Univerzita Palackého v Olomouci
Přírodovědecká fakulta
Katedra geoinformatiky

SOFTWAROVÝ PROCES VÝVOJE WEBOVÉ MAPOVÉ PLATFORMY – PŘÍPADOVÁ STUDIE PRO GEOPROCESSINGOVÉ APLIKACE

Diplomová práce

Bc. Daniel URBAN

Vedoucí práce RNDr. Rostislav NÉTEK, Ph.D.

Olomouc 2023

Geoinformatika a kartografie

ANOTACE

Diplomová práce se zabývá softwarovým procesem vývoje webové mapové platformy. Softwarové inženýrství je klíčovou oblastní dnešní digitální společnosti a její význam spočívá v zajištění systematického a disciplinovaného přístupu k vývoji softwarových aplikací. V teoretické části autor nejdříve vymezil klíčové pojmy softwarového inženýrství a webové kartografie. Dále se zabýval historií a jednotlivými modely a metodikami. Konkrétně se zaměřil na metodiku Rational Unified Process, ke které zpracoval jednotlivé kroky softwarového vývoje. Důležitou součástí teoretické části bylo také zpracování jednotlivých aspektů vývoje webové mapové platformy. Mezi nimi byly popsány kartografické, technické, datové, legislativní, komerční, softwarové a uživatelské aspekty. V praktické části autor uplatnil metody softwarového inženýrství, konkrétně Rational Unified Process, při vývoji webové mapové platformy. Autor ověřil a ilustroval vývoj aplikace v jednotlivých krocích RUP jako jsou specifikace požadavků, analýza, návrh, implementace, testování a nasazení. Výsledkem je ilustrovaný postup vývoje webové mapové platformy dle metodik softwarového inženýrství, která v sobě implementuje geoprocessingové nástroje, prostorové operace a vizualizaci v prostředí webové kartografie. Všechny výsledky diplomové práce, včetně funkční webové mapové platformy, jsou dostupné na webových stránkách, které jsou umístěny na webu katedry.

KLÍČOVÁ SLOVA

Softwarové inženýrství; softwarový vývoj; webová mapová platforma; webová mapa; geoprocessingová aplikace

Počet stran práce: 56

Počet příloh: 3 (z toho 1 volná a 2 elektronické)

ANOTATION

The master thesis deals with the software development process of a web map platform. Software engineering is a key area of today's digital society and its importance lies in ensuring a systematic and disciplined approach to the development of software applications. In the theoretical part, the author first defined the key concepts of software engineering and web cartography. They also dealt with history and individual models and methodologies. Specifically, these are the steps of the Rational Unified Process methodology, to which he processed individual software development. An important part of the theoretical part was also the processing of individual aspects of the development of web map platforms. Among them, cartographic, technical, data, legislative, commercial, software and user aspects were described. In the practical part, the author applied software engineering methods, specifically Rational Unified Process, in the development of web map platforms. The author verified and illustrated the development of the application in individual RUP steps such as requirements specification, analysis, design, implementation, testing and deployment. is an illustrated procedure for the development of a web map platform according to the software engineering methodology, which implements geoprocessing tools, spatial operations and visualizations in the web cartography environment. the results of the thesis, including functional web map platforms, are available on the web pages that are located on the department's website.

KEYWORDS

Software engineering; software development; web mapping platform; web map; geoprocessing application

Number of pages 56
Number of appendixes 3

Prohlašuji, že

- diplomovou práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu.

- jsem si vědom, že na moji diplomovou práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,

- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užívat (§ 35 odst. 3),

- souhlasím, že údaje o mé diplomové práci budou zveřejněny ve Studijním informačním systému UP,

- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé diplomové práce v rozsahu § 12 odst. 4 autorského zákona,

- použít výsledky a výstupy mé diplomové práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

Děkuji vedoucímu práce RNDr. Rostislavu NÉTKOVI, Ph.D za vstřícnost, trpělivost, přínosné podněty a připomínky při vypracování diplomové práce. Dále bych rád poděkoval své rodině a přítelkyni za trpělivost a shovívavost.

UNIVERZITA PALACKÉHO V OLOMOUCI

Přírodovědecká fakulta

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Daniel URBAN**
Osobní číslo:	**R210533**
Studijní program:	**N0532A330009 Geoinformatika a kartografie**
Téma práce:	**Softwarový proces vývoje webové mapové platformy – případová studie pro geoprocessingové aplikace**
Zadávající katedra:	**Katedra geoinformatiky**

Zásady pro vypracování

Cílem diplomové práce je navrhnout, specifikovat a ověřit postup vývoje online mapové platformy dle náležitostí softwarového inženýrství. Student uplatní metody a nástroje softwarového procesu, aplikuje je do oblasti online mapových aplikací. Zpracuje jednotlivé kroky softwarového vývoje dle závazných metodik (např. Rational Unified Process), navíc zapracuje legislativní, uživatelské, komerční/business či manažerské aspekty z praxe. V praktické části student navržený postup ověří a ilustruje na případové studii, zahrnující vývoj online mapové aplikace se zaměřením na implementaci geoprocessingových nástrojů, prostorových operací a vizualizace v prostředí webové kartografie.
Celá práce, tj. text včetně všech příloh, posteru, výstupů, zdrojových i vytvořených dat, map, programových kódů a databází, student odevzdá v digitální podobě na paměťovém nosiči (CD, DVD, SD karta, flash disk) s popisem (jméno, název, KGI, rok). Text práce s přílohami odevzdá ve dvou svázaných výtiscích na sekretariát katedry ve stanoveném termínu. O práci student vytvoří webovou stránku v souladu s pravidly dostupnými na stránkách katedry. Práce bude zpracována podle obecných zásad (Voženílek, 2002) a závazné šablony pro kvalifikační práce na KGI. Povinnou přílohou práce bude poster formátu A2.

Rozsah pracovní zprávy:	**max 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

Booch, G., Jacobson, I., Rumbaugh, J. (1999). The Unified Modeling Language User Guide. Addison Wesley Longman, Inc. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G. (1994). Object Oriented Software Engineering, A Use Case Driven Approach. Addison-Wesley.
Nétek, R. (2020). Webová kartografie – specifika tvorby interaktivních map na webu. Univerzita Palackého v Olomouci. 196 s. ISBN 978-80-244-5827-4.
Rumbaugh, James et al. (1991). Object-Oriented Modeling and Design. Prentice Hall Inc.
Schmuller, J. (1999). Teaching Yourself UML in 24 Hours. Sams.
Vondrák,I. (2002). Úvod do softwarového inženýrství. VŠB – TU Ostrava.
Voženílek, Vít. Diplomové práce z geoinformatiky. Univerzita Palackého, 2002.

Vedoucí diplomové práce: **RNDr. Rostislav Nétek, Ph.D.**
Katedra geoinformatiky

Datum zadání diplomové práce: **9. prosince 2021**
Termín odevzdání diplomové práce: **5. května 2023**

L.S.

doc. RNDr. Martin Kubala, Ph.D.
děkan

prof. RNDr. Vít Voženílek, CSc.
vedoucí katedry

V Olomouci dne 16. prosince 2021

OBSAH

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE
PŘÍLOHY

SEZNAM POUŽITÝCH ZKRATEK

Zkratka	Význam
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BSD	Berkeley Software Distribution
CASE	Computer Aided Software Engineering
CC	Creative Commons
CSS	Cascading Style Sheets
DBMS	Database Management Systems
EPSG	European Petroleum Survey Group
(Geo)JSON	(Geographic) JavaScript Object Notation
GIS	Geografický informační systém
HTML	Hypertext Markup Language
HTTP(S)	Hypertext Transfer Protocol (Secure)
ID	identifikátor
JS	Javascript
MIT	Massachusetts Institute of Technology
MVC	Model-view-controller
OEM	Original Equipment Manufacturer
PC	Personal Computer (osobní počítač)
PDO	PHP Data Objects
PHP	Hypertext Preprocessor
RUP	Rational Unified Process
SHP	Shapefile
SQL	Sctructured Query Language
SWOT	Strengths, Weaknesses, Opportunities, and Threats
UI	User Interface
UML	Unified Modeling Language
UP	Univerzita Palackého
URL	Uniform Resource Locator
UX	User Experience
WGS84	World Geodetic Systém 1984
WMS	Web Map Service
WMTS	Web Map Tile Service

ÚVOD

Softwarové inženýrství je oblastí, která hraje klíčovou roli v dnešní digitální společnosti. S rozvojem technologií a nárůstem počítačových systémů se stalo nezbytné zajistit, aby softwarové aplikace byly efektivní, spolehlivé a bezpečné. Softwarové inženýrství se proto zaměřuje na procesy, postupy a metody, které umožňují vývoj kvalitního softwaru.

Důležitost softwarového inženýrství spočívá v tom, že zajišťuje systematický a disciplinovaný přístup k vývoji softwarových aplikací. Škála softwarových systémů se rozšiřuje od jednoduchých aplikací po složité podnikové systémy. Bez sofistikovaných softwarových inženýrů by bylo nemožné vytvořit takové aplikace, které nám usnadňují každodenní život a podporují rozvoj podniků a organizací.

Jedním z konkrétních příkladů významu softwarového inženýrství jsou webové mapové aplikace. Tyto aplikace nám umožňují prozkoumat svět, najít cesty a orientovat se v neznámém prostředí. Webové mapové aplikace jako Google Maps, Mapy.cz nebo OpenStreetMap se staly neodmyslitelnou součástí našeho každodenního života. Pomáhají nám plánovat trasy, vyhledávat nejbližší místa, získávat aktuální dopravní informace a objevovat nová místa.

Vývoj webových mapových aplikací vyžaduje komplexní využití metod softwarového inženýrství. To zahrnuje návrh uživatelského rozhraní, zpracování geografických dat a jejich uložení, implementaci vyhledávacích algoritmů, optimalizaci výkonu a zajištění bezpečnosti a integrity dat. Kvalitní webové mapové aplikace musí být rychlé, přesné a spolehlivé, aby uspokojily potřeby uživatelů.

Webové mapové aplikace mají široké uplatnění v mnoha oblastech, jako je doprava, turismus, logistika, geografické informační systémy (například ArcGIS Online, CARTO, CloudGIS) a mnoho dalších. Mohou nám být nápomocní nejen při plánování cesty, ale mohou také analyzovat geografická data, vizualizovat prostorové informace a být oporou v rozhodovacích procesech.

Odborných prací na téma softwarového inženýrství je nepřeberné množství, nicméně tématu geoinformačních technologií, resp. webových map se již nevěnují. I proto se tato diplomová práce kromě zpracování běžného softwarového vývoje zaměřuje na specifika a aspekty vývoje webové mapové platformy. Jednotlivé kroky jsou dále aplikovány na nejdůležitějším výstupu práce – webové mapové platformě, která se zaměřuje na implementaci geoprocessingových nástrojů, prostorových operací a vizualizaci v prostředí webové kartografii.

1 CÍLE PRÁCE

Cílem diplomové práce je analyzovat, navrhnout a ověřit postup vývoje webové mapové platformy s důrazem na principy softwarového inženýrství. Student uplatní a aplikuje metody a nástroje softwarového procesu, které jsou specifické pro oblast online mapových aplikací. Dále se zaměří na zpracování jednotlivých kroků softwarového vývoje (např. Rational Unified Process). Součástí teoretické části je také zahrnutí legislativních, uživatelských, komerčních/business či manažerských aspektů.

V praktické části je provedeno ověření a ilustrace metod softwarového inženýrství na konkrétní případové studii. Tato studie se zaměřuje na vývoj webové mapové platformy, která klade důraz na implementaci geoprocessingových nástrojů, prostorových operací a vizualizaci v prostředí webové kartografii.

Součástí diplomové práce je tvorba webových stránek a posteru ve formátu A2. Konečná verze diplomové práce, včetně všech příloh, je volně dostupná na webových stránkách Katedry geoinformatiky UP.

2 VYMEZENÍ POJMŮ

V této kapitole diplomové práce je nutné vymezit pojmy týkající se softwarového inženýrství a webové kartografie. Jejich vymezení je potřebné pro korektní pochopení obsahu práce.

2.1 Softwarové inženýrství

Základním pojmem celé diplomové práce je softwarové inženýrství. Podle Vondráka (2002) se jedná o inženýrskou disciplínu, která se zabývá praktickými problémy vývoje rozsáhlých softwarových systémů. Z dané definice pak vyplývá, že vývoj softwarového systému zahrnuje řadu faktorů nutných k úspěšnému vytvoření požadovaného produktu. Mezi tyto faktory patří:

- technické aspekty zahrnující počítačovou infrastrukturu a softwarové vybavení
- netechnické aspekty dané organizační strukturou organizace, která vyvíjí daný produkt a jejími možnosti v oblasti ekonomiky
- znalostmi z oblasti specifikace požadavků na softwarový produkt jeho analýzy
- lidské zdroje schopné aplikovat výše uvedené znalosti a uplatnit je tak při realizaci požadovaného softwarového systému
- řízení spjaté s vývojem samotného produktu umožňující efektivní využití všech výše uvedených faktorů s cílem vytvořit produkt požadované kvality

2.2 Softwarový proces

Pod pojmem softwarový proces rozumíme po částech uspořádanou množinu kroků, která směřuje k vytvoření nebo úpravě softwarového díla. Během historie vývoje softwarových systému se objevila řada modelů, které popisovaly, jak by měl softwarový proces vypadat. Dodnes ale neexistuje detailně a přesně definovaná podoba softwarového procesu, která by byla závazná a přijata jako referenční. Nicméně i tak lze konstatovat, že základem všech pozdějších modelů se stal tzv. vodopádový model (Vondrák, 2002). Detailněji jsou modely a jejich historie popsány v kapitole 3.

2.3 Pojmy webové kartografie

Následující pojmy se často zaměňují a proto autor diplomové práce považuje za nezbytné je v tomto kroku vymezit.

Roth a kol. (2017) uvádí, že **webová mapa** je mapa zveřejněna na internetu, která je většinou součástí webové stránky. Webové mapy se podle míry interaktivity dále dělí na statické a dynamické.

Podlé Nétka (2008) představuje **mapový portál** server, na kterém jsou pod jednotlivými odkazy umístěny elektronické mapy. Jedná se tedy o internetový portál, který je věnován mapám. Z technického hlediska je nejobvyklejší tzv. třívrstvá architektura, tzn., že celá aplikace se obvykle skládá ze tří částí: mapového serveru, webového serveru a rozhraní pro správu dat.

Pod pojmem **mapová aplikace** rozumíme soubor nástrojů a funkcí, které umožňují prohlížení mapových úloh. Mapová úloha je soubor vektorových či rastrových vrstev, věnovaných určitému tématu. Jedná se tedy o položku, kterou uživatel vybírá v menu mapového portálu (Nétek, 2008).

Webovou kartografii popisuje Kraak a Brown (2001) jako obor zabývající se tvorbou, zobrazováním a údržbou map v prostředí webu. Produktem webové kartografie je webová mapa.

Webová aplikace je počítačový program, který využívá webové prohlížeče a technologie k plnění úloh přes rozhraní Internetu (Gibb, 2016).

3 SOUČASNÝ STAV ŘEŠENÉ PROBLEMATIKY

Tato kapitola se zabývá problematikou softwarového inženýrství. Na úvod je připomenuta jeho historie, dále jsou popsány metodiky a modely životního cyklu, pozornost je věnována metodice Rational Unified Process. Závěr kapitoly je věnován specifikům a aspektům vývoje webové mapové platformy.

3.1 Historie softwarového inženýrství

Během počátků éry počítačů v 50. a 60. letech minulého století byl hlavním nákladem pořízení hardwaru. Sálové počítače vážící desítky tun s velkým příkonem měly omezený výpočetní výkon a nízkou spolehlivost. Programy se vyvíjely ve dávkovém režimu, kde programátoři vyděrovali program na děrné štítky a poté čekali na výsledek. S rozvojem počítačů se hardware stal menším, levnějším, rychlejším a dostupnějším, zatímco programy se zvětšovaly a stávaly se složitějšími

Softwarová krize v 60. letech přinesla problémy s prodlužováním a prodražováním projektů, nízkou kvalitou výsledků a nízkou produktivitou programátorů. Bylo zavedeno strukturované programování jako snaha o řešení těchto problémů. V 70. letech se rozvíjely metody softwarového inženýrství, které zahrnovaly lidský faktor, životní cyklus softwaru, testování a podporu pro řízení tvorby softwaru. Byly objeveny abstraktní datové typy a probíhal rozvoj metodologií.

Přelomová se stala 80. léta, kdy se začaly používat vývojová programovací prostředí a nástroje CASE. Vývoj programovacích jazyků zahrnoval funkcionální, logické a imperativní programování a objevila paradigma objektově orientovaného programování. Metody softwarového inženýrství se zaměřily na prototypování, znuvupoužitelnost a komponenty. Pomocí metrik se pak sledovala kvalita softwarového procesu a výsledného produktu.

V současnosti je důraz kladen na podporu zákazníka a dále na servis a údržbu softwaru. Klasické metodologie se staly byrokratickými a nákladnými, proto se častěji používají tzv. lehké metodologie s flexibilnějším životním cyklem. Formální metody pro simulaci, analýzu a verifikaci systémů se rozvíjejí s cílem zvýšit bezpečnost softwarových systémů v kritických oblastech (Křena a Kočí, 2010).

3.2 Metodiky a modely životního cyklu

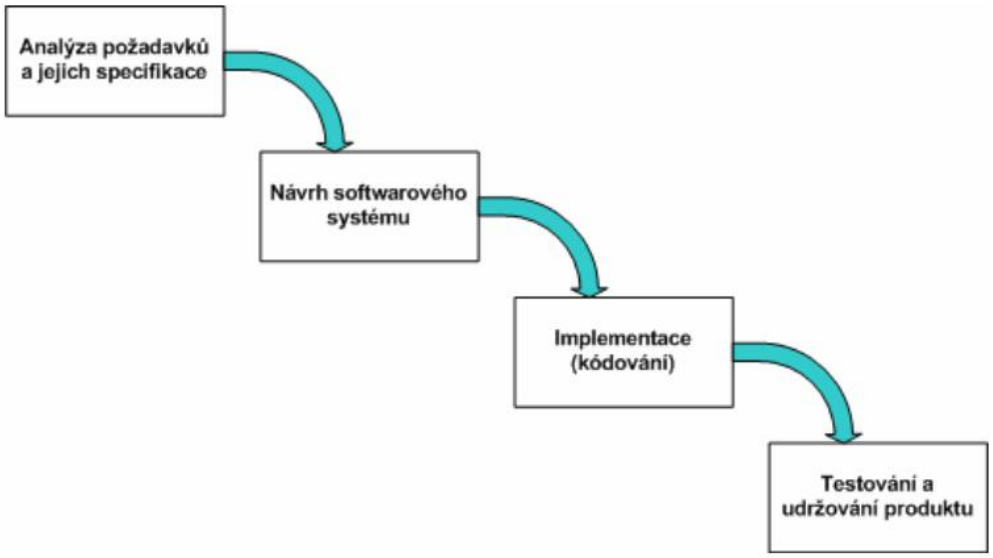
K přístupu vývoje softwaru můžeme nalézt velké množství přístupů. Jejich podstata je vyjádřena v modelech životního cyklu softwaru. Rozdíly mezi jednotlivými modely spočívají zejména v posloupnosti jednotlivých vývojových etap. Každý model životního cyklu softwaru definuje etapy vývoje a pro každou etapu dále specifikuje nutné činnosti se vstupy a výstupy (Křena a Kočí, 2010). Jednotlivé etapy se mezi modely prolínají. Detailněji jsou etapy a fáze popsány v kapitole 3.3, která je věnována modelu Rational Modified Process, nicméně souhrnně by se daly shrnout takto:

- analýza a specifikace požadavků
- architektonický a podrobný návrh
- implementace
- integrace, testování
- provoz a údržba

3.2.1 Vodopádový model

Prvním zavedeným přístupem, který byl vytvořen pro potřeby vývoje softwaru, byl vodopádový model. Model byl definován v roce 1970 Winstonem W. Roycem a získal okamžitou podporu manažerů. Vodopádový model obsahuje etapy, které jsou shrnuty v kapitole 3.2. a předpokládá, že všechny požadavky lze shromáždit již během etapy analýzy a specifikace požadavků. To je však zároveň i největším problémem modelu – uživatel není schopen předem stanovit všechny požadavky. Jednotlivé etapy jsou seřazeny za sebou a následující etapy začíná až v momentě, kdy je předchozí etapa plně ukončena. Při objevení chyby či doplnění nových požadavků ze strany uživatele/zákazníka v pozdních fázích vývoje model selhává a vrací se v podstatě na začátek (Sommerville, 2011).

Ačkoliv se zdá, že model má velké nedostatky, jeho důležitost tkví zejména v tom, že je jakýmsi odrazovým můstkem pro další modely, které z něj vznikly (Hughey, 2009).



Obr. 1 Schéma vodopádového modelu (Vondrák, 2002)

3.2.2 Iterativní a inkrementální model

Na předchozí model se snaží navázat model iterativní, který se snaží odstranit hlavní problém svého předchůdce. Jak napovídá název, rozděluje proces vývoje softwaru do iterací, kterou chápeme jako instanci vodopádového modelu. Funkční software se vytváří brzy v životním cyklu a po každé iteraci tak uživatel vidí funkční verzi softwaru, kde sice nemusí být implementována plná funkcionalita, ale umožní zákazníkovi upřesnit své požadavky. Ty pak mohou být zpracovány v další iteraci (Sushant, 2022).

Podobným modelem je i inkrementální model. Na základě specifikace systému se stanoví ucelené části systému. Ty se vytvářejí a odevzdávají uživateli postupně, což znamená, že struktura softwaru může být navržena lépe než za použití předchozího modelu (Křena a Kočí, 2010).

3.2.3 Spirálový model

Tento model je jedním z nejdůležitějších modelů životního cyklu vývoje softwaru. Je založen na myšlence spirály, přičemž každá iterace spirály představuje kompletní cyklus vývoje softwaru. Model zavádí prototypování a má při analýze za cíl eliminovat možná rizika.

Rizika mohou představovat například technické, časové či rozpočtové faktory. Spirálový model se hodí zejména na velké a složité projekty, kde je důležitá flexibilita modelu a možnost efektivně řídit rizika (Software Engineering | Spiral Model, 2023).

3.2.4 Agilní metody

Zmíněné modely životního cyklu softwaru se vyvíjely dlouho a postupem času se z nich staly kolosy, které se hodí zejména na velké a složité projekty. Při vytváření menších softwarových systémů jsou však tyto modely zbytečně složité, např. z důvodu dodržování byrokratických pravidel, což následně prodražuje vývoj. Jako protiváha ke zmíněným modelům slouží agilní metody (Křena a Kočí, 2010). Jejich základy byly položeny v roce 2001, kdy skupina odborníků sepsala tzv. "Manifest Agilního vývoje software". Tento dokument definoval 12 principů, v nichž například zdůrazňoval hodnoty jako spolupráce s klienty, přizpůsobivost změnám, jednoduchost a dodávka funkčního produktu. Tím byly vymezeny základní principy, které dnes charakterizují agilní metody (Manifest Agilního vývoje software, 2001). Mezi nejznámější příklady agilních metod patří Extreme Programming (XP), Test-Driven Development (TDD), Feature Driven Development (FDD) nebo Scrum (Al-Saqqa a kol., 2020).

3.3 Rational Unified Process (RUP)

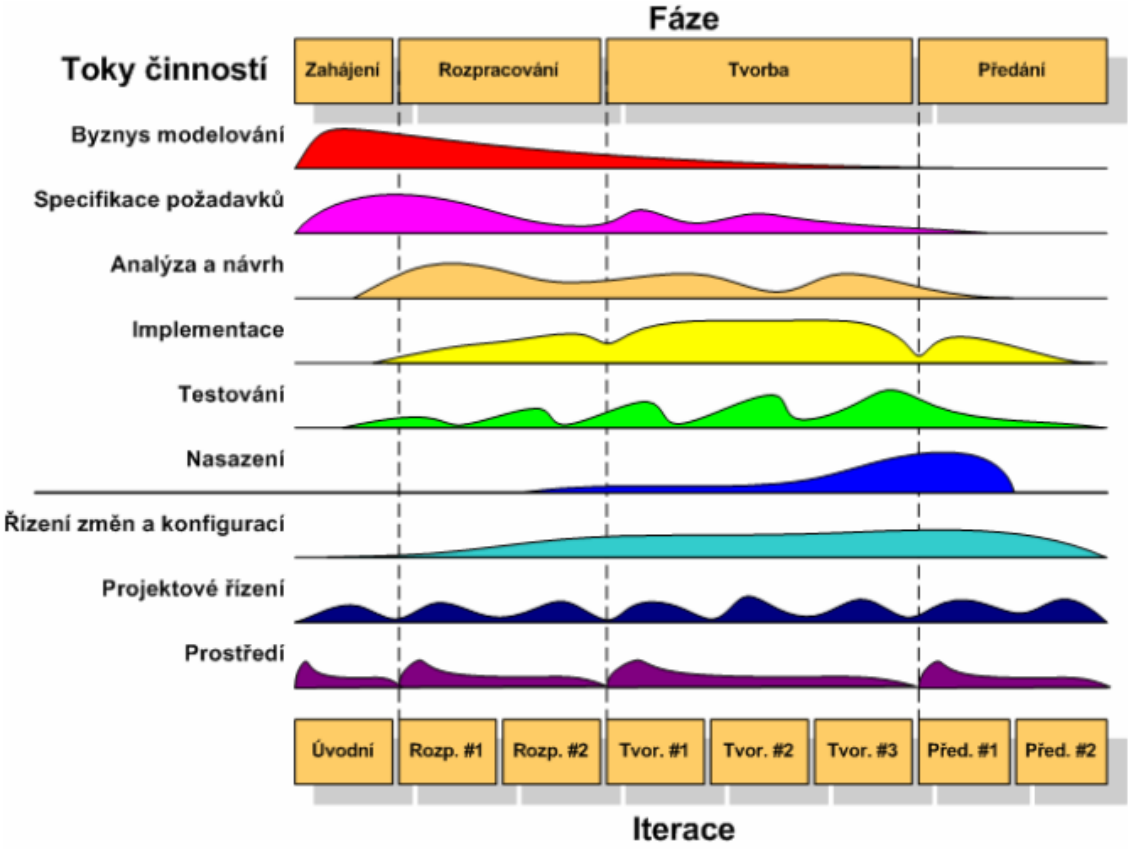
Model RUP byl vytvořen společností Rational Software Corporation, kterou v roce 2003 zakoupil softwarový gigant IBM. Tato metodika nebyla vytvořena jako soubor pravidel, jež se mají za každou cenu striktně dodržovat, ale jako soubor doporučení, které si vývojové týmy mohou vybrat podle jejich potřeb. Díky rozsáhlému plánování a analýzám se RUP hodí spíše k větším projektům (Grossman – Rational Unified Process). RUP klade důraz na vizualizaci softwarového systému pomocí jazyka UML – Unified Modelling Language (Křena a Kočí, 2010). Metodika je velmi komplexní, systematická a uznává těchto 10 principů (Grossman, Rational Unified Process):

- vytvoření jasné vize
- řízení podle daného plánu
- nalezení, zmírnění a odstranění rizik
- pojmenování problému a jejich sledování jejich stavu
- porozumění obchodnímu kontextu
- používání komponentové architektury
- inkrementální vývoj a časté testování
- pravidelné vyhodnocování dosavadních výsledků
- řízení a sledování požadavků na změnu
- poskytování kvalitní uživatelské podpory uživatelům

RUP se řadí k iterativním způsobům vývoje softwaru a klade důraz na vývoj komponent. Každá iterace končí vytvořením takové verze systému, která je spustitelná a je možné ji předvést zadavateli. Iterace se skládá ze 4 fází – zahájení, rozpracování, tvorba a předání. Metodika také definuje kdo, co, jak a kdy dělá – základní stavební kameny. Jsou jimi pracovníci, činnosti, artefakty (produkty) a pracovní procesy (Presmann, 2010).

Podle Vondráka (2002) se skládá metodika z toků činností a aktivit. Základními toky činností jsou byznys modelování, specifikace požadavků, analýza a návrh, implementace,

testování a nasazení. Podpůrné toky jsou řízení změn a konfigurace, projektové řízení a prostředí. Na rozdíl od vodopádového modelu probíhají toky činností souběžně, i když z obr. 2 vyplývá, že objem prací a aktivit v daném toku závisí na rozpracovanosti softwarového systému. Je například zřejmé, že specifikace požadavků nebo analýza a návrh logicky probíhá před nasazením softwaru.



Obr. 2 Schéma procesu RUP (Vondrák, 2002)

3.3.1 Byznys modelování

Smyslem byznys modelování je vytvořit abstrakci procesu, aby umožňovala pochopení všech aktivit a jejich souvislostí. Zároveň poskytuje společný jazyk mezi všemi lidmi, kteří se na projektu podílejí (Vondrák, 2004).

Tímto společným jazykem je UML (Unified Modelling Language), který vznikl v roce 1995. Samotný jazyk však nenabízí metodologii k vytvoření či návrhu systému, ale pouze prostředky pro společnou tvorbu modelů. UML operuje se základním pojmem pohled (view), pod čímž rozumíme projekci systému pod určitým pohledem. Taková projekce se zaměřuje jen na příslušný aspekt a ostatní ignoruje (Booch a kol., 1999). Je vhodné vytvořit více pohledů, mezi základní patří (Křena a Kočí, 2010):

- **Strukturální pohled** – popisuje objekty, třídy a jejich asociace a komunikaci
- **Pohled chování** – popisuje interakci objektů a jejich reakce na vnější systémové operace
- **Datový pohled** – popisuje objekty a jejich vazby
- **Pohled rozhraní** – zaměřeno na zapouzdření systémových částí a jejich použití okolním systémem

Podle Vondráka (2004) nabízí UML několik základních diagramů a dělí se na 2 skupiny:

- **Diagram aktivit** – popisuje proces pomocí jeho stavů reprezentovaných vykonáváním aktivit a pomocí přechodů mezi stavy způsobených ukončením těchto aktivit. Účelem aktivit je popsat tok činností.
- **Diagram tříd** – graf tvořený objekty, které jsou vzájemně propojeny vazbami. Účelem je zachytit statický aspekt modelované domény.

3.3.2 Specifikace požadavků

Cílem specifikace požadavků je popsání základní charakteristiky softwarového systému a jeho funkcionality a stanovení služeb, které zákazník požaduje a vymezení podmínek jeho vývoje a následného provozu. Modely specifikace požadavků slouží zejména k odsouhlasení zadání mezi vývojovým týmem a zákazníkem.

Požadavky se podle Křeny a Kočího (2010) dělí na základní kategorie:

- **Funkcionální požadavky** – jedná se o základní typ požadavku, kde zákazník specifikuje funkcionalitu systému (např. možnost nahrát vlastní data do systému).
- **Požadavky na provoz systému** – definují požadavky za kterých má aplikace fungovat, např. počet uživatelů.
- **Požadavky na výsledný systém** – definují zejména požadavky související s vývojem a nasazením, např. počítačové a softwarové vybavení.
- **Požadavky na vývojový proces** – požadavky na dodržování norem během vývoje
- **Požadavky na rozhraní** – např. uživatelské rozhraní
- **Externí požadavky** – návaznost např. na legislativní požadavky (ochrana osobních údajů apod.)

Důležitým faktorem pro kvalitní software je získání informací o specifikacích požadavků. Nesmírně důležitá je motivace zákazníka, aby s vývojovým týmem spolupracoval a podílel se na specifikaci požadavků. Existuje řada metod, jak tyto informace získat, např. rozhovor, dotazník, pozorování práce u zákazníka, analýza současného softwarového systému. Během tohoto procesu také vyvstávají největší problémy při specifikaci. Jedná se například až o příliš obecné zadání či neschopnost zákazníka přesně formulovat požadavky. Specifikace požadavků se dělí na několik kroků. Jsou jimi definice požadavku, specifikace požadavku a specifikace softwaru (Sommerville, 2011).

Pro zaznamenání požadavku se v RUP používají tzv. případy užití (use cases), které jsou součástí jazyka UML. Jde o nalezení případů užití jejich účastníků a nalezení detailů vybraných případů užití. Můžeme si to představit jako funkci, kterou systém vykonává jménem jednotlivých účastníků nebo v jejich prospěch (Křena a Kočí, 2010).

3.3.3 Analýza a návrh

Během toku činností v etapě analýza a návrh softwarového produktu je cílem ukázat, jakým způsobem bude software realizován v následné implementační etapě. Tato fáze je klíčová a postupem času nabývá na důležitosti. Pomáhá porozumět požadavkům a cílům projektu a slouží k vytvoření základu pro návrh a implementaci softwarového systému.

Jedním z klíčových nástrojů ve fázi analýzy jsou diagramy tříd, které slouží k vizualizaci struktur a vztahů mezi jednotlivými třídami (objekty) v systému. Dále se například

používají diagramy spolupráce, nasazení, stavové diagramy nebo diagramy aktivit pro znázornění interakcí mezi jednotlivými komponenty systému. Tyto diagramy slouží k vytvoření srozumitelného modelu softwarového systému (Vondrák, 2002). K dalším nástrojům patří např. předimplementační analýza, která se zaměřuje na detailní rozbor požadavků, identifikaci možných rizik, příležitostí a plánování strategie pro návrh a implementaci softwaru (Pre-implementation analysis, 2022). Součástí analýzy bývá také analýza konkurence či SWOT analýza, která identifikuje silné a slabé stránky softwaru, příležitosti a hrozby (Sarsby, 2012). Dalším klíčovým prvkem je tvorba person (osob), které představují reprezentativní uživatele nebo zákazníky, kteří budou software používat. Persony pomáhají designérům a vývojářům lépe porozumět, jakým způsobem budou uživatelé produkt používat a jaké funkce jsou pro ně důležité. Kromě toho také vytváří scénáře, které popisují konkrétní situace jakým způsobem bude software používán (Pruitt a Adlin, 2006).

Během návrhové fáze se upřesňují modely a analýzy, které vznikly v předchozí fázi a vytváří se konkrétní specifikace a technický návrh požadovaného softwarového systému. Během této fáze jsou specifikovány technické detaily architektury, systémové a datové struktury, komponenty a moduly. Důležitou součástí návrhu je také specifikace grafického stylu a uživatelského rozhraní. V této fázi jsou například vytvářeny wireframy a prototypy, které umožní ověření funkcionality a designu před následnou implementací. Součástí návrhu mohou být také plány na testování či nasazení systému (Cole a Sampaio, 2004).

3.3.4 Implementace

Cílem činností během implementační fáze je převést návrhy, které vznikly během předchozích fází vývoje softwaru, do spustitelného programu. Podíl implementace vůči dalším fázím se postupně snižuje, což je zapříčiněno několika faktory. Mezi tyto faktory se řadí například zavedení programovacích jazyků s vyšší úrovní abstrakce, vizuální programování, skládání kódu z komponent (knihoven) nebo nástroje pro generování základní kostry aplikace (Pressman, 2010).

Pro tuto fázi je zcela stěžejní výběr programovacího jazyka, přičemž podle Křeny a Kočího (2010) rozhodují následující faktory:

- zkušenost programátora s daným jazykem
- vhodnost jazyka pro danou aplikaci, jeho dostupnost a rozšířenost
- přenositelnost a použitelnost na hardwaru
- požadavky zákazníka

Mezi další faktory by se řadila například cena vývojového prostředí, která však v dnešní době nehraje tak velkou roli. Důležitá je také strategie implementace, u které rozlišujeme dva základní přístupy. První je implementace zdola-nahoru, kde se jako první nejdříve implementují základní části kódu, které se následně spojují do větších celků. Druhou strategií je implementace shora-dolů, ve které se naopak nejdříve vytváří moduly na vyšších úrovních, po kterém následuje dokončení částí na úrovních nižších. V praxi se nejčastěji používá kombinace obou zmíněných strategií (Křena a Kočí, 2010).

3.3.5 Testování

Testování je prováděno z pohledu tří základních dimenzí, které jsou reprezentovány kvalitou, funkcionalitou a výkonností systému. Testy jsou plánovány pro každou iteraci softwarového cyklu a zahrnují integrační a systémové testy. Testy se navrhují a implementují v podobě testovacích úloh, následně se výsledky systematicky zpracovávají a vadné části kódu jsou zaslány zpět do toků činností jako je např. analýza a návrh nebo implementace s cílem nalezené nedostatky eliminovat.

Činnost toků se dělí na verifikaci a validaci. Během verifikace je hledána odpověď otázku, zda-li je software vytvářen správně – jsou hledány nedostatky v samotném systému. Validace je proces, při kterém zjišťujeme, zda-li je implementována požadovaná funkcionalita (Vondrák, 2002).

Testování se dále dá dělit např. na black-box a white-box testování. Dalším dělením je statické a dynamické testování. U statického není vyžadován běh softwaru a výsledkem je např. zpřesnění odhadů časové náročnosti vývoje. Dynamické testování vyžaduje existenci spustitelné verze softwaru a na základně vstupů do systému jsou posuzovány následné výstupy (Vondrák, 2002).

Jako poslední testování je určitě důležité zmínit akceptační testování. Provádí jej uživatel/zadavatel s reálnými daty. Na jeho základě určí, zda software splňuje jeho očekávání a zadání (Křena a Kočí, 2010).

V kontextu psaní diplomové práce na Katedře geoinformatiky by bylo vhodné zmínit i metodu eye-tracking testování, která je používaná v oblasti výzkumu a uživatelského testování a která sleduje a analyzuje pohyb očí při interakci s uživatelským rozhraním (UI). Poskytuje důležité informace o uživatelském chování, porozumění vizuální hierarchii a efektivity uspořádání designových prvků. Pomáhá vylepšit uživatelskou přívětivost, zajišťuje, že důležité informace jsou správně umístěny na obrazovce a umožňuje optimalizovat uspořádání obsahu, ovládacích prvků a navigace. Metoda se často využívá v oblastech jako je webový design, marketing nebo UI. (Popelka, 2018).

3.3.6 Nasazení

Etapa nasazení má za cíl úspěšné dodání výsledného softwaru koncovému uživateli. Nasazení softwarového systému pokrývá řadu činností (Vondrák, 2002):

- vytvoření produktu
- kompletace systému a jeho distribuce
- instalace systému u koncového uživatele
- poskytnutí odpovídající podpory koncovému uživateli
- plánování, případné beta testování
- v případě potřeby migrace již existujících dat z předešlého softwaru

3.3.7 Podpůrné toky činností

Jak již bylo zmíněno v kapitole 3.3, mezi podpůrné toky činností se řadí řízení změn a konfigurace, projektové řízení a prostředí.

Řízení změn a konfigurace představuje správu jednotlivých verzí vytvářených artefaktů odrážejících vývoj změn požadavků zákazníka na software. RUP přistupuje k řízení změn a konfigurace systematicky a zavádí nástroje, postupy a role, které umožňují identifikovat, analyzovat a řešit změny v rámci projektu, a zároveň udržovat sledovatelnost

a integritu konfigurace softwaru. Díky tomu je možné efektivně spravovat změny a minimalizovat rizika spojená s nekontrolovanými úpravami. K řízení změn a konfigurace se používá například Git. Jedná se o verzovací systém, který slouží k správě a sledování změn v kódu a projektových souborech. Umožňuje vývojářům spolupracovat na jednom projektu, uchovávat historii změn nebo vytvářet větve pro nezávislý vývoj různých komponent. Git zachovává komplexní historii všech provedených úprav, což usnadňuje práci v týmu, reverzi změn a zajišťuje konzistenci projektu. Díky svému distribuovanému charakteru je možné pracovat offline a synchronizovat změny později. Git je klíčovým nástrojem pro efektivní softwarový vývoj a řízení verzí. Zároveň umožňuje efektivní správu požadavků na změny, které vznese například zadavatel (Chacon a Straub, 2014)

Projektové řízení zahrnuje problematiku koordinace pracovníků, sestavení týmu, zajištění a dodržení rozpočtu, plánování a průběžné kontroly výsledků. Projektem rozumíme časově ohraničené úsilí, které se vyvíjí s cílem vytvořit jedinečný výsledek (např. softwarový produkt). Základními parametry každého projektu jsou cena, čas, rozsah a kvalita. Tyto parametry jsou vzájemně svázány. Důležitým faktorem každého projektu jsou lidé, proto je vhodné si na výběru týmu dát záležet z hlediska jejich profese, znalostí a zkušeností (Sommerville, 2011).

Prostředí je tok činností poskytující vývojové organizaci metodiku, infrastrukturu a nástroje, které podporují vývojový tým (Vondrák, 2002).

3.4 Aspekty a specifika vývoje webové mapové platformy

Tato kapitola si klade za cíl přiblížit aspekty a specifika, které vychází z softwarového procesu vývoje webové mapové platformy. Jsou shrnuty aspekty jak kartografické a technologické, tak například i legislativní a uživatelské.

3.4.1 Kartografické aspekty

Kromě obecných předpokladů je pro tvorbu webových map důležitý kartografický aspekt. Zatímco u běžných analogových map není možné diskutovat o základních kompozičních prvcích, ve webových mapách lze sledovat trend eliminace některých prvků či jejich značnou úpravu. Navíc se ve webovém prostředí objevují zcela nové prvky, které umožňují uživateli s mapou interagovat (např. vyskakovací okno, přiblížení a oddálení, atd.) a dávají tak webovým mapám zcela nové možnosti.

Konstrukční základy webových map vycházejí ze stejných pravidel jako je to u jiných digitálních či analogových map. Podstatou konstrukce je kartografické zobrazení, které každému bodu na referenční ploše přiřadí právě jeden bod na zobrazovanou plochu (Voženílek a Kaňok, 2011). Kartografických zobrazení, příp. souřadnicových systémů, je nepřeberné množství a každé má svůj unikátní EPSG (European Petroleum Survey Group) kód. Nejběžnějším kartografickým zobrazením ve webové kartografii je Web Mercator (EPSG 3857) a lze jej označit za de facto standard. Zároveň za nejběžnější souřadnicový systém lze označit WGS 84 (World Geodetic Systém 1984). Pro transformaci mezi různými souřadnicovými systémy slouží rozšířená knihovna PROJ. Hlavní nevýhodou kartografického zobrazení Web Mercator je nevhodné zobrazení vyšších zeměpisných šířek (směrem k severnímu a jižnímu pólu). Výsledkem jsou abnormálně velké plochy u pólů (typicky Antarktida, Rusko, Grónsko, …), což může uživatele zmást. Mapa bývá zpravidla orientována na sever, pokud tomu tak není, musí být o tomto faktu uživatel obeznámen směrovkou (Nétek, 2020).

Obr. 3 Zobrazení Web Mercator (zdroj vlastní)

Některé **kompoziční prvky** mohou být z povahy zobrazovacího média mapy odlišné. Základním kompozičním prvkem samozřejmě zůstává mapové pole, které zabírá největší část obrazovky (mnohdy je to celá obrazovka). Zatímco název mapy je v kartografii povinný prvek a musí obsahovat věcné, časové a prostorové určení (Voženílek a Kaňok, 2011), ve webové kartografii bývá název zcela vypuštěn. Za její náhradu bývá považován *title* webové stránky, který se zobrazí v názvu záložky, případně se může jednat například o název článku, pokud je mapa jeho součástí. Dalšími nezbytnými prvky jsou legenda a měřítko. Každá webová mapa musí obsahovat minimálně grafické měřítko, od číselného měřítka se běžně upouští (mnohdy ani není možné jej zobrazit kvůli jeho neceločíselnosti). Absence měřítka a legendy bývá častou chybou ve webové kartografii. Posledním základním kompozičním prvkem je tiráž (v případě webové mapy hovoříme spíše o copyrightu), která uvádí informace autorovi a datu vzniku mapy a bývá zobrazena v pravém dolním rohu webové mapy. Celková kompozice webové mapy je liberálnější a bývá prosazován trend maximalizace mapového pole (Nétek, 2020).

Mimo základních kompozičních prvků se v mapě objevují i **nadstavbové kompoziční prvky,** které zvyšují atraktivitu a interaktivitu. Zcela běžnými bývají texty, grafy nebo infografika. V případě webových map lze tento seznam rozšířit o další prvky jakou jsou například ovládací tlačítka, vyhledávání, menu nebo vyskakovací (pop-up) okna. Většina zmíněných prvků se zobrazuje v postranním panelu, který se dá skrýt.

Základními vyjadřovacími prostředky v mapách jsou bodový znak, liniový znak a plošný znak (polygon). U těchto znaků prostředků je možné zvolit proměnné definující jejich vzhled, např. velikost, barva, výplň, průhlednost apod. Mezi vybrané vyjadřovací metody patří například metoda bodových, liniových či plošných znaků. Dalšími metodami, které se používají ve webové kartografii jsou například kartogram, metoda teček či metoda izolinií. Podrobněji se vyjadřovacím prostředkům věnuje Voženílek a Kaňok (2011).

3.4.2 Technologické a datové aspekty

Technologické aspekty popisují základní technické specifikace a parametry tvorby webových map. Základní principem jakéhokoliv webu je architektura klient – server, kde mezi sebou komunikuje koncové zařízení uživatele se serverem, nejčastěji pomocí protokolu HTTP(S) (Hypertext Transfer Protocol (Secure)).

Datové aspekty popisují datové formáty, manipulaci s nimi a jejich uložení. Každý projekt je unikátní a proto si vyžaduje z hlediska datového aspektu vlastní způsob řešení. Ve webové kartografii rozlišujeme podkladové a tematické vrstvy (data). Podkladové vrstvy zobrazují topografický podklad a fungují na principu vektorových či rastrových dlaždic. Nejčastějším způsobem přidání podkladové vrstvy do mapy je pomocí služeb WMS či WMTS od jejich poskytovatele (Pavlíček, 2019). Tematické vrstvy překrývají podkladové vrstvy, mohou nést jakýkoliv tematický obsah a zobrazují hlavní obsah mapové aplikace.

Tyto vrstvy můžeme ukládat lokálně (souborově, například GeoJSON), na serveru (v databázi, například PostgreSQL, MySQL, SQLite apod.) nebo hybridně. Databáze (prostorové) jsou pokročilejší variantou uložení a sdílení prostorových dat, umožnují provádění prostorových operací nad geografickými daty (Nétek, 2020). V případě, že jsou vrstvy statického charakteru a není s nimi často upravováno, je možné použít mapové servery, které z dat uložených na serveru připraví datový náhled a uživateli zašlou pouze náhled v obrazové formě.

3.4.3 Legislativní a komerční aspekty

Z legislativního hlediska je zcela zásadní rozdělit produkty podle jejich otevřenosti zdrojového kódu, zpoplatnění a licenčních podmínek. Podle Nétka 2020) můžeme podle legislativních aspektů dělit tři stupně otevřenosti a dostupnosti daného řešení.

Proprietální software je software s uzavřeným kódem a jeho autor striktně stanovuje pravidla jeho používání. Dále neumožňuje software nijak upravovat a mnohdy bývá jeho užívání zpoplatněno. Jako příklad bychom mohli uvést ArcGIS Online. **API** (Application Programming Interface) označuje rozhraní pro naprogramování vlastní aplikace, v praxi to znamená, že uživatel může běžně využívat předem připravených nástrojů, služeb či dat. Poskytovatel je zodpovědný za vývoj, stanovuje pravidla používání a k přístupu je běžně vyžadován tzv. API klíč. Příkladem je např. Mapbox API nebo Mapy.cz API. Posledním stupněm otevřenosti a dostupnosti je **Free and Open source**, který odkazuje na svobodu užívání a otevřenost zdrojového kódu (Nétek, 2020).

Autorské právo a licence určuje limity používání daného produktu. V oblasti GIS (geografických informačních systému) se podle Vondrákové (2014) nejčastěji používají tyto licence:

- Komerční licence
- OEM
- Shareware
- Svobodné (otevřené) licence – mezi tyto licence řadíme např. Apache licenci, BSD (Berkeley Software Distribution), CC (Creative Commons) nebo MIT (Massachusetts Institute of Technology) licenci.
- a další …

Z **komerčních/bussiness aspektů** se bavíme zejména o tématu zpoplatnění daného softwaru. Na trhu webových mapových produktů je možné se setkat s mnoha komerčními modely (jednorázový poplatek, platba za funkcionalitu, za určité časové období, počet zobrazení apod.) Produkty z hlediska zpoplatnění můžeme dělit na tři skupiny. Plně zpoplatněné produkty nabízejí špičkové technologie a bývají výsadou velkých firem (ESRI, Google, …). Preferovanější skupinou je částečně zpoplatněný software, který využívají mnohé cloudové firmy jako CARTO, Mapbox, MapTiler atd. Základní produkt bývá uživateli nabídnut zdarma a pro doplňkové či rozšiřující služby je potřeba zaplatit (Nétek, 2020). Další možností zpoplatnění je například zvýšení různých limitů jako jsou počet zobrazení mapy, počet vytvořených map apod. Poslední skupinou jsou zcela bezplatné produkty, které jsou většinou navázané na free and open source řešení. Konkrétně se jedná např. o produkty Leaflet, Openlayers aj.

3.4.4 Softwarové aspekty

V této podkapitole jsou popsány mapové knihovny a jejich charakteristiky. Volbou mapové knihovny je přímo ovlivněna funkcionalita, služby, nástroje a operace výsledného produktu. Ve srovnání s desktopovým GIS je nabídka nástrojů a operací v prostředí webu značně omezená a zpravidla bývají implementovány pouze jednodušší nástroje. Tyto knihovny rozdělujeme do kategorií otevřené, API a cloudové. Jejich přístup k otevřenosti zdrojového kódu, limitací apod. je popsán v podkapitole 3.4.3.

Zdrojový kód u **otevřených knihoven** umožňuje vlastní přizpůsobení kódu svým požadavkům, nevýhodou je nutná znalost programovacího jazyka. Mezi tyto knihovny řadíme např. Openlayers a Leaflet. Openlayers je populární knihovna pro tvorbu webových map, která podporuje velké množství formátů a slouží jako výchozí mapová knihovna data OpenStreetMap (Openlayers). Konkurencí ke zmíněné knihovně je Leaflet, klade důraz na svoji lehkost a jednoduchost. Knihovna nativně pracuje s formátem GeoJSON a je možné ji rozšířit díky mnoha pluginům (Leaflet). Mezi další zástupce otevřených knihoven řadíme např. D3.js, Cesium nebo Flowmap.blue.

API naproti tomu umožňuje v předem definovaných mantinelech užívat předem definované služby, nástroje a operace. Hranice mezi API a otevřenými knihovnami je obtížné stanovit a částečně se mohou překrývat. Příkladem jsou Mapy.cz API, ArcGIS API for JS, Mapbox GL JS a další.

Nejpokročilejší nástroje a operace poskytují v oblasti webové kartografie **cloudové řešení**. Důležitým rozdílem oproti předchozím řešením je uživatelsky přívětivé prostředí, kde není od uživatele vyžadována žádná znalost programování. Mezi nejznámější cloudová řešení patří ArcGIS Online, CARTO, MapTiler Cloud nebo Mapbox (Nétek, 2020).

3.4.5 Uživatelské aspekty

Z hlediska uživatelského aspektu je nutné definovat a vymezit rozdíl mezi základními pojmy **User Experience (UX)** a **User Interface (UI**).

UX se zabývá celkovým dojmem uživatele při interakci s daným rozhraním či produktem a zaměřuje se na optimální uživatelské prostředí, ve kterém se uživatel snadno zorientuje a zvýší svou efektivitu práce. Designéři UX se snaží porozumět chování a potřebám uživatelů a snaží se vytvořit intuitivní, funkční, smysluplné a příjemné rozhraní (Hartson, 2012). Naproti tomu **UI** se zaměřuje na vizuální stránku. Zaměřuje se na všechny

prvky, se kterými může uživatel pracovat (např. menu, tlačítka, ikony, obrázky, barvy apod.). Cílem UI je pozitivní vizuální zážitek uživatele (Galitz, 2007).

Přestože jsou UX a UI odlišné disciplíny, velmi úzce spolu souvisí a bez sebe nemohou fungovat. Zatímco UX se zabývá intuitivností, funkčností a smysluplností uživatelského rozhraní či systému, UI se zabývá jeho grafickou stránkou.

Obr. 4 Rozdíl mezi UX a UI (Difference Between UI and UX Design, 2023)

Návrh a implementace uživatelského rozhraní přináší do softwarového procesu důležitý prvek estetiky a funkčnosti, k čemuž jsou často využívány knihovny kaskádových stylů (CSS) jako Bootstrap nebo Tailwind CSS. Tyto knihovny nabízejí bohatou sadu předem vytvořených komponent a stylizovaných prvků, které vývojářům umožňují rychle a konzistentně vytvářet moderní uživatelské rozhraní. Integrace těchto knihoven do softwarového vývoje může přinést několik výhod. Nejenže usnadňují a zrychlují proces tvorby uživatelského rozhraní, ale také zajišťují konzistenci a profesionální vzhled aplikace. To je zvláště užitečné při rychlém iterativním vývoji, kde je důležité udržet krok s novými funkcemi a zároveň zajistit, že uživatelé budou mít pozitivní zkušenost s uživatelským aspektem aplikace.

4 METODY A POSTUP ZPRACOVÁNÍ

Tato kapitola popisuje kompletní postup prací při psaní diplomové práce. Dále jsou zde uvedeny použité metody, datové zdroje a programy, které byly během práce použity.

Použité metody

V diplomové práci byly použity běžné metody softwarového inženýrství, uplatněna byla zejména metodika Rational Unified Process, která je blíže popsána v kapitole 3.3. Tato metodika využívá iterativní přístup, kde se vývoj opakuje v jednotlivých fázích, jako jsou specifikace požadavků, analýza, návrh, implementacetestování či nasazení.

Použitá data

Vzhledem k charakteru diplomové práce nebyla pořizována ani vytvářena žádná vlastní nová data. Ve webové mapové platformě, která je hlavním výsledkem diplomové práce, byla za účelem testování nahrávání dat použita data z webové stránky geojson.xyz.

Použité programy a technologie

Použité programy jsou spíše z oblasti analýzy a návrhu softwaru a jeho programování. Pro tvorbu diagramů byl použit online nástroj Draw.io, grafické práce a vizuální identita byly vytvořeny v Adobe Photoshop a Adobe Illustrator. Programování probíhalo v softwaru Atom. Z dalšího nutného programového vybavení je nutné zmínit XAMPP pro lokální vývoj aplikace. Tento programový balíček instaluje webový server, databázi a interpret programovacích jazyků PHP a Perl.

Z programovacích jazyků byly pro vytvoření webové mapové platformy vybrány jazyky PHP a Javascript. Zatímco Javasript společně s HTML (značkovací jazyk pro strukturu webové stránky) a CSS (kaskádové styly) tvoří tzv. frontend, PHP zajišťuje backend a obstarává komunikaci s databází, autentizaci a další.

Do aplikace je zahrnuto mnoho metod a knihoven. Na straně Javascriptu jsou to např. AJAX, jQuery, Leaflet, Turf.js nebo Datatables. Důkladnější popis a důvody výběru vybraných technologií je probrán v následující kapitole 5.

Postup zpracování

Postup zpracování diplomové práce se dá rozdělit do dvou etap – teoretické a praktické části. Po obdržení zadání práce byla provedena rešerše zdrojů, které se danou tématikou zabývají.

V teoretické části byly vymezeny základní pojmy a připomenuta stručná historie softwarového inženýrství. Dále byly rozebrány modely a metodiky životního cyklu softwaru a z nich byl vybrán Rational Unified Process, který byl popsán podrobněji a byly nastíněny jeho jednotlivé vývojové fáze. V poslední řadě byly zpracovány aspekty a specifika vývoje webové mapové platformy. Mezi tyto aspekty patřily kartografické, technické, datové, legislativní, komerční, softwarové a uživatelské hledisko.

V praktické části autor specifikuje, navrhuje a programuje vlastní řešení webové mapové platformy, která obsahuje prvky webové kartografie a implementuje geoprocessingové nástroje, prostorové operace a vizualizaci v prostředí webu. Celý postup vývoje se inspiroval v teoretické části v metodice RUP. Součástí výstupu je i návod na instalaci aplikace na vlastní stroj. Na závěr jsou výsledky diskutovány a prezentovány.

Součástí zadání práce byla i tvorba webových stránek a posteru ve formátu A2, které byly dokončeny jako poslední. Konečná verze diplomové práce, včetně všech příloh, je volně dostupná na webových stránkách Katedry geoinformatiky Univerzity Palackého.

5 SOFTWAROVÝ PROCES VÝVOJE WEBOVÉ MAPOVÉ PLATFORMY

Touto kapitolou začíná praktická část diplomové práce. Je v ní podrobně popsán softwarový vývoj webové mapové platformy podle náležitostí a metodik softwarového inženýrství. Vývoj byl veden zejména s ohledem na metodiku Rational Unified Process. Student navrhnul, ověřil a ilustroval postup vývoje webové mapové platformy. Ze zadání je zřejmé, že případová studie se bude věnovat online mapové aplikaci, která se bude zaměřovat na implementaci geoprocessingových nástrojů, prostorových operací a vizualizace v prostředí webové kartografie.

5.1 Byznys modelování

V rámci diplomové práce se autor zaměřil na vývoj webové mapové platformy, která bude uživatelům poskytnuta zdarma. Přestože je byznys model důležitým aspektem vývoje softwaru, v tomto případě z akademických důvodů není jiná možnost než aplikaci poskytnout zdarma. Z tohoto důvodu je kapitola byznys modelování, která se typicky věnuje analýze a způsobu generování příjmů a strukturou podnikání, přeskočena. V případě, že by se nejednalo o diplomovou práce, ale o komerční produkt, jsou nastíněny možné scénáře generování zisku v diskuzi.

5.2 Specifikace požadavků

Úkolem kapitoly specifikace požadavků projektu je stanovit jeho cíle a požadované vlastnosti, které by měl splňovat. Specifikace požadavků je důležitou kapitolou, která přináší podklady pro analýzu a návrh softwaru. Požadavky by měly být konkrétní, měřitelné, relevantní a časově omezené.

Běžnou součástí specifikace požadavků jsou metody získávání informací od zadavatele, např. rozhovor, dotazník, práce u zadavatele či analýza současného softwarového systému. Vzhledem k povaze diplomové práce bude však tato část velmi strohá, jelikož jediné relevantní cíle jsou stručně shrnuty v zadání diplomové práce. Práce u zadavatele či analýza současného softwarového systému byly kompletně přeskočeny, neb žádný současný software neexistuje.

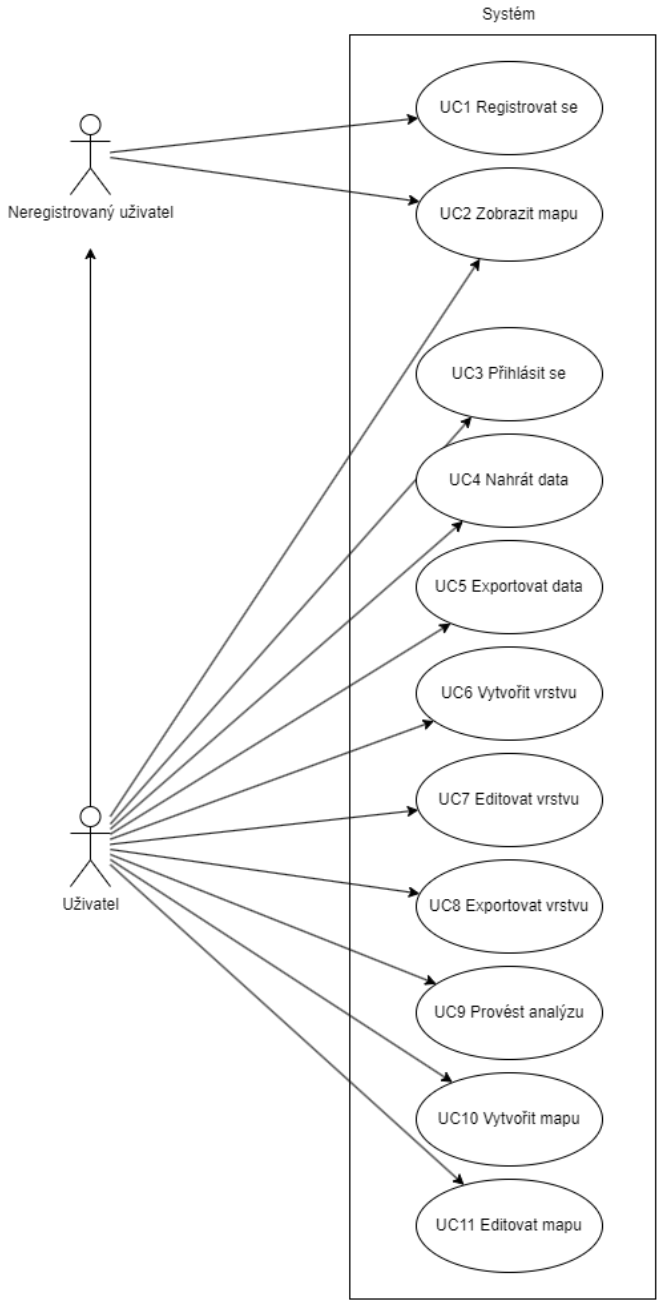
Z hlediska specifikace požadavků je v zadání práce klíčová tato pasáž: *Tato studie se zaměřuje na vývoj webové mapové platformy, která klade důraz na implementaci geoprocessingových nástrojů, prostorových operací a vizualizaci v prostředí webové kartografie.* Byla provedena dekompozice, z čehož vyplynuly základní požadavky na funkcionalitu:

1. Vyvinout webovou mapovou platformu
2. Implementovat geoprocessingové nástroje a prostorové operace
3. Implementovat vizualizaci geografických dat v prostředí webové kartografii

Podrobnější specifikaci požadavků si autor určil sám. Z prvního bodu vyplývá, že platforma musí být prezentována ve webovém prostředí. Dále by měla mít platforma uživatelský systém tak, aby si uživatel mohl svá data uložit a později se k nim vrátit. Navíc aplikace umožní nahrát svá vlastní geografická data ve formátech Shapefile a GeoJSON a uživatel si bude moct data prohlédnout jak z hlediska geometrie, tak atributů. U dat bude

možné upravit geometrii a změnit znakový klíč. Uživatel bude dále moct provést vybrané prostorové operace a z geografických dat vytvořit mapu, kterou bude moct sdílet.

Další cíle a požadavky jako například finanční, tematické či návštěvnost nejsou nijak specifikovány. Z výše uvedených cílů a požadavků byl vytvořen následující diagram základních případů užití (use cases), který byl vytvořen v aplikaci Draw.io. Tato aplikace byla vybrána zejména z důvodu, že je online, zdarma, nabízí podporu několika formátů a širokou škálu šablon pro různé typy diagramů (Flowchart Maker & Online Diagram Software). Pro jednotlivé případy užití byly následně vytvořeny detailní specifikace, které nemají žádnou pevně definovanou formu a můžou být specifikovány např. v tabulce či v textu tak je to v této práci.



Obr. 5 Případy užití (zdroj vlastní)

UC1 (Use Case) - Registrovat se

Krátký popis: Uživatel chce zaregistrovat nový účet v aplikaci.

Aktéři: Neregistrovaný uživatel

Podmínky pro spuštění: Uživatel otevře příslušnou webovou stránku aplikace.

Základní tok:

1. Uživatel otevře webovou stránku aplikace.
2. Uživatel klikne na tlačítko Sign Up (Registrovat se).
3. Aplikace zobrazí registrační okno.
4. Uživatel vyplní e-mail, heslo a ještě jednou stejné heslo pro kontrolu.
5. Klikne na tlačítko Sign Up (Registrovat se)
6. Aplikace ověří zadané údaje a vytvoří v databázi nový uživatelský účet.
7. Aplikace vygeneruje aktivační kód pro ověření e-mailové adresy a odešle ověřovací e-mail.
8. Aplikace informuje o vytvoření účtu.
9. Uživatel vstoupí do své e-mailové schránky, otevře e-mail a klikne na aktivační odkaz.
10. Uživatel je přesměrován na přihlašovací stránku a může se přihlásit do aplikace.

Alternativní tok:

4a. Pokud uživatel zadal již existující e-mailovou adresu, zobrazí se mu informační hláška o již existujícím účtu.

5a. Pokud uživatel nevyplní všechny údaje, zobrazí se informační hláška o nutnosti doplnění všech povinných údajů.

6a. Pokud se heslo a ověření hesla neshodují, zobrazí se informační hláška o neshodě mezi hesly.

Podmínky pro dokončení: Uživatel úspěšně dokončí ověření účtu, je registrován a může se nyní přihlásit do aplikace.

UC2 – Zobrazit mapu

Krátký popis: Uživatel chce zobrazit vytvořenou mapu.

Aktéři: Neregistrovaný uživatel, Uživatel

Podmínky pro spuštění: Uživatel otevře příslušný odkaz vedoucí k vytvořené mapě. Mapa musí být předem vytvořena v systému.

Základní tok:

1. Uživatel otevře URL adresu vedoucí k vytvořené mapě (Přihlášený uživatel v detailu příslušné mapy klikne na tlačítko View map).
2. Aplikace zobrazí mapu.

Alternativní tok:

1a. Uživatel zadá nesprávnou URL adresu, případně je v URL adrese špatně zadán parametr identifikace mapy. V takovém případě aplikace přesměruje uživatele na domovskou stránku.

Podmínky pro dokončení: Uživateli byla zobrazena mapa.

UC3 - Přihlásit se

Krátký popis: Uživatel se chce přihlásit do aplikace.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel otevře příslušnou webovou stránku aplikace.

Základní tok:

1. Uživatel otevře webovou stránku aplikace.
2. Uživatel klikne na tlačítko Sign In (Přihlásit se).
3. Aplikace zobrazí přihlašovací okno.
4. Uživatel vyplní e-mail a heslo.
5. Klikne na tlačítko Sign In (Přihlásit se)
6. Aplikace ověří zadané údaje.
7. Aplikace přihlásí uživatele a přesměruje ho na domovskou stránku aplikace.

Alternativní tok:

5a. Pokud uživatel nevyplní všechny údaje, zobrazí se informační hláška o nutnosti doplnění všech povinných údajů.

6a. Pokud jsou zadané údaje neplatné, zobrazí se informační údaje o špatně zadaných údajích.

Podmínky pro dokončení: Uživatel se úspěšně přihlásí a je přesměrován na domovskou stránku aplikace.

UC4 – Nahrát data

Krátký popis: Uživatel ze svého PC nahraje do aplikace vlastní geografická data.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace v záložce Data. Má u sebe v PC připravena geografická data ve formátu geojson nebo shapefile (komprimovaný v zipu).

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Data.
3. Uživatel klikne na tlačítko Upload data (Nahrát data).
4. Aplikace zobrazí modal (vyskakovací okno) s formulářem pro nahrání dat.
5. Uživatel klikne do formuláře a vybere z počítače požadovaný soubor.
6. Uživatel klikne na tlačítko Upload.
7. Aplikace provede kontrolu validity a správnosti dat, aby se minimalizovaly možné chyby při nahrávání.
8. Aplikace nahraje soubor do databáze.
9. Aplikace zavře vyskakovací okno a zobrazí nově nahraná data v tabulce.
10. Uživatel je o nahrání dat do databáze informován hláškou o úspěšném uložení dat.

Alternativní tok:

3a. Uživatel klikne na tlačítko Close (Zavřít), čímž zavře vyskakovací okno a zruší akci.

5a. Pokud nejsou splněny podmínky pro nahrání (požadovaný formát dat, správný formát dat, maximální velikost dat), zobrazí se uživateli hláška o chybě.

Podmínky pro dokončení: Uživatel úspěšně nahraje svá geografická data do aplikace.

UC5 – Exportovat data

Krátký popis: Uživatel exportuje svá data z aplikace do svého PC.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace, v záložce Data je v detailu konkrétních dat.

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Data a rozklikne požadovaná data.
3. Uživatel rozklikne rozbalovací menu s názvem dat a šipkou dolu.
4. Uživatel vybere tlačítko Export data.
5. Aplikace zobrazí modal (vyskakovací okno) s formulářem pro export dat.
6. Uživatel vybere z rozbalovací nabídky požadovaný formát dat.
7. Uživatel klikne na tlačítko Export.
8. Aplikace vygeneruje odkaz pro stažení a zobrazí ho.
9. Uživatel klikne na odkaz LINK TO DOWNLOAD a stáhne se soubor do počítače.

Alternativní tok:

Podmínky pro dokončení: Uživatel úspěšně exportuje svá geografická data z aplikace do svého počítače.

UC6 – Vytvořit vrstvu

Krátký popis: Uživatel chce vytvořit novou vrstvu.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace a je na záložce Layers (Vrstvy)

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Layers a klikne v ní na tlačítko New layer (Nová vrstva).
3. Uživatel klikne na tlačítko Empty layer (prázdná vrstva).
4. Aplikace zobrazí vyskakovací okno s formulářem.
5. Uživatel vyplní požadované informace jako název vrstvy a typ geometrie.
6. Uživatel klikne na tlačítko Create (Vytvořit).
7. Aplikace zavře vyskakovací okno a informuje o vytvoření vrstvy informační zprávou.

Alternativní tok:

3a. Uživatel klikne na tlačítko From data (z dat), čímž chce vytvořit vrstvu ze svých nahraných geografických dat. V tom případě se uživateli zobrazí vyskakovací okno, kde v rozbalovacím menu vybere data, ze které se vrstva vytvoří.

6a. Uživatel klikne na tlačítko Close, čímž zavře okno a zruší akci.

6b. Pokud nejsou korektně vyplněna požadovaná pole, zobrazí se informační hláška o nutnosti jejich vyplnění.

Podmínky pro dokončení: Uživatel úspěšně vytvoří novou vrstvu.

UC7 – Editovat vrstvu

Krátký popis: Uživatel chce vytvořit editovat vrstvu.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace, je na záložce Layers v detailu konkrétní vrstvy.

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Layers a rozklikne požadovanou vrstvu.
3. Uživatel rozklikne rozbalovací menu s názvem vrstvy a šipkou dolu.
4. Pro úpravu základní informací o mapě uživatel klikne na tlačítko Layer info (Informace o vrstvě).
5. Ve vyskakovacím okně uživatel upraví informace a uloží je tlačítkem Update, u uložení je uživatel informován informační hláškou.

Alternativní tok 1 (úprava znakového klíče):

4a. Chce-li uživatel upravit znakový klíč vrstvy, klikne na tlačítko Symbology. V novém vyskakovacím okně zvolí nastavení zobrazení vrstvy a její znakový klíč, následně jej uloží tlačítkem Save. U uložení je uživatel informován hlášením.

Alternativní tok 2 (úprava geometrie):

3a. Pokud chce uživatel upravit geometrii vrstvy, jsou v mapovém poli v levém horním rohu připraveny tlačítka pro její vytvoření, editaci a smazání.

Podmínky pro dokončení: Uživatel úspěšně upraví svou vrstvu a uloží ji do databáze.

UC8 – Exportovat vrstvu

Krátký popis: Uživatel chce exportovat vrstvu do svých dat.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace, v záložce Layers je v detailu konkrétní vrstvy.

Základní tok:

6. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
7. Uživatel klikne na záložku Layers a rozklikne požadovanou vrstvu.
8. Uživatel rozklikne rozbalovací menu s názvem vrstvy a šipkou dolu.
9. Uživatel vybere tlačítko Export layer.
10. Aplikace informuje o exportu vrstvy do dat informační hláškou.

Alternativní tok:

Podmínky pro dokončení: Uživatel úspěšně exportuje svou do dat, odkud ji dále může stáhnout do svého počítače.

UC9 – Provést analýzu

Krátký popis: Uživatel chce provést analýzu nad svými vrstvami.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace a je na záložce Analysis (Analýza)

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Analysis a klikne v ní na tlačítko Spatial tools.
3. Uživatel z rozbalovacího okna vybere požadovanou funkci.

4. Po výběru aplikace zobrazí vyskakovací okno s formulářem.
5. Uživatel vyplní požadované informace nutné k provedení analýzy jako jsou vstupní vrstvy a parametry.
6. Uživatel klikne na tlačítko Execute (Provést).
7. Aplikace provede analýzu, uzavře vyskakovací okno a zobrazí novou vrstvu v mapě.
8. Aplikace informuje o úspěšném provedení analýzy informační hláškou.

Alternativní tok:

5a. Pokud uživatel chce, aby se nově vytvořená vrstva uložila do databáze, zaškrtne políčko Export to layers (Exportovat do vrstev).

6a. Pokud nejsou vyplněny korektně všechny údaje, aplikace o tom informuje uživatele informační hláškou.

Podmínky pro dokončení: Uživatel úspěšně dokončí analýzu vrstvy.

UC10 – Vytvořit mapu

Krátký popis: Uživatel chce vytvořit novou mapu.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace a je na záložce Maps (Mapy)

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Maps a klikne v ní na tlačítko New map (Nová mapa).
3. Aplikace zobrazí vyskakovací okno s formulářem.
4. Uživatel vyplní požadované informace jako název mapy.
5. Uživatel klikne na tlačítko Create (Vytvořit).
6. Aplikace zavře vyskakovací okno a informuje o vytvoření mapy informační zprávou.

Alternativní tok:

5a. Uživatel klikne na tlačítko Close, čímž zavře okno a zruší akci.

5b. Pokud nejsou korektně vyplněna požadovaná pole, zobrazí se informační hláška o nutnosti jejich vyplnění.

Podmínky pro dokončení: Uživatel úspěšně vytvoří novou mapu.

UC11 – Editovat mapu

Krátký popis: Uživatel chce editovat mapu.

Aktéři: Uživatel

Podmínky pro spuštění: Uživatel je přihlášen do aplikace a je na záložce Maps (Mapy) v konkrétní mapě.

Základní tok:

1. Uživatel otevře webovou stránku aplikace a přihlásí se do ní.
2. Uživatel klikne na záložku Maps a rozklikne požadovanou mapu.
3. Aplikace zobrazí detail mapy s formulářem, ve kterém uživatel může měnit nastavení mapy.
4. Uživatel nastaví požadované parametry mapy jako je název, podkladové mapy nebo pluginy.
5. Kliknutím na tlačítko Add layer (Přidat vrstvu) aplikace zobrazí vyskakovací okno se všemi dostupnými vrstvami, které uživatel může přidat do mapy.

6. Kliknutím na tlačítko Add layer je vrstva přidána do mapy. Uživatel je o tom informován informačním hlášením.
7. Uživatel uloží změny kliknutím na tlačítko Save changes (Uložit změny).

Alternativní tok:

6a. Pokud je již vrstva v mapě přidána, uživateli se objeví informační hláška s chybou.

6b. Pokud uživatel nevybere žádnou mapu, je o tom informován hláškou.

6c. Chce-li uživatel vrstvu z mapy odstranit, učiní tak kliknutím na název vrstvy.

Podmínky pro dokončení: Uživatel úspěšně upraví svou mapu a uloží všechny změny.

5.3 Analýza a návrh

Touto kapitolou autor plynule navazuje na předchozí etapu. V analytické části jsou pro pochopení a vizualizaci struktur použity diagramy tříd. Dále jsou k analýze použity nástroje jako předimplementační analýza, analýza konkurence, SWOT analýza nebo použití person se scénáři. V návrhové části se autor zaměřuje na architekturu aplikace, způsob uložení dat, datové a systémové struktury, použití komponent a modulů. Pozornost je věnována i výběru knihoven a specifikům vývoje webové mapové platformy. Dále se zaměřuje na vizuální stránku aplikace, ke které využívá např. wireframy.

5.3.1 Analýza

V rámci souhrnné předimplementační analýzy byly zodpovězeny základní otázky na požadovaný produkt a provedeny další, detailnější analýzy. Některé základní parametry a specifikace byly zároveň popsány v předchozí kapitole. Jedním z důležitých nástrojů pro analýzu je SWOT (z anglického **S**trengths, **W**eaknesses, **O**pportunities, **T**hreats) analýza, která definuje silné a slabé stránky, příležitosti a hrozby.

Silné stránky:

1. Aplikace je dostupná všem a zdarma.
2. Uživatelsky přívětivé rozhraní, které je responzivní.
3. Flexibilita a rozšiřitelnost – architektura aplikace umožňuje snadné rozšíření o další funkcionalitu a moduly.

Slabé stránky:

1. Složitý vývoj
2. Silná konkurence.
3. Náklady na provoz aplikace

Příležitosti:

1. Široké využití jak mezi veřejností, tak v odborné komunitě.

Hrozby:

1. Silná a početná konkurence, uživatelé jsou zvyklí používat konkurenční produkty.
2. Bezpečností rizika, hrozba kybernetických útoků

Na základě hrozeb SWOT analýzy, ze které vyplynula silná a početná konkurence, byla dále vytvořena podrobnější analýza konkurence. Na poli webových mapových aplikací je konkurence velmi početná a silná, proto bylo do analýzy zahrnuto pouze několik nejznámějších aplikací, a to ArcGIS Online, CARTO a GIS Cloud. Mezi další konkurenci, která stojí alespoň za zmínku, jsou například aplikace Mapbox, Google Maps nebo Mapy.cz.

Obr. 6 Logo ArcGIS Online (ArcGIS Online)

ArcGIS Online (2023)

Poskytovatel: ESRI

Hlavní funkcionalita: ArcGIS Online nabízí komplexní platformu, která nabízí širokou škálu funkcí pro správu, analýzu a sdílení geografických dat.

Silné stránky:

1. Velká funkcionalita a pokročilé nástroje
2. Podpora velkého množství formátů
3. Integrace s dalšími produkty od firmy ESRI a propojení ekosystému
4. Velké množství možností vizualizace dat, včetně např. Story Maps

Slabé stránky:

1. Vysoká cena
2. Pro neznalé uživatele složité ovládání hodící se spíše pro odborníky

Obr. 7 Logo CARTO (Carto)

CARTO (2023)

Poskytovatel: CARTO

Hlavní funkcionalita: CARTO je moderní nástroj, který klade důraz na vizualizaci a analýzu dat. Nabízí velké množství nástrojů pro tvorbu a analýzu vizuálně přitažlivých map.

Silné stránky:

1. Jednoduché a intuitivní prostředí
2. Velké množství nástrojů.
3. Pokročilá vizualizace dat.

Slabé stránky:

1. Cena
2. Limity při práci s velkými objemy dat.

Obr. 8 Logo GIS Cloud (GIS Cloud)

GIS Cloud (2023)

Poskytovatel: GIS Cloud

Hlavní funkcionalita: GIS Cloud je webová mapová platforma, která je zaměřena na rychlou a jednoduchou vizualizaci geografických dat. Nabízí nástroje pro tvorbu a sdílení map, editaci dat a prostorovou analýzu.

Silné stránky:

1. Široká funkcionalita
2. Velké množství podporovaných formátů
3. Rychlé rozhraní

Slabé stránky:

1. Některé pokročilé GIS funkce chybí nebo jsou omezené
2. Vyšší náklady na plné využití a ukládání velkého množství dat
3. Méně přehledné rozložení stránek, starší design

Byly také vytvořeny dvě reprezentativní persony se scénáři. První personu představuje osoba, která má základní IT znalosti a nemá žádnou zkušenost s GIS. Uživatel chce zpravidla vytvořit jednoduchou mapu a sdílet ji se svými blízkými. Druhou personou je osoba, která má určité zkušenosti s GIS a chce využit nástrojů pro prostorové operace.

Persona 1:

Jméno: Karel Normální

Věk: 40

Bydliště: Praha

Vzdělání: Střední odborné

Zaměstnání: Skladník

Jak se o aplikaci dozvěděl: Vyhledávač Google

Motivace: Karel Normální nemá žádné zkušenosti s GIS a má jen základní zkušenost s PC. Aplikaci chce využívat k občasnému vytvoření jednoduché mapy, kterou chce sdílet se svými přáteli a kolegy.

Scénář 1: Karel Normální plánuje letní dovolenou s kamarády, kterou chtějí strávit na vodě. Karel Normální je zodpovědný a chce pro kamarády vytvořit mapu, ve které uvidí jednotlivě rozplánované dny, které na vodě stráví. V mapě se dozví, které úseky řeky pojedou který den, kde se dobře nají nebo jaké zajímavosti se v okolí nalézají. Karel Normální vytvoří potřebné vrstvy, které vloží do své mapy. Odkaz k mapě následně rozešle kamarádům do jejich e-mailových schránek.

Persona 2:
Jméno: Marek Chytrý
Věk: 28
Bydliště: Olomouc
Vzdělání: Vysokoškolské
Zaměstnání: GIS analytik
Jak se o aplikaci dozvěděl: Odborná komunita
Motivace: Marek Chytrý má vysokoškolské vzdělání v oblasti geoinformatiky. Aplikaci chce využívat příležitostně jako alternativu k desktopových GIS aplikacím.

Scénář 1: Marek právě obdržel od zákazníka sadu geografických dat, se kterými má provést další operace. Marek si potřebuje data zkontrolovat a zjistit jak vypadají, ale je na cestách s malým notebookem, na kterém nemá žádný desktopový GIS. K vizuální kontrole geometrie a atributů využije Marek aplikaci.
Scénář 2: Marek je právě na jednání a dostal e-mail od svého nadřízeného. Ten po něm požaduje zmenšit soubor, který je nyní příliš objemný. Marek nahraje data do aplikace, v záložce analýzy vybere možnost pro zjednodušení polygonů, čímž zmenší objem dat. Data exportuje do počítače a obratem je odešle zpět svému nadřízenému.

Dle výše specifikovaných požadavků a provedených analýzách byly vytvořeny diagramy tříd *user*, *data*, *layer* a *map* (Tabulka 1). Na základě těchto třídních diagramů byly identifikovány hlavní objekty, jejich atributy a metody, které budou součástí implementace navrhovaného systému. Bližší popis atributů včetně jejich datového typu je v návrhu databáze na obr. 11.

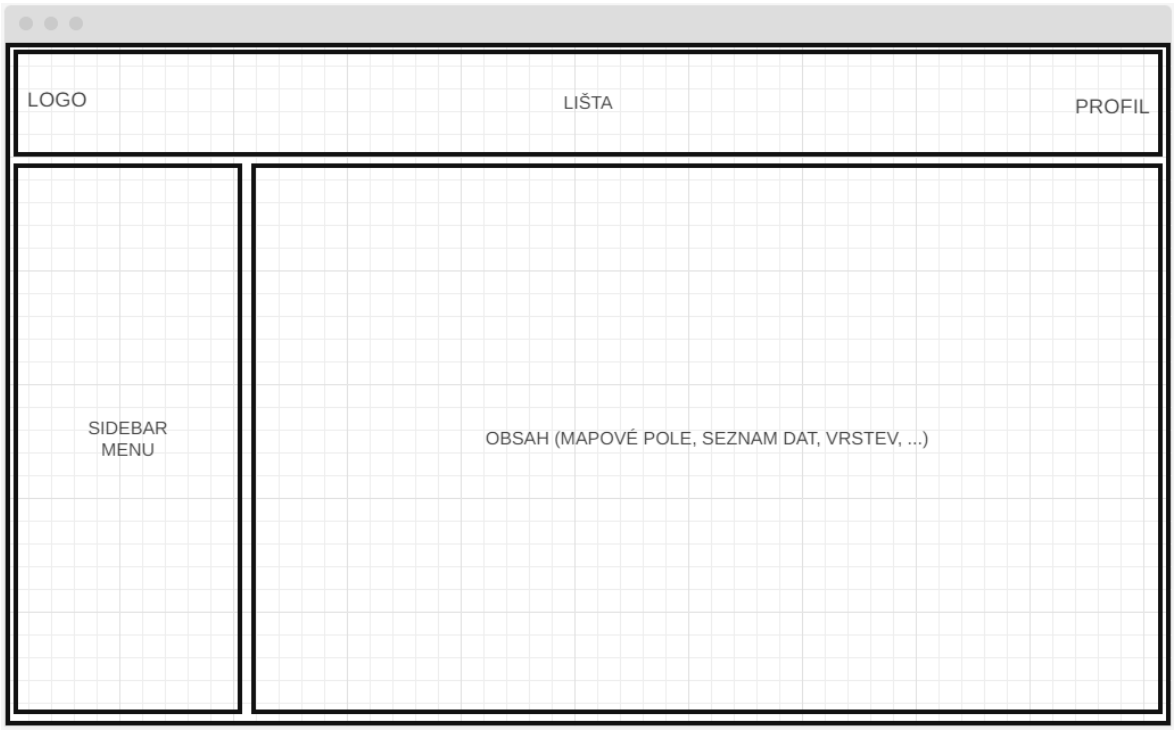
Třída user	**Třída data**	**Třída layer**	**Třída map**
user_id email password level cdate active	data_id name data_type feature_type features user_id cdate	layer_id name description feature_type features style user_id cdate	map_id name description template options layers user_id cdate shared
register() login() recoverPassword()	getAllData() viewData() deleteData() exportData()	getAllLayers() viewLayer() addLayer() deleteLayer() updateLayer() exportLayer() updateSymbology() addGeometry() updateGeometry() deleteGeometry()	getAllMaps() viewMap() getMap() addMap() addLayerToMap() removeLayerFromMap() getMapLayers() updateMap()

Tabulka 1 Diagram tříd

5.3.2 Návrh

Ve fázi návrhu byl specifikován návrh požadovaného softwarového systému. Byla navržena architektura systému, popsán způsob uložení dat, vybrány konkrétní knihovny a moduly, které byly použity. Dále bylo také navrženo uživatelské rozhraní a grafický styl aplikace. Aplikace se dá rozdělit do dvou samostatných částí – mapová aplikace a prezentační část.

Prezentační část má za úkol uživatele seznámit s funkcionalitou aplikace, ukázat její hlavní výhody nebo například umožňuje kontaktovat autora. Hlavním cílem je tedy nalákat uživatele k tomu, aby si v aplikaci založil účet. Stránka je koncipována jako one-page (jedno-stránková) webová stránka a měla by reflektovat dnešní trendy v oblasti web designu. Z prezentační aplikace je možné se skrze registrační nebo přihlašovací stránku prokliknout do **aplikační části**. Ta je přístupná pouze registrovaným uživatelům. Jak prezentační, tak aplikační část je plně responzivní. Aplikační část obsahuje v levé části navigační postranní panel s odkazy do dalších částí aplikace (Obr. 9).

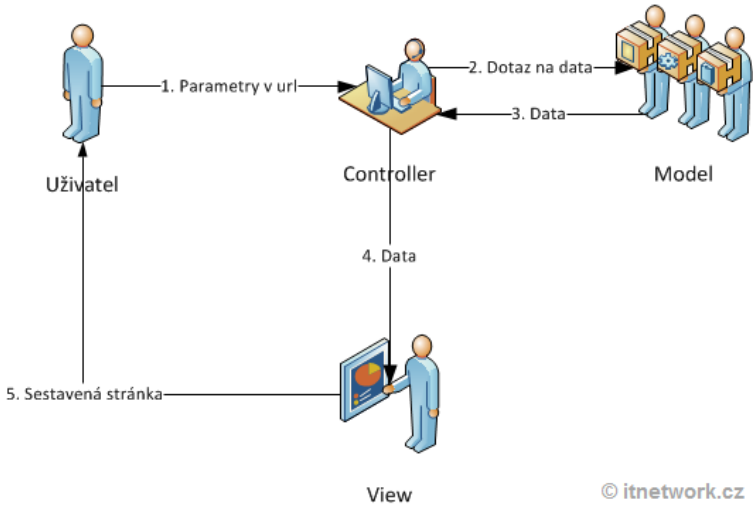


Obr. 9 Wireframe aplikace (zdroj vlastní)

Důležitou částí návrhu je **specifikace architektury systému**. Aplikace potřebuje ukládat data uživatelů a je tak zřejmé, že aplikace obsahuje jak kód na straně uživatele (client-side), tak kód na straně serveru (server-side). Byla zvolena architektura MVC (Model, View, Controller), která je populární zejména při tvorbě webů. Základní myšlenkou MVC je oddělení logiky od výstupu. Nahrazuje tak „špagetový" kód, kde se například míchá v jednom souboru PHP, Javascript a HTML. Logiku aplikace obstarává model, který řeší například databázové dotazy, validace apod. a odehrává se na straně serveru. O zobrazení výstupu uživateli se stará View a nejčastěji se jedná o HTML šablonu. Controller je prostředník mezi výše zmíněnými prvky a který je propojuje (Čápka, 2023). V praxi to tak znamená, že uživatel zadá do webového prohlížeče URL adresu s parametry.

https://mapsquare.io/app/mapView?id=nADmQqks

Jako první je načtena prázdná šablona stránky. Javascript podle parametru v URL pošle požadavek na router na serverové straně. Ten pozná, že jsou po něm požadovány informace o mapě s požadovaným ID. Provede tedy příslušné kontroly a databázový dotaz na mapu. Informace o mapě pak pošle zpět klientovi, který je vykreslí do HTML šablony.



Obr. 10 Schéma MVC architektury (Čápka, 2023)

Součástí návrhu je také specifikace, jakým způsobem bude probíhat **ukládání dat**. Jelikož je ze specifikace požadavků zřejmé, že data bude nutné uložit, rozhodl se autor, že samotná geografická data se budou ukládat do databáze, konkrétně ve formátu JSON. Nahraná data do systému jsou rozdělena na geometrii a atributy, které jsou následně nahrány do sloupců *geometry* a *properties*, oba datového typu JSON, případně LONG_TEXT. Při práci s daty jsou geometrie a atributy opět spojeny do jednoho JSON souboru, který je odeslán uživateli. Tento datový typ umožňuje provádět základní prostorové a analytické operace již v prostředí databáze.

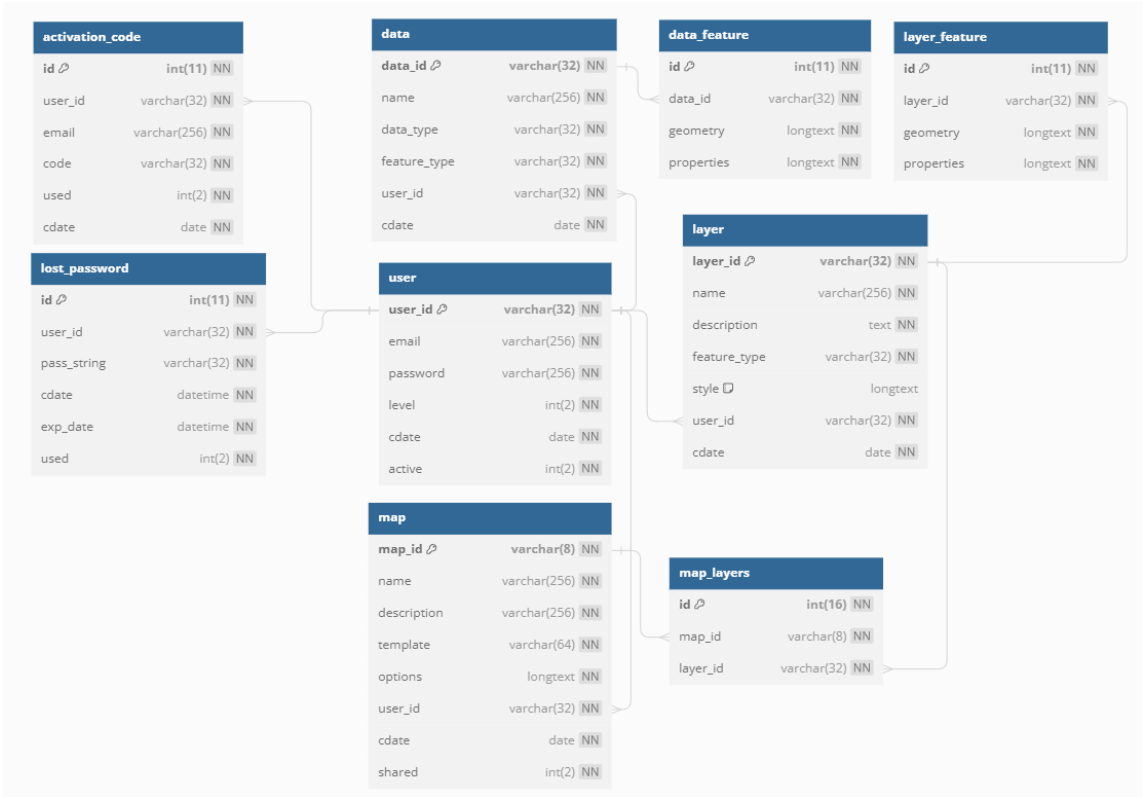
Kvůli jednoduchosti, robustnosti a usnadnění nasazení byly vybrány hojně používané programovací jazyky Javascript a PHP s celou řadou knihoven. Tyto programovací jazyky byly zvoleny zejména kvůli zkušenostem autora, paralelně probíhal vývoj i v Node.js a React, což je blíže popsáno v diskuzi. Jako databáze byla vybrána MySQL, nicméně aplikace by fungovala i na jiných DBMS (systém řízení báze dat), jelikož datový typ JSON, příp. LONG_TEXT, je dostupný ve všech známých DMBS (např. MariaDB, PostgreSQL apod.). Podrobnější popis systému je následující kapitole Implementace. Adresářová struktura aplikace byla navržena takto:

- **kořenový adresář**
  - **app** (kořenový adresář mapové aplikace)
    - assets
    - auth
    - css
    - download
    - server
    - tmp
    - vendor
  - **assets** (styly, skripty, obrázky pro prezentační část)
    - css
    - fonts
    - images
    - js
  - **s** (adresář pro sdílení map)
    - assets

5.4 Implementace

V této kapitole je podrobně popsána implementace jednotlivých komponent. Byla použita implementace v několika iteracích se strategií zdola-nahoru, kde byly jako první implementovány základní části kódu. Jak již bylo zmíněno v předchozí kapitole, důležitým aspektem je výběr programovacího jazyka. Jako jazyk, který je interpretován na straně serveru, byl vybrán PHP. PHP je populární skriptovací jazyk, který se využívá zejména pro vývoj webových aplikací (PHP, 2001-2023). Na straně klienta byl vybrán značkovací jazyk HTML a skriptovací jazyk Javascript. Jako úložiště bylo vybráno DBMS MySQL. Vývoj aplikace probíhal na lokálním počítači, k čemuž posloužil program XAMPP. Jedná se o balíček aplikací, který je distribuován a umožňuje vývojáři lokálně vytvořit webový server na svém počítači. Hlavními komponenty XAMPP jsou Apache (webový server), MySQL/MariaDB (databáze) a PHP a Perl (skriptovací jazyky) (XAMPP, 2023). Prvním krokem bylo **vytvoření databáze**. Schéma databáze je zřejmé z obr. č. 11 a skládá se z následujících tabulek:

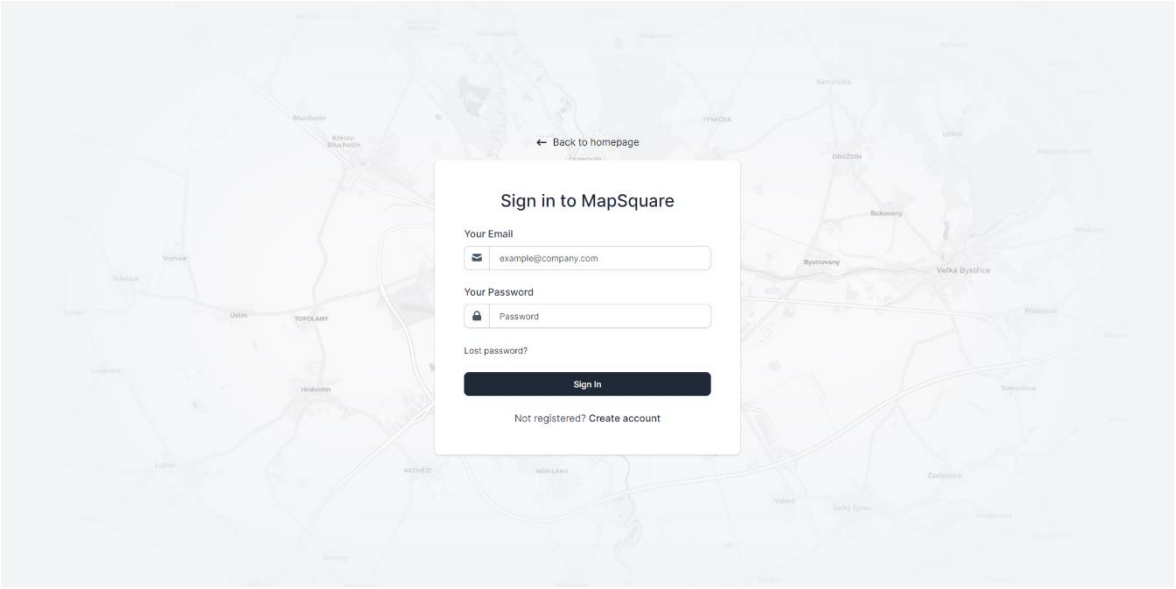
- **activation_code** (aktivační kódy pro aktivaci účtu)
- **data** (základní informace o datech)
- **data_feature** (geometrie a atributy dat)
- **layer** (základní informace o vrstvách)
- **layer_feature** (geometrie a atributy vrstev)
- **lost_password** (obnovovací kódy pro obnovení hesla)
- **map** (mapy)
- **map_layers** (propojovací tabulka map a vrstev)
- **user** (seznam uživatelů)



Obr. 11 Schéma databáze (zdroj vlastní)

Následně byla vybrána šablona, kterou aplikace využívá. Jedná se o šablonu Volt od autora Themesberg, která je distribuována zdarma pod licencí MIT. Šablona využívá známou knihovnu kaskádových stylů Bootstrap ve verzi 5 (Volt - Bootstrap 5 Dashboard Template, 2023). Šablona obsahuje postranní panel po levé straně, ve kterém bylo vytvořeno základní menu s jednotlivými moduly aplikace, jejichž implementace je popsána níže.

Nosným prvkem aplikace je **uživatelský systém**. Aplikace umožňuje vytvořit uživatelský účet, k čemuž je při registraci potřeba e-mailová schránka a heslo. Ke svému účtu si pak uživatel může nahrát data nebo vytvořit vrstvy, ke kterým se může kdykoliv a odkudkoliv dostat. Při registraci je nutné ověřit e-mailovou adresu aktivačním linkem, který je automaticky při registraci odeslán na registrovaný e-mail. Uživateli je při registraci přidělen jednoznačný identifikátor, pomocí kterého jsou v databázi identifikovány jeho data, vrstvy a mapy. Jedná se o řetězec číslic a malých a velkých písmen o délce 32 znaků. Heslo je v databázi ukládáno v šifrované podobě. Do aplikace byla také implementována možnost obnovení hesla při jeho zapomenutí. V tomto případě je na e-mail odeslán odkaz, který má omezenou časovou platnost, a po jehož kliknutí je možné si nastavit nové heslo.



Obr. 12 Přihlašovací stránka (zdroj vlastní)

Dále pokračovala implementace **jednotlivých modulů v pořadí Data – Layers – Analysis – Maps**. V záložce Data se zobrazí seznam všech dat uživatele, který je zobrazen pomocí knihovny Datatables (DataTables | Table plug-in for jQuery, 2023). Jak již bylo nastíněno v předchozí kapitola Analýza a návrh, po kliknutí na záložku Data (obdobně i u Layers a Maps), klient odešle požadavek na data uživatele. Router serveru dále směřuje požadavek na kontroler a databázový dotaz. Výsledkem je seznam dat uživatele, který je vrácen zpět klientovi a který je zobrazí v tabulce uživateli.

Klient odesílá požadavek na server na získání dat (soubor *app/assets/data.js*):

```
$(document).ready(function() {
  getListOfData();
});

function getListOfData() {
  $.ajax({
    method:"POST",
    url: 'server/router.php',
    data: {act: 'data'},
    success: function(data){
      let data_parsed = jQuery.parseJSON (data);
      renderListOfData(data_parsed.data);
      hideLoading();
    },
    error: function(xhr, status, error) {
      alert(xhr.responseText);
    }
  });
}
```

Server zpracuje požadavek a předá ho kontroleru (soubor *app/server/router.php*):

```
$action = $_POST['act'];

switch ($action) {
  case 'data':
    require 'controller/data.php';
    break;
}
```

Kontroler přijme požadavek a spustí danou metodu třídy s příslušným databázovým dotazem (soubory *app/server/controller/data.php* a *app/server/class/data_class.php*):

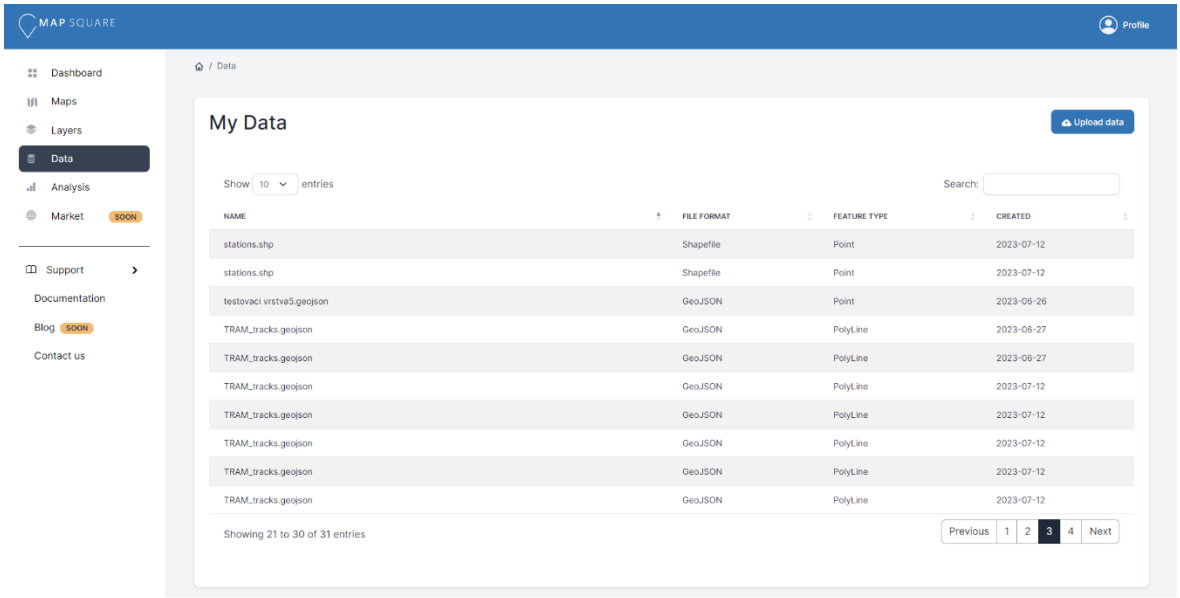
```
$data = new Data($pdo, $user_id);

switch ($action) {
  case 'data':
    echo $data->getAllData();
    break;
}

class Data {
    private $db;
    private $user_id;

    public function __construct($db, $user_id) {
```

```
            $this->db = $db;
            $this->user_id = $user_id;
        }

        //Get All Data
        public function getAllData() {
          $records = $this->db->prepare('SELECT name, data_type,
    feature_type, data_id, cdate FROM data WHERE user_id = :user_id');
          $records->bindParam(':user_id', $this->user_id);
          $records->execute();
          $results = $records->fetchAll(PDO::FETCH_ASSOC);

          $response = json_encode(['status'=> "OK", 'message'=>"Data
    fetched.", 'data'=> $results]);
          return $response;
        }
    }
```

Klient zpracuje data a vykreslí je do tabulky:

```
renderListOfData(data_parsed.data);

function renderListOfData(data) {
  $('#dataTable').DataTable({
    pageLength: 10,
    lengthMenu: [10, 25, 50, 100],
    language: {
        "zeroRecords": "No matching data found",
    },
    data: data,
    columns: [
      {
        …
      { data: "data_type" },
      { data: "feature_type" },
      { data: "cdate" }
    ],
  });
}
```

Uživateli je umožněno nahrávat vlastní data ve formátech GeoJSON a Shapefile, který je komprimován v .zip formátu. Během nahrávání dat probíhá několik kontrol, mimo jiné například kontrola souřadnicového systému (povolen pouze WGS 1984), kontrola validní struktury GeoJSON souboru, počet prvků apod. Pokud systém zjistí chybu v datech, informuje o tom uživatele informační hláškou. V detailu dat je geometrie vykreslena do mapového pole, které dominuje celé stránce. Jako mapovou knihovnu byl vybrán Leaflet, který podporuje velké množství formátů a nativně pracuje velmi dobře s formátem GeoJSON, ve kterém jsou uložena data v databázi (Leaflet, 2023). Při výběru mapové knihovny byla uvažována též knihovna Openlayers, hlavní konkurent pro Leaflet. Autor se však rozhodl pro Leaflet, a to zejména kvůli zkušenost a lepší práci s formátem GeoJSON. Detail dat uživateli slouží hlavně pro vizuální kontrolu geometrie a atributů, dále operovat je možné s vrstvou, kterou uživatel může z dat vytvořit.



Obr. 13 Stránka se seznamem dat (zdroj vlastní)

V dalším kroku byly implementovány **Vrstvy (Layers)**. Zobrazení seznamu vrstev je stejné jako u dat a vrstvu je možné vytvořit jak novou, tak z dat, které uživatel předem nahrál do aplikace. Pokud chce uživatel založit vrstvu, musí vyplnit její název a typ geometrie (bod, linie, polygon). Každý vrstva smí obsahovat pouze jeden typ geometrie. V detailu vrstvy byla implementována editace geometrie za pomoci Leaflet pluginu Leaflet.draw (Leaflet.draw, 2023). Ten umožňuje vytvářet, editovat a mazat geometrii, která se ihned po operaci uloží do databáze. Dále proběhla implementace nastavení znakového klíče. Aplikace podporuje základní nastavení jako barva, průhlednost apod.

Uložení stylu vrstvy v databázi (tabulka *layer*, sloupec *style*):

```
{
  "type": "singleSymbol",
  "style": {
    "fillColor": "#3374b5",
    "weight": 2,
    "opacity": 1,
    "color": "white",
```

```
    "dashArray": 3,
    "fillOpacity": 0.7
  }
}
```

Funkce pro stylování vrstvy (soubor *app/assets/layerView.js*):

```
L.geoJSON(json,
      {
        style: style(feature_type, layer_style)
      }
    ).addTo(map);

function style(feature_type, layer_style) {
  switch(feature_type) {
    case "Point":
      if (layer_style.type == "singleSymbol") {
        return {
          radius: layer_style.style.radius,
          fillColor: layer_style.style.fillColor,
          weight: layer_style.style.weight,
          opacity: layer_style.style.opacity,
          color: layer_style.style.color,
          fillOpacity: layer_style.style.fillOpacity
        };
      }

      break;
    case "PolyLine":
      if (layer_style.type == "singleSymbol") {
        return {
            color: layer_style.style.color,
            weight: layer_style.style.weight,
            opacity: layer_style.style.opacity,
        };
      }
      break;
    case "Polygon":
      if (layer_style.type == "singleSymbol") {
        return {
            fillColor: layer_style.style.fillColor,
            weight: layer_style.style.weight,
            opacity: layer_style.style.opacity,
            color: layer_style.style.color,
            fillOpacity: layer_style.style.fillOpacity
        };
```

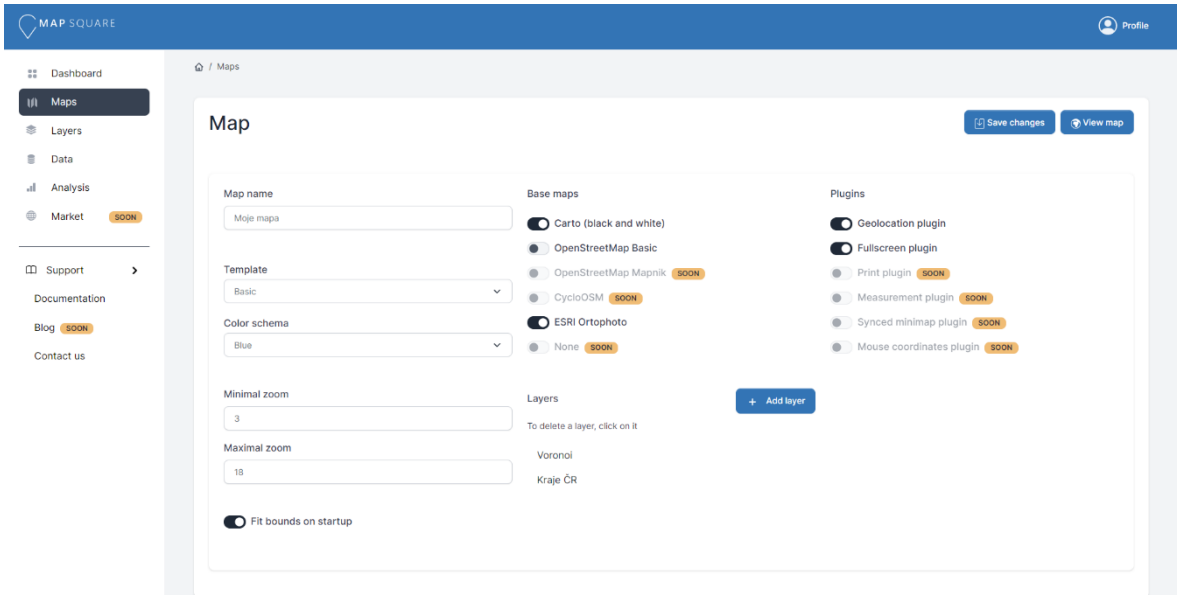
```
      }
      break;
  }
}
```

Klíčový modul **Analysis (Analýza)** umožňuje provést vybrané prostorové operace nad vrstvami. Jedná se o Buffer (obalová zóna), Isochrones (izochrony), Simlify a Smooth polygon (zjednodušení a zjemnění polygonu) a Voronoi polygony. Ke každé operaci náleží vstupní parametry, které uživatel musí zadat, a má také možnost exportovat výsledek do svých vrstev. Modul je navržen tak, aby bylo v budoucnu snadné implementovat další prostorové operace. Operace byly implementovány z knihovny Turf.js, s kterou má autor zkušenosti z bakalářské práce a která poskytuje velké množství funkcí a operací pro zpracování a analýzu geografických dat (Turf.js | Advanced Geospatial Analysis, 2023).

Zdrojový kód pro tvorbu obalové zóny, včetně jejího uložení do databáze (soubor *app/assets/analysis.js*):

```
$('#bufferForm').on('submit', function(e) {
…
    $.ajax({
      method:"POST",
      url: 'server/router.php',
      data: {act: "layerView", layer_id: buffer_source},
      success: function(data){
        const data_parsed = jQuery.parseJSON (data);
        const json = data_parsed.data;

        var buffered = turf.buffer(json, buffer_radius, {units: buffer_unit, steps: buffer_steps});
        addLayerToMap(map, "buffer", "buffer", buffered);

        if (document.getElementById('buffer-export').checked) {
          const layer_type = "analysis";

          $.ajax({
            method:"POST",
            url: 'server/router.php',
            data: {act: "layerAdd", layer_type: layer_type, name: "Buffer", description: "Layer executed in analysis", source: JSON.stringify(buffered)},
            success: function(data){
              …
      }
          });
    hideLoading();
  });
});
```

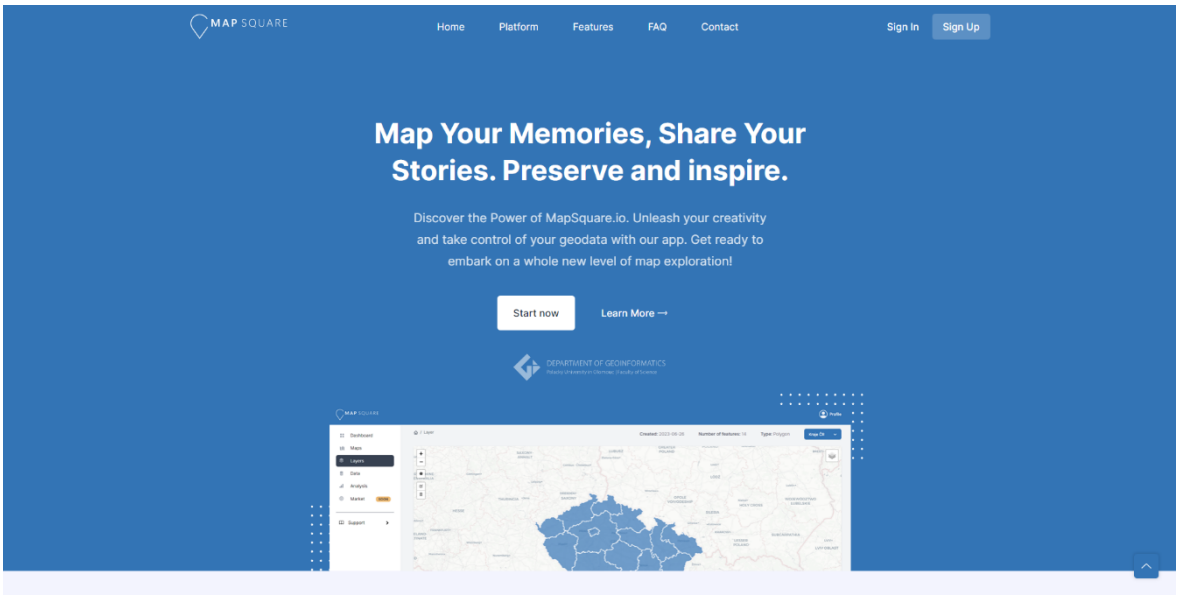
Posledním implementovaným modulem byly **Mapy (Maps)**, sloužící zejména ke sdílení geografických dat. Mapy umožňují základní nastavení v podobě podkladových map, výběru pluginů či změnu jejího názvu. Do mapy uživatel přidává své vlastní vrstvy, které vytvořil. Uživatel po kliknutí na tlačítko View map (Zobrazit mapu) zobrazí mapu, ke které je vygenerována krátká URL v podobě *https://mapsquare.io/s/map?id=ALNUtmTZ*.



Obr. 14 Nastavení mapy (zdroj vlastní)

Nedílnou součástí vývoje webové aplikace je také její vizuální identita. Pro aplikaci byl vybrán název MapSquare a byla registrována doména mapsquare.io. Bylo vytvořeno logo, jako hlavní aplikační barva byla vybrána modrá barva Katedry geoinformatiky UP. Následovala implementace prezentační části, která je koncipována jako jedno-stránková (one-page) a využívá šablonu Play od společnosti UIdeck (Play – Free HTML Startup Landing Page Template, 2023). Šablona využívá, stejně jako předchozí šablona Volt, knihovnu Bootstrap 5 a je distribuována zdarma v komerční/premium verzi. Šablona byla následně upravena pro prezentaci aplikace a jsou v ní základní informace, klíčové vlastnosti, časté dotazy, kontaktní formulář nebo odkazy vedoucí do samotné aplikace. Tímto krokem byla zakončena fáze implementace.

Obr. 15 Logo (zdroj vlastní)



Obr. 16 Prezentační část, úvodní stránka (zdroj vlastní)

5.5 Testování

Testování je klíčovou fází vývoje aplikace a probíhalo během každé iterace. Při implementaci jednotlivých komponent a modulů byly prováděny jednotkové testy, které ověřovaly funkčnost každého prvku. Tyto testy se zaměřovaly na ověření, zda jednotlivé funkce vrací očekávané výsledky a zda jsou při neočekávaných vstupních datech správně zpracovány výjimky. V rámci testování také probíhalo testování uživatelského prostředí, kde byli osloveni vybraní uživatelé, kteří měli za úkol plnit určité úkoly v aplikaci (např. nahrát data, provést analýzu, přihlásit se). Tímto způsobem bylo možné získat zpětnou vazbu a identifikovat některé nedostatky, které mohly být opraveny. V rámci testování uživatelského prostředí je vhodné zmínit, že existuje metoda eye-tracking testování, kterou katedra geoinformatiky UP disponuje. Autor si je této možnosti vědom, avšak vzhledem k rozsahu a charakteru diplomové práce by mohlo testování přesáhnout aktuální a časové limity práce. Autor se tedy rozhodl neprovést eye-tracking testování webové mapové platformy, protože uznal, že tato metoda by vyžadovala rozsáhlý výzkum a analýzu, která by mohla přesáhnout rámec této práce.

Aplikace byla testována na různých zařízeních a prohlížečích. Byly také provedeny testy rychlosti nahrávání a vykreslení dat do mapy. Tyto testy byly provedeny jak na lokálním serveru, tak na webovém hostingu. V konečné fázi bylo také provedeno testování integrity a funkčnosti jednotlivých komponent dohromady v aplikaci. Akceptační testování nebylo z důvodu absence zákazníka/zadavatele provedeno.

5.6 Nasazení

Fáze nasazení, stejně jako předchozí etapy, probíhala v iteracích. Po implementaci a otestovaní jednotlivých modulů byla vždy vytvořena taková verze softwaru, která byla funkční a spustitelná. Kompletní systém byl po poslední iteraci nasazen na webový hosting. Ten má několik omezení a aplikace v současné chvíli na webovém hostingu slouží zejména pro demonstrativní účely pro obhajobu diplomové práce. Pro nasazení aplikace na vlastním hardwaru byl vytvořen podrobný návod instalace, který je k nalezení v příloze.

6 VÝSLEDKY

Tato kapitola shrnuje výsledky praktické části této diplomové práce, které byly dosaženy. Hlavním výsledkem praktické části je webové mapová platforma s názvem MapSquare, která umožňuje uživatelům práci s geografickými daty ve webovém prostředí. Základem aplikace je uživatelský systém, pod vlastní správu si uživatel může nahrát vlastní data ve formátech GeoJSON a Shapefile. Aplikace dále nabízí prohlížení jak geografické, tak atributové složky dat, přičemž je možné z dat vytvořit vrstvu. U vrstvy je možné provést změny v geometrii, jako je přidání, úprava či smazání prvků. Navíc je možné upravit znakový klíč vrstvy. MapSquare dále disponuje nástroji prostorové analýzy z knihovny Turf.js. V neposlední řadě nabízí aplikace uživatelům vlastní mapy, ve kterých mohou vizualizovat svá data a výsledky analýz. Tyto mapy lze snadno zveřejnit pomocí krátké URL adresy.

Během vývoje této aplikace byly uplatněny metody softwarového inženýrství, konkrétně Rational Unified Process. V první řadě bylo analyzováno zadání a specifikovány požadavky na výsledný software. Byly definovány tři základní požadavky a cíle, přičemž bližší specifikace si autor kvůli krátkému zadání práce určil sám. Těmito základními požadavky byly:

1. Vyvinout webovou mapovou platformu
2. Implementovat geoprocessingové nástroje a prostorové operace
3. Implementovat vizualizaci geografických dat v prostředí webové kartografie

Z uvedených požadavků a cílů byl vytvořen diagram základních případů užití, které byly dále detailněji popsány. V následující fázi byla provedena analýza, během níž byla vytvořena SWOT analýza, při které byly definovány příležitosti, hrozby, silné a slabé stránky. Z této analýzy mimo jiné vzešla jako hrozba a slabá stránka silná konkurence, kvůli čemuž byla následně vytvořena analýza konkurence. Do ní byly zahrnuty nejznámější aplikace jako ArcGIS Online, CARTO nebo GIS Cloud, které byly analyzovány a zhodnoceny jejich silné a slabé stránky. Byly také vytvořeny dvě persony se scénáři, které hrají důležitou roli v pochopení a analyzování chování uživatelů. V poslední části analýzy byly vytvořeny diagramy tříd.

V návrhové části byla platforma rozdělena na prezentační a aplikační část a byl rozvržen grafický layout apod. Byla navržena základní architektura systému v podobě MVC a ukládání dat v databázi v podobě JSON, což mimo jiné umožňuje základní prostorové operace i v prostředí databáze. Volbou pro programovací jazyky byly PHP, Javascript a značkovací jazyk HTML.

Během implementace probíhal vývoj jednotlivých komponent způsobem zdola-nahoru implementující základní části kódu jako první. Po vytvoření databáze a šablony byl jako první implementován uživatelský systém. Dále proběhla implementace nahrávání a prohlížení dat, úprava geometrie a znakového klíče vrstvy, analýzy a vytváření či sdílení vlastních map. V posledním kroku byla vytvořena vizuální identita aplikace (logo), upraveny šablony a implementována prezentační část aplikace.

Během předchozích fází, které probíhaly iteračně, docházelo i k testování, které je klíčovou součástí vývoje softwaru. Prováděny byly zejména jednotkové testy, které ověřovaly funkčnost jednotlivých prvků. Dále proběhlo testování uživatelského prostředí, při které byly osloveni vybraní uživatelé, mající za úkol plnit určité úkoly v aplikaci. Aplikace byla také testována na různých zařízeních a webových prohlížečích a byly provedeny testy rychlosti nahrávání dat. Na závěr bylo provedeno testování integrity, při kterém se testovala funkčnost jednotlivých komponent dohromady. Konečné nasazení

proběhlo na základní verzi webového hostingu. Pro zprovoznění aplikace na vlastním stroji byl vytvořen český návod, který uživatele provede krok za krokem instalací. Zdrojový kód aplikace byl zveřejněn na platformě github.com.

7 DISKUZE

Softwarové inženýrství je komplexní obor, kterým se zabývá mnoho knih či akademických prací. Samotnou kapitolou je však vývoj mapových aplikací, která mají svá specifika a kterým není věnován takový prostor. Během řešení diplomové práce se objevilo několik problémů či překážek, které bylo nutné vyřešit.

První problém se objevil již v teoretické části práce. Softwarové inženýrství je totiž natolik komplexní a složitý obor, že nebylo jednoduché vybrat ty části, které jsou relevantní k této diplomové práci. Bylo tedy zapotřebí provést velkou selekci informací. Ukázkou může být třeba kapitola analýza a návrh, ve které byly vybrány pouze některé nástroje analýzy. Je však zřejmé, že nástrojů a způsobů analýzy existuje obrovské množství, které se zkrátka do diplomové práce nevešlo.

Pro praktickou část, ve které byla vyvíjena webová mapová platforma, byla zvolena metodika RUP, která se sice hodí spíše pro větší projekty, ale jedná se o jednu z nejznámějších metodik a navíc byla zmíněna v zadání práce. Jelikož ale RUP počítá s většími projekty a více lidmi, kteří se na vývoji podílí, musely být některé části vývoje potlačeny nebo úplně vynechány. Aplikace byla vyvíjena pouze autorem diplomové práce, a proto byla například vynechána část RUP byznys modelování. Absencí zadavatele/zákazníka také nebylo možné spravovat požadavky. Ty si autor musel částečně vymyslet a upřesnit. Efektivní správu požadavků dokáže obstarat velké množství aplikací, jmenovitě například platforma github.com.

Ze SWOT analýzy a analýzy konkurence vyplynulo, že hrozbou pro aplikaci a její slabou stránkou je silná konkurence. Autor si je plně vědom silné konkurence, kterou však mnohdy tvoří velké společnosti, kterým se aplikace MapSquare nemůže rovnat. Konkurenční aplikace mohou v některých případech vyvíjet i stovky lidí s velkým kapitálem. Hlavní výhodou aplikace MapSquare je tedy její dostupnost zdarma.

Pro implementaci byly jako programovací jazyky zvoleny PHP a Javascript. U této kombinace je vhodné založit aplikaci na MVC architektuře. Úložištěm data se stala databáze, konkrétně MySQL. Ačkoliv by se pro GIS aplikaci hodila více například varianta PostgreSQL, případně PostGIS, konkrétní řešení bylo zvoleno zejména kvůli jejímu širokému rozšíření na běžných webhostingových službách. Geometrie a atributy nahraných dat jsou v databázi uloženy v tabulce *data_feature*, přičemž pokud uživatel chce upravit geometrii, znakový klíč nebo vložit data do vrstvy, je nutné z dat vytvořit vrstvu. Tím se geometrie a atributy zkopírují do tabulky *layer_feature*, čímž může vznikat redundance dat. Autorovým záměrem bylo umožnit uživateli vytvořit z dat několik vrstev s několika různými nastaveními, příp. znakovými klíči, nicméně je si zpětně vědom možnosti redundance dat a vyřešení tohoto problému jiným způsobem.

Během implementace bylo dále potřeba vyřešit několik dalších problémů. Jedním z nich je například bezpečnost. Hesla uživatelů se do databáze ukládají v šifrované podobě. Databázové dotazy jsou prováděny pomocí PDO (PHP Data Objects), které by měly zabránit některým útokům například v podobě SQL Injection. Autor si je ovšem vědom toho, že bezpečnost aplikace je neustále potřeba vyvíjet. Příkladem by mohla být session (sezení) při přihlášení uživatele. To nyní nemá timeout a ukládá do cookies uživatelské ID, přičemž je zde prostor pro vylepšení bezpečnosti. Dále se objevil problém s knihovnou Leaflet.draw umožňující úpravu geometrické složky vrstvy. Ta pracuje efektivně a svižně s menším počtem prvků v mapě, nicméně v případě, že je třeba editovat vrstvu s velkým počtem prvků, dojde ve většině případů k zaseknutí webového prohlížeče. Tento problém se autorovi nepodařilo vyřešit. Dle diagramu tříd byly implementovány třídy *data*, *layer* a *map*. Třída *user* nebyla úspěšně implementována objektově orientovaným

programováním a bude dokončena zřejmě až po odevzdání práce. Stejně tak je to u metody *uploadData* třídy *data*. V návodu instalace aplikace je soubor *Databaze.sql*, který vytvoří potřebné tabulky v databázi. Před odevzdáním musel autor odstranit vazby mezi tabulkami a vytvořit dodatečně u některých tabulek atribut *id*. Ukázalo se totiž, že ačkoliv vše funguje na lokálním stroji v prostředí XAMPP, na webhostingu s provázanými tabulkami a bez atributů *id* nefungovaly některé funkce.

K testování přistupoval autor zejména pomocí jednotkových testů jednotlivých částí kódu, jelikož s nimi má zkušenost. K podrobnějšímu testování, např. metodě eye-tracking, nebylo přistoupeno kvůli rozsahu a účelu práce. Testování zároveň probíhalo na vybraných webových prohlížečích. Následné nasazení finální aplikace proběhlo na webhostingu společnosti Wedos. Ta byla vybrána kvůli předchozím zkušenostem autora, požadovaným vlastnostem a ceně.

Vzhledem k aktivní práci s geografickými daty a jejich častým změnám zatím nebylo přistoupeno k použití mapového serveru jako například Mapserver nebo Geoserver. Nicméně do budoucna by bylo vhodné aplikaci obohatit o mapový server, zejména s ohledem na zobrazení sdílených map. Uživatelům by tak byl na jejich počítač posílán pouze připravený obrázek mapovým serverem, který by prohlížeč následně zobrazil, a nemusela by být uživateli zasílána data z databáze.

Jak již bylo řečeno, volba programovacích jazyků padla na Javascript na straně klienta a PHP na straně serveru. Při implementaci však vznikala i další varianta řešení, kterou bylo Node.js. Jedná se o prostředí, které umožňuje spouštět Javascriptový kód mimo webový prohlížeč. Jeho primární účel tedy patří tvorba serverové části webových aplikací v jazyku Javascript (Node.js). Node.js byl doplněn o React, který obstarává klientskou část, rovněž ve stejném jazyce, a databázi MongoDB. Knihovna React je vyvíjena společností Meta (dříve Facebook) a slouží pro tvorbu uživatelského rozhraní (React). MongoDB se řadí mezi NoSQL databáze a místo tabulek (relací) ukládá data pomocí dokumentů ve formátu JSON (MongoDB, 2023). Toto řešení má několik výhod, ale také několik nevýhod. Tou zásadní nevýhodou je nízké rozšíření Node.js a zatímco PHP je běžně připraveno na všech webhostingových službách, nainstalovat a nasadit Node.js aplikaci je daleko složitější. Výhodami je ale vyšší výkon, vysoká škálovatelnost (může být obslouženo více klientů současně) nebo spuštěný Javascritový kód na straně server. Veškeré výpočty například v knihovně Turf.js, které jsou nyní prováděny na straně klienta, by tak mohly být počítány serverem. Výhodou také může být databáze MongoDB, která nativně ukládá data v JSON a lépe tak spolupracuje s knihovnou Leaflet. Autor vyvíjel kód v PHP i Node.js současně, nicméně se nakonec z důvodu znalostí a zkušeností rozhodl pro jazyk PHP. Znalost a zkušenosti programátora jsou totiž stále rozhodujícím a stěžejním faktorem při výběru programovacího jazyka.

V současném stavu může mít webová mapová platforma využití v nahrazení jednoduchých GIS operací proti desktopovým aplikacím či ve sdílení map s ostatními lidmi. Vývoj platformy byl veden podle metodik softwarového inženýrství, čímž autor ověřil postupy na vývoji webové mapové platformy. V jednotlivých krocích ilustroval vývoj, čímž může pomoct při vývoji dalších webových mapových aplikací.

Zároveň však existuje mnoho námětů na další vylepšení. Příkladem může být propojení účtů s Google účty, čímž se usnadní systém registrace a přihlášení. Dále může být rozšířen seznam přípustných souborových formátů nebo nabídka prostorových a geoprocessingových nástrojů. Jiným námětem na rozšíření je například úprava atributů či větší možnost přizpůsobení znakového klíče (kartogramy atp.). Rozšířit lze aplikaci také o širší možnosti v oblasti nastavení mapy. Zde by mohla být přidána možnost vlastní podkladové mapy, více pluginů či více připravených šablon. Data či vrstvy by také bylo

možné sdílet mezi uživateli, případně doplnit aplikaci o dokumentaci, návody či blog. Doplněn by také mohl být uživatelský systém o nabídku nastavení či gamifikaci – sbírání XP (Experience points) za jednotlivé úkony v aplikaci. Některé odkazy jsou v aplikaci označeny příznakem SOON (brzy), čímž naznačují možný budoucí vývoj. V případě, že by se nejednalo o diplomovou práci, ale o komerční projekt, by bylo možné v aplikaci generovat zisk. V aplikaci by například mohla být zobrazena reklama nebo by uživatelé platili za prémiové funkce či nadlimitní počet provedených prostorových operací, nahraných vrstev, vytvořených map apod. Námětů na pokračování je tedy mnoho a ačkoliv je pro jednu osobu vývoj náročný, autor by v něm rád pokračoval i po odevzdání diplomové práce.

ZÁVĚR

Hlavním cílem diplomové práce bylo analyzovat, navrhnout a ověřit postup vývoje webové mapové platformy s důrazem na principy softwarového inženýrství. Za tímto účelem byla nejdříve provedena rešerše, po které následovala praktická část diplomové práce.

V teoretické části práce autor nejprve definoval klíčové pojmy spojené se softwarovým inženýrstvím a webovou kartografií. Následně se zaměřil na historii softwarového inženýrství a rozebral jednotlivé modely a metodiky používané při softwarovém vývoji. Podrobněji se zabýval metodikou Rational Unified Process, kde popsal jednotlivé kroky softwarového vývoje. Významnou část teoretické části tvořilo zpracování různých aspektů spojených s vývojem webové mapové platformy. Mezi tyto aspekty patřily kartografické, technické, datové, legislativní, komerční, softwarové a uživatelské hledisko. Autor se podrobně věnoval každému z těchto aspektů, představil jejich klíčové prvky a vztahy mezi nimi.

V praktické části bylo provedeno ověření a ilustrace metod softwarového inženýrství na případové studii, která je zaměřena na vývoj webové mapové platformy. Vývoj byl veden dle metodiky Rational Unified Process a v jednotlivých krocích, jako je specifikace požadavků, analýza, návrh, implementace, testování a nasazení byl ověřen a ilustrován vývoj aplikace. Webová mapová platforma s názvem MapSquare umožňuje uživatelům práci s geografickými daty ve webovém prostředí. Uživatelé mohou nahrát vlastní data formátech GeoJSON a Shapefile a prohlížet jak geometrické, tak atributové složky dat. Aplikace umožňuje vytváření a úpravu vrstev včetně změn v geometrii prvků či jejich znakového klíče. K dispozici jsou také nástroje prostorové analýzy z knihovny Turf.js. Uživatelé mohou vytvářet vlastní mapy, které mohou jednoduše sdílet.

Hlavním přínosem diplomové práce je vytvoření webové mapové platformy MapSquare a zasazení softwarového vývoje do kontextu vývoje webových map. Diplomová práce rozšiřuje běžnou problematiku softwarového vývoje o důležité aspekty webové kartografie a další související oblasti. Tím přináší nový pohled na vývoj aplikací zaměřených na práci s geografickými daty v online prostředí. V současné podobě má MapSquare potenciál nahradit jednoduché GIS operace, které se dosud prováděly prostřednictvím desktopových aplikací, a umožňuje také snadné sdílení map s ostatními uživateli. Vývoj platformy byl veden podle metodik softwarového inženýrství, čímž autor ověřil postupy na vývoji webové mapové platformy. Práce ilustruje každý krok vývoje, což poskytuje užitečné poznatky a podněty pro budoucí vývoj dalších webových mapových projektů.

Na závěr byly k diplomové práci vytvořeny webové stránky, které shrnují cíle, metody, výsledky a závěr práce. Součástí diplomové práce bylo také vytvoření posteru ve formátu A2.

POUŽITÁ LITERATURA A INFORMAČNÍ ZDROJE

AL-SAQQA, Samar, Samer SAWALHA a Hiba ABDEL-NABI. Agile Software Development: Methodologies and Trends [online]. THE UNIVERSITY OF JORDAN, AMMAN, JORDAN. 2020 [cit. 2023-08-07]. Dostupné z: doi:10.3991/ijim.v14i11.13269

ArcGIS Online [online]. 2023 [cit. 2023-07-19]. Dostupné z: https://www.esri.com/en-us/arcgis/products/arcgis-online/overview

BOOCH, G., J. RUMBAUGH a I. JACOBSON. Unified modeling language user guide. Massachusetts: Addison-Wesley, 1999. ISBN 0-201-57168-4 [cit. 2023-07-20].

CARTO [online]. 2023 [cit. 2023-07-19]. Dostupné z: https://carto.com/

COLE, Leonardo a Augosto SAMPAIO. RUP Based Analysis and Design with Aspects [online]. Federal University of Pernambuco, 2004 [cit. 2023-07-18].

DataTables | Table plug-in for jQuery [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://datatables.net/

Difference Between UI and UX Design [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://graffersid.com/here-is-the-difference-between-ui-and-ux-design/

ČÁPKA, David. MVC architektura [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

Flowchart Maker & Online Diagram Software [online]. [cit. 2023-08-06]. Dostupné z: https://app.diagrams.net/

GALITZ, Wilbert O. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques [online]. Third Edition. Wiley Publishing, 2007 [cit. 2023-07-11]. ISBN 978-0-470-05342-3.

GIBB, Robert. What is a Web Application. Stackpath [online]. 2016 [cit. 2023-07-19]. Dostupné z: https://blog.stackpath.com/web-application/

GIS Cloud [online]. 2023 [cit. 2023-07-19]. Dostupné z: https://www.giscloud.com/

CHACON, Scott a Ben STRAUB. Pro Git [online]. Second edition. apress, 2014 [cit. 2023-08-07]. Dostupné z: https://git-scm.com/book/en/v2

GROSSMAN, Lukáš. Rational Unified Process [online]. [cit. 2023-07-01]. Dostupné z: https://www.itnetwork.cz/navrh/metodiky/rational-unified-process-a-metodiky-rizeni

HARTSON, Rex. The UX book: process and guidelines for ensuring a quality user experience. Waltham, MA: Morgan Kaufmann, [2012]. ISBN 978-0-12-385241-0 [cit. 2023-07-20].

HUGHEY, Douglas. The Traditional Waterfall Approach [online]. 2009 [cit. 2023-07-01]. Dostupné z: https://www.umsl.edu/~hugheyd/is6840/waterfall.html

KŘENA, Bohuslav a Radek KOČÍ. Úvod do softwarového inženýrství [online]. 2010 [cit. 2023-07-01]. Studijní opora. Vysoké učení technické v Brně, Fakulta informačních technologií.

SUSHANT, Kumar. SDLC - Iterative Model [online]. 2022 [cit. 2023-07-01]. Dostupné z: https://www.scaler.com/topics/software-engineering/iterative-model-in-software-engineering/

KRAAK, Menno-Jan a Allan BROWN. Web Cartography: develompents and prospects. Taylor & Francis, 2001. ISBN 0-7484-0869-X [cit. 2023-07-20].

LEAFLET. Leaflet [online]. [cit. 2023-07-01]. Dostupné z: https://leafletjs.com/

Leaflet.draw [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://github.com/Leaflet/Leaflet.draw

Manifest Agilního vývoje software [online]. 2001 [cit. 2023-08-07]. Dostupné z: https://agilemanifesto.org/iso/cs/manifesto.html

MongoDB: The Developer Data Platform | MongoDB [online]. 2023 [cit. 2023-08-02]. Dostupné z: https://www.mongodb.com/

NÉTEK, R. Frekvence využívání mapových metod na mapových portálech. Bakalářská práce. Univerzita Palackého v Olomouci, Přírodovědecká fakulta, Katedra geoinformatiky. Olomouc, 2008 [cit. 2023-07-20].

NÉTEK, Rostislav. Webová kartografie - specifika tvorby interaktivních map na webu. Olomouc: Univerzita Palackého v Olomouci, 2020. ISBN 978-80-244-5827-4 [cit. 2023-07-20].

Node.js [online]. [cit. 2023-08-02]. Dostupné z: https://nodejs.org/en

OPENLAYERS. Openlayers [online]. [cit. 2023-07-01]. Dostupné z: https://openlayers.org/

PAVLÍČEK, František. Analýza generování rastrových a vektorových mapových dlaždic. Olomouc, 2019. Diplomová práce. Univerzita Palackého, Přírodovědecká fakulta, Katedra geoinformatiky. Vedoucí práce RNDr. Rostislav NÉTEK, Ph.D.

PHP [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://www.php.net/

Play – Free HTML Startup Landing Page Template [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://uideck.com/templates/play-html/

POPELKA, Stanislav. Eye-tracking (nejen) v kognitivní kartografii: praktický průvodce tvorbou a vyhodnocením experimentu. Olomouc: Univerzita Palackého v Olomouci pro katedru geoinformatiky, 2018. ISBN 978-80-244-5313-2.

PRESSMAN, Roger S. Software Engineering: A Practitioner's Approach. Seventh Edition. 2010. ISBN 978–0–07–337597–7.

Pre-implementation analysis. Why is it essential to the success of the project? [online]. 2022 [cit. 2023-08-02]. Dostupné z: https://studiosoftware.com/blog/pre-implementation-analysis-why-is-it-essential-to-the-success-of-the-project/

PRUITT, John S. a Tamara ADLIN. The Persona Lifecycle: Keeping People in Mind Throughout Product Design. Elsevier, 2006. ISBN 978-0-12-566251-2.

React [online]. [cit. 2023-08-02]. Dostupné z: https://react.dev/

Roth, R. E., Sack, C. M., Baldrica-Franklin, G., Chen, Y., Donohue, R., Houtman, L., Prestby, T., Tolochko, R., and Underwood, N.: Introducing Web Mapping: A Workbook for Interactive Cartography and Visualization on the Open Web, Abstr. Int. Cartogr. Assoc., 3, 254, https://doi.org/10.5194/ica-abs-3-254-2021, 2021.

SARSBY, Alan. A Useful Guide to SWOT Analysis [online]. Pansophix Online, 2012 [cit. 2023-07-20]. ISBN 978-1-906460-89-1. Dostupné z: https://www.cii.co.uk/media/6158020/a-useful-guide-to-swot-analysis.pdf

Software Engineering | Spiral Model [online]. 2023 [cit. 2023-07-01]. Dostupné z: https://www.geeksforgeeks.org/software-engineering-spiral-model/

SOMMERVILLE, Ian. Software Engineering. Ninth Edition. 2011. ISBN 0-13-703515-2.

Turf.js | Advanced Geospatial Analysis [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://turfjs.org/

Volt - Bootstrap 5 Dashboard Template [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://themesberg.com/product/admin-dashboard/volt-bootstrap-5-dashboard

VONDRÁK, Ivo. Úvod do softwarového inženýrství [online]. Ostrava, 2002 [cit. 2023-07-01]. VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky.

VONDRÁK, Ivo. METODY BYZNYS MODELOVÁNÍ [online]. Ostrava, 2004 [cit. 2023-07-01]. VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky.

VONDRÁKOVÁ, Alena. Netechnologické aspekty mapové tvorby. Olomouc: Univerzita Palackého v Olomouci pro katedru geoinformatiky, 2014. Terra notitia. ISBN 978-80-244-3970-9 [cit. 2023-07-20].

VOŽENÍLEK, Vít a Jaromír KAŇOK. Metody tematické kartografie: vizualizace prostorových jevů. Olomouc: Univerzita Palackého v Olomouci pro katedru geoinformatiky, 2011. ISBN 9788024427904.

XAMPP [online]. 2023 [cit. 2023-07-20]. Dostupné z: https://www.apachefriends.org/

PŘÍLOHY

SEZNAM PŘÍLOH

Volné přílohy:

Příloha 1 Poster

Elektronické přílohy

Příloha 2 Web

Příloha 3 Webová mapová platforma (včetně návodu na instalaci)