obecných podmínek různých bank, která by kupujícím dala velmi obecný přehled o tom, kolik, a jak dlouho by danou nemovitost spláceli při uvážení dalších parametrů.

Nákup nebo pronájem nemovitosti může být někdy neodkladnou záležitostí, ale pro zajímavost, informovanost a případně pro zvážení investiční příležitosti investorů by mohla aplikace poskytovat uživatelům cenovou mapu nemovitostí za uplynulé období v podobě grafů, seřazenou podle krajů, typů nemovitostí a dalších parametrů zvolených uživatelem. Trend cen posledních let na trhu s nemovitostmi má rostoucí tendenci a tato cenová mapa by mohla kupujícím pomoct v rozhodování, jestli je vhodná doba k nákupu, či pronájmu nemovitosti, a kam trh směřuje.

Pro zlepšení výkonu a efektivnosti aplikace na straně serveru by bylo možným vylepšením způsob, jakým jsou klientovi „servírovány“ statické soubory. Výrazné zmenšení velikosti obsahu posílaného do prohlížeče by bylo dodatečné zpracování obrázků k nemovitostem a ke každému nahranému obrázku by se do filesystému uložil ještě jeden obrázek s menším rozlišením, který by byl využit například při vyhledávání a následném výpisu vyhledaných nemovitostí, kde není zapotřebí obrázků v plném rozlišení, ale postačilo by například

rozlišení 300x300px. K dalšímu zredukování velikosti obsahu posílaného do prohlížeče by bylo možné zminifikovat scripty a CSS soubory.

ZÁVĚR

Realizací této bakalářské práce bylo dosaženo vytvoření portálu pro nabídku realit, který má intuitivní a jednoduché ovládání a poskytuje svým uživatelům, jakožto realitním kancelářím, majitelům nemovitostí a zákazníkům, prostředí, které splňuje obecné požadavky na systém tohoto typu.

Pro inzerenty byly vytvořeny uživatelské účty s kompletním dashboardem, který slouží jako rozcestník pro správu nemovitostí a svého uživatelského účtu, naproti tomu pro zákazníky byla zpracována vizuálně přitažlivá prezentace nabídek nemovitostí inzerentů s možností rozsáhlého filtrování nemovitostí, které zaručí relevantní výsledky vyhledávání a aplikace také komunikuje se službami třetích stran pro poskytnutí ještě většího uživatelského komfortu.

Při realizaci byl kladen důraz na každý aspekt vývoje softwaru a byla provedena analýza požadavků, vytvoření databáze, reaktivní a dynamická webová prezentace, která dává uživatelům zpětnou vazbu, byla vytvořena efektivní a spolehlivá implementace serveru a v poslední řadě byly zavedeny a implementovány bezpečnostní opatření proti nejčastějším typům útoků na webové aplikace.

SEZNAM POUŽITÉ LITERATURY

[1] Iamluminousman: Asynchronous programming. Blocking I/O and non-blocking I/O [online]. 2019 [cit. 2019-05-12]. Dostupné z: https://luminousmen.com/post/asynchronous-programming-blocking-and-non-blocking

[2] PHP Mind: Synchronous and Asynchronous [online]. 2017 [cit. 2019-05-11]. Dostupné z: http://www.phpmind.com/blog/2017/05/synchronous-and-asynchronous/

[3] Node.js: Overview of Blocking vs Non-Blocking [online]. [cit. 2019-05-12]. Dostupné z: https://nodejs.org/de/docs/guides/blocking-vs-non-blocking/

[4] PILGRIM, Mark. HTML5: Up and Running: Dive into the Future of Web Development [online]. Sebastopol: O'Reilly Media, 2010 [cit. 2019-05-07]. ISBN 0596806027. Dostupné z: http://shop.oreilly.com/product/9780596806033.do

[5] A. MAYER, Eric. CSS Pocket Reference: Visual Presentation for the Web [online]. 5th. Sebastopol: O'Reilly Media, 2018 [cit. 2019-05-07]. ISBN 1492033391. Dostupné z: http://shop.oreilly.com/product/0636920015055.do

[6] BEM: Block Element Modifier [online]. [cit. 2019-05-11]. Dostupné z: http://get-bem.com/

[7] LEWIS, Paul. Rendering Performance. Google [online]. Mountain View: Google, ©2019 [cit. 2019-05-16]. Dostupné z: https://developers.google.com/web/fundamentals/performance/rendering/

[8] HAVERBEKE, Marijn. Eloquent JavaScript: a modern introduction to program-ming [online]. Third edition. San Francisco: No Starch Press, [2019] [cit. 2019-05-16]. ISBN 15-932-7950-7. Dostupné z: https://eloquentjavascript.net/

[9] DRUMMOND, Robert. How to build a REST Web API on a Raspberry PI in JavaScript. Robert Drummond [online]. ©2019, 2013 [cit. 2019-05-14]. Dostupné z: http://www.robert-drummond.com/2013/05/08/how-to-build-a-restful-web-api-on-a-raspberry-pi-in-javascript-2/

[10] Bootstrap: Grid system [online]. San Francisco: Twitter, [cit. 2019-05-16]. Dostupné z: https://getbootstrap.com/docs/4.3/layout/grid/

[11] Bootstrap [online]. San Francisco: Twitter, 2011 [cit. 2019-05-13]. Dostupné z: https://getbootstrap.com

[12] CodeEasy: C# compilation process [online]. ©2018 [cit. 2019-05-10]. Dostupné z: https://codeasy.net/lesson/c_sharp_compilation_process

[13] ALBAHARI, Joseph a Ben ALBAHARI. C# 7.0 in a nutshell. 7th edition. Sebastopol: O´Reilly, 2018. ISBN 9781491987650.

[14] Docs Microsoft: Úvod do ASP.NET Core [online]. Washington: Microsoft, 2019 [cit. 2019-05-14]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/

[15] SASIDHARAN, Mithun. Should I Or Should I Not Use ORM ?. Medium [online]. San Francisco, ©2019, 2016 [cit. 2019-05-16]. Dostupné z: https://medium.com/@mithunsasidharan/should-i-or-should-i-not-use-orm-4c3742a639ce

[16] LOCK, Andrew. ASP.NET Core in action. Shelter Island: Manning, [2018]. ISBN 1617294616.

[17] Visual Studio 2019 [online]. Washington: Microsoft, 2019 [cit. 2019-05-11]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

[18] CHACON, Scott. Pro Git: [everything you need to know about the Git distibuted source control tool]. 2nd ed. New York, NY: Apress, 2014. ISBN 9781484200773.

[19] SHEMA, Mike. Seven Deadliest Web Application Attacks. Amsterdam: Syngress, 2010. ISBN 1597495433.

[20] Refreshless: noUiSlider [online]. [cit. 2019-05-12]. Dostupné z: https://refreshless.com/nouislider/

[21] HTML5 Web Storage. W3 Schools [online]. Refsnes Data, ©2019 [cit. 2019-05-14]. Dostupné z: https://www.w3schools.com/html/html5_webstorage.asp

[22] Overview of ASP.NET Core MVC. Docs Microsoft [online]. Washington: Microsoft, 2018 [cit. 2019-05-09]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/mvc/

[23] Data Transfer Objects [online]. ©2019 [cit. 2019-05-15]. Dostupné z: https://aspnetboilerplate.com/Pages/Documents/Data-Transfer-Objects

[24] What is Code-First?. Entity Framework Tutorial [online]. ©2019 [cit. 2019-05-15]. Dostupné z: https://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
API	Application Programming Interface
BEM	Block – Element - Modifier
ES6	ECMA Script verze 6
DOM	Document Object Model
AJAX	Asynchronous Javascript and XML
JSON	Javascript Object Notation
IL	Intermediate Language
CLR	Common Language Runtime
CLI	Command Line Interface
ORM	Object – Relational Mapping
SQL	Structured Query Language
LINQ	Language Integrated Query
CSFR	Cross-Site Request Forgery
XSS	Cross-Site Scripting
UI	User Interface
GPS	Global Positioning System
MVC	Model – View - Controller
DTO	Data Transfer Object
ERD	Entity Relationship Diagram
URL	Unified Resource Locator

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Přiložené CD obsahuje:

- Text
    - Navrh_aplikace_pro_realitni_portal_RaiskubKrystof.docx – text k bakalářské práci ve formátu .docx
    - Navrh_aplikace_pro_realitni_portal_RaiskubKrystof.pdf – text k bakalářské práci ve formátu .pdf
    - Navod_ke_spusteni.docx – návod ke spuštění aplikace ve formátu .docx
- Zdrojový kód
    - Navrh_aplikace_pro_realitni_portal_RaiskubKrystof.zip – zdrojový kód k aplikaci