Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj webové aplikace ASP.NET Core pro sdílení fotografií na mapě

Vypracoval: Pavel Vojta
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Pavel VOJTA**
Osobní číslo:	**E17456**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj webové aplikace ASP.NET Core pro sdílení fotografií na mapě**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce bude vytvořit webovou aplikaci ASP.NET Core, která umožní uživatelům umisťovat své fotografie na mapu a sdílet je s ostatními návštěvníky webu. V aplikaci budou mít uživatelé možnost přidávat ke svým fotografiím další informace, zařazovat svoje příspěvky do skupin či vytvářet skupiny vlastní.

Metodický postup:

1. Studium odborné literatury.
2. Zjištění specifikace pro výslednou aplikaci.
3. Návrh a popis vývoje a implementace výsledné aplikace.
4. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
5. Závěr.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Chiaretta, S. (2018). *Front-end Development with ASP.NET Core, Angular, and Bootstrap.* Indianapolis, IN (USA): John Wiley.
2. Dean, J. (2019). *Web programming with HTML5, CSS, and JavaScript.* Burlington, MA (USA): Jones & Bartlett Learning.
3. Dennis, A., Wixom, B. H., & Roth, R. M. (2012). *Systems analysis and design.* Hoboken, NJ: John Wiley.
4. Price, M. J. (2019). *C# 8.0 and .NET Core 3.0 -Modern Cross-Platform Development.* Birmingham, UK: Packt.
5. Seidl, M., Scholz, M., Huemer, C., & Kappel, G. (2015). *UML @ Classroom: An Introduction to Object-Oriented Modeling.* Cham, CH: Springer.
6. Microsoft Corporation. (2020). *Microsoft Learn* [online]. Dostupné z https://docs.microsoft.com/en-us/learn/

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **31. března 2021**
Termín odevzdání bakalářské práce: **15. dubna 2022**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 31. března 2021

Prohlášení

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

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

…… ……

Datum Podpis studenta

Poděkování

Rád bych touto cestou poděkoval Mgr. Radimu Remešovi za jeho pomoc, cenné rady a vstřícnost při vedení mojí bakalářské práce.

Obsah práce

1 Úvod

Když na jaře 2020 natvrdo ochromila svět pandemie tehdy ještě nového typu koronaviru, mnoho lidí si nedokázalo představit, že ani rok poté stále nebude situace pod kontrolou. Jedním z důsledků opatření, která mají zabránit šíření nákazy, je, že se lidé nemohou volně pohybovat, jak byli do té doby zvyklí, a cesty za výlety na místa, která nejsou v jejich okolí, musejí odložit na lepší časy.

Naštěstí se protiepidemická opatření netýkají internetu. Zůstává nám tak prostor, kde je možné sdílet fotografie zajímavých míst, ať už z výletů daleko od domova nebo z míst kudy se běžně procházíme. A dělit se o takové fotografie je ústřední myšlenka, na které jsem se rozhodl, že postavím tuto svoji bakalářskou práci.

Ta se skládá ze dvou částí. V teoretické části práce se věnuji převážně popisu webových aplikací a technologií související s jejich vývojem. Naproti tomu se v praktické části soustředím na popis vyvíjené aplikace GEOscéna.

1.1 Cíl práce

Cílem této práce je vytvořit webovou aplikaci, která umožní uživatelům sdílet svoje fotografie. Ty pak budou vyobrazeny na mapě podle místa, kde byly pořízeny. Souřadnice tohoto místa se aplikace při nahrávání fotografie pokusí sama získat z metadat fotografie, popřípadě takovéto místo uvede uživatel. Dále pak aplikace nabídne uživatelům vytvořit si svůj vlastní účet. Tím pak získají možnost upravovat svoje příspěvky i po nahrání, zakládat různé tematické skupiny, kam bude možné zařazovat nahrané fotografie, přidávat se do nich, anebo si jednotlivé příspěvky oblíbit. Vytvořená aplikace pak bude všem zájemcům zpřístupněna na internetu.

2 Přehled řešené problematiky

2.1 Webová aplikace

S rozvojem webových technologií začaly webové aplikace hrát čím dál významnější roli a dnes již asi nikoho nepřekvapí, že aplikace mohou běžet ve webovém prohlížeči. Příkladem mohou být aplikace Photopea a Microsoft 365, které jsou také ukázkou použití softwaru jako služby.

2.1.1 Web

Historie *webu* (WWW), tedy kolekcí webových stránek, se začala psát v roce 1989 ve švýcarských fyzikálních laboratořích CERN, kde Tim Berners-Lee přišel s myšlenkou přidat do dokumentů hypertextové odkazy, které by umožnily přejít z jednoho dokumentu na další. Sám tak navrhl jazyk *HTML*, koncept WWW, včetně protokolu HTTP používaného pro přenos webových stránek, vytvořil prototyp prohlížeče webových stránek a založil konsorcium W3C standardizující HTML. (Dean, 2019)

Značkovací jazyk HTML (Hypertext Markup Language) se používá pro tvorbu webových stránek. Jeho účelem je popsat všechny prvky použité na stránce, jako jsou například nadpisy, odstavce nebo seznamy. Pro určení vzhledu obsahu stránky se používají *kaskádové styly* (CSS). Ty lze také použít i pro přidání animací. (Niederst Robbins, 2018)

Postupem času se web proměnil z převážně stránek nabízející jen minimální interakci s návštěvníky, přes Web 2.0 umožňující uživatelům, aby se sami stali tvůrci obsahu, což otevřelo cestu blogům a sociálním sítím, a dovolující použít obsah webu na jiných webech prostřednictvím aplikačního rozhraní API, až do podoby sémantického Webu 3.0, jenž využívá umělou inteligenci, díky které dokáže porozumět samotnému obsahu. (Ranjan, Sinha, & Battewad, 2020; Lacko, 2011b)

2.1.2 Definice webové aplikace

Webové aplikace jsou aplikace, které byly vyvinuty pomocí webových technologií a jsou dostupné přes webový prohlížeč (Vora, 2009). Podle Birnir (2020) se liší od webových stránek tím, že získávají od svých uživatelů obsah (např. Twitter), nebo data sbírají z jiných zdrojů (např. Google Analytics) a následně jej uživatelům prezentují.

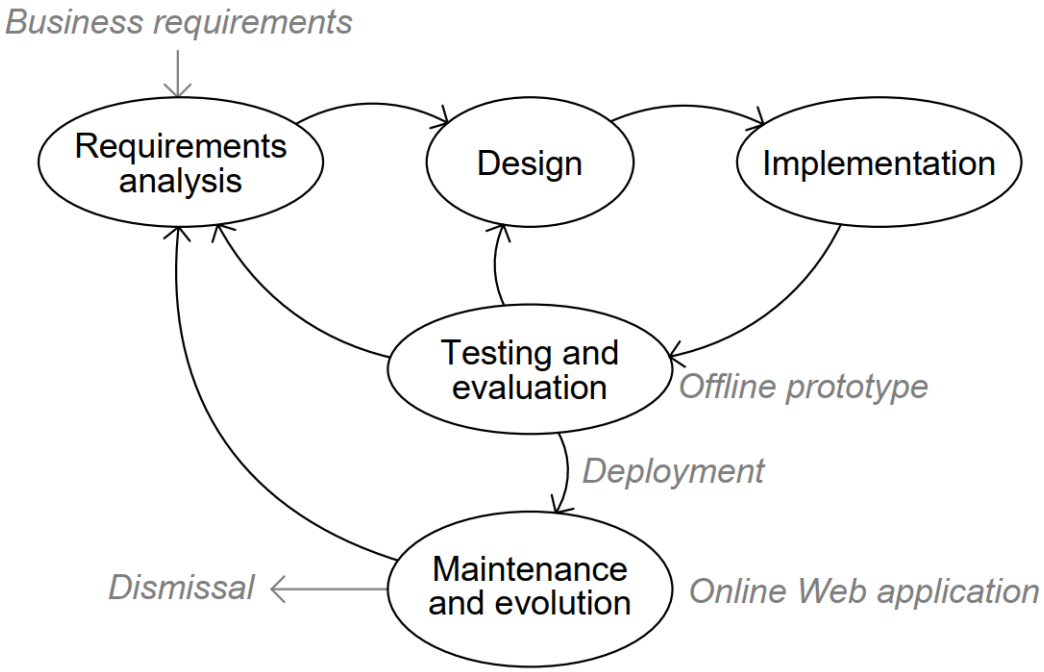
2.1.3 Proces vývoje webové aplikace

Proces vývoje softwaru se zpravidla skládá z těchto následujících fází (Casteleyn, Daniel, Dolog, & Matera, 2009):

- *Analýza požadavků* – cílí na porozumění problému, pochopení požadavků na vyvíjený produkt a stanovení jeho funkcí (funkční požadavky) a kvality (nefunkční požadavky),
- *Návrh* – soustředí se na plánování řešení problému s ohledem na stanovené požadavky, stejně tak i na omezení daná cílovým prostředí,
- *Implementace* – vytváří se podle plánu kód aplikace,
- *Testování a zhodnocení* – zaměřuje se na hledání chyb v kódu a pátrá po rozporech mezi navrženými požadavky a jejich implementací, typicky probíhá současně s předchozí fází,
- *Nasazení* – označuje poskytnutí vyvíjené aplikace svým uživatelům a zaškolování uživatelů, obzvláště když aplikace představuje výraznou změnu,
- *Údržba* – cílí na monitorování běžícího systému, aby se předešlo jeho selhání a aby byla zachována jeho dostupnost, opravování chyb a nasazování bezpečnostních záplat,
- *Rozvíjení* – znamená postupné zlepšování již vyvinutého řešení a ve formě nových požadavků představuje nový vstup do procesu vývoje.

Dále Casteleyn a kol. (2009) upřesňují popis vývoje přímo webových aplikací. Ten se velmi podobá výše uvedenému procesu, hlavní rozdíl ale podle nich spočívá v tom, že se fáze nasazení chápe pouze jako přechodný proces. Zároveň, jak ukazuje Obrázek 1, se v něm vyskytují cykly. První z nich, kterému se přezdívá *sestavovací a testovací cyklus*, zahrnuje návrh, implementaci a fázi testování a zhodnocení. Druhý cyklus označovaný jako *vývojový* spojuje fázi údržby a rozvoje s fází analýzy požadavků.

Obrázek 1: Životní cyklus webových aplikací



Zdroj: Casteleyn a kol. (2009)

2.1.4 Architektura aplikací

Správně zvolená architektura aplikace usnadňuje její vývoj. Pokud se kód aplikace rozdělí do vrstev, které spolu logicky souvisejí (logika aplikace, data, uživatelské rozhraní), může se využít na více místech zároveň, stává se přehlednějším a tím pádem je i jednodušším na údržbu.

Například ke třívrstvé architektuře, jež se skládá z *prezentační vrstvy* starající se o komunikaci aplikace s uživatelem, *datové vrstvy*, která je zodpovědná za správu dat, a *aplikační vrstvy* spojovanou s logikou aplikace, se mimo jiné řadí následující:

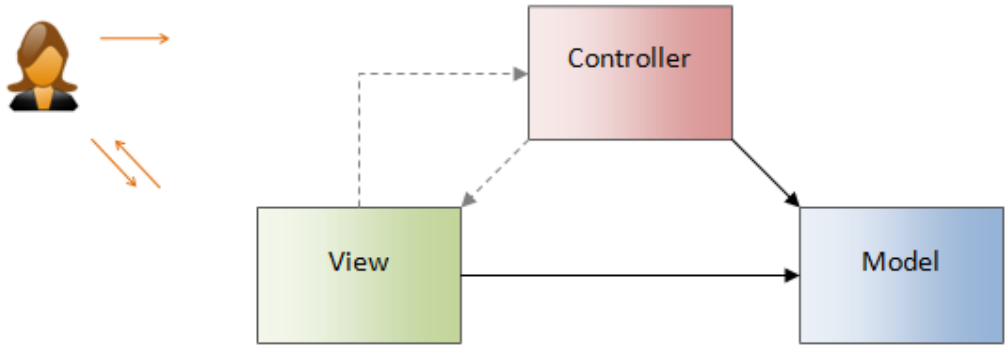
2.1.4.1 Model-View-Controller

Architekturu *Model-View-Controller* (MVC) tvoří části *Model*, který reprezentuje data a business logiku aplikace, *View* zobrazující uživatelské rozhraní a *Controller*, jenž se v aplikaci stará o tok událostí a aplikační logiku obecně. Aplikace založená na architektuře MVC se dělí na tyto tři části, aby bylo možné je upravovat samostatně a aby byl dopad případných změn na ostatní části co nejmenší. (Bernard, 2009a)

Komunikaci v architektuře MVC ukazuje Obrázek 2. Vazby na této úrovni jsou mezi Controllerem a Modelem, aby mohl pracovat s jeho daty, a podobně tak i mezi View a Modelem, aby jeho data měl možnost zobrazit. Naproti tomu ale Model nesmí držet

přímý odkaz na zbylé dvě komponenty. View se dále stará o výstup dat aplikace a u „widgetových“ systémů (jako jsou např. WPF, ASP.NET Web Forms, PHP) také pracuje se vstupem od uživatele. (Bernard, 2009a)

Obrázek 2: Architektura MVC

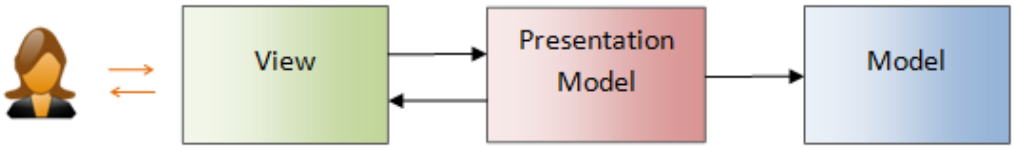


Zdroj: (Bernard, 2009a)

2.1.4.2 Model-View-ViewModel

Návrhový vzor *Model-View-ViewModel* (MVVM) se skládá ze tří vrstev. *View* představuje uživatelské rozhraní, data používaná v aplikaci popisuje *Model*, který nesmí nic vědět o stavu ovládacích prvků, a *ViewModel* propojuje Model s View, kterému poskytuje data, a zároveň drží stav aplikace. (Dajbych, 2009) Vzor MVVM, který zobrazuje Obrázek 3, se také označuje jako *Presentation Model* a řídí se podobnými principy jako MVC (Bernard, 2009b).

Obrázek 3: Návrhový vzor MVVM



Zdroj: (Bernard, 2009b)

2.1.5 Cloud computing

Pojmem *cloud computing* (dále jen jako CC) se označuje model poskytující uživateli pohodlný přístup ke konfigurovatelným výpočetním zdrojům (např. sítě, servery, úložiště, aplikace a služby) kdykoli na vyžádání, přičemž uživatel vyvíjí minimální úsilí na velmi rychlé zprovoznění a ukončení služby, případně na komunikaci s poskytovatelem takové služby. (Mell & Grance, 2011)

Zároveň Mell a Grance (2011) upřesňují charakteristiky této služby, která je *samoobslužná*, *zcela síťová*, *velmi rychle škálovatelná*, což zákazníkovi umožňuje nastavovat rozsah objednané služby, jak v daném okamžiku potřebuje, dále *sdílí zdroje s ostatními zákazníky* a *měří využití služeb* pro jejich optimalizaci a vyúčtování.

Modely způsobu nasazení CC se člení na *veřejný cloud*, kdy jedna organizace poskytuje takovouto službu jiným organizacím či jednotlivcům, *privátní cloud*, u kterého si organizace sama vyvinula službu CC a jeho služby se používají pouze v rámci této společnosti, *komunitní cloud*, kdy skupina organizací nebo komunita se stejnými zájmy vybudovala prostředí CC a na vyžádání využívá jeho služeb, a na *hybridní cloud*, v jenž je několik cloudů propojeno, čímž se zajišťuje přenositelnost (např. aplikací) mezi nimi a tím pádem i efektivnějších využití zdrojů. (Gála, Pour, & Šedivá, 2015)

Za modely služeb CC se považují model *software jako služba (SaaS)* pronajímající zákazníkům aplikace dostupné například z prohlížeče, přičemž se zákazník dál nestará o správu cloudové infrastruktury, na které služba běží, model *platforma jako služba (PaaS)* umožňující zákazníkovi provozovat vlastní aplikace nebo aplikace třetích stran na cloudové infrastruktuře poskytovatele a model *infrastruktura jako služba (IaaS)*, kdy je zákazníkovi poskytnuta část výpočetních zdrojů, aby si mohl na ní sám nasadit libovolný software, včetně operačních systémů a aplikací. (Mell & Grance, 2011)

2.2 Vývojové nástroje

2.2.1 Technologie

ASP.NET Core je technologie od společnosti Microsoft určená pro vytváření webových stránek a webových služeb zveřejněná v roce 2016. Je založena na platformě .NET Core, která vznikla jako snaha učinit .NET Framework plně multiplatformní. S .NET Framework, vývojářskou platformou, jež je složena z *Common Language Runtime* starající se o provádění kódu a z *Base Class Library* poskytující bohatou knihovnu tříd pro vytváření aplikací, měl Microsoft při jejím návrhu podobnou ambici, ale postupem času se soustředil převážně na to, aby fungovala především v operačním systému Windows. (Price, 2019)

A ačkoli je původní platforma .NET Framework udržována i nadále, nové funkce a vylepšení přibývají jen pro .NET Core, u které jsou nové verze zpravidla vydávány každý listopad a u které se počínaje verzí 5 začalo vypouštět označení Core. (“What is .NET?”, n.d.)

Jako výhody ASP.NET Core uvádí Smith (2020) následující vlastnosti:

- *Optimalizace pro cloud a škálovatelnost*. Díky malým nárokům na paměť a vysokému výkonu je možné spouštět na stejném hardwaru více aplikací, a naopak platit méně za využívané zdroje.
- *Multiplatformost*. Platforma ASP.NET Core podporuje Windows, Linux i macOS. Aplikace ASP.NET Core mohou běžet i v kontejnerech Docker.
- *Modulárnost*. Aplikace .NET Core a ASP.NET Core jsou složeny z mnoha knihoven z balíčků NuGet, což vede k tomu, že se vytvářejí pouze nezbytné závislosti a nasazuje se jen taková funkcionalita, která je skutečně potřeba, a tak mohou být aplikace výkonnější a bezpečnější. ASP.NET Core také podporuje *vkládání závislostí*, tedy techniku, která umožňuje volně vázat různé části aplikace, což je žádoucí stav, protože je pak jednodušší tyto samostatné části otestovat či je nahradit.
- dále zmiňuje také i *snadnou testovatelností automatizovanými testy* a *snadný vývoj a nasazení aplikací* pomocí vývojářských nástrojů.

2.2.2 Programovací jazyk

Speciálně pro potřeby platformy .NET Framework vznikl programovací jazyk C# (Hanák, 2008). Jedná se o vyšší objektový programovací jazyk, který byl vydán v roce 2002 společností Microsoft (Virius, 2021). Přičemž objekty se rozumí datové struktury složené z metod, které vykonávají daný úkol, a z vlastností, tedy datových položek popisující tento objekt (Kroenke & Auer, 2015).

Verze 1.0 jazyka C# nedosahovala ani zdaleka tolik možností jako nabízí nejnovější verze, umožnovala především tvořit *třídy*, *rozhraní*, *struktury*, *události* apod. Ve verzi 2.0 přišla hlavně možnost specifikovat datové typy až při vytváření instance, takzvaná *genericita* (Čápka, 2020), a *iteratory* umožňující postupně procházet prvky v kolekcích. Velké změny nastaly s verzí 3.0, kdy se v C# objevilo *LINQ*, které dovoluje psát dotazovací výrazy připomínající jazyk SQL nad kolekcemi, příkladem může být zjišťování průměru čísel pomocí jednoduchého `seznamCisel.Average()`, a mimo jiné se dostalo i na lambda výrazy. Ve verzi 4.0 se kromě jiného objevilo nové klíčové slovo `dynamic`, které do jinak silně typového jazyka C#, kde je u každé proměnné a konstanty nutné předdefinovat její datový typ, vneslo možnost toto pravidlo obejít a datový typ v průběhu programu měnit (Chand, 2014), ukázku takového použití zobrazuje Ukázka kódu 1. Verze 5.0 se soustředila na zpřístupnění asynchronního programování, které

zajišťuje že po zavolání asynchronní metody zpracovávající operaci náročnou na čas se neblokuje hlavní vlákno, jinými slovy se dá říct, že takováto metoda ještě, než svoji operaci dokončí, zdánlivě skončí do doby, než je tato operace dokončena (Veith, 2011). Z tohoto důvodu se zavedla nová klíčová slova `async` a `await`. Následná verze 6.0 byla zaměřena zejména na zlepšení produktivity a ve verzi 7.0 se umožnilo například deklarovat proměnné v argumentu při volání metody při využívání klíčového slova `out`. (Dietrich et al., 2020)

Ukázka kódu 1: Ukázka použití dynamic

```
dynamic x = 6;  // x = 6 (int)

x = x * 2;      // x = 12 (int)

x = x + 3.14;   // x = 15.14 (double)

x = x + "a string"; // x = 15,14 (string)

x = true;       // x = true (bool)
```

Zdroj: Autor

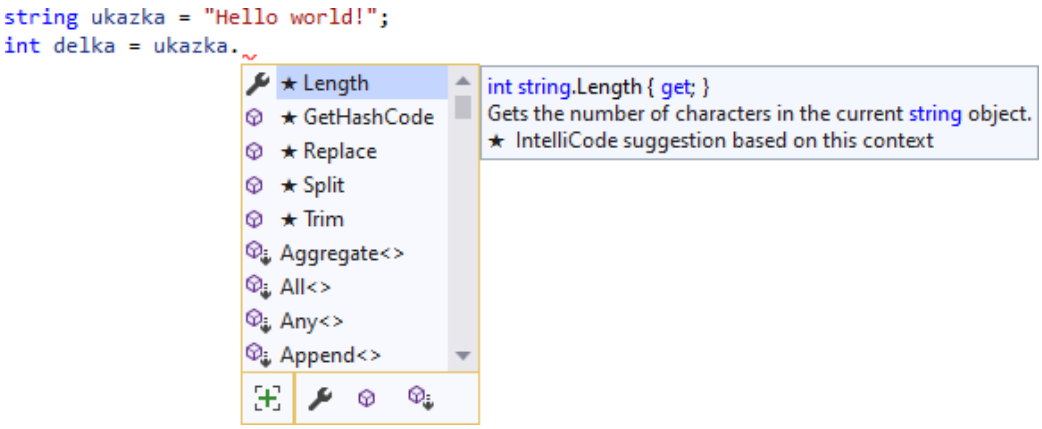
C# je možné použít jak pro vývoj klasických okenních aplikací, tak i aplikací určených pro Microsoft Store, webových aplikací, webových API, tedy služeb pro klienty jako jsou například webové prohlížeče nebo mobilní zařízení, anebo WCF služeb, které umožňují virtuální výměnu dat přes místní síť nebo přes internet (Perkins, Hammer, & Reid, 2018)

2.2.3 Vývojové prostředí

Vývoj aplikací velmi usnadňují integrovaná vývojová prostředí (IDE) a jedním z nich je *Visual Studio*, jehož poslední verze v současné době je 2019. Visual Studio nabízí sady například pro vývoj webových aplikací, desktopových aplikací, aplikací pro chytré telefony či doplňků pro Office anebo nasazení aplikací a služeb v Microsoft Azure. Zároveň poskytuje velké množství nástrojů, jako je třeba *IntelliSence*, které se snaží pomoci při programování tím, že našeptává další možné části kódu, navrhuje opravy chyb a doporučuje vylepšení v něm, *CodeLens* poskytující přehled o změnách v kódu a jejich autorech a částech programu, kde se tento kód používá, anebo nástroje pro zkoumání běhu programu, testování a nasazení výsledné aplikace. (“Visual Studio 2019”, 2020)

Ukázku IntelliSence, kde tento nástroj navrhuje podle už napsaného kódu doplnit takový, který by mohl následovat, nabízí Obrázek 4.

Obrázek 4: Ukázka použití nástoje IntelliSence



Zdroj: Autor

2.2.4 Razor Pages

Jak uvádí Smith (2020), ve Visual Studiu se se pro vývoj nových webových aplikací preferují *Razor Pages* (dále jako RP) založené na ASP.NET Core. RP používají syntaxi jazyka Razor, což umožňuje kombinovat jazyky HTML, CSS a C#, ergo skládá se ze dvou druhů obsahu – pro klienta a pro server. Klientský obsah je složen z jazyků HTML, CSS a případně skriptů JavaScript. Serverový kód, jenž se provádí jako první, slouží jak pro uskutečňování náročnějších činností, jako je například přístup k databázi, tak především ke generování dynamického obsahu. Pro webový prohlížeč jsou oba druhy obsahů úplně stejné. (“Understand when and why to use Razor Pages”, n.d.) Z pohledu architektury jsou RP postaveny na návrhovém vzoru MVVM (Smith, 2019).

RP používají konstrukci dvou souborů. Prvním z nich je soubor stránky s příponou `.cshtml`, kde se s použitím jazyka Razor definuje obsah stránky, a druhý soubor končící na `.cshtml.cs` obsahuje kód jazyka C#, pomocí kterého se určuje, co se bude na stránce dít a jak budou používaná data na stránce vypadat.

Pro obsluhování požadavků HTTP a vykreslování dat na stránce využívají RP v souboru s příponou `.cshtml.cs` třídu `PageModel`, která řešení těchto problémů z RP odděluje a aplikace jsou tak více modulární a jednodušší na údržbu (“Handle the product creation form submission”, n.d.).

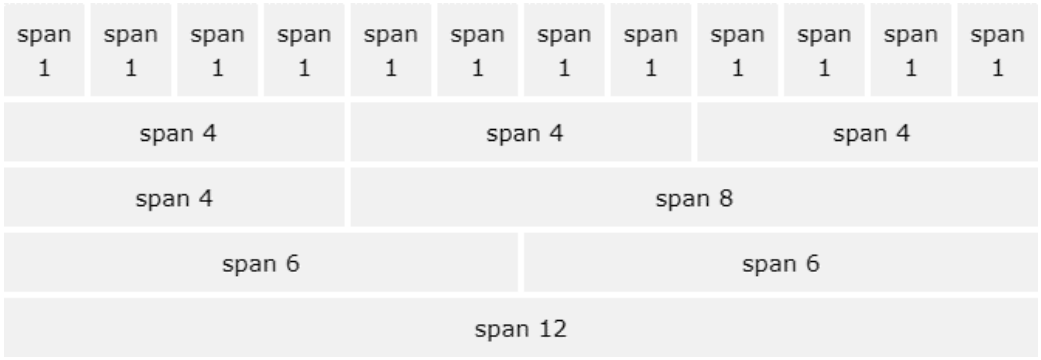
2.2.5 Bootstrap

Dobrou podporu ve Visual Studiu má i *Bootstrap*, což je framework, který nabízí již předpřipravenou množinu tříd CSS pro vylepšení vzhledu stránky, komponenty, čímž se v tomto případě rozumí komplexnější prvky grafického rozhraní, jako je například

navigační lišta či skupina formulářových polí, a pluginy JavaScriptu, bez nichž by některé tyto komponenty nefungovaly. (Chiaretta, 2018)

Právě třídy kaskádových stylů považuje u frameworku Bootstrap Chiaretta (2018) za stěžejní. Pomocí nich je také možné učinit stránku responzivní, tedy aby přizpůsobovala vzhled prvků na ní podle velikosti obrazovky. Zároveň se styly v tomto frameworku dělí do skupin pro tlačítka, formuláře, tabulky, typografii a souřadnicový systém. Ten, jak znázorňuje Obrázek 5, se skládá z řádků a z dvanácti sloupců, a tak je možné vytvářet na stránce stejně široké prvky, nebo snadno nastavit jejich šířku na polovinu, třetinu, čtvrtinu apod. nadřazeného prvku stránky či dokonce měnit velikost prvku na šířku podle velikosti obrazovky.

Obrázek 5: Souřadnicový systém frameworku Bootstrap



Zdroj: (“Bootstrap 4 Grid System”, n.d.)

2.2.6 NuGet

NuGet je oficiální mechanismus pro sdílení balíčků, které obsahují užitečný kód, mezi vývojáři na platformách .NET a .NET Core. NuGet určuje způsob tvorby těchto balíčků, jejich poskytování a používání, zároveň pro všechny tyto činnosti poskytuje nástroje. Takovýto balíček je jednoduchý soubor ZIP a obsahuje soubor `.nupkg`, kde se nachází zkompilovaný kód (DLL), další soubory související s tímto kódem a manifest, kde je balíček popsán, například jeho verze. (Douglas et al., 2019)

2.3 Databáze

Za databázi považuje Procházka (2009) uspořádanou množinu dat uloženou na paměťovém médiu, zároveň však dodává, že je možné označit pojmem databáze i data spolu se softwarovými prostředky umožňující přístup k datům a manipulaci s nimi neboli *systém řízení báze dat* (SŘBD), který pak odděluje datové struktury od samotných programů.

Firma IBM vyvinula první verzi dotazovacího jazyka SQL určeného pro práci s relačními databázemi a prvním SŘBD, který využíval jazyk SQL byl Oracle. Koncepci relační databáze popsal E. F. Codd už v roce 1970, ale staví se na ní dodnes. V relační databázi se data ukládají ve formě tabulek, přičemž data jsou uspořádána do řádků a sloupců takovéto tabulky. K vyjádření vzájemných vztahů (relací) mezi datovými tabulkami slouží hodnoty, jež jsou v různých tabulkách společné. (Laurenčík, 2018) Právě proto, že relace tvoří základní kámen tohoto druhu databází, se nazývají relační databáze.

Pro přístup dat v relačních databází v aplikacích založených na ASP.NET Core se doporučuje používat *Entity Framework Core* (dále jen jako EF Core), jenž se možné také využít i pro ukládání dat. Pro práci s EF Core je potřeba použít třídu `DbContext`, která obsahuje vlastnosti kolekcí entit používaných v aplikaci, prostřednictvím kterých EF Core získává příslušná data. Tato data lze filtrovat pomocí LINQ. (Smith, 2020)

Existuje také hnutí *NoSQL* (Not only SQL), přičemž většina databázových systémů, které s tímto hnutím souvisejí se řadí mezi nerelační databázové systémy a jsou často označovány jako strukturované úložiště. Obvykle je databázový systém typu NoSQL založen na distribuované replikované databázi, tj. databázi rozdělené na více částí uložené a zkopírované do více počítačů, a používá se v případech, kdy není možné bez něj zpracovávat velké datové množiny. Popřípadě NoSQL databáze používá k ukládání dat struktury dokumentů XML. (Kroenke & Auer, 2015)

2.4 Nasazení

Společnost Microsoft nabízí svoji vlastní cloudovou službu *Microsoft Azure*, kterou lze spravovat prostřednictvím *Azure Portal*.

Součástí Azure je služba *App Service*. Umožňuje rychle nasadit a škálovat webové aplikace a API vytvořené pomocí .NET, .NET Core, Node.js, Python, Java a PHP v kontejnerech nebo přímo ve Windows či v Linuxu, zároveň se pyšní mimo jiné vysokou dostupností spuštěných aplikací, bezpečností a výkonem, o čemž svědčí to, že App Service denně zpracovává přes 40 miliard požadavků. (“App Service”, n.d.)

Mezi dalšími službami Azure lze nalézt i *Azure SQL Database*. Tato služba poskytuje vysoce dostupnou a výkonnou plně spravovanou relační databázi SQL. (“Azure SQL Database”, n.d.)

2.5 Fotografie

Obecně lze říct, že účelem fotografie je zachytit v obrazové formě danou scénu v čase jejího pořízení.

Digitální fotografie vznikne tak, že senzor, který se nachází uvnitř fotoaparátu, skládající se z mřížky milionů malých čtverečků, pixelů, zachytí na ně dopadající světlo, která je pak převedena na elektrickou energii, což umožní softwaru fotoaparátu získat informace o barvě a jasu z každého jednotlivého pixelu. Výsledkem je pak dlouhý řetězec čísel, které popisují vlastnosti všech pixelů fotografie. (Woodford, 2020; McDowell, 2009)

2.5.1 Vlastnosti fotografií

Soubory obrázků mohou nést kromě počtu pixelů na šířku a na výšku i další informace. Nejoblíbenější způsob, jak ukládat vlastnosti fotografií, je použití formátu EXIF (Exchangeable image file format), jenž podporuje mnoho formátů obrázků, jako je JPEG, TIFF, RIFF ("Understand metadata concepts", 2020). Do EXIF vkládá fotoaparát údaje automaticky (Pecinovský, 2017).

V rámci formátu EXIF, zmiňuje Maître (2015) následující oblasti, které mohou být do něj ukládány:

- *informace o hardwaru*, jako je výrobce a model fotoaparátu nebo blesk,
- *podmínky zachycení*, například orientace fotoaparátu či hodnota ISO,
- *interní nastavení fotoaparátu*, kam zařadil mimo jiné použitý barevný rozsah nebo vyvážení bílé,
- *zvláštní data pro obsluhující program* určené pro uložení data zpracování a použité aplikaci.

Do oblasti podmínek zachycení zařadil Maître (2015) i souřadnice GPS. Ty se skládají ze dvou hodnot – zeměpisné šířky a zeměpisné délky.

Zeměpisná šířka značí úhel mezi s rovníkem, tedy rovnoběžkou vzdálenou stejně daleko od obou zemských pólů rozdělující zemské těleso na severní a jižní polokouli, a příslušnou rovnoběžkou, na které leží určovaný bod (Brzóska, 2020). Možné hodnoty zeměpisné šířky jsou mezi 90° jižní šířky a 90° severní šířky, přičemž jeden stupeň koresponduje se zhruba 111 km (Lacko, 2011a).

Zeměpisná délka se definuje náležícím poledníkem, tedy pomyslnou spojnicí mezi severním a jižním zemským pólem, a její hodnotou je úhel, který tento poledník svírá s nultým poledníkem procházející londýnskou čtvrtí Greenwich, jenž zároveň s protilehlým poledníkem nultého poledníku dělí Zemi na východní a západní polokouli (Brzóska, 2020). Hodnoty zeměpisné délky mohou ve stupních nabývat od 180° západní délky do 180° východní délky, nebo mohou být uvedeny v hodinách v rozmezí od -12 do +12 hodin, kdy jedna hodina odpovídá zhruba 15° zeměpisné délky (Lacko, 2011a).

Právě tyto souřadnice se dají získat pomocí GPS satelitů, kterých bylo v lednu 2021 v provozu 27 ("GPS.gov: Frequently Asked Questions", 2020). Jak popsal Kuruc (2011), zjištění polohy je možné díky tomu, že každý GPS satelit vysílá informace o své oběžné dráze (efemeridy), přehled o pozicích dalších satelitů (almanach) a přesný čas odeslání těchto navigačních zpráv podle atomových hodin satelitu. Přijímač z těchto údajů a ze známé rychlosti šíření rádiových signálů dokáže vypočítat vzdálenost od satelitu a zjistí, že se nachází na pomyslné kouli o tomto poloměru, ještě předtím je ale potřeba, aby byly hodiny přijímače přesně se synchronizované podle signálu ze satelitů. Po zachycení signálu z druhého satelitu a opětovného výpočtu vzdálenosti i k němu, přijímač získá další pomyslnou kouli, čímž pak může vzniknout kružnice, na jejíž obvodu se nachází. Signál ze třetího satelitu umožní přijímači vyznačit na této pomyslné kružnici dva body, z nichž jeden se nachází buďto vysoko ve vesmíru, nebo hluboko pod zemským povrchem, a naopak bod druhý ukazuje místo na povrchu země. Avšak pro získání pozice, včetně nadmořské výšky, jsou potřeba alespoň satelity čtyři.

3 Metodika

Pro vývoj webové aplikace jsem rozhodl použít na základě svých předchozích zkušeností s tvorbou aplikací moderní programovací jazyk C#, a také fakt, že je tento jazyk spjatý frameworkem ASP.NET Core, můj výběr potvrdilo. Důsledkem tohoto rozhodnutí bylo, že jsem musel najít vhodný způsob pro samotnou tvorbu webové aplikace, a tak moje padla volba na technologii Razor Pages, která umožňuje kombinovat jazyk C# s jazyky používané pro tvorbu webových stránek, jako je HTML, CSS a JavaScript.

Taktéž jsem si pro vývoj aplikace vybral vývojové prostředí Visual Studio, které dává k dispozici spoustu nástrojů, které mi ulehčují samotný vývoj, a také nabízí i předpřipravené šablony aplikací. Použití takovéto šablony velice usnadňuje práci, například s řešením záležitostí týkající se uživatelských profilů. Současně jsem dospěl k rozhodnutí využít knihovny ostatních vývojářů, které jsou k dispozici prostřednictvím NuGet.

Protože pro funkčnost mapy jsou potřeba mapové podklady, zvolil jsem, že využiji ty podklady, které poskytuje *OpenStreetMap* (OSM), protože jsou všem volně k dispozici zdarma a komunita je neustále aktualizuje. Zároveň pro OSM je možné použít i pluginy, které tak rozšiřují jejich možnosti.

Stejně tak jsem se rozhodl, že výslednou aplikaci nasadím na cloudové službě Azure, protože díky nabídce Microsoftu pro studenty *Azure for Students* jsem získal možnost zdarma využít služeb, které Azure nabízí. A jelikož se na Jihočeské univerzitě běžně používá účet Microsoft, prokázat, že mám na tuto nabídku nárok, a začít ji využívat bylo velmi snadné.

Neboť jsem předtím neměl s vývojem webové aplikace žádné zkušenosti, velmi mi pomohly výukové materiály, které Microsoft dává k dispozici prostřednictvím své platformy *Microsoft Learn* a svého oficiálního kanálu na YouTube *dotNET*[1], a dokumentace na webu *Microsoft Docs*.

[1] Přehled videí je dostupný na adrese https://www.youtube.com/c/dotNET/playlists

4 Řešení a výsledky

4.1 Požadavky na aplikaci

Na začátku tvorby aplikace jsem si potřeboval ujasnit, co vše bude ve výsledku umět. Ze základního cíle této práce sdílet fotografie na mapě je již patrné, že aplikace musí uživatelům umožnit nahrávat svoje fotky a následně je zobrazovat na mapě.

Aplikace se sama při nahrávání fotografie měla pokusit získat souřadnice místa, kde byla vyfocena, stejně tak je ale umožní zadat přímo spolu s dalšími informace o fotografii. Uživatelům aplikace se také nabídne možnost registrovat se, což jim dovolí později upravovat údaje o svých fotografiích, označovat fotky jako oblíbené a případně toto značení zrušit.

Uživatelé po přihlášení budou moci taktéž zakládat nové skupiny. Ty jim dovolí procházet tematicky podobné fotografie a jejich členům přidávat fotografie nové. První člen ve skupině se stane jejím moderátorem. Moderátoři skupiny budou moci schvalovat nové členy, spravovat členství ostatních členů příslušné skupiny, vybírat úvodní snímek skupiny a odebírat z ní příspěvky.

Zároveň bude v aplikaci k dispozici přehled příslušných fotografií na speciální stránce, na mapě, ve skupině, a i na profilu každého uživatele. Taktéž budou uživatelé aplikace moci sdílet příspěvky na sociálních sítích.

4.2 Vzhled aplikace

Webové aplikace založené na Razor Pages používají v základní šabloně, kterou jsem si pro vývoj vybral, framework Boostrap, a tak jsem vzhled konkrétních stránek aplikace nemusel již téměř řešit. Pouze v pozdější fázi vývoje jsem si v souboru `style.css` vytvořil několik vlastních tříd, kterými jsem řešil specifické záležitosti mojí aplikace, především pak vzhled map.

Daleko podstatnější bylo ale vytvořit konkrétní stránky, aby mohl uživatel s aplikací pracovat. Vznikly tak stránky určené pro správu fotografií a skupin, procházení fotografií na mapě a několik dalších.

Snímky obrazovek jsou uvedeny v přílohách.

4.2.1 Rozložení stránek

Aby byl vzhled jednotlivých stránek co nejvíce konzistentní a aby se nemuselo zbytečně definovat rozložení stránky pro každou zvlášť, se v aplikacích založených na

ASP.NET Core používají soubory rozložení, které obsah stránky „obalí“ o tyto společné prvky, např. navigační lišta, patička apod.

V mé aplikaci používám dvě verze takového rozložení. První z nich, `_Layout.cshml`, obsahuje patičku a používá se téměř na všech stránkách aplikace. To druhé, jehož definice se nachází v souboru `_LayoutMinimalistic.cshtml`, patičku nemá a používá se pouze na stránce s hlavní mapou, kde by patička zbytečně zabírala místo na obrazovce. Následující Ukázka kódu 2 představuje část takového souboru, na které můžete vidět, že se obsah stránky načítá pomocí metody `RenderBody()` a že se položky navigační lišty do tagů `header` vkládají ze souboru `_NavMenuPartial`.

Ukázka kódu 2: Část souboru rozložení

```
<header>
    <partial name="_NavMenuPartial" />
</header>
<div class="container">
    <main role="main" class="pb-3">
        @RenderBody()
    </main>
</div>
```

Zdroj: Autor

4.2.2 Úvodní stránka

Úvodní stránka aplikace slouží jako rozcestník na další stránky. Uživatel zde může rychle přejít na seznam skupin, přehled fotografií či velkou mapu světa. Na úvodní stránce se také nachází pole pro zadání místa, které se na mapě zobrazí.

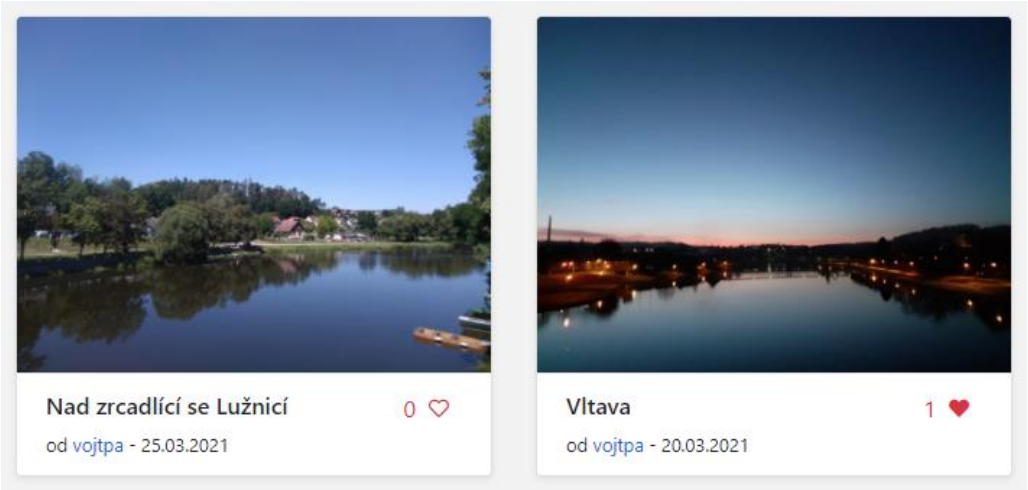
4.2.3 Fotografie

Aby bylo vůbec možné v aplikaci pracovat s fotografiemi uživatelů, vznikly za tímto účelem následující stránky nacházející se v adresáři `Pages/Photos`:

- `Index` – pro zobrazení přehledu všech veřejný fotografií, spolu s neveřejnými fotkami autora,
- `Create` – pro vytvoření nového příspěvku,
- `Upload` – pro nahrání souboru fotografie, součást stránky `Create`,
- `Details` – pro zobrazení detailu nahrané fotky,
- `Edit` – pro úpravu informací fotografie,
- `Delete` – pro smazání nahrané fotky.

Obrázek 6 představuje náhled fotografií na stránce `Index`. Zároveň se na tomto obrázku ukazuje rozdíl mezi příspěvkem na pravé straně, který uživatel označil jako svůj oblíbený, a příspěvkem na levé straně bez takového označení.

Obrázek 6: Náhled fotografií

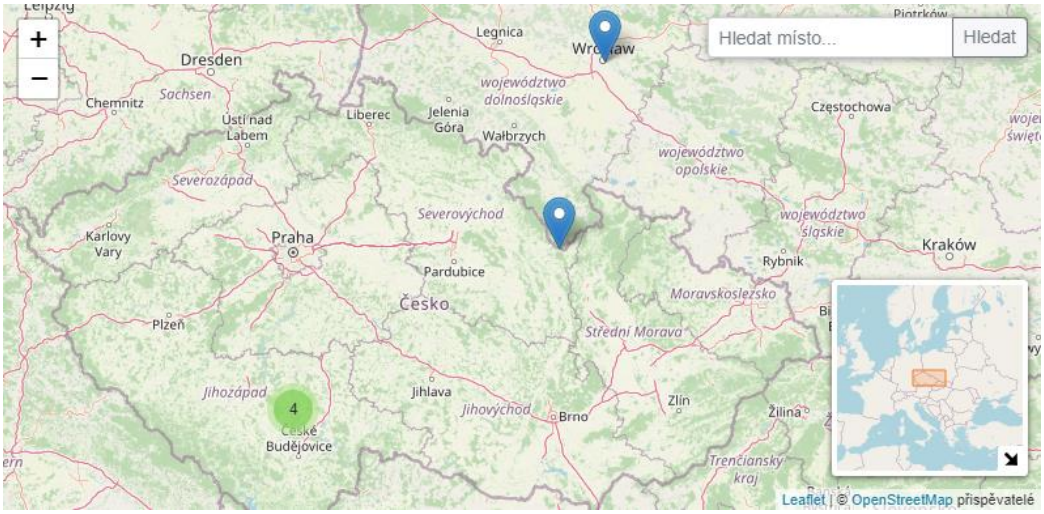


Zdroj: Autor

4.2.4 Mapa

Vytvořením stránky `Map` se sleduje jediný cíl, a to umožnit uživatelům zobrazit si fotografie ve formě jednotlivých bodů umístěných na mapě světa. Nicméně nabízí také pole pro vyhledání konkrétního místa, a pokud se v nějaké části mapy nachází více fotografií blízko u sebe, seskupí se do jednoho bodu.

Obrázek 7: Mapa



Zdroj: Autor

Pro zobrazení mapy, kterou zachycuje Obrázek 7, se používá knihovna *Leaflet* napsaná v jazyce JavaScript. Pokud uživatel přešel na tuto stánku z úvodní stránky a zadal tam do pole hledané místo, pokusí se pomocí vyhledávacího enginu *Nominatin* takové místo najít a nastavit jej jako střed mapy, jinak se použije jako výchozí zobrazovaná oblast Česká republika. Následně dojde ke přizpůsobení ovládacích prvků mapy, určí se OpenStreetMap jako zdroj mapových podkladů a použijí se pluginy, které přidávají mapku poskytující náhled širší oblasti a vyhledávací pole pro rychlý přesun na zadané místo. Posléze dojde k založení clusteru, který umožňuje slučovat více bodů do jednoho, jestliže je mezi nimi krátká vzdálenost. Poté se do tohoto clusteru ve formě jednotlivých bodů vloží náhledy fotografií, které se umístí na mapě na místo, kde byly tyto snímky pořízeny, a následně se tento cluster přidá do mapy.

4.2.5 Skupiny

Motivací pro založení skupin bylo nabídnout uživatelům třídit a sdílet tematicky stejně laděné fotografie (například příroda, zajímavá místa apod.) i jinak než pouze ve formě jednotlivých bodů na mapě. Pro práci se skupinami vzniklo v adresáři `Pages/Groups` pět různých stránek:

- `Index` – pro poskytnutí seznamu skupin,
- `Create` – pro založení nové skupiny,
- `Details` – pro zobrazení fotek ve skupině a správu skupiny, včetně jejích členů,
- `Edit` – pro úpravu názvu a popisu skupiny,
- `Delete` – pro smazání celé skupiny.

Obrázek 8 znázorňuje pohled na seznam skupin na stránce `Index`. Je zde vidět, že u každé skupiny je uveden její název, počet členů a fotografií a jako pozadí úvodní snímek. Pokud ale moderátor skupiny ještě žádný nenastavil, což je případ na obrázku skupiny napravo, nebo už byl smazán, použije se obrázek z webu *Lorem Picsum*.

Obrázek 8: Ukázka seznamu skupin

Zdroj: Autor

4.2.6 Ostatní stránky

Pro práci s uživatelskými účty využívám v aplikaci vygenerovanou sadu stránek identit. Jedná se o všechny stránky umístěné v adresáři `Areas/Identity/Pages`, které dohromady umožňují registraci, přihlašování a odhlašování uživatelů a také úpravu informací o nich.

V aplikaci taktéž můžete najít stránku `Search/Index` pro zobrazení výsledků hledání z výrazu napsaného do pole v navigační liště, podobně tak stránku `Profiles/Index` určenou pro zobrazení fotek konkrétního uživatele a skupin, kterých je členem. A na stránce `Privacy` jsou dostupné informace o cookies, podmínkách použití a použitém softwaru a grafice třetích stran.

4.3 Datový model

V rámci vývoje aplikace jsem si vytvořil datový model ukládaných dat. Protože se o část datového modelu týkající se uživatelských profilů postarala použitá šablona aplikace, musel jsem navrhnout pouze jeho zbývající část. Definice navržených modelových tříd, které používá EF Core pro práci s databází, jsou uloženy ve stejnojmenných souborech v adresáři `Models`. Grafické znázornění tohoto datového modelu je součástí diagramu tříd, který nabízí Příloha A.

Proto, aby bylo schéma databáze ve shodě s těmito modelovými třídami, EF Core nabízí možnost provádět migrace. Prostřednictvím nich se databázové schéma zachovává v souladu s těmito třídami, a přitom nedochází ke smazání již uložených dat v databázi. (Lambson et al., 2020)

4.3.1 Modelová třída Photo

Prvně vytvořená modelová třída `Photo` se používá pro uchovávání informací o dané fotografii. Skládá se z následujících vlastností:

- `PhotoID` (`int`) – jedinečný identifikátor fotografie,
- `Title` (`string`) – název fotografie,
- `Description` (`string`) – popis fotografie,
- `UploadedTime` (`DateTime`) – čas nahrání fotografie do aplikace,
- `PublicPhoto` (`bool`) – hodnota, která určuje, zda bude fotka viditelná i ostatním uživatelům,
- `AuthorID` (`string`) – identifikátor uživatele, který fotku nahrál,
- `FileName` (`string`) – název souboru, pod kterým je fotografie uložena,
- `GPSlat` (`string`) – zeměpisná šířka místa, kde byla fotografie vyfocena,
- `GPSlng` (`string`) – zeměpisná délka místa, kde byla fotka pořízena,
- `CreatedTime` (`DateTime`) – čas, kdy byla fotka vyfocena,
- `Group` (`Group`) – objekt skupiny, kam byla fotografie zařazena,
- `Likes` (`ICollection<PhotoLike>`) – seznam lajků fotografie.

V této modelové třídě se taktéž omezují hodnoty, které lze zadat. Například Ukázka kódu 3 ukazuje, že u zeměpisné šířky jsou pomocí regulárního výrazu její možné hodnoty omezeny na číslo v rozmezí od -90 do 90.

Ukázka kódu 3: Omezení hodnot zeměpisné šířky

```
[Required (ErrorMessage = "{0} je nutné vyplnit")]
[Display(Name = "Zeměpisná šířka")]
[RegularExpression(@"^([-+]?)\b([0-9]|[1-8][0-9])(((.)(\d+))?)$",
    ErrorMessage = "Zeměpisná šířka může být číslo v rozmezí od -90 do 90")]
16 references
public string GPSlat { get; set; }
```

Zdroj: Autor

4.3.2 Modelová třída PhotoLike

Protože mají přihlášení uživatelé možnost vyjádřit, že se jim daná fotografie líbí, vytvořil jsem modelovou třídu `PhotoLike`, která se skládá z těchto vlastností:

- `PhotoLikeID` (`int`) – jednoznačný identifikátor lajku,
- `Photo` (`Photo`) – objekt fotografie, která se uživateli líbí,
- `UserID` (`string`) – identifikátor takového uživatele,
- `LikedTime` (`DateTime`) – čas, kdy uživatel fotku označil lajkem.

4.3.3 Modelová třída Group

Modelová třída `Group` definuje ukládaná data o skupinách. Tvoří ji tyto vlastnosti:

- `GroupID` (`int`) – unikátní identifikátor skupiny,
- `GroupName` (`string`) – název skupiny,
- `Description` (`string`) – popisek skupiny,
- `PhotoTitle` (`Photo`) – objekt fotografie, která je označena jako úvodní obrázek skupiny,
- `Photos` (`ICollection<Photo>`) – kolekce všech objektů fotek zařazené do dané skupiny.

4.3.4 Modelová třída GroupMembership

Bez modelové třídy `GroupMembership` by se v aplikaci nemohli uživatelé přidávat do skupin. Vlastnosti, ze kterých je tato třída složena, jsou následující:

- `GroupMembershipID` (`int`) – jedinečný identifikátor členství ve skupině,
- `Group` (`Group`) – objekt skupiny, které je daný uživatel členem,
- `UserID` (`string`) – identifikátor uživatele jako člena skupiny,
- `MembershipCreatedTime` (`DateTime`) – čas, kdy bylo členství vytvořeno,
- `MemberRole` (`Role`) – role uživatele ve skupině.

V poslední jmenované vlastnosti se používá výčtový datový typ `Role`, jenž se skládá z členů `Empty` značící, že uživatel není členem dané skupiny, a `Moderator`, `Regular` a `Waiting` určených pro specifikování konkrétní role v příslušné skupině.

4.4 Aplikační logika

Prostřednictvím aplikační logiky se definuje průběh činností, od vstupních dat až k výstupům těchto činností. Za tímto účelem jsem vytvořil několik tříd, které jednotlivé stránky v aplikaci používají buďto k ukládání dat, nebo k jejich zobrazení.

Metody obsluhující činnosti, u kterých se dá úspěšně předpokládat, že budou trvat delší dobu (např. přístup do databáze), jsem se snažil vytvořit jako metody asynchronní. Tím by měla být aplikace rychlejší, protože nemusí čekat až se tyto úlohy dokončí a může se mezitím věnovat i dalším úkonům. Názvy těchto metod proto končí slovem `Async`.

Pro úplnost dodávám, že všechny třídy, které uvádím níže a které končí na `Service`, se nacházejí v adresáři `Services`. Zároveň velká část metod, ze kterých se tyto třídy skládají, má parametr typu `ClaimsPrincipal`. Tímto způsobem se do příslušných metod předává identita uživatele, který danou akci vyvolal.

4.4.1 Datové struktury

Současně většina těchto tříd, respektive některé jejich metody, vrací různé datové struktury. Děje se tak, protože při čtení určitých dat se načítají i data, která s nimi pouze souvisejí, ale jsou uložena jinde, a použití těchto struktur mi pak na stránkách usnadnilo přístup ke všem těmto datům.

V aplikaci nejpoužívanější datová struktura `PhotoStruct` se skládá z datových členů `photo` (datový typ `Photo`), do kterého se ukládá objekt fotografie, `author` (`GeoscenaUser`) uchovávající entitu uživatele, který danou fotku nahrál, `likes` (`int`) pro počet lajků fotografie, `liked` (`bool`) pro určení, zda si uživatel fotku již oblíbil, a `owner` (`bool`) ke zjištění, jestli je uživatel autorem této fotografie.

V další datové struktuře `GroupStruct` se nachází datové členy `group` (`Group`), kam se ukládá objekt skupiny, `photosCount` (`int`) pro načtení počtu fotek ve skupině, `usersCount` (`int`) pro získání počtu členů dané skupiny a `member` (`bool`) pro určení, jestli je uživatel členem skupiny.

Poslední vytvořená datová struktura je `UserStruct`. Zde jsou vytvořeny datové členy `geoscenaUser` (`GeoscenaUser`), jenž uchovává entitu uživatele jakožto člena skupiny, `role` (`GroupMembership.Role`) pro načtení druhu role v této skupině a do `created` (`DateTime`) se ukládá čas, kdy uživatel tuto roli získal. Důvodem pro vznik této datové struktury bylo přání, aby byl seznam uživatelů uvedený na stránce skupiny primárně seřazený podle významu jejich rolí, a tak jsem vytvořil třídu `MembersByRole` implementující rozhraní `IComparer`, které se stará o třídění a které však vyžadovalo použití datové struktury.

Grafické znázornění všech těchto datových struktur je součástí diagramu tříd, který uvádí Příloha A.

4.4.2 Nahrávání fotografií

Pro nahrávání fotografií se v aplikaci používá stránka `Create` v adresáři `Pages/Photos`, kam uživatelé zadávají informace o dané fotografii. Tato stránka se pomocí JavaScriptu po vložení souboru fotografie pokusí z EXIF dat zjistit, kdy a kde byl snímek pořízen, a pokud tato data nalezne, vyplní příslušná pole pro zadání času vyfocení a zeměpisné šířky a délky. Protože ale souřadnice uložené v EXIF mají jiný zápis, než s jakým se v aplikaci dále pracuje, je proto nutné tyto údaje o poloze nejprve převést do

potřebné podoby. Například Ukázka kódu 4 představuje způsob, kterým se získá zeměpisná šířka a vloží se do náležitého pole na stránce.

Ukázka kódu 4: Získání zeměpisné šířky

```
exif = EXIF.getTag(this, "GPSLatitude");
console.log("Lat:" + exif);
if (exif != "undefined") {
    var deg = exif.toString().split(",");
    if (deg.length == 3) {
        var lat = Number(deg[0]) + Number(deg[1]/60) + Number(deg[2]/3600);
        document.getElementById('gps-lat').value = lat;
    }
}
```

Zdroj: Autor

S odesláním formuláře na této stránce, se nahraje do aplikace nahraje i daný soubor, tentokrát však přes stránku `Upload` v témže adresáři. Pro uživatele se ale jeví, že vše probíhá na jednom místě. S daty z obou stránek se pak dále pracuje už jen ve třídě `CreateModel`, což je `PageModel` stránky `Create`, respektive v její metodě `OnPostAsync()`, která se volá po přijetí dat formuláře z této stránky.

Tato metoda postupně přidává data do objektu `Photo`. Nejdříve ověří, zda je uživatel přihlášen, a pokud ano, tak přidá jeho identifikátor do vlastnosti `AuthorID`. V opačném případě zůstane zde hodnota `null` a vynutí se, že fotografie bude veřejná nastavením hodnoty `PublicPhoto` na `true`. Následuje vložení časového razítka do vlastnosti `UploadedTime` a vygenerování náhodného názvu souboru, pod jakým má být nahrávaná fotografie uložena. Pak se pomocí metody `ImageValidate()` potvrdí, že nahraný soubor je skutečně obrázek. Pokud tímto testem neprojde, dojde k označení modelu za chybný, což znamená, že se poté vrátí původní stránka pro nahrání fotografie s vyznačenými chybami.

Jestliže je model v pořádku, přidá se do vlastností fotografie již vygenerovaný název souboru jako `FileName` a do `Group` skupina, do které fotku uživatel zařadil a zároveň které je členem. Potom se soubor fotografie uloží pod daným jménem do `pictures` a, jak ukazuje Ukázka kódu 5, pomocí balíčku NuGet `SixLabors.ImageSharp` se vygeneruje náhled fotky, který má na výšku vždy 288 pixelů, což by mělo nabídnout dostatečný kompromis mezi velikostí souboru a rozlišením, a přitom zachovává původní poměr stran fotografie, a tak nedochází k jejím zbytečným deformacím. Načež se přidá

záznam o této nové fotografii do databáze a přejde se na stránku s přehledem fotografií. Tím je celý proces nahrávání fotografie dokončen.

Ukázka kódu 5: Generování náhledu fotografie

```
using (var image = Image.Load(path))
{
    // Nastavení velikosti náhledu na výšku 288px při zachování poměru stran
    double height = 288, width = (height / image.Height) * image.Width;

    image.Mutate(c => c.Resize((int)width, (int)height));
    var pathThumbnail = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "thumbnails", filename);
    await image.SaveAsync(pathThumbnail);
}
```

Zdroj: Autor

4.4.3 Třída UserInfoService

Účelem třídy `UserInfoService` je poskytnout informace o příslušném uživateli. Tato třída je velmi stručná, skládá se z pouze jedné metody `GetUserID()`, jež má parametr typu `ClaimsPrincipal`, prostřednictvím kterého se předává identita příslušného uživatele. V případě, že je tento uživatel přihlášený, vrátí jeho identifikátor, jinak bude výstupem této metody hodnota `null`.

4.4.4 Třída PhotoService

Třída `PhotoService` je v aplikaci klíčová. Prostřednictvím této třídy se načítají fotografie, umožňuje také jejich úpravu či je mazat. Zároveň využívá pro svoji činnost i jiné třídy.

Asynchronní metoda `GetPhotoAsync()` slouží k načtení seznamu všech fotografií, které patří přihlášenému uživateli. Z identity přihlášeného uživatele získá jeho uživatelské ID a do seznamu fotografií načte z databáze ty, které mají u vlastnosti `AuthorID` stejnou hodnotu, jaká byla získána v předcházejícím kroku. Tento seznam metoda následně vrátí jako svůj výstup.

Další asynchronní metoda je `GetPhotoStructsAsync()`. U ní dochází k přetěžování podle toho, s jakými parametry se zavolá, čímž pak upravuje svoje chování. Tato metoda pracuje vždy s alespoň jedním argumentem, a sice identitou uživatele, a zároveň funguje obecně tak, že si vytvoří seznam požadovaných objektů fotografií a následně do dalšího seznamu datových struktur `PhotoStruct` postupně vkládá příslušné prvky získané metodou `GetPhotoStructAsync()`, kde se jako argument použijí fotografie z prvně uvedeného seznamu. Výstupem této celé metody je nově naplněný seznam prvky `PhotoStruct`.

Pakliže je tato metoda volána pouze s argumentem identity uživatele, vrátí seznam náležejících struktur naplněný všemi veřejnými fotografiemi a neveřejnými fotkami přihlášeného uživatele. Pokud se stejná metoda volá i s identifikátorem libovolného uživatele jako jejím argumentem, načte se seznam struktur s fotkami, ke kterým má ten uživatel, jenž tuto akci vyvolal, přístup. A jestliže je ve stejné metodě součástí parametru objekt skupiny, je výsledkem seznam struktur pro daného uživatele dostupných fotografií, které jsou do této skupiny zařazeny.

V této třídě dále následuje metoda `GetPhotoStructAsync()`, jež je opět asynchronní a opět se jedná o metodu, která je přetěžována. Jejím účelem je vytvořit a vrátit datovou strukturu `PhotoStruct`. Jestliže byla metoda zavolána s argumenty objektu fotografie a identitou uživatele, pokusí se nejprve nalézt autora dané fotky, dále získá pomocí metody `GetLikeCount()` ze třídy `LikeService` počet lajků fotografie, potom se určí, zda uživatel, který akci vyvolal, si sám fotku oblíbil a zda je ten, kdo ji do aplikace nahrál. To jsou již dostatečné informace, které tato metoda potřebuje k vytvoření patřičné struktury, a tak ji může vrátit jako výsledek. Ukázka kódu 6 nabízí ukázku kódu, který se v této metodě používá.

Ukázka kódu 6: Získání datové struktury PhotoStruct

```
// nalezení autora
GeoscenaUser author = await _userManager.FindByIdAsync(photo.AuthorID);

int likes = _likeService.GetLikeCount(photo);

// ověření přihlášeného uživatele
string userID = _userInfoService.GetUserID(claimsPrincipal);
bool liked = false;
bool owner = false;

if (userID != null)
{
    // zjištění, zda se uživateli fotka líbí
    PhotoLike photoLike = await _likeService.GetLikeInfoAsync(photo, userID);
    if (photoLike != null)
    {
        liked = true;
    }

    // zjištění, zda je daný uživatel autorem fotky
    if (photo.AuthorID == userID)
    {
        owner = true;
    }
}

return new PhotoStruct(photo, author, likes, liked, owner);
```

Zdroj: Autor

Stejnou metodu je možné také zavolat i s identifikátorem fotografie a identitou uživatele. V tomto případě metoda získá z ID fotky celý objekt fotografie a zavolá sebe samu, ovšem s objektem fotky a identitou uživatele jako příslušné parametry, čímž se celá akce převede na předchozí případ.

Pomocí asynchronní metody `UpdatePhotoAsync()` je možné také ukládat upravené informace o fotografiích. Tato metoda očekává v argumentu již upravený objekt fotografie, identifikátor skupiny, do které má být fotografie zařazena, a identitu uživatele. Postup ukládání probíhá tak, že se zavolá ze třídy `GroupService` metoda `GetGroupAsync()`, předá se jí jako argument identifikátor skupiny a hodnota, kterou vrátí, se přiřadí do vlastnosti `Group` aktualizované fotografie. Následně se tato metoda pokusí uložit nové hodnoty do databáze, a nenastane-li chyba, vrátí hodnotu `true`.

Další přetěžovanou a asynchronní metodou je `DeletePhotoAsync()`, která umožňuje příslušnou fotografii smazat. Pokud je zavolána pomocí argumentů objektu fotografie a identity uživatele, nejdříve se metodou `IsOwner()` ověří, že uživatel, jenž tuto akci vyvolal, je zároveň ten, kdo tuto fotku do aplikace nahrál, a může ji tak smazat. Není-li

tomu tak, metoda už jen vrátí hodnotu `false`. Jinak se pokračuje odebráním záznamu o fotografii z databáze, smazáním souborů fotografie, včetně jejího náhledu, a vrácením hodnoty `true` na znamení, že vše proběhlo v pořádku.

Stejnou metodu je možné zavolat i argumenty identifikátoru fotografie a identity uživatele. V tomto případě se k takového identifikátoru nalezne odpovídající objekt fotografie a zavolá metoda zavolá sebe samu, avšak jako parametry se použijí právě takto získaný objekt spolu s původní identitou uživatele.

Již zmíněná metoda `IsOwner()` se skládá také z parametrů objektu fotografie a identity uživatele. Jejím cílem je určit, zda příslušný uživatel je zároveň tím, kdo danou fotku do aplikace nahrál. Aby byla vrácená hodnota `true`, nesmí být objekt fotografie prázdný, a přitom hodnota jeho vlastnosti `AuthorID` je shodná s identifikátorem daného uživatele.

Prostřednictvím metody `GroupsAuthorizedToUpload()` se najdou skupiny, do kterých může daný uživatel vkládat svoje fotografie. Tato asynchronní metoda, která bere jako parametr identitu uživatele, pouze předává seznam skupin vrácený metodou `GetSubscribedGroupsAsync()` ze třídy `GroupService`.

Podobně tak se chovají i asynchronní metody `LikePostAsync()` a `UnLikePostAsync()`, které pouze volají stejnojmenné metody ve třídě `LikeService` a předávají jim svoje parametry, aby umožnili přihlášeným uživatelům označovat příslušné fotky za oblíbené, nebo jim toto označení odebrat.

Asynchronní metoda `FindPhotosAsync()`, která očekává v argumentu hledaný výraz a identitu uživatele, vyhledává v názvu a popisku fotografie tento zadaný text. Na začátku si pomocí výrazu, jenž uvádí Ukázka kódu 7, načte do seznamu objektů fotografií všechny pro daného uživatele přístupné fotky a poté z nich prostřednictvím metody `GetPhotoStructAsync()` získá požadované struktury. Právě ty postupně přidává do dalšího seznamu, který je vytvořený speciálně pro tyto struktury, přičemž výsledkem celé této metody je vrácení tohoto seznamu.

Ukázka kódu 7: Získání objektu fotografie podle hledaného textu

```
List<Photo> photos = await _context.Photo.Where(p => (
    p.Title.Contains(query) || p.Description.Contains(query)) &&
    (p.PublicPhoto == true || p.AuthorID == userId)
    ).OrderByDescending(p => p.UploadedTime)
    .ToListAsync();
```

Zdroj: Autor

4.4.5 Třída LikeService

Účelem třídy `LikeService` je poskytnout metody, které řeší nastavování fotografií jako oblíbených a které poskytují informace o těchto označeních.

Za označení fotografie jako oblíbené je zodpovědná asynchronní metoda `LikePostAsync()`, jež bere jako svoje parametry objekt fotografie a identitu uživatele, ze které následně získává identifikátor uživatele. Poté se pokusí najít, zda v databázi již není už uložen jeden záznam o lajku u dané fotografie od tohoto uživatele. Jestliže žádný takový záznam nenajde, metoda jej vytvoří a následně ho uloží.

Naproti tomu asynchronní metoda `UnLikePostAsync()` se stará o zcela opačný proces. Opět si bere jako argumenty objekt fotografie a identitu uživatele a znovu si najde identifikátor uživatele, který vyjádřil přání svůj lajk odebrat. Metoda se posléze pokusí v databázi vyhledat, zda se takový záznam existuje, a pokud ano, z databáze jej vymaže.

Asynchronní metoda `GetLikeInfoAsync()` vyhledá v databázi a následně vrátí záznam o lajku dané fotografie a příslušného uživatele, který ji označil jako oblíbenou. K tomu slouží této metodě její parametry, tedy objekt fotografie a identifikátor uživatele.

Metodou `GetLikeCount()` se vrátí počet všech lajků příslušné fotografie, jejíž objekt byl předán v parametru této metody.

4.4.6 Třída GroupService

Třída `GroupService` se stará zejména o získávání informací o skupinách a jejich úpravu. Zároveň je ale možné ji využít pro změnu rolí členů v těchto skupinách, včetně vytváření a rušení jejich členství v nich.

Pro získání objektu skupiny se používá přetěžovaná asynchronní metoda `GetGroupAsync()`. Je-li ní předán pouze identifikátor skupiny, načte se z databáze objekt skupiny, která tento identifikátor obsahuje.

Jinou možností, jak tuto metodu zavolat, je předat jí identifikátor skupiny spolu s identitou přihlášeného uživatele. V tomto případě si načte seznam skupin, kde je příslušný uživatel schváleným členem, k čemuž využije metodu `GetSubscribedGroupsAsync()`, a v tomto seznamu se pokusí najít a vrátit takovou skupinu, jejíž identifikátor je stejný jako ten v argumentu.

Zmíněná asynchronní metoda `GetSubscribedGroupsAsync()` pracuje následovně. Pokusí se zjistit si identifikátor uživatele předaný v jejím parametru, a pokud nějaký takový identifikátor nalezne, načte si do seznamu všechny informace o členstvích ve skupinách, které se váží k tomuto uživateli a zároveň které již moderátor skupiny schválil. Z těchto všech informací metoda už dále jen vrátí seznam skupin, jež tímto filtrem prošly. Jestli identifikátor uživatele nebyl nalezen, vrátí se hodnota `null`.

Další přetěžovanou asynchronní metodou je `GetGroupStructAsync()`. A ačkoli je možné ji také zavolat jen s pomocí parametru pro objekt skupiny, v ideálním případě by měl být v parametru zároveň uveden i textový identifikátor uživatele. Výsledkem je v obou případech datová struktura `GroupStruct`, jež se skládá z předaného objektu skupiny, zjištěného množství fotek v dané skupině metodou `CountPhotos()`, počtu jejích členů z metody `CountUser()` a pokud je uveden i uživatel, tak se pomocí metody `IsMember()` určí, zda on sám k ní náleží, jinak bude poslední hodnota nastavena na `false`. Tento postup jsem zvolil kvůli tomu, aby, když už se o výsledku poslední hodnoty předem ví, že bude nepravda, bylo vytvoření této datové struktury rychlejší.

Právě toho se využívá v asynchronní metodě `GetGroupsAsync()` určená pro získávání seznamu datových struktur skupin. Ta potom, co si načte seznam všech skupin, se pokusí zjistit uživatelský identifikátor. Pokud se jí to podaří, postupně načte do příslušného seznamu struktur `GroupStruct` metodou popsanou v předchozím odstavci a jako argumenty jí předá objekt skupiny a získaný identifikátor uživatele. Avšak jestli žádný takový identifikátor nenajde, znamená to, že uživatel, který tuto akci vyvolal, není přihlášený, a tak není třeba u každé skupiny ověřovat, zda je, nebo není členem jakékoli skupiny, protože výsledek je znám již dopředu, a proto se může pro získávání těchto struktur bez obav zavolat metoda `GetGroupStructAsync()` jen s argumentem objektu skupiny. Nakonec se vrátí takto naplněný seznam datových struktur `GroupStruct`.

Pro hledání skupin podle jejich názvu se používá asynchronní metoda `FindGroupsAsync()`, která ve svém parametru očekává hledaný textový řetězec. Výstupem je seznam všech skupin, které mají ve svém názvu tento text.

Pomocí asynchronní metody `GetGroupMembersAsync()` se získá seznam datových struktur `GroupMembership`, které se skládají z podrobností o členství všech členů dané skupině, jež je předána metodě jako její parametr. Výsledný seznam je seřazen tak, že uživatelé s rolí s vyššími právy jsou uváděni dříve.

Seznam informací o všech členství daného uživatele ve skupinách vrací asynchronní metoda `GetGroupMembershipsAsync()`, která přijímá identifikátor uživatele jako svůj parametr.

K získání konkrétní role v určité skupině se použije asynchronní metoda `GetRoleAsync()`, jež příjme jako svůj argument objekt dané skupiny a buďto identifikátor uživatele, nebo jeho identitu.

Asynchronní metoda `SetRoleAsync()` se stará o nastavování, úpravu a odebírání členství uživatelů ve skupinách. Jejími parametry jsou identifikátor skupiny, které se toto členství týká, nově přiřazená role, identita uživatele, který akci vyvolal, a případně i identifikátor uživatele, jehož role se mění. Metoda po ověření, že předané argumenty jsou platné, a že původce této akce má dostatečná práva na změnu členství ve skupině, nastaví u daného uživatele jeho novou roli. Jestliže celá akce proběhne v pořádku, metoda navrátí hodnotu `true`.

Pro zjednodušení je možné použití metody `SubscribeGroupAsync()` pro požádání o členství samotným uživatelem a metody `UnsubscribeGroupAsync()` pro zrušení vlastního členství ve skupině. Obě tyto asynchronní metody volají metodu `SetRoleAsync()` s potřebnými argumenty.

Nastavit a změnit úvodní fotografii skupiny umožňuje asynchronní metoda `SetPhotoAsTitleAsync()`. Pro svoji činnost přebírá ve svých parametrech identifikátory dané skupiny a nové úvodní fotografie a identitu uživatele. Po kontrole správnosti těchto údajů a potvrzení, že tento uživatel je moderátorem skupiny, metoda přiřadí do vlastnosti `PhotoTitle` této skupiny objekt nově zvolenou fotografii a změny uloží do databáze. Pokud vše proběhne bez chyby, vrátí se hodnota `true`.

4.4.7 Ostatní třídy a rozhraní

Třída `EmailSender`, která realizuje rozhraní `IEmailSender`, obsahuje pouze metodu `SendEmailAsync()`, jež slouží pro odesílání e-mailů z aplikace. Tato metoda si ve svých parametrech bere e-mailovou adresu příjemce, předmět a text zprávy zasílané zprávy. Aplikace totiž při zakládání účtu zasílá email, aby jej ověřila, stejně tak posílá odkaz emailem odkaz na resetování hesla, když jej uživatel zapomene. Využívá se zde řešení *Mailjet*.

Aplikace dále obsahuje velké množství tříd, především pak se jedná o třídy spojené s jednotlivými stránkami, ale rozhodl jsem se, že je nechám bez dalšího popisu. Přesto je jejich kód k dispozici v Příloze G této práce.

Pokládám však za vhodné ještě zmínit, že několik těchto tříd realizuje rozhraní, která jsem si zavedl pro potřeby této aplikace. Jejich definice je uložena ve stejnojmenných souborech z adresáře `Interfaces`.

Prvním takovým je rozhraní `ILike`, které vynucuje implementaci dvou metod určených pro vypořádání se s funkčností tlačítek pro oblíbení si dané fotografie, popřípadě zrušení tohoto označení. Konkrétně pak se jedná o metody `OnPostAddLikeAsync()` a `OnPostRemoveLikeAsync()`.

Dalším takovým rozhraním je `IManagePost`, jež ukládá zavést metody pro správu příspěvků ve skupině, a to `OnPostRemoveFromGroup()` a `OnPostSetAsTitleImage()`.

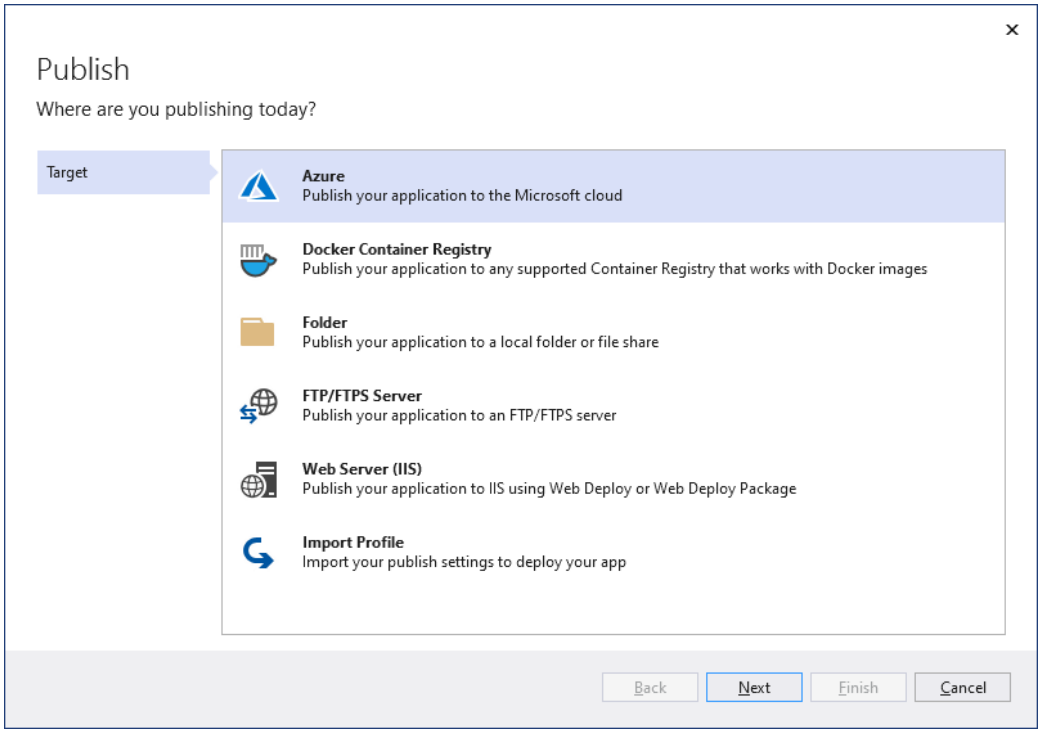
A konečně pak rozhraní `IMembership`, které vyžaduje zavedení několika metod používaných pro řízení členství uživatelů ve skupině. Metody požadované tímto rozhraním jsou `OnPostSubscribe()`, `OnPostUnsubscribe()`, `OnPostSetAsModerator()`, `OnPostSetAsRegular()` a `OnPostRemoveUser()`.

4.5 Nasazení aplikace

Když bylo možné aplikaci bez větších problémů používat, rozhodl jsem se ji nasadit na cloudovou službu Microsoft Azure. K tomuto je ale nutné mít účet Microsoft s aktivovaným předplatným pro tuto službu. Rozhodl jsem se využít nabídku pro studenty, kterým společnost Microsoft nabízí předplatné Azure na jeden rok zdarma, ačkoli jsou služby v něm omezené. A i když Microsoft vyžadoval prokázat, že mám na tuto nabídku nárok, celé ověření nebylo nijak složité, protože stačilo pouze se přihlásit pomocí univerzitního emailu.

Nasazení webové aplikace do Azure probíhá ve Visual Studiu tak, že se v nabídce `Build` vybere položka `Publish` a následně se postupuje podle instrukcí zobrazeného průvodce. Ten, jak ukazuje Obrázek 9, se v první kroku zeptá na cílovou platformu, kde má být aplikace nasazena, což je právě Azure, a poté přijde na řadu zvolit konkrétní typ služby, která se použije. Rozhodl jsem se pro *Azure App Service (Linux)*. Následně průvodce vyzve k určení konkrétní instance služby, kde bude aplikace nasazena, současně ale umožní i takovou instanci vytvořit.

Obrázek 9: Průvodce pro nasazení aplikace

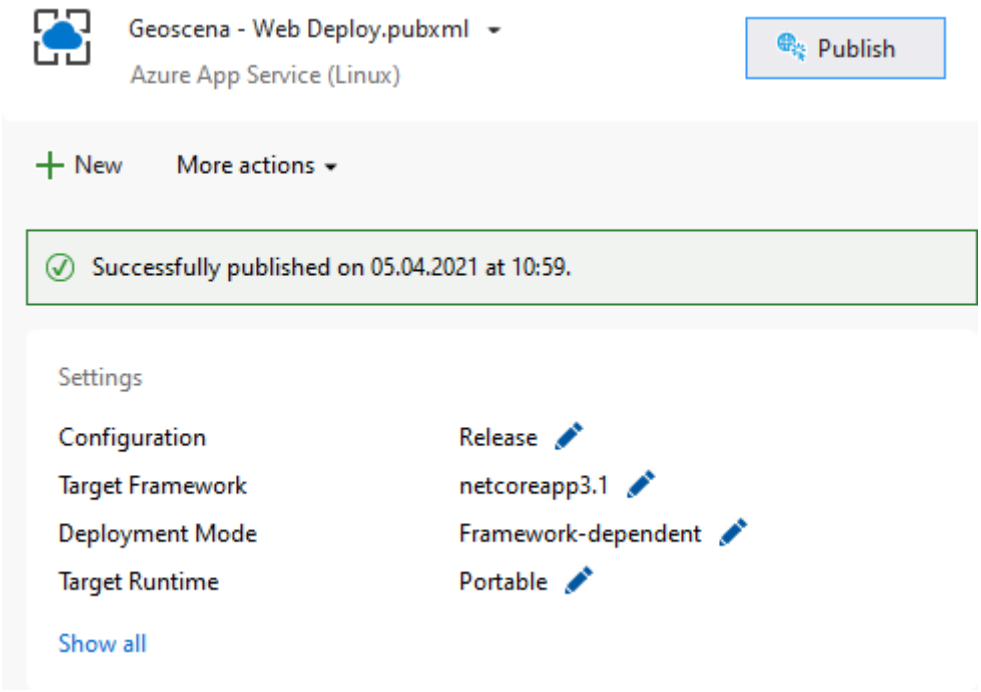


Zdroj: Autor

Pro zprovoznění databáze na Azure je potřeba ještě nastavit závislost. Za tímto účelem je opět ve Visual Studiu k dispozici průvodce, který v prvním kroku vyzývá k vybrání náležité služby. Zde jsem si vybral službu *Azure SQL Database*. Pokud se již v Azure nachází vytvořená databáze, je možné si ji v dalším kroku zvolit pro potřeby nasazované aplikace, zároveň průvodce nabízí vytvoření nové databáze na Azure. Právě k tomu je nutné vybrat SQL server, na kterém bude databáze uložena, avšak je zde také k dispozici možnost tento server vytvořit nový. Po vybrání příslušné databáze se do průvodce dále zadají přihlašovací údaje k danému SQL serveru a tím je konfigurace této služby již u konce.

Přesto, aby se definice databáze z modelových tříd popsaných v kapitole 4.3 projevila i ve schématu databáze na Azure, musel jsem ještě v dialogu nastavení povolit provedení migrace během procesu publikování. Tento dialog lze vyvolat na obrazovce, kterou představuje Obrázek 10, kliknutím na text `Show all`.

Obrázek 10: Obrazovka nasazení



Zdroj: Autor

K nasazení aplikace slouží tlačítko `Publish`, které taktéž zobrazuje Obrázek 10. Prostřednictvím tohoto tlačítka jsem tak nasadil aplikaci do reálného prostředí a zpřístupnil jsem ji všem zájemcům na webové adrese geoscena.azurewebsites.net. V případě, že se následně publikuje nová verze aplikace, už není za potřebí tento zdlouhavý proces procházet od začátku, ale stačí pouze znovu použít toto tlačítko. Tím dojde k nasazení nové verze aplikace do Azure.

5 Závěr

Praktickým výstupem této bakalářské práce je webová aplikace GEOscéna, která umožňuje sdílet fotografie a pokládat je ve formě bodů na mapu. Během nahrávání fotografie se aplikace pokusí načít z metadat souřadnice místa, kde byla vyfocena, popřípadě je uvede uživatel spolu s dalšími informacemi o fotografii. V aplikaci je možné také vyhledávat příspěvky podle jejich názvu a popisu, podobně tak mohou hledat skupiny podle jejich jména. Zároveň je možné příspěvky sdílet na sociálních sítích. Registrovaní uživatelé mohou navíc označovat fotografie jako oblíbené, zakládat skupiny a zařazovat tam svoje fotografie na dané téma.

Při vývoji aplikace jsem na základě zpětné vazby upravil chování tlačítka pro uložení nového příspěvku. A tak potom, co na něj uživatel klikne, dojde k jeho deaktivaci, aby uživatelé nemohli při čekání na uložení nové fotografie ji omylem nahrát vícekrát.

Do budoucna je možné přidat do aplikace funkce, které umožní uživatelům například zakládat si svoje vlastní seznamy příspěvků, upravovat fotografie či vkládat mapy s fotografiemi na další stránky.

Vytvořená aplikace je založena na platformě ASP.NET Core. K jejímu vývoji jsem použil Razor Pages, což je podle mého názoru elegantní technologie pro tvorbu webových aplikací, mapové podklady OpenStreetMap, a zhotovenou aplikaci jsem poté nasadil na cloudovou službu Microsoft Azure s využitím App Service a Azure SQL Database. Na adrese geoscena.azurewebsites.net je dostupná veřejnosti, zároveň zdrojové kódy aplikace a snímky obrazovek jsou k dispozici v přílohách této práce.

V teoretické části jsem začal popisem webových aplikace a fází jejich životního cyklu. Uvedl jsem v ní také technologie ASP.NET Core a Razor Pages, programovací jazyk C#, vývojové prostředí Visual Studio, framework Bootstrap a balíčky NuGet. Rovněž jsem se v teoretické časti zmínil o databázích, Microsoft Azure, fotografiích a způsobu získávání souřadnic GPS.

Samotnou aplikaci jsem popsal v praktické části. V ní jsem se zabýval stanovení požadavků na vyvíjenou aplikaci a představil jsem strukturu v aplikaci používaných stránek. Taktéž jsem se věnoval popisu navrženého datového modelu. Následně jsem zdokumentoval aplikační logiku, zejména pak způsob fungování jednotlivých tříd, respektive jejich metod. Praktickou část uzavírám popisem konfigurace nezbytných služeb pro používání aplikace v Azure pomocí Visual Studia.

I. Summary and keywords

This paper describes a design and development of a web application for sharing photographs on a map. Additionally, this app allows to users use groups and put their photos in there on the chosen topic too. The created web application is based on modern ASP.NET Core technology and deployed on Microsoft Azure cloud service. Furthermore, a Razor Pages framework and Visual Studio were used during the development of this web app. The theoretical part primarily describes web applications in general, their life cycle, developer tools, and databases. Additionally, the practical part solves requirements for the web application, explains application logic, and describes the proposed data model. It also presents the process of publishing the app to Microsoft Azure in Visual Studio.

Key words: web application, ASP.NET Core, Razor Pages, map

II. Seznam literatury

App Service [Online]. (n.d.). Retrieved March 17, 2021, from https://azure.microsoft.com/en-us/services/app-service/

Azure SQL Database [Online]. (n.d.). Retrieved March 17, 2021, from https://azure.microsoft.com/en-us/services/sql-database/

Bernard, B. (2009a, May 07). Úvod do architektury MVC [Online]. Retrieved March 29, 2021, from https://zdrojak.cz/clanky/uvod-do-architektury-mvc/

Bernard, B. (2009b, May 15). Alternativy k MVC a závěrečné poznámky [Online]. Retrieved March 29, 2021, from https://zdrojak.cz/clanky/alternativy-k-mvc-a-zaverecne-poznamky/

Birnir, A. (2020, April 23). WEBSITES VS. WEB APPLICATIONS [Online]. Retrieved March 13, 2021, from https://skillcrush.com/blog/websites-vs-web-applications/

Bootstrap 4 Grid System [Online]. (n.d.). Retrieved March 29, 2021, from https://www.w3schools.com/bootstrap4/bootstrap_grid_system.asp

Brzóska, M. (2020). *Nový zeměpis I. v kostce pro SŠ*. Praha: Fragment.

Casteleyn, S., Daniel, F., Dolog, P., & Matera, M. (2009). *Engineering Web Applications*. Dordrecht: Springer. https://doi.org/10.1007/978-3-540-92201-8

Čápka, D. (2020, November 20). Úvod do kolekcí a genericita [Online]. Retrieved March 14, 2021, from https://www.itnetwork.cz/csharp/kolekce-a-linq/c-sharp-tutorial-uvod-do-kolekci-a-genericita/

Dajbych, V. (2009, April 21). MVVM: Model-View-ViewModel [Online]. Retrieved March 30, 2021, from https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

Dean, J. (2019). *Web programming with HTML5, CSS, and JavaScript*. Burlington, MA: Jones & Bartlett Learning.

Dietrich, E., Mutta, E., Wagner, B., Kulikov, P., Warren, G., Roszko, M., et al. (2020, August 04). The history of C# [Online]. Retrieved March 14, 2021, from https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history

Douglas, J., Jones, M., Zivkovic, A., Gill, C. R., Sharma, L., Nandwani, K., et al. (2019, May 24). An introduction to NuGet [Online]. Retrieved March 20, 2021, from https://docs.microsoft.com/en-us/nuget/what-is-nuget

Gála, L., Pour, J., & Šedivá, Z. (2015). *Podniková informatika: počítačové aplikace v podnikové a mezipodnikové praxi* (3., aktualizované vydání). Praha: Grada Publishing.

Hanák, J. (2008). *Objektovo orientované programovanie v jazyku C# 3.0: (príručka pre vývojárov, programátorov a softvérových expertov)*. Brno: Artax.

Handle the product creation form submission [Online]. (n.d.). Retrieved March 18, 2021, from https://docs.microsoft.com/en-us/learn/modules/create-razor-pages-aspnet-core/6-create-pagemodel

Chand, M. (2014, January 28). Dynamic Keyword in C# [Online]. Retrieved March 14, 2021, from https://www.c-sharpcorner.com/UploadFile/mahesh/dynamic-keyword-in-C-Sharp/

Chiaretta, S. (2018). *Front-end Development with ASP.NET Core, Angular, and Bootstrap*. Indianapolis, IN: John Wiley.

Kroenke, D., & Auer, D. J. (2015). *Databáze*. Brno: Computer Press.

Kuruc, J. (2011). Vím, kde jsem. *Computer*, *18*(13-14), 8-9.

Lacko, Ľ. (2011a). *1001 tipů a triků pro SQL*. Brno: Computer Press.

Lacko, Ľ. (2011b). *Vytvárajte moderné Web 2.0 aplikácie*. Praha: Microsoft CZ.

Lambson, B., vytotas, Patel, S., Anderson, R., Rojansky, S., Schonning, N., et al. (2020, October 28). Migrations Overview [Online]. Retrieved April 05, 2021, from https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli

Laurenčík, M. (2018). *SQL: podrobný průvodce uživatele*. Praha: Grada Publishing.

Maître, H. (2015). *From Photon to Pixel: The Digital Camera Handbook*. London, England: ISTE Ltd and John Wiley.

McDowell, G. (2009, October 19). How Does A Digital Camera Work? [Technology Explained] [Online]. Retrieved March 13, 2021, from https://www.makeuseof.com/tag/technology-explained-how-does-a-digital-camera-work/

Mell, P., & Grance, T. (2011, September 28). The NIST Definition of Cloud Computing [Online]. Retrieved March 16, 2021, from https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

National Coordination Office for Space-Based Positioning, Navigation, and Timing. (2020, May 29). GPS.gov: Frequently Asked Questions [Online]. Retrieved March 07, 2021, from https://www.gps.gov/support/faq/#sats

Niederst Robbins, J. (2018). *Learning web design: a beginner's guide to HTML, CSS, Javascript, and web graphics* (Fifth edition). Beijing: O'Reilly.

Pecinovský, J. (2017). *Zoner Photo Studio X*. Praha: Grada Publishing.

Perkins, B., Hammer, J. V., & Reid, J. D. (2018). *Beginning C# 7 Programming with Visual Studio 2017*. Indianapolis, IN: Wrox.

Price, M. J. (2019). *C# 8.0 and .NET Core 3.0: modern cross-platform development : build applications with C#, .NET Core, Entity Framework Core, ASP.NET Core, and ML.NET using Visual Studio Code* (Fourth edition). Birmingham: Packt.

Procházka, D. (2009). *Oracle: průvodce správou, využitím a programováním nad databázovým systémem*. Praha: Grada.

Ranjan, A., Sinha, A., & Battewad, R. (2020). *JavaScript for Modern Web Development: Building a Web Application Using HTML, CSS, and JavaScript*. New Delhi: BPB Publications.

Smith, S. (2019, January 15). Simpler ASP.NET MVC Apps with Razor Pages [Online]. Retrieved March 29, 2021, from https://docs.microsoft.com/en-us/archive/msdn-magazine/2017/september/asp-net-core-simpler-asp-net-mvc-apps-with-razor-pages

Smith, S., Wenzel, M. (Ed.). (2020). *Architect Modern Web Applications with ASP.NET Core and Azure* [Online] (5.0 ed.). Redmond, WA: Microsoft Corporation. Retrieved from https://dotnet.microsoft.com/download/e-book/aspnet/pdf

Understand metadata concepts [Online]. (2020, December 06). Retrieved March 06, 2021, from https://experienceleague.adobe.com/docs/experience-manager-65/assets/administer/metadata-concepts.html

Understand when and why to use Razor Pages [Online]. (n.d.). Retrieved March 14, 2021, from https://docs.microsoft.com/en-us/learn/modules/create-razor-pages-aspnet-core/2-why-when-use-razor-pages

Veith, M. (2011, May 8). Budoucnost asynchronního programování v C# [Online]. Retrieved March 14, 2021, from http://blog.netcorex.cz/c-sharp/budoucnost-asynchronniho-programovani-v-c/

Virius, M. (2021). *Programování v C#: od základů k profesionálnímu použití*. Praha: Grada Publishing.

Visual Studio 2019 [Online]. (2020, November 23). Retrieved March 12, 2021, from https://visualstudio.microsoft.com/vs/

Vora, P. (2009). *Web Application Design Patterns*. Burlington, MA: Morgan Kaufmann.

What is .NET? [Online]. (n.d.). Retrieved March 14, 2021, from https://docs.microsoft.com/en-us/learn/modules/dotnet-introduction/2-what-is-dotnet

Woodford, C. (2020, November 1). Digital cameras [Online]. Retrieved March 13, 2021, from https://www.explainthatstuff.com/digitalcameras.html

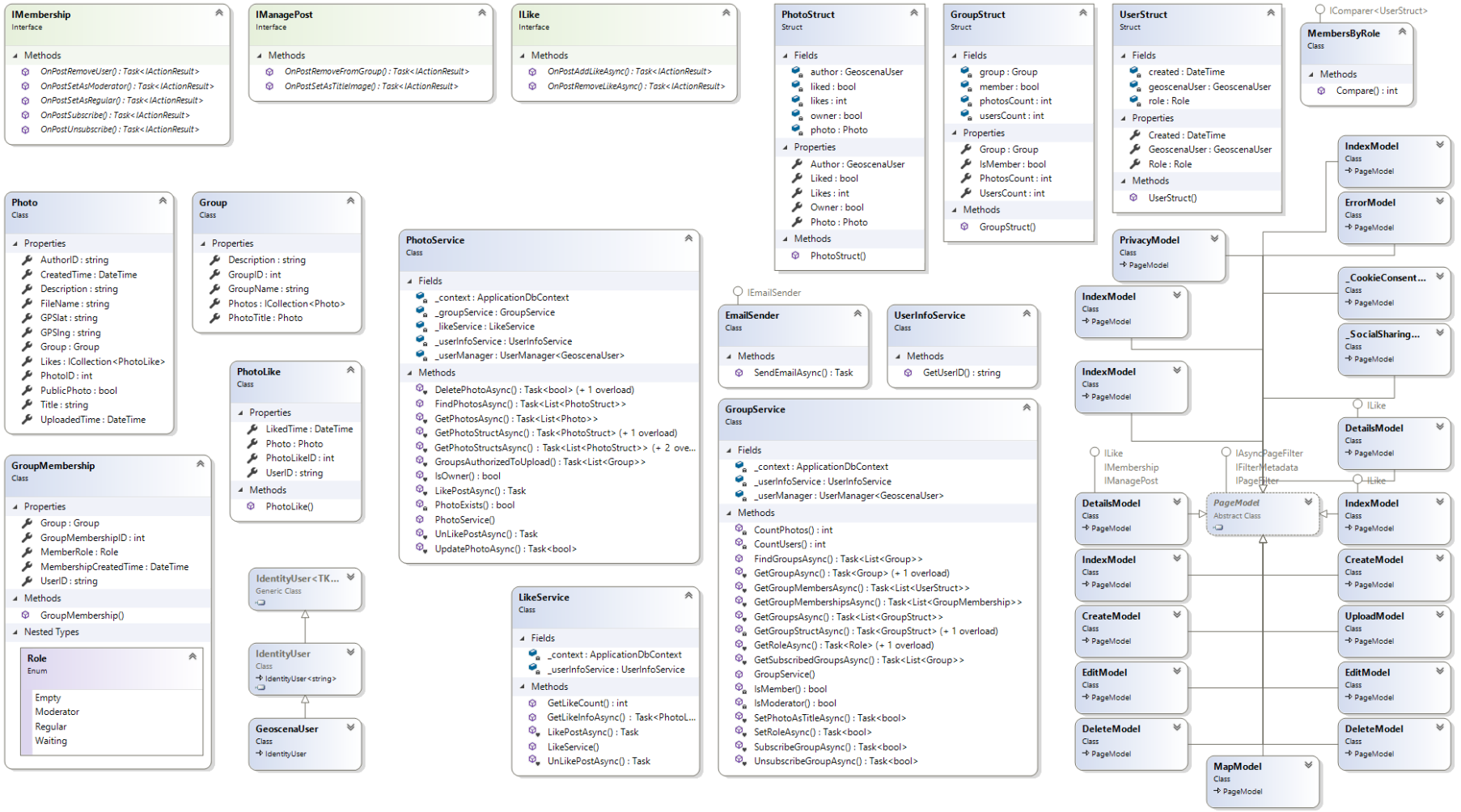
III. Seznam obrázků

IV. Seznam ukázek kódů

V. Seznam příloh

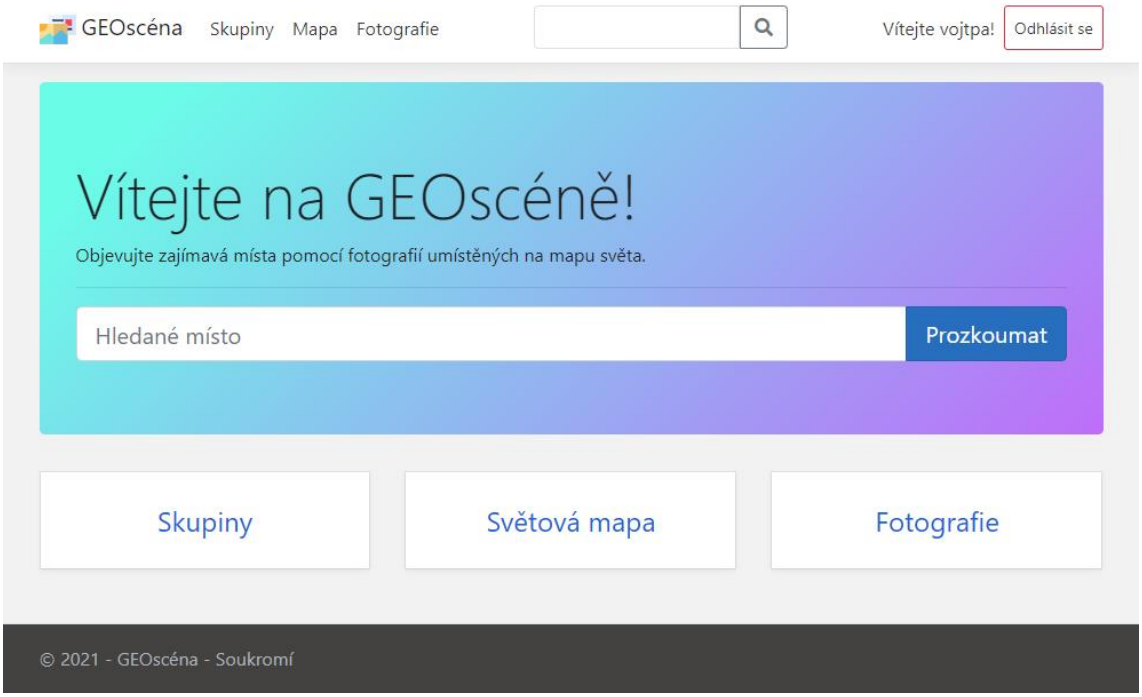
VI. Přílohy

Příloha A: Diagram tříd



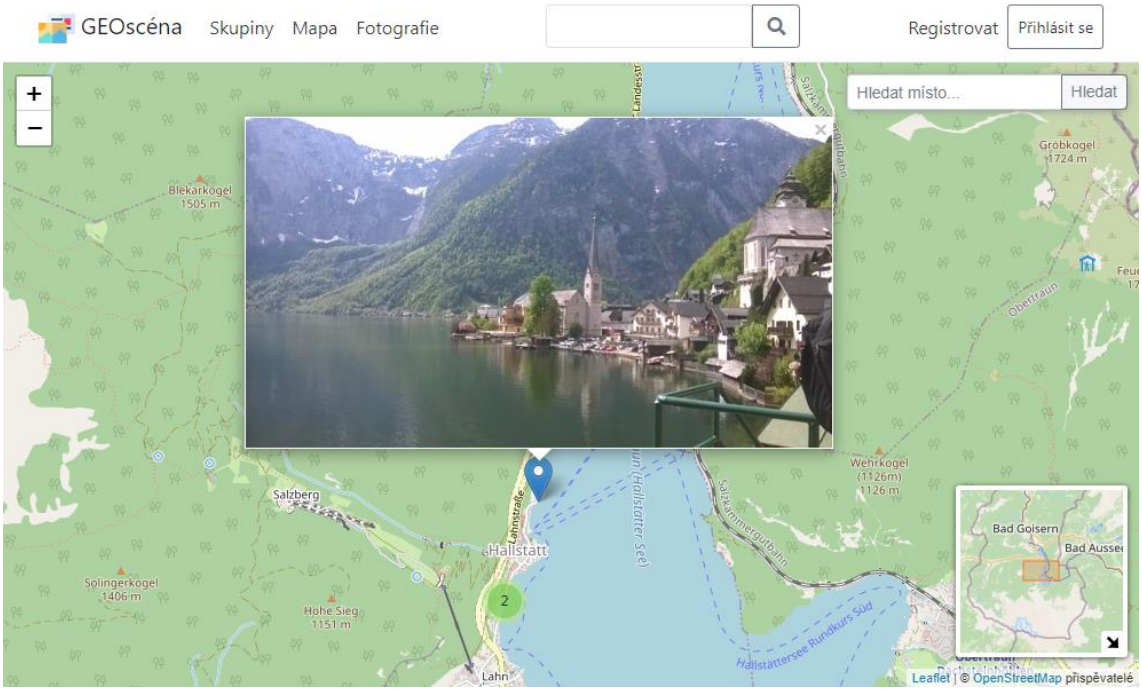
Zdroj: Autor

Příloha B: Snímek Úvodní stránky



Zdroj: Autor

Příloha C: Snímek stránky Mapa

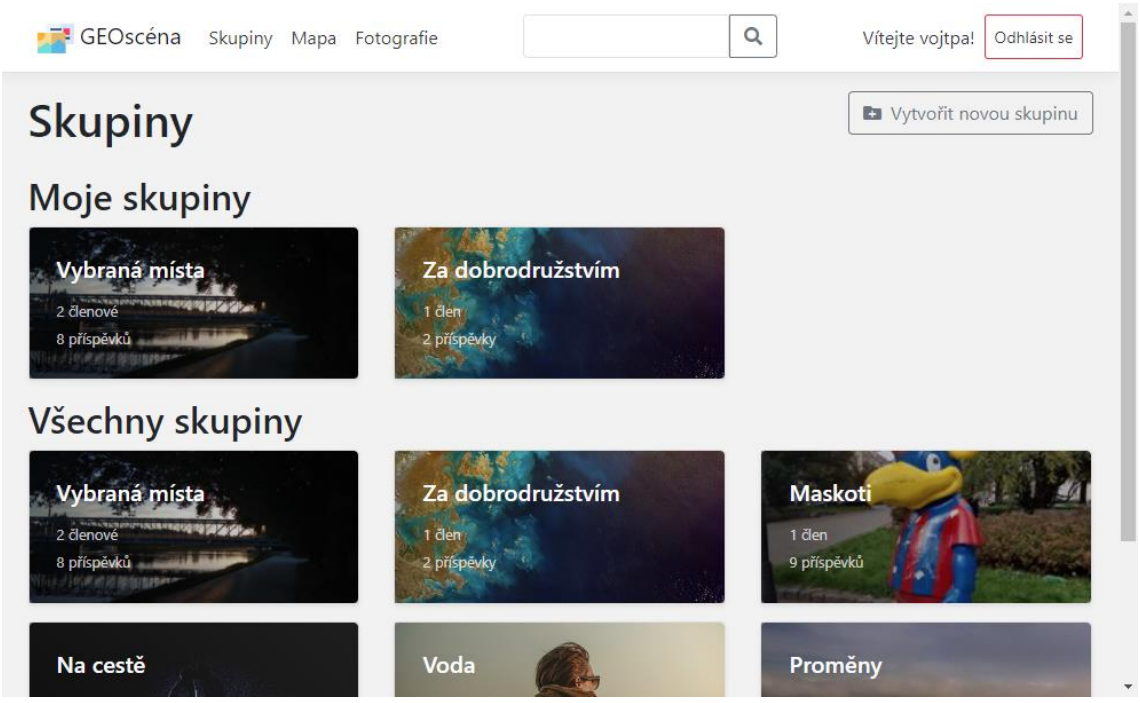


Zdroj: Autor

Příloha D: Snímek stránky pro přehled fotografií

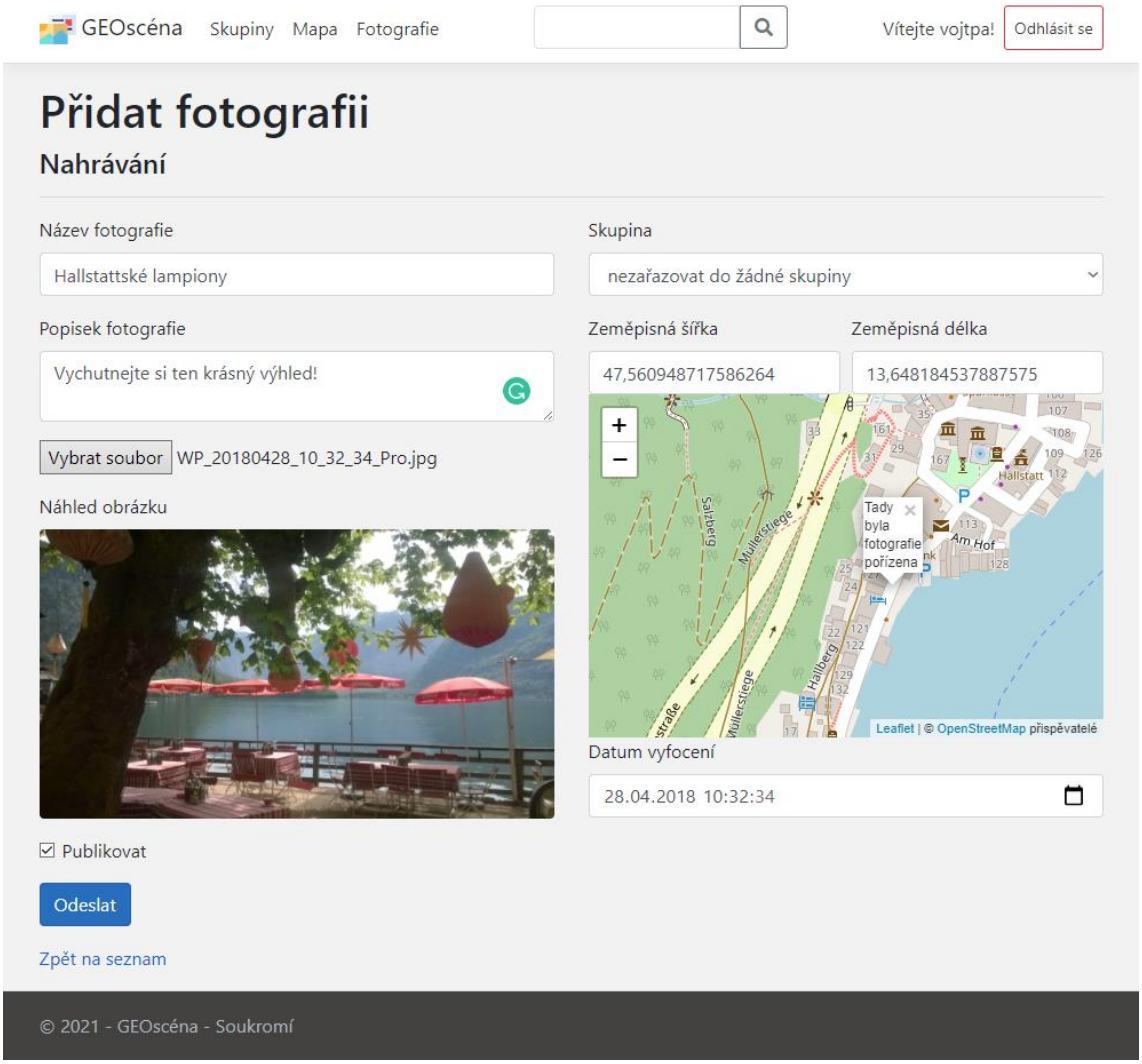
Zdroj: Autor

Příloha E: Snímek stránky pro přehled skupin



Zdroj: Autor

Příloha F: Snímek stránky pro nahrání fotografie



Zdroj: Autor