

Webová aplikace pro práci s daty predikce kriminality

Robin Tetour

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Robin Tetour
Osobní číslo: A22036
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Prezenční
Téma práce: Webová aplikace pro práci s daty predikce kriminality
Téma práce anglicky: Web Application for Working with Crime Prediction Data

Zásady pro vypracování

- Navrhněte a popište požadavky na aplikaci.
- Zvolte a popište vhodné technologie pro implementaci.
- Navrženou aplikaci implementujte.
- Vyhodnotte implementované řešení.
- Výsledky vhodně prezentujte.

Seznam doporučené literatury:

1. HUNT, Joel, 2019. *From crime mapping to crime forecasting: The evolution of place-based policing*. Online. U.S. DEPARTMENT OF JUSTICE. NIJ Journal. Dostupné z: <https://www.ojp.gov/pdffiles1/nij/252036.pdf>. [cit. 2024-10-16].
2. JOUBERT, Shayna, 2021. *Predictive Analytics: What It Is & Why It's Important*. Online. Northeastern University. Dostupné z: <https://graduate.northeastern.edu/resources/predictive-analytics/>. [cit. 2024-10-16].
3. ODBOR STATISTIK A ROZVOJE SPOLEČNOSTI, ČESKÝ STATISTICKÝ ÚŘAD, 2023. *Kriminalita v ČR a EU – 2012–2022*. Online. *Kriminalita v ČR a EU*. Roč. 2023, č. 080009-23, s. 109. Dostupné z: <https://csu.gov.cz/docs/107508/3d68511e-a10c-eae7-aeb3-26f323b66e5d/08000923.pdf?version=1.0>. [cit. 2024-10-16].
4. HRUŠKA, Lubor; FOLDYNOVÁ, Ivana; ŠOTKOVSKÝ, Ivan; ZAPLETALOVÁ, Ladislava; VÁCLAVÍK, Tomáš et al., 2015. *Mapy Budoucnosti*. Online. Ministerstvo vnitra České republiky. Dostupné z: <https://www.mvcr.cz/soubor/prevence-dokumenty-mapy-budoucnosti-pdf.aspx>. [cit. 2024-10-16].
5. POLICIE ČR, 2024. *Statistické přehledy kriminality za rok 2023*. Online. Policie České Republiky. Dostupné z: <https://www.policie.cz/clanek/statisticke-prehledy-kriminality-za-rok-2023.aspx>. [cit. 2024-10-16].
6. SCHEINOST, Miroslav; DIBLÍKOVÁ, Simona; FRYDRYCH, Jakub; HÁKOVÁ, Lucie; HOLAS, Jakub et al., 2021. *Analýza trendů kriminality v České republice v roce 2020*. Praha, 262 s. 2021. Institut pro kriminologii a sociální prevenci. ISBN 9788073381967.
7. WOOD, Matt, 2022. *AI Algorithm Predicts Future Crimes One Week in Advance With 90% Accuracy*. Online. Scitechdaily. Dostupné z: <https://scitechdaily.com/ai-algorithm-predicts-future-crimes-one-week-in-advance-with-90-accuracy/>. [cit. 2024-10-16].
8. DAKALBAB, Fatima; ABU TALIB, Manar; ABU WARAGA, Omnia; BOU NASSIF, Ali; ABBAS, Sohail et al., 2022. Artificial intelligence & crime prediction: A systematic literature review: A systematic literature review. *Social Sciences & Humanities Open*. Roč. 6, č. 1, s. 100342. ISSN 2590-2911. Dostupné z: <https://doi.org/10.1016/j.ssaho.2022.100342>.

Vedoucí bakalářské práce:

Ing. Petr Žáček, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **27. října 2024**

Termín odevzdání bakalářské práce: **2. června 2025**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Tato práce pojednává o historii kriminality, dosavadních softwarových řešení a popisuje komponenty, které se používají v praktické části pro návrh a implementaci řešení. Práce dále obsahuje popis požadavků aplikace a popis implementace. Výsledek pracuje hlavně s daty pro město Zlín, ale měl by být schopen pracovat s libovolnou lokalitou.

Klíčová slova: predikce kriminality, webová aplikace, Python, datová analýza

ABSTRACT

This thesis focuses on history of criminality, real world software implementations and describes components used during implementation of app in practical part. Thesis contains description of requirements for the application and description of implementation process. Result of this thesis uses mainly data from city Zlín, but it should be sufficient for any given location.

Keywords: crime prediction, web application, Python, data analysis

Chtěl bych poděkovat panu vedoucímu Ing. Petru Žáčkovi, Ph.D. za jeho podporu, pomoc a poskytnutí volnosti při psaní této práce. Děkuji také Kristýně Hargašové, bez které by tento projekt neexistoval. Hlavně moc děkuji mé mamince, dědečkovi a nevlastní babičce za neutuchající mentální a finanční podporu při studiu na vysoké škole. Bez nich bych neměl příležitost studovat a napsat tuto práci.

OBSAH

ÚVOD	11
I TEORETICKÁ ČÁST	12
1 KRIMINALITA A JEJÍ PREDIKCE	13
1.1 HISTORIE	13
1.2 SOUČASNÉ POUŽITÍ	14
1.3 KONTROVERZE	14
1.4 ZPŮSOBY PREDIKCE	15
1.5 APLIKACE V PRAXI	15
2 SOFTWARE	17
2.1 PYTHON	17
2.2 POSTGRESQL, DATABÁZE A POSTGIS	18
2.3 HTML, CSS A VÝVOJ WEBOVÝCH STRÁNEK	18
2.4 DJANGO FRAMEWORK	19
2.5 JAVASCRIPT A JQUERY	20
2.6 KNIHOVNY PRO PRÁCI S DATY	21
II PRAKTICKÁ ČÁST	22
3 NÁVRH APLIKACE	23
3.1 POPIS APLIKACE	23
3.2 FUNKCIONÁLNÍ POŽADAVKY	23
3.3 NEFUNKCIONÁLNÍ POŽADAVKY	24
4 IMPLEMENTACE DATABÁZE	25
4.1 IMPLEMENTACE TABULKY ZÁKONŮ	26
4.2 IMPLEMENTACE TABULKY LOKACÍ	27
4.3 IMPLEMENTACE TABULKY OSOB	27
4.4 IMPLEMENTACE TABULKY UDÁLOSTÍ	29
4.5 IMPLEMENTACE TABULKY UŽIVATELSKÝCH PROFILŮ	30

5	IMPLEMENTACE BACKENDU A STRUKTURA PROJEKTU.....	32
5.1	MODUL PRO SPRÁVU OSOB	34
5.2	MODUL PRO SPRÁVU UDÁLOSTÍ.....	37
5.3	MODUL PRO ZOBRAZENÍ PREDIKCE.....	43
5.4	MODUL PRO PŘEVOD EXISTUJÍCÍCH DAT	48
5.5	MODUL PRO SPRÁVU UŽIVATELSKÉHO ÚČTU.....	54
6	GRAFICKÝ DESIGN A FRONTEND	55
6.1	HLAVNÍ STRÁNKA	55
6.2	SPRÁVA OSOB	56
6.3	SPRÁVA UDÁLOSTÍ	58
6.4	UŽIVATELSKÝ PROFIL	61
6.5	IMPORT VLASTNÍCH DAT	62
7	SHRNUTÍ IMPLEMENTACE APLIKACE	64
	ZÁVĚR.....	65
	SEZNAM POUŽITÉ LITERATURY	67
	SEZNAM OBRÁZKŮ	71
	SEZNAM VÝPISŮ	72
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	74
	SEZNAM PŘÍLOH.....	75

ÚVOD

Moje motivace pro tuto práci bylo vylepšení a reimplementace mnou vytvořeného praktického projektu s pracovním názvem PredCrime k původní diplomové práci Kristýny Hargašové s názvem Aplikace pro predikci kriminality ve městě Zlín. S touto prací jsme se rovněž zúčastnili 28. ročníku soutěže STOČ 2024 ve Zlíně. Touto vylepšenou verzí bych rovněž chtěl poskytnout policejnímu odvětví kvalitnější a efektivnější systém v boji proti zločinu, kterým by tento projekt jednou mohl být.

Cílem této práce je hlavně navrhnout a poskytnout základní funkční systém, který by splňoval základní požadavky pro správu jednotlivých záznamů, snadnou a efektivní práci s nimi a jednoduchý predikční algoritmus, který by reprezentoval schopnosti aplikace.

V kapitole 1 práce pojednává o historii a problematice zločinu a jeho predikci. Kapitola 2 čtenáři popisuje stručnou historii a poskytuje základní popis komponent použitých v praktické implementaci výsledné aplikace.

V praktické části v kapitole 3 se čtenář dozví popis navrhované aplikace a její funkcionální a nefunkcionální požadavky. Proces implementace databáze aplikace je popsán v kapitole 4. Popis implementace backend části aplikace obsahuje kapitola 5. Poslední kapitola 6 pojednává o návrhu a vytvoření uživatelského rozhraní aplikace a obsahuje názorné ukázky tohoto rozhraní ve formě obrázků.

I TEORETICKÁ ČÁST

1 KRIMINALITA A JEJÍ PREDIKCE

Zločin a kriminalita je zde od počátku lidstva. Důvod, prevence a způsoby jak tyto činy minimalizovat popisuje studium kriminologie. [1]

V následujících podkapitolách je popsána stručná historie kriminologie, způsoby minimalizace, kontroverze přehnané minimalizace a moderní algoritmické metody predikce zločinu.

1.1 Historie

Technologie se stále pohybuje kupředu, proto je pro bezpečnostní složky klíčové nezůstat pozadu a mít přehled o minulých, současných, ale i v tomto případě budoucích kriminálních činech nebo přestupcích. Predikce kriminality avšak není moderní koncept.

Termín kriminologie byl zaveden italským psychiatrem Cesarem Lombrosoem. Zastával názor, že kriminálníci se kriminálníky rodí. Věřil, že kriminálník se dá poznat podle vzhledu, tvaru lebky, barvy pleti, stavby těla nebo i mentálních poruch. Jeho teorie daly vzniku institutu Scuola di Polizia Scientifica již v roce 1907, který se věnoval profilování a predikce kriminality. [2]



Obrázek 1.1 Cesare Lombroso (Zdroj: [1])

I přes tyto dnes již kontroverzní teorie a metody je Lombroso považován jako průkopník tohoto oboru, právě kvůli jeho snaze najít vědecké vysvětlení pro důvody vzniku samotného zločinu. Jeho výzkum měl značný vliv na další vývoj snahy o předpověď zločinu a vývoj kriminologie. [1]

1.2 Současné použití

Dnes se samozřejmě dají použít mnohem důmyslnější a spolehlivější metody pro předpovědi. V této podkapitole jsou uvedeny nejznámější a zároveň nejúspěšnější příklady.

Do predikce kriminality šlo spoustu výzkumu a financí. Jeden z výzkumů poukázal na analýzu příchozích hovorů Minneapolisské policie od roku 1985 až 1986, která odkryla, že 50% hovorů nahlásilo 3% všech možných míst. Toto vzbudilo otázku, jestli není lepší přemýšlet nad místem činu, než nad pachatelem činu. [3]

V roce 2011 policejní stanice v Santa Cruz začala testovat systém PredPol, který na základě minulých a současných dat dokáže policisty na obchůzce navádět na takzvané hotspoty, čtvercovou oblast, které tento systém vyhodnotí jako místo s vysokou pravděpodobností výskytu trestního činu, či podobné události. Bylo ale jisté, že nenahradí intuici zkušených policistů a používá se jako prevence ve smyslu seznamu lokací, kterými policista může projít na obchůzce pokud se nevěnuje přednějším věcem. [4]

V dnešní době se predikce kriminality uplatňuje společně s GIS softwarem. GIS je software, který nám umožňuje pracovat s prostorovými daty (v tomto případě, nejlepším příkladem jsou mapy). Stejně tak jako v Santa Cruz, tato forma softwaru se spoléhá na vizualizaci míst na mapách ve formě bodů, linií nebo polygonů (heatmap). Lokalizace těchto míst je možná podle GPS, nebo konkrétních adres. Jelikož jde o GIS software, můžeme rovněž mapovat polohy konkrétních hlídek i provést jejich plánování. [5]

1.3 Kontroverze

V dnešní době čím dál tím více stává relevantním strach z algokracie. Termín algokracie znamená přímo kontrolu a správu státu za pomoci algoritmů. Do této problematiky spadá právě i prediktivní analýza kriminálních činů. [6]

Mezi nejintenzivnější příklady algokracie by se mohla řadit Čína a její obrovská síť kamer a systému rozpoznání obličejů. Bohužel se tato technologie spíše používá pro zachování tamějšího totalitního režimu a umlčení a potlačení protestů proti vládě. [7]

1.4 Způsoby predikce

Způsobů jakými provádět prediktivní analýzu je nespočet, ale nejčastější příklady jsou lineární regrese, neuronové sítě, odhad optima a různé modely strojového učení. Clustering modely strojového učení jsou velmi vhodné pro predikce. Tyto modely fungují na principu hledání skupin dat které si jsou podobné, nebo obsahují podobné kvality. Gaussovský, Density-Based Scan nebo hierarchické seskupování je jen několik příkladů takových modelů. [8]

Predikční algoritmy nejčastěji používané a diskutované konkrétně k predikci zločinu jsou random forest a naive Bayes, což jsou zároveň příklady modelů strojového učení s učitelem. Rovněž se používají různé hybridní algoritmy, nebo více modelů strojového učení za sebou. [9]

Příklad konkrétního algoritmu v praxi by mohl být již zmíněný systém PredPol testovaný policií v Santa Cruz. Pro predikci hotspotů se používá takzvaný earthquake aftershock algorithm. Jedná se o algoritmus, který publikoval Dr. George Mohler na tamější univerzitě Santa Clara ve své práci roku 2010. [4]

Další konkrétní algoritmus vytvořený na univerzitě v Chicagu, který uvádí až 90% úspěšnost bere v potaz topologii měst. Například autobusové linky, vlaky, chodníky a cesty. Tento model dále používá souřadnice a čas zločinů, hledá vzorce a podle nich se snaží předpovědět budoucí zločiny. [10]

1.5 Aplikace v praxi

Mezi příklady konkrétních softwarových implementací pro predikci zločinu se řadí software HunchLab od společnosti Azavea. HunchLab používá techniku strojového učení pro predikci za použití množství proměnných, mezi které patří dopad a výskyt kriminality v oblasti, ale i typy zločinu a jejich časy. Modely se dají i přizpůsobit podle upřesňování znaků daných zločinů. [11]



Obrázek 1.2 Jeff Brantingham - tvůrce softwaru PredPol (Zdroj: [12])

Dalším příkladem by mohl být PredPol. Software, který vyvinul Jeff Brantingham na Kalifornské univerzitě v Los Angeles jako výzkumný projekt. Ve své době dokázal předpovědět dvakrát více zločinů než jiný systém a měřit se i s živými analytiky. Proto na začátku roku 2013 začala policie ve městě Seattle používat PredPol pro predikci majetkových přestupků a zločinů. Později toho roku začala používat software k predikci násilí se zbraněmi. [12]

PRECOBS je software pro predikci zločinu, který byl testovaný v německém Mnichově roku 2014 až 2015. Systém používá data mladší než jeden týden a generuje tak krátkodobé předpovědi. Konkrétně využívá pro predikci místo, čas, způsobu spáchání zločinu, zeměpisnými údaji a odcizeném majetku (v případě majetkových zločinů). Při testování klesl výskyt vloupání (pro které systém byl testován) o 29% (v některých oblastech až 42%) a stoupl počet zatčených pachatelů. [5]

2 SOFTWARE

Základní poučka o softwaru říká, že jde o sadu instrukcí, které počítače následují, aby provedli danou úlohu. Toto popisuje veškerý software, ať už ovladače pevných zařízení, operační systémy, jejich komponenty, ale i software a knihovny, které jsou používány pro vývoj jiného softwaru. [13]

V následujících podkapitolách se čtenář seznámí se softwarem, který je následně použit v teoretické části této práce k implementaci aplikace. Dočte se o programovacím jazyku Python, způsobech ukládání dat, statickém, ale i dynamickém webovém vývoji a o knihovnách pro práci s daty.

2.1 Python

Python je vysokoúrovňový objektivě orientovaný programovací jazyk. Je velice snadno čitelný a tak se i snadno učí. Programy napsané v tomto jazyce se spouští pomocí interpreteru, který samotný je napsaný v Pythonu, takže programování neobsahuje krok kompilace programu a tím ve výsledku zjednodušuje manuální testování během programování. [14]

Program napsaný v Pythonu je zároveň modulem a ten samotný může obsahovat jiné moduly a balíčky. Tímto přístupem jazyk podporuje znovupoužitelnost a modularitu kódu. [14]

Python vytvořil Guido van Rossum 2.1 v roce 1989, aby si usnadnil programování shell skriptů v práci. Pracoval na vývoji operačního systému Amoeba a programovacím jazyku ABC v národním institutu Centrum Wiskunde & Informatica v Nizozemsku. [15]



Obrázek 2.1 Guido van Rossum - tvůrce programovacího jazyka Python (Zdroj: [16])

Jazyk pojmenoval podle jeho oblíbeného britského komediálního seriálu - Monty Python. V únoru 1991 po konzultaci se svým manažerem publikoval zdrojový kód celého Python interpreteru. [15]

2.2 PostgreSQL, databáze a PostGIS

PostgreSQL je open source objektově-relační databázový systém, který se vyvinul z projektu POSTGRES Kalifornské Univerzity v Berkeley. Podporuje primitivní, strukturované, ale i vlastní datové typy. Mezi typy, které můžeme uchovávat v databázi patří celá i desetinná čísla, text, pole hodnot, JSON nebo XML soubory, ale i již zmíněné vlastní datové typy. [17]

PostGIS je rozšíření pro PostgreSQL databázový systém, který přidává podporu pro tři zásadní funkce. Jedná se o prostorové typy, indexy a funkce. Těmito třemi pilíři docílíme plné podpory pro uchovávání prostorových dat v PostgreSQL databázích.

Nové prostorové typy, které můžeme díky tomuto rozšíření ukládat jsou body, úsečky a mnohoúhelníky. Pro manipulaci s těmito novými daty doplňují nové funkce, které PostGIS rovněž přidává. Jedná se o funkce, které nám umožňují porovnávat, převádět, ale i generovat tyto prostorová data. Tyto nové typy a funkce dělají z PostGIS skvělý základ pro jakýkoliv GIS software. [18]

2.3 HTML, CSS a vývoj webových stránek

První webová stránka vznikla už v roce 1990. Vytvořil ji Tim Berners Lee a původně byla hostovaná na jeho stolním počítači v laboratořích CERN. Projekt WWW popsal společně s hyper-text projektem v jeho dvou předešlých pracích z let 1989 a 1990. Tyto projekty popisovaly strukturu sítě, která obsahovala dokumenty hypertextu. Lee společně se spuštěním jeho první webové stránky poskytl první webový prohlížeč a software pro webový server. To vše v roce 1991 poskytl i veřejnosti společně s knihovnou pro vývojáře. Projekt se postupně rozrůstal a později Lee založil konsorcium W3C, které od té doby dohlíží na vývoj webu. [19]

HTML je jazyk pro vývoj webových stránek. Jedná se konkrétně o standard jazyka, který každý prohlížeč musí přesně podporovat, aby byl schopen zobrazovat webové stránky a jejich obsah. Tento jazyk byl popsán jako součást projektu WWW. Tento standard byl původně vydáván jako jednotlivé verze jazyka HTML. Společnosti Apple, Mozilla a Opera se sjednotili v skupinu WHATWG, která se udržovala HTML specifikaci. V roce 2019 se ale skupiny W3C a WHATWG dohodli, že se bude udržovat jednotná verze specifikace HTML. Tato verze je konstantě aktualizovaná a tím se zamezuje problémům, které nastávají s jednotlivým vydáváním statických verzí. [20]

CSS je stejně jako HTML specifikace jazyka. Tento jazyk ale slouží pro úpravu vizuální prezentace stránek a je tak prakticky neoddělitelné od jazyka HTML. Dnes jazyk CSS hraje prim v designu webových stránek. [21]

Bootstrap je knihovna používaná pro vývoj responzivních webových stránek. Jedná se o předvytvořené CSS styly, které může vývojář využít bez potřeby psát a udržovat vlastní styl. Poskytuje tak jednotnou cestu, jak designovat responzivní a uživatelsky přívětivé webové stránky poskytnutím jednotlivých komponent pro různé účely. Knihovna rovněž poskytuje kompatibilitu s JavaScriptem. Vývojář tak může využít tento programovací jazyk pro přímou manipulaci s Bootstrap komponenty. [22]

2.4 Django Framework

Django je open-source webový framework pro rychlý vývoj dynamických webových stránek v jazyce Python. Má velkou aktivní komunitu a skvělou dokumentaci. Zjednodušuje vývoj, jelikož obsahuje základní důležité funkce pro každou stránku a aplikaci, takže se vývojář může soustředit na vývoj hlavních částí aplikace. [23]

Vytvořili jej Adrian Holovaty a Simon Willison jako kolekci komponentů, aby předešli opakované implementaci základních funkcí pro každou novou stránku, na které pracovali. Původně byl vytvořen pro web novin Lawrence Journal-World ve městě Lawrence v Kansasu. [24]

Mezi funkce, které Django poskytuje patří zabudované zabezpečení proti CSRF, SQL injection a ClickJacking útokům. Disponuje předpřipraveným systémem pro autorizaci a autentizaci uživatelů, vlastní rozhraní pro administrátory, ORM, neboli Object Relation Mapper, který napíše SQL za vývojáře pomocí Python kódu a mnohem více funkcí, které usnadňují vývoj webových aplikací. [25]



Obrázek 2.2 Logo frameworku Django (Zdroj: [26])

Django Template je jedna z hlavních částí tohoto frameworku. Jedná se o HTML, ale se speciální syntaxí, která pomáhá vývojářům spravovat dynamický obsah jejich stránek. Díky tomuto jazyku nemusí programátoři používat HTML v jejich Python kódu. Odděluje backend od frontendu a umí vygenerovat plně statické stránky. Díky tomuto se kód snáze čte a je znovupoužitelný. [27]

2.5 JavaScript a jQuery

Jazyk JavaScript byl vytvořen v květnu 1995 během dvou týdnů Brendanem Eichem. Byl původně zamýšlen jako kompetitor Java appletů firmy Sun Microsystems. JavaScript byl vytvořen ve firmě Netscape jako skriptovací jazyk pro jejich prohlížeč Netscape Navigator. Eich během deseti dní před vydáním alpha verze Netscape Navigator 2.0 vytvořil tak první implementaci JavaScriptu. [28]



Obrázek 2.3 Brendan Eich (Zdroj: [29])

JavaScript je multiplatformní objektově orientovaný skriptovací jazyk, který vývojářům webových stránek umožňuje implementovat interaktivní prvky. V prohlížeči slouží pro ovládání prvků jako tlačítek, formulářů, ale i poskytuje logiku pro funkcionalitu jako navigace na stránkách a další. [30]

Knihovna jQuery je používána pro zjednodušení psaní kódu při vývoji stránek, jelikož poskytuje abstrakce na základním JavaScriptu. [31] Jedná se o nejpoužívanější knihovnu JavaScriptu na světě. První verzi knihovny vytvořil John Resig v roce 2005. Tehdy knihovna sloužila pro zjednodušení vývoje skriptů pro všechny prohlížeče. Dnes je její hlavní výhoda v jejích abstrakcích, které poskytují vývojářům snazší cestu jak psát kód a vybírat elementy. Zároveň ale v poslední době ubývá na jejím používání. [32]

2.6 Knihovny pro práci s daty

Jedna z nejdůležitějších knihoven pro práci s daty v jazyce Python je bezpochyby Pandas. Jde o balíček pro manipulaci s daty ve formě tabulek. Pandas funguje se spousty dalších balíčků (jako je NumPy, scikit-learn a další) a vytváří tak ekosystém a standard pro práci s daty v jazyku Python. [33]

Hlavní komponenty v knihovně Pandas jsou série (Series) a datové rámce (DataFrames). Datové rámce jsou tabulky, které se skládají ze sloupců (sérií). Pandas umí bez problémů převádět tyto virtuální tabulky mezi známými formáty jako CSV, JSON, ale i SQL tabulky. Lze rovněž velice snadno provádět různé úkony (ať už NumPy funkce, či vlastní) na jednotlivých sloupcích, řádcích, ale i samotných buňkách. [34]

Již zmíněná knihovna NumPy slouží primárně pro práci s listy. Implementuje rovněž matematické funkce, které s těmito listy pracují mnohem rychleji, než při použití klasických Python listů a cyklů. Vzhledem k tomu, že NumPy vytváří abstrakci, implementace při použití této knihovny vyžadují méně řádků kódu. Využívá ho nespočet dalších Python knihoven. [35]

Pokročilejší knihovna, která rovněž využívá knihoven NumPy a Pandas je scikit-learn. Jedná se o knihovnu implementující moduly pro vytváření algoritmů strojového učení, umělé inteligence a statistického modelování. Knihovna podporuje učení modelů s učitelem i bez učitele, lze s její pomocí i formátovat trénovací data pro modely a spousty dalších věcí, které velmi napomáhají pokročilým i začátečníkům vytvářet solidní modely pro specifické účely. [36]

II PRAKTICKÁ ČÁST

3 NÁVRH APLIKACE

Před začátkem implementace je potřeba mít představu o tom, jak by měla aplikace ve výsledku vypadat a co by uživateli měla umožnit. Tyto požadavky by měly být stěžejní při výběru knihoven, služeb a dalších komponent použité při implementaci.

3.1 Popis aplikace

Hlavní myšlenka aplikace je poskytnout přehledné, spolehlivé a robustní rozhraní pro správu událostí (přestupků a zločinů), osob (podezřelých, či přímo zodpovědných za události) a míst, kde se tyto události odehrály. Přes toto rozhraní například policejní hlídka vyplní hlášení o události, vyplní osobu zodpovědnou a další podrobnosti a tyto data se přímo projeví ve výstupu predikce, která bude ve formě jednoduché heatmapy na hlavní stránce aplikace.

3.2 Funkcionální požadavky

Mezi nejdůležitější a hlavní funkcionální požadavek této aplikace je schopnost predikovat zločin a poskytnout jednoduchý způsob pro vývojáře jak upravit, či změnit predikční model. Výstup tohoto modelu bude vizualizace predikce ve formě heatmapy na mapě přímo v uživatelském rozhraní. Tato vizualizace by se aktualizovala podle aktuálních událostních dat dostupných v databázi aplikace. Tato predikce bude pracovat s určitým data-setem, který do databáze aplikace budou uživatelé odesílat průběžně přes jednoduché formuláře s kompletní client-side i server-side validací. Příkladem je policista vypisující hlášení o události. Zadá adresu, nebo použije aktuální polohu. Může rovněž vyplnit podrobnosti o zločinu i přestupku, nebo i vytvořit či doplnit o profil zodpovědné osoby.

Tento predikční algoritmus by měl být jednoduchý, snadno čitelný a měl dostatečně předvést schopnost aplikace nakládat s dostupnými daty a provádět s těmito daty danou predikci. Zároveň není možné implementovat model tak univerzální, aby postačil pro potřeby všech možných instancí a prostředí. Právě proto je potřeba poskytnout alespoň solidní příklad, tak aby předvedl schopnosti aplikace a reprezentoval projekt jako solidní a použitelný základ pro policejní systém.

Je nutné, aby aplikace podporovala responzivní grafické rozhraní, tak aby bylo možné ji používat na velkých i malých obrazovkách bez potíží. To je nutnost pro použití na mobilních zařízeních v terénu, ale i na stolních, či přenosných počítačích.

Samozřejmě administrátor instance aplikace by měl být schopný přes jednoduché rozhraní převést již existující data-set do databáze aplikace, v případě, že by například stanice policie daného města rozhodla migrovat čistě na tuto aplikaci z jiného systému, nebo jiných statistických dat. Konkrétní příklad by mohly být excelové tabulky zločinů a přestupků předešlých let. Poskytnutí přehledného uživatelského rozhraní pro provedení této migrace pro administrátory aplikace je důležité pro správné prvotní nastavení aplikace.

Vzhledem k tomu, že se v praxi jedná o systém, jehož instance budou spravovány státní institucí (policejními služebnami), bude potřeba dobrého zabezpečení, uživatelských účtů a profilů jednotlivých uživatelů, aby se zajistila transparentnost akcí uživatelů a minimalizoval risk zneužití systému.

Registrace uživatelů z tohoto důvodu nedává smysl a měla by být prováděna přes administrátora, či rozšíření tohoto systému, které by automaticky registrovalo nové policisty jako uživatele a mazala uživatelské účty, jehož vlastníci již pro instituci nepracují. Tato funkcionality je ale nad rámec implementace a může se lišit dle prostředí, kde je aplikace provozována.

3.3 Nefunkcionální požadavky

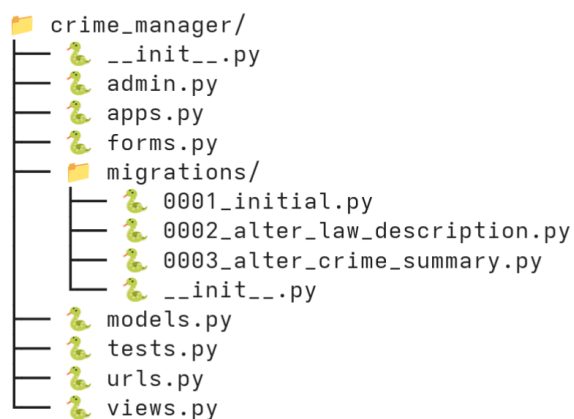
Mezi nefunkcionální požadavky se samozřejmě řadí škálovatelnost aplikace, to je obzvláště důležité pro tento sektor, kde množství uchovávaných záznamů jen narůstá. Z tohoto důvodu je potřeba dbát na správnou implementaci, aby nedocházelo ke zpomalování aplikace s rostoucím počtem těchto záznamů. Tento problém by se dal v budoucnu ještě dále řešit selektivním odstraňováním až příliš starých dat, která už nejsou relevantní. Implementace tvorby cache paměti je rovněž potřeba pro rychlou odezvu systému.

Dokumentace a čitelnost kódu je také důležitá součást tohoto projektu, hlavně z důvodu, že každá instance může vyžadovat uchovávání jiných dat a provádět predikci zločinu jiným způsobem. Nemělo by být pro vývojáře příliš složité změnit dané části aplikace. Tudíž by měla poskytovat stabilní, solidní celek a základ pro snadné rozšiřování a přidávání dalších funkcí.

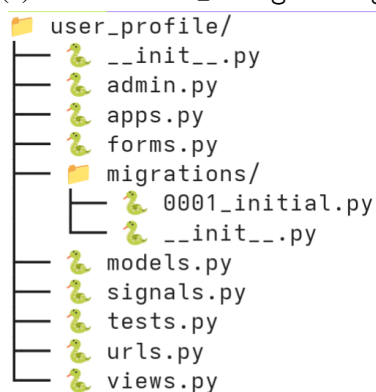
Zabezpečení aplikace je kritická část, na kterou by nemělo být zapomínáno, zvláště pokud se jedná o citlivá data uživatelů, nebo i podrobnosti potenciálně utajených zločinů. Tyto znalosti se dají zneužít pokud se do systému dostane útočník, ať už s jakýmkoliv úmysly. Funkce aplikace by měly být napsané tak, aby se minimalizovalo takové riziko narušení.

4 IMPLEMENTACE DATABÁZE

Samotná implementace databáze záleží na frameworku Django (popsán v kapitole 2.4). Konkrétně není potřeba ručně psát SQL kód, ale knihovna nám pomocí definovaných modelů vytvoří takzvané migrace, které se následně aplikují na zvolenou instanci DBMS. Konkrétní DBMS, který je implementací používán je PostgreSQL s rozšířením PostGIS (z kapitoly 2.2).



(a) Modul `crime_manager` s `migrations` složkou



(b) Modul `user_profile` s `migrations` složkou

Obrázek 4.1 Struktury souborových systémů

V následujících kapitolách je popsána implementace jednotlivých tabulek databáze pomocí Django ORM. Definice těchto tabulek je umístěna v daných modulech v souboru `models.py`. Při spuštění pomocného skriptu `manage.py` s argumentem `makemigrations`. Django tyto soubory najde a vytvoří migrace podle změn v souborech (4.1b a 4.1a). Migrace můžeme ručně aplikovat pomocí již zmíněného `manage.py` souboru a argumentu `migrate`. Tento příkaz použije databázi námi nastavenou v Django `settings.py` souboru a modifikuje tabulky databáze v DBMS podle definovaných modelů.

4.1 Implementace tabulky zákonů

Díky Django ORM nemusíme psát SQL dotazy pro tvorbu tabulek databáze. Python třídy, které obsahují atributy dědící z abstraktní třídy `Field` nám vytváření tabulek databáze velice usnadňují.

Jako první vytvoříme společnou třídu pro všechny zákony, které budou spjaty s událostmi. Takto budeme vždy vědět, který zákon byl porušen u daného hlášení.

Využijeme české notace zákonů, například *Zákon č. 261/2016 §8 odst. 1 písm. a bod. 1.* označuje krádež. Pomocí ORM je výsledná implementace tabulky pro zákon následující 4.1.

```
1 only_letters = RegexValidator(r"^[a-z]+$", _("Only lowercase letters are
  ↳ allowed."))
2 class Law(models.Model):
3     year = models.PositiveIntegerField(verbose_name=_("year"))
4     number = models.PositiveIntegerField(verbose_name=_("number"))
5     paragraph = models.PositiveIntegerField(verbose_name=_("paragraph"))
6     article = models.PositiveIntegerField(verbose_name=_("article"))
7     letter = models.CharField(verbose_name=_("letter"), max_length=1,
  ↳ validators=[only_letters])
8     item = models.PositiveIntegerField(verbose_name=_("item"),
  ↳ blank=True, null=True)
9     description = models.CharField(max_length=200,
  ↳ verbose_name=_("description"))
10    wording = models.TextField(blank=True, null=True, max_length=500,
  ↳ verbose_name=_("wording"))
11    class Meta:
12        verbose_name = _("law")
13        verbose_name_plural = _("laws")
14        constraints = [models.UniqueConstraint(fields=["year", "number",
  ↳ "paragraph", "article", "letter", "item"], name="unique
  ↳ law")]
```

Výpis 4.1 Definice modelu Law (zdroj: vlastní)

Dodatečné atributy `description` a `wording` slouží k podrobnějšímu popisu zákona a jeho znění. Rozhodl jsem se využít i `UniqueConstraint` omezení, aby se zamezilo duplicitním zákonům v databázi. 4.1

Zároveň ve všech modelech využívám program `gettext`, který slouží k implementaci lokalizace a překladu aplikace do jiných jazyků. V tomto případě primárně vyvíjím aplikaci v angličtině a přidávám překlady do českého jazyka.

4.2 Implementace tabulky lokací

Tabulka všech umístění využívá nastavby GeoDjango pro uložení souřadnic ve formě bodu pomocí rozšíření PostGIS pro PostgreSQL, který GeoDjango vyžaduje. Konkrétně se ukládá do databáze objekt typu Point. Rovněž model obsahuje popis lokace description, která je zamýšlená jako upřesnění umístění (například v druhém patře, nebo ve sklepě). Pole adresy je prozatím zamýšleno spíše pro převod z textových adres na číselné souřadnice, v případě importu vlastního data-setu. 4.2

```
1 class Location(models.Model):
2     coordinates = models.PointField(verbose_name=_("coordinates"))
3     address = models.CharField(max_length=70, null=True, blank=True,
4     ↪ verbose_name=_("address"))
5     description = models.CharField(max_length=50,
6     ↪ verbose_name=_("description"))
```

Výpis 4.2 Definice modelu Location (zdroj: vlastní)

4.3 Implementace tabulky osob

Důležitá data jsou rovněž záznamy podezřelých osob. Tabulka bude uchovávat data jako je křestní a prostřední jméno, příjmení, datum narození, textový popis osoby a rodné číslo. Jako další možná data, která by bylo potřeba ukládat je například fotografie člověka, jako příklad uvedena pod proměnnou photo. 4.4

```
1 czech_id_regex = r"^[0-9]{6}[/]?[0-9]{3,4}$"
2 person_number = RegexValidator(czech_id_regex, _("This is not a valid
3 ↪ identification number."))
```

Výpis 4.3 Validační objekt pro rodná čísla (Zdroj: vlastní)

Validace pro rok narození je implementována ve funkci clean, která uživateli zabrání vytvoření záznamu o osobě s datem narození v budoucnosti. Dále se využívá validace pomocí regulárního výrazu pro ověření zda rodné číslo bylo zadáno v platném tvaru a vrácení chybové hlášky pokud nebylo. 4.4

```

1 class Person(models.Model):
2     identification_number = models.CharField(unique=True,
3     ↪ validators=[person_number], verbose_name=_("identification
4     ↪ number"))
5     name = models.CharField(max_length=40, verbose_name=_("first name"))
6     middle_name = models.CharField(max_length=40, blank=True, null=True,
7     ↪ verbose_name=_("middle name"))
8     surname = models.CharField(max_length=40, verbose_name=_("last
9     ↪ name"))
10    birth_date = models.DateField(verbose_name=_("date of birth"))
11    description = models.TextField(max_length=1000, blank=True,
12    ↪ null=True, verbose_name=_("description"))
13    photo = models.ImageField(upload_to="Person_Pictures", null=True,
14    ↪ blank=True, verbose_name=_("photo"))
15    class Meta:
16        verbose_name = _("person")
17        verbose_name_plural = _("people")
18    def __str__(self):
19        return f"{self.fullname} (* {self.birthdate_format})"
20    def clean(self):
21        if self.birth_date > datetime.date.today():
22            raise ValidationError(_("Birth date cannot be in the
23            ↪ future!"))

```

Výpis 4.4 definice modelu Person (zdroj: vlastní)

Django nám rovněž umožňuje implementovat vlastnosti modelů v podobě Python funkcí. Tímto způsobem můžeme vytvořit vlastnosti, které nám vypočítají přesný věk osoby podle data narození, vrátí nám z databáze zločiny a přestupky, které tento člověk spáchal nebo provede jednoduché textové formátování data narození nebo jména. 4.5

```

1 @property
2 def fullname(self):
3     return f"{self.name} {self.middle_name or ''} {self.surname}"
4 @property
5 def age(self):
6     return relativedelta(datetime.date.today(), self.birth_date).years
7 @property
8 def birthdate_format(self):
9     return formats.date_format(self.birth_date, "d. E Y")
10 @property
11 def crimes(self):
12     return Crime.objects.filter(culprit=self)

```

Výpis 4.5 dodatečné vlastnosti modelu Person (zdroj: vlastní)

4.4 Implementace tabulky událostí

Událostmi se rozumí zločiny či přestupky, aplikace nerozeznává rozdíly mezi závažnostmi událostí. Může to být později implementováno pomocí tabulky zákonů, které se pojí s tabulkou událostí. 4.6

```
1 class Crime(models.Model):
2     summary = models.CharField(max_length=200, verbose_name=_("incident
    ↳ description"))
3     description = models.TextField(blank=True, null=True,
    ↳ max_length=1000, verbose_name=_("description"))
4     law = models.ForeignKey(Law, on_delete=models.PROTECT,
    ↳ verbose_name=_("law"))
5     location = models.ForeignKey(Location, on_delete=models.PROTECT,
    ↳ verbose_name=_("location"))
6     culprit = models.ManyToManyField(to=Person, blank=True,
    ↳ verbose_name=_("culprit"))
7     date = models.DateTimeField(default=timezone.now,
    ↳ verbose_name=_("date"))
8     class Meta:
9         verbose_name = _("crime")
10        verbose_name_plural = _("crimes")
11    def __str__(self):
12        return f"{self.summary} - {self.dateformat}"
13    @property
14    def dateformat(self):
15        return formats.date_format(self.date, "j. E Y, H:i")
16    def clean(self):
17        if self.date > timezone.now():
18            raise ValidationError(_("You cannot add future crimes!"))
```

Výpis 4.6 Definice modelu Crime (zdroj: vlastní)

Tento model nám spojuje všechny předešlé modely do jednoho celku, které tvoří hlášení jedné události. Součástí tohoto modelu je opět validace pro omezení uživatele v přidání záznamu s budoucím datem. Mimo klíčů odkazujících k záznamům z předchozích tabulek je součástí také popis incidentu, ten je přidán pomocí dvou proměnných, krátký popis summary, který je omezen do 200 znaků a detailnější description. Pro kompletní obsažení všech potřebných dat zbývá pouze čas, který pokud není poskytnut bude nastaven na čas vytvoření záznamu, tím se rozumí čas systému, kde webový server běží. Tento čas je v kontextu časového pásma nastaveném v souboru `settings.py` (popsán v následující kapitole 5).

4.5 Implementace tabulky uživatelských profilů

Django poskytuje vlastní model uživatelů, ale pokud chceme tento model rozšířit o další položky je potřeba vytvořit model Profile, který se bude vázat na model uživatele. Takto můžeme každému uživateli na tento nový profil přidat například fotografii. Podle potřeby je možné tento Profile model 4.7 rozšiřovat. Implementace je umístěná v modulu `user_profile` 4.1b.

```
1 class Profile(models.Model):
2     user = models.OneToOneField(User, on_delete=models.CASCADE,
3     ↪ verbose_name=_("user"))
4     picture = models.ImageField(upload_to="Profile_Pictures",
5     ↪ height_field=None, width_field=None, max_length=100, null=True,
6     ↪ blank=True, verbose_name=_("picture"))
7     def __str__(self):
8         return self.user.username
9     def save(self, *args, **kwargs):
10        super().save(*args, **kwargs)
11        if self.picture:
12            img = Image.open(self.picture.path)
13            cropped = self.crop_center(img)
14            cropped.save(self.picture.path)
15    def crop_center(self, pil_img):
16        crop = min(pil_img.size)
17        img_width, img_height = pil_img.size
18        return pil_img.crop(((img_width - crop) // 2, (img_height - crop)
19        ↪ // 2, (img_width + crop) // 2, (img_height + crop) // 2 ))
20    class Meta:
21        verbose_name = _("profile")
22        verbose_name_plural = _("profiles")
```

Výpis 4.7 definice modelu Profile (zdroj: vlastní)

Jelikož se jedná o profilovou fotografii, která je většinou čtvercová, můžeme před uložením této fotografie manuálně oříznout největší čtverec a tento ořez následně uložit jako novou fotografii. Tímto zajistíme, že fotografie, ať už od uživatele dostaneme jakoukoliv, bude vždycky mít vždy poměr stran 1:1.

Aby se zajistilo, že při vytvoření uživatele se vytvoří i příslušný model, Django umožňuje vytvářet signály. Nový signál vytvoří prázdný profil pro nově vytvořeného uživatele. 4.8

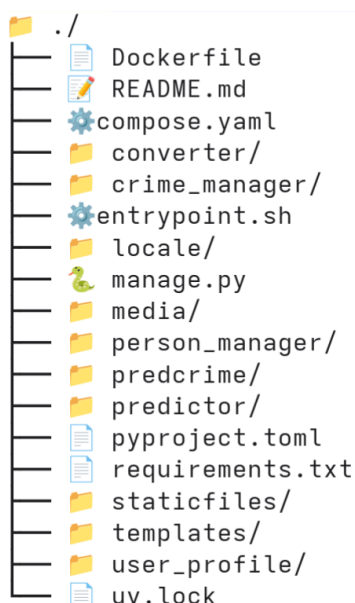
```
1 User = get_user_model()
2 @receiver(post_save, sender=User)
3 def create_profile(sender, instance, created, **kwargs):
4     if created:
5         Profile.objects.create(user=instance)
```

Výpis 4.8 Signál pro automatické vytváření uživatelských profilů (zdroj: vlastní)

5 IMPLEMENTACE BACKENDU A STRUKTURA PROJEKTU

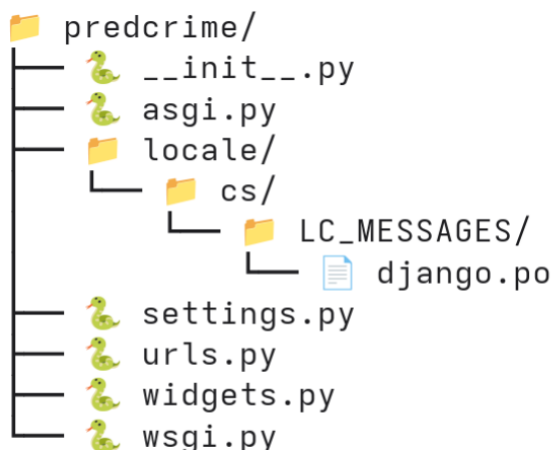
Následující kapitoly popisují práci s knihovnou Django a jejím Template jazykem (popsány v kapitole 2.4). Dále se používá client-side JavaScript s jQuery knihovnou (kapitola 2.5) a knihovny pro práci s daty popsané v kapitole 2.6.

Django projekt se řadí do samostatných aplikací, které sice nejsou myšlené jako samostatné celky, ale usnadňují čitelnost a rozdělují části naší aplikace na menší moduly. 5.1



Obrázek 5.1 Kořenová složka projektu

Hlavní modul, který slouží jako výchozí nese název predcrime, tento hlavní modul aplikace je vytvořen automaticky při vytvoření Django projektu. Obsahuje několik souborů, které Django framework potřebuje, aby se aplikace úspěšně spustila.



Obrázek 5.2 Obsah složky hlavního modulu predcrime

První soubor, který je zde zároveň nejpoužívanější, je `settings.py`, tento soubor obsahuje veškeré nastavení projektu, je zde adresa DBMS 5.1, ale i interní Django nastavení, jako je seznam aplikací, podle kterých Django hledá jak v složce projektu, tak i oficiálními rozšířeními, které jsou součástí Django balíčku. Je zde také možné nastavit jazyk aplikace, upřesnit jazyky pro které chceme překládat naši aplikaci a spousty dalších nastavení. Vývojář také může použít tento soubor pro uchování vlastních nastavení pro svou aplikaci.

```
1 DATABASES = {  
2     "default": {  
3         "ENGINE": "django.contrib.gis.db.backends.postgis",  
4         "NAME": os.environ.get("POSTGRES_DB"),  
5         "USER": os.environ.get("POSTGRES_USER"),  
6         "PASSWORD": os.environ.get("POSTGRES_PASSWORD"),  
7         "HOST": "db",  
8         "PORT": "",  
9     }  
10 }
```

Výpis 5.1 Nastavení DBMS v souboru `settings.py` (zdroj: vlastní)

Soubor `urls.py` je seznam adres aplikace. V případě hlavního modulu tyto adresy odkazují na ostatní soubory `urls.py` v ostatních modulech, které pak odkazují na konkrétní Python funkce (zobrazení), které následně vrací vykreslenou webovou stránku za pomoci Django Template jazyka nebo jednoduchou HTTP odpověď.

Soubory `asgi.py` a `wsgi.py` jsou předem vytvořené soubory, díky kterým můžeme aplikaci spustit na svém vlastním webovém serveru. Django obsahuje ASGI a WSGI implementace webových serverů, které používá ve výchozím nastavení.

WSGI je specifikace, která poskytuje popis standardního rozhraní mezi Python aplikacemi a webovými servery. Přesněji říká vývojářům webových frameworků jak implementovat jejich webové servery. Díky tomuto se nemusíme při vývoji v Django starat o low-level implementaci vlastního webového serveru. [37]

ASGI je modernější specifikace zpětně kompatibilní s WSGI a s podporou asynchronních aplikací. [38]

Další moduly z obrázku 5.1 jsou popsány v následujících kapitolách.

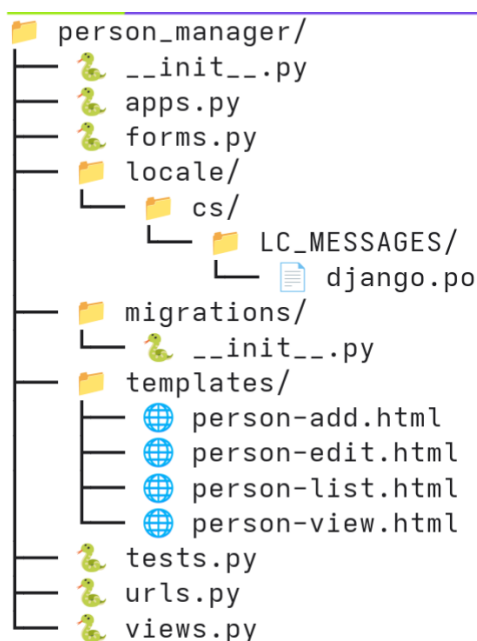
5.1 Modul pro správu osob

Tento modul ve složce `person_manager` 5.1 bude poskytovat funkcionality pro výpis a přidávání osob, zobrazení podrobností o osobě a úpravě těchto informací pomocí jednoduchého formuláře.

Výpis kódu 5.2 popisuje URL cesty pro všechna zobrazení v tomto modulu. Django může očekávat i proměnné v odkazu pomocí uzavření názvu proměnné v adrese do ostrých závorek.

```
1 urlpatterns = [  
2     path("", views.list_all, name="list"),  
3     path("view/<personId>/", views.view, name="view"),  
4     path("view/<personId>/edit/", views.edit, name="edit"),  
5     path("add/", views.add, name="add"),  
6 ]
```

Výpis 5.2 Obsah souboru `urls.py` v modulu `person_manager`



Obrázek 5.3 Obsah složky modulu `person_manager`

Na tyto cesty se napojují Django zobrazení definované v souboru `views.py` 5.3, které vrací odpověď ve formě zprávy s HTTP statusem nebo webové stránky namísto samotné zprávy. Každá tato funkce představuje logiku dané webové stránky, přesněji tedy zobrazení jednotlivé stránky, z pohledu Django frameworku.

Funkce `list_all` slouží k zobrazení webové stránky obsahující seznam všech osob z databáze. Tento seznam používá objektu `Paginator`, který nám převede všechny záznamy na pomyslnou knihu. V této knize každá stránka obsahuje daný počet záznamů, v příkladu konkrétně deset. Pomocí `request.GET.get()` funkce zjistíme parametry v adrese. Funkce hledá hodnotu pro parametr `page`, který v URL adrese může vypadat jako `https://example.com/person?page=3` (jsme na stránce číslo 3). Objekt `Paginator` se tak postará o zbytek a vrátí správných deset záznamů pro požadovanou třetí stránku. 5.3

```
1 def list_all(request):
2     paginator = Paginator(Person.objects.all().order_by("-id"), 10)
3     page_number = request.GET.get("page")
4     page_obj = paginator.get_page(page_number)
5     context = {"people": page_obj}
6     return render(request, "person-list.html", context)
```

Výpis 5.3 Zobrazení pro výpis všech osob (zdroj: vlastní)

Funkce `view` slouží k zobrazení stránky s podrobnostmi o dané osobě podle jejího ID čísla v databázi. Používá se příslušná Django funkce, která vyhledá v databázi osobu a vrátí instanci třídy osoby. Tuto instanci dále předá jako kontext pro Django Template soubor `person-view.html`. 5.4

```
1 def view(request, personId):
2     try:
3         person = Person.objects.get(id=personId)
4         return render(request, "person-view.html", {"person": person})
5     except ObjectDoesNotExist:
6         return HttpResponseBadRequest("Person with this ID does not
           ↪ exist.")
```

Výpis 5.4 Zobrazení osoby konkrétní osoby (zdroj: vlastní)

Funkce pro přidávání osob využívá vlastní Django `ModelForm` třídy, která využívá již implementovaného modelu k vytvoření formuláře podle polí tohoto modelu. Data se ukládají automaticky pomocí funkce `save`, která je součástí všech `Form`, ale i `Model` tříd v Django. Před samotným uložením se provádí validace, jak klientská (prováděná přes HTML template), tak serverová. 5.5

```
1 @login_required
2 def add(request):
3     if request.method == "POST":
4         form = PersonAddForm(request.POST)
5         if form.is_valid():
6             instance = form.save()
7             return redirect("person_manager:view", instance.id)
8     else:
9         form = PersonAddForm()
10    return render(request, "person-add.html", {"form": form})
```

Výpis 5.5 Zobrazení pro přidání osoby (zdroj: vlastní)

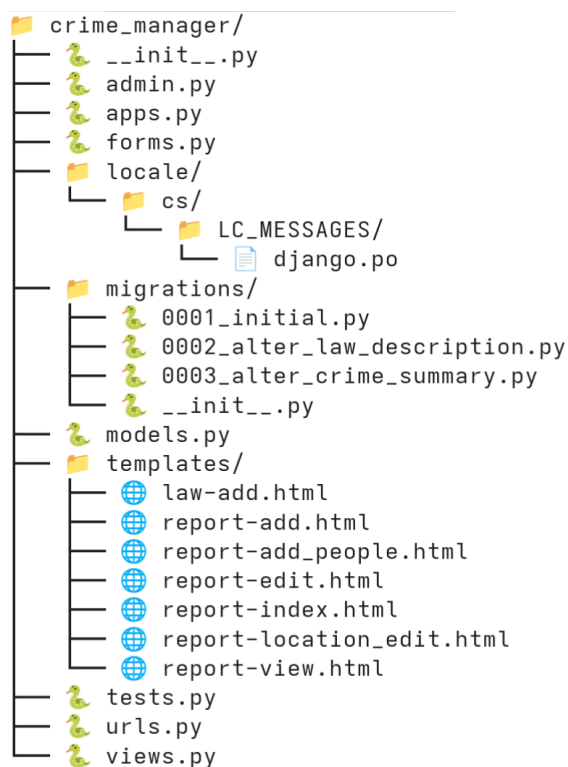
Poslední funkce edit je implementována stejným způsobem jako funkce add 5.5, liší se pouze v použitém formuláři, který uživateli odkrývá pouze pole pro popis osoby, na rozdíl od formuláře PersonAddForm, který od modelu Person ukazuje pole pro jméno, druhé jméno, příjmení, rodné číslo a datum narození. Widget data narození je manuálně změněn na DateInput, aby se zamezilo zobrazení textového pole namísto správného widgetu pro výběr data. 5.6

```
1 class DateInput(forms.DateInput):
2     input_type = "date"
3
4 class PersonAddForm(forms.ModelForm):
5     class Meta:
6         model = Person
7         fields = [
8             "name",
9             "middle_name",
10            "surname",
11            "identification_number",
12            "birth_date",
13        ]
14        widgets = {
15            "birth_date": DateInput(),
16        }
17
18 class PersonEditForm(forms.ModelForm):
19     class Meta:
20         model = Person
21         fields = ["description"]
```

Výpis 5.6 Formuláře pro přidání a úpravu osoby (zdroj: vlastní)

5.2 Modul pro správu událostí

Tento nejrozsáhlejší modul ve složce `crime_manager` 5.1 obsahuje logiku pro správu událostí (zločinů a přestupků) a přiřazováním zákonů a osob těmto událostem.



Obrázek 5.4 Obsah složky modulu `crime_manager`

Soubor `urls.py` opět obsahuje cesty pro všechny zobrazení v tomto modulu. 5.7

```

1 urlpatterns = [
2     path("", views.home, name="index"),
3     path("add/", views.add, name="add"),
4     path("add_law/", views.add_law, name="add_law"),
5     path("view/<crimeId>/", views.view, name="view"),
6     path("view/<crimeId>/add_person/", views.add_person,
7         ↪ name="add_person"),
8     path("view/<crimeId>/edit/", views.edit, name="edit"),
9     path("view/<crimeId>/edit/location/", views.edit_location,
10        ↪ name="edit_location"),
11 ]
  
```

Výpis 5.7 Obsah souboru `urls.py` v modulu `crime_manager` (zdroj: vlastní)

Jelikož se jedná o modul s názvem `crime_manager`, zobrazení `add` obsahuje logiku pro přidávání jednotlivých událostí (zločinů a přestupků). 5.8

```

1 @login_required
2 def add(request):
3     if request.method == "POST":
4         form = CrimeMultiForm(request.POST)
5         if form.is_valid():
6             instance = form.save()
7             return redirect("crime_manager:view", instance["crime"].id)
8     else:
9         form = CrimeMultiForm()
10    return render(request, "report-add.html", {"form": form})

```

Výpis 5.8 zobrazení pro přidání události (zdroj: vlastní)

Toto zobrazení využívá opět vlastního formuláře definovaného v souboru `forms.py`. Tento formulář využívá malé knihovny `betterforms`, která umožňuje poslat více formulářů v jediném form objektu.

```

1 class CrimeForm(forms.ModelForm):
2     class Meta:
3         model = Crime
4         fields = ["summary", "law"]
5         widgets = {"law": forms.Select(attrs={"style": "width:320px"})}
6 class LocationForm(forms.ModelForm):
7     coordinates = PointField(widget=LeafletPointWidget())
8     class Meta:
9         model = Location
10        fields = ["coordinates", "description"]
11 class CrimeMultiForm(MultiModelForm):
12     form_classes = {"crime": CrimeForm, "location": LocationForm}
13     def save(self, commit=True):
14         objects = super(CrimeMultiForm, self).save(commit=False)
15         crime = objects["crime"]
16         location = objects["location"]
17         if commit:
18             location.save()
19             crime.location = location
20             crime.save()
21     return objects

```

Výpis 5.9 formuláře pro přidávání události (zdroj: vlastní)

`CrimeMultiForm` třída využívá již zmiňované knihovny `betterforms`, která zabalí jednotlivé formuláře do jediného `MultiForm` objektu, který se může vykreslovat na stránce jako jediný formulář, nebo umožňuje přistupovat k jednotlivým formulářům v objektu. 5.9

Formulář LocationForm navíc využívá vlastního widgetu definovaného v hlavním modulu aplikace v souboru `widgets.py`. 5.10

```
1 class LeafletPointWidget(Widget):
2     template_name = "widgets/leaflet_point.html"
3
4     def __init__(self, attrs=None, readonly=False, setposition=True):
5         self.readonly = readonly
6         self.setposition = setposition
7         default_attrs = {
8             "map_id": "leaflet-map",
9             "default_lat": 0,
10            "default_lon": 0,
11            "default_zoom": 17,
12        }
13        if attrs:
14            default_attrs.update(attrs)
15        super().__init__(default_attrs)
16
17    def render(self, name, value, attrs=None, renderer=None):
18        if attrs is None:
19            attrs = {}
20
21        attrs.update(self.attrs)
22        attrs["readonly"] = self.readonly
23        attrs["setposition"] = self.setposition
24
25        point_lat = point_lon = None
26        if value and hasattr(value, "y") and hasattr(value, "x"):
27            point_lat = value.y
28            point_lon = value.x
29
30        context = {
31            "name": name,
32            "map_id": attrs.get("map_id", f"map-{name}"),
33            "default_lat": attrs.get("default_lat", 0),
34            "default_lon": attrs.get("default_lon", 0),
35            "default_zoom": attrs.get("default_zoom", 2),
36            "point_lat": point_lat,
37            "point_lon": point_lon,
38            "readonly": attrs.get("readonly", False),
39            "setposition": attrs.get("setposition", True),
40        }
41
42        return mark_safe(render_to_string(self.template_name, context))
```

Výpis 5.10 definice vlastního LeafletPointWidget widgetu (zdroj: vlastní)

Tento na první pohled komplexní widget je pouze backendová logika, která předává hodnoty do template souboru, se kterými pracuje už client-side JavaScript 5.12 a přijímá hodnoty z onoho HTML 5.11 zobrazení. JavaScript kód v tomto zobrazení následně převádí na typ Point, se kterým už GeoDjango dokáže pracovat.

```
1 <div id="{{ map_id }}" style="height: 400px; width: 100%;"></div>
2 <input type="hidden" name="{{ name }}" {% if point_lat and point_lon
  →   %}value="{{ point_lat }},{{ point_lon }}" {% endif %} id="input-{{
  →   map_id }}" />
```

Výpis 5.11 HTML element pro Leaflet.js mapu a skrytý vstup (zdroj: vlastní)

Celé zobrazení využívá JavaScript knihovny Leaflet.js, která nám poskytuje mapy využívající online služby OpenStreetMap. Template soubor obsahuje prvek na kterém Leaflet.js inicializuje mapu a skryté vstupní pole, přes které předává souřadnice vybraného bodu na mapě backendové logice. 5.12

Celý tento widget je uzpůsoben tak, aby nám umožnil vytvořit mapu, přes kterou bude uživatel vybírat bod nebo v opačném případě jednoduše ukázat bod na mapě bez možnosti výběru, či jiné interakce s elementem mapy. Součástí je také funkce, která nám zobrazí bod znázorňující aktuální pozici zařízení na mapě. 5.12

```
1 var map;
2 {% if not readonly and setposition %}
3     navigator.geolocation.getCurrentPosition(setMapPosition);
4     function setMapPosition(position) {
5         map.setView([position.coords.latitude,
6             ↪ position.coords.longitude], {% default_zoom %});
7     }
8 {% endif %}
9 document.addEventListener('DOMContentLoaded', function() {
10     var initialLat = {% point_lat|default:default_lat|stringformat:"f"
11         ↪ %};
12     var initialLon = {% point_lon|default:default_lon|stringformat:"f"
13         ↪ %};
14     map = L.map('{{ map_id }}').setView([initialLat, initialLon], {%
15         ↪ default_zoom %});
16     L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png',
17         ↪ {maxZoom: 19, attribution: '&copy; <a
18         ↪ href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'}
19     ).addTo(map);
20     var marker = null;
21     {% if point_lat and point_lon %}
22         coordinates = [initialLat, initialLon];
23         marker = L.marker(coordinates).addTo(map);
24         document.getElementById('input-{{ map_id }}').value =
25             ↪ initialLat.toFixed(6) + ',' + initialLon.toFixed(6);
26     {% endif %}
27     {% if not readonly %}
28         map.on('click', function(e) {
29             if (marker) {
30                 map.removeLayer(marker);
31             }
32             marker = L.marker(e.latlng).addTo(map);
33             document.getElementById('input-{{ map_id }}').value =
34                 ↪ e.latlng.lat.toFixed(6) + ',' + e.latlng.lng.toFixed(6);
35         });
36     {% else %}
37         map.dragging.disable();
38         map.touchZoom.disable();
39         map.doubleClickZoom.disable();
40         map.scrollWheelZoom.disable();
41         map.boxZoom.disable();
42         map.keyboard.disable();
43         map.zoomControl.remove();
44     {% endif %}
45 });
```

Funkce view v souboru views.py slouží k zobrazení podrobností o události. Součástí zobrazení je stejný widget, který využívá vstupních argumentů tak, aby nepovolil výběr lokace, ale aby pouze zobrazil lokaci na mapě. 5.13

```
1 def view(request, crimeId):
2     crime = get_object_or_404(Crime, id=crimeId)
3     location = get_object_or_404(Location, id=crime.location.id)
4     form = ReadOnlyLocationForm(instance=location)
5     return render(request, "report-view.html", {"crime": crime, "form":
        ↪ form})
```

Výpis 5.13 funkce zobrazení podrobností o události (zdroj: vlastní)

ReadOnlyLocationForm je už jiný formulář, ale využívá stejného widgetu jako jeho již zmíněný protějšek. 5.14

```
1 class ReadOnlyLocationForm(forms.ModelForm):
2     coordinates = PointField(widget=LeafletPointWidget(readonly=True),
        ↪ label=False)
3
4     class Meta:
5         model = Location
6         fields = ["coordinates"]
7
8     def __init__(self, *args, **kwargs):
9         super().__init__(*args, **kwargs)
10        for field_name in self.fields:
11            self.fields[field_name].disabled = True
```

Výpis 5.14 Formulář pro zobrazení mapy s lokací (zdroj: vlastní)

Další zobrazení jsou implementovány již klasicky, za použití stejných principů, jako u předchozích zobrazení využívajících ModelForm formulářů. Tyto funkce jsou přidání osoby, zákona a úpravy lokace u události. Funkce, která slouží k úpravě lokace využívá stejného principu vytvoření formuláře využívající již zmíněný LeafletPointWidget.

Zobrazení pro hlavní stránku modulu obsahuje opět seznam všech událostí zaobalené do objektu Paginator, který nám opět pomůže zobrazovat jen 10 záznamů na stránku bez komplexního JavaScriptu. Záznamy jsou řazeny podle data vytvoření události v systému.

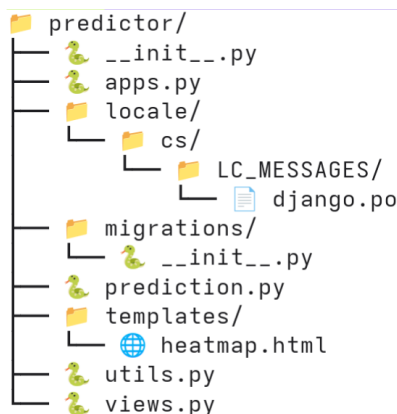
5.15

```
1 def home(request):
2     paginator = Paginator(Crime.objects.all().order_by("-date"), 10)
3     page_number = request.GET.get("page")
4     page_obj = paginator.get_page(page_number)
5     context = {"crimes": page_obj}
6     return render(request, "report-index.html", context)
```

Výpis 5.15 použití objektu Paginator pro výpis všech událostí (zdroj: vlastní)

5.3 Modul pro zobrazení predikce

Tento modul je paradoxně nejméně komplexní.



Obrázek 5.5 Obsah složky modulu predictor

Modul pro predikci ve složce predictor 5.1 obsahuje pouze jedno zobrazení. Toto zobrazení je umístěné na hlavní stránce aplikace.5.16

```

1 def show_heatmap(request):
2     try:
3         threshold = request.GET.get("threshold", 0)
4         hour = request.GET.get("hour", timezone.now().hour)
5         threshold = float(threshold)
6         hour = int(hour)
7         if not 0 <= threshold <= 1 or hour not in range(0, 24):
8             raise ValueError()
9     except ValueError:
10        threshold = 0
11        hour = timezone.now().hour
12    cache_key = f"heatmap_data_{threshold}_{hour}"
13    normalized_heatmap = cache.get(cache_key)
14    if normalized_heatmap is None:
15        dataset = get_crimes_with_locations()
16        locations = pd.concat([dataset["latitude"],
17                               ↪ dataset["longitude"]], axis=1)
18        model = CrimePredictionModel()
19        model.fit(dataset)
20        normalized_heatmap = model.get_heatmap_data(locations, threshold,
21                                                    ↪ hour)
22        cache.set(cache_key, normalized_heatmap, 3600)
23    if len(normalized_heatmap) != 0:
24        position = [normalized_heatmap[0][0], normalized_heatmap[0][1]]
25    else:
26        position = [49.22553, 17.66533] # Default
27    zoom = 14
28    context = {
29        "heatmap": normalized_heatmap,
30        "defaultPosition": position,
31        "zoom": zoom,
32    }
33    return render(request, "heatmap.html", context)

```

Výpis 5.16 zobrazení pro výstup predikce (zdroj: vlastní)

Toto zobrazení využívá stejné knihovny Leaflet.js jako předchozí widget, navíc je zde rozšíření Leaflet-heat pro tuto knihovnu, která přidává schopnost zobrazování heatmap. HTML zobrazení obsahuje prvky, pomocí kterých je možné upřesnit čas predikce a nastavit mezní hodnotu, od které se jednotlivé tepelné body budou zobrazovat. 5.17

```

1 <div id="map" style="height: 600px; width: 100%;"></div>
2 <form method="GET" class="container border border-secondary rounded p-2
  ↳ mt-2">
3   <div class="row align-items-start">
4     <div class="col-3 align-self-center">
5       <label for="threshold" class="form-label">{% translate
  ↳ "Threshold" %}</label>
6     </div>
7     <div class="col">
8       <input id="threshold" class="form-range" type="range"
  ↳ name="threshold" step="0.01" min="0" max="1" value="0">
9     </div>
10  </div>
11  <div class="row align-items-start">
12    <div class="col-3 align-self-center">
13      <label for="hour" class="form-label">{% translate "Hour"
  ↳ %}</label>
14    </div>
15    <div class="col">
16      <input id="hour" class="form-control" name="hour"
  ↳ type="number" min="0" max="23" value="14">
17    </div>
18  </div>
19  <button type="submit" class="btn btn-primary mt-3">{% translate
  ↳ "Submit" %}</button>
20 </form>

```

Výpis 5.17 Obsah hlavní stránky s predikcí (zdroj: vlastní)

Stránka obsahuje logiku pro změnu parametrů v URL adrese, které následně čte show_heatmap zobrazení 5.16. Tyto prvky jsou již umístěné jako součást předdefinovaného formuláře na hlavní stránce 5.17. Zmíněná logika je umístěna v JavaScriptu na stejné stránce.

5.18

```

1 var hour = /hour=([^&#]=*)/.exec(window.location.search);
2 var threshold = /threshold=([^&#]=*)/.exec(window.location.search);
3 if (threshold != null) {
4   document.getElementById("threshold").value = threshold[1]
5 }
6 if (hour != null) {
7   document.getElementById("hour").value = hour[1]
8 }

```

Výpis 5.18 Logika pro změnu hodiny a prahové hodnoty

Stejným způsobem jako JavaScript v LeafletPointWidget nastavuje výchozí zobrazení mapy, tak stránka s predikcí zobrazuje modrý špendlík jako znázornění aktuální pozice zařízení (uživatele). Po kliknutí na špendlík se zobrazí pouze nápis, který vysvětluje význam tohoto bodu. Tato funkce je závislá na oprávnění stránky v prohlížeči. Uživatel tak musí udělit oprávnění k přístupu k poloze, pro zobrazení tohoto bodu. 5.19

```
1 function setMapPosition(position) {  
2     coordinates = [position.coords.latitude, position.coords.longitude];  
3     marker = L.marker(coordinates);  
4     marker.bindPopup("{% translate 'You are here.' %}").openPopup();  
5     marker.addTo(map);  
6 }  
7 navigator.geolocation.getCurrentPosition(setMapPosition);
```

Výpis 5.19 Funkce pro přidání špendlíku na pozici uživatele

Samotné přidávání dat do heatmap vrstvy je velice snadné. Knihovna Leaflet-heat přebírá data ve formátu pole s hodnotami zeměpisné šířky, délky a teploty (hodnota, která ve výchozím nastavení knihovny nabírá hodnot od nuly až po jedničku). Jedno toto trojčlenné pole představuje jeden bod v této vrstvě, knihovna očekává více těchto bodů. Tyto body jsou umístěné v dalším poli, které následně po inicializování vrstvy lze přímo přidat do již existující Leaflet.js mapy. 5.20

```
1 const heatmapData = {{ heatmap }};  
2 map = L.map('map').setView([{{ defaultPosition.0|stringformat:"f" }},  
    → {{ defaultPosition.1|stringformat:"f" }}, {{ zoom }}]);  
3 L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',  
    → {attribution: '&copy; OpenStreetMap contributors'}).addTo(map);  
4 heat = L.heatLayer(heatmapData).addTo(map);
```

Výpis 5.20 Přidání heatmap vrstvy k Leaflet.js mapě (zdroj: vlastní)

Toto pole bodů heatmap 5.20, obsahující lokace a předpovídané hodnoty jsou v zobrazení získány pomocí následující vlastní třídy. 5.21

```
1 class CrimePredictionModel:
2     def __init__(self, bandwidth=0.01, kernel="gaussian"):
3         self.kde = KernelDensity(bandwidth=bandwidth, kernel=kernel)
4         self.scaler = StandardScaler()
5     def fit(self, X):
6         hours = pd.Series([t.hour for t in X["date"]], name="hour")
7         features = pd.concat([X["latitude"], X["longitude"], hours],
8                               ↪ axis=1)
9         X_scaled = self.scaler.fit_transform(features)
10        self.kde.fit(X_scaled)
11        self.is_fitted = True
12    def predict_probability(self, locations, hour):
13        hour = pd.Series([hour] * len(locations), name="hour")
14        features = pd.concat([locations, hour], axis=1)
15        scaled_features = self.scaler.transform(features)
16        probabilities = self.kde.score_samples(scaled_features)
17        return probabilities
18    def get_heatmap_data(self, locations, threshold, hour):
19        probs = self.predict_probability(locations, hour)
20        value = ((probs.max() - probs.min()) * threshold) + probs.min()
21        heatmap_data = []
22        for i in range(len(locations)):
23            if probs[i] > value:
24                mapped_prob = interp(probs[i], [probs.min(),
25                ↪ probs.max()], [0,1])
26                heatmap_data.append(
27                    [
28                        float(locations["latitude"][i]),
29                        float(locations["longitude"][i]),
30                        float(mapped_prob),
31                    ]
32                )
33        return heatmap_data
```

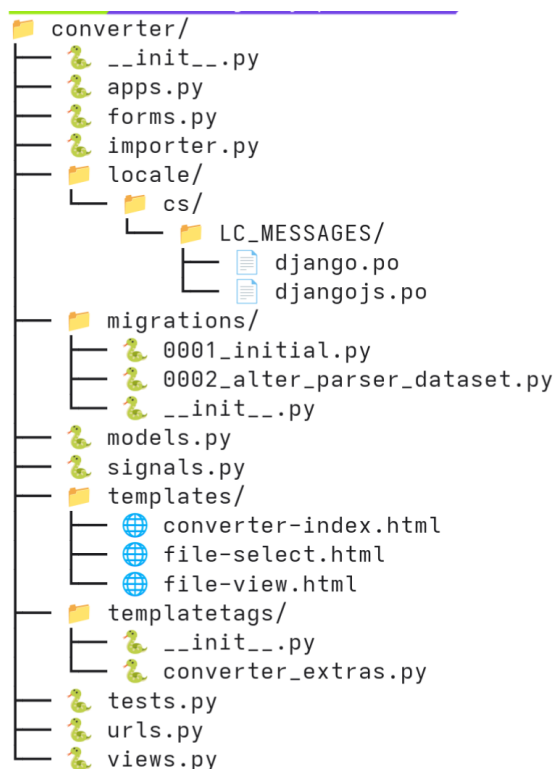
Výpis 5.21 Predikční třída využívající scikit-learn knihovnu (zdroj: vlastní)

Predikce hodnot využívá jednoduchého příkladu s knihovnou scikit-learn a modelu KernelDensity. Tento model je pro učení bez dohledu, to znamená, že model nemá výstupní hodnoty při učení, ale snaží se nalézt vzory v poskytnutých datech. Pomocí tohoto naučeného modelu můžeme získat naši heatmapu ve formě seznamu zeměpisných souřadnic a hodnoty, která znázorňuje intenzitu tepla bodu na heatmapě. Tato hodnota je v rozsahu od nuly až do jedničky. 5.21

Samotný predikční model pracuje pouze se zeměpisnými souřadnicemi a hodinou, kdy se čin stal. Tyto data jsou pro základní implementaci aplikace s tímto základním modelem dostačující.

5.4 Modul pro převod existujících dat

Primárně určen pro administrátory aplikace, tento modul ve složce converter 5.1 je zamýšlen spíše pro přenos dat z jiného systému na tuto aplikaci.



Obrázek 5.6 Obsah složky modulu converter

Využívá se knihovny Pandas pro převodu excelových tabulek, které budou poskytnuty administrátorem, na datové rámce. Tyto rámce jsou ve skutečnosti Python typ `dict`, ale Pandas poskytuje užitečné abstrakce, které pomáhají při manipulaci s těmito daty. Převedený datový rámec obsahuje pojmenované sloupce, které jsou shodné se sloupci v původní excelové tabulce. Útržek této tabulky bude zobrazen na stránce pro administrátora, který podle tohoto náhledu bude moci vybrat a nastavit mapování jednotlivých sloupců na atributy modelů pomocí regulárních výrazů se skupinami. Tyto výrazy se spojí s daným atributem v libovolném Django modelu, který jsme definovali v kapitole 4.

Vlastní modely jsou využity pro převodník. Model Dataset znázorňuje daný soubor s převedeným datovým rámcem a model Parser ukládá informaci o požadovaném mapování regulárního výrazu v daném sloupci datového rámce z Dataset modelu. 5.22

```
1 def dataset_path(instance, filename):
2     # file will be uploaded to MEDIA_ROOT/PredCrime_Datasets/<filename>
3     return "PredCrime_Datasets/{0}".format(filename)
4 class Dataset(models.Model):
5     supported_formats = ["xls", "xlsx", "xlsm", "xlsb", "odf", "ods",
6         ↪ "odt"]
7     file = models.FileField(upload_to=dataset_path, validators=[
8         FileExtensionValidator(allowed_extensions=supported_formats)
9     ], verbose_name=_("file"))
10    data = models.JSONField(null=True, blank=True, verbose_name=_("raw
11        ↪ data"))
12    date = models.DateTimeField(default=timezone.now,
13        ↪ verbose_name=_("date"))
14    @property
15    def filename(self):
16        return os.path.basename(self.file.name)
17    @property
18    def parsers(self):
19        return Parser.objects.filter(dataset=self)
20    def __str__(self):
21        return self.filename
22    class Meta:
23        verbose_name = _("dataset")
24        verbose_name_plural = _("datasets")
```

Výpis 5.22 Model reprezentující datový soubor (zdroj: vlastní)

Model Parser je implementován volně, administrátor může teoreticky nastavit jakoukoliv hodnotu, ale tyto modely se nezobrazují v Django panelu správce. Důvod je, že je očekáváno, že administrátor použije poskytnuté implementace ve frontendu této aplikace, která stejně povolí uživateli nastavit pouze očekávané hodnoty. 5.23

Tato logika je primárně použita pro hodnoty v atributu `field_reference`, která nám uchovává odkaz ve formě textu, který přímo ukazuje na pole v daném modelu. Získání těchto textových hodnot polí je umístěno v ukázce 5.24 jako funkce `get_all_model_attributes`.

```
1 class Parser(models.Model):
2     field_reference = models.CharField(max_length=50, null=False,
3         ↪ blank=False, verbose_name=_("field reference"))
4     column = models.PositiveIntegerField(verbose_name=_("column
5         ↪ number"))
6     matcher = models.TextField(max_length=200, null=False, blank=False,
7         ↪ verbose_name=_("regular expression"))
8     dataset = models.ForeignKey(Dataset, null=False, blank=False,
9         ↪ on_delete=models.PROTECT, verbose_name=_("dataset"),)
10    class Meta:
11        verbose_name = _("parser")
12        verbose_name_plural = _("parsers")
13    def __str__(self):
14        return str(_(f"{self.field_reference} at column {self.column}
15            ↪ with regex group {self.matcher}"))
```

Výpis 5.23 Model reprezentující mapování sloupce (zdroj: vlastní)

Přidávání instance modelu Parser je řešeno pomocí vlastní implementace Django formuláře. Jelikož předem není možné vědět, kolik sloupců bude chtít administrátor převést, využije se `inlineformset_factory`, pomocí kterého lze zobrazit libovolný počet formulářů jednoho typu naráz. 5.24

```
1 def get_all_model_attributes(models, exclude_fields):
2     result = []
3     for model in apps.get_models():
4         model_name = None
5         if model.__name__ in models:
6             model_name = model.__name__
7         if model_name:
8             fields = model._meta.fields
9             for field in fields:
10                 if field.name not in exclude_fields:
11                     result.append(f"{model_name}.{field.name}")
12     return result
13 class ParserForm(forms.ModelForm):
14     importable_models = ["Law", "Crime", "Location"]
15     exclude_fields = ["id", "photo", "coordinates"]
16     choices = [
17         (attr, _(attr))
18         for attr in get_all_model_attributes(importable_models,
19                                             ↪ exclude_fields)
19     ]
20     choices.append((None, _("Not selected")))
21     field_reference = forms.ChoiceField(choices=choices)
22     def __init__(self, *args, **kwargs):
23         super().__init__(*args, **kwargs)
24         self.fields["field_reference"].choices = self.choices
25         self.fields["field_reference"].label = _("field reference")
26     class Meta:
27         model = Parser
28         fields = ["field_reference", "column", "matcher"]
29 ParserFormSet = inlineformset_factory(Dataset, Parser, form=ParserForm,
    ↪ fields=["field_reference", "column", "matcher"], extra=2)
```

Výpis 5.24 vlastní formulář pro vytváření Parser objektů (zdroj: vlastní)

Celkem modul obsahuje pět zobrazení v souboru `views.py`. Na hlavní stránce je opět pomocí objektu `Paginator` zobrazen seznam všech souborů v systému. Je možné mazat, nahrávat a zobrazovat dané soubory v seznamu. Na stránce zobrazení je uživateli povoleno přidat Parser objekt na libovolný sloupec pomocí již zmíněného formuláře. 5.25

```

1 @permission_required("admin")
2 def view_sourcefile(request, fileId=None):
3     file = get_object_or_404(Dataset, id=fileId)
4     if request.method == "POST":
5         form = ParserFormSet(request.POST, instance=file)
6         if form.is_valid():
7             form.save()
8         else:
9             print(form.data)
10            return redirect("converter:index")
11    else:
12        form = ParserFormSet(instance=file)
13        df = pd.read_json(StringIO(file.data))
14        html = df.to_html(max_rows=4)
15        context = {"file": file, "formset": form, "table": html}
16        return render(request, "file-view.html", context)

```

Výpis 5.25 zobrazení podrobností o souboru s daty (zdroj: vlastní)

S touto množinou formulářů se pracuje skoro stejně jako s jedním formulářem. Takže použitím tohoto přístupu se nijak závažně nezvyšuje komplexnost kódu.

Jakmile je administrátor spokojen s nastavením Parser objektů na souboru. Může použít zobrazení pro provedení import procesu, aby se pokusil převést data ze souboru do databáze aplikace. 5.26

```

1 @permission_required("admin")
2 def parse_sourcefile(request, fileId):
3     file = get_object_or_404(Dataset, id=fileId)
4     unparsed = create_tables(file)
5     if unparsed:
6         output = ""
7         for i in unparsed:
8             output += i
9         return HttpResponse(_(f"File parsed, but some rows could not be
           ↳ parsed {output}"), status=200)
10    return HttpResponse(_("All records parsed successfully"),
           ↳ status=200)

```

Výpis 5.26 zobrazení pro spuštění procesu importu dat (zdroj: vlastní)

V tomto procesu existují speciální výjimky pro adresy a časy událostí. S těmito daty je potřeba manipulovat jiným způsobem, aby se import úspěšně provedl.

Manipulace s časy a datумы je snazší, jelikož je třeba jen přetypování, které se děje při převodu mezi textem z původních tabulek na Timestamp typ.

Složitější část tohoto procesu je práce s adresy, jelikož náš systém ukládá zeměpisné souřadnice a s nimi pracuje dál. Data, která poskytne administrátor, s nejvyšší pravděpodobností bude obsahovat jen adresy, tudíž je potřeba tyto adresy převést na souřadnice, ještě před uložením modelu, aby se zabránilo chybě. Pro tento proces je potřeba využít služby Nominatim a knihovny GeoPy. 5.27

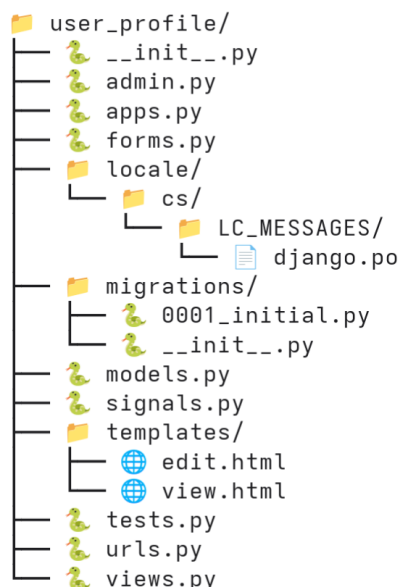
```
1 geocoder = Nominatim(  
2     domain=settings.NOMINATIM_URL,  
3     scheme=settings.NOMINATIM_PROTOCOL,  
4     user_agent=settings.USER_AGENT,  
5 )  
6 def create_tables(dataset):  
7     for index, row in crimes.iterrows():  
8         result = geocoder.geocode(locations.iloc[index]["address"],  
9             ↪ exactly_one=True, language=settings.NOMINATIM_LANG)  
10        if result is None:  
11            unparsed.append(_(f"Failed geocode for  
12                ↪ {locations.iloc[index]['address']}"))  
13            continue  
14            location, _created = Location.objects.get_or_create(  
15                address=locations.iloc[index]["address"],  
16                ↪ coordinates=Point(result.longitude, result.latitude),  
17                ↪ description=_("imported automatically"))  
18            crime.save()  
19    return unparsed
```

Výpis 5.27 použití knihovny GeoPy pro práci s adresami (zdroj: vlastní)

Tento způsob je spíše neefektivní a měl by být definitivně předmětem dalšího vylepšování aplikace.

5.5 Modul pro správu uživatelského účtu

Nejmenší modul, který je umístěn ve složce `user_profile` 5.1 této aplikace, je pro správu uživatelských účtů, který obsahuje stránku zobrazení profilu uživatele a formulář pro změnu fotografie přihlášeného uživatele.



Obrázek 5.7 Obsah složky modulu `user_profile`

Tento modul je implementován stejnými způsoby, které byly použity do teď, nově je pouze potřeba implementace mazání starých fotografií při změně na novou. Tato funkcionality je implementována pomocí dalšího Django signálu 5.28.

```

1 @receiver(pre_save, sender=Profile)
2 def delete_old_picture(sender, instance, **kwargs):
3     if instance.pk:
4         try:
5             old_profile = Profile.objects.get(pk=instance.pk)
6             if old_profile.picture and instance.picture !=
7                 ↪ old_profile.picture:
8                 old_profile.picture.delete(save=False)
9         except Profile.DoesNotExist:
10             pass

```

Výpis 5.28 signál pro mazání starých obrázků (zdroj: vlastní)

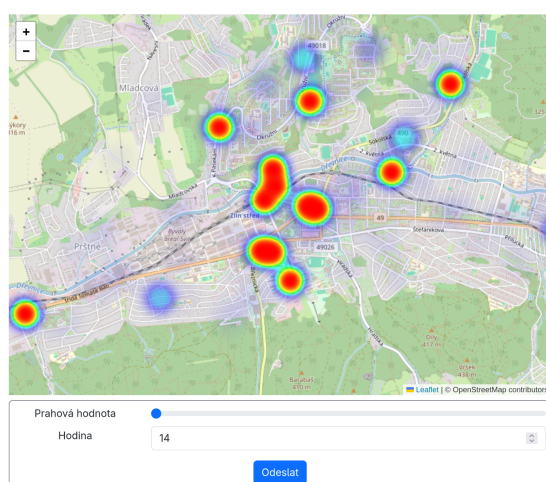
Tento signál se spustí před každým uložením instance modelu `Profile`, proto není potřeba mazat obrázek přímo v signálu a je možné ponechat uložení této změny až na `save` metodě u modelu `Profile`.

6 GRAFICKÝ DESIGN A FRONTEND

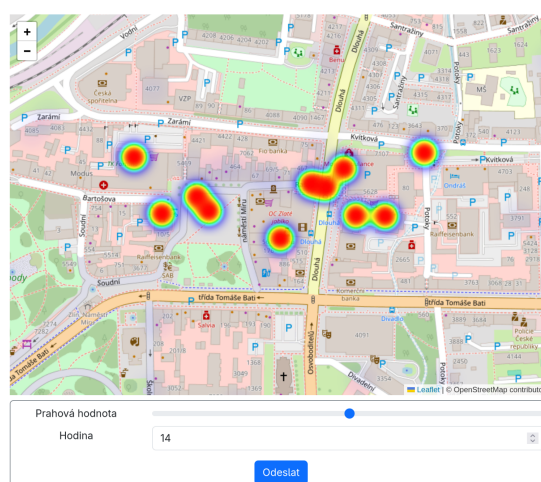
Grafický design celé aplikace závisí na knihovně Bootstrap 2.3, která poskytuje aplikaci hezké, přehledné a také responzivní rozhraní. Tímto způsobem docílíme bezproblémové funkčnosti a správnosti zobrazení na jakémkoliv zařízení. Frontend je zároveň řešen pomocí Django Template jazyka 2.4, který rozšiřuje klasické HTML.

6.1 Hlavní stránka

Při otevření webové aplikace se uživatel setká s mapou, která zobrazuje heatmapu 6.1a jako výstup predikčního modelu. Tuto mapu může uživatel libovolně přibližovat a při získání oprávnění prohlížeče se zobrazí aktuální pozice zařízení uživatele 6.1b. Vlastnosti heatmapy lze upravovat pomocí jednoduchých ovládacích prvků umístěných pod mapou.



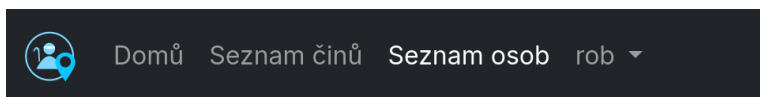
(a) Výchozí přiblížení mapy



(b) Detail místa na mapě

Obrázek 6.1 Zobrazení heatmapy při načtení hlavní stránky

Jako další si uživatel může všimnout menu v horní části stránky 6.2. Obsahuje odkaz, který uživatele přenesení na stránku přihlášení. V případě, že je uživatel již přihlášen, obsahuje jméno uživatele. Při kliknutí na jméno se zobrazí nabídka obsahující odkaz na profil uživatele. V případě role administrátora obsahuje také odkaz směřující na správu datových souborů a na Django panel správce. Samozřejmě v obou případech obsahuje i tlačítko pro odhlášení uživatele.



Obrázek 6.2 Hlavní menu umístěné na stránce s přihlášeným uživatelem

6.2 Správa osob

Seznam všech osob se skrývá pod odkazem "Seznam osob". Stránka disponuje stránkovací funkcí, vypisuje prvních deset osob na jednu stránku. Uživatel může přepínat mezi jednotlivými stránkami pomocí přepínače v horní liště. Ve stejné liště se nachází tlačítko pro přidání nové osoby, které přesměruje uživatele na formulář. 6.3a

Křestní jméno*

Druhé jméno

Příjmení*

Rodné číslo*

Datum narození*

Vytvořit

Popis

Testovací popis pro osobu

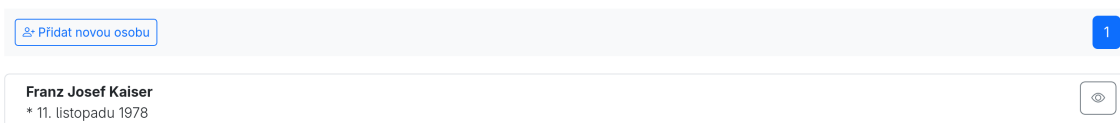
Odeslat

(a) Formulář pro přidání osoby

(b) Formulář pro přidání popisku osobě

Obrázek 6.3 Formuláře pro manipulaci s objektem osoby

Po přidání nové osoby se zobrazí v seznamu 6.4 a uživatel může pomocí ikonky oka zobrazit stránku s podrobnostmi o této osobě 6.5a.



Obrázek 6.4 Zobrazení seznamu osob

Stránka s podrobnostmi o osobě obsahuje celé jméno této osoby, datum narození, věk, popis a seznam událostí, se kterými je osoba spojena. Pokud není popis obsažen, je zde tlačítko, které uživateli dovolí popisek přidat. 6.5a

Osobní informace

Franz Josef Kaiser

narozen: 11. listopadu 1978

věk: 46

➕ Přidat popis osoby

Spáchané činy



Nebyly ohlášeny žádné činy na tuto osobu!

(a) Osoba ve výchozím stavu po přidání

Popis

Upravit popis

Testovací popis pro osobu

(b) Přidaný popisek

Spáchané činy

Testovací Hlášení - 27. dubna 2025, 21:23

(c) Přiřazená událost

Obrázek 6.5 Zobrazení podrobností o osobě

6.3 Správa událostí

Po kliknutí na odkaz s popisem "Seznam činů" bude uživatel přesměrován na seznam všech událostí uložených v databázi. 6.6

+ Nahlásit nový čin		1 2 »
Testovací Hlášení vedle Zlatého jablka Porušila noční klid , § 5 odst. 2 písm. a) bod zákona č. 251/2016 Sb.		27. dubna 2025 23:23
Způsobila škodu na cizím majetku - 1. krádeží Importováno automaticky Způsobila škodu na cizím majetku - 1. krádeží , § 8 odst. 1 písm. a) bod 1 zákona č. 251/2016 Sb.		31. prosince 2022 17:20
Způsobila škodu na cizím majetku - 1. krádeží Importováno automaticky Způsobila škodu na cizím majetku - 1. krádeží , § 8 odst. 1 písm. a) bod 1 zákona č. 251/2016 Sb.		31. prosince 2022 13:40
Způsobila škodu na cizím majetku - 1. krádeží		31. prosince 2022 10:19

Obrázek 6.6 Zobrazení seznamu událostí

Tento seznam je řazen podle data a opět obsahuje deset záznamů na stránku, které lze přepínat pomocí čísel stránek v pravém horním rohu. V levém horním rohu je přítomné tlačítko, které uživatele přenesne na formulář přidání nové události. 6.7

Informace o hlášení

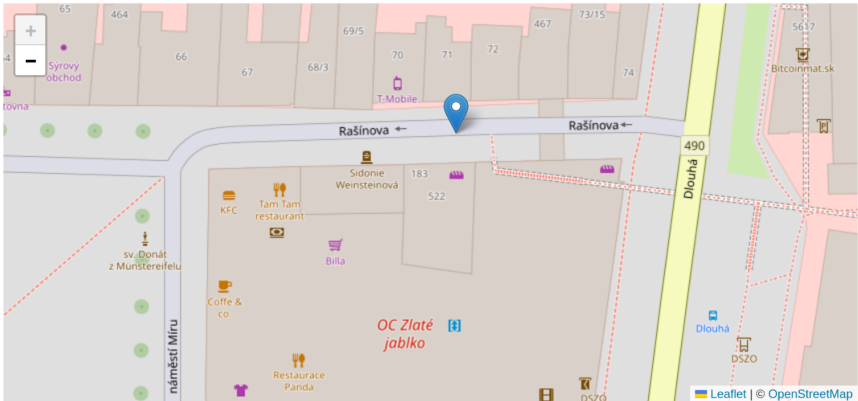
Popis události*

Zákon*

[Přidat nový zákon](#)

Informace o místě

Souřadnice*



Popis*

[Vytvořit](#)

Obrázek 6.7 Formulář vytvoření hlášení

Po úspěšném vyplnění formuláře bude uživatel přenesen zpět na seznam událostí. Při kliknutí na libovolnou událost v seznamu se zobrazí stránka obsahující podrobnosti o události.

6.8a Stránka obsahuje datum, kdy se událost stala, výpis osob, popis a náhled místa.

Informace o hlášení

27. dubna 2025 23:23

Žádní pachatelé

Přidat pachatele

Přidat popis činu

(a) Bez popisku a osoby

Informace o hlášení

27. dubna 2025 23:23

Franz Josef Kaiser (* 11. listopadu 1978)

Přidat pachatele

(b) S přiřazenou osobou

Popis

Upravit popis

Tento popis je testovací

(c) Přiřazený popisek

Obrázek 6.8 Zobrazení informací o činu

V případě, že událost nemá popis je zde zobrazeno tlačítko, které uživateli umožňuje pomocí formuláře přidat popis nový. Stejná funkcionality je nabízena pro přidání osob. 6.9

Viník*

Franz Josef Kaiser (* 11. listopadu 1978)

Josef Pepa Novák (* 11. února 1988)

Odeslat

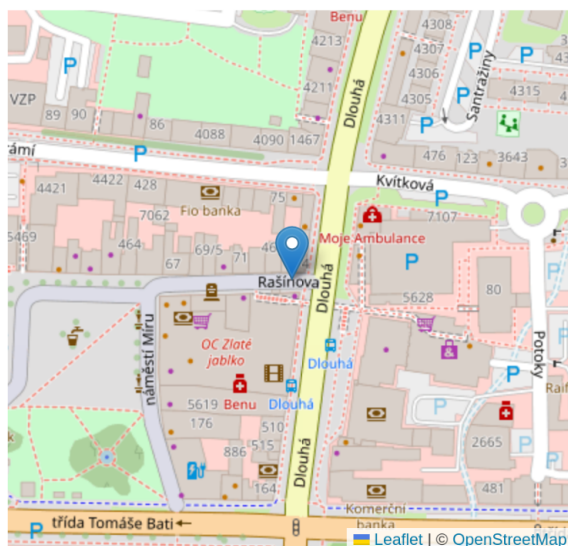
Obrázek 6.9 Formulář pro vybrání osoby ze seznamu

Po přidání nové osoby je zobrazena ve výpisu podrobností o události. 6.8b

Součástí této stránky je rovněž již zmíněný náhled lokace ve formě mapy. Tato mapa nenabízí žádný způsob interakce, jelikož slouží pouze k náhledu přesného místa události. Jde o stejný widget používaný předešlou mapou, pouze s atributem, který slouží pro vypnutí posouvání, přibližování, klikání a všech ostatních interakcí, které uživatel může normálně s mapou provádět. 6.10

Místo hlášení

 Změna umístění



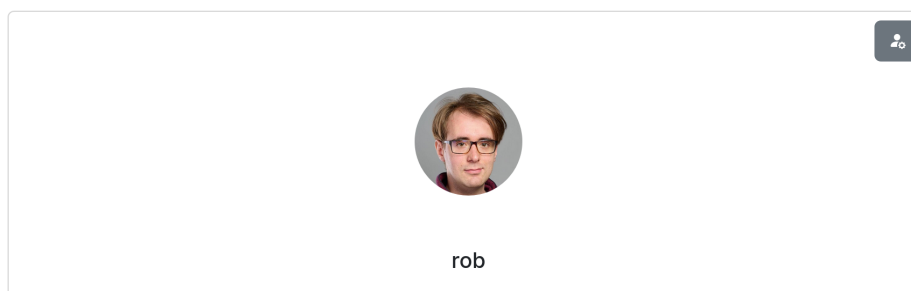
Obrázek 6.10 Mapa na stránce zobrazení hlášení

6.4 Uživatelský profil

Odkaz na zobrazení profilu se nachází v menu pod jménem uživatele. Po kliknutí na odkaz je uživatel přesměrován na jednoduchou stránku, zobrazující profil tohoto aktuálně přihlášeného uživatele. 6.11a



(a) bez vlastní fotografie



(b) s vlastní fotografií

Obrázek 6.11 Zobrazení profilu

V pravé horní části profilu se nachází ikona nastavení profilu. Tímto tlačítkem se uživatel přesune na editaci profilu, kde si může nastavit vlastní fotografii. 6.12

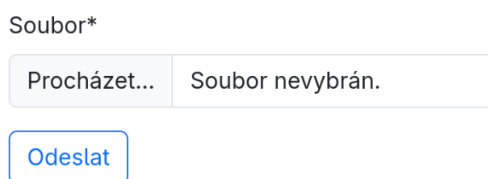
Fotografie

Aktuálně	Profile_Pictures/extralowres_pfp.jpg	☐ Zrušit
Procházet...	Soubor nevybrán.	
Odeslat		

Obrázek 6.12 Nahrání vlastní fotografie profilu

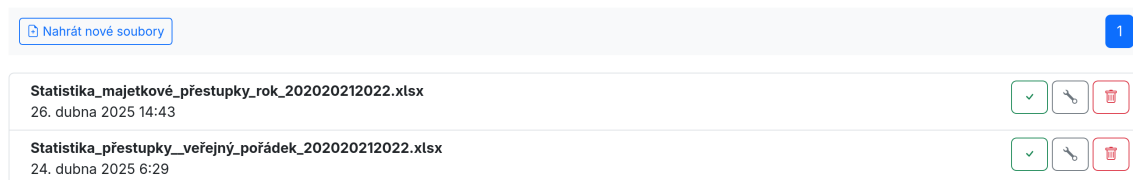
6.5 Import vlastních dat

Pro administrátory systému se v menu pod uživatelským jménem nachází odkaz nesoucí popisek "Správa dat", po kliknutí na tento odkaz se administrátor dostane na stránku, na které může spravovat všechny soubory excelových tabulek, nastavovat mapování těchto souborů na Django modely a provést import. Stránka je strukturovaná jako ostatní seznamy v aplikaci, je zde opět ovládání přepínání stránek a tlačítko pro nahrání nového souboru, které administrátora přesměruje na formulář pro nahrání. 6.13



Obrázek 6.13 Nahrání souboru s daty

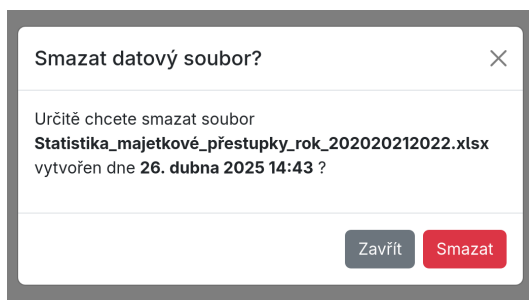
Po nahrání tohoto souboru se zobrazí opět seznam, již obsahující nově nahraný soubor. 6.14



Nahrát nové soubory		1
Statistika_majetkové_přestupky_rok_202020212022.xlsx	26. dubna 2025 14:43	
Statistika_přestupky_veřejný_pořádek_202020212022.xlsx	24. dubna 2025 6:29	

Obrázek 6.14 Seznam všech souborů s daty

Každý řádek se souborem obsahuje tlačítka pro manipulaci. Tlačítko s ikonkou odpadkového koše zobrazí potvrzovací dialog před definitivním smazáním souboru. 6.15



Obrázek 6.15 Dialogové okno pro smazání souboru

Administrátor může nastavit nové Parser objekty pro soubor pomocí tlačítka s ikonkou klíče. Po kliknutí na toto tlačítko se zobrazí náhled záznamů v tabulce. Aby se zamezilo zobrazování celé tabulky, zpomalování prohlížeče a nepřehlednosti stránky, zobrazují se pouze první řádky tabulky. 6.16

Nastavení datového souboru

Statistika_majetkové_přestupky_rok_202020212022.xlsx
Přidáno 26. dubna 2025 14:43

Číslo jednací	Datum a čas	Typ podrobně	Způsob řešení	Ulice
0	2020/070430 1609434420000	Zákon č. 251/2016 Sb., o některých přestupcích - § 8 - Přestupky proti majetku - odst. 1 - Fyzická osoba úmyslně - písm. a) - Způsobila škodu na cizím majetku - 1. krádeží	Řešení ostatní - Na místě - Bez podezření na protiprávní jednání	Zlín, Tyršovo nábřeží 5496
1	2020/070419 1609427760000	Zákon č. 251/2016 Sb., o některých přestupcích - § 8 - Přestupky proti majetku - odst. 1 - Fyzická osoba úmyslně - písm. a) - Způsobila škodu na cizím majetku - 1. krádeží	Řešení přestupku - Řízení o přestupku - Správní trest - Pokuta - PBMN, Řešení přestupku - Sankce - BPN - odesláno na OP-OVP	Zlín, Přímá 671
...	...	...	...	...
795	2022/000284 1641143820000	Zákon č. 251/2016 Sb., o některých přestupcích - § 8 - Přestupky proti majetku - odst. 1 - Fyzická osoba úmyslně - písm. a) - Způsobila škodu na cizím majetku - 4. zničením nebo poškozením věci z takového majetku	Řešení ostatní - Předáno - Přestupkové sekci MP Zlín, Řešení přestupku - Před zahájením řízení - Odloženo dle § 76 zák. č. 250/2016 Sb. - odst. 1 - písm. d) - Podezřelý z přestupku v době spáchání skutku nedovršil 15. rok svého věku	Zlín, Jilová 4568
796	2022/000125 1641064980000	Zákon č. 251/2016 Sb., o některých přestupcích - § 8 - Přestupky proti majetku - odst. 1 - Fyzická osoba úmyslně - písm. a) - Způsobila škodu na cizím majetku - 4. zničením nebo poškozením věci z takového majetku	Řešení ostatní - Poučení o možném způsobu řešení, Řešení ostatní - Řešeno - V součinnosti s Policií ČR, Řešení ostatní - Na místě - Pachatel nezjištěn, Řešení přestupku - Před zahájením řízení - Podezření ze spáchání přestupku, Řešení ostatní - Předáno - Policií ČR - K linému ošetření	Zlín, Partyzánská 3230

Obrázek 6.16 Náhled tabulky z datového souboru

Pod tímto zkráceným náhledem se zobrazují jednotlivé formuláře pro nastavení mapování libovolných sloupců. V této množině formulářů může administrátor vybrat v seznamu pole libovolného modelu, pro který chce daný regulární výraz použít a číslo sloupce v tabulce, ve kterém se má výraz vyhodnocovat. Tyto vybrané hodnoty se následně ukládají na novou instanci Parser objektu 6.17, která je následně uložena do databáze.

Odkaz na pole: Crime.date

Číslo sloupce: 2

(.*)

Regulární výraz:

Smazat: ☐

Obrázek 6.17 Formulář pro vytvoření Parser objektu

Po přidání všech požadovaných hodnot může administrátor spustit proces importování pomocí zeleného tlačítka v seznamu u požadovaného souboru. Po úspěšném importu se zobrazí jednoduchá stránka s textem zda byl import úspěšný. V případě, že se import některých záznamů nezdařil, je administrátorovi ukázán seznam těchto neúspěšně importovaných záznamů.

7 SHRNUÍ IMPLEMENTACE APLIKACE

Aplikace využívá Django ORM pro tvorbu tabulek databáze, které obsahují jednotlivé zákony, lokace, osoby, události, uživatelské profily a dodatečně i objekty pro funkcionalitu importu dat z jiných systémů nebo datových sad. Tvorba těchto tabulek je popsána v kapitole 4. Implementace tabulek pro funkci importu je popsána zvlášť společně s implementací této funkcionality v kapitole 5.4.

Backend aplikace je řešen primárně za použití designových principů frameworku Django. Aplikace obsahuje vlastní formuláře a metody pro zobrazení jednotlivých stránek. Tyto principy a postupy jsou popsány v kapitole 5. Tento backend se přímo váže s Django Template zobrazeními, které jsou popsány s názornými ukázkami v kapitole 6.

Výsledná implementace obsahuje responzivní grafický frontend vytvořený pomocí Django Template jazyka popsáný v kapitole 2.4. Při designu stránek byla použita knihovna Bootstrap (kapitola 2.3). Pro client-side funkcionalitu stránek byla použita JavaScript knihovna jQuery, která byla popsána v kapitole 2.5. Pro funkcionalitu map byly na konkrétních stránkách použity knihovny Leaflet.js, Leaflet-heat a dodatečné styly potřebné pro funkčnost těchto komponent (použití těchto knihoven je popsáno v kapitolách 5.2 a 5.3).

ZÁVĚR

Cíl této práce byl navrhnout a popsat základní požadavky aplikace pro práci s daty predikce kriminality. Cílem bylo rovněž zvolit vhodné technologie pro implementaci a tuto implementaci následně provést a vhodně prezentovat tyto výsledky. Součástí práce má být rovněž hodnocení tohoto implementovaného řešení.

Výsledná aplikace obsahuje responzivní grafické rozhraní pro manipulaci s daty (implementováno a popsáno v kapitole 6). Aplikace poskytuje správcům možnost jednoduše importovat data z jiných systémů či datových zdrojů. Tato funkcionality je popsána v kapitole 5.4. Uživatelé aplikace jsou schopni jednoduše přidávat instance objektů zločinů, osob a porušených zákonů přes formulářová rozhraní popsaných v kapitole 5. Tyto data jsou následně použity pro predikci zločinu. Implementace modelu a logiky pro tuto predikci je popsána v kapitole 5.3. Aplikace je zároveň škálovatelná a modulární díky použitému webovému frameworku Django.

Práce mi dala novou znalost vývoje webových stránek s knihovnou Django, se kterou jsem doposud nepracoval, ale pracovalo se mi velice dobře a rozhodně bych této knihovny velice rád využil i v budoucích projektech, zároveň bych chtěl na tomto projektu pracovat i dále a rád bych tak udělal pod licencí svobodného softwaru, například pod licencí GPLv2, která by mohla poskytnout svobodu pro jednotlivce i instituce využívat a rozšiřovat tento software. Konkrétní další cíle do budoucna by byly sepsání dokumentace, implementace testů, úprava a optimalizace procesů v kódu. Vzhledem k tomu, že se jedná o aplikaci s uživatelským rozhraním, chtěl bych i tuto stránku dále vylepšovat. Nakonec bych tak rád poskytl stabilní základ, který by někdy v budoucnu mohl být nápomocný při policejní práci.

SEZNAM POUŽITÉ LITERATURY

- [1] LONGLEY, Robert, 2020. *Criminology Definition and History*. Online. Dostupné z: <https://www.thoughtco.com/the-history-of-criminology-part-1-974579>. [cit. 2025].
- [2] WILLS, Matthew, 2020. *Criminal Minds? Try Criminal Bodies*. Online. Dostupné z: <https://daily.jstor.org/criminal-minds-try-criminal-bodies/>. [cit. 2025].
- [3] HUNT, Joel, 2019. From crime mapping to crime forecasting. *NIJ Journal*. 2019. 281. Dostupné z: <https://www.ojp.gov/pdffiles1/nij/252036.pdf>.
- [4] ZACH FRIEND, M.P.P., 2013. *Predictive Policing: Using Technology to Reduce Crime*. Online. Dostupné z: <https://leb.fbi.gov/articles/featured-articles/predictive-policing-using-technology-to-reduce-crime>. [cit. 2025].
- [5] HRUŠKA, Lubor; FOLDYNOVÁ, Ivana; ŠOTKOVSKÝ, Ivan; ZAPLETALOVÁ, Ladislava; VÁCLAVÍK, Tomáš et al., 2015. *Mapy Budoucnosti*. Online. Dostupné z: <https://www.mvcr.cz/soubor/prevence-dokumenty-mapy-budoucnosti-pdf.aspx>. [cit. 2025].
- [6] THOMSON, Jonny, 2022. *Algocracy would replace politicians with algorithms. Should we try it?* Online. Dostupné z: <https://bigthink.com/thinking/algocracy-algorithm-government/>. [cit. 2025].
- [7] WANG, Vivian, 2024. *Xi Jinping's Recipe for Total Control: An Army of Eyes and Ears*. Online. Dostupné z: <https://www.nytimes.com/2024/05/25/world/asia/china-surveillance-xi.html>. [cit. 2025].
- [8] JOUBERT, Shayna, 2021. Predictive Analytics. *Northeastern University*. 2021. Dostupné z: <https://graduate.northeastern.edu/resources/predictive-analytics/>.
- [9] DAKALBAB, Fatima; TALIB, Manar Abu; WARAGA, Omnia Abu; NASSIF, Ali Bou; ABBAS, Sohail et al., 2022. Artificial intelligence & crime prediction: A systematic literature review. *Social Sciences & Humanities Open*. 2022. 6, č. 1, s. 100342. ISSN 2590-2911. Dostupné z: <http://dx.doi.org/10.1016/j.ssaho.2022.100342>.

- [10] WOOD, Matt, 2022. *AI Algorithm Predicts Future Crimes One Week in Advance With 90% Accuracy*. Online. Dostupné z: <https://scitechdaily.com/ai-algorithm-predicts-future-crimes-one-week-in-advance-with-90-accuracy/>. [cit. 2025].
- [11] SHAPIRO, Aaron, January 2017. Reform predictive policing. *Nature*. January 2017. 541, č. 7638, s. 458–460. ISSN 1476-4687. Dostupné z: <http://dx.doi.org/10.1038/541458a>.
- [12] KASTE, Martin, 2013. *Can Software That Predicts Crime Pass Constitutional Muster?* Online. Dostupné z: <https://www.npr.org/2013/07/26/205835674/can-software-that-predicts-crime-pass-constitutional-muster>. [cit. 2025].
- [13] HASHEMI-POUR, Cameron, 2024. *What is software?* Online. Dostupné z: <https://www.techtarget.com/searchapparchitecture/definition/software>. [cit. 2025].
- [14] ROSSUM, Guido, 2006. *What is Python? Executive Summary*. Online. Dostupné z: <https://www.python.org/doc/essays/blurbs/>. [cit. 2025].
- [15] OSTROWSKA, Kamila, 2022. *A Brief History of Python*. Online. Dostupné z: <https://learnpython.com/blog/history-of-python/>. [cit. 2025].
- [16] ROSSUM, Guido, 2023. *Guido van Rossum - Brief Bio*. Online. Dostupné z: <https://gvanrossum.github.io/bio.html>. [cit. 2025].
- [17] GROUP, The PostgreSQL Global Development, 2025. PostgreSQL. Dostupné z: <https://www.postgresql.org/about/>.
- [18] RAMSEY, Paul, 2018. Introduction to PostGIS. Dostupné z: <https://postgis.net/workshops/postgis-intro/introduction.html>.
- [19] CERN, 2021. A short history of the Web. Dostupné z: <https://home.web.cern.ch/science/computing/birth-web/short-history-web>.
- [20] WHATWG, 2025. HTML. Dostupné z: <https://html.spec.whatwg.org>.
- [21] RAJ, Anupama, 2024. *History of CSS: The Evolution of Web Design*. Online. Dostupné z: <https://www.almabetter.com/bytes/articles/history-of-css>. [cit. 2025].

- [22] SANDU, Bogdan, 2024. *What Is Bootstrap? A Beginner's Guide*. Online. Dostupné z: <https://wpdean.com/what-is-bootstrap/>. [cit. 2025].
- [23] CONTRIBUTORS, Mozilla, 2025. Django introduction. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Introduction.
- [24] SILVA, Pedro Anisio, 2023. *Understanding Django: A Journey Through Its History*. Online. Dostupné z: <https://pedroanisiosilva.com/understanding-django-a-journey-through-its-history-23c8ce4350a3>. [cit. 2025].
- [25] JOY, Ashwin, 2021. *What Can You Do With Django? (Applications & Examples)*. Online. Dostupné z: <https://pythonistaplanet.com/what-can-you-do-with-django/>. [cit. 2025].
- [26] FOUNDATION, Django Software, 2025. Official Django Logos. Dostupné z: <https://www.djangoproject.com/community/logos/>.
- [27] VERBINA, Evgenia, 2025. *The Ultimate Guide to Django Templates*. Online. Dostupné z: <https://blog.jetbrains.com/pycharm/2025/02/the-ultimate-guide-to-django-templates/>. [cit. 2025].
- [28] MACMANUS, Richard, 2020. *1995: The Birth of JavaScript*. Online. Dostupné z: <https://cybercultural.com/p/1995-the-birth-of-javascript/>. [cit. 2025].
- [29] CASSEL, David, 2018. *Brendan Eich on Creating JavaScript in 10 Days, and What He'd Do Differently Today*. Online. Dostupné z: <https://thenewstack.io/brendan-eich-on-creating-javascript-in-10-days-and-what-hed-do-differently-today/>. [cit. 2025].
- [30] CONTRIBUTORS, Mozilla, 2025. JavaScript Introduction. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>.
- [31] AMADO, Miguel, 2025. *What is jQuery?* Online. Dostupné z: <https://www.websiteplanet.com/glossary/website-builders/what-is-jquery/>. [cit. 2025].
- [32] GUO, Danny, 2019. *The history and legacy of jQuery*. Online. Dostupné z: <https://blog.logrocket.com/the-history-and-legacy-of-jquery/>. [cit. 2025].

- [33] CHUGH, Vidhi, 2025. *Python pandas Tutorial: The Ultimate Guide for Beginners*. Online. Dostupné z: <https://www.datacamp.com/tutorial/pandas>. [cit. 2025].
- [34] MCINTIRE, George; MARTIN, Brendan a WASHINGTON, Lauren, 2025. *Python Pandas Tutorial: A Complete Introduction for Beginners*. Online. Dostupné z: <https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/>. [cit. 2025].
- [35] DUNBAR, Ricky, 2025. *Python NumPy Tutorial: An Applied Introduction for Beginners*. Online. Dostupné z: <https://www.learndatasci.com/tutorials/applied-introduction-to-numpy-python-tutorial/>. [cit. 2025].
- [36] CLARK, Bryan, 2025. *What is Scikit-Learn (Sklearn)?* Online. Dostupné z: <https://www.ibm.com/think/topics/scikit-learn>. [cit. 2025].
- [37] BROWN, Titus, 2006. *An Introduction to the Python Web Server Gateway Interface (WSGI)*. Online. Dostupné z: <http://ivory.idyll.org/articles/wsgi-intro/what-is-wsgi.html>. [cit. 2025].
- [38] TEAM, ASGI, 2018. ASGI Documentation. Dostupné z: <https://asgi.readthedocs.io/en/latest/>.

SEZNAM OBRÁZKŮ

Obrázek 1.1	Cesare Lombroso (Zdroj: [1]).....	13
Obrázek 1.2	Jeff Brantingham - tvůrce softwaru PredPol (Zdroj: [12]).....	16
Obrázek 2.1	Guido van Rossum - tvůrce programovacího jazyka Python (Zdroj: [16])	17
Obrázek 2.2	Logo frameworku Django (Zdroj: [26])	20
Obrázek 2.3	Brendan Eich (Zdroj: [29])	20
Obrázek 4.1	Struktury souborových systémů	25
Obrázek 5.1	Kořenová složka projektu.....	32
Obrázek 5.2	Obsah složky hlavního modulu predcrime	32
Obrázek 5.3	Obsah složky modulu person_manager	34
Obrázek 5.4	Obsah složky modulu crime_manager.....	37
Obrázek 5.5	Obsah složky modulu predictor	43
Obrázek 5.6	Obsah složky modulu converter	48
Obrázek 5.7	Obsah složky modulu user_profile	54
Obrázek 6.1	Zobrazení heatmapy při načtení hlavní stránky	55
Obrázek 6.2	Hlavní menu umístěné na stránce s přihlášeným uživatelem.....	56
Obrázek 6.3	Formuláře pro manipulaci s objektem osoby	56
Obrázek 6.4	Zobrazení seznamu osob	56
Obrázek 6.5	Zobrazení podrobností o osobě	57
Obrázek 6.6	Zobrazení seznamu událostí.....	58
Obrázek 6.7	Formulář vytvoření hlášení	58
Obrázek 6.8	Zobrazení informací o činu	59
Obrázek 6.9	Formulář pro vybrání osoby ze seznamu	59
Obrázek 6.10	Mapa na stránce zobrazení hlášení.....	60
Obrázek 6.11	Zobrazení profilu.....	61
Obrázek 6.12	Nahrání vlastní fotografie profilu	61
Obrázek 6.13	Nahrání souboru s daty	62
Obrázek 6.14	Seznam všech souborů s daty	62
Obrázek 6.15	Dialogové okno pro smazání souboru	62
Obrázek 6.16	Náhled tabulky z datového souboru	63
Obrázek 6.17	Formulář pro vytvoření Parser objektu.....	63

SEZNAM VÝPISŮ

Výpis 4.1	Definice modelu Law (zdroj: vlastní)	26
Výpis 4.2	Definice modelu Location (zdroj: vlastní)	27
Výpis 4.3	Validační objekt pro rodná čísla (Zdroj: vlastní)	27
Výpis 4.4	definice modelu Person (zdroj: vlastní)	28
Výpis 4.5	dodatečné vlastnosti modelu Person (zdroj: vlastní)	28
Výpis 4.6	Definice modelu Crime (zdroj: vlastní)	29
Výpis 4.7	definice modelu Profile (zdroj: vlastní)	30
Výpis 4.8	Signál pro automatické vytváření uživatelských profilů (zdroj: vlastní)	31
Výpis 5.1	Nastavení DBMS v souboru settings.py (zdroj: vlastní)	33
Výpis 5.2	Obsah souboru urls.py v modulu person_manager	34
Výpis 5.3	Zobrazení pro výpis všech osob (zdroj: vlastní)	35
Výpis 5.4	Zobrazení osoby konkrétní osoby (zdroj: vlastní)	35
Výpis 5.5	Zobrazení pro přidání osoby (zdroj: vlastní)	36
Výpis 5.6	Formuláře pro přidání a úpravu osoby (zdroj: vlastní)	36
Výpis 5.7	Obsah souboru urls.py v modulu crime_manager (zdroj: vlastní)	37
Výpis 5.8	zobrazení pro přidání události (zdroj: vlastní)	38
Výpis 5.9	formuláře pro přidávání události (zdroj: vlastní)	38
Výpis 5.10	definice vlastního LeafletPointWidget widgetu (zdroj: vlastní)	39
Výpis 5.11	HTML element pro Leaflet.js mapu a skrytý vstup (zdroj: vlastní)	40
Výpis 5.12	JavaScript v LeafletPointWidget (zdroj: vlastní)	41
Výpis 5.13	funkce zobrazení podrobností o události (zdroj: vlastní)	42
Výpis 5.14	Formulář pro zobrazení mapy s lokací (zdroj: vlastní)	42
Výpis 5.15	použití objektu Paginator pro výpis všech událostí (zdroj: vlastní)	43
Výpis 5.16	zobrazení pro výstup predikce (zdroj: vlastní)	44
Výpis 5.17	Obsah hlavní stránky s predikcí (zdroj: vlastní)	45
Výpis 5.18	Logika pro změnu hodiny a prahové hodnoty	45
Výpis 5.19	Funkce pro přidání špendlíku na pozici uživatele	46
Výpis 5.20	Přidání heatmap vrstvy k Leaflet.js mapě (zdroj: vlastní)	46
Výpis 5.21	Predikční třída využívající scikit-learn knihovnu (zdroj: vlastní)	47
Výpis 5.22	Model reprezentující datový soubor (zdroj: vlastní)	49

Výpis 5.23	Model reprezentující mapování sloupce (zdroj: vlastní)	50
Výpis 5.24	vlastní formulář pro vytváření Parser objektů (zdroj: vlastní)	51
Výpis 5.25	zobrazení podrobností o souboru s daty (zdroj: vlastní)	52
Výpis 5.26	zobrazení pro spuštění procesu importu dat (zdroj: vlastní)	52
Výpis 5.27	použití knihovny GeoPy pro práci s adresami (zdroj: vlastní)	53
Výpis 5.28	signál pro mazání starých obrázků (zdroj: vlastní)	54

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ASGI	Asynchronous Server Gateway Interface
CERN	Conseil Européen pour la Recherche Nucléaire
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CSV	Comma-separated values
DBMS	Database Management System
GIS	Geographic information system
GPL	General Public License
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
SQL	Structured Query Language
URL	Uniform resource locator
W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language

SEZNAM PŘÍLOH

Zdrojový kód programu PredCrime