VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MODERNÍ WEBOVÉ ROZHRANÍ SYSTÉMU NERD

MODERN WEB INTERFACE FOR NERD SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. KRISTÍNA OLTMANOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN ŽÁDNÍK, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

148079

Ústav: Ústav informačních systémů (UIFS)
Studentka: **Oltmanová Kristína, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Kybernetická bezpečnost
Název: **Moderní webové rozhraní systému NERD**
Kategorie: Uživatelská rozhraní
Akademický rok: 2022/23

Zadání:

1. Seznamte se se systémem NERD (Network Entity Reputation Database) vyvíjeným a provozovaným sdružením CESNET, zejména s jeho webovým rozhraním.
2. Navrhněte nové webové rozhraní systému NERD, které bude poskytovat alespoň stejnou funkcionalitu, jako rozhraní stávající, avšak bude založené na moderních webových technologiích. Webové rozhraní bude poskytovat přehlednou prezentaci dat včetně responzivního chování a moderního vzhledu. Webové rozhraní bude podporovat registraci uživatelů a přihlašování pomocí různých poskytovatelů identity (např. sociálních sítí).
3. Navržené webové rozhraní implementujte, ověřte jeho funkcionalitu a s pomocí vedoucího připravte na nasazení na server nerd.cesnet.cz.
4. Diskutujte dosažené výsledky a případné možnosti pokračování práce.

Literatura:

- Bartoš, Václav. (2019). NERD: Network Entity Reputation Database. ARES '19: Proceedings of the 14th International Conference on Availability, Reliability and Security. 1-7. 10.1145/3339252.3340512.
- Další literatura dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:
Splnění bodů 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Žádník Martin, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 17.5.2023
Datum schválení: 24.10.2022

Abstrakt

Táto diplomová práca sa zaoberá problematikou návrhu moderného webového rozhrania pre existujúci systém databázy nebezpečných sieťových entít. V práci je opísaný návrh nového používateľského rozhrania, ďalej sa práca venuje aj zmenám v existujúcom systéme NERD, ktorý bolo do značnej miery potrebné adaptovať potrebám nového webového rozhrania. Pri zmenách existujúceho systému sa stavalo na rozširovaní predošlej funkcionality v jazyku Python a frameworku Flask. Na implementáciu nového webového rozhrania bol využitý Vue.js (framework jazyka JavaScript). Táto práca predstavuje aj vylepšenú správu používateľov s podporou externých poskytovateľov identít. Výsledkom práce je použiteľné moderné webové rozhranie systému NERD, ktoré zachováva funkcionalitu pôvodného rozhrania a ďalej ho rozširuje.

Abstract

This thesis deals with the issue of designing a modern web interface for an existing system of a database of malicious network entities. The thesis describes the design of a new user interface and also addresses the changes in the existing NERD system, which had to be adapted to the needs of the new web interface. The changes to the existing system were based on expanding the previous functionality in the Python language and the Flask framework. Vue.js (a JavaScript framework) was used to implement the new web interface. This thesis also presents an improved user management with support for external identity providers. The final result of the thesis is a usable modern web interface for the NERD system, which preserves the functionality of the original interface and further extends it.

Kľúčové slová

webová aplikácia, používateľské rozhrania, NERD, nebezpečné sieťové entity, Python, Flask, JavaScript, Vue.js, správa používateľov, REST, externý poskytovatelia identít, OAuth, Perun

Keywords

web application, user interface, NERD, neferious network entities, Python, Flask, JavaScript, Vue.js, user management, REST, external identity providers, OAuth, Perun

Citácia

OLTMANOVÁ, Kristína. *Moderní webové rozhraní systému NERD.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádník, Ph.D.

Moderní webové rozhraní systému NERD

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracovala samostatne pod vedením pána doktora Martina Žadníka a pod technickým vedením pána doktora Václava Bartoša. Uviedla som všetky literárne zdroje, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Kristína Oltmanová
16. mája 2023

Poďakovanie

Na tomto mieste by som chcela poďakovať vedúcemu diplomovej práce, Ing. Martinovi Žádníkovi, PhD., za usmernenie a cenné rady pri písaní práce. Taktiež by som chcela poďakovať pánovi doktorovi Václavovi Bartošovi za pomoc pri technickej špecifikácii požiadaviek na nový systém, ako aj pri jeho nasadzovaní a testovaní.

Obsah

1 Úvod **5**

2 Zhrnutie doterajšieho stavu **6**
2.1 Systém NERD 6
2.1.1 Architektúra systému NERD 7
2.1.2 Profily sieťových entít 7
2.1.3 Využívané technológie 9
2.2 Aktuálna funkcionalita systému NERD 10
2.2.1 Ponúkané endpointy aplikačného programového rozhrania 11
2.2.2 Zdroje získavania dát o nebezpečných internetových entitách 11
2.3 Moderné webové rozhrania 13
2.3.1 Najpopulárnejšie webové frameworky jazyka JavaScript 13

3 Analýza existujúceho a návrh nového systému NERD **16**
3.1 Analýza aktuálneho webového rozhrania 16
3.1.1 Hlavná stránka 16
3.1.2 Vrchná lišta 18
3.1.3 Prihlasovanie 19
3.1.4 Detail IP adresy 19
3.1.5 Zobrazovanie na mobilných zariadeniach 24
3.2 Analýza existujúceho systému NERD 25
3.2.1 Existujúci frontend systému NERD vo Flasku 25
3.2.2 Existujúci backend systému NERD vo Flasku 26
3.2.3 Existujúce aplikačné programové rozhranie 26
3.3 Návrh nového webového rozhrania systému NERD 29
3.3.1 Analýza webových rozhraní poskytovateľov zdrojov 29
3.3.2 Rozloženie hlavnej stránky 30
3.3.3 Nová vrchná lišta 32
3.3.4 Stránky pre prihlasovanie a vytváranie profilu 33
3.3.5 Stránka s detailami IP adresy 34
3.3.6 Zobrazovanie na mobilných zariadeniach 35
3.4 Návrh nového systému 36
3.4.1 Nový frontend vo frameworku Vue.js 36
3.4.2 Aktualizovaný backend vo frameworku Flask 37
3.4.3 Nové aplikačné programové rozhranie 37

4 Implementácia nového webového rozhrania a systému NERD **38**
4.1 Prvá implementačná fáza 39

4.1.1 Frontend moderného webového rozhrania vo Vue.js 39
4.1.2 Práca vo frameworku Vue.js 42
4.1.3 Získavanie dát z existujúceho aplikačného programového rozhrania . 42
4.2 Druhá implementačná fáza 43
4.2.1 Backend moderného webového rozhrania vo frameworku Flask . . . 44
4.2.2 Nové aplikačné programové rozhranie vo frameworku Flask 45
4.3 Lokálna správa používateľov 47
4.3.1 Databáza používateľov v PostgreSQL 47
4.3.2 Ukladanie session používateľa 48
4.3.3 Registrácia používateľov a potvrdenie emailovej adresy 51
4.3.4 Rozlišovanie poskytovateľov identít a rozdielne prístupy k prihlasovaniu 52
4.4 Nová rozšírená funkcionalita systému NERD 53
4.4.1 Formátovanie emailov pre používateľov 53
4.4.2 Konzola pre admina 53
4.4.3 Mini grafy histórie reputačného skóre 54
4.4.4 Zobrazenie prítomnosti na blacklistoch za posledných tridsať dní . . 55
4.5 Prihlasovanie pomocou externých poskytovateľov identít 56
4.5.1 Google 56
4.5.2 Twitter 58
4.5.3 GitHub 59
4.5.4 EduGain 60

5 Testovanie **63**
5.1 Nasadenie implementácie na testovací server Apache 63
5.1.1 Požiadavka zachovania existujúcej funkcionality 63
5.1.2 Kompilácia súborov frontendu 63
5.2 Testovanie výkonnosti frontendu 64
5.3 Testovanie responzivity a použiteľnosti na mobilných zariadeniach 66
5.3.1 Používateľské testovanie webového rozhrania 66
5.4 Testovanie zabezpečenia nového aplikačného programového rozhrania 68
5.5 Možné budúce rozšírenia systému NERD 68

6 Záver **70**

Literatúra **71**

Zoznam obrázkov

2.1 Architektúra systému NERD prevzatá z [2] 7
2.2 Popularita frameworkov JavaScript za posledných 5 rokov získaná z dát Google Trends [11] dňa 09.11.2022 14

3.1 Hlavná stránka v pôvodnom dizajne 17
3.2 Vyhľadávací formulár pre jednu a viac IP adries 17
3.3 Ukážka vysvetlivky s informáciami k políčku kategórie Hostname suffix . . 18
3.4 Ukážka typov používateľských vstupov vo vyhľadávacom formulári 18
3.5 Vrchná lišta v pôvodnom dizajne 19
3.6 Ukážka výsledkov dvoch možností prihlásenia lokálny Log in (vľavo) a prihlasovanie cez EduGain (vpravo) 19
3.7 Detail IP adresy v pôvodnom dizajne 20
3.8 Prehľad aktuálnej IP adresy v pôvodnom dizajne 21
3.9 Dodatočné informácie zobrazené po kliknutí na farebný odkaz CI Army . . 21
3.10 Ukážka dodatočných informácií z externých zdrojov 22
3.11 Tabuľka s podrobnosťami o IP adrese 23
3.12 Ukážka grafov aktuálnej IP adresy pre Warden, DShield a prítomnosť na čiernych listinách 24
3.13 Ukážkový príklad používateľského rozhrania hlavnej stránky Greynoise . . . 29
3.14 Možné rozloženia vyhľadávacieho formulára a tabuľky výsledkov 30
3.15 Nový návrh hlavnej stránky 31
3.16 Ukážka stĺpca s IP adresami 31
3.17 Ukážka vyhľadávacieho poľa pre IP adresy a nápovedy pred zadávaním . . . 32
3.18 Ukážka vyhľadávania krajiny spomedzi možností (vľavo) a ukážka výberu viacerých zdrojov naraz (vpravo) 32
3.19 Vrchná lišta pre neprihláseného používateľa 33
3.20 Nastavenia časových dát 33
3.21 Formulár na prihlasovanie existujúceho používateľa 33
3.22 Formulár na vytvorenie profilu pre nového používateľa 34
3.23 Stránka s detailami pre IP adresu 35
3.24 Ukážka zobrazenia na mobilných zariadeniach: hlavná stránka (vľavo), mobilné menu (v strede) a detail IP adresy (vpravo) 36

4.1 Dve fázy implementácie 38
4.2 Základná štruktúra súborov zdrojového kódu 40
4.3 Routing – spájanie URL adries s príslušnými komponentami 41
4.4 Naznačenie problémovej komunikácie CORS medzi webovým prehliadačom a BE (backend) serverom 43
4.5 Nový návrh systému s podrobnou štruktúrou zmien backendu 44

4.6 Ukážka konzoly pre admina 54
4.7 Ukážka umiestnenia mini grafu zobrazujúceho históriu reputačného skóre . 55
4.8 Ukážka grafického zobrazenia prítomnosti IP adresy na blacklistoch za posledných tridsať dní 55
4.9 Nákres komunikácie NERD aplikácie a Google API pri prihlasovaní používateľa 57
4.10 Nákres komunikácie NERD aplikácie a Twitter API pri prihlasovaní používateľa 59
4.11 Nákres komunikácie NERD aplikácie a GitHub API pri prihlasovaní používateľa 60
4.12 Nákres komunikácie NERD aplikácie a Perun API pri prihlasovaní používateľa 61

5.1 Výkonnostná analýza hlavnej stránky 65
5.2 Časová analýza získania odpovede z REST API 65

Kapitola 1

Úvod

S rastúcou popularitou Internetu a možnosťou pripojiť sa naň pre čoraz väčšie skupiny obyvateľstva môžeme dnes konštatovať, že svet sa premiestnil do online priestoru. Avšak, priniesol tam so sebou všetko, čo svet vo svojej podstate zahŕňa, a teda aj to negatívne. Podvody, útoky, pokusy o odcudzenie finančných prostriedkov alebo identity – všetko známe a neželané činnosti z reálneho sveta. V súčasnosti sa tieto zločiny čoraz viac objavujú aj v digitálnom svete. Rovnako ako polícia identifikuje zločincov na základe ich údajov (meno či adresa bydliska), vieme aj v online priestore nebezpečné entity identifikovať a zhromažďovať o nich údaje (hostiteľské mená a IP adresy).

Systém NERD (Network Entity Reputation Database), ktorý je predmetom diplomovej práce sa venuje práve zhromažďovaniu informácií o nebezpečných internetových adresách. Inak povedané, poskytuje zoznamy stránok, ktorým už neveríme a radšej by sme sa im mali vyhnúť. Zoznamy si vytvára na základe hlásení z rôznych zdrojov, ktoré spája a prehľadne zobrazuje vo webovom rozhraní. Tento systém je medzinárodne využívaný napríklad spoločnosťami CSIRT a CERT a je vyvíjaný spoločnosťou CESNET.

Cieľom diplomovej práce je vytvorenie a implementácia moderného webového rozhrania pre existujúci systém NERD. V súčasnosti je tento systém využívaný primárne akademickými pracovníkmi, pre ktorých ponúka aj možnosti prihlasovania, a teda prístupu k rozšírenej funkcionalite. Plánovaným rozšírením, ktoré táto diplomová práca ponúka, je sprístupnenie systému širšej verejnosti pomocou využitia aj neakademických foriem prihlasovania (napr. Google, Twitter, GitHub). Okrem toho sa práca zameriava najmä na dizajnérske a návrhové úpravy, ktoré by mali prispieť k modernizácii a celkovému zlepšeniu práce so systémom pre existujúcich, ale aj nových používateľov.

K výberu témy diplomovej práce ma motivoval záujem o vytváranie moderných a pre používateľa prívetivých webových rozhraní s využitím súčasných trendov. Považujem za prínosné zobrať existujúci systém, analyzovať ho a na základe odvodených zistení priniesť jeho vylepšenia. Nakoľko sú technológie využívané súčasným webovým rozhraním systému NERD pomerne zastaralé, je práve tento systém vhodným kandidátom na modernizáciu a poskytuje zaujímavú dizajnérsku aj programátorskú výzvu.

Práca sa v úvode venuje zhrnutiu doterajšieho stavu systému NERD a predstavuje jeho časti. Následne je poskytnutý prehľad moderných frameworkov jazyka JavaScript. V ďalších častiach sa práca zaoberá analýzou aktuálneho riešenia a návrhom nového webového rozhrania, ako aj návrhom nadstavby nad existujúcim systémom. Následne je opísaná samotná implementácia nového používateľského rozhrania a systému. V závere sú objasnené kroky testovania výslednej implementácie.

Kapitola 2

Zhrnutie doterajšieho stavu

Doterajší stav problematiky riešenej v diplomovej práci sa skladá zo špecifikácie systému NERD. Tento systém v celej svojej komplexnosti obsahuje rozsiahle časti a moduly. Táto kapitola upozorňuje na webové celky tohto systému, ktoré budú dôležité pre pochopenie ďalších častí práce.

Ďalej je v tejto kapitole predstavená súčasná funkcionalita systému NERD, ktorú bude potrebné v ďalších častiach práce podrobne analyzovať a pretaviť návrhu ju do nového systému.

Na záver tejto kapitoly sú všeobecne predstavené základné prístupy a používané frameworky pre tvorbu moderných webových rozhraní.

2.1 Systém NERD

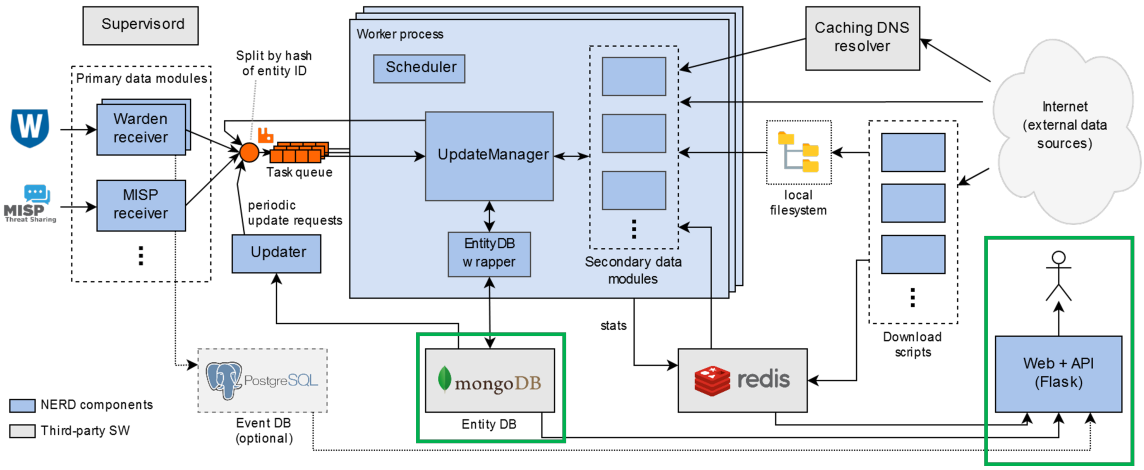
Systém pod skratkou NERD (Network Entity Reputation Database), teda v preklade reputačná databáza sieťových entít (ďalej len NERD), je softvér a služba, ktorá získava, ukladá a spracováva rôzne dáta o známych a nebezpečných sieťových entitách (reprezentované IP adresami), a ponúka používateľom pohľad na tieto zhromaždené dáta prostredníctvom webového rozhrania. IP je skratka pre Internet protocol.

Softvér je postavený na dostatočne všeobecnej platforme s modulárnou architektúrou, ktorá umožňuje jednoduché pridávanie modulov pre viacero zdrojov dát.

Hlavná inštancia systému NERD sa snaží o zhromaždenie čo najväčšieho množstva informácií o nebezpečných IP adresách a ich sprostredkovanie pre širšiu komunitu zaoberajúcu sa kybernetickou bezpečnosťou.

Tento systém je vyvíjaný a prevádzkovaný spoločnosťou CESNET a tímom Liberouter.

2.1.1 Architektúra systému NERD



Obr. 2.1: Architektúra systému NERD prevzatá z [2]

V grafickom zobrazení architektúry na obrázku 2.1 je možné vidieť časti systému zaoberajúce sa najmä zhromažďovaním a aktualizovaním dát z rôznych externých zdrojov. Pre oblasť záujmu diplomovej práce sú najdôležitejšie súčasti Entity DB a Web + API (Flask), v obrázku vyznačené zeleným rámčekom. API je skratka pre Application Programming Interface, teda pre aplikačné programové rozhranie. Databáza entít (Entity DB) uchováva záznamy profilov sieťových entít, ktoré sú bližšie predstavené v nasledujúcej podkapitole. Databáza využíva technológiu NoSQL databáz, konkrétne databázu MongoDB, ktorá je bližšie predstavená v podkapitole 2.1.3.

Web + API (Flask) predstavujú súčasné webové rozhranie, ktoré využíva interné funkcie systému na zobrazovanie dát používateľom. Rozhranie taktiež ponúka REST (Representational state transfer) API pre registrovaných používateľov. Webové rozhranie aj ponúkaná API využívajú Flask, nadstavbovú knihovňu jazyka Python, ktorá je bližšie špecifikovaná v podkapitole 2.1.3.

2.1.2 Profily sieťových entít

Systém NERD, ako už aj z preloženého názvu (reputačná databáza sieťových entít) vyplýva, slúži najmä na uchovávanie jednotlivých záznamov o nebezpečných a škodlivých sieťových entitách (IP adresách). Za nebezpečné sieťové entity považujeme tie, ktoré boli nahlásené ako pôvodcovia nejakého sieťového útoku alebo nebezpečnej aktivity. Systém NERD sleduje takéto nahlásenia z rôznych externých zdrojov, ukladá si ich do svojej databázy a kombinuje ich s cieľom vytvorenia profilu sieťovej entity. Väčšina dát je periodicky obnovovaná. Webové rozhranie umožňuje získanie detailu každej individuálnej IP adresy a aj vyhľadávanie IP adries podľa zadaných kritérií. [1]

Ukladané atribúty sieťových entít

Základné evidované atribúty o každej IP adrese sú:

- IP adresa,
- meno domény,

- ASN – autonómne systémové meno a číslo,
- krajina pôvodu,
- počty udalostí,
- reputačné skóre – bližšie špecifikované v nasledujúcej podkapitole,
- čierne listiny,
- zdroje,
- čas pridania,
- posledná aktivita.

Reputačné skóre

Informácie v tejto podkapitole boli získané z [3]. Všetky získané informácie o každej IP adrese sú využité k výpočtu reputačného skóre, teda čísla, ktoré reprezentuje mieru hrozby, ktorú táto IP adresa predstavuje. V súčasnosti sa na výpočet tohto skóre využíva jednoduchý vzorec.

Výpočet berie do úvahy počet udalostí a počet zdrojov, ktoré rovnakú IP adresu nahlásili za posledných 14 dní.

Algoritmus výpočtu:

1. Pre každý z posledných štrnástich dní vypočítaj:
   - `events(d)` – počet udalostí nahlásených zdroju Warden (bližšie špecifikovaný v 2.2.2), ktoré uvádzajú danú IP adresu ako zdroj,
   - `nodes(d)` – počet nezávislých uzlov (detektorov), ktoré nahlásili danú udalosť,
   - potom *denné reputačné skóre* je:

$$rep(d) = \left(1 - \frac{1}{2}^{events(d)}\right) \cdot \left(1 - \frac{1}{2}^{nodes(d)}\right) \tag{2.1}$$

2. Finálne reputačné skóre získame výpočtom váženého priemeru posledných 14 dní s lineárne klesajúcou váhou (najbližší deň má najvyššiu váhu):

$$rep = \frac{\sum_{d=0}^{13} rep(d) \cdot \frac{14-d}{14}}{7.5} \tag{2.2}$$

FMP skóre

Skratka FMP znamená *Future Misbehavior Probability*, teda pravdepodobnosť budúcej nebezpečnej aktivity. FMP skóre predstavuje odhad pravdepodobnosti, že daná IP adresa alebo iná entita bude v určenom budúcom časovom intervale nahlásená ako škodlivá. Odhad tejto pravdepodobnosti využíva model strojového učenia a snaží sa využívať čo najväčšie množstvo dostupných dát o danej entite. Budúci časový interval, pre ktorý sa táto pravdepodobnosť počíta je obvykle nastavený na 24 hodín.

Pri výpočte FMP skóre sa berie do úvahy:

- počet hlásení,

- celkové množstvo nahlásených útokov,
- počet zdrojov, ktoré entitu nahlásili,
- čas od posledného nahlásenia,
- priemer a medián intervalov v histórii predošlých útokov.

2.1.3 Využívané technológie

Systém NERD je postavený na zhromažďovaní dát o nebezpečných internetových entitách a o prezentovaní získaných dát. V súčasnej podobe je hlavným využívaným programovacím jazykom jazyk Python vo verzii 3.2. Sú na ňom postavené zhromažďovacie procesy, ktoré komunikujú s externými systémami a v pravidelných intervaloch obnovujú dáta v databáze.

Databázu tvorí moderná a pružná MongoDB s podporou NoSQL dopytov, ktorá udržuje dáta v kolekciách a umožňuje rýchly prístup a filtrovanie. Základ systému tvorený kombináciou MongoDB databázy a jazyka Python dáva zmysel aj pre budúcu modernosť a uplatnenie systému. Nedostatok a hlavná oblasť vylepšení, ktorými sa diplomová práca zaoberá sú práve prezentácia dát používateľovi, ktorú zabezpečuje frontend. Ten je v súčasnosti takisto implementovaný pomocou jazyka Python (statické HTML stránky sú vytvárané pomocou frameworku Flask). HTML je skratka pre Hyper Text Markup Language. Spomínané technológie budú bližšie predstavené v ďalších podkapitolách.

Pre možnosť využitia moderného webového systému na implementáciu nového používateľského rozhrania bude takisto potrebné odkryť a zapuzdriť viac endpointov API, ktorá v súčasnej podobe podporuje len základné dopyty.

Flask

Flask je ľahký WSGI (Web Server Gateway Interface) webový aplikačný framework. Je navrhnutý pre rýchle a jednoduché rozbehy aplikácií, pričom ponúka možnosť škálovateľnosti až po komplexné aplikácie. Spočiatku predstavoval len jednoduchú nadstavbu nad modulmi Werkzeug a Jinja, odvtedy sa však stal jedným z najpopulárnejších webových aplikačných frameworkov jazyka Python.

Flask ponúka návrhy, ale nevynucuje žiadne závislosti alebo projektové šablóny. Je na vývojároch, aby si sami vybrali nástroje a knižnice, ktoré chcú používať. K dispozícii je množstvo doplnkov a knižníc a pridávanie nových balíčkov je jednoduché. [16]

MongoDB

MongoDB je open-source dokumentačná databáza, ktorá je najpoužívanejšou spomedzi všetkých NoSQL databáz. Je napísaná v jazyku C++. Jej výhodou je škálovateľná architektúra a podpora širokého množstva aplikácií aj s meniacimi sa dátovými schémami.

Ako dokumentačná databáza umožňuje vývojárom uchovávať štruktúrované aj neštrukturované dáta. Na ukladanie týchto dát používa formát podobný formátu JSON (skratka pre JavaScript Open Notation). Tento formát je priamo kompatibilný s väčšinou súčasných moderných programovacích jazykov, čo z MongoDB robí jasnú voľbu pre mnohých vývojárov. Použitím tejto databázy sa totiž môžu vyhnúť potrebe normalizovať dáta.

MongoDB je navyše schopná uložiť veľké množstvá dát a je škálovateľná vertikálne aj horizontálne. [13]

REST API

REST je skratka pre Represantional State Transfer. API je skratka pre Application programming interface. Ich kombináciou získavame cestu, ako jednoducho čítať, mazať a aktualizovať informácie na serveri za využitia jednoduchých HTTP (Hypertext Transfer Protocol) dopytov. REST API prvýkrát navrhol počítačový vedec Roy Fielding.

Samotná API predstavuje vrstvu medzi zdrojom informácií a používateľom, pomáha nám pristupovať k prostriedkom systému pomocou dopytov a odpovedí.

V kontexte dizajnu moderných webových aplikácií spravuje REST API všetky úkony spojené s dodávaním dát (priame výpisy alebo komplikovanejšie agregácie z databáz). [18]

Základné pravidlá, ktorá musia byť dodržané pri implementácii REST API [17]:

1. Prijímať dáta a odpovedať na ne vo formáte JSON. Práve tento formát je ideálny na prenos informácií z MongoDB do jazyku JavaScript a opačne.
2. Používať chybové kódy pre uľahčenie hľadania prípadných chýb a takisto správnu interpretáciu na frontende.
3. Nepoužívať v odkazoch URL (Uniform Resource Locator) prídavné mená, používať len podstatné mená.
4. V databázach využívať plurály podstatných mien na pomenovania kolekcií. Kolekcie v MongoDB potom priamo odpovedajú príslušným endpointom.
5. Dobre zdokumentovať, využiť napríklad OpenAPI.
6. K chybovému hláseniu pridať podrobnosti do tela odpovede, a tým uľahčiť hľadanie chýb v kóde, aby sme vedeli presne identifikovať, čo sa stalo.
7. Využívať vnáranie zdrojov:
   - `/users` – výpis všetkých používateľov,
   - `/users/123` – výpis špecifického používateľa,
   - `/users/123/orders` – výpis všetkých objednávok špecifického používateľa,
   - `/users/123/orders/0001` – výpis konkrétnej objednávky špecifického používateľa.
8. Využívať Transport Layer Security alebo Secure Sockets Layer na šifrovanie komunikácie.
9. Zabezpečiť API s využitím HTTP Strict Transport Security (HSTS) policy.

2.2 Aktuálna funkcionalita systému NERD

V prechádzajúcej kapitole sme sa zamerali na systém NERD v celej jeho komplexnosti. V tejto kapitole je zhrnutá aktuálna funkcionalita systému NERD z pohľadu používateľského rozhrania. Systém v aktuálnej podobe ponúka:

- zobrazenie dát z databázy nebezpečných sieťových entít (najmä IP adries),
- zobrazenie prepojení a dát z externých bezpečnostných systémov, ktoré konkrétnu IP adresu evidujú,

- jednoduchú evidenciu používateľov a prihlasovanie cez EduGain,
- vyhľadávanie, filtrovanie a zoraďovanie hlavnej prehľadovej tabuľky s IP adresami,
- možnosť vyhľadávania viacerých IP adries naraz,
- možnosť stiahnutia tabuľky s výsledkami vo formáte CSV,
- zobrazenie detailov IP adries s dátovými výpismi a grafmi udalostí.

Jedným z cieľov diplomovej práce je vyššie uvedenú funkcionalitu zachovať a zlepšiť ju z hľadiska používateľského rozhrania.

2.2.1 Ponúkané endpointy aplikačného programového rozhrania

Na prístup k endpointom je potrebné mať vygenerovaný prístupový token, ktorý systém v súčasnej podobe generuje pre všetkých registrovaných používateľov. Prístupový token sa pri dopytoch zadáva do HTTP hlavičky v podobe: `Authorization: token <token>` alebo len `Authorization: <token>`. V súčasnej podobe sú implementované nasledujúce endpointy:

1. základné informácie o IP adrese:

   `GET: https://nerd.cesnet.cz/nerd/api/v1/ip/<ip_address>`,

2. detailné informácie o IP adrese:

   `GET: https://nerd.cesnet.cz/nerd/api/v1/ip/<ip_address>/full`,

3. reputačné skóre:

   `GET: https://nerd.cesnet.cz/nerd/api/v1/ip/<ip_address>/rep`,

4. FMP skóre:

   `GET: https://nerd.cesnet.cz/nerd/api/v1/ip/<ip_address>/fmp`,

5. vyhľadávanie IP adries:

   `GET: https://nerd.cesnet.cz/nerd/api/v1/search/ip/?<search_query>`,

6. reputačné skóre pre skupinu adries:

   `POST: https://nerd.cesnet.cz/nerd/api/v1/ip/bulk`,

   `body: <comma_separated_list>`.

Podrobnejšie informácie o vstupoch a výstupoch existujúcich API endpointov a ich analýza sú v podkapitole 3.2.3.

2.2.2 Zdroje získavania dát o nebezpečných internetových entitách

Nasledujúce informácie o jednotlivých zdrojoch a ich využití v systéme NERD bolo prevzaté z [1].

Warden

Warden je systém na zdieľanie upozornení, ktorý je prevádzkovaný spoločnosťou CESNET. Jeho úlohou je zbierať hlásenia z niekoľkých systémov, napríklad honeypots, analyzátorov toku dát a iných detektorov nebezpečných sieťových prevádzok, ktoré sú monitorované prevažne (ale nie výlučne) v českých akademických sieťach.

Denne je zdieľaných približne milión upozornení z niekoľkých desiatok detekčných systémov. Ku kompletným dátam upozornení zo zdroja Warden má prístup len uzavretá komunita ľudí. Verejnosti prístupné sú iba základné metadáta, ako napríklad počty upozornení podľa rôznych kategórií a IP adries. Tieto informácie sú verejnosti dostupné práve cez systém NERD.

Donedávna tvoril Warden jediný primárny zdroj dát pre systém NERD a niektoré atribúty stále neberú do úvahy prítomnosť iných detekčných systémov. Napríklad reputačné skóre je počítané čisto z dát zo systému Warden.

V súčasnosti sa pracuje na zovšeobecnení systému NERD, aby vo väčšej miere zohľadňoval aj ostatné nižšie spomenuté systémy. Tieto zmeny budú predstavovať zásah do vnútra samotného systému a z pohľadu používateľského rozhrania s nimi musíme počítať len ako s budúcimi rozšíreniami.

Blacklists

NERD využíva takmer 50 verejne dostupných čiernych listín, ktoré obsahujú plaintext zoznamy nebezpečných IP adries. Tieto listiny sú získavané z 20 rôznych zdrojov. Pre každú IP adresu, ktorá sa vyskytne na listine je vytvorený záznam v systéme NERD.

DShield

DShield je firewall korelačný systém založený na spolupráci v správe inštitútu SANS. Rozsiahla komunita prispievateľov tu zdieľa informácie o neočakávaných alebo neoprávnených prichádzajúcich pripojeniach. Verejnosti sú dostupné aj štatistiky o individuálnych portoch a IP adresách.

NERD využíva denný výpis všetkých IP adries nahlásených pomocou systému DShield a do svojej databázy pridá také IP adresy, ktoré boli nahlásené minimálne desaťkrát, a to aspoň tromi rôznymi zdrojmi.

AlienVault Open Threat Exchange (OTX)

OTX je open source komunitný portál, na ktorom počítačoví bezpečnostní výskumníci a profesionáli zdieľajú denne milióny indikácií hrozieb.

NERD odoberá niekoľko účtov a pulzov, ktoré boli vyhodnotené ako spoľahlivé a užitočné. Konkrétne účet AlienVault a niekoľko honeypot zdrojov je použitých k dopĺňaniu informácií do systému NERD, ktorý prehľadá v nich prítomné IP adresy a vytvorí nové alebo modifikuje už existujúce záznamy vo svojej databáze.

Malware Information Sharing Platform (MISP)

NERD je pripojený na rozsiahlu inštanciu systému MISP, ktorá je prevádzkovaná centrom hlásení počítačových incidentov v Luxembursku, CIRCL (The Computer Incident Response Center Luxembourg). NERD z tohto systému získava všetky IP adresy s označením `tlp:white`, teda verejne dostupné informácie.

Sekundárne zdroje

Každá IP adresa získaná z vyššie uvedených zdrojov je navyše obohatená o nasledujúce informácie nadobudnuté zo sekundárnych zdrojov:

- doménové meno – získané reverzným DNS dotazom,
- ASN – autonómne systémové číslo,
- informácie z `whois` (zdroja pre Domain Name System (DNS) informácie),
- geografická lokalizácia – získaná z databázy GeoLite2 od MaxMind,
- prítomnosť na sekundárnych čiernych listinách – listiny dostupné len cez DNSBL alebo tie, ktoré neposkytujú zoznamy nebezpečných IP adries (napríklad zoznam výstupných uzlov TOR),
- pasívny DNS systém spoločnosti CESNET,
- rôzne označenia opisujúce IP adresu – odvodené z iných informácií.

2.3 Moderné webové rozhrania

Trendy napredovania vývoja frontendu sa nedajú jednoducho predpovedať. Niekedy trvá roky, kým si vývojári osvoja konkrétne technológie. Napriek tomu vieme do budúcnosti konštatovať, že Single Page aplikácie, TypeScript a JavaScript budú v popredí. V súčastnosti väčšina frontendových projektov využíva frameworky jazyka JavaScript, ako React, Vue.js alebo Angular (viac o jednotlivých frameworkoch a rozdieloch medzi nimi v nasledujúcej podkapitole). [8]

JavaScript je v dnešnej dobe už konštantným trendom vo vývoji frontendu. Je zároveň jedným z najpopulárnejších a najžiadanejších programovacích jazykov, a taktiež sa predpokladá, že si túto poprednú priečku v najbližších rokoch udrží. [21]

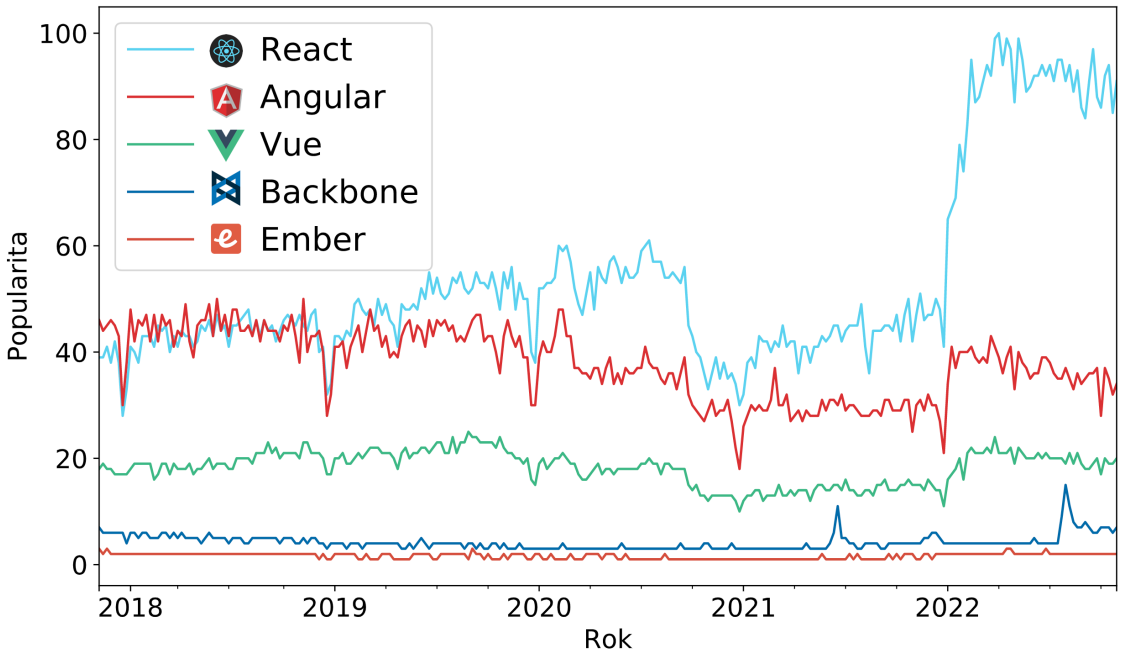
Takmer každá moderná stránka sa spolieha na použitie JavaScriptu. JavaScript sa neustále vyvíja a dopĺňajú sa nové funkcionality. V súčasnosti existuje 1.4 milióna rôznych knižníc jazyka JavaScript, ktoré sú vývojárom k dispozícii.

Svoju zásluhu na popularite JavaScriptu nepochybne nesú možnosti jeho frameworkov. Tieto frameworky značne zjednodušujú prácu a vedú k jednoduchému a rýchlemu vývoju, nakoľko sa vďaka ich požitiu vieme vyhnúť potrebe hľadania riešení na už vyriešené problémy.

Frameworky obvykle obsahujú sady znovu-použiteľných komponentov a widgetov, ktorých vkladaním je možné vytvoriť ľubovoľné webové aplikácie. Porovnaniu frameworkov JavaScriptu a toho, čo ponúkajú sa venuje nasledujúca podkapitola.

2.3.1 Najpopulárnejšie webové frameworky jazyka JavaScript

Na obrázku 2.2 sú zjavne viditeľné trendy preferencií jednotlivých webových frameworkov v priebehu času. V nasledujúcich odstavcoch sú bližšie predstavené a porovnané 3 najpopulárnejšie frameworky.



Obr. 2.2: Popularita frameworkov JavaScript za posledných 5 rokov získaná z dát Google Trends [11] dňa 09.11.2022

React

Prednosťami najpopulárnejšieho frameworku React sú jeho rýchlosť, jednoduchosť a schopnosť vytvárať pre používateľa prívetivé aplikácie optimalizované aj pre mobilné zariadenia. React navyše ponúka aj výrazné zlepšenia výkonu vďaka schopnosti blokovania updatov z DOM (Document Object Model), čo vo výsledku vedie k rýchlejšej a plynulejšej aplikácii.

React bol vyvinutý vývojármi firmy Facebook v roku 2011. Postupne si vybudoval pevnú používateľskú základňu, rozsiahlu ponuku nástrojov a aj komunitnú podporu. Revolučnou funkcionalitou je napríklad možnosť využitia concurrent módu, ktorý adaptívne prispôsobuje efektivitu výkonnostným možnostiam používateľského zariadenia.

Angular

Angular je známy pre svoju výkonnosť a robustnosť vyvíjaných riešení, ktoré ponúkajú responzívny a atraktívny vzhľad. Napriek jeho komplexnosti je často používaný pre solídne veľkoplošné podnikové riešenia.

Keď Google vydal Angular v roku 2010, priniesol tým revolúciu v tomto odvetví, čo programátori ocenili. V roku 2016 vyšiel Angular2, ktorý je od základov napísaný v jazyku TypeScript. Táto verzia prináša úplne novú architektúru, menšiu vstupnú bariéru aj vysoko kvalitnú dokumentáciu.

Ďalší dôležitý milník v jazyku Angular prišiel v roku 2021, kedy Angular dostal hneď niekoľko vylepšení vrátane Package formátu pre rýchlejšie vykonávanie, nezávislosť na vnútornej API, či zjednodušenie vytvárania dynamických komponentov z hľadiska využitia API.

Vue.js

Ďalším z populárnych frameworkov je Vue.js, ktorý sa už niekoľko rokov drží na popredných priečkach rebríčkov obľúbenosti. Vue.js je známy svojou modulárnou architektúrou, flexibilitou, prepracovanou dokumentáciou a rozsiahlou komunitou vývojárov. Je to framework, ktorý je pre vývojárov jednoduchý na osvojenie.

V roku 2014 Vue.js vyvinul bývalý zamestnanec spoločnosti Google. Tento framework je primárne preferovaný u čínskych technologických gigantov ako napríklad Alibaba, Xiaomi, ale aj spoločnosťou GitLab.

Vue.js je vhodný pre menšie až stredne veľké aplikácie. Veľká úvodná rýchlosť umožňuje vytvorenie MVC (Model View Controller) v krátkom čase. Škálovateľosť aplikácie vytvorenej pomocou Vue.js taktiež napomáha dynamickému vývoju projektu.

Vue.js vo verzii 3.0 poskytuje plnú podporu jazyku TypeScript. V auguste 2021 vyšla verzia 3.2 s viacerými vylepšeniami, ktoré by mohli ovplyvniť trendy frameworkov jazyka JavaScript v najbližšom období. Píši sa novým prekladačom tém (template complier), ktorý je schopný kompilovať statický obsah veľmi rýchlo. Zabudovaná kompozičná API uľahčuje organizáciu a znovu-použiteľnosť kódu, čo predstavuje benefit najmä pri väčších projektoch.

Kapitola 3

Analýza existujúceho a návrh nového systému NERD

Hlavnou úlohou diplomovej práce je modernizácia súčasného webového rozhrania systému NERD. Táto kapitola opisuje návrh dizajnu a technológie, ktoré boli k tejto modernizácii použité.

Najskôr sa analýza zameriava na vzhľad a rozloženie prvkov existujúceho frontendu. Následne sa systém NERD do hĺbky rozoberá a analyzuje sa, ako sa v súčasnosti jednotlivé stránky vykresľujú a zostavujú. Podrobnejšie je popísaná aj analýza ponúkaných endpointov API v1.

V tejto kapitole sú predstavené aj vylepšenia v podobe návrhu nového webového rozhrania systému NERD a celkovej novej vnútornej štruktúry backendu a API v1.

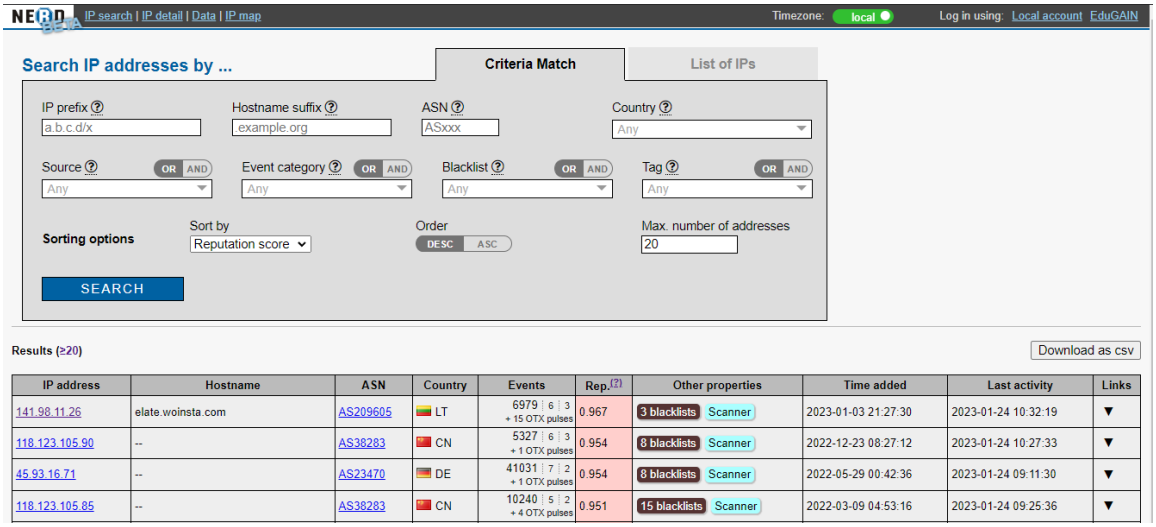
3.1 Analýza aktuálneho webového rozhrania

Aktuálna podkapitola poskytuje pohľad na pôvodný dizajn jednotlivých stránok webového rozhrania systému NERD. Stránky v tejto podobe sú tvorené klasickým HTML v kombinácii s príkazmi z jazyku Flask. Podrobnou analýzou existujúceho riešenia zachováme funkčné a dobre navrhnuté existujúce súčasti, a zároveň identifikujeme oblasti možných vylepšení, na ktoré sa neskôr pri tvorbe modernejšieho návrhu zameriame.

3.1.1 Hlavná stránka

Hlavná stránka vyobrazená na snímke obrazovky 3.1 ponúka sekciu vyhľadávania a tabuľku s výsledkami. Medzi výhody tohto dizajnu patrí dobré využitie šírky stránky, ktoré vedie k prehľadnému zobrazeniu veľkého množstva informácií v hlavnej tabuľke. Ak je ale cieľom stránky, aby využívala priestor obrazovky maximálne, potom nevyužitý priestor napravo od vyhľadávacieho formulára predstavuje nevýhodu.

V časti stránky pod vyhľadávacím formulárom je k dispozícii tabuľka s výsledkami, ktorá využíva kombináciu linkových odkazov, bežného textu, farebných označení a dátumov. Dátumy sú získavané vo formáte priamo z databázy bez dodatočného formátovania na frontende.

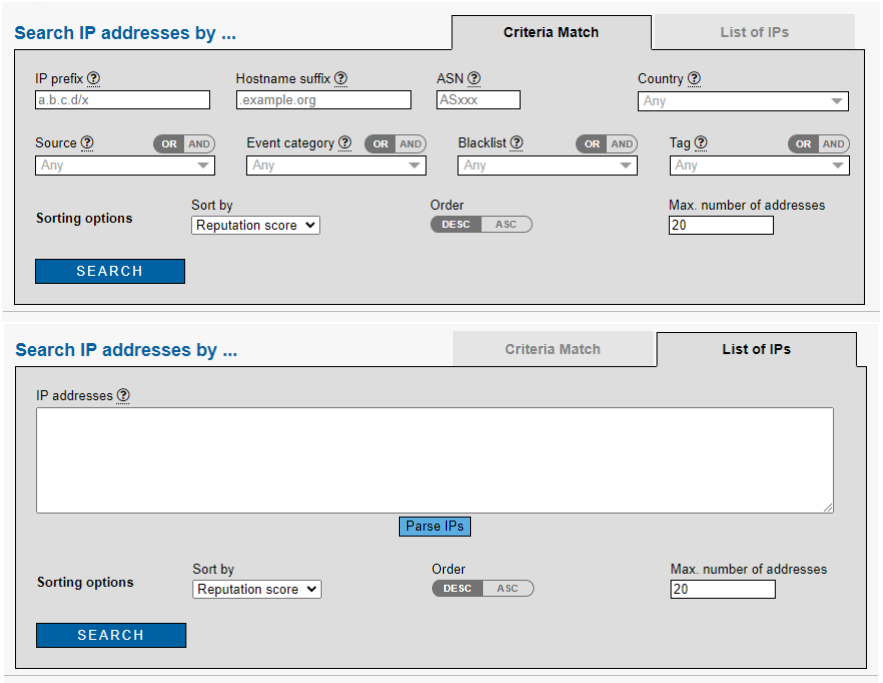


Obr. 3.1: Hlavná stránka v pôvodnom dizajne

Vyhľadávací formulár

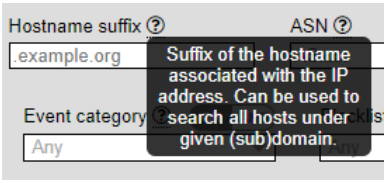
Samotný vyhľadávací formulár na obrázku 3.2 je tvorený dvomi prepínacími časťami. Vyhľadávanie môže byť buď pre jednu IP adresu alebo pre celú skupinu. Vyhľadávanie ponúka kombinácie parametrov pomocou logických operátorov AND/OR, ktoré vedú ku konjunkcii alebo disjunkcii výsledných množín podľa kategórií.

Pod kategóriou rozumieme jednotlivé vyhľadávacie polia, ktoré nesú nadpis a očakávajú nejaký konkrétny typ používateľského vstupu.



Obr. 3.2: Vyhľadávací formulár pre jednu a viac IP adries

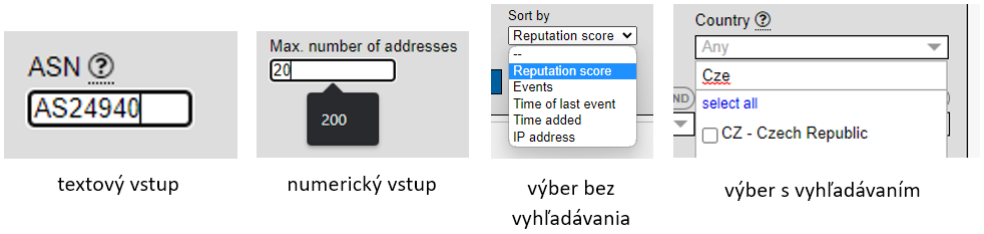
Všetky nadpisy kategórií obsahujú odkaz na vysvetlivku, ktorá používateľom bližšie špecifikuje danú kategóriu ako aj obsah, ktorý očakáva ako vstup od používateľa.



Obr. 3.3: Ukážka vysvetlivky s informáciami k políčku kategórie Hostname suffix

Vyhľadávací formulár obsahuje vstupné polia, ktoré očakávajú používateľský vstup. Typy používateľských vstupov na obrázkoch 3.4:

- textový vstup – jednoduchý textový vstup bez dynamickej syntaktickej kontroly,
- numerický (číselný) vstup – jednoduchý číselný vstup bez kontroly,
- výber bez vyhľadávania – pevne dané možnosti, výber jednej z nich,
- výber s vyhľadávaním – pevne dané možnosti, vyhľadávanie medzi nimi, ako aj podpora výberu viacerých možností.



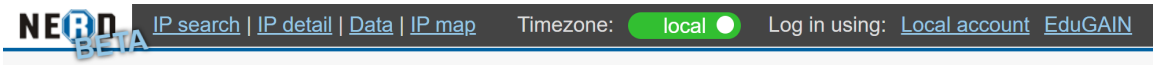
Obr. 3.4: Ukážka typov používateľských vstupov vo vyhľadávacom formulári

Rozmiestnenie samotného formulára v kombinácii s komplexnosťou rôznych vstupov môžu na používateľa pôsobiť príliš robustným dojmom.

3.1.2 Vrchná lišta

Vrchná lišta na obrázku 3.5 poskytuje hlavnú navigáciu a identitu stránky (pomocou líšt identifikujeme, na akej stránke sa aktuálne nachádzame), a v pôvodnom návrhu obsahuje štyri logické časti:

- logo stránky – úplne naľavo, poskytuje identitu stránke, a zároveň slúži ako návrat na hlavnú obrazovku,
- navigácia stránky – odkazy na existujúce podstránky,
- nastavenia časovej zóny – prepínač typu switch, ktorý umožňuje prestavenie zobrazovaného času z lokálneho (local) na UTC,
- možnosti prihlásenia – v zobrazenom pôvodnom riešení dve možnosti.



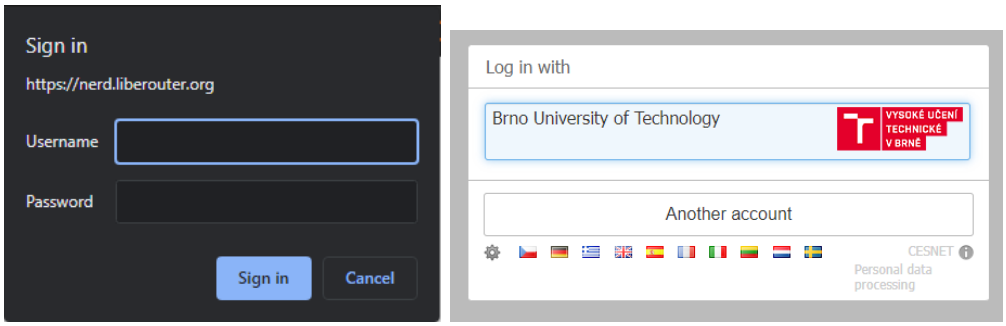
Obr. 3.5: Vrchná lišta v pôvodnom dizajne

3.1.3 Prihlasovanie

Prihlasovanie, ktoré je dostupné v pravej hornej časti stránky, ponúka dve možnosti. Prvá z nich, lokálny Log in, používateľov presmeruje na automatické modálne okno prehliadača, v ktorom je požadované prihlasovacie meno a heslo. Toto je jednoznačne zastaralá metóda získavania prihlasovacích údajov. Používateľ môže byť zmätený, nakoľko mu úplne zmizla identita stránky, chýbajú aj možnosti vytvoriť si účet, zobraziť alebo obnoviť heslo, mať kontrolu zadaných vstupov a zobrazovanie chybových hlášok o tom, čo je nesprávne vyplnené. Používateľ nie je nijak informovaný o tom, či účet neexistuje alebo či len zle zadal heslo.

Pri využití možnosti prihlásenia sa cez službu EduGain je používateľ presmerovaný na stránku, ktorá mu umožňuje výber akademickej inštitúcie, cez ktorú sa vie prihlásiť. Následne je na dokončenie procesu v prípade nového používateľa nutné manuálne povolenie administrátorom.

Obe možnosti sú vyobrazené na obrázkoch 3.6.



Obr. 3.6: Ukážka výsledkov dvoch možností prihlásenia lokálny Log in (vľavo) a prihlasovanie cez EduGain (vpravo)

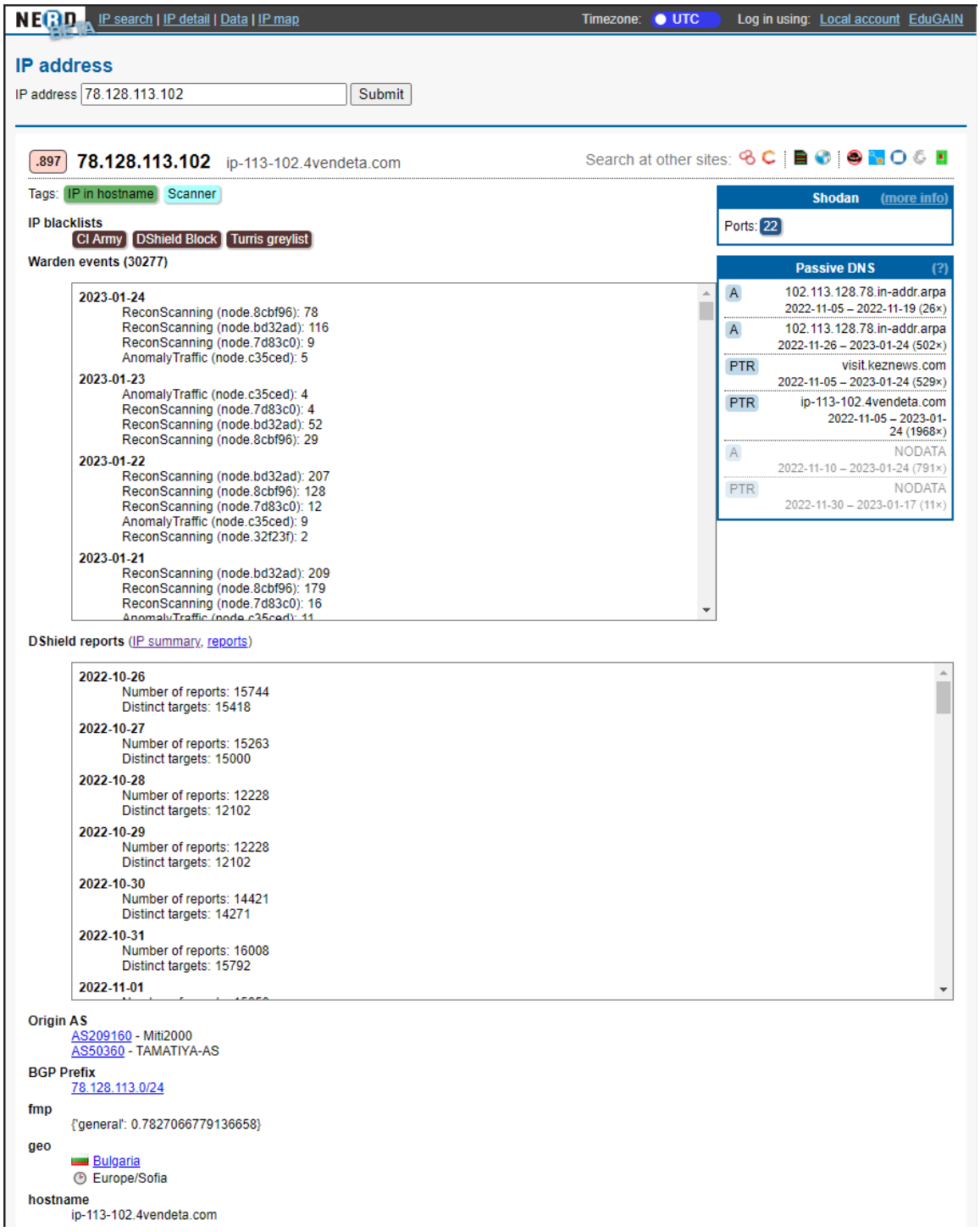
Môžeme konštatovať, že pôvodný návrh systému sa do veľkej miery spoliehal na manuálne zásahy od administrátora. Tieto zásahy museli byť realizované priamo do databázy nakoľko v pôvodnom návrhu chýba aj systémová administrátorská časť pre správu používateľov.

3.1.4 Detail IP adresy

Táto dôležitá stránka prestavuje komplexný detailný pohľad na IP adresu, ktorý je používateľovi ponúknutý ak klikne na niektorú z IP adries vo výsledkovej tabuľke na hlavnej stránke. Stránka s detailami zobrazuje tabuľky, ktoré obsahujú záznamy z rôznych zdrojov. Tabuľky majú pevnú výšku, nakoľko záznamov v nich môže byť veľa, a vtedy je potrebné stránku vertikálne posúvať. V spodnej časti stránky sa nachádzajú komplexné údaje o IP adrese a prehľadové grafy.

Podstránka zobrazujúca detaily IP adresy primárne využíva vertikálny priestor a dostať sa ku konkrétnej časti vyžaduje niekoľko vertikálnych posunov. Vo vylepšenej verzii návrhu

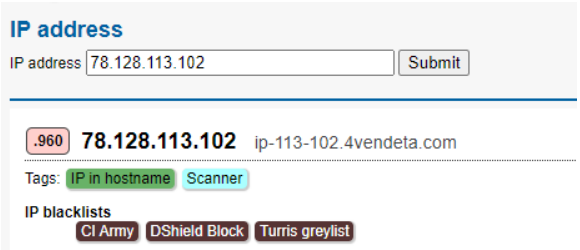
by bolo dobré aplikovať rozloženie typu dashboard s väčšou modularitou a možnosťami na prispôsobenie pre každého individuálneho používateľa. Snímka obrazovky z tejto stránky je obrázok 3.7.



Obr. 3.7: Detail IP adresy v pôvodnom dizajne

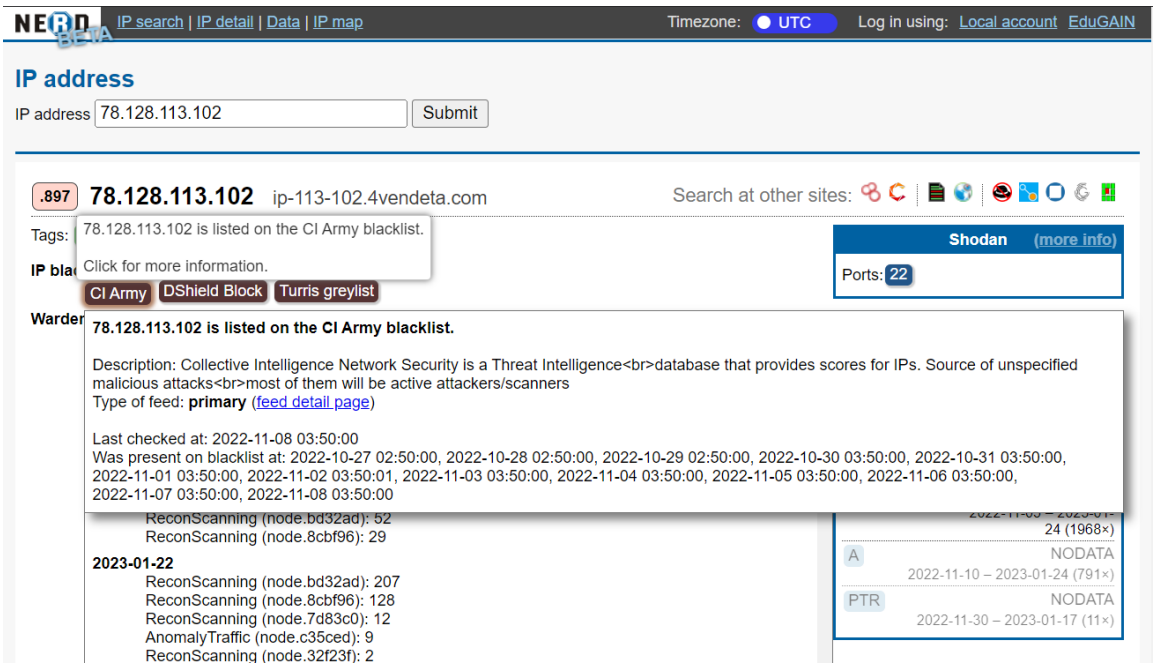
Krátky prehľad aktuálnej IP adresy

Prehľad na obrázku 3.8 je umiestený v pravom hornom rohu, a teda v mieste, kde používateľ začína skenovať stránku pohľadom. V tomto mieste špecifikujeme, aké má aktuálne zvolená IP adresa reputačné skóre, a aké tagy sa k nej viažu. Taktiež tu vidíme výpis čiernych listín, na ktorých daná IP adresa figuruje.



Obr. 3.8: Prehľad aktuálnej IP adresy v pôvodnom dizajne

Každý z okrúhlych farebných odkazov v sebe nesie dodatočné informácie, ktoré sú používateľovi k dispozícii po kliknutí myšou na odkaz. Takto zobrazené informácie zobrazujú bližší popis čiernej listiny a výpis posledných dátumov, v ktorých bola aktuálna adresa na aktuálne zobrazovanej čiernej listine prítomná. Zvýraznenie a zobrazenie podrobností blacklistu CI Army je možné vidieť na obrázku 3.9.



Obr. 3.9: Dodatočné informácie zobrazené po kliknutí na farebný odkaz CI Army

Dodatočné informácie z externých zdrojov

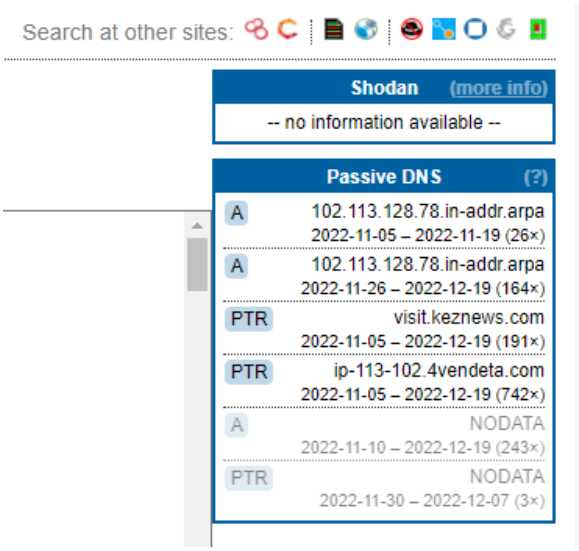
V pravej časti stránky sa nachádzajú odkazy na externé služby, ktoré aktuálnu IP adresu tiež evidujú. Pri týchto odkazoch sa využíva fixné presmerovanie podľa IP adresy, keďže

všetky zdroje ponúkajú detail aktuálnej adresy v nejakej nimi definovanej URL forme: `URL_zdroj/aktualna_IP`.

Externé zdroje sú nasledujúce:

- Shodan,
- Censys,
- Valli,
- AbuseIPDB,
- Talos Intelligence Reputation Center,
- GreyNoise Visualizer,
- DShield.

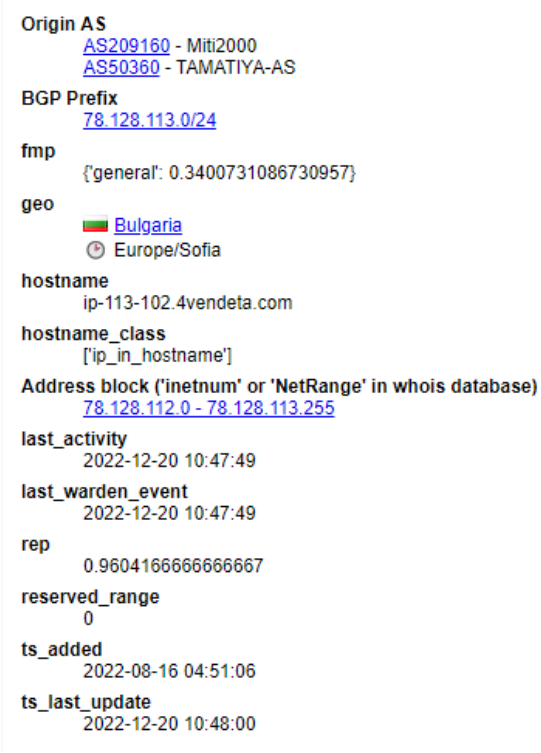
V nižšej časti tejto sekcie zobrazujeme podrobnejšie informácie zo zdroja Shodan, ak sú k dispozícii, a verejne dostupné pasívne DNS záznamy. Tieto informácie sú viditeľné na obrázku 3.10.



Obr. 3.10: Ukážka dodatočných informácií z externých zdrojov

Tabuľka s podrobnosťami

Tabuľka s podrobnosťami o danej IP adrese je umiestnená pod tabuľkami zdrojov vo vrchnej časti stránky a nad grafmi v spodnej časti stránky. Sú tu zobrazené databázové hodnoty v tvare kľúč-hodnota, pričom môžeme vidieť, že niektoré sú dostatočne naformátované pre používateľské rozhranie (napríklad krajina). Iné sú zas na prvý pohľad v nenaformátovanom tvare priamo z databázy (napríklad FMP skóre), čo kazí celkový dojem tejto tabuľky.



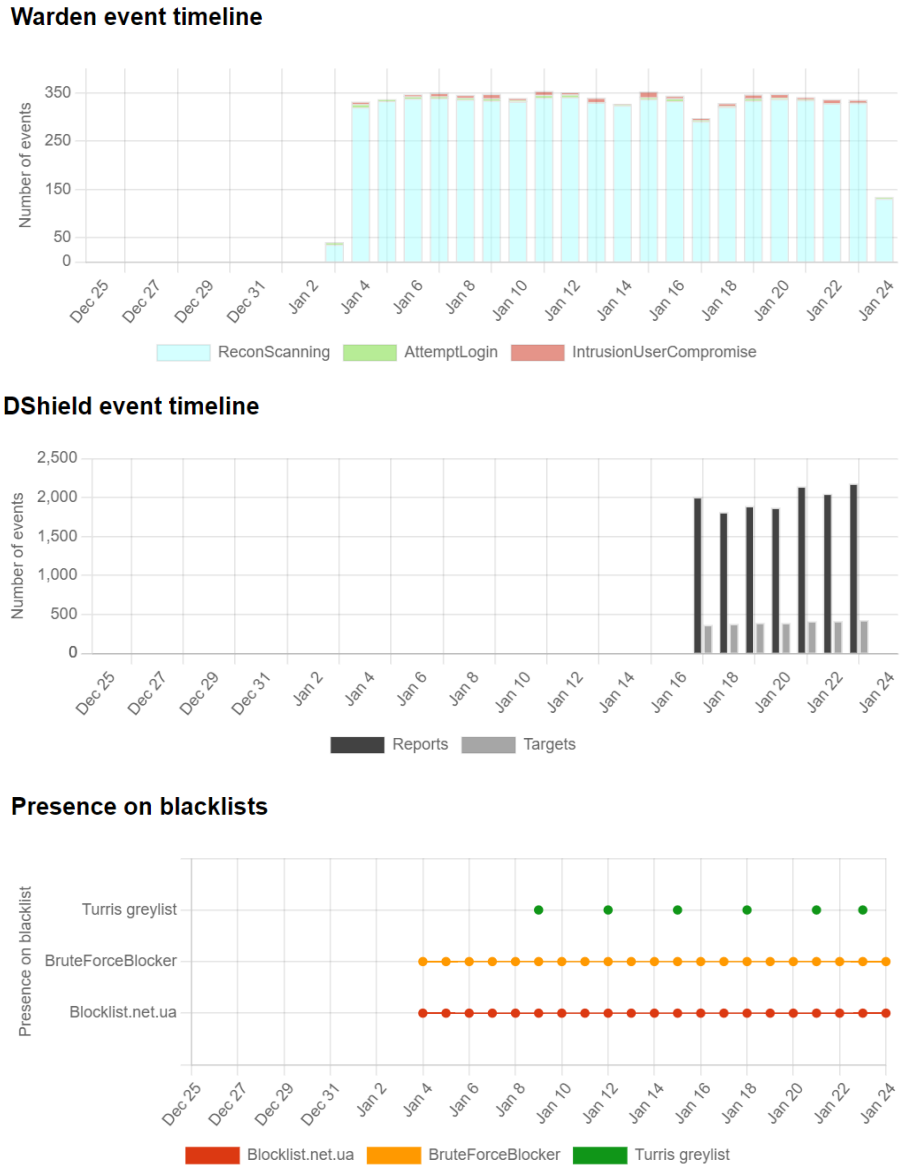
Obr. 3.11: Tabuľka s podrobnosťami o IP adrese

Grafy

Grafy sú v súčasnom riešení vykresľované s využitím knižnice Chart.js vo verzii 3, ktorá ich vytvára nad čistým jazykom JavaScript. Jedná sa o dva stĺpcové kombinované grafy a jeden čiarový graf. Všetky grafy majú legendu, dynamické nápovedy pri bodoch a zobrazujú dáta v rovnakej časovej osi za posledných 30 dní. Grafy sú vyobrazené na obrázkoch 3.12.

Prvý graf zobrazuje udalosti zo zdroja Warden za posledných 30 dní. Na druhom grafe nájdeme počty pre hlásenia a ciele získané zo zdroja DShield. Posledný graf zobrazuje prítomnosť danej IP adresy na čiernych listinách.

Všetky grafy sa snažia o zachovanie rovnakej alebo aspoň podobnej časovej osi.



Obr. 3.12: Ukážka grafov aktuálnej IP adresy pre Warden, DShield a prítomnosť na čiernych listinách

3.1.5 Zobrazovanie na mobilných zariadeniach

Hlavnou nevýhodou existujúceho návrhu používateľského rozhrania je, že bol od začiatku stavaný na čistom HTML a statických štýloch. Toto existujúce rozhranie by sa len ťažko podarilo optimalizovať bez väčších zásahov alebo kompletne odlišného rozloženia pre mobilné zariadenia. Pri novom návrhu budeme musieť rátať s množstvom v súčasnosti využívaných zariadení a adaptovať zobrazovacie prvky tak, aby boli prispôsobiteľné.

Súčasťou poskytovanej funkcionality sú aj nástroje pre admina – Admin box, ktorého funkcionalitu je potrebné zachovať aj v novom návrhu.

3.2 Analýza existujúceho systému NERD

V ďalšom priebehu analýzy od seba logicky oddelíme časti systému, aj keď v samotnej implementácii sú veľmi úzko prepojené. Budeme samostatne analyzovať frontend, teda časti kódu zaoberajúce sa vykresľovaním webových stránok. Následne v ďalšej časti si predstavíme súčasný backend, ktorý vykonáva dopyty do databázy s nebezpečnými internetovými entitami a aj do databázy používateľov. V neposlednom rade analyzujeme aj API v1, teda implementačnú časť poskytujúcu základné endpointy na dopyty do databáze nebezpečných internetových entít.

3.2.1 Existujúci frontend systému NERD vo Flasku

Aplikácia využívajúca Flask framework pozostáva z dekorovaných funkcií pre jednotlivé stránky (URLs). Každá stránka, ktorú chceme zobraziť je definovaná vlastnou funkciou, ktorá na záver poskytne návratovú hodnotu s využitím `render_template`. Všetky logické úkony a predprípravа dát na zobrazenie na HTML stránke prebieha v tele týchto funkcií.

Následné naformátovanie a špecifikácia zobrazovaných dát prebieha v šablónových súboroch `templates`, ktoré v sebe okrem HTML kódu obsahujú aj príkazy a hodnoty jazyka Python v špeciálnych oddeľovacích značkách (`{% <Python príkaz> %}`) pre príkazy alebo (`{{ <Python hodnota> }}`) pre hodnoty. Hodnoty sa do šablóny posielajú z vyššie spomínaných obslužných funkcií.

Hlavné nevýhody súčasnej implementácie sa prejavia pri pohľade na komplexnejšiu stránku. Napríklad stránku s detailmi IP adresy. Táto stránka využíva spomínanú dekorovanú funkciu, ktorá framework Flask informuje o tom, ktorú URL chceme priradiť k akej šablóne, a ktoré dáta na jej vykreslenie budeme potrebovať. Ďalej využíva samotnú HTML šablónu a okrem nej aj pomocné JavaScript súbory s dodatočnou funkcionalitou na vykresľovanie grafov.

Nevýhody súčasnej Flask implementácie frontendu

Nasledujúce nevýhody sa týkajú hlavne úzkeho prepojenia backendu s frontendom v súčasnej implementácii vo frameworku Flask. Samotný vzhľad webového rozhrania bol analyzovaný v podkapitole 3.1.

1. Nedostatok zabudovanej podpory pre klientský JavaScript: Flask je predovšetkým webový framework na strane servera a umožňuje integráciu klientského JavaScriptu, chýba mu ale zabezpečenie zabudovanej podpory. To môže viesť k ťažkostiam pri implementácii pokročilej funkcionality JavaScriptu. Pri pohľade na súčasnú implementáciu je kódu JavaScript naozaj veľa, dáva preto zmysel prejsť na framework jazyka JavaScript.

2. Obmedzené možnosti frontendu: Flask sa sústreďuje na backend a neposkytuje toľko možností na frontend ako niektoré iné frameworky. To môže viesť k zbytočným komplikáciám pri implementácii komplexného používateľského rozhrania alebo dynamických funkcií frontendu.

3. Integrácia s knižnicami tretích strán: Flask je veľmi prispôsobiteľný, to však znamená, že si vyžaduje viac manuálnej konfigurácie pri integrácii knižníc tretích strán. To vedie k navyšovaniu času potrebného na integráciu JavaScriptových knižníc a plug-inov.

4. Náročnejší na osvojenie si pre vývojárov: Flask je ľahký a flexibilný framework, ale to znamená, že na jeho úspešné použitie sa vyžaduje viac znalostí a skúseností. Navyše nedovoľuje oddelené zásahy čisto do frontendu. Aj keby poverený vývojár s frontendami už pracoval, je pravdepodobné, že to bolo s nejakým frameworkom JavaScriptu. Oddelenie frontendu od backendu a použitie REST API na komunikáciu medzi nimi zaručí možný budúci vývoj po častiach.

5. Problémy s výkonom: Flask nie je tak výkonný ako niektoré iné frameworky pri manipulácii s veľkým množstvom klientského JavaScriptu. Môže to viesť k pomalším odpovediam a menej efektívnemu využívaniu zdrojov servera.

3.2.2 Existujúci backend systému NERD vo Flasku

Hlavná časť pre nás dôležitého zdrojového kódu backendu webového rozhrania systému NERD je v rámci celkovej implementácie systému umiestnená do priečinku `NERDweb`. Existujúci backend aj API majú spoločnú implementáciu (framework Flask v Pythone) v jednom súbore `nerd_main.py`. Pri rýchlom pohľade na tento zdrojový súbor a prípadnom pokuse o jeho analýzu zistíme, že v sebe zahŕňa takmer celú implementáciu. Súbor obsahuje vyše dva a pol tisíca riadkov kódu, v ktorých sa kombinujú obslužné funkcie pre webové rozhranie s databázovými dopytmi aj s definíciou endpointov API v1.

Jedným z cieľov implementácie nového webového rozhrania bude v tomto súčasne nastavenom trende nepokračovať. Budeme naopak chcieť vytvoriť menšie, zmysluplné súborové celky, z ktorých každá sa bude viazať na určitú časť systému a budú riešiť konkrétny problém. Systému taktiež treba pridať väčšiu modularitu.

Hlavný priečinok webovej implementácie `NERDweb` ďalej obsahuje podpriečinky `static` a `templates`, kde priečinok `static` obsahuje statické webové prvky (hlavne obrázky a znovupoužiteľné funkcie jazyka JavaScript). Priečinok `templates` zase obsahuje jednotlivé HTML podstránky.

Za zmienku stojí aj súbor `userdb.py`, ktorý v sebe uchováva funkcionalitu prístupu do lokálnej PostgreSQL databázy používateľov. Práve takéto modulárne oddelenie určitej funkcionality sa budeme snažiť dosiahnuť a vylepšiť aj v novom návrhu systému NERD.

3.2.3 Existujúce aplikačné programové rozhranie

Existujúce aplikačné programové rozhranie (API v1) využíva architekrúrny štýl REST (Representational state transfer). Slúži používateľom na základné dopyty, najmä s dôrazom na poskytovanie zoznamov IP adries a ich reputačného skóre. Táto verzia REST API bola prístupná len používateľom, ktorí mali prístup do NERD webového rozhrania (teda mali vytvorený a aktivovaný účet). Vo webovom rozhraní v sekcii svojho profilu mohli nájsť pridelený unikátny autorizačný token na prístup k REST API v1.

Autorizácia na prístup k API v1 pomocou Authorization Bearer

Prístup k API v1 bol obmedzený a na strane servera sa pred odpoveďou kontrolovali hlavičky, ktoré poslal používateľ. Povolené tvary boli nasledujúce:

- `Authorization Bearer <token>` alebo
- `Authorization <token>`.

Authorization Bearer je štandardná HTTP hlavička.

Ponúkané endpointy

Detailné pochopenie funkcionality existujúcich endpointov je dôležité z hľadiska porozumenia celkovým výstupom systému NERD. To, čo sme schopní vrátiť v podobe štruktúrovaných údajov JSON tvorí akýsi obraz a komplexnú extrakciu z celého systému. Na existujúcich výstupoch endpointov bol založený aj návrh nového rozhrania frontendu a API v2. V nasledujúcej časti sú vymenované všetky pre bežného používateľa zmysluplné výstupy systému NERD.

API v1 ponúkala šesť základných endpointov, rozdelených do troch logických kategórií. V nasledujúcej časti sú popísané základné ukážkové endpointy.

Dopytovanie na jednotlivé adresy

Základné informácie o IP adrese:

`<base_url>/ip/<ip_address>`

Parametre:

- `ip_address` – IPv4 adresa v desatinnom formáte s bodkou alebo IPV6 adresa v plnom alebo skrátenom formáte

Výstupné polia:

- `ip`, `rep`, `fmp`, `hostname`, `asn`, `bgpref`, `ipblock`, `geo`, `tags`

Ukážka:

```
$ curl -H "Authorization: TOKEN"
https://nerd.cesnet.cz/nerd/api/v1/ip/198.51.100.1
{
    "ip": "198.51.100.1",
    "rep": 0.65,
    "fmp": {
       "general": 0.692
    },
    "hostname": "test.example.com",
    "asn": [
        206776,
        58222
    ],
    "bgppref": "198.51.0.0/16",
    "ipblock": "198.51.100.0 - 198.51.100.255"
    "geo": {
        "ctry": "CZ"
    },
    "bl": [
        "blacklist_name",
    ],
    "tags": [
        {"n": "tag_name", "c": 1.0},
    ]
}
```

Podrobné informácie o IP adrese:

`<base_url>/ip/<ip_address>/full`

Parametre:

- `ip_address` – IPv4 adresa v desatinnom formáte s bodkou alebo IPV6 adresa v plnom alebo skrátenom formáte

Výstupné polia:

- `ip` – (string) dopytovaná IP adresa,
- `rep` – (float) reputačné skóre,
- `fmp` – (object) FMP skóre,
- `hostname` – (string) hosťovské meno preložené DNS dopytom,
- `asn` – (array of objects) Autonomous system info,
- `bgpref` – (object) BGP prefix do ktorého IP adresa patrí,
- `ipblock` – (object) IP blok, dko ktorého IP adresa patrí,
- `geo` – (object) geolokačné informácie,
- `ts_added` – (string, `YYYY-mm-ddTMM:HH:SS, UTC`) čas vytvorenia záznamu, obvykle prvého hlásenia,
- `ts_last_update` – (string, `YYYY-mm-ddTMM:HH:SS, UTC`) čas poslednej aktualizácie záznamu,
- `ts_last_event` – (string, `YYYY-mm-ddTMM:HH:SS, UTC`) čas poslednej udalosti tejto IP,
- `bl` – (array of objects) zoznam čiernych listín, na ktorých sa IP nachádza,
- `events` – (array of objects) počet ohlásených udalostí,
- `events_meta` – meta informácie o udalostiach

Ukážka:

```
$ curl -H "Authorization: TOKEN"
https://nerd.cesnet.cz/nerd/api/v1/ip/198.51.100.1/full
{
    "ip": "198.51.100.1",
    "rep": 0.65,
    "fmp": {
       "general": 0.692
    },
    ...
}
```

Z vyššie uvedenej prehľadovej špecifikácie endpointov vyplýva, že návratovou hodnotou sú dáta vo formáte JSON, čo vyhovuje aj potrebám budúceho moderného webového rozhrania. Všetky dopyty API v1 sú však typu GET, čo konkrétne pri endpointe na vyhľadávanie podmnožiny IP adries `<base_url>/search/ip/?<query_parameters>` predstavuje potrebu vypisovania všetkých parametrov vyhľadávania, ktorých je až pätnásť. Pri návrhu nových endpointov pre API v2 v takomto prípade využijeme POST, ktorý nám povolí využiť JSON aj v tele samotného dopytu.

3.3 Návrh nového webového rozhrania systému NERD

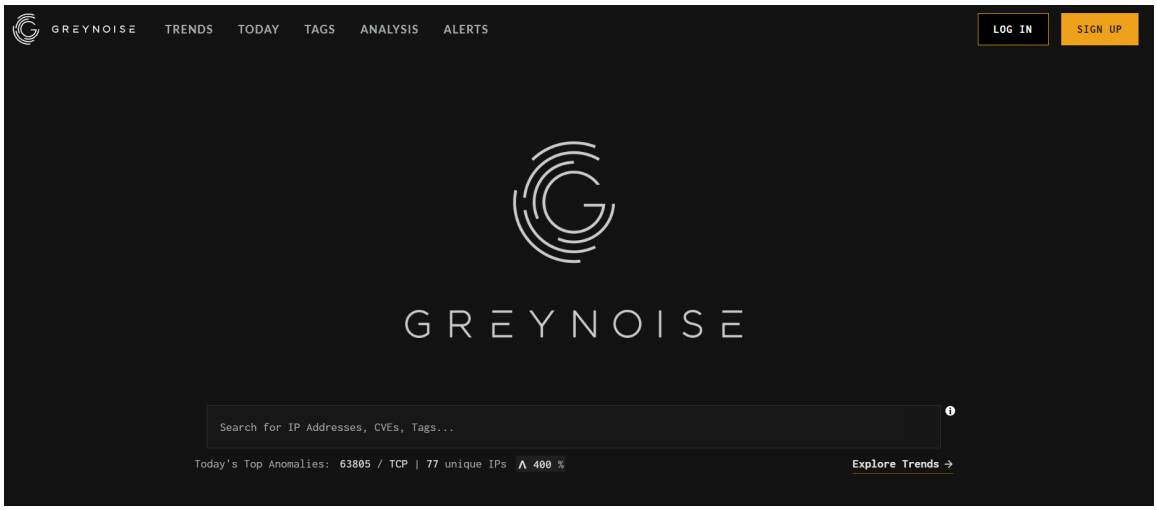
Pri návrhu nového rozhrania budeme zohľadňovať výstup z predošlej analýzy existujúceho rozhrania aj poznatky o tvorbe moderných webových rozhraní. Snahou bude vylepšiť niektoré časti systému, dodať celému systému moderný nádych, novú farebnú paletu a znovu-použiteľné prvky. Podstatný bude aj návrh rozloženia jednotlivých komponentov spolu so zakomponovaním optimalizácie pre mobilné zariadenia.

3.3.1 Analýza webových rozhraní poskytovateľov zdrojov

Ako už bolo spomenuté, NERD získava svoje dáta aj z externých zdrojov. Tieto zdroje majú vlastné webové rozhrania, na ktorých tiež zobrazujú IP adresy a k nim vlastné dáta. Štúdiom a analýzou existujúcich riešení môžeme prispieť k lepšiemu finálnemu návrhu nového webového rozhrania systému NERD.

Pri analýze existujúcich riešení boli skúmané stránky súčasných poskytovateľov zdrojov (Shodan, Talos a Greynoise), nakoľko ich stránky boli z hľadiska používateľských rozhraní najmodernejšie. Väčšina ostatných poskytovateľov, ktorých stánky boli takisto zvážené na analýzu, mali používateľské rozhrania podobné aktuálnemu webovému rozhraniu systému NERD, teda značne zastaralé.

Na obrázku 3.13 je ukážka aktuálneho webového rozhrania stránky Greynoise.



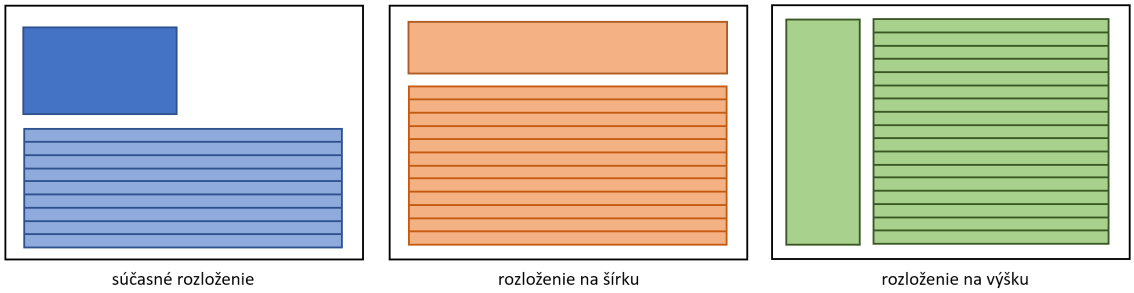
Obr. 3.13: Ukážkový príklad používateľského rozhrania hlavnej stránky Greynoise

3.3.2 Rozloženie hlavnej stránky

Najdôležitejšie komponenty hlavnej stránky sú vyhľadávací formulár a tabuľka s výsledkami. Súčasné rozloženie ich umiestňuje pod seba. Vyhľadávací formulár však nevyužíva celú šírku stránky. V rámci maximalizácie využívaného miesta teda prichádzajú do úvahy dve nové možné vylepšenia. Obrázok 3.14 poskytuje pohľad na možné rozloženia.

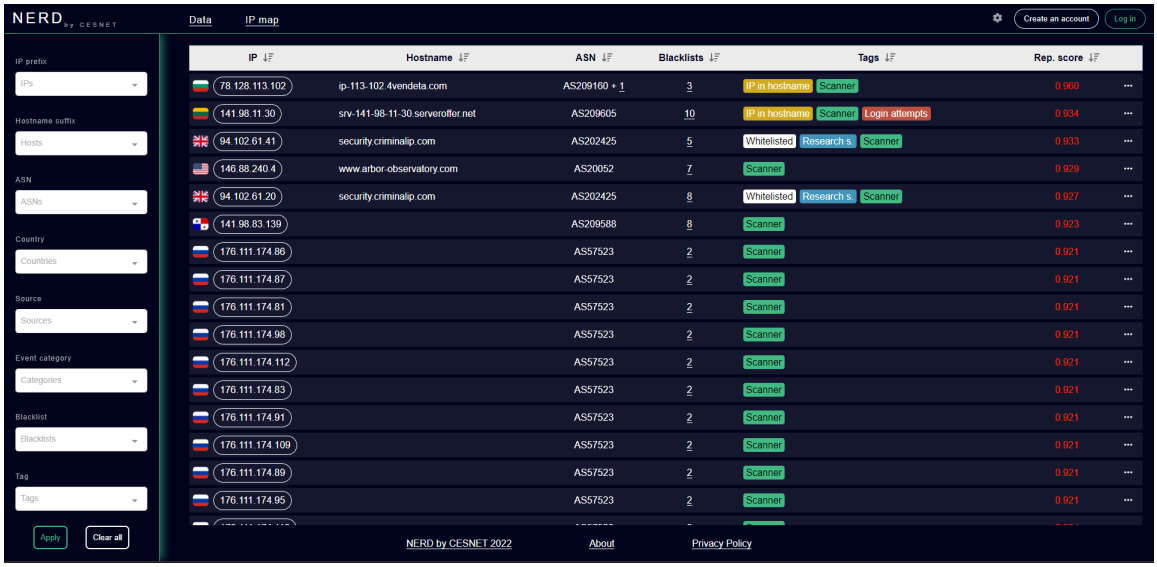
Prvým z nich je natiahnutie vyhľadávacieho formulára na šírku tak, aby ju zaberal celú. Týmto získame zmenšenie plochy, ktorú formulár zaberal na výšku, a vzniká možnosť zobrazenia väčšieho množstva informácií, teda viacerých riadkov tabuľky s výsledkami.

Druhým riešením je potiahnuť vyhľadávací formulár až na spodok stránky a umiestniť tabuľku s výsledkami vedľa neho, čím dosahujeme rozloženie na výšku. Na úkor straty miesta v šírke tabuľky s výsledkami získavame rozloženie, ktoré najlepšie vyhovuje našim potrebám. Vo vyhľadávacom formulári bude možné usporiadať jednotlivé používateľské vstupy pod seba do jedného stĺpca. V tabuľke s výsledkami navyše dostaneme najvyšší možný počet výsledkov pod sebou.



Obr. 3.14: Možné rozloženia vyhľadávacieho formulára a tabuľky výsledkov

Po výbere vhodného rozloženia hlavnej stránky je nutné zamyslieť sa nad celkovou dizajnovou filozofiou, ktorou by sa nový návrh webového rozhrania systému NERD mal uberať. Po rešerši existujúcich riešení zaoberajúcich sa nebezpečnými internetovými entitami špecifikovanej v predchádzajúcej podkapitole (Shodan, Talos, GreyNoise, Censys, Valli), môžeme konštatovať, že väčšina stránok, ktoré prešli modernizáciou dizajnu používateľského rozhrania zvolila tmavú tému. Tmavé stránky s kontrastnými farbami tvoria aj základný dizajnový koncept novo-navrhnutého vzhľadu webového rozhrania pre systém NERD. Ako hlavná kontrastná farba bola zvolená nefritovo zelená, ktorá v kombinácii s tmavšími farbami pozadia, môže simulovať dojem ako v počítačovom termináli. Stránka vďaka zvolenej farebnej kombinácii taktiež nadobúda pocit elegancie, jednoduchosti a profesionality.

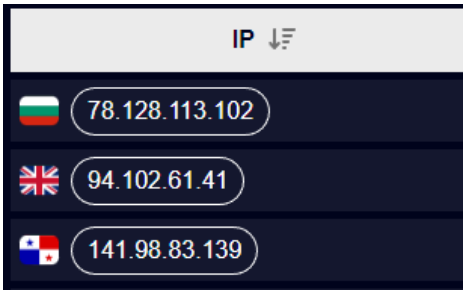


Obr. 3.15: Nový návrh hlavnej stránky

V návrhu, na obrázku 3.15 si môžeme všimnúť aplikovanie rozloženia na výšku, ktoré bolo bližšie predstavené a zdôvodnené vyššie. Ďalej tu vidíme modernú tabuľku, ktorá maximalizuje horizontálne dostupné miesto. Moderné dizajnové návrhy tabuliek často neobsahujú tradičné ohraničenia. Vizuálne sa od seba oddeľujú len jednotlivé riadky, na oddelenie stĺpcov postačuje vynechanie miesta. Tento princíp bol uplatnený aj v prípade tabuľky na hlavnej stránke systému NERD, ktorá tu predstavuje najvýraznejší komponent.

Stĺpce v tabuľke s výsledkami

Pri bližšom pohľade na zobrazenie v stĺpci na obrázku 3.16 môžeme vidieť značku filtrovania, ktorá usporadúva výsledky od najmenšieho po najväčšie alebo podľa abecedy. Celá hlavička je oddelená kontrastnou farbou a aj pri potrebe vertikálneho posunu po stránke scrollovaním zostáva hlavička navrchu, aby bola zjavná príslušnosť jednotlivých kategórií do stĺpcov.



Obr. 3.16: Ukážka stĺpca s IP adresami

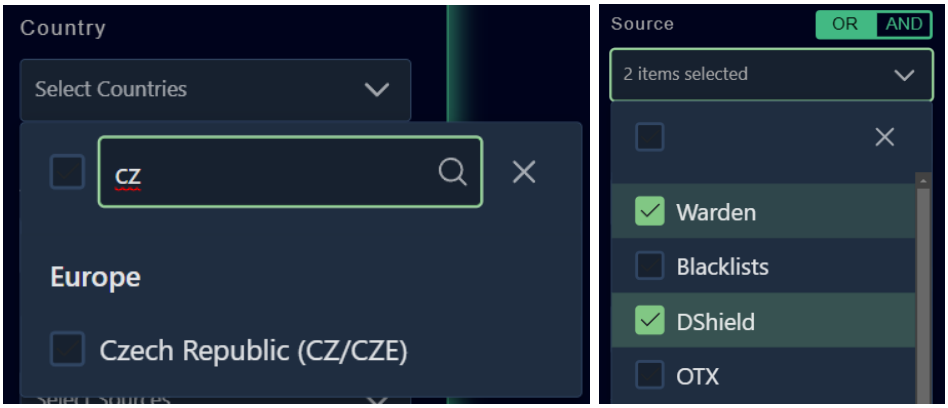
Stĺpec v pôvodnom návrhu, ktorý reprezentoval krajinu vlajkou bol vynechaný. Informácie o krajine (vlajke) sú po novom pred IP adresou, čím šetríme horizontálne miesto. IP adresy sú v tomto stĺpci zarovnané doľava, čo prispieva k lepšej čitateľnosti. Už na prvý pohľad môže používateľ získať predstavu o dĺžke danej IP adresy.

Vyhľadávací formulár

Nové rozloženie stránky viedlo k využitiu užšieho vyhľadávacieho formulára, ktorý má však k dispozícii celú výšku stránky. Používateľské vstupy sú navonok dizajnovo zjednotené, teda všetky vyzerajú rovnako. Bližšie informácie o tom, čo má používateľ do každého poľa zadať získa prejdením kurzora ponad názov každého poľa. Vstupy tiež disponujú dynamickou syntaktickou kontrolou. Zadané informácie v nevhodnom tvare sú ignorované. Bližší pohľad na vstupné pole pre prefix IP adresy je na obrázku 3.17.

Obr. 3.17: Ukážka vyhľadávacieho poľa pre IP adresy a nápovedy pred zadávaním

Všetky vstupné polia podporujú zadanie viacerých možností. Pri zadávaní IP prefix adresy, hostname a ASN by ponúknutie všetkých možností zbytočne komplikovalo načítavanie stránky a prehľadnosť výberu. Pri týchto parametroch vyhľadávania je z tohto dôvodu umiestnené tlačidlo multi, ktoré používateľom umožňuje do textového poľa naraz vpísať viacero hodnôt. Pri všetkých ostatných kategóriách sú možnosti ponúkané pevne a je podporované aj vyhľadávanie medzi nimi.



Obr. 3.18: Ukážka vyhľadávania krajiny spomedzi možností (vľavo) a ukážka výberu viacerých zdrojov naraz (vpravo)

V pôvodom návrhu systému NERD bolo dobre prepracované zobrazenie dodatočných informácií po prechode myšou alebo po kliknutí. Takýmito informáciami disponujú všetky dôležité komponenty. Funkcionalita je zachovaná aj v novom návrhu.

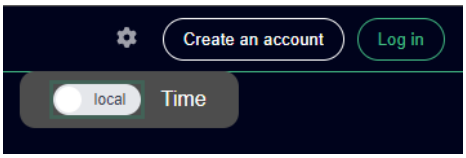
3.3.3 Nová vrchná lišta

Vrchná lišta, na obrázku 3.19, sa skladá z troch základných častí:

- logo – identita stránky a odkaz na domovskú stránku,
- navigácia – odkazy na podstránky,
- nastavenia a možnosti prihlásenia.

Obr. 3.19: Vrchná lišta pre neprihláseného používateľa

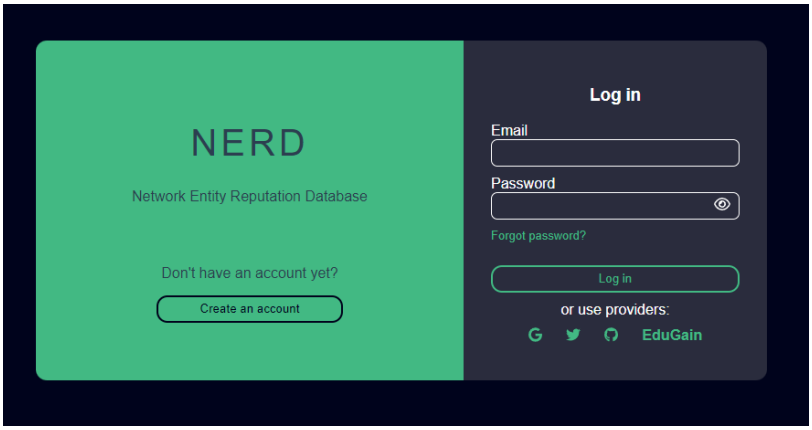
Nastavenia v aktuálnej podobe, ako vyobrazené na obrázku 3.20, predstavujú prepínanie časových údajov na stránke medzi lokálnym a UTC časom. V budúcnosti je tu priestor pre viac nastavení, prípadne dodatočné nastavenia pre prihlásených používateľov.



Obr. 3.20: Nastavenia časových dát

3.3.4 Stránky pre prihlasovanie a vytváranie profilu

Pôvodný systém nemal žiadne stránky pre prihlasovanie a vytváranie profilu (prihlasovanie prebiehalo dialógovým oknom prehliadača). Nový návrh ponúka jednoduché prihlásenie s možnosťou využitia externých poskytovateľov prihlasovania.

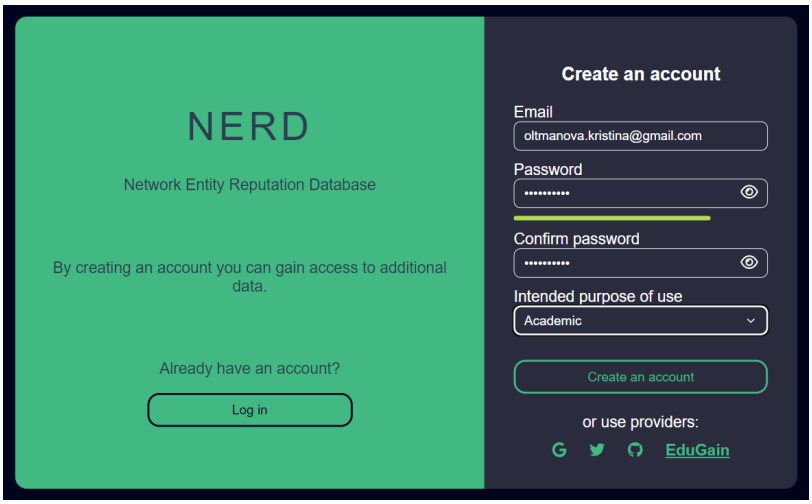


Obr. 3.21: Formulár na prihlasovanie existujúceho používateľa

Ako vidno na obrázku 3.21, je možné zobraziť napísané heslo a je tu dostatok miesta na zobrazenie prípadných chybových hlásení pri nesprávnom prihlasovaní. Vo formulári nesmie chýbať ani možnosť obnovy zabudnutého hesla. Ak používateľ ešte nemá vytvorený účet, je tu aj presmerovanie na tvorbu profilov pre nových používateľov.

Registračný formulár, na obrázku 3.22, slúži novým používateľom. Na ľavej strane je preto priestor na krátke predstavenie systému NERD a informácie o tom, čo používatelia

vďaka vytvoreniu účtu získajú. Naľavo sú polia potrebné na vytvorenie účtu: email, heslo, potvrdenie hesla a špecifikácia oblasti, pre ktorú sa používateľ chystá dáta zo systému NERD využiť.



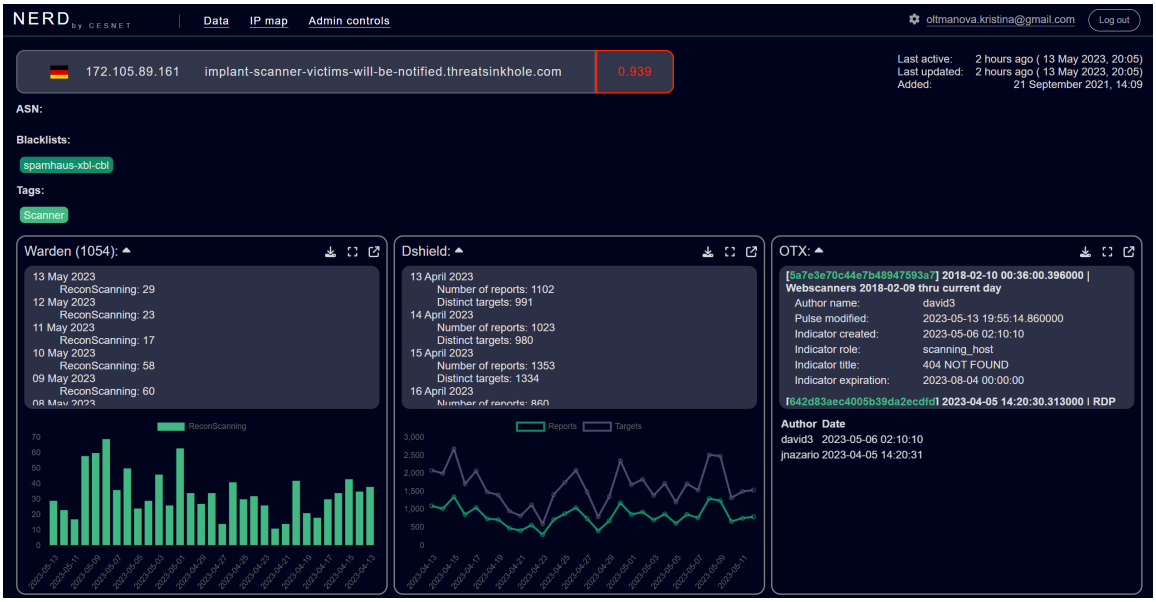
Obr. 3.22: Formulár na vytvorenie profilu pre nového používateľa

3.3.5 Stránka s detailami IP adresy

Hlavnou nevýhodou pôvodnej stránky s detailami bolo množstvo vertikálneho miesta, ktoré jednotlivé informácie zaberali. Používateľovi nebol poskytnutý rýchly a komplexný náhľad, v ktorom by nemusel stránku často vertikálne posúvať. Umiestenie tabuliek s bližšími informáciami získanými zo zdrojov Warden, DShield a OTX vedľa seba šetrí miesto. Používatelia vidia naraz dáta zo všetkých troch zdrojov. Ak by sa chceli na niektorý zo zdrojov zamerať bližšie, vedia si náhľadovú tabuľku zväčšiť, prípadne otvoriť v novom okne.

Pod tabuľkami zdrojov sú umiestnené aj grafy. Grafy využívajú nadstavbu Chart.js nad Vue.js, ktorá je z hľadiska použiteľnosti oproti prechádzajúcej implementácii podstatne jednoduchšia. Každý graf má vlastný komponent, ktorý využije dáta vrátené z nového aplikačného programového rozhrania (API) a je ich schopný zobraziť s minimom ďalšieho spracovania.

Aktuálny návrh môžeme vidieť na obrázku 3.23.

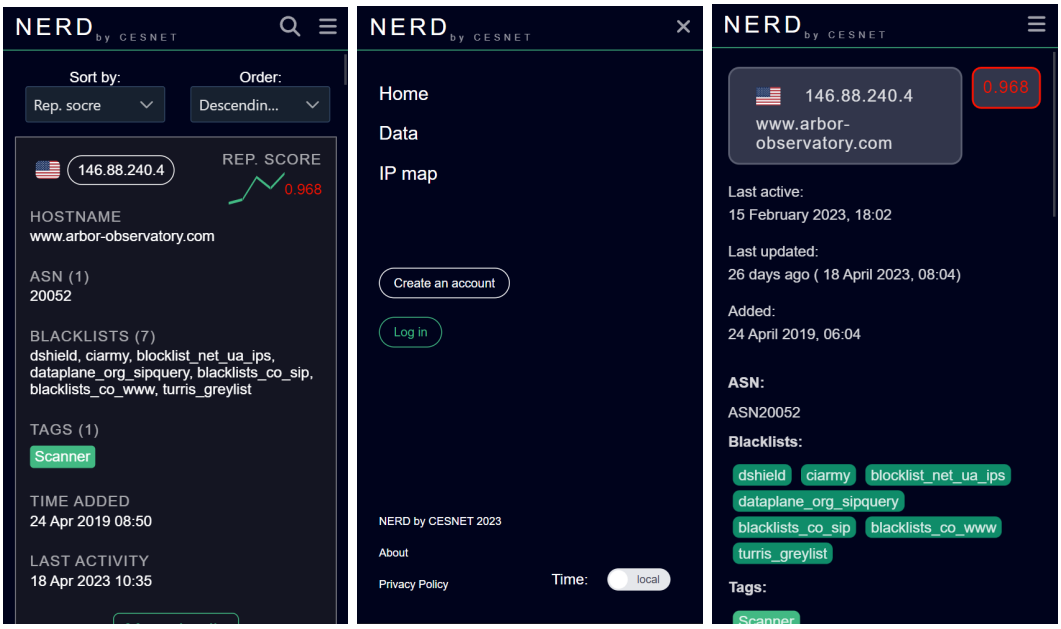


Obr. 3.23: Stránka s detailami pre IP adresu

3.3.6 Zobrazovanie na mobilných zariadeniach

Dôležitou súčasťou moderného webového rozhrania je responzivita návrhu. Cieľom nového rozloženia pre mobilné zariadenia bolo poskytnú používateľom pohodlnú možnosť zobrazovania a využívania webového rozhrania systému NERD na všetkých zariadeniach, aj tých s užšou obrazovkou.

Na obrázkoch 3.24 vidno hlavú obrazovku, mobilné menu dostupné po kliknutí na ikonu v hornej lište, a takisto detail konkrétnej IP adresy. V hornej lište je okrem ikony na otváranie mobilného menu aj ikona na otvorenie vyhľadávacieho formulára.



Obr. 3.24: Ukážka zobrazenia na mobilných zariadeniach: hlavná stránka (vľavo), mobilné menu (v strede) a detail IP adresy (vpravo)

3.4 Návrh nového systému

Ako už bolo naznačené v analýze existujúceho riešenia frontend treba oddeliť od backendu a pridať obslužnú funkcionalitu novej API v2 tak, aby bola schopná zastrešovať všetku komunikáciu medzi týmito novo oddelenými súčasťami.

3.4.1 Nový frontend vo frameworku Vue.js

Návrh nového frontendu pozostáva zo samotného dizajnu, ktorý už bol predstavený v predchádzajúcej podkapitole 3.3. Okrem toho sa ale musíme nad frontendom zamyslieť aj ako nad novou samostatnou súčasťou systému. Táto súčasť bude očakávať odpovede vo formáte JSON, ktorý je najvhodnejší či už na prenos sieťou alebo na ďalšie spracovanie v jazyku JavaScript.

Nový frontend bude z hľadiska návrhu obsahovať súbor `index.html`, ktorý tvorí základ single-page aplikácií a povoluje nám špecifikovať potrebné hlavičky, či už na optimalizáciu SEO, dodanie ikony stránky, alebo na metadáta a viewport nastavenie pre responzívny dizajn.

Ďalej bude obsahovať samotné zdrojové kódy pre každú podstránku systému. Tieto zdrojové kódy s koncovkou `.vue` v sebe kombinujú postupne kódy HTML, JavaScript a CSS (Cascading Style Sheets). Všetky potrebné zdroje na vytvorenie jednej stránky sú teda v jednom súbore, čo zaručuje vyššiu prehľadnosť.

Jednou z analyzovaných nevýhod pôvodnej implementácie bolo, že na strane frontendu sme nemohli jednoducho importovať knižnice. Bolo nutné ich buď sťahovať lokálne alebo odkazovať na konkrétne verzie pomocou načítavanie externých zdrojov v HTML. Použitie frameworku Vue.js rieši tieto nevýhody. Môžeme používať komplexné vyhľadávače existujúcich moderných komponentov a začať využívať ľubovoľný z nich pomocou troch riadkov

kódu. Následne môžeme s komponentom pracovať ako s novým HTML tagom a špecifikovať mu potrebné parametre.

V rámci výstavby nových stránok môžeme využívať aj vlastné komponenty, ktorých oddelenie od hlavného kódu prispieva k modularite, prehľadnosti a znovu-použiteľnosti.

3.4.2 Aktualizovaný backend vo frameworku Flask

Nový backend bude zachovávať (aspoň dočasne) funkcionality vykresľovania pôvodného frontendu. Je takto vykonané za účelom možnosti súbežného behu starého aj nového systému.

Všetky nové súčasti budú dostupné v oddelených súboroch. Budeme potrebovať dodatočné súbory na zaistenie funkcií:

- ponúknutie nových API v2 endpointov,
- rozšírenie databázovej funkcionality pre pokrytie nových potrieb evidencie používateľov (vrátane externých poskytovateľov identít),
- modul na overovanie autorizácie pri pokusoch o prístup k chráneným endpointom.

3.4.3 Nové aplikačné programové rozhranie

Návrh architektúry pôvodného rozhrania bol zachovaný. Nadstavujeme nad REST API v1, pričom využívame oddelený modul s vlastnými endpointami s novými prefixami. Endpointy prispôsobujeme novému backendu a všetky v ňom definované funkcie ponúkame v podobe HTTP endpointov na prístup z frontendu. V rámci návrhu výraznejšie meníme pôvodný endpoint na vyhľadávanie IP adries, a konkrétne využijeme typ dopytu POST, ktorý nám umožní odosielať komplexný na frontende zostavený JSON v tele dopytu.

API v novej verzii v2 je implementovaná v samostatnom Blueprinte frameworku Flask, aby sme dosiahli vyššiu modularitu systému a logické oddelenie novej funkcionality.

Otvorené endpointy a autorizácia pomocou JSON Web Token

Nový návrh počíta s tým, že niektoré endpointy musia byť z pohľadu zaistenia chodu systému aj pre neprihlásených používateľov otvorené a verejne prístupné. Otvorené budú musieť byť endpointy zabezpečujúce nasledujúce funkcie:

- získavanie výsledkov vyhľadávania záznamov,
- detailné informácie o zázname,
- prihlasovanie používateľov,
- registrácia používateľov,
- obnova zabudnutého hesla,
- návratové funkcie pre externých poskytovateľov identít.

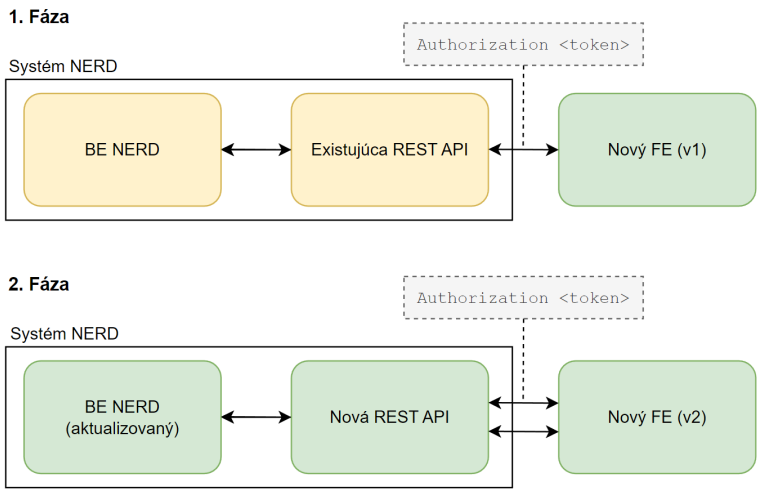
Všetky zvyšné endpointy budú mať prístup obmedzený pomocou JWT (JSON Web Token) autorizačných tokenov.

Kapitola 4

Implementácia nového webového rozhrania a systému NERD

V tejto kapitole je bližšie predstavená implementácia nového webového rozhrania systému NERD. Ako už vyplynulo z analýzy nebude sa jednať len o nahratie nových štýlov a dizajnu stránok. Nový frontend, ktorý bude fungovať ako samostatná súčasť systému musí vedieť komunikovať s existujúcimi časťami. Samotná implementácia sa dá rozdeliť na dve fázy (znázornené aj na obrázku 4.1):

1. fáza – napojenie nového frontedu na doposiaľ ponúkané endpointy,
2. fáza – adaptovanie REST API a backendu, aby odpovedali potrebám nového frontendu.



Obr. 4.1: Dve fázy implementácie

Tieto implementačné fázy sú bližšie popísané v nasledujúcich podkapitolách.

4.1 Prvá implementačná fáza

Prvá implementačná fáza bola vytvorená za účelom rýchleho napojenia nového frontendu na už existujúce časti. Získanie skutočných dát do vyvíjaného webového rozhrania pomohlo vývoju tejto časti systému. Zároveň sme mohli zisťovať, či nám dáta v ponúkanom formáte vyhovujú, alebo či budú potrebné zásahy do backendu, ktoré umožnia iné predspracovanie.

Táto implementačná fáza nám zároveň ponúkla priestor na vyskúšanie prepojení súčastí frontend-backend cez REST API. Nečakanú a nepríjemnú komplikáciu v tomto priebehu vývoja predstavoval Cross-Origin Resource Policy, teda mechanizmus zdieľania dát medzi rôznymi doménami.

4.1.1 Frontend moderného webového rozhrania vo Vue.js

Systém NERD svojím rozsahom a komplexnosťou predstavuje vhodný projekt na implementáciu vo frameworku Vue.js. Vue.js v tomto prípade vyhovuje najmä svojou jednoduchosťou, škálovateľnosťou, množstvom podpory a existujúcich knižníc s komponentami. Angular by bol pri tomto projekte zbytočne komplexný a tento projekt by nevyužil ani hlavnú prednosť frameworku React, ktorou je multiplatformný vývoj. Systém NERD je primárne zameraný na zobrazovanie dát na počítačovej obrazovke v prehliadači. Súčasťou nového webového rozhrania bude aj responzívna optimalizácia pre mobilné zariadenia. Nie je to však priorita, a preto jej nebude podriadený vývoj.

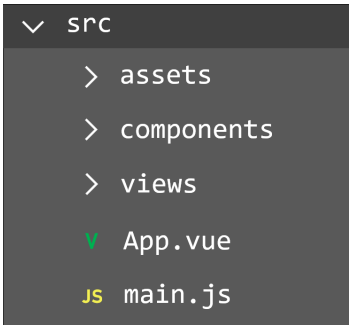
Framework Vue.js je pre implementáciu nového moderného systému NERD zvolený v kombinácii s jazykom JavaScript. TypeScript síce poskytol rozsiahlejšiu podporu pre komplexné dátové objekty, tých však v systéme NERD nie je veľa. Navyše získavanie dát z neštruktúrovanej NoSQL MongoDB databáze prispieva k lepšiemu celkovému výsledku v spojení s jednoduchším jazykom JavaScript.

Zvolená verzia frameworku Vue.js je najnovšia: Vue3. Pre viacero knižníc a balíčkov je teda potrebné nájsť ich najnovšie verzie.

Štruktúra zdrojových súborov

Vue.js navonok disponuje najmä špecifickým skladaním kódu. Súbory typu `.vue` v sebe postupne obsahujú časti `<template>`, `<script>` a `<style>`, teda tradičné samostatné súbory HTML, JavaScript a CSS sú zjednotené v jednom súbore. Pre vyššiu prehľadnosť vedia vývojové prostredia (napríklad VS Code alebo IntelliJ Idea) kód vizuálne oddeliť do viacerých okien. Z pohľadu programátora to predstavuje pomerne pohodlné riešenie, najmä pri dodatočných zásahoch, kde má všetky komponenty sústredené v jednom súbore a nemusí hľadať príslušné časti inde v dokumentovej štruktúre zdrojových súborov.

Načrtnutie základnej štruktúry zdrojových súborov je k dispozícii na obrázku 4.2.



Obr. 4.2: Základná štruktúra súborov zdrojového kódu

Bližšie predstavenie častí zdrojových súborov

App.vue

Obsahuje základnú stavebnú štruktúru všetkých stránok. Práve tu vieme špecifikovať, že každá stránka má mať hlavičku a pätu. V časti `<style>` môžeme pomocou globálnych CSS selektorov (napríklad `body`) špecifikovať štýly pre všetky stránky.

main.js

Tento súbor v jazyku JavaScript má na starosti spúšťanie samotnej webovej aplikácie, obsahuje všetky potrebné globálne importy knižníc. Najdôležitejšie skupiny importov:

- základné Vue knižnice (createApp, App),
- balíčky s ikonami (použité ikony FontAwesome),
- dôležité súčasti pre Vue (Router a Store),
- často používané komponenty, pre ktoré sa oplatí globálny import (nápovedy, štylizované indikátory posunu na stránke a ďalšie).

assets

Dodatočné pomocné skripty a dodatky. Napríklad pre vymenovanie dostupných krajín, pre ktoré chceme vedieť filtrovať výsledky. Takáto informácia nie je uložená v databáze a je lepšie mať ju priamo v systéme než sa na ňu dopytovať z externého zdroja. V tomto podpriečinku sú takisto dostupné obrázky, ktoré v aplikácii využívame. Vďaka tomuto umiestneniu ich Vue.js automaticky rozpozná a správne exportuje pri zostavovaní optimalizovanej verzie pre produkciu.

components

Pomocné komponenty využívané stránkami v priečinku views alebo využívané v iných komponentoch. Základné stavebné prvky (komponenty), ktoré už existujú a sú dostatočne všeobecné alebo upraviteľné do takej miery, že vyhovujú požiadavkám systému sú importované z externých zdrojov. Tento priečinok obsahuje vlastné komponenty, teda časti kódu, ktoré majú potenciál byť znovu-použiteľné. Základné vlastné komponenty sú napríklad: hlavička stránky, päta stránky, vyhľadávací formulár a ďalšie.

views

Views predstavujú tradične samotné stránky, ktoré používateľ môže priamo uvidieť (view). Stránky vo Views špecifikujú najmä štruktúru v akej držia jednotlivé komponenty.

V systéme NERD môžeme identifikovať niekoľko takýchto stránok (Vue implicitne vynucuje viacslovné názvy views aj komponentov). Najdôležitejšie z nich sú:

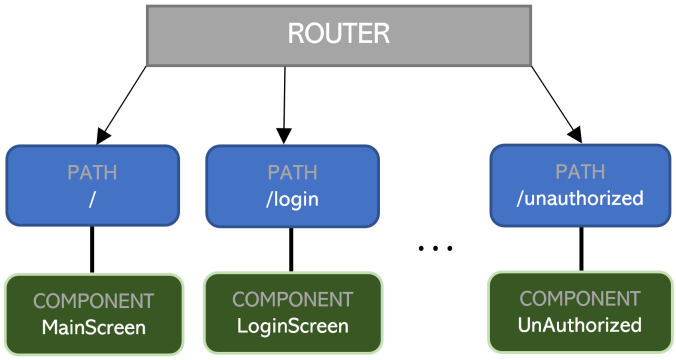
- hlavná stránka (`MainScreen.vue`),
- stránka s detailami o IP adrese (`IpDetailScreen.vue`),
- stránka prihlasovania (`LoginScreen.vue`),
- stránka vytvárania nového profilu (`CreateAccountScreen.vue`),
- stránka s profilom (`ProfileScreen.vue`),
- stránka s detailami o systéme NERD (`AboutScreen.vue`),
- nenájdená stránka – pri chybnej URL (`NotFound.vue`),
- stránka neoprávneného prístupu (`UnAuthorized.vue`).

Každá z vyššie spomínaných stránok sa viaže na nejakú špecifickú URL. O tom, ktorá stránka patrí ktorej URL rozhodujeme v súbore `router.js`.

router.js

Router je komponent, ktorý importujeme v hlavnej obslužnej funkcii. V jeho špecifikácii pri zostrojovaní vytvárame prepojenia path-component. Spájame vďaka nemu teda URL adresy a časti kódu, ktoré sa pri ich načítaní majú zobraziť.

Naznačenie využívanie komponentu router je naznačené na obrázku 4.3.



Obr. 4.3: Routing – spájanie URL adries s príslušnými komponentami

api.js

Tento modul je veľmi dôležitý pre zaistenie komunikácie frontendu s REST API v2. Využíva sa tu knižnica `axios`, ktorá zapuzdruje inak komplikované a zdĺhavé špecifikácie HTTP dopytov.

Sú tu funkcie na automatické prikladanie prístupového tokenu (dostupného z lokálnej pamäte prehliadača) ku všetkým dopytom. Takisto sa tu nachádza sieťové overovanie, že

token ešte nevypršal. Ak vypršal žiadame REST API o nový, v prípade neúspechu odstraňujeme prístupový token z lokálnej pamäti a používateľa odhlasujeme.

Modul ďalej obsahuje jednotlivé asynchrónne funkcie na prístup k endpointom. Sieťový dopyt špecifikujeme priamo v týchto funkciách. Zo stránok alebo komponentov aplikácie môžeme vďaka tomu pristupovať už len k volaniu daných funkcií.

store.js

Store nám umožňuje správu lokálne uložených hodnôt. Lokálne v prehliadači sa nám oplatí ukladať hodnoty z API, ku ktorým by sme inak často pristupovali. Rovnako si do store ukladáme aj parametre filtrov vyhľadávania. K hodnotám zo store máme prístup zo všetkých súborov stránok a komponentov. Môžeme ich priamo namapovať, aby sa do nich ukladali hodnoty z vyhľadávacích polí (tvorených multiselektívnymi výberovými poliami), a tým boli uložené pre ďalší priebeh vyhľadávania alebo interakcie so stránkou.

4.1.2 Práca vo frameworku Vue.js

Na implementáciu nového webového rozhrania systému NERD bola zvolená najnovšia verzia frameworku Vue.js, teda Vue.js 3. Nová verzia ponúka lepší výkon, menšiu veľkosť výslednej aplikácie a lepšiu škálovateľnosť. Počas prác vo Vue.js 3 nastalo len pár komplikácií, ktoré nebolo ťažké vyriešiť. Nakoľko ale ovplyvňujú aj budúci vývoj, pretože sa v priebehu prác a pridávaní súčastí bežne objavujú, je vhodné ich tu predstaviť.

Kompatibilita verzií Vue.js

Nie všetky súčasti majú aktualizovanú kompatibilitu na Vue3. Často sa dá nájsť novšia, prípadne aktualizovaná verzia, ale v prípade, že nie, existuje možnosť po chybnom pokuse o inštaláciu pomocou `npm i <package>` zadať prepínač `legacy-peer-deps`, ktorý vyrieši konflikty vzniknuté čiastočnými rozdielmi medzi Vue verziami 2 a 3, a povolí tak nainštalovanie balíčka pôvodne určeného pre staršiu verziu.

4.1.3 Získavanie dát z existujúceho aplikačného programového rozhrania

Pre implementáciu nad skutočnými zdrojmi a najmä finálnymi tvarmi dát bola spočiatku implementácia frontendu napojená, a teda získavala dáta priamo z existujúcej implementácie REST API. Prispôsobenie nového frontendu existujúcim endpointom viedlo k zaručeniu integrity systému. Jednou z požiadaviek pri návrhu nového systému bolo aj zachovanie pôvodných endpointov a rozšírenie funkcionality o API v2.

Existujúca API v1 dodávala dáta z MongoDB databáze nebezpečných entít, vyžadovala však autorizáciu používateľským tokenom. Tento krok bol počas prvotných fází vývoja preskočený. Dôležité bolo overiť, či existujúce API generované dáta budú vyhovovať novej štruktúre a implementácii frontendu vo Vue.js. Pôvodne dáta neopúšťali Python, avšak API ponúkala webu vyhovujúce JSON odpovede, ktoré sú vhodné na ďalšie webové spracovanie.

Pre získavanie dát z REST API je využitá knižnica `axios`, ktorá zapuzdruje a zjednodušuje dotazovanie. Pri nastavovaní komunikácie s pôvodnou existujúcou API vieme špecifikovať, aby sa k všetkým odosielaným dopytom pridávala hlavička `Authorization` s tokenom, na dočasné obídenie a testovanie s API v1.

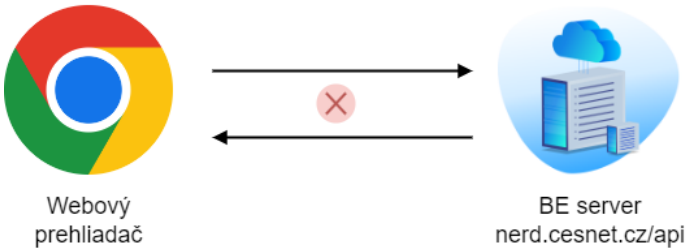
Získané dáta sú typu JSON a vieme ich využiť na dalšie spracovanie priamo v komponentoch Vue.js.

Prístup k REST API z webového prehliadača

Pri prvotných pokusoch o základné napojenie implementácie frontendu na dáta z API v1 vznikla komplikácia s CORS (Cross-origin resource sharing) policy. Zobrazovali sa chyby spojené s chýbajúcim `Access-Control -Allow-Origin`, ktoré obvykle napovedajú, že backend spravujúci API dopyty nemá povolenie odosielať na ľubovoľné zdroje – origins.

Riešenie povolenia `Access-Control-Allow-Origin` sa vykonáva pridávaním hlavičiek – `Access-Control-Allow-Origin: *` do všetkých requestov zo strany backend servera. Vo Flasku je možné využiť knižnicu `flask-cors`, ktorá automaticky dopĺňa požadovaná hlavička a zabezpečuje aj ďalšiu výmenu správ s webovými klientmi.

Problémová komunikácia je naznačená na obrázku 4.4.



Obr. 4.4: Naznačenie problémovej komunikácie CORS medzi webovým prehliadačom a BE (backend) serverom

Tento vyššie spomínaný postup na obídenie problémov však nepomohol. Po skontrolovaní správnosti odosielania z backend servera bolo potrebné odhaliť potenciálny problém na strane frontendu.

Pre správnu funkcionalitu a vyhnutie sa ťažko odstrániteľným problémom s CORS využijeme konfiguráciu Vue s proxy serverom, kde povoľujeme zmenu `changeOrigin` a nastavujeme existujúcu URL REST API ako cieľ presmerovania dopytov na `/api`.

Proxy server funguje ako medzivrstva medzi implementáciou frontendu a Flask API.

4.2 Druhá implementačná fáza

V predošlej implementačnej fáze sme si vytvorili základ frontendu. Získali sme tiež predstavu o tom, aké dáta nám systém ponúka a hlavne sme identifikovali, čo nám chýba, a čo je potrebné do systému pridať.

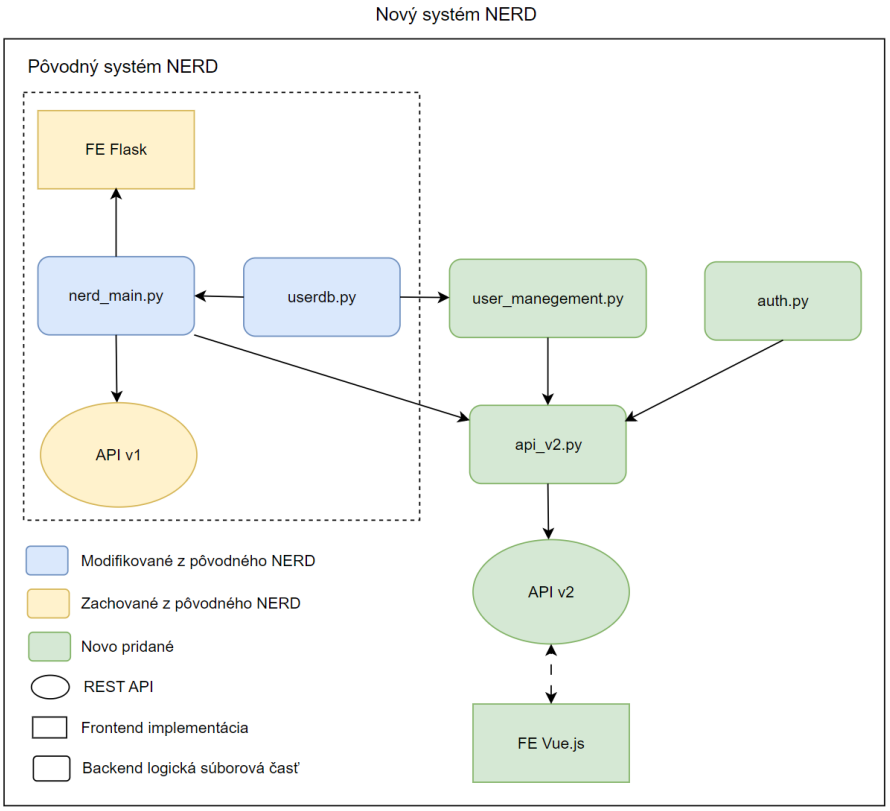
V rámci ďalšej udržateľnosti bola implementácia novej nadstavby backendu oddelená do nových súborov. Využíva a napája sa na určité časti starého systému, ale celá nová funkcionalita je obsiahnutá v nových zdrojových súboroch.

Nový návrh architektúry systému NERD môžeme vidieť na obrázku 4.5. Je na ňom zakreslené aj využitie častí pôvodného systému, na ktoré sa napájajú nové komponenty.

Zatiaľ čo v prvej implementačnej fáze sme nový frontend vo Vue.js dočasne napojili na API v1, v druhej implementačnej fáze frontend komunikuje plne s API v2. Umožňuje nám to z frontendu pristupovať ku všetkým zdrojom backendu pomocou REST API HTTPS volaní.

Na obrázku vidno aj farebne odlíšené implementačné časti. Pôvodný frontend a API v1 zostávajú v plnom rozsahu v pôvodnej podobe. Do implementácií zdrojových súborov `nerd_main.py` a `userdb.py` bolo potrebné zasiahnuť z hľadiska ich rozšírenia. Počas imple-

mentácie boli vytvorené aj nové zdrojové súbory, ktoré sú bližšie popísané v nasledujúcej podkapitole.



Obr. 4.5: Nový návrh systému s podrobnou štruktúrou zmien backendu

4.2.1 Backend moderného webového rozhrania vo frameworku Flask

V novom návrhu backendu sa nezaoberáme zostavovaním a predprípravou jednotlivých častí na zobrazenie frontendu. Nový backend využívame len na získanie funkcií, ktoré nám umožnia vytvoriť obslužné endpointy pre API v2, na ktoré bude následne možné pripojiť implementáciu frontendu. Je dôležité pripomenúť, že nový backend bude obsahovať modulárne časti koré sa len pripoja a doplnia existujúcu funkcionalitu. Pokračovalo sa preto vo využívaní frameworku Flask a implementácia prebiehala v jazyku Python.

V nasledujúcich podkapitolách sú podrobnejšie predstavené novo vytvorené alebo modifikované súbory a ich funkcie.

`user_management.py`

V tomto zdrojov súbore obohacujeme jednoduché databázové dopyty z `userdb.py` a tvoríme z nich komplexnejšie funkcie napríklad na generovanie tokenov.

`user_management.py`

V tomto súbore definujeme znovu-použiteľný dekorátor funkcií, ktoré budú mať obmedzený prístup len na prihlásených používateľov. Pracujeme tu s JWT, overujeme ich platnosť,

prípadne využívame obnovovacie `refresh` tokeny na predĺženie sedení používateľov. Z každého JWT vieme získať kontext aktuálneho používateľa endpointu, ktorý vieme využiť pri generovaní odpovede.

`api_v2.py`

Tento súbor obsahuje popisy všetkých nových endpointov. Využíva deokrátory `@token_required` na autentizáciu prístupu používateľov a `@swag_from` na vytvorenie dokumentácie a automatickú validáciu s využitím popisu endpointov pomocou OpenAPI. Podrobnejší popis generovania dokumentácie a použitia OpenAPI špecifikácie na automatickú validáciu je možné nájsť v podkapitole 4.2.2.

4.2.2 Nové aplikačné programové rozhranie vo frameworku Flask

Nová REST API v2 pretavuje funkcionalitu nového backendu do prístupných HTTP endpointov. Ciele vytvorenia nového REST API v2 sú objasnené v nasledujúcich podkapitolách.

Prechod k decentralizovanému prístupu k operáciám nad databázou

Doposiaľ centralizovaná správa frontendu v priamej kombinácii s backendom a databázou, kde jednotlivé časti spolu mohli interne komunikovať musí byť v prípade využitia nového samostatne bežiaceho frontendového webového rozhrania rozdelená. Spomínané súčasti spolu doposiaľ komunikovali v rámci použitého Flask frameworku, ktorý poskytoval aj webové rozhranie. Funkcionalita, ktorú existujúce riešenie ponúkalo v podobe vstavaných lokálnych funkcií v kóde, musí byť otvorená a komplexne zastrešená s využitím nového REST API v2. Doposiaľ existujúce endpointy API v1 spomínané v kapitole 3.2.3 sú aj v novej verzii systému NERD zachované. Bolo tak vykonané na základe informácií, ktoré napovedajú, že značná časť aktuálne registrovaných používateľov mala vytvorený účet práve za účelom získania tokenu na prístup k endpointom API v1. Je preto dôležité zachovať funkcionalitu, ktorá je v súčasnosti najviac využívaná.

Požiadavka nadstavby nad existujúcou funkcionalitou

Snaha o zachovanie existujúcej funkcionality API v1 pre jej súčasných používateľov ovplyvňuje okrem samotnej implementácie API takisto používanie databáz. NERD využíva primárne dve databázy. Prvá databáza bola z pohľadu zachovania spätnej kompatibility a dobrého návrhu zachovaná aj ako databáza pre nový systém NERD. V ďalších častiach tejto práce bude spomínaná analyzovaná už len databáza používateľov, nad ktorou bola postavená aj nová správa používateľov bližšie špecifikovaná v podkapitole 4.3.1.

Z hľadiska využitia existujúcich dopytov a funkcií vo Flasku, sa API v2 snaží robiť veci zaužívaným spôsobom, avšak pridáva si potrebné funkcie prípadne skratky všade tam, kde by existujúca implementácia viedla k zbytočným krokom.

Implementácia endpointov API v2 bola vložená do existujúcej štruktúry súborov, avšak je dostupná vo vlastnom súbore, ktorý je doplnený do `nerd_main.py` súboru ako Flask Blueprint. Implementácia spočíva v súbore `api_v2.py`, ktorý najskôr implementuje potrebné súčasti z troch základných oblastí:

- `nerd_main.py` – importujeme z neho znovu použiteľné funkcie, obzvlášť tie, ktoré zobrazujú výsledky internetových entít a ich detaily,

- `user_management.py` – modul poskytujúci obslužné funkcie k správe používateľov,
- `userdb.py` – zapuzdruje všetky funkcie priamo vykonávajúce dopyty nad databázou.

Endpointy API v2 na dopyty do NERD

- /ip/<ipaddr> – základné informácie o IP adrese,
- /search/ip/ – vyhľadávanie IP adries,
- /details/<ipaddr> – detaily o IP adrese.

Endpointy API v2 na správu používateľov

- `/login` – prihlásenie, vráti JWT prihlasovací token,
- `/register` – vytvorenie nového profilu,
- `/verify` – verifikácia emailovej adresy,
- `/me` – základné informácie o aktuálne prihlásenom profile,
- `/users` – prehľad používateľov pre admina,
- `/reset_password` – obnova hesla,
- `/change_password` – zmena hesla.

Endpointy API v2 na správu používateľov pre admina

- `/users` – výpis všetkých používateľov z databázy,
- `/roles` – zmena role používateľa,
- `/delete-user` – odstránenie používateľa,

Endpointy API v2 pre externých poskytovateľov identít

- `/oauth/google` – redirect URI pre Google,
- `/oauth/twitter/url` – generovanie URL na prihlásenie cez Twitter,
- `/oauth/twitter` – redirect URI pre Twitter,
- `/oauth/github` – redirect URI pre GitHub,
- `/oauth/edugain/url` – generovanie URL na prihlásenie cez EduGain,
- `/oauth/eduid` – redirect URI pre EduGain,

Ďalej boli pridané endpointy na zaistenie funkcionality externých poskytovateľov identít. Jednalo sa o endpointy, ktoré presmerovávali na stránky externých poskytovateľov, a následne o endpointy, ktoré slúžili ako návratové body (`redirect URIs`), teda Uniform Resource Identifiers. Tieto endpointy sú podrobnejšie spomínané v podkapitole venovanej externým poskytovateľom identít 4.5.

Dokumentácia a verifikácia pomocou OpenAPI

Všetky hlavné endpointy sú zdokumentované pomocou OpenAPI špecifikačného jazyka. Táto dokumentácia poskytuje rýchly náhľad na ponúkané možnosti systému. Dokumentácia pre otvorené verejne prístupné endpointy môže byť v budúcnosti zverejnená pre širokú verejnosť.

Generovanie dokumentácie využíva Swagger, konkrétne jeho nadstavbu na frameworkom Flask, teda Flasgger. Flasgger nám umožňuje pomocou dekorátorov pri jednotlivých endpointoch definovať `.yml` súbory alebo priamo v komentároch popísať daný endpoint.

Výsledkom špecifikácie je vygenerovanie webovej stránky, kde sú jednotlivé endpointy prehľadne popísané. Webstránka je naviazaná na URL backendu a využífa suffix `/nerd/apidocs`.

Špecifikácia Open API môže byť použitá aj na pridanie funkcionality overovania a vynucovania špecifických formátov pri dopytovaní. Implicitne je schopná z dokumentácie overiť, či sa k danému endpointu pristupuje správne. Chceme zistiť aj, či boli dodané všetky potrebné parametre, a či sedia ich názvy a typy.

Využitie overovania pomocou OpenAPI sa obzvlášť hodí pri kontrolách parametrov schematicky zložitých dopytov typu POST. Najkomplexnejším z týchto dopytov je práve samotné vyhľadávanie, kde sa štrukturované dáta z vyhľadávacieho formulára na frontende posielajú REST API vo formáte JSON. V niektorých parametroch chceme povoliť len špecifickú podmnožinu povolených textových reťazcov, čo OpenAPI umožňuje s využitím parametra `enum`.

4.3 Lokálna správa používateľov

Správa používateľov predstavuje v rámci implementácie nového systému rozsiahlu modifikovanú oblasť. Bolo tu potrebné domyslieť a implementovať najväčšie množstvo novej funkcionality. Pôvodný systém pozostával z registrácie pomocou EduGain, a následne čakania na schválenie účtu administrátorom a lokálneho prihlasovania. Neboli tu prítomné doplňujúce funkcionality, ako napríklad riešenie zabudnutého hesla, riešenie neverifikovanej emailovej adresy a opätovného zaslania verifikačného emailu, zobrazenie používateľov pre admina a ďalšie.

4.3.1 Databáza používateľov v PostgreSQL

Nový backend systému NERD stavia na použití existujúcej databázy a tabuľky používateľov, ktorá využíva PostgreSQL. Využitie existujúcej databázy zabezpečí hladší prechod na nový systém, zachovanie pôvodných údajov o používateľoch a správcom systému odoberá z náročnosti a časovej zložitosti práce s novou technológiou alebo štruktúrou dát.

Existujúca tabuľka na evidenciu používateľov `users` má nasledujúcu štruktúru:

```
        Column          |           Type              | Nullable |
------------------------+-----------------------------+----------+
 id                     | character varying           | not null |
 groups                 | character varying[]         | not null |
 name                   | character varying           |          |
 email                  | character varying           |          |
 org                    | character varying           |          |
 api_token              | character varying           |          |
```

```
 rl_bs                    | real                        |          |
 rl_tps                   | real                        |          |
 verified                 | boolean                     |          |
 password                 | character varying           |          |
 verification_email_sent  | timestamp without time zone |          |
 last_login               | timestamp without time zone |          |
Indexes:
    "users_pkey" PRIMARY KEY, btree (id)
```

Parameter `id` v tabuľke slúžia ako jedinečný identifikátor používateľa a v pôvodnej implementácii mal tvar:

```
local:<user_name>
```

pre bežného používateľa. Alebo:

```
devel:<user_name>
```

pre automaticky prihlásený developerský účet s funkcionalitou admina.

Stĺpec s menom používateľa `user_name` (v DB len `name`) nebude v novej verzii systému NERD využívaný. Stĺpec `id`, avšak zachováme ako primárny kľúč tabuľky (teda jednoznačný identifikátor riadku tabuľky). Preto implementujeme nasledujúce zmeny:

- na lokálne prihlasovanie bude slúžiť `email`, ktorý nemusí byť unikátny (unikátne je až `id`, teda kombinácia prefixu a emailu),
- parametre `id` pre lokálne registrovaných používateľov zostanú v tvare `local:<user_email>`,
- pre používateľov registrovaných prostredníctvom externých poskytovateľov identity bude tvar: `<id_provider>:<user_email>`.

Ukážky identifikátorov nových používateľov v existujúcej tabuľke `users`:

- `local:admin@email.com` – lokálny používateľ s emailom: admin@email.com,
- `google:user@email.com` – používateľ registrovaný cez externého poskytovateľa identity (Google) s emailom: user@email.com.

4.3.2 Ukladanie session používateľa

Pôvodný systém využíval `flask session`, ktorá mohla byť priamo odovzdávaná medzi integrovanými časťami backendu a frontendu. V novom systéme ale bolo potrebné zvoliť riešenie vhodné pre oddelené súčasti webu a jeho komunikácie s API. Týmto riešením je použitie autorizačných tokenov JWT (JSON Web Token).

JSON Web Token

Obsah nasledujúcej časti o JWT bol prevzatý z [15].

JSON web token (JWT), je otvorený štandard definovaný v špecifikácii RFC 7519, ktorý definuje kompaktný a nezávislý spôsob bezpečného prenosu informácií medzi stranami v podobe objektu JSON. JWT je štandardizovaný, teda môžeme konštatovať, že všetky JWT sú tokeny, ale nie všetky tokeny sú JWT.

Vďaka svojej relatívne malej veľkosti môže byť JWT posielaný cez URL, vrámci POST parametrov alebo vnútri HTTP hlavičiek a stále zaručuje rýchly prenos. JWT v sebe obsahuje všetky potrebné informácie, ktoré zabezpečia, že sa vyhneme zbytočným dopytom do databázy. Príjemca JWT nemusí kontaktovať žiaden externý server na overenie tokenu.

Výhody používania JWT tokenov

V porovnaní s jednoduchými web tokenmi (simple web tokens – SWTs) a Security Assertion Markup Language (SAML) tokenmi JWT ponúkajú:

- väčšiu kompaktnosť – JSON je stručnejší ako XML, takže po zakódovaní do webového prenosu vedie k menším veľkostiam ako SAML tokeny, preto sa JWT obzvlášť hodí na využívanie v HTML a HTTP prostrediach,
- bezpečnosť – JSON Web tokeny môžu používať kombináciu verejných a súkromných kľúčov na vytvorenie a podpisovanie X.509 certifikátu. JWT môže byť taktiež symetricky podpísaný spoločným tajomstvom,
- rozšírenosť – spracovanie JSON dát je rozšírenou súčasťou väčšiny programovacích jazykov, nakoľko je možné ich priamo namapovať na objekty. Naopak XML priame mapovanie na objekty nepodporuje, aj preto sú JWT na prácu a ďalšie spracovanie menej náročné,
- jednoduchosť spracovania – JWT sú využívané po celom internete, sú obzvlášť jednoduchšie pre spracovanie na mobilných klientoch.

Štruktúra JWT

Vo svojom kompaktnom formáte sa JWT skladá z troch častí oddelených znakom bodky (`.`), tieto časti sú nasledujúce:

- hlavička,
- payload,
- podpis.

Základným princípom autentizácie pomocou JWT je, že po úspešnom prihlásení používateľa je mu vrátený ID token. Podľa štandardu OpenID Connect (OIDC) je ID token vždy zároveň JWT tokenom. V nasledujúcich častiach je bližšie predstavený princíp autentizácie za použitia JWT využívaný v novom backende systému NERD.

Implementácia JWT v novom systéme NERD

V rámci implementácie novej REST API v2 bolo potrebné zaviesť endpointy, ktoré budú umožnovať správu používateľov. Tieto endpointy (okrem `/login` a `/register`) musia byť chránené a prístup k nim overený.

Jedná sa napríklad o endpointy typu zmena hesla. Pri takomto type endpointu potrebujeme overiť nasledujúce:

- či je používateľ prihlásený,

- aká je rola používateľa,
- či má používateľ, ktorý je prihlásený, právo vykonať danú operáciu (teda snaží sa napríklad heslo zmeniť vo svojom profile alebo je adminom a môže heslo zmeniť hocikomu).

Identifikácia používateľa, ktorá chráni API endpointy musí byť uložená a spravovaná frontendovou implementáciou Vue,js, ktorá si identifikátor uloží pomocou svojho `store` modulu. Následne bude tento identifikátor zasielaný pri každom HTTP dopyte v hlavičke `Authorization`.

Naivný prístup v podobe uloženia jedinečného identifikátora (napríklad emailovej adresy) úspešne prihláseného používateľa do lokálneho úložiska, je z hľadiska bezpečnosti bezpredmetný. Hocikto by bol schopný parameter v lokálnom úložisku prepísať a získať tým prístup nad zdrojmi daného profilu, na ktorý sa emailová adresa viaže.

Uložený identifikátor v lokálnom úložisku nevieme spraviť nemenným, vieme ho ale spraviť na prvý pohľad nečitateľným a zašifrovaným. Preto ako identifikátor použijeme v predchádzajúcej kapitole spomínaný JWT.

Generovanie JWT na backende

JWT nám okrem uloženia emailovej adresy používateľa ponúkne priestor aj na uloženie dodatočných parametrov. Konkrétne v súbore nového backendu inplementujúceho fnkcionalitu API v2 `api_v2.py` vo funkciii `login_user_v2()` využívame nasledujúci kód na uloženie potrebných parametrov, nastavenie doby expirácie, zakódovanie JWT pomocou nami zvoleného tajomstva a algoritmu HS256, a následné odoslanie hotového JWT.

```
    out = {}
    out['user'] = {
        'login_type': 'local',
        'id': user['id'],
        'email': user['email'],
    }
    out['exp'] = datetime.utcnow()+timedelta(hours=4)
    encoded_jwt = jwt.encode(out, "nami_zvolene_tajomstvo", algorithm="HS256")
    return encoded_jwt
```

Uloženie JWT do lokálnej pamäte na frontende

JWT je teda vracané po úspešnom logine, na strane frontendu ho získame ako odpoveď z dopytu `/login` a uložíme do lokálnej pamäte prehliadača.

```
// LoginScreen.vue
    let accessToken = null;
    try {
        accessToken = await api.login(this.email, this.password);
    } catch (e) {
        this.message = e.message;
        this.$refs.myRefError.open();
        this.loading = false;
        return;
```

```
    }
    api.setAxiosAccessToken(accessToken);
// api.js
    export function setAxiosAccessToken(accessToken) {
      if (accessToken) {
        axios.defaults.headers.common.Authorization = `${accessToken}`;
      } else {
        delete axios.defaults.headers.common.Authorization;
      }
    }
```

Prikladanie JWT do dopytov a ich overenie na strane backendu

Prikladanie `accessTokenu` je po jeho uložení do `axios.defaults.headers.common.Authorization` (v predchádzajúcej ukážke)zabezpečované automaticky, pri každom dopyte cez knižnicu `axios`.

Na strane backendu sa kontrola autentizačných hlavičiek vyžaduje len pri niektorých endpointoch. Konkrétne pri prístupe k:

- používateľským endpointom:`/me` a `/change_password`,
- administrátorským endpointom:`/uesre` a `/roles`.

Tieto endpointy sú označené dekorátorom `@token_required`, ktorý je bližšie definovaný v súbore `auth.py` vo funkcii `token_required()`. Táto funkcia vykoná nasledujúce kroky:

1. Overenie prítomnosti Authorization hlavičky.
2. Dekódovanie tokenu v Authorization hlavičke (očakávaná syntax: `Authorization <token>`).
3. Overenie existencie používateľa v DB.
4. Priradenie dát prislúchajúcich aktuálnemu overenému používateľovi do kontextu aktuálneho používateľa.
5. Vrátenie kontextu aktuálneho používateľa.

4.3.3 Registrácia používateľov a potvrdenie emailovej adresy

Počas vytvárania nového účtu pomocou registrácie prebiehajú viaceré overenia:

- emailová adresa – musí byť v správnom formáte a ešte nezaregistrovaná v systéme,
- heslo – overenie bezpečnosti (dĺžka + použité symboly), overenie zhodnosti (zadávané dvakrát na kontrolu),
- zamýšľaná oblasť využitia systému – výber zo zoznamu ponúkaných možností.

Po úspešnom overení všetkých polí registračného formulára (kontrola, okrem duplicity emailu, sa vykonáva na strane frontendu) sú údaje odoslané backendu, ktorý vloží nového

používateľa do DB, nastaví mu prázdne role a odošle emailovú správu na potvrdenie emailovej adresy.

Odoslaná emailová správa obsahuje linkovú adresu na verifikáciu, ktorej parametrom je aj JSON Web Token. Práve podpísaný token v sebe nesie informáciu o tom, ktorá adresa sa má overiť. Po kliknutí na odkaz je používateľ presmerovaný na stránku frontendu, kde počká na priebeh verifikácie. Po úspešnej verifikácii je mu oznámené, že sa už môže prihlásiť. Jeho účet sa tým stáva aktívnym. Link v emaily má platnosť 24 hodín.

Okrajové prípady verifikácie emailovej adresy

Okrajové prípady, ktoré bolo treba ošetriť boli:

- používateľ klikne na overenie emailovej adresy opäť v priebehu 24 hodín (JWT je stále platný) – riešené informovaním, že emailová adresa už bola aktivovaná,
- používateľ klikne na overenie emailovej adresy opäť po 24 hodinách – riešené informovaním, že emailová adresa už bola aktivovaná,
- používateľ klikne na overenie emailovej adresy prvý krát až po 24 hodinách – riešené informáciou o vypršaní platnosti aktivačného linku a inštrukcie/presmerovanie na opätovné zaslanie verifikačného emailu.

Zabránenie predčasnému odosielaniu verifikačných emailov

V databáze pri každom používateľovi evidujeme aj položku `Verification email sent`, ktorá v sebe uchováva údaje o dátumoch posledných odoslaní verifikačných emailov. Používateľ si môže vyžiadať nový verifikačný email až po uplynutí doby platnosti predchádzajúceho linku v emaily (teda doba vypršania JWT – 24 hodín).

Takisto pri pokuse o prihlásenie sa emailovou adresou, ktorá je v systéme evidovaná ako nepotvrdená, je používateľ dodatočne vyzvaný na potvrdenie tejto adresy, prípadne je mu ponúknutá možnosť opätovného zaslania verifikačného emailu (pri splnení podmienky o uplynutom čase od posledného odoslania špecifikovanej vyššie).

4.3.4 Rozlišovanie poskytovateľov identít a rozdielne prístupy k prihlasovaniu

V predchádzajúcej podkapitole bola vysvetlená tvorba nových unikátnych identifikátorov používateľov. Tieto identifikátory v sebe nesú aj informáciu o tom, cez ktorého poskytovateľa identity bol profil zaregistrovaný. Pomocou tejto informácie vieme teda, počas loginu používateľa overiť, či sa jedná o lokálny profil (bežné prihlasovanie) alebo o externého poskytovateľa. V prípade externého poskytovateľa sa od používateľa očakáva prihlásenie cez toho daného poskytovateľa.

Teda napríklad používateľ user1@email.com, ktorý použil na registráciu svojho účtu poskytovateľa Google, a je teda vedený v DB pod `id: google:user1@email.com` nemá vytvorený tradičný lokálny účet a nemôže sa lokálne prihlásiť svojim emailom a heslom.

Pri účtoch vytvorených cez externých poskytovateľov identity neevidujeme v databázy heslo používateľa. Nemôže si ho ani zmeniť v možnostiach profilu. Jediné, čo vieme používateľovi ukázať v prípade pokusu o lokálne prihlásenie, je chyba, že účet neexistuje, prípadne rozšírená poznámka s odporúčaním na prihlásenie cez svojho poskytovateľa identity, ktorého konkrétne v DB evidujeme.

4.4 Nová rozšírená funkcionalita systému NERD

Nové webové rozhranie so sebou prináša aj novú funkcionalitu. Nápady na implementovanie dodatkov boli známe dlhšie, avšak implementovať ich do pôvodného systému bolo zdĺhavé práve kvôli obtiažnej integrácii s existujúcimi knižnicami jazyka JavaScript.

4.4.1 Formátovanie emailov pre používateľov

Odosielané emaily v pôvodnej implementácii systému NERD boli tvorené iba textom. Konkrétne emailová správa na potvrdenie emailovej adresy novo registrovaného používateľa obsahovala okrem textu aj odkaz URL, ktorý bol do klikateľnej podoby automaticky formátovaný emailovým klientom.

V novej verzii systém odosiela emaily naformátované pomocou HTML. Emaily majú prehľadnejšiu štruktúru a namiesto URL adresy s priamym odkazom obsahujú nalinkovaný odkaz s využitím HTML tagu `<a href=...></a>`. V rámci zachovania vysokej kompatibility odosielané emaily obsahujú aj plaintextovú alternatívu.

4.4.2 Konzola pre admina

Adminská konzola predstavuje novú rozšírenú funkcionalitu pre používateľov s rolou admin. Môžu tu vidieť zoznam všetkých používateľov, ich role, identifikátori, príslušnosť do organizácie, API tokeny pre prístup k API v1 a ďalšie potrebné informácie. Tabuľka zrkadlí PostgreSQL tabuľku Users a zastrešuje funcionalitu, ktorá bola pôvodne dostupná len s použitím skriptov, teda mimo pôvodného webového rozhrania systému NERD.

Čo však systém NERD mal aj pôvodne bol admin box (miesto pre rýchle zmeny rolí, hoci aj samotného admina). Na ponechanie tejto funkcionality vznikla nová rola superadmin, ktorý má ako jediný možnosť odobrať alebo pridať si aj rolu admina. Odôvodnenie tejto funkcionality je, že niekedy je potrebné pozrieť sa, čo presne je viditeľné adminovi, a čo naopak nie. Rovnako je možné prestavovať role ľubovoľným používateľom (pre zvýšenie úrovne prístupu k dátam, alebo naopak jej zníženie).

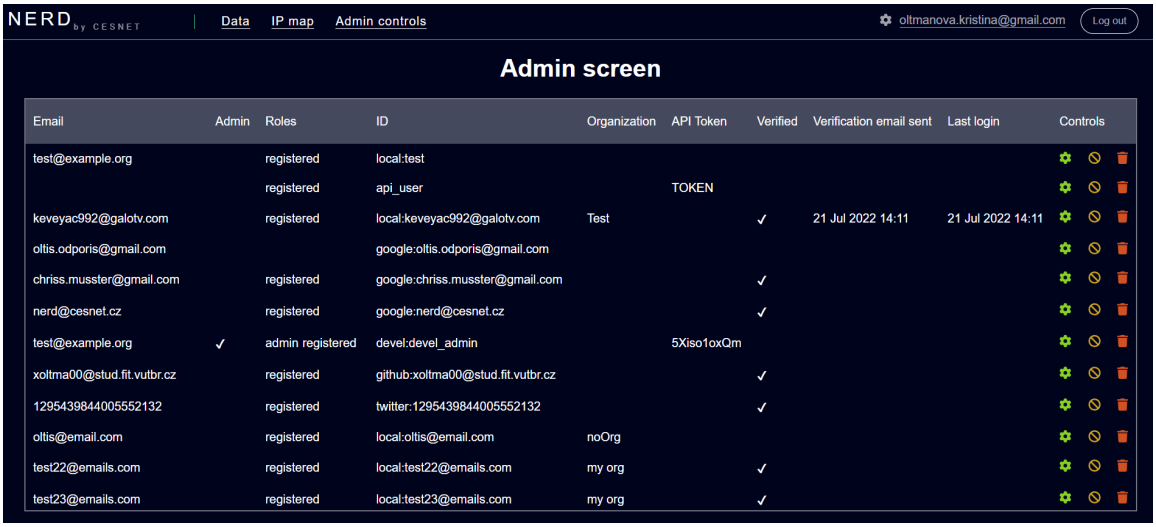
Pre každého používateľa je vo výslednej zobrazenej tabuľke 1 riadok. Na konci riadka sú k dispozícii tri základné možnosti úpravy daného používateľa:

- nastavenia rolí používateľa – zmena rolí používateľa: odobratie, pridanie jednotlivých rolí pomocou zaškrtávacích políčok,
- zablokovanie používateľa – používateľovi bude odobraná rola (registered), čo mu znemožní prihlásenie sa,
- zmazanie používateľa – používateľ bude kompletne odstránený z databázy.

Po kliknutí na vybranú možnosť sa používateľovi zobrazí modálne okno, v ktorom buď vyplní potrebné dodatočné údaje, alebo potvrdí nemennú akciu (akou je napríklad vymazanie používateľa z databáze).

Admin má takisto možnosť vytvoriť účet pre nového lokálneho používateľa. Zobrazí sa mu modálne okno, v ktorom môže vyplniť všetky potrebné polia. Má dokonca možnosť používateľa rovno aj verifikovať, a to v prípade ak nechce odosielať verifikačný email na špecifikovanú emailovú adresu.

Ukážka konzoly pre admina je na obrázku 4.6.



Obr. 4.6: Ukážka konzoly pre admina

4.4.3 Mini grafy histórie reputačného skóre

Jednou z dodatočných požiadaviek nového systému bolo umiestnenie malých grafov do riadkov výsledkovej tabuľky. Tieto grafy predstavujú históriu reputačného skóre v čase. Počas riešenia diplomovej práce nebola dokončená externá podpora zhromažďovania a ukladania historických dát reputačného skóre, preto je v rámci frontendu predpripravená len grafická funkcionalita.

Výkonnostné požiadavky vykresľovania mini grafov

Pri výbere technologického postupu vykresľovania grafov bola zvážená aj knižnica `Chart.js`, ktorá v svojej podstate využíva HTML element `<canvas>` a vkladá do neho grafické objekt podľa potreby. Využitie tejto knižnice (resp. jej nadstavby nad Vue.js `vue-chartjs`, ktorá je využitá aj na vykresľovanie grafov v stránkach s detailami) by viedlo na zbytočné výkonnostné zaťaženie klientských zariadení, ktorá by vykresľovanie funkcie jazyka JavaScript museli volať pre každú stránky s výsledkami niekoľko desiatok krát.

Vzhľadom na veľkosť, jednoduchosť aj zamýšľanú výpovednú hodnotu mini grafov v riadkoch tabuľky bola napokon na ich realizáciu zvolená knižnica `vue-charts-css`. Táto knižnica je schopná vykresliť mini grafy čisto za použitia kaskádových štýlov CSS. Podobu grafu dosahuje pomocou HTML elementov tabuľky. Nevyžaduje sa žiadne dodatočné plnenie alebo zbytočné iterácie cez zobrazované hodnoty, aj preto je implementované riešenie na celkovom vyťažení stránky s výsledkami nebadateľné.

Prijímanie dát na vykreslenie mini grafu

V implementácii majú mini grafy vlastný komponent (`/components/MiniChart.vue`). Tomuto komponentu je možné dodať rôzne dáta, pričom ich očakáva v tvare listu (array). List predpokladá hodnoty reputačných skóre 0-1 (ale akceptuje ľubovoľné). Z hľadiska grafického návrhu je odporúčané zobrazovať v grafe 4-7 hodnôt, teda poslať komponentu argument s listom o 4-7 numerických hodnotách.

Umiestnenie mini grafov

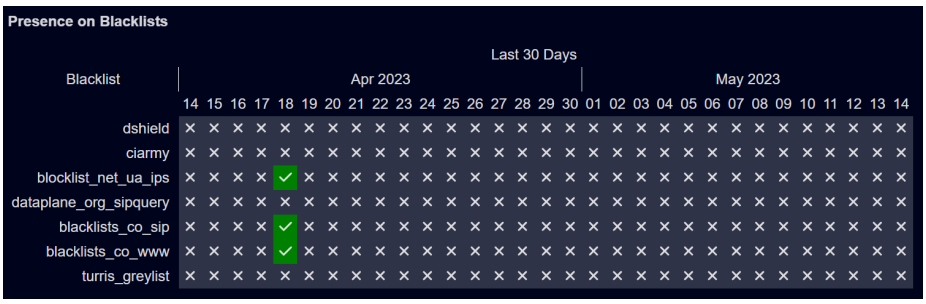
Mini grafy zobrazujú históriu reputačného skóre, preto sú zobrazované priamo v jeho riadku. Hneď za minigrafom je umiestnená desatinná hodnota reputačného skóre, ktorá je farebne odlíšená. Farby sa generujú podľa hodnoty reputačného skóre. Pre hodnoty blízke nule sú zelené, a pre rastúce hodnoty postupne prechádzajú do červených farieb. Umiestnenie minigrafov v rámci tabuľky je možné vidieť na obrázku 4.7.

Obr. 4.7: Ukážka umiestnenia mini grafu zobrazujúceho históriu reputačného skóre

4.4.4 Zobrazenie prítomnosti na blacklistoch za posledných tridsať dní

Pôvodný návrh grafického používateľského rozhrania obsahoval vykresľovanie prítomnosti IP adresy na blacklistoch ako grafu. V novom webovom systéme bola využitá varianta zobrazenia v tabuľke, ktorá sa snaží pripomínať zobrazenie časovej osi, prípadne kalendára. Tabuľka je implementovaná pomocou vlastného komponentu a podporuje priame zadanie dát o blacklistoch s minimom predspracovania. Táto implementácia vedie na pružné a efektívne riešenie.

Na obrázku 4.8 môžeme vidieť zobrazenie v podobe tabuľky. Zelená farba v kombinácii s ikonou indikuje prítomnosť na blackliste. V záhlaví tabuľky je zobrazená časová os s mesiacmi a dňami.



Obr. 4.8: Ukážka grafického zobrazenia prítomnosti IP adresy na blacklistoch za posledných tridsať dní

4.5 Prihlasovanie pomocou externých poskytovateľov identít

Poslednou s požiadaviek na nové webové rozhranie bola podpora externých poskytovateľov identít. Do úvahy prichádzalo množstvo poskytovateľov, ktorí ponúkajú OAuth prihlasovanie pomocou svojich API. Vzhľadom na oblasť fungovania aplikácie boli zvolený poskytovatelia Google, ktorý zastrešuje pomerne veľkú časť potenciálnych externých prihlásení. V rámci bezpečnostnej komunity je taktiež vhodné ponúkať ako poskytovateľa Twitter, ktorý môžeme považovať za najkonzervatívnejšiu voľbu spomedzi sociálnych sietí. Pre komunitu vývojárov, ktorý by taktiež mohli mať záujem o prístup do systému NERD je tu aj možnosť prihlásiť sa cez poskytovateľa GitHub.

Po výbere vhodných poskytovateľov identít bolo potrebné si stanoviť ako budú zapadať do existujúceho systému správy používateľov. Systém na správu lokálnych používateľov využíva primárne emailové adresy, ktoré však v prípade viacerých poskytovateľov nemôžeme považovať za unikátne identifikátory. Ako pozostatok z pôvodného návrhu databázy používateľov v PosgreSQL existuje v záznamoch aj položka `id` pre každý užívateľský profil. V rámci lokálnych prihlásení je id tvorené reťazcom `local:<email>`, kde email je emailová adresa používateľa. Na rozlíšenie rôznych poskytovateľov môžeme teda použiť prefixy id používateľa (`google:<email>`, `twitter:<email>` a `github:<email>`).

Za unikátny identifikátor používateľa v databáze považujeme jeho `id`. Lokálne prihlásenie sa napríklad pod adresou `nerd@cesnet.cz` vedie na vytvorenie lokálneho profilu `local:nerd@cesnet.cz`. Po nasledujúcom pokuse o prihlásenie sa napríklad pomocou GitHub účtu využívajúceho rovnakú emailovú adresu `nerd@cesnet.cz`, bude zase vytvorený nový profil `github:nerd@cesnet.cz`. Profily sa nezlučujú. Je očakávané, že používateľ bude používať vždy rovnaký spôsob prihlásenia na rovnaké účty. Prihlasovanie sa cez externých poskytovateľov navyše nevyžaduje evidovanie hesiel v systéme a práve zlučovanie lokálnych profilov s externými by mohlo viesť k nežiadúcim efektom.

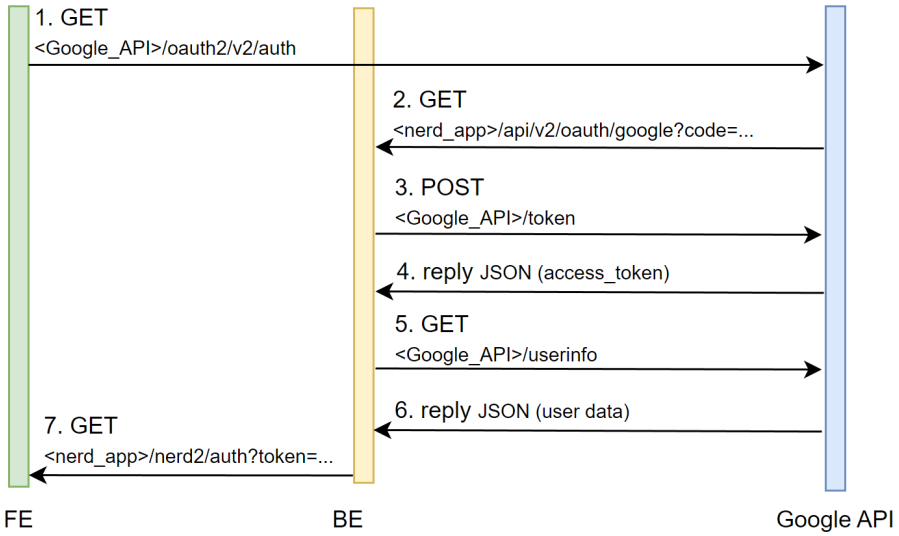
Posledným z podporovaných poskytovateľov identít je EduGain postavený na autentizačnom systéme Perun, ktorý využíva OpenID. V pôvodnom systéme už bolo podporované prihlasovanie používateľov pomocou EduGain, avšak bolo postavené na staršom prístupe Shibboleth. V rámci interných noriem spoločnosti Cesnet bude do budúcna vyžadovaná podpora práve prihlasovania EduGain cez Perun, a preto rozhodne nesmie chýbať ani v novom modernom systéme. Na rozdiel od predošlých poskytovateľov identity, ktorí budú v systéme úplne noví, musíme pri EduGain rátať s tým, že používateľ teoreticky už mohol mať vytvorený profil cez tohto poskytovateľa v staršom systéme. Musíme taktiež zvoliť vhodný postup, čo v takom prípade z profilom urobiť. Postup pre poskytovateľa EduGain je bližšie opísaný v podkapitole 4.5.4.

Nasledujúce podkapitoly se bližšie venujú implementácii prihlasovania cez konkrétnych poskytovateľov.

4.5.1 Google

Prvým krokom pri implementácii Google OAuth prihlasovania bolo získanie parametrov `client_id` a `client_secret` z konzoly pre developerov. Google v tejto konzole umožňuje vytvorenie novej aplikácie, vyplnenie potrebných údajov a následné získanie prístupu. Novú aplikáciu NERD bolo potrebné naviazať na NERD emailový účet, ktorý bude môcť slúžiť pre prípadné dodatočné administratívne zmeny. Obdobne boli vytvorené účty aj na zvyšných dvoch platformách (Twitter a GitHub) a boli z ich príslušných developerských nástrojov získané klientské identifikátory a tajomstvá.

Proces komunikácie, ktorý zahŕňa kroky od kliknutia na tlačidlo prihlásenia až po samotné prihlásenie v systéme NERD je načrtnutý na obrázku 4.9.



Obr. 4.9: Nákres komunikácie NERD aplikácie a Google API pri prihlasovaní používateľa

Zostavenie prihlasovacej URL

Prihlasovanie sa pomocou externého poskytovateľa navonok zahŕňa len dva kroky. Musíme mať tlačidlo, ktoré nás informuje, že klikom naň sa vieme prihlásiť pomocou niektorého poskytovateľa. Toto tlačidlo navyše musí obsahovať presmerovanie, ktoré nám naozaj otvorí stránku poskytovateľa. Na správne fungovanie to však nestačí. Poskytovateľ musí vedieť, ktorá služba o autenitizáciu žiada. Identifikátorom služby je v prípade poskytovateľa Google získaný `client_id`, ktorý musí byť súčasťou dopytovanej prihlasovacej URL adresy. Kompletná dopytovaná URL adresa má nasledujúcu štruktúru:

```
rootUrl: https://accounts.google.com/o/oauth2/v2/auth
queryArgs:
    - client_id: <client_id>
    - redirect_uri: <our_app>/nerd/api/v2/oauth/google
    - response_type: code
    - prompt: consent
    - scope: https://www.googleapis.com/auth/userinfo.profile,
             https://www.googleapis.com/auth/userinfo.email
```

Ako sme už načrtli pomocou `client_id` Google zistí, do ktorej služby sa používateľ bude snažiť prihlasovať. Argument `redirect_uri` je vypĺňaný aj v úvodnej konfigurácii služby a obsahuje návratovú URL adresu. Ak prihlásenie cez Google prebehne správne, Google presmeruje používateľa naspäť do aplikácie práve presmerovaním na špecifikovanú `redirect_uri`. Argument `response_type` špecifikuje, aký formát odpovede aplikácia očakáva. V našom prípade je zvolený `code`. To znamená, že po úspešnej autentizácii bude Google kontaktovať `redirect_uri`, a ako špeciálny parameter nám zašle `code`. Posledný argument `scope` môže obsahovať viacero hodnôt, ktoré udávajú k akým informáciám od

používateľa naša služba požaduje prístup. Našej službe stačia základné informácie o profile a email prihláseného používateľa.

Všetky tieto argumenty odošleme GET dopytom základnej URL (`root_url`). Vedie to na presmerovanie používateľa na Google prihlásenie. Vyššie spomínané zostavovanie prihlasovacej URL môže prebiehať aj priamo na frontende aplikácie. V nasledujúcej časti si predstavíme finálne kroky autentizácie, ktoré už musia prebiehať na backende.

Návrat na špecifikovanú `redirect_uri`

V prípade úspešného prihlásenia používateľa Google odošle parameter `code` na `redirect_uri`. Endpoint `redirect_uri` musí byť súčasťou REST API v2 na backende. Je to endpoint v tvare `<our_app>/nerd/api/v2/oauth/google`, ktorý očakáva odpoveď od služby Google. V prípade kontaktovania tohto endpointu metódou GET prebehnú nasledujúce kroky.

1. Overenie, že súčasťou dopytu je aj parameter `code`.
2. Odoslanie POST dopytu na Google API, ktorý prevedie `code` na `access_token`.
3. Využitie získaného `access_token` na odoslanie dopytu GET na `/userinfo` endpoint.
4. Overenie správnosti odpovede a získanie emailu používateľa.
5. Zistenie, či používateľ s danou emailovou adresou už je registrovaný (`id: google:<email>`), ak nie vytvoriť profil.
6. Vystavenie a vrátenie vlastného tokenu, kontaktovanie frontendu.

Vo vyššie uvedených krokoch kontaktujeme Google ešte dvakrát. Prvýkrát chceme z dodaného `code` získať `access_token`, ktorý nás následne podľa vyžiadaných `scopes` oprávňuje k prístupu k požadovaným informáciám. K informáciám sa dostaneme kontaktovaním `/userinfo` endpointu s priložením `access_token`. Google vráti požadované informácie len v prípade správnosti `access_token` parametra. Následne využijeme získaný email na overenie prítomnosti daného používateľa v našej databáze. Ak ešte nemá vytvorený profil, tak mu ho vytvoríme. Heslo pri profiloch externých poskytovateľov nepotrebujeme evidovať. Rovnako nepotrebujeme overovať emailovú adresu, predpokladáme, že ju už overil externý poskytovateľ.

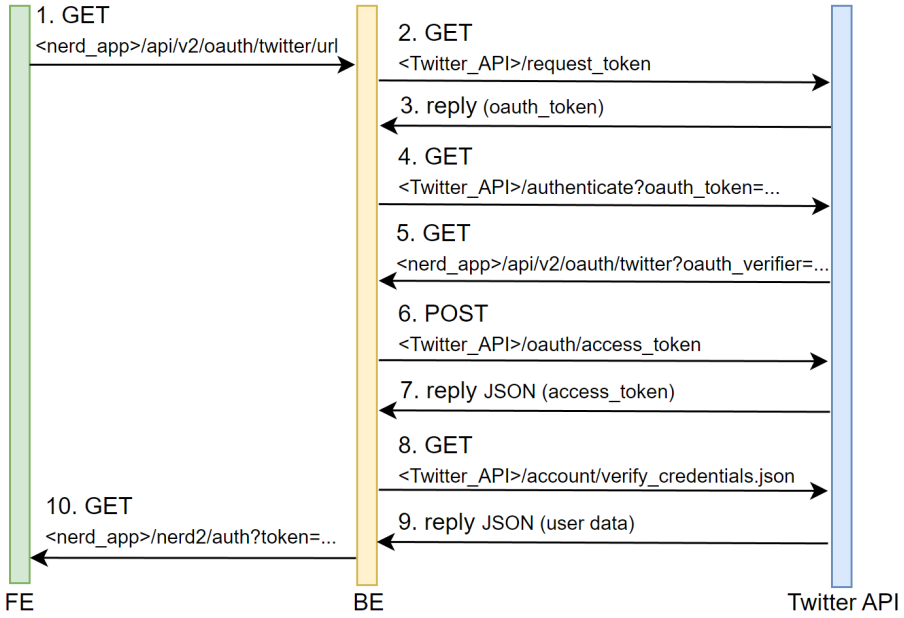
Návrat do frontendu

Na frontende kontaktujeme presmerovaciu stránku `/auth`, v argumetoch špecifikujeme `token` a `refreshToken`. Aby sme mohli využívať naše autentizačné overenia všeobecne, vydávame vlastné prihlasovacie tokeny aj pri prihlasovaní cez externých poskytovateľov.

Presmerovacia stránka na frontende zachytí dané argumenty z URL a uloží ich do lokálnej pamäti prehliadača. Vykonané kroky sú totožné z lokálnym prihlásením a na záver je používateľ presmerovaný na domovskú stránku. V prípade hladkého priebehu tak používateľ presmerovanie na stránku `/auth` ani neeviduje.

4.5.2 Twitter

Pri implementácii prihlasovania pomocou poskytovateľa Twitter bola použitá Twitter dokumentácia pre developerov [22] a postup pre Python [4]. Postup komunikácie forntedu, backend a Twitter API je znázornený na obrázku 4.10.



Obr. 4.10: Nákres komunikácie NERD aplikácie a Twitter API pri prihlasovaní používateľa

Zostavenie prihlasovacej URL

Pri poskytovateľovi Twitter potrebujeme ešte pred samotným získaním prihlasovacej URL kontaktovať Twitter API so zámerom získania parametra `oauth_token`. Twitter ho totiž vyžaduje vo volaní prihlasovacej URL. Oproti prihlasovaniu pomocou poskytovateľa Google nám teda pribudol jeden krok. Z frontendu nevieme používateľa presmerovať na prihlasovací Twitter endpoint priamo, najskôr backend požiada Twitter API o pridelenie `oauth_token` parametru. Následne je proces podobný ako pri providerovi Google.

Vyžiadanie emailovej adresy používateľa

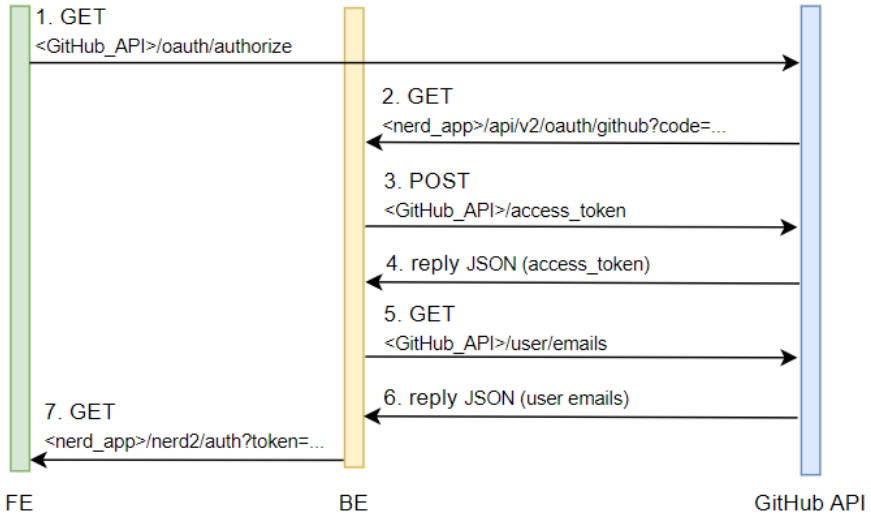
Twitter narozdiel od Google neposiela emailovú adresu v dátach používateľa automaticky, na vyžiadanie musíme špecifikovať parameter `'include_email': 'true'`.

4.5.3 GitHub

Pri implementácii prihlasovania pomocou poskytovateľa GitHub bola použitá GitHub dokumentácia autorizácie OAuth aplikácií [9] a implementačný postup pre Python [14]. Samotná implementácia bola takmer totožná s postupom pri poskytovateľovi Google. Graf komunikácie je viditeľný na obrázku 4.11.

Vyžiadanie emailovej adresy používateľa

Ani GitHub neposkytuje emailovú adresu prihláseného používateľa implicitne. Ponúka endpoint na získanie všetkých evidovaných emailových adries používateľa, pričom pri hlavnej, preferovanej adrese špecifikuje že email je `primary`. Informácie potrebné na získane emailovej adresy v správnom formáte boli získané z [10].



Obr. 4.11: Nákres komunikácie NERD aplikácie a GitHub API pri prihlasovaní používateľa

4.5.4 EduGain

Posledným z podporovaných poskytovateľov je EduGain. Keďže samotný systém NERD je vyvíjaný v akademickom prostredí a primárne slúži na poskytovanie informácií v rámci akademickej komunity, je potrebné pridať poskytovateľa prihlasovania aj z tejto oblasti. Na rozdiel od predchádzajúcich poskytovateľov, ktorých nebolo treba bližšie predstavovať, poskytovateľ EduGain nemusí byť všeobecne známy. Jeho bližšiemu predstaveniu sa venuje nasledujúca časť.

Predstavenie poskytovateľa EduGAIN

EduGAIN je služba federácie identít pre akademické a vedecké spoločenstvá. Táto služba umožňuje jednoduché a bezpečné poskytovanie prístupu k online službám a zdrojom, akými sú výskumné databázy, knižnice a laboratóriá. EduGAIN sa snaží o zjednodušenie procesu autentifikácie a autorizácie pre používateľov, ktorí majú prístup k rôznym online službám v rámci akademickej a vedeckej komunity. Služba je dostupná pre rôzne organizácie, akými sú univerzity, výskumné inštitúcie a organizácie z oblasti vzdelávania. Cieľom EduGAINu je umožniť rýchle a jednoduché zdieľanie zdrojov a služieb medzi rôznymi organizáciami. Takisto sa snaží o poskytovanie bezpečnej a spoľahlivej autentifikácie a autorizácie používateľov. [7]

Aj inštitúcie CESNET a VUT sú súčasťou služby EduGAIN, takže ľubovoľný člen týchto inštitúcii vie využiť svoj interný profil na austenitizáciu pomocou tohto poskytovateľa.

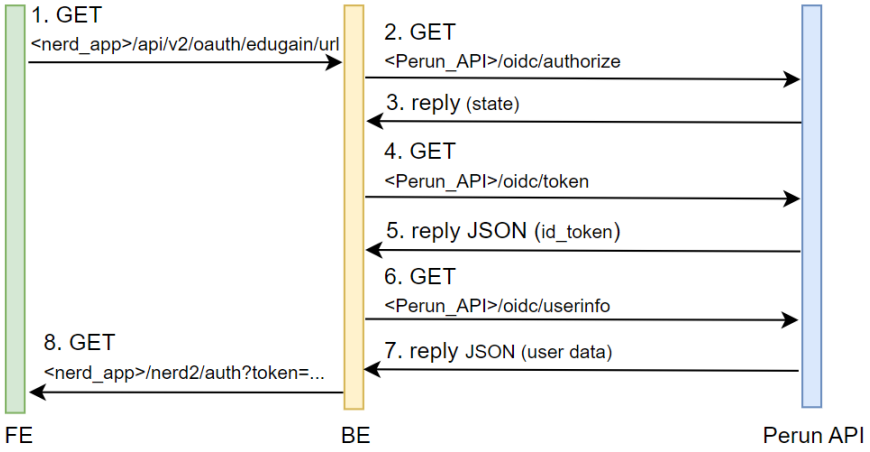
Z hľadiska technického pozadia služby EduGAIN stojí za zmienku, že EduGAIN podporuje širokú škálu systémov riadenia identít, vrátane Perun a Shibboleth, pomocou použitia štandardných protokolov, ako napríklad SAML 2.0. Ako už bolo spomenuté v predchádzajúcej podkapitole 3.1.3, pôvodný systém NERD podporoval prihlasovanie pomocou poskytovateľa EduGAIN práve cez systém Shibboleth. Nové požiadavky (v rámci internej politiky spoločnosti CESNET) špecifikujú potrebu podpory a integrácie so systémom Perun.

Perun je centralizovaný systém riadenia identít, ktorý poskytuje širokú škálu služieb na podporu riadenia prístupu v akademických a výskumných inštitúciách. [12] EduGAIN môže interagovať s Perunom pomocou protokolu SAML 2.0.

Shibboleth je open-source systém riadenia identít a prístupu, ktorý umožňuje bezpečnú autentifikáciu a autorizáciu medzi rôznymi organizáciami. [20] EduGAIN tiež používa Shibboleth na umožnenie bezpečného prístupu k zdrojom naprieč rôznymi organizáciami.

Pri implementácii prihlasovania pomocou poskytovateľa EduGAIN s využitím systému Perun a OpenID bola použitý popis parametrov Perunu [5] špecifikácia OpenID userinfo endpointu [6] a všeobecný prístup ku komunikácii pomocou OAuth v jazyku Python [19].

Schéma komunikácie, ktorá sa krokmi implementácie najviac podobá na poskytovateľa Twitter, je zobrazená na obrázku 4.12.



Obr. 4.12: Nákres komunikácie NERD aplikácie a Perun API pri prihlasovaní používateľa

Zostavenie prihlasovacej URL

Samotná implementácia poskytovateľa EduGAIN bude využívať Perun a konkrétne technológiu OpenID. K pripojeniu sú poskytnuté metadáta, z ktorých vieme získať autorizačný endpoint. Kontaktovať tento endpoint opäť nevieme priamo z frontendu (obdobne ako pri poskytovateľovi Twitter). Na URL prihlasovací endpoint nás presmeruje backend, pričom k autorizačnej požiadavke musí pridať:

- klientský kľúč – získaný po registrácii služby cez `https://spreg.aai.cesnet.cz/` (registrácia aj následné zmeny musia byť ručne odsúhlasené správcom),
- rozsah oprávnení (scopes) – budeme nás zaujímať najmä email používateľa, žiadame oprávnenia: `profile`, `email`, `openid`,
- návratová adresa (redirect URI) – návrat na náš endpoint, ktorý dokončí proces prihlasovania.

Vyžiadanie emailovej adresy používateľa

Po získaní stavového tokenu a jeho výmeny za autorizačný token vieme kontaktovať používateľský endpoint, ktorý vracia základné dáta z profilu používateľa. Jednou z vracaných

hodnôt je aj email. Pri tomto poskytovateľovi potrebujeme rátať s možnosťou, že akademický používateľ už využil prihlasovanie pomocou pôvodného systému NERD, ktorý podporoval prihlasovanie poskytovateľom EduGAIN cez Shibboleth. Takíto používatelia majú v databáze používateľov PostgreSQL uložené profily s unikátnymi identifikátormi v tvare `shibboleth:<email>`. V nasledujúcich krokoch prihlasovania preto overíme, že daná emailová adresa nie je naviazaná na nejaký takýto existujúci profil.

Ak by sme zistili, že emailová adresa už bola použitá a je naviazaná na Shibboleth profil, vytvoríme nový profil s identifikátorom `edugain:<email>` prenesieme do neho existujúce záznamy z pôvodného profilu (prenášame meno, organizáciu a API v1 token) a starý profil odstránime. Prenesenie pôvodného API v1 tokenu je dôležité. Veľa používateľov z akademického prostredia využíva webové rozhranie práve na prístup k tomuto tokenu.

Kapitola 5

Testovanie

Táto podkapitola predstavuje testovanie vykonané na nasadenom systéme. Je potrebné testovať už nasadený systém, ktorý je optimalizovaný a vrámci servera interaguje s REST API v2. Pred začiatkom samotného testovania preto najskôr implementáciu z prechádzajúcej kapitoly nasadíme na testovací server, čomu sa venuje nasledujúca podkapitola.

5.1 Nasadenie implementácie na testovací server Apache

Riešenie vývoja nového webového rozhrania systému NERD počíta s nasadením novej verzie do existujúcej používanej architektúry. V súčasnosti beží hlavná inštancia `nerd.cesnet.cz` na CentOS serveri s využitím Apache.

Na testovacie účely nasadenia bol použitý pomocný server `nerd-test.liberouter.org`, ktorý má obmedzený prístup cez VPN (Virtual Private Network). Architektúrou však kopíruje hlavný server, preto môžeme konštatovať, že výstupy z analýzy a nasadenia práve na pomocný server sú použiteľné aj na hlavný server.

5.1.1 Požiadavka zachovania existujúcej funkcionality

V rámci testovania vznikla požiadavka dočasne umiestniť nový moderný systém na podstránku `/nerd2`, čo by dovolilo nerušenú prevádzku pôvodného systému (bežiaci aj v pôvodnej implementácii na podstránke `/nerd`). Na zaistenie tejto požiadavky boli implementované nasledujúce kroky:

1. konfigurácia `webpack` v súbore `vue.config.js` upresňuje `publicPath` na `/nerd2`,
2. Flask implementácia backendu využíva blueprints a špecifikuje adresy API v2 na URLs s prefixom `/api/v2`,
3. vytvorenie podpriečinka `/nerd2` v umiestnení `/var/www/html/` na cieľovom serveri,
4. povolenie `.htaccess` pomocného súboru v `/etc/httpd/conf/httpd.conf` pomocou `AllowOverride All`,
5. presmerovanie sieťovej premávky smerujúcej na `/nerd2` pomocou `.htaccess`.

5.1.2 Kompilácia súborov frontendu

Zdrojové súbory Vue.js už boli špecifikované v podkapitole 4.1.1, táto súborová štruktúra je ale po kompilácii do produkčného prostredia zmenená. Kompilácia prebieha za použitia

príkazu `npm run build` a výsledkom je priečinok `/dist`, ktorého obsah je pripravený na nasadenie na server. Súbory boli počas zostavenia kompilované a optimalizované. Výstupné súbory sú:

- `index.html` – tento súbor je vstupným bodom do celej aplikácie a je prvým súborom, ktorý je poskytnutý klientovi. Obsahuje HTML štruktúru aplikácie, a tiež zahŕňa odkazy na ďalšie vygenerované CSS a JavaScript súbory. Taktiež obsahuje upozornenie, ktoré sa zobrazí v prípade vypnutého jazyka JavaScript.
- `/css/chunk-vendors.<hash>.css` – obsahuje celý CSS kód, ktorý je generovaný Vue.js a jeho závislosťami. Tento súbor je generovaný počas procesu zostavovania pomocou `mini-css-extract-plugin` a jeho názov obsahuje hash, ktorý sa mení pri každej aktualizácii súboru, aby sa zabránilo problémom s ukladaním do cache.
- `/js/chunk-vendors.<hash>.js` – obsahuje celý JavaScript kód, ktorý je generovaný Vue.js a jeho závislosťami. Obsahuje runtime Vue, Vue Router, Vuex a ďalšie balíky, ktoré boli nainštalované. Názov súboru taktiež obsahuje hash.
- `/js/app.hash.js` – obsahuje zostavený JavaScript kód pre všetky komponenty aplikácie. Obsahuje kompilátor šablón `Vue` a `runtime Vue`, ktoré spolupracujú na vykresľovaní aplikácie na strane klienta. Tento súbor je generovaný pomocou `vue-loader` a `webpack`.
- `/img` – tento adresár obsahuje všetky obrázky, ktoré sa v aplikácii používajú. V zdrojovom adresári musia byť obrázky uložené v `src/assets` ak chceme aby sa počas zostavovania preniesli práve do `/img` priečinka.

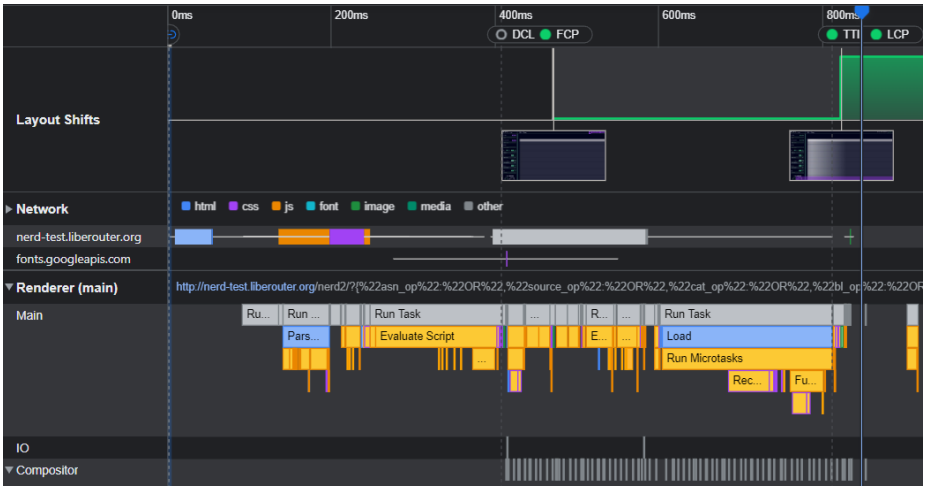
Ako si môžeme všimnúť v štruktúre výsledných súborov vyššie, Vue.js využíva statické súbory. Doterajšia implementácia využívajúca Flask používala WSGI protokol na rozhraní s Apache. Výrazným rozdielom je aj, že Vue.js aplikácia je plne vykresľovaná na strane klienta, na rozdiel of Flask aplikácie, ktorá HTML v celej komplexnosti generovala na strane servera a klientovi posielala už len výsledok. Apache server musel pri frameworku Flask zvládať statické aktíva (JS, CSS a obrázky) oddelene od kódu aplikácie. Pri Vue.js sú dodávané priamo ako statické súbory.

Vyššie spomínané výstupné súbory stačí na Apache serveri umiestniť do priečinka `/nerd2`, ktorého vytvorenie bolo bližšie predstavené na začiatku tejto podkapitoly.

5.2 Testovanie výkonnosti frontendu

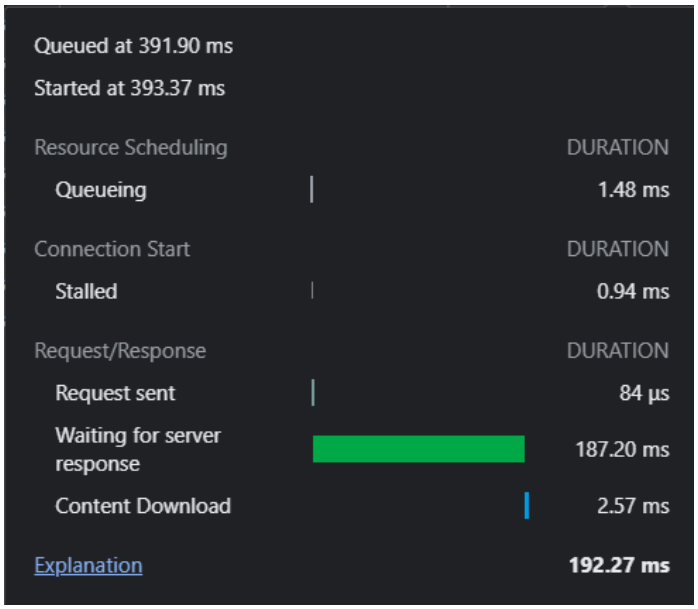
Vykonanie testov výkonnosti nového frontendu je potrebné, nakoľko nové riešenie prenáša zátaž zostavovania stránky zo strany servera na stranu klienta. Testy boli vykonávané pomocou webového prehliadača Google Chrome a jeho nástrojov pre devolperov. Testovací stroj mal operačný systém Windows s procesorom Intel (4,6 GHz) a 16 GB operačnej pamäte. Testovaná bola nasadená optimalizovaná verzia uverejnená na nerd-test serveri.

Na obrázku 5.1, ktorý zobrazuje výkonnostnú analýzu hlavnej stránky môžeme vidieť farebne odlíšené načítavania rôznych častí kódu. HTML objekty sa načítajú v úvode a následne prekreslia po získaní obsahu. Tieto oblasti vidíme modrou farbou. Oranžová nám zase ukazuje vykonávanie funkcií jazyka JavaScript na pozadí. Sivá zobrazuje dopyty na REST API. Celkový pohľad nám v hornej sekcii `Layout Shifts` zobrazuje ako sa pre používateľa menil pohľad na stránku. Teda kedy videl načítavanie a kedy už samotný obsah.



Obr. 5.1: Výkonnostná analýza hlavnej stránky

Najčastejšie používanou funkciou REST API z frontendu bude funkcia vracajúca samotné výsledky IP adries. Je to aj jeden z prvých dopytov hneď po úvodnom načítaní domovskej stránky. Z obrázka 5.2 nám vyplýva, že načítavanie odpovede na tento dopyt trvalo príjemných 192 milisekúnd.



Obr. 5.2: Časová analýza získania odpovede z REST API

Merania boli realizované pomocou zabudovanej funkcionality prehliadača Google Chrome, ktorý v možnostiach `Network` ponúka rozpis jednotlivých dopytov a ich trvaní. Merania boli opakované pre viacero kombinácií dopytov vyhľadávania a z meraní vyplynulo, že načítavanie výsledkov, pre dopyty, ktoré vedú k malému počtu vyhovujúcich záznamov, trvali dlhšie. Pri analýze príčiny sa podarilo zistiť, že niektoré špecifické dopyty na menej často vyžadované záznamy (napríklad Tag Malware) sa oneskoria kvôli chýbajúcim indexom v

databáze MongoDB. Toto oneskorenie teda vyplýva z pôvodného návrhu systému a nie je spôsobené novou nadstavbou.

Komplexné výsledky z dvadsiatich meraní viedli na priemerný výsledný čas načítavania 206.8 milisekúnd, z čoho čakanie na server priemerne zaberalo 97,48 percent času a zvyšok zaberala sieťová réžia.

5.3 Testovanie responzivity a použiteľnosti na mobilných zariadeniach

Aj keď je webové rozhranie systému NERD vyvíjané predovšetkým pre zobrazenie na počítačových obrazovkách, v rámci jeho modernizácie nemohol chýbať krok využitia responzívneho dizajnu a prispôsobovania rozloženia objektov ľubovoľnej veľkosti obrazovky.

Vývojové prostredie Vue.js ponúka po preložení aplikácie adresu na lokálne vývojové prostredie `localhost`, ale aplikáciu vysiela aj sieťou pomoocu `network URL`. Ponúkanú sieťovú adresu si môžeme otvoriť na ľubovoľnom inom zariadení v sieti a sledovať zmeny v reálnom čase aj na ňom.

V rámci testovania boli overené nasledujúce kroky:

1. Používateľské rozhranie bolo testované na rôznych zariadeniach a operačných systémoch (mobilné zariadenia: Apple aj Android, tablety: Apple aj Android). Mobilné zariadenia majú k dispozícii nižší výkon a slabšie sieťové pripojenie. testované bolo aj, či tieto faktory negatívne neovplyvňujú interakciu s webovým rozhraním systému NERD na mobilných zariadeniach.

2. Preskúmanie prechodov, ako sa stránka prispôsobuje zmenám veľkosti obrazovky pomocou Google nástrojov pre developerov (Google Dev Console). Štúdium, ako sa časti stránky menia po dosiahnutí rôznych menších šírok (s využitím možností zobrazovania `Responsive`).

3. Používateľská interakcia na mobilnom zariadení s cieľom zistenia, či je používateľské rozhranie intuitívne a ľahko ovládateľné na dotykových obrazovkách. Zistenie, či používatelia s ľahkosťou a bez zbytočného rozmýšľania vedia, čo ktoré tlačidlo znamená a ako by postupovali, keby chceli vykonať špecifické akcie. Napríklad, kde je vyhľadávací formulár, ako ho otvoriť, alebo ako získať detailnejšie informácie o IP adrese.

4. Overenie, či sa obsah na stránke správne zobrazuje, a či nie je nutné horizontálne posúvať obrazovku pre zobrazenie celej stránky.

5. Kontrola nastavení meta tagu viewport.

Nakoľko sú všetky vyššie spomínané body v novom webovom rozhraní otestované a funkčné, môžeme nové rozhranie označiť za responzívne, a teda aj použiteľné na mobilných zariadeniach.

5.3.1 Používateľské testovanie webového rozhrania

V rámci testovania webového rozhrania sme sledovali interakciu nových používateľov so systémom. Postup testovania sa nesnažil o meranie rýchlosti alebo zostrojovanie grafov pohybov kurzora po obrazovke. Hlavné informácie, ktoré nás pri používateľskom testovaní

zaujímali boli, či používatelia systému rozumejú, a aké z neho majú celkové dojmy. Ďalej nás zaujímalo ako o systéme rozmýšľajú. Systém bol testovaní na vzorke štyroch používateľov v rámci lokálnej siete, nakoľko v aktuálnej podobe nie je prístupný verejnosti.

Pri testovaní sme postupovali podľa nasledujúcich bodov:

1. Používatelia dostali notebook s otvorenou domovskou stránkou nového systému NERD. Mali opísať, aké majú dojmy z toho, čo vidia. Ďalej mali špecifikovať, na čo si myslia že daná webová stránka slúži, a čo používateľom umožňuje.

2. Ďalší krok sa zameriaval na vyhľadávanie s využitím všeobecne známeho parametra – krajiny. Používatelia sa mali pokúsiť o vyhľadávanie výsledkov z krajiny Česká republika.

3. Používatelia mali opísať, ako by postupovali, keby chceli zobraziť podrobnejšie informácie o ľubovoľnom výsledku.

4. Používatelia dostali za úlohu výsledky zoradiť od najnovšie pridaných.

5. Používatelia mali z hlavnej výsledkovej tabuľky zistiť, aké konkrétne blacklisty sa viažu na ľubovoľný záznam v tabuľke.

6. Používatelia dostali pokyn k prechodu na stránku s detailom IP adresy. Tu mali opísať čo vidia a aké informácie sú schopný vyčítať.

7. Poslednou úlohou bolo stiahnutie záznamov vo formáte JSON pre zdroj Warden.

Výsledky testovania boli nasledujúce:

1. Všetci testovaní používatelia mali zo stránky pozitívny dojem. Pôsobila na nich usporiadanie. Niektorí konštatovali, že nie všetkému rozumejú, po vyzvaní, aby skúsili zistiť dodatočné informácie zistili, že prechodom nad komponentami stránky sa zobrazia dodatočné vysvetlivky. Používatelia si mysleli, že sa jedná o akýsi typ vyhľadávacej tabuľky, niektorí spoznali IP adresy a aj vedeli, čo znamenajú vlajky pri nich. Používatelia, ktorí si stále neboli istí, čo stránka ponúka našli v dolnej časti odkaz na sekciu About.

2. Pri vyhľadávaní podľa krajiny všetci používatelia hneď vedeli, kde sa nachádza vyhľadávací formulár. Našli v ňom aj, pre nich podstatnú, položku Country (niektorí špecifikovali, že by bolo vhodné mať aj lokálnu jazykovú variantu). Z ponúknutej možnosti krajín niektorí najskôr začali v liste položku hľadať posúvaním, potom si však všetci uvedomili rozsiahlosť a vyplnili vyhľadávacie pole.

3. Všetci používatelia rozoznali, že IP adresa v tabuľke slúži ako odkaz. Niektorí následne skúšali aj ikonu v ľavej časti záznamu.

4. Používatelia najskôr hľadali túto možnosť vo vyhľadávacom formulári alebo v hornej časti nad tabuľkou. Jeden subjekt konštatoval, že to systém nepodporuje. Ostatní výsledky úspešne od-filtrovali, pričom padla aj poznámka že je to podobné umiestenie ako v programe Excel.

5. Všetci používatelia identifikovali stĺpec s blacklistami. Niektorí na číslo počtu blacklistov klikli, iní len prešli myšou. V oboch prípadoch sa však požadovaný výsledok zobrazil.

6. Všetci používatelia sa správne dostali na stránku s detailami. Všimli si grafy a stránku aj vertikálne posúvali, aby zistili, čo všetko sa na nej nachádza. Stránku opisovali, že obsahuje tri rôzne štatistiky, že grafy majú rôzne typy a zobrazujú dáta v čase. Niektorí si spojili súvislosť, že tabuľkové dáta nad grafmi zobrazujú rovnaké dáta ako sú aj v grafoch.

7. Všetci používatelia vedeli, kde sa nachádza sekcia zo zdrojom Warden. Niektorí chvíľu váhali nad konkrétnou ikonou v pravej časti, ale všetkým sa podarilo otvoriť modálne okno a stiahnuť dáta v špecifikovanom formáte.

Z testovania hlavných stránok nového systému sme zistili, že implementovaný návrh je dobre použiteľný aj pre nových používateľov. Systém poskytuje nádpovedy, ktoré sú skryté, aby nerušili a nezaberali miesto v prípade častých používateľov, ale zároveň ponúkali vysvetlenie pre tých neznalých. Testovanie nám prinieslo lepší pohľad na to, ako skutoční používatelia so systémom interagujú. Takisto sme získali nové poznatky, napríklad o tom, že niektorým používateľom by vyhovovali lokálne jazyky stánky.

5.4 Testovanie zabezpečenia nového aplikačného programového rozhrania

Zabezpečenie REST API spočívalo vo vynútení validácie s využitím špecifikácie OpenAPI pre každý endpoint. Špecifikácia OpenAPI bola bližšie podstavná v podkapitole 4.2.2. Testovanie správnosti implementácie by nám malo preukázať správnosť vytvorenej špecifikácie. Spomínaná špecifikácia pritom musí byť dostatočne všeobecná, aby dovoľovala variáciu vstupov, ale na druhej strane musí byť aj dostatočne prísna, aby sa zamedzilo prijímaniu dát v nesprávnom formáte. Obmedzovanie prijímaných dát navyše zaručuje vyššiu bezpečnosť používania systému, kde útočník nemôže do endpointov odosielať ľubovoľné dáta.

Testovanie prebiehalo s využitím aplikácie Postman, kde boli vytvorené testovacie dopyty pre každý zo štrnástich endpointov (celkovo dve až päť testovaných variácií podľa komplexnosti endpointu). Následne boli otestované parametre pri GET a DELETE dopytoch a JSON objekty v tele ostatných dopytov. Snahou bolo skúšať nepovolené dátové typy alebo hodnoty. Napríklad odoslanie textovej hodnoty `TEST` parametru `tag_op` v JSON štruktúre, ktorý ako povolené textové hodnoty prijíma len `AND` a `OR`. Testovalo sa, či daná nepovolená zmena naozaj vyvolá výnimku.

Prípadné odhalené nedostatky, ktoré viedli na možnosti obídenia OpenAPI špecifikácie, boli odstránené upravením špecifikácií jednotlivých endpointov.

5.5 Možné budúce rozšírenia systému NERD

Moderné webové rozhranie spolu s novým systémom, ktoré tvoria výstupy tejto práce sú pripravené na budúce nasadenie na server `nerd.cesnet.cz`. Budúce rozšírenia môžu zahŕňať implicitnú podporu na odosielanie žiadostí od používateľov, ktorí majú záujem o dodatočné systémové oprávnenia (napríklad vidieť aj skrytý a pre bežných používateľov vynechaný obsah).

Taktiež môže byť pridaná väčšia miera filtrovania a rozlišovania zobrazovaných údajov pre jednotlivé role, aby odlišné skupiny používateľov mali prístup k odlišným dátam. Pridanie administrátorskej konzoly výrazne uľahčuje evidovanie a modifikáciu rolí jednotlivých používateľov.

Ďalšie z možných budúcich rozšírení rátajú s rozširovaním samotného systému NERD, ktorý zhromažďuje dáta o nebezpečných internetových entitách. Ak sa tento systém bude ďalej rozširovať, či už o dodatočné zdroje alebo štatistiky IP adries bude vďaka novému modulárnemu webovému rozhranie ľahšie zobrazovať nové výsledky zo systému.

V systéme je pripravená aj frontendová implementácia zobrazovania reputačného skóre, ktorá ráta s budúcim rozšírením pridania funkcií zhormažďujúcich a uchovávajúcich tieto informácie, teda štyri až šesť posledných hodnôt tohto skóre.

Kapitola 6

Záver

V priebehu riešenia diplomovej práce bol získaný všeobecný prehľad o systéme NERD, ako aj o moderných webových rozhraniach a frameworkoch, ktoré sa na ich tvorbu používajú. Bol zdokumentovaný a zhodnotený aktuálny stav webového rozhrania systému NERD, pričom dôraz sa kládol na identifikáciu potrebných a dobre fungujúcich súčastí, ktoré bolo treba aj v novom systéme zachovať, ale aj na nájdenie oblastí, v ktorých boli priestory na zlepšenie. V práci je tiež popísané, o aké konkrétne zlepšenia sa jedná.

V priebehu návrhu nového webového rozhrania bolo identifikované, aké rozsiahle zásahy budú do súčasného systému potrebné. Implementačná časť tejto diplomovej práce sa preto nezaoberá len novým návrhom webového rozhrania, ale aj modifikáciou a nadstavbou nad existujúcim systémom, ktorý bolo treba pozmeniť aby odpovedal požiadavkám pripojenia a komunikácie s novým webovým rozhraním.

Moderné webové rozhranie v rámci dizajnu získalo novú farebnú paletu, množstvo znovu-použiteľných prvkov s využitím externých ale aj vlastných komponentov, ktoré vytvárajú na popredí rozhrania pohodlné a na pozadí priamočiare riešenia. Web takisto získal responzivitu a podporu pre obrazovky všetkých veľkostí. Zároveň boli do webového rozhrania pridané nové funkcionality, ktoré zahŕňajú konzolu pre admina, grafy reputačných skóre, a aj zobrazovanie priehľadovej tabuľky prítomnosti na čiernych listinách.

V systéme sa podarilo vytvoriť úplne nové aplikačné programové rozhranie, ktoré poskytuje koncové body na výmenu informácii s moderným webovým rozhraním, nakoľko v pôvodnom systéme táto komunikácia prebiehala interne. Vytvorené koncové body obsluhujú všetky požiadavky webového rozhrania, pričom najväčšie množstvo koncových bodov bolo potrebné vytvoriť hlavne pre správu používateľov.

Správa používateľov okrem iného disponuje aj podporov externých poskytovateľov identít, ktorými sú konkrétne služby Google, Twitter, GitHub a federácie EduGAIN. Používatelia majú teda pohodlnú možnosť využiť pri prihlasovaní alebo registrácii externé služby, ktoré už používajú.

Výslednú implementáciu, ktorá je predstavená v tejto práci, sa úspešne podarilo nasadiť na testovací server o rovnakej špecifikácii, akú má aj hlavný server systému NERD, pre ktorý bol nový systém vyvíjaný. Môžeme preto konštatovať, že v tejto práci sa podarilo implementovať použiteľné moderné webové rozhranie systému NERD.

Literatúra

[1] Bartoš, V. *NERD: Network Entity Reputation Database*. Association for Computing Machinery, 2019 [cit. 2022-10-02]. Dostupné z: https://dl.acm.org/doi/10.1145/3339252.3340512.

[2] Bartoš, V. *Nerd Wiki - Architecture* [online]. 2019 [cit. 2022-12-11]. Dostupné z: https://github.com/CESNET/NERD/wiki/Architecture.

[3] Bartoš, V. *Nerd Wiki - Reputation score* [online]. 2022 [cit. 2022-12-11]. Dostupné z: https://github.com/CESNET/NERD/wiki/Reputation-score.

[4] Bilesanmi, A. *How to login with Twitter API using Python* [online]. 2023 [cit. 2023-04-20]. Dostupné z: https://medium.com/bilesanmiahmad/how-to-login-with-twitter-api-using-python-997333d436d2.

[5] CESNET Perun. *Attributes and scopes* [online]. 2020 [cit. 2023-04-25]. Dostupné z: https://aai.cesnet.cz/en/index/documentation/sp/proxy/attributes_and_scopes.

[6] Connect2id Ltd.. *OpenID Connect UserInfo endpoint* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://connect2id.com/products/server/docs/api/userinfo.

[7] eduGAIN. *What is eduGAIN* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://edugain.org/about-edugain/what-is-edugain/.

[8] ELITEX Team. *Front-end and JavaScript trends in 2022* [online]. 2022 [cit. 2022-12-13]. Dostupné z: https://elitex.systems/blog/front-end-javascript-development-trends-2022/.

[9] GitHub, Inc. Docs. *Authorizing OAuth Apps* [online]. 2023 [cit. 2023-04-21]. Dostupné z: https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps.

[10] GitHub, Inc. Docs. *List email addresses for the authenticated user* [online]. 2023 [cit. 2023-04-21]. Dostupné z: https://docs.github.com/en/rest/users/emails?apiVersion=2022-11-28#list-email-addresses-for-the-authenticated-user.

[11] Google Trends. *Porovnanie webových frameworkov v priebehu času* [online]. 2022 [cit. 2022-11-09]. Dostupné z: https://trends.google.com/trends/explore?cat=5&date=today%205-y&q=React,Angular,Vue,Backbone,Ember.

[12] Masaryk University Perun. *Overview* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://perun-aai.org/about-perun/overview.

[13] MongoDB, Inc.. *Why Use MongoDB and When to Use It?* [online]. 2022 [cit. 2022-12-13]. Dostupné z: https://www.mongodb.com/why-use-mongodb.

[14] Okoth, L. *How To Authorize a User Using the GitHub OAuth API , Python and Flask: Part Two.* [online]. 2023 [cit. 2023-04-21]. Dostupné z: https://medium.com/@lyle-okoth/github-oauth-using-python-and-flask-a385876540af.

[15] Okta, Inc.. *JSON Web Tokens* [online]. 2023 [cit. 2023-04-17]. Dostupné z: https://auth0.com/docs/secure/tokens/json-web-tokens.

[16] Pallets project. *Flask - Web development, one drop at a time* [online]. 2010 [cit. 2022-12-12]. Dostupné z: https://flask.palletsprojects.com/en/2.2.x/#.

[17] ParTech Media. *REST API Development* [online]. 2020 [cit. 2022-12-13]. Dostupné z: https://www.partech.nl/nl/publicaties/2020/07/9-trending-best-practices-for-rest-api-development#.

[18] Red Hat, Inc.. *What is a REST API?* [online]. 2020 [cit. 2022-12-13]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[19] Reitz, K. *OAuth 2 Workflow* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://requests-oauthlib.readthedocs.io/en/latest/oauth2_workflow.html.

[20] Shibboleth Consortium. *What is Shibboleth?* [online]. 2023 [cit. 2023-04-25]. Dostupné z: https://www.shibboleth.net/about-us/the-shibboleth-project/.

[21] Veeraraghavan, S. *20 Most Popular Programming Languages to Learn in 2023* [online]. 2022 [cit. 2022-12-16]. Dostupné z: https://www.simplilearn.com/best-programming-languages-start-learning-today-article.

[22] X Corp. Developer Platform. *User Access Tokens (3-legged OAuth flow)* [online]. 2023 [cit. 2023-04-20]. Dostupné z: https://developer.twitter.com/en/docs/authentication/oauth-1-0a/obtaining-user-access-tokens.