Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Analytický nástroj pro databázi rádiových fingerprintů

Diplomová práce

Autor: Bc. Kateřina Zemánková

Studijní obor: Aplikovaná Informatika

Vedoucí práce: Ing. Pavel Kříž Ph.D.

Hradec Králové Duben 2017

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne Podpis:

Poděkování

Děkuji panu doktoru Pavlovi Křížovi za trpělivost, cenné připomínky a odborné vedení během mé práce.

Anotace

Cílem této diplomové práce je analýza, návrh a zdokumentování implementace aplikace sloužící jako analytický nástroj pro databázi radiových fingerprintů (soubor signálů zachycených na určitém místě na mobilním zařízení) podle zadaných požadavků a zároveň přiblížit technologie použité pro vývoj a chod aplikace. V návrhové části je při popisování technologií kladen důraz na databázový systém Couchbase a Nette Framework. Velká část popisu implementace je věnována vysvětlení účelu tříd, funkcí a algoritmů tvořících aplikaci.

Annotation

Title: Radio Fingerprint Database Analytical Tool

The objectives of this diploma thesis is analysis, design and implementation of an application that could be used as an analytical tool for radio fingerprint (a group of signals caught on a mobile device) database while meeting obtained requirements and describing technologies, that were used to achieve them, with emphasis put on the Couchbase database system and Nette Framework. Large part is dedicated to explain the purpose of classes, functions and algorithms that form this application.

Obsah

1 Úvod

Fingerprinting je lokalizační metodou, při které se vytvoří databáze otisků signálu - hodnot RSSI (popř. dalších volitelných charakteristik) zachycených na měřícím zařízení v určité lokaci spolu se souřadnicemi místa, kde měření proběhlo. Samotná lokalizace je pak prováděna porovnáváním (např. pravděpodobnostními metodami, algoritmem k-nejbližších sousedů apod.) zachycených hodnot RSSI se sesbíranými daty v databázi. Tuto techniku lze využít v hustě zabydlených městských oblastech nebo uvnitř budov, kde není přímá viditelnost na oblohu a je tedy obtížné získávat signály z družic potřebných pro fungování satelitních navigačních systémů, jako jsou GPS, GLONASS nebo Galileo (1).

Tato práce se zabývá návrhem a implementací webové aplikace, která bude sloužit jako analytický nástroj pro již existující databázi fingerprintů.

Data byla získána v prostorách budovy fakulty informatiky a managementu (FIM) univerzity Hradec Králové. Uživatelé skenovali pomocí speciální aplikace ve smartfonu signály všech dostupných sítí a vysílačů, přičemž každé měření proběhlo na jednom fyzickém místě, trvalo 10 vteřin a jeho výsledky byly spolu s informacemi o použitém zařízení, přesným časem pořízení a souřadnicemi na mapě uloženy jako jeden fingerprint. Úložištěm použitým pro uchovávání fingerprintů je dokumentově orientovaná databáze Couchbase.

Záznamy obsahují mj. informace o naměřených Wi-Fi a Bluetooth signálech, síle těchto signálů, souřadnicích, kde měření proběhlo, označení podlaží v budově, na kterém měření proběhlo, název uživatele, který měření provedl, název zařízení, na kterém bylo měření provedeno a typ tohoto zařízení. Při práci s daty se bude pracovat především s těmito částmi záznamů.

Podle obdržených požadavků bude základní funkcionalitou nástroje vizuální zobrazení naměřených dat na mapě, filtrování těchto dat podle určitých parametrů, možnost přejmenování hodnot parametrů do uživatelsky přívětivějšího tvaru a zobrazení struktury jednotlivých záznamů v Couchbase databázi.

Práce je rozdělena podle postupu vývoje softwaru na tři hlavní části – analýza, návrh a implementace. Analytická část obsahuje specifikaci zadaných požadavků na aplikaci. Druhá část je věnována návrhům způsobu řešení jednotlivých požadavků, možných problémů a použitých technologií. Po ní následuje zdokumentování imple-

mentace softwaru zaměřené na detailní vysvětlení tříd, funkcí, konfigurace, způsobu získávání dat z databáze apod. Na konci je jedna kapitola věnována nasazení aplikace na server a testování.

2 Analýza

2.1 Požadavky na aplikaci

Při zadání práce byl obdržen seznam prvků, které je třeba implementovat, aby výsledná aplikace mohla sloužit jako analytický nástroj pro fingerprinty. Zároveň byl kladen důraz na to, aby byla aplikace navržena způsobem, který by umožnil snadnou implementaci možnosti zobrazení záznamů pořízených v případných dalších budovách.

2.1.1 Zobrazení fingerprintů

Hlavní funkcí aplikace bude umožnit uživateli prohlédnout si fingerprinty na mapě a třídit je podle žádoucích parametrů.

Všechna měření proběhla v prostorách budovy fakulty informatiky a managementu Univerzity Hradec Králové. Každý záznam tedy bude znázorněn do obrázku, podle souřadnic, které jsou součástí tohoto záznamu, ve formě bodu a tyto body budou vizuálně rozlišeny podle podlaží, na kterém byly signály zachyceny.

Byl vznesen požadavek, aby barva bodů na mapě v případě filtrování fingerprintů podle konkrétní zachycené MAC adresy (ať už Wi-Fi nebo Bluetooth) indikovala sílu tohoto zachyceného signálu. Každý záznam obsahuje seznam MAC adres zařízení, ze kterých byl obdržen signál (viz kapitola 2.2 *Způsob uložení dat*), a síla tohoto signálu. U některých měření došlo k zachycení stejného signálu vícekrát, přičemž u každého pokusu se mohly hodnoty intenzity lišit. V takových případech se tedy bude pracovat s hodnotou průměrnou. Zelená složka barvy určené pro vykreslení bodu bude s rostoucí silou signálu intenzivnější, zatímco červená složka méně intenzivní, a naopak.

Mapu bude možné přiblížit, oddálit nebo posunout a to bez znovunačtení stránky.

Po vykreslení bodů do mapy si uživatel bude moci zjistit detailnější informace týkající se měření - seznam zachycených MAC adres s naměřenými hodnotami signálů

název uživatele, který provedl měření, název a značka zařízení, na kterém měření proběhlo, podlaží, kde k měření došlo, a datum a čas pořízení záznamu. Tyto hodnoty pak půjde kliknutím myši přidat do aktuálního nastavení filtru (viz kapitola 2.1.2 *Filtrování dat*). Bude přidána i možnost zobrazit celý jeden záznam z databáze ve formě stromu.

2.1.2 Filtrování dat

Uživatel bude schopen filtrovat data, která mají být na mapě zobrazena, pomocí HTML formuláře. Parametry, podle kterých se budou vybírat záznamy k zobrazení, budou MAC adresa u Wi-Fi a Bluetooth skenů, uživatel (autor fingerprintu), identifikátor a značka zařízení, podlaží a čas vzniku.

Při filtrování dat podle MAC adresy zařízení, ze kterého byl přijat signál, bude po aktivaci filtru změněn způsob znázornění bodů podle síly signálu (vysvětleno v kapitole 2.1.1 *Zobrazení fingerprintů*).

Ostatní filtry žádné speciální vyobrazení nevyžadují, body budou rozlišeny pouze na základě poschodí.

Ovládací rozhraní bude umožňovat automatické předvyplnění textu (tzv. „našeptávač“), aby se docílilo větší uživatelské přívětivosti. U filtrů, u kterých se nepředpokládá velké množství různých dat v databázi, není automatického vyplňování potřeba, postačí input typu „select box“.

Zvláštní pozornost bude potřeba věnovat vyhledávání fingerprintů podle poschodí, kde došlo k měření. Při výběru jednoho patra ve formuláři se vyselektované body zobrazí do obrázku mapy určené pro toto konkrétní podlaží. Pokud tento filtr není aktivní, vykreslí se body měřené na různých patrech budovy do jedné mapy, a to do mapy přízemí. Hlavně v tomto případě se uplatní vizuální rozlišení fingerprintů podle poschodí (viz kapitola 2.1.1 *Zobrazení fingerprintů*).

Zároveň bude uživateli umožněno zadávat vlastní dotazy do Couchbase přes konzoli zobrazenou na stránce s mapou. Odesláním dotazu se zobrazí odpovídající fingerprinty.

2.1.3 Přejmenování záznamů

Názvy hodnot některých atributů nejsou uživatelsky přívětivé - například identifikátor zařízení je třináctimístné číslo. Je vhodné umožnit uživateli tyto hodnoty přejmenovat a nadále je zobrazovat pod symbolickým názvem. Konkrétně se bude jednat o atributy MAC adresa, jméno uživatele a identifikátor zařízení.

2.2 Způsob uložení dat

Fingerprinty jsou uloženy v databázovém systému Couchbase, kde jsou jednotlivé záznamy skladovány v podobě JSON dokumentů, přičemž každý dokument reprezentuje jeden fingerprint. Jedním z požadavků na aplikaci je, aby byla schopná pracovat právě s tímto druhem databáze. Mělo by tedy být možné v Couchbase procházet jednotlivé dokumenty a filtrovat výběr dokumentů podle konkrétních hodnot[1].

Každý záznam má svůj unikátní identifikátor složený z písmen a číslic a samotný obsah záznamu v poli „value“.

```
{
	"id":"0004f611-e08f-410f-84f3-f5a546060643",
	"key":"0004f611-e08f-410f-84f3-f5a546060643",
	"value":
	{
		"_sync":
		{
			"rev":"1-057a9d4fa3d94976a9e549cbeb8b7a8c",
			"sequence":681,
			"recent_sequences": [681],
			…
		}
	…
}
```

Ukázka kódu 1 - Struktura Couchbase dokumentu

[1] Obdržený výpis z databáze ve formě souboru obsahujícího JSON dokumenty jako jeden velký JSON objekt obsahoval nejprve 200 záznamů, v pozdější části vývoje pak 1026 záznamů

Dokumentace ke způsobu vytváření původní databáze a významu všech atributů nebyla v době vytváření aplikace k dispozici, ale podle způsobu pojmenování lze význam většiny informací odhadnout („level“, „bleScans“, „address“, „rssi“ apod.).

V nástroji pro analýzu fingerprintů se bude využívat jen zlomek uložených informací. Níže je znázorněno, jak by vypadala struktura JSON dokumentu pouze s atributy relevantními pro aplikaci.

```
{
	"id":" 2a105d31-d7b7-4e71-8f22-c399f95f7b35",
	"value":
	{
		"brand": "Sony",
		"couchbase_sync_gateway_id": " google-110350053603587200109",
		"deviceId": " 357652066748937",
		"level": "J2NP",
		"wifiScans": [
			{
				"mac": " 00:25:45:24:7e:b1“,
				"rssi": -79
			},
			{
				"mac": "00:25:45:24:7e:bf",
				"rssi": -81
			},
			…
		],
		"bleScans": [
			{
				"address": " E6:02:16:CA:D2:67“,
				"rssi": -94
			},
			…
		],
		"createdAt": " 2015-11-13 13:28:24",
		"x": "1978",
		 "y": "502"
	}
}
```

Ukázka kódu 2 - Relevantní atributy dokumentu

Pro účel aplikace jsou důležité informace o zařízení a uživateli:

- *brand* – značka zařízení
- *deviceID* – unikátní identifikátor zařízení
- *couchbase_sync_gateway_id* – identifikátor uživatele, který provedl měření

a informace o samotném měření a jeho výsledcích:

- *level* – označení podlaží, kde k měření došlo
- *wifiScans* – pole naměřených hodnot, obsahující MAC adresu[2] zachyceného Wi-Fi zařízení a sílu tohoto signálu (hodnotu RSSI)
- *bleScans* – obdobně jako u "wifiScans", pouze se jedná o zařízení Bluetooth
- *createdAt* – čas pořízení fingerprintu ve formátu "YYYY-MM-DD hh-mm-ss"
- *x* a *y* [3]– souřadnice na mapě podlaží (v pixelech)

Každý ze záznamů má (kromě výše vypsaných) dalších 36 atributů jen na první úrovni JSON pole. Vnořená pole, jako je například *„wifiScans"*, mohou obsahovat i desítky záznamů.

3 Návrh

Vzhledem k zadaným požadavkům bylo rozhodnuto, že nástroj pro práci s databází fingerprintů bude implementován jako webová aplikace za použití PHP frameworku *Nette* založeném na softwarové architektuře MVC (Model-View-Controller). Mezi jeho výhody patří například znovupoužitelnost kódu nebo tzv. „šablonovací systém" Latte (viz kapitola 3.2.3 *Nette Framework*).

Záznamy o fingerprintech se budou vyhledávat v již existující Couchbase databázi a uživatel nebude mít možnost tyto informace skrze aplikaci měnit - pro funkci vytváření aliasů bude vytvořena nová databáze.

3.1 Problémy k řešení

V následujících kapitolách jsou rozebrány návrhy řešení jednotlivých funkcionalit vyplývajících z požadavků.

---

[2] Stejná MAC adresa se může vyskytovat v poli wifiScans více než jednou, protože měření probíhalo po dobu 10 vteřin a signál mohl být zachycen víckrát

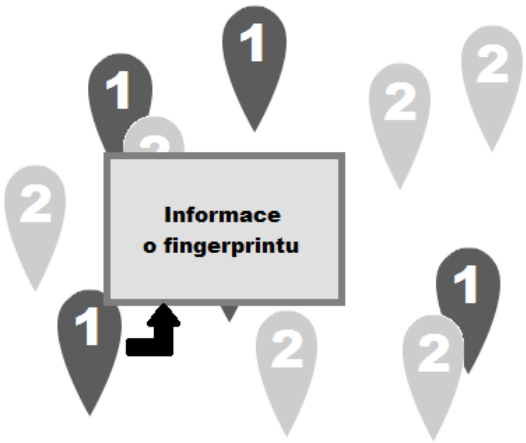
[3] Hodnoty souřadnic *x* a *y* jsou v databázi uloženy v datovém typu string (viz kapitola 3.1.3 *Problém posunutých souřadnic v jednotlivých patrech*)

3.1.1 Zobrazení dat na mapě

Hlavním úkolem nástroje je vizuálně zobrazit jednotlivé fingerprinty jako body na mapě s detailnějšími informacemi o naměřených hodnotách, uživateli, který pokus provedl, a o zařízení, na kterém proběhlo měření. Selekce bodů podle parametrů bude probíhat pomocí HTML formuláře. Mapu by mělo jít přibližovat, oddalovat a při přiblížení posouvat do všech stran. Při rozhodování, jakým způsobem tohoto cíle dosáhnout, byly vyzkoušeny dvě možnosti a to za 1) použití javascriptového modulu s předdefinovanými funkcemi na zaznamenání bodů do mapy a za 2) vytvoření vlastních funkcí v GD grafické knihovně.

Ze zdarma dostupných jQuery rozšíření zabývajících se řešením zakreslováním bodů do mapy byl vybrán plugin s názvem *imgNotes* (2), který implementuje přibližování a oddalování obrázku při scrollování na myši, vykresluje do mapy značky podle zadaných souřadnic (naznačeno na obrázku 1), umožňuje změnit jejich barvu i popisek a při kliknutí na jednu konkrétní vytvoří okno, kam lze vypsat libovolný text. Procházení pole záznamů i vykreslování probíhá v jQuery, tedy na straně klienta.

Velkou výhodou je to, že značky („markery“) zakreslené do mapy si zachovávají svou velikost při přibližování i oddalování obrázku, což zpřehledňuje situaci, kdy se na jednom místě shlukuje více fingerprintů. Navíc se dají detaily o jednotlivých záznamech navázat přímo na objekt (značku), který záznam reprezentuje.



Obrázek 1 - Způsob zobrazení fingerprintů imgNotes

Tento způsob zobrazení fingerprintů byl implementován při první fázi vývoje, kdy byl počet záznamů v databázi menší (dvě stě). Při navýšení počtu dokumentů na tisíc však trvalo přidávání značek do mapy příliš dlouho (řádově desítky vteřin), proto bylo potřeba najít jiné řešení.

Druhá uvažovaná alternativa byla provést vykreslování do mapy na straně serveru pomocí GD grafické knihovny a detaily o fingerprintu získávat vyhledáním konkrétních záznamů v databázi podle nejbližších souřadnic od místa kliknutí do obrázku přes volání ajaxové funkce.

Do prázdné mapy se přes funkce grafické knihovny dokreslí body (viz obrázek 2) a vznikne nový obrázek. Kvůli zvýšení rychlosti načítání se nebude pokaždé při příchodu na stránku vytvářet nový obrázek, ale pokud to bude možné, použije se již existující.

Obrázky s body se budou ukládat do samostatné složky pod názvem souboru odvozeným z textu dotazu a zároveň se uloží samotný text tohoto dotazu jako nový textový soubor. Před zobrazením mapy s fingerprinty se zkontroluje tvar dotazu, který má získat pole vyselektovaných záznamů, a proběhne pokus o vyhledání již existujícího textového souboru s odpovídajícím dotazem a souvisejícího obrázku. Pokud se najde shoda, pak se použije již vytvořený soubor, což by mělo proces zobrazení fingerprintů zrychlit. Během příchodu na stránku se ověří stáří souborů ve složce s obrázky a texty dotazů, přičemž každý soubor přesahující určený limit bude smazán. Tato složka s nově vytvořenými soubory bude umístěna do složky s obrázky nazvané „images“, která je součástí struktury Nette frameworku.

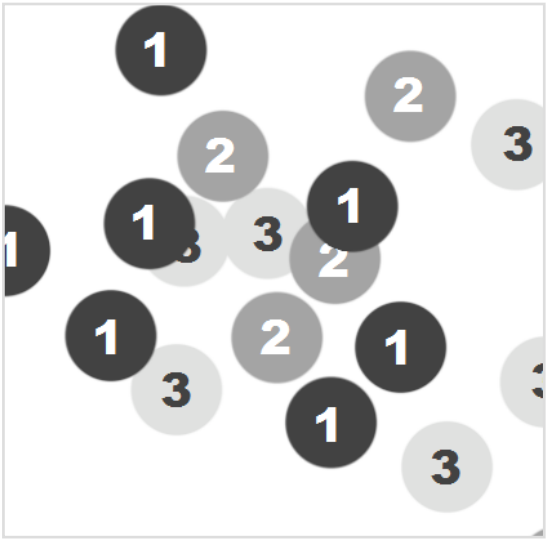
Zároveň bude přidána ochrana před zaplněním disku - konkrétně se bude jednat o nastavení limitu maximální velikosti složky a přidání funkce kontroly velikosti všech obsažených souborů, kdy při přesahu hodnoty limitu bude určité množství nejstarších vygenerovaných souborů vymazáno. Funkce se bude volat před každým pokusem o vytvořený nového obrázku a dokumentu s textem dotazu.

Jde o rychlejší způsob vyobrazení než při použití pluginu *imgNotes*. Na druhou stranu nakreslení bodů přímo do obrázku má následující nevýhody:

1. zobrazení detailů o fingerprintech bude náročnější na implementaci
2. při přibližování mapy se budou nakreslené body zvětšovat, protože budou součástí obrázku

I přes nedostatky bylo toto řešení vyhodnoceno jako vhodnější alternativa.

Je vyžadováno, aby byly body z různých pater barevně odlišeny, což lze pomocí GD nadefinovat snadno přidáním podmínek, za předpokladu, že číslo patra bude součástí pole fingerprintu získaného z databáze. Barvy jednotlivých pater budou definovány v MySQL tabulce spolu s číslem patra a jeho označení v Couchbase.



Obrázek 2 - Způsob zobrazení fingerprintů GD

Dále by se při vykreslení fingerprintů obsahujících jednu konkrétní MAC adresu (filtrování podle MAC adresy) měla zobrazit síla přijatého signálu (viz kapitola 2.1.1 *Zobrazení fingerprintů*). Při procházení pole získaných záznamů se bude kontrolovat atribut s klíčem „rssi“ (*Received Signal Strength Indicator*) u každého zachycení signálu filtrované MAC adresy a spočte se průměr této hodnoty. Po prozkoumání uložených dokumentů dosahovala nejvyšší hodnota signálu (předpokládá se v dBm) -14 a nejnižší -99. Škála pro přepočítání hodnoty na procenta byla zavedena podle (3), tedy jako minimální síla signálu RSSI je stanovena hodnota -13 (0%) a maximální -10 (100%)[4]. Následně se při obarvování bodu vypočítá intenzita červené a zelené složky odstínu barvy.

Pro umožnění přibližování a oddalování mapy bude přidán jQuery plugin Wheelzoom (4) (více v kapitole 3.2.5.1 *Wheelzoom*).

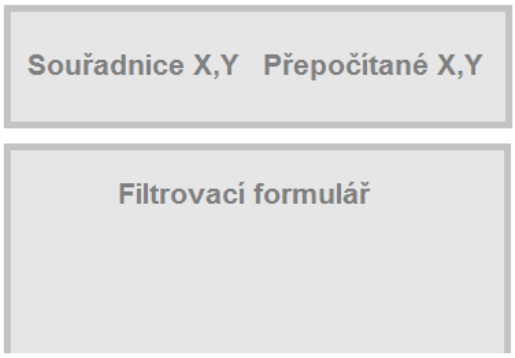
[4] S hodnotami vyššími (nebo naopak nižšími) se bude počítat jako s hodnotami rovnými maximální (nebo minimální) stanovené hodnotě

3.1.2 Detailní informace o fingerprintech

Bude potřeba zajistit, aby si uživatel mohl prohlédnout detaily jednotlivých měření, konkrétně:

- souřadnice fingerprintu
- vzdálenost fingerprintu od místa kliknutí do mapy
- čas vytvoření fingerprintu
- číslo poschodí, kde došlo k měření, jméno uživatele (autora), ID zařízení a značky zařízení – tyto informace budou automaticky skryty a zobrazeny přes kliknutí na tlačítko
- seznam všech MAC adres (Bluetooth a Wi-Fi zařízení) zachycených během měření včetně průměrné síly signálu

Jak již bylo zmíněno v předchozí kapitole, detaily o fingerprintech zjistí uživatel přes kliknutí do mapy. Pro snadnější orientaci se budou při přejíždění myší přes obrázek ukazovat aktuální souřadnice kurzoru v pravém horním rohu stránky (nad filtrovacím formulářem). Kvůli častému používání přepočítaných souřadnic (viz kapitola 3.1.3 *Problém posunutých souřadnic v jednotlivých patrech*) bude uživateli umožněno vidět kromě souřadnic pro aktuální mapu i souřadnice pro mapu přízemí (viz obrázek 3), podle které byly hodnoty *recalculatedX* a *recalculatedY* vytvořeny, v případech, kdy budova, ze které pocházejí fingerprinty, vyžaduje přepočítávání souřadnic (mapy na sebe nesedí) a zobrazuje se mapa jiná než ta prvního patra.



Obrázek 3 – Zobrazení aktuálních souřadnic

Po kliknutí se zjistí přesné souřadnice na původním obrázku mapy (viz kapitola 3.1.4 *Zjištění aktuálních souřadnic na mapě*) a odešle se ajaxový požadavek na server. Aby se zajistilo, že nalezené nejbližší body budou podmnožinou těch aktuálně zobrazených, odešle se kromě souřadnic i text dotazu použitého při jejich vyselektování a

podle něj se sestaví nový dotaz. Pro nalezení nejbližších bodů bude využita euklidovská vzdálenost (vypočtená ve 2D ploše Pythagorovou větou).

Data vrácená z Couchbase databáze se esteticky zpracují v jQuery. Méně důležité informace budou umístěny v rozbalovacích boxech, aby byl výstup přehlednější. MAC adresy se signály seřazenými sestupně podle síly signálu se vypíší do tabulek. Také se vyhledají případné symbolické názvy, které se pak zamění za ty originální. K hodnotám, které lze filtrovat přes formulář, se přidají odkazy, které nastaví odpovídající input filtrovacího formuláře na tuto konkrétní hodnotu. Každý záznam s informacemi bude obsahovat i ID dokumentu v Couchbase, na které bude navázán odkaz vedoucí na stránku zobrazující celý JSON dokument (viz kapitola 3.1.4. *Zobrazení struktury dokumentu*).

To vše bude obsaženo ve výsledném HTML kódu, který se vloží do HTML elementu *div* v pravé části stránky soužícího jako informační panel („Information Panel“). Odezva potrvá pár vteřin. Informační panel se nejdříve vyprázdní (odstraní se případné předchozí záznamy) a zobrazí se v něm „točící se kolečko“ („spinner“) indikující, že se čeká na odpověď ze serveru. Nakonec kolečko zmizí a nahradí ho již nastylovaný text a tabulky s výsledky měření.

3.1.3 Problém posunutých souřadnic v jednotlivých patrech

Po detailnějším prozkoumání obdržených map podlaží (*J1NP.png*, *J2NP.png*, *J3NP.png*, *J4NP.png*) bylo zjištěno, že jsou obrázky uloženy v rozdílném rozlišení a navíc jsou různě pixelově posunuté. Souřadnice měření jsou v databázi uloženy podle souřadnic na jedné z map pater (pixelové souřadnice obrázku) – např. pokud proběhlo měření v prvním patře, v záznamu má atribut *level* hodnotu „J1NP“ a atributy *x* a *y* hodnoty pozice na mapě prvního podlaží (soubor *J1NP.png*) v pixelech. To představuje problém při řešení jednoho z požadavků na aplikaci - vykreslování bodů skrze všechna patra najednou (viz kapitola 2.1.2 *Filtrování dat*) a s tím souvisejícím hledáním nejbližších bodů kliknutím do mapy.

Podle měření provedeného v grafickém editoru byly sesbírány hodnoty a podle nich navrženy úpravy souřadnic podle pater (viz tabulka 1).

Tabulka 1- Posunutí souřadnic

	Šířka	Výška	Měřítko X	Měřítko Y	Posunutí X	Posunutí Y
J1NP	2000	2500	1	1	0	0
J2NP	2700	3200	0,8	0,758	-21	-66
J3NP	2206	3000	0,907	0,84	-32	65
J4NP	2500	3400	0,8	0,742	-43	110

Sloupce „Šířka“ a „Výška“ zaznamenávají šířku a výšku obrázků v pixelech. „Měřítko X“ je chápáno u prvního patra jako 100%, u dalších pater je zaznamenána hodnota (zaokrouhlená na 3 desetinná místa), kterou se musí šířka obrázku vynásobit, aby výsledná vzdálenost dvou zvolených bodů na mapě byla stejná jako u obrázku *J1NP* (obdobně pro výšku). „Posunutí X“ a „Posunutí Y“ je měřeno jako rozdíl vybraného bodu na mapě druhého, třetího a čtvrtého podlaží od stejného bodu na mapě podlaží prvního (až po změně měřítka).

V MySQL tabulce s uloženými názvy pater a jejich barev budou zaznamenány údaje o posunutí i měřítku (použijí se při zjišťování souřadnic na mapě – viz kapitola níže) a v konfiguračním souboru bude vytvořeno pole obsahující seznam budov, které vyžadují přepočítávání souřadnic.

Aby správně fungovaly dotazy do Couchbase pro vyhledání nejbližších bodů, bude potřeba u každého dokumentu vytvořit další dva atributy „recalculatedX“ a „recalculatedY“ (typu *integer*), jejichž hodnotami budou přepočítané souřadnice x a y (ty jsou již uloženy v Couchbase jako atributy „x“ a „y“). Například hodnota souřadnice x se nejprve vynásobí číslem z tabulky – „Měřítko x“ a poté se od ní odečte nebo se k ní přičte číslo ze sloupce „Posunutí X“. Tyto atributy také zjednoduší vykreslování bodů ze všech pater do jedné mapy, protože odpadne nutnost přepočítávání hodnot u každého fingerprintu zvlášť.

Dalším problémem, na který se narazilo při pokusu o vyhledávání fingerprintů podle souřadnic, bylo to, že v Couchbase jsou souřadnice x a y uloženy jako textové řetězce. Aby bylo možné získat nejbližší fingerprinty od určitého místa za použití původních (nepřepočítaných) souřadnic, bude potřeba vytvořit u každého záznamu atributy ve formátu integer, jinak by se hodnoty při podmínce *ORDER BY* řadily podle abecedy, nikoliv podle číselné hodnoty. Nové atributy budou nazvány „*intX“* a „*intY“*.

Skript[5] s algoritmy použitými pro výpočet a nahrání nových dat do Couchbase byl přidán do složky „www“(„changeDataScript“) v adresářové struktuře frameworku.

3.1.4 Nalezení souřadnic na původním obrázku

Pro správné fungování akce vyhledání nejbližších bodů od místa klinutí do mapy bude nejdříve potřeba zjistit souřadnice tohoto kliknutí na původním obrázku uloženém na serveru, který může být zmenšený/zvětšený podle rozlišení obrazovky nebo přiblížený a/nebo posunutý použitím rozšíření *Wheelzoom*. Všechny tyto možnosti musejí být ošetřeny. Souřadnice kliknutí se upraví pomocí jQuery funkcí navázaných na element obrázku a podle přijatých informací o mapě (rozměry, posunutí, měřítko, nutnost použití souřadnic „recalculated“ nebo originálních).

3.1.5 Filtrování fingerprintů

Fingerprinty bude možno třídit podle následujících parametrů:

1. Wi-Fi MAC adresa
2. Bluetooth MAC adresa
3. jméno uživatele
4. ID zařízení
5. značka zařízení
6. podlaží budovy
7. datum a čas vytvoření fingerprintu

Akci filtrování bude moci uživatel provést dvěma způsoby. Jednak přes formulář (viz obrázek 4), který se bude zobrazovat v pravé části stránky, a přes „konzoli“ umístěnou v dolní části (viz obrázek 5).

Formulář

Textové inputy pro Wi-Fi adresu, Bluetooth adresu, jméno uživatele a ID zařízení budou používat autocomplete, který bude nabízet seznam všech hodnot daného typu a zároveň všechny vytvořené aliasy (viz kapitola 3.1.6 *Přejmenování záznamů*).

Další dva textové inputy budou sloužit pro filtrování podle data vzniku fingerprintu – minimální a maximální hodnota. Datum je v Couchbase uloženo jako string ve

[5] Stávající změny jsou určené pro fingerprinty v budově J

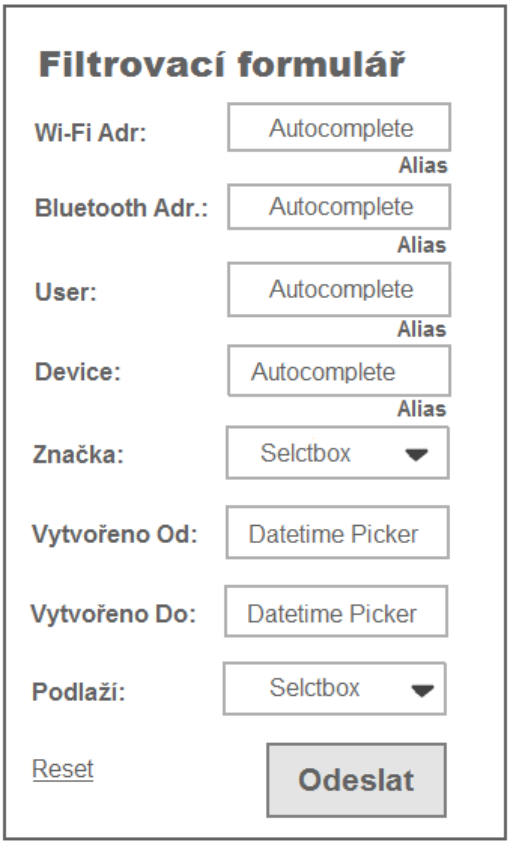
formátu *YYYY-MM-DD hh-mm-ss*. K inputu bude navázán doplněk „Datetime-picker“, který umožní vybrat hodnoty kliknutím do kalendáře a výsledek přeformátuje do tvaru potřebného pro dotaz.

Výběr značky zařízení bude dostupný přes *selectbox* obsahující všechny hodnoty *brand* v Couchbase.

Posledním filtrovacím parametrem je číslo patra. Kvůli podpoře více budov se bude seznam dostupných podlaží v budově přidávat dynamicky. Výběr budovy bude proveden příchodem na stránku - při prvním příchodu se vybere výchozí budova definovaná v konfiguračním souboru a přechod na jinou se provede přes položku v hlavním menu.

Výchozí nastavení filtru vrací všechny fingerprinty naměřené ve vybrané budově (např. všechna patra budovy „J“ – 1., 2., 3. a 4.). Po odeslání a znovunačtení stránky zůstanou vstupy vyplněné a pod nimi se zobrazí případné existující symbolické názvy.

Na konci formuláři bude odkaz, který resetuje filtr.



Obrázek 4 – Filtrovací formulář

N1QL konzole

Na stránce s mapou fingerprintů bude umístěna „konzole“ (nastylovaný HTML formulář), přes který bude uživateli umožněno zadávání vlastních dotazů[6] do Couchbase (viz obrázek 5).

Obrázek 5 - N1QL konzole

Jak bylo zmíněno dříve, při prvním příchodu na stránku se zobrazí mapa s fingerprinty, které už byly nějakým způsobem vyfiltrovány a přidány do mapy. Text dotazu, který byl pro získání dat použit, bude vypsán do N1QL konzole. Zároveň se tento text změní pokaždé, když dojde ke změně nastavení filtru. Do textu bude uživatel moci libovolně zasahovat a pak vytvořený dotaz odeslat přes klávesu *Enter* nebo tlačítko napravo od konzole.

Protože správné vykreslení bodů vyžaduje vyhledání konkrétních polí dokumentů, je třeba výsledek dotazu od uživatele zkontrolovat, zda tyto požadavky splňuje. Pokud by důležitá data chyběla, bude upozorněn na absenci konkrétního pole a bude mu navržena změna v textu dotazu. Musí být ošetřeno i vyhledání správných souřadnic – přepočítané vs. nepřepočítané.

Kvůli minimalizaci rizika zásahů do Couchbase (nástroj má sloužit pouze pro čtení dokumentů, nikoliv úpravu) bude přidána kontrola textu získaného od uživatele ještě před odesláním dotazu do databáze. V textovém řetězci se vyhledají slova shodná s názvy N1QL příkazů pro smazání dokumentu, úpravu, vytvoření a smazání indexů… apod. (DELETE, UPDATE, CREATE, DROP, SET a UNSET). Pokud se najde shoda, k provedení dotazu nedojde a uživatel bude na zavedené omezení upozorněn.

Předpokládá se, že by tato metoda při používání současného způsobu pojmenovávání atributů a hodnot v databázi fingerprintů neměla zapříčinit problémy ve vyhledávání informací o fingerprintech. Pokud by se názvy hodnot nebo atributů změni-

[6] Dotazovací jazyk, který je používán pro práci s daty v Couchbase, se nazývá N1QL

ly (např. *couchbase_sync_gateway_id* by bylo přejmenováno na „user**Set**“) bylo by potřeba najít řešení vhodnější.

Dalším problémem, který vzniká použitím konzole místo formuláře je obtížná kontrola dotazu kvůli zjištění filtrovaných hodnot a následné volbě způsobu zobrazení dat:

1. Vybrané jedno patro vs. více pater (změna mapy podlaží)
2. Filtrování jedné Wi-Fi adresy, Bluetooth adresy nebo více najednou (změna obarvení bodů)

Ad 1)

Výběr patra bude „odhadnut“ podle záznamů vrácených z Couchbase. Všechny tyto záznamy se projdou FOEACH cyklem a bude se kontrolovat, zda všechny obsahují stejnou hodnotu v atributu *level* či nikoliv. Pokud ano, pak se vybere mapa, do které se fingerprinty vykreslí, podle této hodnoty. Může se tedy stát, že původní dotaz do databáze neobsahoval podmínku pro výběr pouze jednoho patra, ale obdržené záznamy tomu nasvědčovaly, proto se použila mapa odpovídajícího podlaží.

S tím souvisí problém při „cashování obrázků“ a hledání nejbližších bodů k souřadnicím kliknutí. Zatímco u formuláře *filterData* jsou k dispozici proměnné s uloženými hodnotami filtru, včetně čísla podlaží, po odeslání dotazu z konzole a následného zobrazení staršího obrázku nelze zjistit, která mapa se použila, a tudíž nelze ani zajistit informaci o nutnosti použití přepočítaných souřadnic, která je nezbytná při hledání nejbližších bodů nebo zobrazování dvojích souřadnic při přejíždění kurzorem myši přes mapu.

Z toho důvodu bude ve fázi vytváření nového obrázku a textového souboru s dotazem do tohoto textového souboru uložen i text s názvem podlaží a v případě použití starého obrázku po odeslání dotazu z konzole umožní přístup k této informaci, podle níž už bude zajištění správných souřadnic snadné.

Ad 2)

Zpětné zjišťování filtrované MAC adresy z vrácených výsledků by bylo složité, protože získané pole fingerprintů může obsahovat Bluetooth i Wi-Fi adresy, které mohou být shodné u všech záznamů jen náhodou, nebo jich může být shodných vícero i při filtrování pouze podle jedné z nich.

Aby tedy bylo umožněno zobrazovat sílu signálu jedné MAC adresy, bude kromě vstupu pro dotaz přidán ještě vstup pro název MAC adresy a její typ - Wi-Fi nebo Bluetooth (viz obrázek 6).

Obrázek 6 - N1QL konzole (síla signálu)

K obarvení bodů podle signálu bude potřeba splnit následující podmínky:

1. Zaškrtnout checkbox nad konzolí (touto akcí se zobrazí skrytá část formuláře)
2. Vyplnit název MAC adresy (originální nebo symbolický)
3. Zvolit typ adresy
4. Zahrnout pole se signály do části SELECT v dotazu zobrazeném v konzoli („SELECT wifiScans, bleScans“)

Pokud podmínka v dotazu nezahrnuje vybranou adresu, zobrazí se i fingerprinty, které tuto adresu neobsahují. Takovéto body se obarví šedou barvou. Automatické vyplnění obou hodnot (adresy i typu) bude možné přes kliknutí na odkaz s adresou v informačním panelu.

Vyplnění MAC adresy do formuláře nijak neovlivní text dotazu, takže stejný text dotazu může vygenerovat různé obrázky (rozdíl v obarvení bodů). tím ohou nastat dvě situace, které bude potřeba ošetřit:

1. Uživatel zvolil možnost obarvení bodů podle určité MAC adresy, ale zobrazil se „cachovaný“ obrázek s jiným obarvením (podle poschodí nebo podle jiné MAC adresy).
2. Zobrazuje se cachovaný obrázek s body obarvenými podle signálu MAC adresy, přestože si to uživatel nevyžádal (body se měly obarvit podle poschodí).

Do souboru s textem dotazu a názvu podlaží se navíc přidají další dvě informace: zda byla pro filtrování použita konzole nebo formulář a název filtrované MAC adresy. Obrázek vytvořený při filtrování formulářem se nikdy nezobrazí při filtrování přes konzoli a naopak. Navíc při používání konzole se v uloženém textovém dokumentu bude kromě textu dotazu kontrolovat i název MAC adresy. Pokud nebude souhlasit, vytvoří se nový obrázek[7].

Zamezení vykreslení bodů nesouvisející s aktuálně zvolenou budovou a tudíž nepatřící do mapy této budovy (dotaz může obsahovat např. podmínku pro vyhledání úplně všech fingerprintů ze všech budov) proběhne ve vykreslovací funkci, kdy se bude vybírat barva bodu podle hodnoty v poli *level* obsažené ve vráceném dokumentu. Zkontroluje se, jestli název patra souvisí s aktuální budovou, a když ne, bod se nevykreslí.

3.1.6 Přejmenovávání záznamů

Vzhledem k nepřehledným tvarům názvů MAC adres, uživatelů a ID zařízení je vhodné umožnit uživateli tyto názvy přejmenovat a zobrazovat místo originálních názvů aliasy.

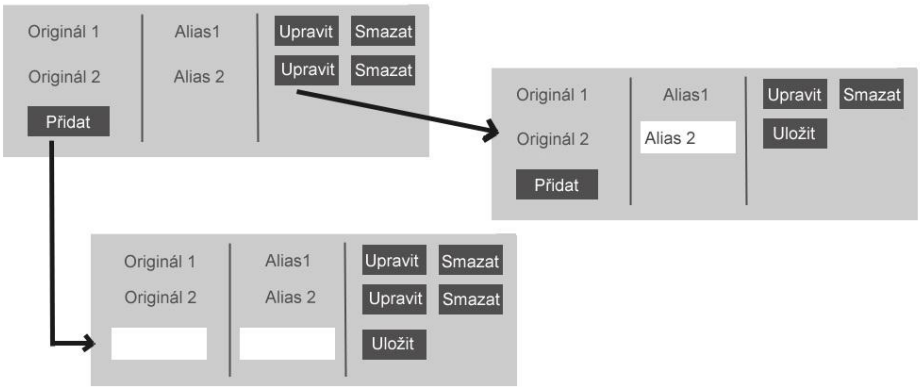
Všechny symbolické názvy vytvořené uživatelem se budou ukládat do jedné tabulky v MySQL databázi, kde se bude rozlišovat, o který typ aliasu jde, podle hodnoty ve sloupci „type“. Tabulka bude mít celkem tři sloupce: „original“ pro uložení původního názvu v Couchbase, „alias“ pro nový symbolický název a „type“ pro označení typu aliasu (Wi-Fi nebo Bluetooth adresa, název uživatele a ID zařízení).

Kvůli přehlednosti pro uživatele budou všechny operace nad symbolickými názvy (vytváření, úprava i mazání) probíhat na jedné stránce, rozdělené do čtyř tabulek podle typu aliasu (viz obrázek 7), pomocí jQuery a ajaxových funkcí.

[7] Obrázky i textové soubory se stejným názvem (odvozený z textu dotazu) se navzájem přepisují

Obrázek 7 - Tabulky se symbolickými názvy

Vedle každého řádku budou umístěna tlačítka pro editaci a smazání aliasu a pod každou tabulkou bude umístěno další tlačítko pro vytvoření nového aliasu (viz obrázek 8). Editace i vytváření nových aliasů bude umožněna přidáním nových HTML elementů typu *input*.



Obrázek 8 – Úprava symbolických názvů

Pokud se budou na stránce vyskytovat jakékoliv neuložené změny a uživatel se pokusí opustit stránku, bude nejprve o existenci změn upozorněn a teprve po potvrzení mu bude umožněno stránku opustit.

Pro líbivější vzhled se ke stylování tabulek využije framework Bootstrap a místo tlačítek (HTML elementů typu „button“) se vloží ikona z toolkitu „Font-awesome“ s odkazem navázaným na odpovídající jQuery funkci.

3.1.7 Zobrazení struktury dokumentu

Uživatel bude mít možnost prohlédnout si strukturu a hodnoty všech atributů v JSON dokumentu uloženém v Couchbase na stránce *Document.*

Jednotlivé dokumenty jsou uloženy jako pole dvojic „klíč – hodnota“, kde může být jako hodnota obsaženo další pole obsahující další pole… atd. Aby bylo možno celý objekt jednoduše a přehledně zobrazit, využije se jQuery rozšíření *JSON.VIEW* [4], který transformuje přijatý JSON objekt do přehledné formy stromu.

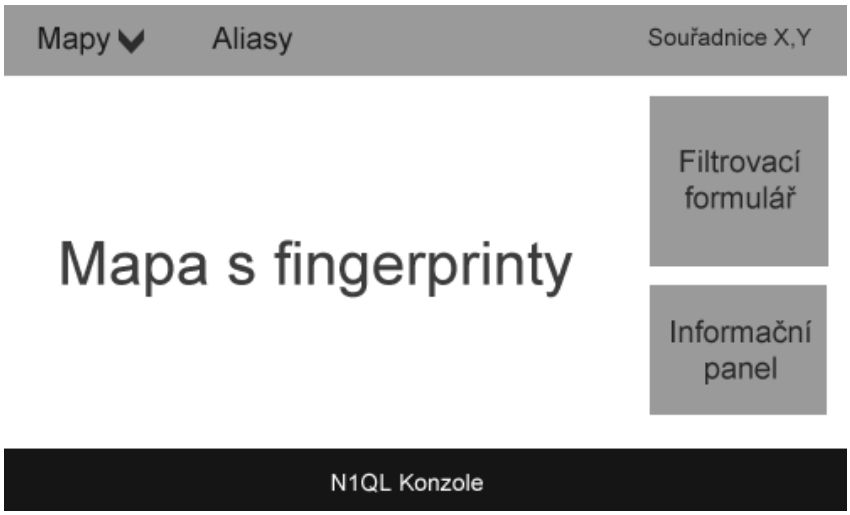
Podle stanovených požadavků se k hodnotám dokumentu, které lze filtrovat na hlavní stránce, přidají odkazy vedoucí právě na hlavní stránku s filtrem nastaveným na tuto konkrétní hodnotu. Půjde o atributy:

- *mac*
- *address*
- *createdAt*
- *couchbase_sync_gateway_id*
- *deviceId*
- *brand*
- *level*

Navíc budou k hodnotám přidány i uživatelem vytvořené symbolické názvy, pokud budou existovat v MySQL tabulce.

3.1.8 Navigační menu

Pro jednoduché přecházení mezi stránkami bude vždy v horním okraji umístěn navigační panel (viz obrázek 9) obsahující položky „Map“ a „Aliases“.



Obrázek 9 – Rozvržení stránky včetně navigace

První, sloužící pro přechod na stránku s vyobrazenými body na mapě a filtrovacím formulářem, bude obsahovat rozbalovací nabídku s výběrem budov. Zde zvolená budova bude sloužit jako výchozí filtr pro filtrování fingerprintů. Další je odkaz vedoucí přímo na stránku určenou pro práci s aliasy (vytváření, mazání, editace).

Odkaz na stránku zobrazující JSON dokument bude do tohoto menu umístěn jen na již otevřené stránce *Document*, protože ke správnému načtení stránky bude potřeba znát ID dokumentu, který se má zobrazit. Uživatel se sem dostane kliknutím na odkaz „ID“ v informačním panelu s detailními informacemi o fingerprintu nebo zadáním URL ručně v prohlížeči.

3.2 Použité technologie a rozšíření

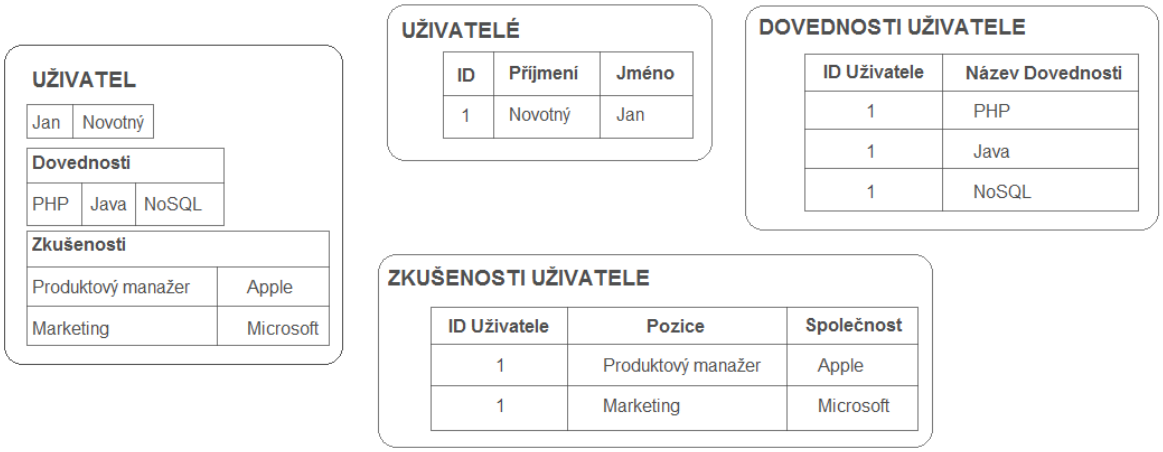
3.2.1 Couchbase

Couchbase je databázový systém, který pro ukládání a práci s daty využívá tzv. „NoSQL“ koncept. Na rozdíl od SQL databází, tedy nepoužívá tabulky a relační model. Místo toho jsou primárními entitami uloženými v databázi „dokumenty“, které mohou obsahovat strukturovaná, semistrukturovaná nebo nestrukturovaná data (není požadována normalizace). Výhodou tohoto způsobu ukládání dat je vysoká úroveň škálovatelnosti a flexibility v datové struktuře. Navíc má Couchbase server vysoký výkon díky používání asynchronních operací a souběžnosti (5).

Server může být nakonfigurován jako nezávislý nebo jako „cluster“, který spojuje několik Couchbase serverů dohromady a tím vytváří jeden distribuovaný zdroj dat, kde jsou všechny uzly identické a poskytují stejnou funkcionalitu. Každý uzel („node“) je schopný organizovat cluster, poskytovat agregované statistiky a operační informace o clusteru (6).

3.2.1.1 Bližší porovnání SQL a NoSQL databází

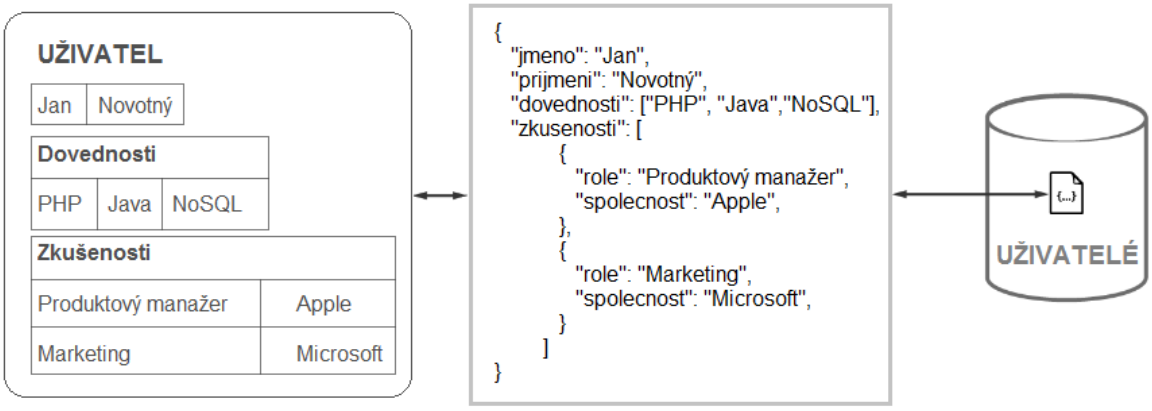
Datový model v relační databázi je pevný a definovaný statickým schématem. Je vhodný pro práci se strukturovanými daty a pro případy, kdy mají všechny záznamy stejné vlastnosti. Všechna data se ukládají do tabulek s tím, že jedna entita může být rozložena do několika různých tabulek a její vyhledání nebo aktualizace vyžaduje několik spojení (viz obr. 10). Konzistence dat je zde vynucována a upřednostňována před dostupností a výkonem (7).



Obrázek 10 - Entita v relační databázi (převzato a upraveno z (5))

Tento přístup může způsobit problémy například při změně nároků na aplikaci vedoucím ke změně v modelu dat. V takovém případě je pak potřeba existující schéma modifikovat nebo dokonce vyjednat změnu schématu od administrátorů databáze, což by mělo za následek zpomalení nebo i zastavení vývoje, obzvlášť pokud má toto schéma vliv na další aplikace a služby (5).

Naproti tomu dokumentová NoSQL databáze žádné schéma nemá a ani staticky nedefinuje, jak musejí být data modelována. S NoSQL je model dat definován aplikačním modelem - aplikace a služby modelují data jako objekty, vícehodnotová data jako kolekce a relační data jako kolekce nebo hnízděné objekty (viz obr. 11).



Obrázek 11 - Entita v NoSQL databázi (převzato a upraveno z (5))

V relačních databázích jsou data čtena a zapisována rozloženými a znovu skládanými objekty, dokumentově orientovaná NoSQL databáze čte a zapisuje data v JSON formátu, takže jeden objekt může být zapsán jako jeden dokument a objekty mohou být čteny a zapisovány bez toho, aby byly „rozkouskovány“.

3.2.1.2 Dokumenty

Tělo dokumentu je tvořeno formou JSON (JavaScript Object Notation) objektu (viz obr. 11), což je kolekce párů klíč-hodnota, kde hodnota může být různého typu – číslo, řetězec, pole, pole polí atd. Každý dokument je identifikován pomocí ID, které může být automaticky vygenerováno nebo určeno aplikací. Jediným omezením je, že musí být unikátní v celé databázi a nelze jej změnit (8).

Dokumenty jsou ukládány do tzv. „bucketů“, což jsou logické oddíly v *clusteru*, využívané k organizování, řízení a analyzování skupiny dokumentů (analogie databáze). Existují dva typy – „Memcached“a „Couchbase“, přičemž v „Memcache clusteru“ nejsou data perzistentně uchovávána na disku, ale v RAM paměti (lze využít například zároveň s jinou databází pro cachování často využívaných dat). „Couchbase cluster“ poskytuje vysoce spolehlivé dynamicky konfigurovatelné distribuované datové úložiště. Maximální velikost jednoho záznamu (dokumentu) je 20 MB (9).

S dokumenty lze pracovat pomocí „pohledů“ (z angl. view) nebo dotazovacího jazyka N1QL.

Pohled je perzistentní index dokumentů v databázi, založený na technice map/reduce, pomocí kterého lze k uloženým dokumentům přistupovat, zpracovávat je, řadit je a vyhledávat podle různých kritérií. Důležitou komponentou je „mapovací funkce“, která přijímá dokument jako vstup a dvojice klíč - hodnota jako výstup. Pohled generuje index zavoláním mapovací funkce nad všemi uloženými dokumenty. Při každé změně dokumentu je potřeba pohledy aktualizovat (10).

3.2.1.3 Jazyk N1QL

N1QL (čte se „nikl“) je deklarativní dotazovací jazyk využívající SQL dialekt. Jeho struktura se skládá z příkazů, výrazů a komentářů (jsou podporovány blokové komentáře se syntaxí „/* text komentáře */“) (11).

Příkazy se dělí do dvou skupin:

1. DDL (Data Definition Language) – příkazy pro vytváření, úpravy a mazání indexů
2. DML (Data Manipulation Language) – příkazy pro výběr, vložení, upravení a smazání dat z dokumentů.

Typy výrazů v N1QL jsou např. skutečná hodnota, identifikátor, logická hodnota, kolekce, poddotaz, aritmetická hodnota apod.

Dotazy do databáze se posílají použitím jakéhokoliv DML příkazu. Tyto dotazy se provádějí nad JSON dokumenty, přičemž lze vytvářet dotazy nad několika dokumenty naráz pomocí příkazu JOIN. Jelikož mohou být data v dokumentech „hnízděná“ („pole polí“), jsou specifikovány operátory a funkce, které umožní procházení skrz hnízděná pole. Také lze používat standardní příkazy jako WHERE, GROUP BY, LIMIT atd. (viz ukázka kódu 3).

```
SELECT intX, intY, level, recalculatedX, recalculatedY FROM fingerprints WHERE (level =
'J1NP' OR level = 'J2NP' OR level = 'J3NP' OR level = 'J4NP') LIMIT 5
```

Ukázka kódu 3 – N1QL, Příklad dotazu SELECT

Výsledkem je sada dokumentů, které mohou, ale nemusí mít stejnou strukturu. Pokud je požadováno, aby měly vrácené dokumenty stejný tvar, měl by příkaz SELECT obsahovat pevný seznam názvů atributů (sloupců). K nejednotné struktuře vrácených dokumentů může dojít při použití dotazu „SELECT * …“, pak je dokument, který vyhovuje zaslanému dotazu, vrácen tak, jak je uložen v databázi.

Pro manipulaci s hnízděnými daty jsou v N1QL používány 1) tečkové notace a 2) operátory nad kolekcemi.

ad 1) jsou podporovány cesty využívající tečkovou notaci pro identifikaci logické polohy atributu v rámci dokumentu.

V úkázce kódu 4 je příklad tečkové notace, kdy cesta odkazuje na hodnotu atributu „city“ v objektu „address“(převzato z (12)).

```
address.city
```

Ukázka kódu 4 – N1QL, Tečková notace

Další možností, jak zjistit informaci uvnitř pole je pomocí „array syntaxe“. Ukázka kódu 5 ukazuje, jak lze získat hodnotu „2012“ z atributu "revisions", která obsahuje pole hodnot „[2013, 2012, 2011, 2010]“.

```
revisions[1]
```

Ukázka kódu 5 – N1QL. Array syntaxe

Použití cesty poskytuje možnost vyhledání dat ve struktuře dokumentu bez nutnosti získání celého dokumentu (13).

ad 2) Pro dotazy nad kolekcemi se mohou použít operátory ANY, EVERY, ARRAY, FIRST, EXISTS, IN a WITHIN (14).

Například ANY a EVERY procházejí elementy v kolekci a vrací boolean hodnotu true, pokud element (ANY – alespoň jeden, EVERY - všechny) splňuje stanovenou podmínku (viz ukázka kódu).

```
ANY variable ( IN | WITHIN ) expression
        [ , variable ( IN | WITHIN ) expression ]*
        SATISFIES condition END
```

Ukázka kódu 6 – N1QL, Výraz ANY

ARRAY mapuje a filtruje elementy nebo atributy kolekce, objektu nebo objektů. Vyhodnocuje pole v operandu výrazu, které vyhovuje podmínce WHEN (viz ukázka kódu 7).

```
ARRAY expression FOR variable ( IN | WITHIN ) expression
        [ , variable ( IN | WITHIN ) expression ]* [ ( WHEN condition) ] END
```

Ukázka kódu 7 – N1QL, Výraz ARRAY

3.2.1.4 Indexování

Přístup k datům je v Couchbase urychlován pomocí indexů. Záznamy v produkčních Couchbase clusterech jsou distribuovány napříč mnoha uzly. Couchbase podporuje škálu globálních sekundárních indexů a indexů založených na view, které lze využít pro zvýšení výkonu za cenu režijních nákladů na údržbu indexů. Vzhledem k tomu, že data indexu jsou menší než samotný dokument, je často možné skladovat celý index v jednom uzlu raději než ho distribuovat přes několik uzlů. Díky tomu může být poskytnut přístup k uzlu jako lokální operace, čímž se zredukuje čas přístupu do sítě (15).

Couchbase verze 4.5 na rozdíl od předchozích verzí přidává možnost vytvoření globálních indexů nad elementy v poli (viz ukázka kódu 8) a tím optimalizuje provádění dotazů týkajících se hodnot uložených v nižších úrovních hnízděných polí (16).

```
CREATE INDEX [ index_name ]
        ON named_keyspace_ref ( expression || array_expression, … )
        [ WHERE filter_expressions ]
        [ USING GSI ]
        [ WITH { "nodes": [ "node_name" ],
                 "defer_build": true | false
                 }
        ];
```

Ukázka kódu 8 – N1QL, Globální index nad elementy pole (převzato z (16))

Vysvětlivky:

- *index_name* – název nového indexu
- *named_keyspace_ref* – název bucketu
- *expression* – název atributu nebo funkce, která vrací jeden prvek
- *array_expression* – mapování a filtrování elementů nebo atributů použitím operátoru ARRAY (viz ukázka kódu 9)
- *filter_expression* – specifikace *WHERE* podmínky
- *USING GSI* – použití typu indexu
- *WITH options* – dodatečné podmínky

```
array_expression ::=
        [ ALL ] DISTINCT ARRAY var_expression FOR
                variable1 ( IN | WITHIN ) [TO_ARRAY] expression1
                [ , variable2 ( IN | WITHIN ) [TO_ARRAY] expression2 ]
                [ ( WHEN condition ) ] END
```

Ukázka kódu 9 – N1QL, Array expression (převzato z (16))

V aplikaci Fingerprints je např. využíváno filtrování dokumentů podle hodnoty *mac* uložené v poli *wifiScans*. V ukázce kódu 10 je uveden příklad vytvoření globálního indexu nad elementy tohoto pole, který byl vytvořen podle návodu výše, a v ukázce kódu 11 je příklad dotazu pro nalezení všech dokumentů obsahujících hodnotu „nazevWifi“ v poli *wifiScans*.

```
CREATE INDEX `mwifi` ON `fingerprints`((distinct (array `m` for `m` in `wifiScans` end)))
```

Ukázka kódu 10 – N1QL, Globální index nad elementy pole wifiScans

```
SELECT * FROM fingerprints
WHERE ANY m IN wifiScans SATISFIES m.mac = 'nazevWifi' END
```

Ukázka kódu 11 – N1QL, Příkaz pro vyhledání položek v poli za použití glob. indexu

3.2.2 MySQL

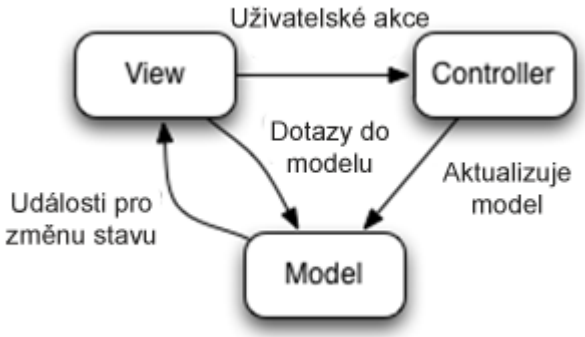
MySQL je relační databázový sytém, ve kterém (jak již bylo zmíněno v kapitole 3.2.1.1 *Bližší porovnání SQL a NoSQL databází*) jsou data uspořádávána do tabulek, řádků a sloupců, nad nimiž jsou pak vykonávány dotazy v jazyce SQL. Každá tabulka tedy obsahuje záznamy tvořené řádky (jeden řádek je právě jeden záznam) a každý sloupec reprezentuje část dat (určitého typu) záznamu (5).

3.2.3 Nette Framework

Nette je open-source framework pro vývoj webových aplikací v jazyce PHP. Jeho cílem je mj. ulehčit, zrychlit a zpřehlednit psaní kódu. Obsahuje šablonovací systém, ladící nástroje a zabezpečení před zranitelnostmi, konkrétně před útoky Cross-Site Scripting, Cross-Site Request Forgery, URL attack, control codes, invalid UTF-8 , Session hijacking, session stealing, a session fixation (17).

Framework využívá architektonický vzor MVP (Model-View-Presenter), který je odvozen ze vzoru MVC (Model-View-Controller).

V MVC reprezentuje model základní data objektu, například stav zaškrtnutí/nezaškrtnutí check boxu nebo textový řetězec ze vstupního pole. View (pohled) přistupuje k datům v modelu a specifikuje, jak se data z modelu přenesou na obrazovku, například skutečný zaškrtnutý check box nebo vstupní pole obsahující textový řetězec. Controller (řadič) určuje, jak interakce uživatele s pohledem ve formě událostí, která způsobí změnu dat v modelu – kliknutí na check box změní stav ze „zaškrtnutý“ na „nezaškrtnutý“ nebo obráceně a vypsání textu do vstupního pole změní hodnotu textového řetězce, atd. Model uzavírá smyčku upozorněním pohledu, že byl jeho stav změněn a je potřeba pohled překreslit (viz obr. 12). GUI objekty vznikají vytvářením podtříd a přizpůsobováním základních tříd pro abstrakce modelu, pohledu a řadiče, které poskytuje knihovna tříd (18).



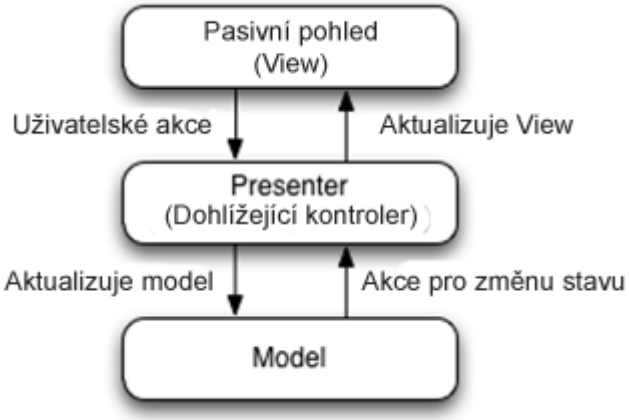
Obrázek 12 – Model MVC (převzato a přeloženo z (19))

Přístup MVP, který používá Nette Framework je zobecněná forma vzoru MVC. Model zajišťuje přístup k datům a manipulaci s nimi a pohled (tvořen z velké části šablonami) převádí data reprezentovaná modelem do podoby vhodné k prezentaci uživateli (viz obrázek 13).

Role presenteru je interpretace událostí inicializovaných uživatelem a poskytnutí logiky, která události mapuje na vhodné příkazy pro manipulaci s modelem.

Model sám o sobě s pohledem a presenterem vůbec nekomunikuje. Pohled s modelem komunikovat nemusí (pak jde o tzv. „pasivní pohled“), nebo naopak může získávat data přímo z něj, záleží na zvoleném přístupu. Presenter propojuje pohled s modelem (ne naopak) a realizuje uživatelské akce (změna pohledu, interakce v rámci aktuálního pohledu nebo příkaz pro model).

Ideálně by se měla činnost modelu omezit pouze na práci s databází (získávání dat), logika v pohledu na podmínky if/else a iterace a logika presenteru na plnění šablon, registraci helperů a filtrů, sestavení stromu komponent a úkolování tříd z modelu (20).



Obrázek 13 – Model MVP (převzato a přeloženo z (19))

Natte obsahuje šablonovací systém Latte, používající makra, která zjednodušují a zpřehledňují psaní kódu v šabloně (21).

K nastavení aplikace slouží třída *Configurator*, která se použije v zaváděcím souboru *bootstrap.php* (umístěný ve složce app). Konfigurace se načte a zpracuje pouze jednou, výsledek se uloží do cache (složka *temp*). Startuje se zde *RobotLoader*, který má na starosti automatické načítání zdrojových souborů. Samotná konfigurace se zapisuje v *NEON* souboru (do třídy *Configurator* se přidá cesta). V něm lze nastavit databáze, routování, formuláře, e-maily, sessions a mnoho další (22).

V Nette jsou podporovány Ajaxové požadavky (23), avšak pro implementaci AJAX požadavků na straně klienta je potřeba stáhnout a připojit (např.) knihovnu *nette.ajax.js* ze zdroje (24).

3.2.4 jQuery

jQuery je javascriptová knihovna (otevřený software), která zjednodušuje práci s javascriptem na webových stránkách tím, že běžné úkoly vyžadující několik řádků javascriptového kódu zapouzdřuje do metod, které stačí zavolat jedním řádkem kódu. Umožňuje manipulaci s HTML/DOM, CSS, HTML událostní metody, efekty a animace, AJAX a další (25).

3.2.4.1 Wheelzoom

Přibližování a oddalování mapy (scrollováním) je prováděno přes plugin zvaný „Wheelzoom“ (4). Funguje na způsobu převedení obrázku („img“ element) na obrázek

pozadí („background-image“) a měněním hodnot „background-size“ a „background-position“ v CSS stylu. Původní „img“ element je nahrazen průhledným obrázkem. Software je vedený pod MIT licencí. Pro zpřístupnění funkcí pluginu se naváže funkce *wheelzoom* na element *div* s konkrétní třídou v jQuery skriptu.

3.2.4.2 JSON.VIEW

JSON.VIEW (26) je jQuery plugin vedený pod licencí MIT, který formátuje JSON data a převádí je na stromovou strukturu, kde jsou jednotlivé klíče i hodnoty stylovány podle typu a navíc u uzlů stromu obsahujících podstromy přidává možnost „zabalení“ a „rozbalení“. Implementace spočívá ve vytvoření prázdného HTML elementu DIV s libovolným atributem ID, na který se pak naváže funkce v jQuery.

4 Implementace

4.1 Couchbase

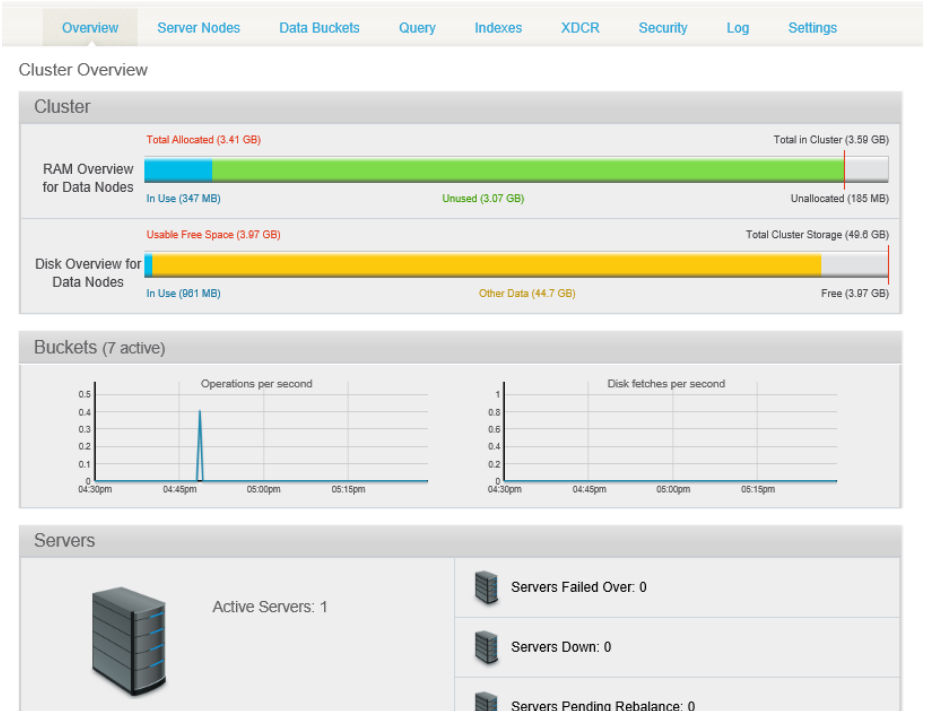
Data s fingerprinty jsou uložena v databázovém systému Couchbase na školním serveru. Nebylo vhodné, aby se při vývoji používala ostrá databáze, takže bylo potřeba Couchbase server stáhnout, nainstalovat a nastavit na lokálním serveru. Postup bude shrnut v kapitolách 4.1.1 *Zavedení databáze*, 4.1.2 *Nahrávání dat na server* a 4.1.3 I*ndexování*. Operační systém, na kterém se vyvíjelo a posléze nasazovalo, je Windows, veškeré postupy zde popsané tedy platí právě pro tento systém. Verze použitých rozšíření budou uvedeny v poznámkách.

4.1.1 Zavedení databáze

Couchbase server[8] je dostupný ke stažení zdarma na oficiálních stránkách Couchbase (27). Po instalaci je potřeba vytvořit nový bucket, kam se pak nahrají data (dokumenty), se kterými bude aplikace pracovat.

Konzole (viz obr. 14) sloužící jako uživatelské rozhraní pro Couchbase je dostupná v internetovém prohlížeči z adresy lokálního serveru, obvykle na portu 8091. Zde lze zobrazit např. přehled clusterů, uzly, vytvořené buckety, vytvořené indexy apod.

[8] Verze 4.5.0 Couchbase Server Community Edition



Obrázek 14 – Couchbase konzole

Nové buckety se vytváří v záložce „Data Buckets“ kliknutím na tlačítko „Create New Data Bucket“. Po zadání jména bucketu a jeho typu (výchozí je nastaven na Couchbase), nastavení velikosti paměti, možnosti replikace atd. se kliknutím na tlačítko „Create“ vytvoří nový bucket.

Připojení k databázi a práci s daty v aplikaci Fingerprints umožňuje PHP rozšíření *Couchbase PHP SDK*, které je k dispozici ke stažení ze zdroje (28). Je potřeba vybrat verzi balíčku SDK[9] pro používanou verzi PHP[10]. Instalace pro webový server Xampp spočívá ve zkopírování souboru *libcouchbasse.dll* do složky „c:\xampp\php” a “c:\xampp\apache\bin“, souboru *php_couchbase.dll* do složky „c:\xampp\php\ext“ a nakonec přidáním řádku (viz ukázka kódu 12) do souboru *php.ini*.

```
extension=php_couchbase.dll
```

Ukázka kódu 12 – Couchbase rozšíření v *php.ini*

[9] Couchbase Server PHP extension verze 2.1.0 (TS x86)

[10] PHP verze 5.6.12

4.1.2 Nahrávání dat na server

Záznamy o měření signálů obdržené ve formě JSON dokumentu byly extrahovány z již existující Couchbase. Každý záznam má svoje *ID* (unikátní identifikátor přidělený databází) a pole *value* obsahující informace o měření (viz kapitola 2.2 *Způsob uložení dat*).

Nejjednodušším způsobem nahrání JSON dokumentů na server je použitím nástroje *cbdocloader*. Předtím je ale potřeba převést záznamy do správného tvaru.

Každý záznam musí být převeden na samostatný soubor typu „.txt“, kde název souboru bude shodný s ID JSON dokumentu. K tomu poslouží jednoduchý PHP skript. Zároveň se v něm přepočítají souřadnice místa měření a vytvoří se nové souřadnice v datovém typu integer – obojí bude mj. sloužit pro zjišťování nejbližších bodů (vysvětleno v kapitole 3.1.3 *Problém posunutých souřadnic na mapě*) a přidají se jako nové atributy dokumentu.

Pro tento úkol byl vytvořen PHP skript *uploadData.php*[11], ve kterém se převede obsah textového souboru na asociativní pole, a každý záznam v tomto poli se uloží jako samostatný dokument s názvem vytvořeným podle ID (viz ukázky kódu 13, 14 a 15).

Nejprve se nahraje obsah ze souboru *data.txt* a převede se na asociativní pole (viz ukázka kódu 13).

```
$file = file_get_contents('./data.txt', true);
$decoded = json_decode($file, true);
```

Ukázka kódu 13 – Script pro úpravu dokumentů (1. část)

Všechny záznamy se projdou FOREACH cyklem a vytvoří se nový atribut *intX* a *intY* se stejnou hodnotou, jako mají atributy *x* a *y*, ale ve formátu *integer*, a *recalculatedX*, *recalculatedY*, které budou u záznamů z druhého, třetího a čtvrtého patra upraveny, aby fingerprinty pasovaly do mapy prvního podlaží (rozdílná měřítka a posunutí od levého a horního okraje). Při vytváření nových atributů se vždy ověřuje, zda dokument obsahuje původní naměřené souřadnice x a y (viz ukázka kódu 14).

[11] PHP soubor se skriptem je součástí přiložené přílohy s aplikací ve složce fingeprints/www/changeDataScript

```
foreach ($decoded['rows'] as $row) {
	if(!isset($row['value']['newX']) && isset($row['value']['x'])) {
		$row['value']['intX'] = intval($row['value']['x']);
	}
	if(!isset($row['value']['newY']) && isset($row['value']['y'])) {
		$row['value']['intY'] = intval($row['value']['y']);
	}
	if($row['value']['level'] == 'J2NP' && isset($row['value']['x']) && isset
	($row['value']['y']))
	{
		$recalculatedX = intval(round($row['value']['x'] * 0.8) - 21);
		$row['value']['recalculatedX'] = $recalculatedX;
		$recalculatedY = intval(round($row['value']['y'] * 0.758) - 66);
		$row['value']['recalculatedY'] = $recalculatedY;
	} elseif($row['value']['level'] == 'J3NP' && isset($row['value']['x']) && isset
	($row['value']['y']))
	{
		$recalculatedX = intval(round($row['value']['x'] * 0.907) - 32);
		$row['value']['recalculatedX'] = $recalculatedX;
		$recalculatedY = intval(round($row['value']['y'] * 0.840) + 65);
		$row['value']['recalculatedY'] = $recalculatedY;
	} elseif ($row['value']['level'] == 'J4NP' && isset($row['value']['x']) && isset
	($row['value']['y']))
		{
		$recalculatedX = intval(round($row['value']['x'] * 0.8) - 43);
		$row['value']['recalculatedX'] = $recalculatedX;
		$recalculatedY = intval(round($row['value']['y'] * 0.742) + 110);
		$row['value']['recalculatedY'] = $recalculatedY;
	} elseif(isset($row['value']['x']) && isset($row['value']['y']))
	{
		$row['value']['recalculatedX'] = intval($row['value']['x']);
		$row['value']['recalculatedY'] = intval($row['value']['y']);
}
```

Ukázka kódu 14 – Script pro úpravu dokumentů (2. část)

Každý záznam se (v rámci *foreach* cyklu) uloží jako nový dokument do složky *new_data*. Název souboru je ID záznamu (viz ukázka kódu 15).

```
$path = 'new_data/';
	$value = json_encode($row['value']);
	$myFile = fopen($path . $row['id'] . '.txt', 'w');
	fwrite($myFile, $value);
}
```

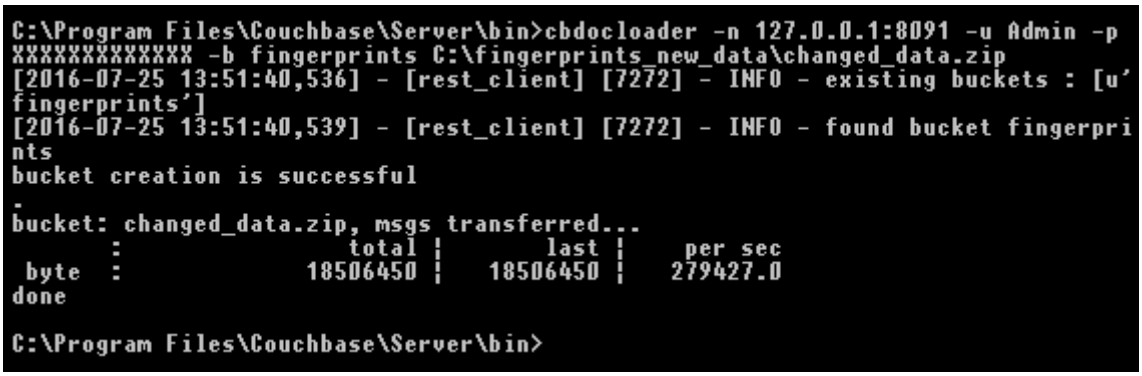
Ukázka kódu 15 – Script pro úpravu dokumentů (3. část)

Nově vytvořené dokumenty je potřeba zabalit do *zip* souboru. Poté se přes příkazový řádek vstoupí do adresáře, kde se nachází nástroj Couchbase *cbdocloader* (obvykle „C:\Program Files\Couchbase\Server\bin") a provede se příkaz k nahrání dat na server. Tvar příkazu je předveden v ukázce kódu 16.

```
cbdocloader –n adresaServeru –u uzivatelskeJmeno –p heslo –b nazevBucketu
C:\cesta\k\soubor.zip
```

Ukázka kódu 16 – Cbdocloader, nahrání dokumentů do Couchbase

Při úspěšném nahrání dat na server se po pár vteřinách objeví oznámení o dokončení operace (viz obr. 15).



Obrázek 15 – Nahrávání dokumentů do Couchbase

4.1.3 Vytvoření indexů

Aby bylo možné procházet data v Couchbase pomocí dotazovacího jazyka N1QL, je potřeba vytvořit primární index na bucketu obsahujícím dokumenty se záznamy o fingerprintech (29). K tomu lze použít např. Couchbase konzoli nebo program *cbq.exe* (přes příkazový řádek), který se nachází ve stejném adresáři jako *cbdocloader*. Index se vytvoří příkazem uvedeným v ukázce kódu 17.

```
CREATE PRIMARY INDEX nazev_indexu ON nazev_bucketu USING GSI;
```

Ukázka kódu 17 – N1QL, Vytvoření primárního indexu

V tomto případě je názvem bucketu „fingerprints" a název indexu může být například „#primary". Po vytvoření primárního indexu lze nad bucketem provádět operace SELECT, JOIN, GROUP BY atd. podobně jako by se pracovalo s SQL databází.

Další dva indexy jsou potřebné pro rychlejší filtrování podle hodnot na nižší úrovni vnořených polí, konkrétně v polích *wifiScans* a *bleScans*. Příkazy uvedené

v ukázce kódu 18 vytvoří index s názvem „mwifi“ nad položkami pole *wifiScans* a obdobně „nble“ u pole *bleScans*.

```
CREATE INDEX `mwifi` ON `fingerprints`((distinct (array `m` for `m` in `wifiScans` end)))
CREATE INDEX `nble` ON `fingerprints`((distinct (array `n` for `n` in `bleScans` end)))
```

Ukázka kódu 18 – N1QL, Vytvoření globálních indexů

4.2 MySQL

4.2.1 Popis tabulky symbolických názvů

Na stránce „Aliases“ má uživatel možnost přejmenovat atributy MAC adresa, název uživatele a ID zařízení. Všechny nově vytvořené aliasy se ukládají do MySQL tabulky „aliases“, ze které se pak budou při prezentování informací o fingerprintech uživateli nově vytvořená pojmenování získávat a nahrazovat ta původní.

Tabulka s názvem „aliases“ obsahuje tři atributy:

- *original*
- *alias*
- *type*

U každého záznamu musí být všechny atributy nenulové a *original* a *alias* zároveň unikátní.

Pole *original* a *alias* jsou typu „varchar“ a mají maximální délku 30 znaků. Do sloupce *original* se ukládají originální názvy záznamů obsažené v couchbase a do sloupce *alias* symbolické názvy, které vytvořil uživatel.

Poslední sloupec *type* má definovaný typ „enum“ a slouží k rozpoznání typu atributu v couchbase, který byl přejmenován. Hodnoty, kterých může atribut typ nabývat, jsou „wifi“, „ble“, „user“, a „deviceId“.

4.2.2 Popis tabulky s informacemi o podlažích

Tabulka „levels“ bude obsahovat všechny informace o podlažích potřebné při vykreslování a filtrování fingerprintů. Konkrétně jsou zde přiřazeny názvy pater v Couchbase k názvu budovy, číslu patra a názvu souboru s prázdnou mapou, jsou

definovány barvy bodů patřících k podlaží (RGB a průhlednost), šířka a výška obrázku mapy v pixelech (používané při zjišťování souřadnic na obrázku) a kvůli zobrazování dvojích souřadnic při přejíždění kurzorem myši přes obrázek s mapou byly do tabulky přidány i údaje o rozdílu v měřítku a posunutí pater vůči jednomu vybranému patru.

Výčet sloupců:

- *building* (ENUM)
- *level_number* (UNSIGNED TINYINT)
- *couchbase_label* (CHAR)
- *file* (CHAR)
- *map_width* (UNSIGNED SMALLINT)
- *map_height* (UNSIGNED SMALLINT)
- *shift_x* (SMALLINT)
- *shift_y* (SMALLINT)
- *zoom_x* (FLOAT)
- *zoom_y* (FLOAT)
- *color_red* (UNSIGNED TINYINT)
- *color_green* (UNSIGNED TINYINT)
- *color_blue* (UNSIGNED TINYINT)
- *opacity* (UNSIGNED TINYINT)

Atribut *building* je typu ENUM, přičemž hodnoty množiny jsou shodné s názvy budov v konfiguračním souboru aplikace Fingerprints (viz kapitola 4.3.1 Konfigurace). Hodnoty složek barev se budou pohybovat v rozmezí 0 – 255 a *opacity* (průhlednost) 0 – 100. Informace o posunutí (*shift_x*, *shift_y*) jsou uvedeny v pixelech a může se jednat o čísla kladná i záporná.

4.3 Nette Framework

4.3.1 Konfigurace

Konfigurační soubor *config.neon* obsahuje:

1. parametry (viz ukázka kódu 19)
2. nastavení časového pásma
3. nastavení aplikace

4. nastavení session
5. vytvořené služby
6. nastavení MySQL databáze

Parametry *couchbaseAddress*, *couchbaseName* a *n1qAddress* se použijí pro konstrukci služby *Couchbase service*:

- *couchbaseAddress* - adresa Couchbase serveru
- *couchbaseName* - název bucketu, kde jsou uložené záznamy o fingerprintech
- *n1qlAddress* – adresa pro používání N1QL query

Další hodnoty parametrů se použijí při vytváření a mazání vygenerovaných souborů:

- *cacheFolderPath* - nadefinování cesty, kam se budou ukládat nově vytvořené obrázky s fingerprinty a textové soubory s textem N1QL dotazu, který tyto fingerprinty vyhledal v Couchbase
- *emptyMapsFolderPath* – cesta k obrázkům prázdných map (do nich se budou fingerprinty vykreslovat)
- *cacheFileExpirationTime* - časové omezení (v jednotkách vteřin) existence vygenerovaných souborů - starší soubory budou smazány
- *cacheImageFormatString* - typ souboru obrázku
- *imageFolderLimit* - maximální velikost (v MB) adresáře "generated"

Poslední skupina parametrů se týká budov podporovaných v aplikaci. Jednotlivé budovy jsou označeny velkými písmeny (např. „J“) zde v konfiguračním souboru a v MySQL databázi. Při přidávání nových budov je potřeba vložení nové dvojice klíč - hodnota do pole *menuBuildings*, přičemž klíč musí být stejný jako atribut *building* v MySQL tabulce *levels* (viz kapitola 4.2.2 *Popis tabulky s informacemi o podlažích*) a hodnota je pak text v menu (nemá vliv na běh aplikace). Pokud na sebe mapy budovy pixelově nesedí (jako je tomu u budovy J), je potřeba přidat označení budovy i do pole *buildingsRecalculation*.

- *defaultBuilding* – stanovení defaultní budovy
- *menuBuildings* – seznam budov, které se přidají do hlavního menu, do záložky „Map“
- *buildingsRecalculation* – seznam budov vyžadujících přepočítání souřadnic

```
parameters:
	couchbaseAddress: 'http://127.0.0.1:8091'
	couchbaseName: 'fingerprints'
	n1qlAddress: '127.0.0.1:8093'
	cacheFileExpirationTime = 300
	cacheFolderPath = 'images/generated/'
	cacheImageFormatString = 'png'
	imageFolderLimit = 200
	emptyMapsFolderPath = 'images/maps/'
	minSigalStrength = -113
	maxSignalStrength = -10
	defaultBuilding = "J"
	menuBuildings = ["J" = "J Building", "X" = "X Building (fake)"]
	buildingsRecalculation = ["J"]
```

Ukázka kódu 19 – Nette Konfigurace, Parametry

Nastavení časového pásma sessions a mapování presenterů bylo ponecháno v původním stavu.

Kromě *FormFactory* a *RouterFactory* slouží položky seznamu „services“ k nadefinování vlastních služeb (viz ukázka kódu 20), tvořící modelovou část aplikace (především práce s daty). Hodnoty v závorkách jsou parametry obsažené v části „parameters“ a budou použity v konstruktorech těchto služeb. Více informací bude sděleno v kapitole 4.3.4 *Model*.

```
services:
	- App\Model\ConfigParametersService(%cacheFileExpirationTime%,
	                                    %defaultBuilding%, %menuBuildings%,
	                                    %buildingsRecalculation%,
	                                    %imageFolderLimit%)
	- App\Model\CouchbaseService(%couchbaseAddress%, %couchbaseName%,
	                             %n1qlAddress%)
	- App\Model\MysqlDatabaseService()
	- App\Model\GraphicsService(%cacheFolderPath%, %cacheImageFormatString%,
	                            %emptyMaps_folderPath%, %minSigalStrength%,
	                            %maxSigalStrength%)
	- App\Model\N1qlTextService()
	- App\Forms\SignFormFactory
	router: App\RouterFactory::createRouter
```

Ukázka kódu 20 – Nette Konfigurace, Služby

Pomocí *database* se vytváří připojení k jednotlivým databázím (viz ukázka kódu 21). Těch může být v konfiguračním souboru nadefinováno několik, přičemž je potřeba mj. zadat název připojení, adresu serveru a název databáze. V aplikaci *Fingerprints*

se bude pracovat s MySQL databází „fingerprints“ a připojení k ní je nazváno „default“ (viz ukázka kódu 21).

```
database:
	default:
		dsn:		"mysql:host=127.0.0.1;dbname=fingerprints"
		user:		"root"
		password:	""
		options:	[PDO::MYSQL_ATTR_COMPRESS = true]
		debugger:	true
		explain:	true
		 conventions:	discovered
		autowired:	true
```

Ukázka kódu 21 – Nette Konfigurace, Databáze

Mimo to se bude používat ještě připojení do Couchbase. Parametry s adresou a názvem databáze byly definovány v sekci *Parameters* a připojení se vytvoří ve službě *CouchbaseService*.

4.3.2 Model

Podle definice MVP (viz kapitola 3.2.3 *Nette Framework*) má modelová část aplikace za úkol přístup a práci s daty. V aplikaci Fingerprints bylo vytvořeno celkem pět služeb:

- *ConfigParametersService* - získání hodnot parametrů z konfiguračního souboru)
- *CouchbaseService* – připojení ke Couchbase a odesílání dotazů
- *GraphicsService* – práce s obrázky (vytváření nových, hledání „cachovaných“ apod.)
- *MysqlDatabaseService* – funkce pro práci s MySQL tabulkami
- *N1qlTextService* – dynamické skládání textů dotazů do Couchbase

V kapitolách 4.3.2.1 až 4.3.2.5 jsou stručně vysvětleny vytvořené služby a jejich metody. Konkrétní případy využití budou vymezeny při popisu presenterů (kapitoly 4.3.3.2 až 4.3.3.4).

4.3.2.1 Config Parameters Service

Tato třída slouží k získání hodnot parametrů (jejich význam vysvětlen v kapitole 4.3.2 *Konfigurace*) definovaných v konfiguračním souboru *config.neon*.

V konstruktoru se parametry uloží do globálních proměnných (viz ukázka kódu 22).

```
private $cacheFileExpirationTime;
private $defaultBuilding;
private $menuBuildings;
private $buildingsRecalculation;
private $imageFolderLimit;
private $imageFolderPath;

public function __construct($cacheFileExpirationTime , $defaultBuilding, $menuBuildings,
$buildingRecalculation, $imageFolderPath, $imageFolderLimit)
{
        $this->cacheFileExpirationTime = $cacheExpirationTime;
        $this->defaultBuilding = $defaultBuilding;
        $this->menuBuildings = $menuBuildings;
        $this->buildingsRecalculation = $buildingRecalculation;
        $this->imageFolderLimit = $imageFolderLimit;
        $this->imageFolderPath = $imageFolderPath;
}
```

Ukázka kódu 22 – Config Parameters Service, Konstruktor

Ostatní funkce po zavolání vracejí požadovaný parametr jako návratovou hodnotu. *GetBuildingRecalculation* přijímá atribut *building*, podle kterého zjišťuje, jestli je daný název budovy obsažený v seznamu budov vyžadující přepočítání souřadnic (seznam je získán z konfiguračního souboru) a vrací hodnotu TRUE nebo FALSE.

4.3.2.2 Couchbase Service

Služba CouchbaseService umožňuje připojení ke Couchbase databázi a obsahuje funkce pro vyhledávání dokumentů.

Ve funkci *_construct* (viz ukázka kódu 23) se z konfiguračního souboru získá název bucketu, adresa Couchbase serveru a N1QL serveru a vytvoří se připojení k databázi. Funkcí *enableN1ql* se zpřístupní podpora jazyka N1QL.

```
private $bucket;

public function __construct($couchbaseAddress, $couchbaseName, $n1qlAddress)
{
	$cluster = new\CouchbaseCluster($couchbaseAddress);
	$bucket = $cluster->openBucket($couchbaseName);
	$bucket->enableN1ql(array($n1qlAddress);
	$this->bucket = $bucket;
}
```

Ukázka kódu 23 – Couchbase Service, Konstruktor

Metoda *getData* (viz ukázka kódu 24) se volá při každém dotazu do couchbase. Přijímá jako parametr text N1QL dotazu, provede dotaz nad Couchbase databází a vrátí výsledek dotazu. Aby nedošlo k vypsání chybové hlášky na stránku při špatně zadaném textu dotazu (např. přes N1QL konzoli v dolní části stránky s mapou fingerprintů), je pokus o získání dat z databáze ošetřen příkazem TRY-CATCH. Při úspěšně vykonaném dotazu je vrácenou hodnotou z Couchbase instance třídy *stdClass*. Pomocí funkce *decodeData* jsou data dekódována do tvaru PHP array (json_encode, json_decode) a následně poslána zpět jako návratová hodnota. Při neúspěchu (pravděpodobně syntaktická chyba v textu dotazu) se vrátí pole s klíčem „error“ a hodnotou obsahujícím text, který se může použít k upozornění uživatele o vzniklé chybě.

```
public function getData($queryText)
{
	$query = \CouchbaseN1qlQuery::fromString($queryText);
	try {
		$data = $this->bucket->query($query);
	} catch (\CouchbaseException $e)
	{
		return (['error' => 'The query failed.']);
	}
	$decoded_data = $this->decodeData($data);
	return $decoded_data;
}
```

Ukázka kódu 24 – Couchbase Service, Získání dat z Couchbase

GetAllBles, *getAllWifis*, *getAllUsers* a *getAllDeviceIds* se používají při vytváření našeptávačů a *getAllBrands* při vytváření seznamu pro HTML element *select box* ve filtrovacím formuláři. Každá z těchto metod má pevně daný text dotazu do Couchbase, který se použije při volání *getData*. Vráceným výsledkem jsou všechny existující hodnoty v Couchbase, protříděné tak, aby byla každá hodnota obsažena v poli jen jednou (použitím výrazu „DISTINCT“ v textu dotazu, viz ukázka kódu 25).

```
public function getAllWifis()
{
        $queryText =  'SELECT DISTINCT wifi.mac FROM fingerprints p UNNEST p.wifiScans AS
                        wifi';
        $result = $this->getData($queryText);
        return $result;
}
```

Ukázka kódu 25 – Couchbase Service, Nelezení všech MAC adres ve Wi-Fi skenech

FindDocumentById (viz ukázka kódu 26) slouží k vyhledání konkrétního záznamu (fingerprintu) v Couchbase podle hodnoty v přijaté proměnné *id* a je používána pro zobrazení stromové struktury fingerprintu na stránce *Document*.

```
public function findDocumentById($id)
{
        $queryText = 'SELECT fingerprints, meta(fingerprints) AS meta FROM fingerprints
                        WHERE meta(fingerprints).id="'. $id .'"';
                $result = $this->getData($queryText);
                return $result;
}
```

Ukázka kódu 26 – Couchbase Service, Nalezení dokumentu podle ID

4.3.2.3 MySQL Database Service

MysqlDatabaseService je třída používaná k získávání dat z MySQL databáze. Má jednu globální proměnnou *database,* která se naplní ve funkci *_construct*. Ta se volá automaticky a injektuje databázi (parametry funkce jsou obsažené v konfiguračním souboru *config.neon*). Další metody se použijí k obstarání dat, o které požádal jeden z presenterů, o vytvoření či úpravu záznamů podle přijatých hodnot nebo o smazání záznamů. Metody pracují se dvěma existujícími tabulkami – *aliases* a *levels*.

```
private $database;

public function __construct(\Nette\Database\Context $database)
{
        $this->database = $database;
}
```

Ukázka kódu 27 – MySQL Database Service, Konstruktor

Funkce pro práci se symbolickými názvy:

GetAllAliases (viz ukázka kódu 28) přijímá parametr *type* a najde všechny záznamy z tabulky *aliases* obsahující tuto určitou hodnotu ve sloupci *type*. Návratovou hodnotou je asociativní pole s páry „originální název – alias“.

```
public function getAllAliases($type)
{
        $result = $this->database->table('aliases')->where('type', $type)->order('alias')
                ->fetchPairs('original','alias');
        return $result;
}
```

Ukázka kódu 28 – MySQL Database Service, Nalezení aliasů podle typu

FindPairAliasOriginal vyhledá jeden konkrétní záznam v tabulce podle přijatých parametrů *alias* a *type*. Vrátí pole (klíč-hodnota) „alias“ => „original“.

CheckIfAliasExists najde hodnotu atributu *alias* podle přijatého parametru. Bude se používat k ověření, zda příslušný alias již existuje. Obdobně funguje metoda *checkIfOriginalExists*.

FindAliasToOriginal vyhledá v tabulce „aliases“ dvojici *„original – alias“* podle přijatých hodnot *original* a *type*.

DeleteAlias slouží k smazání řádku. Přijímá dva parametry – *alias* a *type* a nic nevrací.

AddAlias vytvoří nový řádek v tabulce. Přijímá všechny tři hodnoty atributů – *original*, *alias* a *type*.

Poslední metodou pracující s tabulkou *aliases* je *UpdateAlias* s parametry *original* a *alias.* Má za úkol upravit již existující záznam. Nejprve zkontroluje, jestli tabulka přijatou kombinaci hodnot originálního názvu a symbolického názvu již obsahuje (nějakému jinému originálu už byl tento alias přidělen) a pokud ne, pak změní název aliasu patřící zadanému originálu na přijatou hodnotu v proměnné *alias*.

Funkce pro získání informací o budovách a podlažích:

Vyhledání všech budov v tabulce přes *getAllBuildings* se používá pro vytvoření položek v menu.

Kvůli označení pater budov v Couchbase textovými řetězci (např. „J1NP“, „A3NP“) a absenci atributu s číslem patra byl v tabulce *levels* vytvořen sloupec *couchbase_label* a ve službě *MysqlDatabaseService* několik funkcí, které k sobě přiřazují názvy a čísla pater nebo vyhledávají označení pater v Couchbase pro patra konkrétní budovy:

GetAllLevelLabels přijímá název budovy (např. „J“) a vrací pole se všemi Couchbase označeními pro podlaží patřící do této budovy ([“J1NP”, “J2NP”,…]).

GetLevelNumber přijímá Couchbase označení podlaží a vrací jeho číslo.

GetAllLevels se používá pro nalezení všech existujících pater v budově pro HTML element *selectbox* ve filtrovacím formuláři a vrací dvojice „číslo podlaží – číslo podlaží“ seřazené vzestupně (např. [1 => 1, 2 => 2]).

GetAllLevelNumbersAndLevelLabels vrací dvojice „číslo patra – označení patra v Couchbase“ podle přijatého názvu budovy (např. [1 => “J1NP”, 2 => “J2NP”]).

GetLabelsAndLevels je stejná jako předchozí funkce, jen jsou klíče a hodnoty v poli prohozené (např. [“J1NP” => 1, “J2NP” => 2]).

GetCouchbaseLabelForLevel zjišťuje, jak je určité patro určité budovy pojmenováno v Couchbase a vrací tuto hodnotu (např. „J4NP“).

GetCouchbaseLevelForLabel vyhledá číslo patra podle přijatého označení patra a názvu budovy (např. podle „J3NP“ vyhledá a vrátí hodnotu „3“).

Funkce pro práci s mapami budov a vykreslování fingerprintů:

GetMapWidth vrací šířku mapy v pixelech. Přijímá označení patra a název budovy. Obdobně *getMapHeight* vrací výšku mapy.

GetLevelColors má za úkol sestavit pole barev používaných k vykreslení fingerprintů na jednotlivých patrech. V rámci *FOREACH* cyklu jsou volány funkce *getColorRed*, *getColorGreen*, *getColorBlue* a *getColorOpacity*. Proměnná *level* zde obsahuje číslo patra.

Metody pro hledání hodnot barev (*getColorRed*, *getColorGreen* a *getColorBlue*) přijímají název budovy a číslo patra. Hodnoty ve sloupcích *color_red*, *color_green* atd. definují intenzitu dané složky v barevném modelu RGB (0 - 255). *GetColorOpacity* vrací průhlednost barvy (0 - 100).

GetEmptyMapName vrací název prázdné mapy, vyhledaný podle názvu budovy a čísla patra.

GetLevelZoomAndShift přijímá označení patra a vrací pole s hodnotami přiblížení a posunutí souřadnic.

4.3.2.4 Graphics Service

Třída *Graphics Service* poskytuje funkce používané při vykreslování fingerprintů do mapy. Využívá instanci třídy Nette\Utils\Image, která je určena pro manipulaci s obrázky.

V konstruktoru služby se přijímají parametry ze souboru *config.neon*, které se následně uloží do globálních proměnných třídy:

- *cacheFolderPath* – cesta k adresáři, kam se vytvářejí nové soubory
- *cacheImageFormatString* – typ obrázku (např. PNG)
- *emptyMapsFolderPath* – cesta k adresáři s obrázky prázdných map
- *minSigalStrength* – minimální hodnota síly signálu
- *maxSigalStrength* – maximální hodnota síly signálu

Funkce *drawPicture* řeší samotné vytváření souboru obrázku, vykreslování fingerprintů (barevných bodů) do něj a jeho následné uložení. Kromě toho také vytváří textový soubor s informacemi o obrázku (text dotazu korespondujícím s vytvořeným obrázkem, název filtrovaného patra, název filtrované MAC adresy a informaci, zda byly vykreslené fingerprinty filtrovány přes formulář nebo konzoli).

Přijatými proměnnými jsou:

- *levelLabel* – označení podlaží v Couchbase
- *emptyMapFileName* – název prázdné mapy, která se má použít jako plátno
- *rows* – data získaná z Couchbase
- *queryText* – text dotazu do Couchbase, podle kterého se data v *rows* získala
- *wifiAddr* – filtrovaná Wi-Fi MAC adresa nebo NULL, pokud se filtr nepoužil
- *bleAddr* – obdobně jako wifiAddr, pouze pro Bluetooth adresu
- *levels* – pole dvojic „číslo patra – označení v couchbase“ všech podlaží v aktuálně zvolené budově
- *colors* – pole s barvami, které se mají použít pro rozlišení fingerprintů z různých pater

- *recalculation* – boolean hodnota, která informuje o nutnosti použití přepočítaných souřadnic
- *console* – obsahuje *string* hodnotu „yes“, pokud byly záznamy z Couchbase získány přes konzoli dole na stránce Homepage, nebo „no“, pokud byl použit filtrovací formulář

Cesta, která se použije k vytvoření souboru obrázku a jeho pojmenování, se poskládá zavoláním metody *getPicturePath*. Název souboru tvoří řetězec „map-“ a zakódovaný text vyhledávacího dotazu do databáze. Cesta k souboru se skládá z cesty k adresáři (definované v konfiguračním souboru), kam se ukládají vytvořené obrázky, vytvořený název souboru a přípona souboru určující formát obrázku (taktéž zadaná v souboru *config.neon*). Tato metoda se volá i z presenteru v případě použití existujícího obrázku.

Samotné zahashování textu dotazu probíhá v metodě *getHash.* Pro kódování se použil algoritmus *MD5* (Message-Digest algorithm), jehož výstup tvoří řetězec znaků fixní délky (třicet dva znaků). Metoda přijímá původní N1QL text a vrací text zakódovaný.

Vytvoření plátna, do kterého se budou kreslit body, proběhne ve funkci *loadEmptyMap* (viz ukázka kódu 29). Vstupním parametrem je název souboru s prázdnou mapou. Podle něj a názvu adresáře, kde se prázdné mapy nacházejí (získaný z konfiguračního souboru) se složí cesta, odkud se následně přes funkci grafické knihovny načte obrázek, který se pošle zpět do *drawImage*.

```
private function loadEmptyMap($emptyMapFileName)
{
        $path = $this->emptyMapsFolderPath . $emptyMapFileName;
        $image = Image::fromFile($path);
        return $image;
}
```

Ukázka kódu 29 – Graphics Service, vytvoření plátna pro zakreslení značek

V cyklu FOREACH, který prochází pole přijatých výsledků z Couchbase (pole *rows*), se rozhoduje, jak a kam se má fingerprint nakreslit. Kvůli možnosti uživatele zadávat vlastní dotazy je ošetřeno, aby všechny zobrazené body patřily do mapy, na kterou se kreslí (např. mapa druhého patra budovy J). Pokud by tedy zadal dotaz „SELECT * FROM fingerprints“, vyhledají se všechny dokumenty v couchbase, ale vykreslí se pouze body patřící do výchozí mapy aktuální budovy. Hodnota *level* v poli *row* (označení patra) se porovná s polem *levels* (obsahujícím dvojici „číslo patra – označe-

ní v Couchbase“) přijatého v hlavičce metody (viz ukázka kódu 30). Pokud by se zde hodnota nenašla, přeskočí se na další iteraci cyklu. Pokud se ale najde, nalezené číslo patra uložené do proměnné *levelNr* se použije pro pozdější napsání číslovky doprostřed kruhu.

```
$cbLevel = $row['level'];
$levelNr = array_search ($cbLevel, $levels);
if ($levelNr == false) continue;
```

Ukázka kódu 30 – Graphics Service, zamezení vykreslení fingerprintů nepatřících do budovy

Výběr souřadnic, kam se má bod vykreslit, je ovlivněn hodnotou proměnné recalculation (řešeno v Homepage presenteru).

Při výběru barvy, která se použije na obarvení kruhu představujícího fingerprint, nastává jedna ze tří možných situací:

1. Pole *colors* **není prázdné**, použijí se barvy rozlišující patra, které jsou nadefinované v MySQL tabulce *levels* (viz ukázka kódu 31).

```
if($colors != null)
{
        $colorArray = $colors[$levelNr];
        $color = Image::rgb($colorArray['r'], $colorArray['g'], $colorArray['b'], $colorArray['o']);
```

Ukázka kódu 31 – Graphics Service, Obarvení bodu podle podlaží

2. Pole *colors* **je prázdné**, ale hodnota proměnné *wifiAddr* **není prázdná** a hodnota *bleAddr* **je prázdná** (došlo k filtrování podle Wi-Fi MAC adresy).

```
$color = $this->calculateColor($row['wifiScans'], $wifiAddr, 'mac');
```

Ukázka kódu 32 – Graphics Service, Obarvení bodu podle Wi-Fi adresy

3. Pole *colors* je prázdné a došlo k filtrování podle Bluetooth MAC adresy.

V případech 2. a 3. bude barva vypočítána podle průměrné síly signálu z filtrované adresy. K tomu poslouží metoda *calculateColor* (viz ukázky kódu 32 a 33). Parametry volání jsou pole skenů, název MAC adresy a textový řetězec obsahující název klíče, který se má v poli hledat („mac“ ve *wifiScans* nebo „address“ v *bleScans*). Minimální a maximální hodnota signálu byla definována v konfiguračním souboru. V poli skenů se najdou všechny naměřené hodnoty intenzity signálu přijaté MAC adresy, vypočítá se průměrná hodnota, ta se převede na procentuální hodnotu vzhledem k maximu a mi-

nimu a následně se odvodí barva bodu. Pokud by se stalo, že se adresa v poli nenachází, použije se barva šedá.

```
$onePercent = ($maxSigalStrength - $minSigalStrength) / 100;
$signalPercentage = ($maxSigalStrength - $rssi)/ $onePercent;
$red = (round((255/100)*(100 - $signalPercentage)));
$green = (round((255/100)*($signalPercentage)));
$color = Image::rgb($red, $green, 0, 50);
```

Ukázka kódu 33 – Graphics Service, Počítání barvy bodu podle signálu

Po vypočítání barvy se algoritmus vrátí do FOREACH cyklu v *drawPicture*, kde se bod (obarvený kruh) vykreslí do mapy na zvolených souřadnicích a doprostřed se zapíše číslo patra (viz ukázka kódu 34).

```
$image->filledEllipse($x, $y, 25, 25, $color);
$image->string(5, $x-3, $y-7, $letter, $white);
```

Ukázka kódu 34 - Graphics Service, Zakreslení fingerprintu do mapy

Zakreslení kruhů do mapy proběhne pomocí GD funkce *filledEllipse*. Přijímá parametry souřadnic *x* a *y* (střed kruhu), výšku a šířku obrazce a barvu výplně. Metodou *string* se do kruhu vypíše číslo podlaží. První parametr značí velikost písma, pak následují souřadnice upravené tak, aby bylo číslo vycentrované doprostřed kruhu, text, který se má vypsat, a barvu.

Kvůli „cachování“ obrázků se s každým novým souborem PNG vytvoří ještě navíc jeden textový dokument (viz ukázka kódu 35), který bude obsahovat JSON pole se čtyřmi položkami:

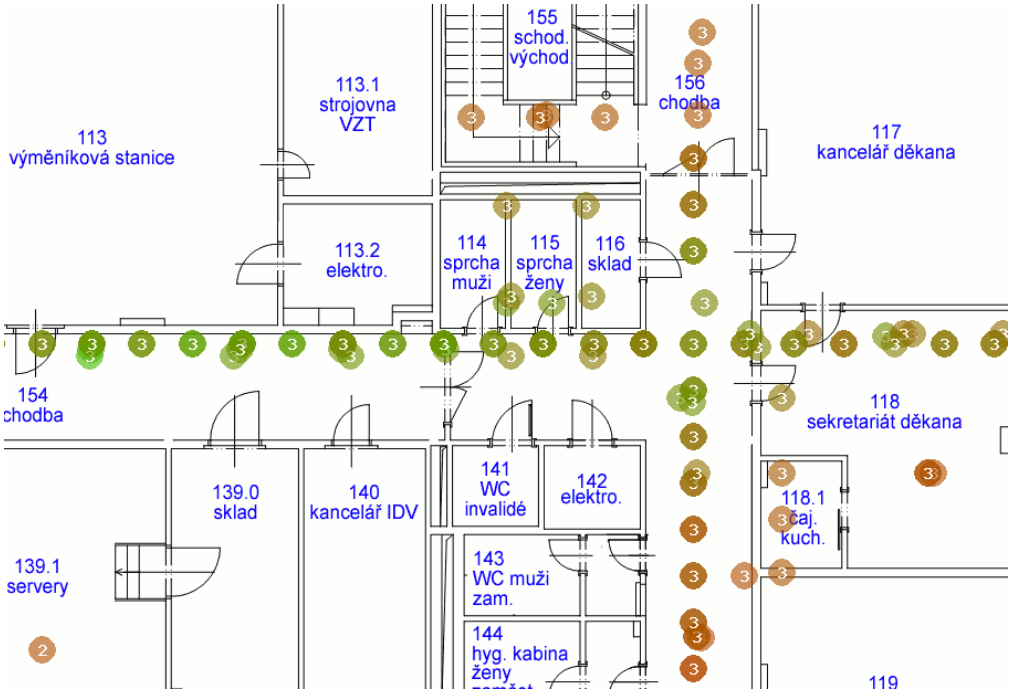
- *queryText* – text dotazu, který byl použit pro vyhledání fingerprintů
- *level* – označení podlaží (může být i hodnota NULL)
- *console* – podle toho, jestli se použila k odeslání dotazu konzole, hodnota „yes“ nebo „no“
- *mac* – název filtrované Mac adresy (Wi-Fi nebo Bluetooth, může být i hodnota NULL)

Funkce vytvářející textový dokument má název *saveInfoToFile* a přijímá argumenty *hashText* (zahashovaný queryText), *queryText*, *levelLabel*, *console* a *mac*. Název souboru se vytvoří obdobně jako název mapy, ale začíná textovým řetězcem „info-“ a má příponu „.txt“.

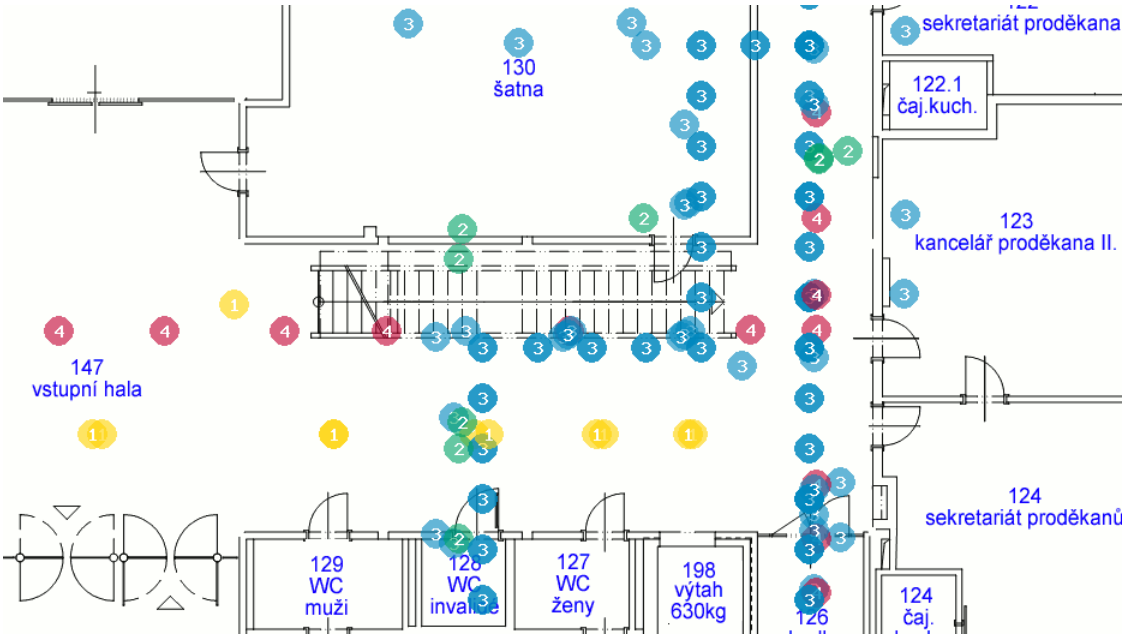
```
$file = fopen($this->cacheFolderPath . 'info-' . $hashText . '.txt', 'w');
$info = json_encode(['queryText' => $queryText, 'level' => $levelLabel,
                     'console' => $console, 'mac' => $mac]);
fwrite($file, $info);
fclose($file);
```

Ukázka kódu 35 – Graphics Service, Vytvoření textového souboru

Po uložení obrázku a textového souboru vrací metoda *drawPicture* cestu k souboru s nově vytvořeným obrázkem mapy se zakreslenými body (viz obrázky 16 a 17).



Obrázek 16 – Obarvení značek fingerprintů podle síly signálu



Obrázek 17 – Barvy fingerprintů podle podlaží

Kromě funkce pro vykreslení bodů do mapy a funkcí s ní souvisejících, obsahuje třída ještě metody, které se používají při hledání a používání již existujících obrázků.

Pro nalezení existujícího obrázku podle dotazu do Couchbase slouží metoda *checkForCache*. Přijatými hodnotami jsou *queryText* (text dotazu, který se má poslat do Couchbase), *console* (informace, jestli byla použita konzole) a *mac* (filtrovaná MAC adresa). Podle proměnné *queryText* se zjistí zahashovaný tvar dotazu a následně ověří existence souboru s takovýmto jménem („map-hashText.png“). Pro potvrzení, že jde opravdu o obrázek zobrazující fingerprinty nalezené podle toho dotazu je hodnota *queryText* porovnána s textem obsaženým v textovém souboru „info-“ v JSON poli pod klíčem „queryText“. Dále je kontrolována shoda přijaté hodnoty *console* a uložené hodnoty pod klíčem „console“. Pokud byl k filtrování použit formulář, je tato kontrola dostatečná a vrátí se hodnota TRUE (použije se „cachovaný“ obrázek). V případě použití konzole se ještě navíc zkontroluje shoda přijaté hodnoty *mac* s uloženou hodnotou v poli pod klíčem „mac“ (viz ukázka kódu 36).

```
…
$infoArray = json_decode($fileContent, true);
$fileQueryText = $infoArray['queryText'];
if($fileQueryText == $queryText)
{
        $infoConsole = $infoArray['console'];
        if($console == 'no' && $infoConsole == $console)
        {
                return true;
        } elseif (($console == 'yes') && ($infoConsole == $console)
                        && $infoArray['mac'] == $mac)
        {
                return true;
        } else
        {
                return false;
        }
} else
{
        return false;
 }
…
```

Ukázka kódu 36 – Graphics Service, Kontrola položek JSON pole v souboru „–info“

GetLevelLabelFromFile umožňuje určit název podlaží v případech použití „cachovaných“ obrázků vzniklých odesláním dotazu z N1QL konzole (problematika vysvětlena v kapitole 3.1.5 *Filtrování fingerprintů*). Podle přijatého textu dotazu nalezne příslušný textový dokument, získá hodnotu v JSON poli uloženou pod klíčem „level“ a tu pošle zpět jako návratovou hodnotu. Hodnotou může být i NULL (prázdný řetězec) a to v případě, že filtrované body nepocházely pouze z jednoho patra.

4.3.2.5 N1qlTextService

Důvod zavedení třídy *N1qlTextService* byl ten, že během filtrování (přes formulář) a hledání nejbližších fingerprintů je potřeba dynamicky skládat text dotazu podle vyplněných hodnot, aktuálně vybrané budovy (konkrétních názvů pater) a zároveň určit, zda se má pracovat se souřadnicemi originálními nebo přepočítanými (viz kapitola 3.1.3 *Problém posunutí souřadnic v jednotlivých patrech*) a bylo vhodné oddělit množství funkcí k tomu určených od ostatních metod v presenteru. Tato třída bude tedy sloužit výhradně k práci s textem dotazů do Couchbase.

Text dotazu podle filtru

Funkce *getTextForPointsFiltering* se volá při každém použití formuláře na stránce s mapou fingerprintů *filterData*. Přijímá dva argumenty – *activeFilters*, což je pole s hodnotami odeslaných z formuláře a názvem aktuální budovy, a boolean *recalculation* určující, zda je potřeba použít přepočítané souřadnice či nikoliv. Podle těchto přijatých hodnot poskládá text, který se má použít pro vyhledání bodů vyhovujících nastavení filtru (viz ukázka kódu 37).

```
$selectText = "SELECT intX, intY, level";
if($recalculation)
{
        $selectText =  $selectText . ", recalculatedX, recalculatedY";
}
if($activeFilters['wifi'] != null && $activeFilters['ble'] == null)
{
        $selectText = $selectText . ", wifiScans";
}
if($activeFilters['ble'] != null && $activeFilters['wifi'] == null)
{
        $selectText = $selectText . ", bleScans";
}
$selectText = $selectText . " FROM fingerprints";
$conditionText = $this->composeConditionText($activeFilters);
$queryText = $selectText . $conditionText;
```

Ukázka kódu 37 – N1QL Text Service, Skládání textu dotazu při použití filtrovacího formuláře

Výchozí text dotazu SELECT je výběr souřadnic a podlaží. Podle hodnoty *recalculation* se navíc vyhledají i souřadnice přepočítané. Další dvě podmínky určují, jestli se má vyhledat i pole se skeny – to je potřeba pouze v případě, že uživatel chce filtrovat body podle jedné MAC adresy (Wi-Fi nebo Bluetooth). Přidá se informace ze kterého bucketu se mají data získat a text podmínky, který se poskládá pomocí metody *composeConditionText*. Výsledek se pošle zpět do presenteru, že kterého byla funkce zavolána (*homepagePresenter* – viz kapitola 4.3.3.2 *Homepage – Zobrazení fingerprintů*).

Metoda *ComposeConditionText* skládá text podmínky podle hodnot přijatých v poli *activeFilters*. Hodnota zavedené proměnné *whereCondiBool* (typu boolean) bude při skládání jednotlivých částí podmínek pomáhat určit, zda se má použít výraz „WHERE“ nebo „AND“ – WHERE se použije pouze u první podmínky, všechny ostatní se musí navázat pomocí AND. Na začátku algoritmu je tedy nastavená na FALSE, po prvním použití výrazu WHERE se změní na TRUE.

Každá podmínka se přidá k textu obsaženému v *conditionText*. Postupně probíhá kontrola klíčů v poli *activeFilters*, jestli klíč s konkrétním názvem obsahuje hodnotu, která není rovna NULL – v takovém případě je potřeba přidat podmínku obsahující tuto hodnotu nebo podmínku od této hodnoty odvozenou.

Pro každý klíč v *activeFilters* byla vytvořena zvláštní funkce, protože texty podmínek se budou různit. Například pro vyhledání dokumentů podle ID zařízení bude text vypadat jako v ukázce kódu 38, zatímco podle Wi-Fi MAC adresy jako v ukázce kódu 39.

```
WHERE deviceID LIKE '%nazev_zarizeni%'
```

Ukázka kódu 38 – N1QL Text Service – Podmínka Id zařízení

```
WHERE  ANY n IN wifiScans SATISFIES n.address LIKE '%nazev_wifi_mac%' END
```

Ukázka kódu 39 – N1QL Text Service, Podmínka název Wi-Fi adresy

U každé nenulové hodnoty klíče v *activeFilters* se tedy zavolá odpovídající funkce, která vrátí text podmínky vztahující se k této hodnotě, ten se naváže na text podmínky již existující a hodnota *whereConditionBool* se nastaví na TRUE, ať už se u podmínky použil výraz WHERE nebo AND.

Po dokončení se vrátí text podmínky jako návratová hodnota (viz ukázka kódu 40).

```
if($activeFilters['wifi'] != null)
{
        $text = $this->getWifiCondition($activeFilters['wifi'], $whereCondiBool);
        $conditionText = $conditionText . $text;
        $whereCondiBool = true;
}
…
return $conditionText;
```

Ukázka kódu 40 – N1QL Text Service, Funkce pro skládání podmínek

Zvláštní případ je podmínka pro vyhledání podle podlaží (viz ukázka kódu 41). Pokud je aktivní filtr pro jedno patro, volá se funkce *getLevelCondition*, ale pokud je tento filtr prázdný nebo je hodnota „all“, je potřeba vyhledat dokumenty jen z určité budovy. K tomu poslouží metoda *getBuildingCondition*.

```
if($level != null && $level != 'all' )
{
        $text = $this->getLevelCondition($level, $whereCondiBool);
        $conditionText = $conditionText . $text;
        $whereCondiBool = true;
} else
{
        $text = $this->getBuildingCondition($activeFilters['buildingLevels'], $whereCondiBool);
        $conditionText = $conditionText . $text;
        $whereCondiBool = true;
}
```

Ukázka kódu 41 - N1QL Text Service, Podmínka podlaží a název budovy

Metody pro vytvoření podmínky podle jednoho filtru (uživatel, ID zařízení apod.) si jsou relativně podobné. Přijímají hodnotu, podle které se mají dokumenty v Couchbase protřídit a proměnnou *whereCondiBool* nastavenou na TRUE nebo FALSE. „FALSE“ značí, že do celkové WHERE podmínky ještě nebyl přidán výraz „WHERE“, tzn., jde o první přidanou podmínku. Při hodnotě TRUE se použije výraz „AND“. V ukázce kódu 42 je uveden příklad funkce *getWifiCondition*, která vrací podmínku použitou při filtrování podle Wi-Fi adresy přes filtrovací formulář.

```
$text = " ANY m IN wifiScans SATISFIES m.mac LIKE '%";
$likeBool = true;
$conditionText = $this->composeWhereCondition($text, $wifi, $likeBool, $whereCondiBool);
$conditionText = $conditionText . " END";
return $conditionText;
```

Ukázka kódu 42 – N1QL Text Service, Podmínka Wi-Fi MAC

Každá z těchto funkcí tedy poskládá část textu podmínky složené z předdefinované formulace a přijaté (filtrované) hodnoty.

Podmínka pro filtrování podle budovy bude složena v metodě *getBuldingCondition.* Dokumenty v Couchbase neobsahují atribut „building“, pouze „level“ se string hodnotou označující patro i budovu („J1NP“). Vyhledají se všechna označení pater patřících do budovy (*buildingLevels* – pole přijaté z presenteru), ze kterých se za použití logického výrazu OR vytvoří jedna podmínka (viz ukázka kódu 43).

```
$orOperator = false;
$conditionText = $this->addWhereOrAnd($whereCondiBool);
$conditionText = $conditionText . " (";
foreach ($buildingLevels as $key => $label)
{
	if($orOperator)
	{
		$conditionText = $conditionText . " OR ";
	}
	$conditionText = $conditionText . "level = '" . $label . "'";
	$orOperator = true;
}
$conditionText = $conditionText . ")";
return $conditionText;
```

Ukázka kódu 43 – N1QL Service, Podmínka název budovy

Aby bylo minimalizováno opakování kódu, byla vytvořena metoda *composeWhereCondition* (viz ukázka kódu 44), která přijímá parametry:

- ***text*** – část textu podmínky unikátní podle hledaného parametru („deviceID LIKE '%“, „level = “)
- ***var*** - filtrovaná hodnota
- ***likeBool*** – boolean proměnná, značící jestli se v podmínce používá výraz LIKE nebo rovnost
- ***whereCondiBool*** – bylo vysvětleno již dříve, rozhoduje mezi použitím WHERE nebo AND na začátku části podmínky

```
private function composeWhereCondition($text, $var, $likeBool, $whereCondiBool)
{
	$whereOrAnd = $this->addWhereOrAnd($whereCondiBool);
	if($likeBool)
	{
		$endText = "%'";
	} else {
		$endText = "'";
	}
	$conditionText = $whereOrAnd . $text . $var . $endText;
	return $conditionText;
}
```

Ukázka kódu 44 – N1QL Text Service, Složení podmínky podle jednoho parametru

Vrácenou hodnotou bude string složený z výrazu WHERE/AND, textu podle hledaného atributu, konkrétní filtrovací hodnoty a koncového textu (podle hodnoty *like-*

Bool se na konec textu přidá pouze koncová uvozovka nebo ještě navíc znak procenta). Přidávání WHERE a AND probíhá zavoláním metody *addWhereOrAnd* (viz ukázka kódu 45).

```
private function addWhereOrAnd($whereCondiBool)
{
        if(!$whereCondiBool) { $text = " WHERE";
        } else { $text = " AND"; }
        return $text;
}
```

Ukázka kódu 45 – N1QL Text Service, Vložení výrazu WHERE|AND

Návratovou hodnotou funkce *getTextForPointsFiltering* může být například text:

```
SELECT intX, intY, level, recalculatedX, recalculatedY FROM fingerprints
WHERE ANY m IN wifiScans SATISFIES m.mac LIKE '%c8:3a:35:46:05:00%' END
        AND ANY n IN bleScans SATISFIES n.address LIKE '%D2:33:17:B6:8F:E2%' END
        AND (level = 'J1NP' OR level = 'J2NP' OR level = 'J3NP' OR level = 'J4NP')
        AND couchbase_sync_gateway_id LIKE '%google-115285573410142095623%'
        AND deviceId LIKE '%359864063303115%'
        AND brand = 'Vodafone'
        AND createdAt BETWEEN '2010-04-06 06:30:52' AND '2017-04-12 09:45:52'
```

Ukázka kódu 46 – N1QL Text Service, Složený dotaz pro filtrování fingerprintů

Text dotazu pro nalezení nebližších bodů

GetTextForNearestPoints (viz ukázka kódu 47) má za úkol složit text dotazu pro nalezení nejbližších bodů po kliknutí do mapy s fingerprinty v případě, že k filtrování bodů zobrazených na mapě nebyl použit text poslaný z N1QL konzole (tzn., uživatel použil filtrovací formulář).

Přijatými proměnnými jsou stejně jako u metody *getTextForPointsFiltering* pole *activeFilters*, boolean *recalculation* určující, jestli se mají používat přepočítané souřadnice či nikoliv, a navíc x-ové a y-ové souřadnice kliknutí – *x*, *y*).

```
public function getTextForNearestPoints($activeFilters, $recalculation, $x, $y)
{
	$selectText = "SELECT meta(fingerprints).id, intX, intY, deviceId, wifiScans, bleScans,
			level, brand, couchbase_sync_gateway_id, createdAt";
	if($recalculation)
	{
		$selectText =  $selectText . ", recalculatedX, recalculatedY";
	}
	$selectText = $selectText . " FROM fingerprints";
	$conditionText = $this->composeConditionText($activeFilters);

	if($recalculation)
	{
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-recalculatedX),2)
				+ POWER((" .$y."-recalculatedY),2))) LIMIT 5";
	} else {
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-intX),2)
				+ POWER((" .$y."-intY),2))) LIMIT 5";
	}
	$queryText = $selectText . $conditionText . $orderCondition;
	return $queryText;
}
```

Ukázka kódu 47 – N1QL Text Service, Složení dotazu nejbližší body 1

Dotaz SELECT musí obsahovat pevný seznam atributů, které se mají vrátit v poli výsledků, protože ty se pak budou vypisovat jako informace o fingerprintu do „Informačního panelu“:

- ID dokumentu - *meta(fingerprints).id*
- souřadnice fingerprintu - *intX, intY* nebo *recalculatedX, recalculatedY*
- další informace – *deviceId, wifiScans, bleScans, level, brand, couchbase_sync_gateway_id* a *createdAt*

WHERE podmínka se získá stejným způsobem jako v *getTextForPointsFiltering* – voláním funkce *composeConditionText* s parametrem *activeFilters*, čímž se zajistí, že vyhledané dokumenty budou podmnožinou již zobrazených bodů. Získání pěti nejbližších bodů se dosáhne přidáním výrazů ORDER BY a LIMIT na konec dotazu.

Pro měření vzdálenosti mezi body se použije metoda výpočtu euklidovské vzdálenosti, tedy spočte se odmocnina ze součtu mocnin rozdílů souřadnic kliknutí a souřadnic v dokumentu (viz rovnice 1).

$$\sqrt{(klik_x - doc_x)^2 + \left(klik_y - doc_y\right)^2}$$

Rovnice 1 Vzdálenost bodů

Jak již bylo zmíněno, *getTextForNearestPoints* se použije, pokud byly body zobrazené na mapě získány přes filtrovací formulář. Pokud uživatel odeslat svůj dotaz přes konzoli, proběhne nalezení nejbližších fingerprintů a všech potřebných atributů dokumentů zavoláním dvou funkcí – *modifyUsedQueryText* a *getQueryForPointsById*.

Funkce *modifyUsedQueryText* (viz ukázka kódu 48) přijímá text dotazu, který použil uživatel, souřadnice x, y a boolean hodnotu podle níž se použijí originální nebo přepočítané souřadnice. Pomocí PHP funkce *str_ireplace* (není citlivá na velká a malá písmena) se v textu vyhledá řetězec „SELECT“ a zamění za „SELECT meta(fingerprints).id“, čímž se zajistí zahrnutí ID dokumentů do výsledku z couchbase, a příkaz ORDER BY sestavený identicky jako u *getTextForNearestPoints*. Výsledný string se pošle zpět jako návratová hodnota.

```
public function modifyUsedQueryText($usedN1qlText, $x, $y, $recalculation)
{
	$selectText = str_ireplace('SELECT', 'SELECT meta(fingerprints).id,', $usedN1qlText);
	if($recalculation)
	{
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-recalculatedX),2)
				+ POWER((" .$y."-recalculatedY),2))) LIMIT 5";
	} else
	{
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-intX),2)
				+ POWER((" .$y."-intY),2))) LIMIT 5";
	}
	$queryText = $selectText . $orderCondition;
	return  $queryText;
}
```

Ukázka kódu 48 – N1QL Text Service, Úprava uživatelova dotazu

GetQueryForPointsById (viz ukázka kódu 49) sestaví podle ID pěti dokumentů obsažených v poli pointsID, souřadnic x, y a informaci o potřebě přepočítaných souřadnic text dotazu pro nalezení pěti záznamů. Příkaz SELECT obsahuje výčet atributů potřebných pro zobrazení detailů o fingerprintu v „Informačním panelu“, obdobně jako u g*etTextForNearestPoints.* Podmínka WHERE nyní obsahuje jen přesné hodnoty ID dokumentů, které se mají vyhledat. Nakonec se přidá podmínka pro seřazení výsledků podle vzdálenosti od přijatých souřadnic.

```
public function getQueryForPointsById($pointsIds, $x, $y, $recalculation)
{
	$selectText = "SELECT meta(fingerprints).id, intX, intY, deviceId, wifiScans,
			bleScans, level, brand, couchbase_sync_gateway_id, createdAt";
	if($recalculation)
	{
		$selectText = $selectText . ", recalculatedX, recalculatedY";
	}
	$selectText = $selectText . " FROM fingerprints";
	$conditionText = " WHERE meta(fingerprints).id = '" . $pointsIds[0]
				. "' OR meta(fingerprints).id = '" . $pointsIds[1]
				. "' OR meta(fingerprints).id = '" .$pointsIds[2]
				. "' OR meta(fingerprints).id = '" . $pointsIds[3]
				. "' OR meta(fingerprints).id = '" . $pointsIds[4] . "'";
	if($recalculation)
	{
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-recalculatedX),2)
					+ POWER((" .$y."-recalculatedY),2)))";
	} else
	{
		$orderCondition = " ORDER BY (SQRT(POWER((". $x . "-intX),2)
						+ POWER((" .$y."-intY),2)))";
	}
	$queryText = $selectText . $conditionText . $orderCondition;
	return $queryText;
}
```

Ukázka kódu 49 – N1QL Text Service, Složení dotazu nejbližší body 2

Kontrola textu dotazu

Aby bylo uživateli zabráněno v provádění úprav v Couchbase, byla vytvořena funkce c*heckQueryText* (viz ukázka kódu 50), která se volá pokaždé, když uživatel odešle svůj text dotazu přes konzoli, a tento text zkontroluje. Vrácená hodnota TRUE povede k přerušení procesu zobrazení fingerprintů a k zobrazení chybové zprávy v šabloně (řešeno v presenteru). Jako potenciálně nebezpečné výrazy byly určeny "delete ", "update ", "create ", "drop ", "set ", "unset ". V textu se metoda *contains* pokusí pomocí PHP funkce *strpos* vyhledat každý z uvedených řetězců a pokud vrátí hodnotu TRUE, pak vrátí stejnou hodnotu i *checkQueryText*.

```
public function checkQueryText($queryText)
{
	$queryText = strtolower($queryText);
	$forbidenStrings = ["delete ", "update ", "create ", "drop ", "set ", "unset "];
	$ban = false;
	foreach ($forbidenStrings as $string)
	{
		$containString = $this->contains($string, $queryText);
		if($containString) { $ban = true; }
	}
	return $ban;
}
private function contains($needle, $haystack)
{
	return strpos($haystack, $needle) !== false;
}
```

Ukázka kódu 50 – N1QL Text Service, Kontrola zakázaných výrazů

4.3.3 View a Presentery

Každá webová stránka aplikace Fingerprints je rozdělena do tří částí:

1. Presenter – PHP funkce
2. Latte šablona – HTML kód
3. Javascriptový soubor – jQuery funkce

Zároveň mají všechny společnou HTML hlavičku dokumentu, obsaženou v souboru *layout.latte*, kde se také nastavují cesty k javascriptovým rozšířením, CSS stylům apod.

4.3.3.1 Base Presenter

Podobně jako je layout.latte základní šablonou, do které se pak přidávají ostatní šablony, tak je *basePresenter* základním presenterem, který pak rozšiřují ostatní presentery. Je to třída typu *abstract*, která je potomkem Nettovské třídy „Presenter“ a v této aplikaci bude použita pro zajištění seznamu všech implementovaných budov do navigačního menu, které se bude zobrazovat na všech vytvořených webových stránkách (viz ukázka kódu 51).

Seznam budov je uložen v konfiguračním souboru jako pole „menuBuildings“ a bude se získávat voláním funkce *getMenuBuildings* (bez parametrů) ve třídě *Config-*

ParametersService. Tuto třídu je potřeba injektovat vytvořením veřejné globální proměnné s anotací **@inject**. Aby byl seznam budov přístupný ve všech pohledech („view"), pošle se do šablon v metodě *beforeRender* (volá se před metodami *render*).

```
abstract class BasePresenter extends Nette\Application\UI\Presenter
{
	/** @var \App\Model\ConfigParametersService @inject */
	public $configParametersService;

	public function beforeRender()
	{
		parent::beforeRender();
		$this->template->menuBuildings = $this->configParametersService
							->getMenuBuildings();
	}
}
```

Ukázka kódu 51 – Base Presenter

4.3.3.2 Homepage - zobrazení fingerprintů

Hlavním účelem nástroje pro analýzu fingerprintů je znázornění fingerprintů do mapy a umožnit uživateli třídit je podle různých parametrů. To vše bude probíhat na hlavní stránce (Homepage).

Samotné vykreslování značek do mapy probíhá ve třídě *GraphicsService*. *HomepagePresenter* bude pouze zjišťovat, které body mají být zobrazeny. Přes třídu *CouchbaseService* získá pole fingerprintů z databáze, to odešle do *GraphicsService*, jenž vrátí cestu k vytvořenému obrázku, a následně je tato cesta poslána z presenteru do šablony, kde se použije v HTML elementu *img*.

Přibližování a posouvání mapy má na starosti jQuery plugin *Wheelzoom* (viz kapitola 3.2.4.1 *Wheelzoom*), který se na obrázek mapy naváže v javascriptovém souboru (viz níže část *Javascriptový soubor*).

Selekce bodů k vykreslení probíhá dvěma způsoby:

1. Uživatel vyplnil a odeslal formulář *filterData* (umístěný vpravo nahoře a stránce)
2. Uživatel napsal a poslal dotaz do Couchbase přes „konzoli" *filterN1ql* (v dolní části stránky)

Ad 1) Presenter podle přijatých dat poskládá za pomoci třídy *N1qlTextService* dotaz do Couchbase, který vrátí pole relevantních fingerprintů.

Ad 2) Dotaz se použije ve tvaru, který uživatel odeslal (pokud nebude obsahovat zakázané výrazy jako např. DELETE), ale před vykreslováním bodů bude potřeba zkontrolovat, jestli pole, které dotaz vrátil, obsahuje všechny potřebné informace.

Informace o jednotlivých fingerprintech (např. seznam zachycených Wi-Fi signálů) bude probíhat přes jQuery a ajaxovou funkci, která bude komunikovat s presenterem (viz *Javascriptový soubor*).

Presenter

Ve třídě HomepagePresenter bylo vytvořeno celkem jedenáct persistentních proměnných:

- *wifi* – Wi-Fi MAC adresa
- *ble* – Bluetooth MAC adresa
- *levelLabel* – označení podlaží v Couchbase (např. J1NP)
- *levelNumber* – číslo podlaží (např. 1)
- *user* – jméno uživatele, který vytvořil fingerprint
- *deviceId* – ID zařízení na kterém proběhlo měření signálů
- *brand* – značka zařízení
- *createdFrom* – dolní hranice času vytvoření fingerprintů
- *createdTo* – horní hranice
- *building* – aktuálně vybraná budova
- *usedN1qlText* – text N1ql dotazu, který odeslal uživatel

Hodnoty uložené v těchto proměnných se budou předávat automaticky, čehož se využije při odesílání formulářů *filterData* a *filterN1ql* (viz část *Formuláře*) nebo vytváření odkazů vedoucích na tuto stránku s již přednastaveným filtrem (viz kapitola 4.3.3.4 *Document – JSON Dokumenty*). Persistentní proměnná se definuje pomocí anotace **@persistent** (viz ukázka kódu 52).

```
/** @persistent */
    public $wifi;
    ...
```

Ukázka kódu 52 – Homepage Presenter, Definice persistentní proměnné

HomepagePresenter bude využívat funkce všech pěti tříd vytvořených v modelu - *ConfigParametersService*, *CouchbaseService*, *MysqlDatabaseService*, *GraphicsService* a *N1qlTextService*. Závislosti se předají do proměnných s anotací **@inject** (viz ukázka kódu 53).

```
/** @var \App\Model\ConfigParametersService @inject */
public $configParametersService;
…
```

Ukázka kódu 53 – Homepage Presenter, injektování služeb

Poslední dvě proměnné jsou *recalculation* a *displayedQueryText*. Hodnota v *recalculation* určuje, jestli je potřeba používat přepočítávané souřadnice a do *displayedQueryText* se uloží text dotazu, který se poslal do Couchbase (ať už přes filtrovací formulář nebo přes konzoli) a zobrazí se v konzoli dole na stránce.

Renderovací funkce

Akce, které se mají provést při příchodu na Homepage jsou obsaženy v metodě *renderDefault* (viz ukázka kódu 54). Nejprve se ve složce s vytvořenými obrázky smažou soubory starší než je limit definovaný v konfiguračním souboru zavoláním metody *deleteTemps* a nastaví se název budovy (do persistentní proměnné *building*) přes funkci *setDefaultBuilding*.

V šabloně se vytvoří proměnné *incorrectQuery* a *emptyResults*, které budou sloužit pro zobrazení informace (v šabloně) o chybě v textu N1QL dotazu nebo při vrácení prázdného pole fingerprintů z Couchbase.

Lokální proměnná *banQueryText* s výchozí hodnotou FALSE značí přítomnost potencionálně nebezpečných výrazů v textu dotazu vytvořeným uživatelem (DELETE apod.).

Persistentní proměnná *usedN1qlText* obsahuje text přijatý po odeslání formuláře *filterN1ql*. Podle podmínky IF se algoritmus rozděluje do dvou větví:

1. Hodnota *usedN1qlText* není NULL -> uživatel použil konzoli:

   Po kontrole textu na zakázané výrazy se hodnota z persistentní proměnné *usedN1qlText* uloží do lokální proměnné *queryText*, která se použije jako parametr volání při hledání fingerprintů v Couchbase, do globální proměnné *displayedQueryText* (použije se při vytváření formuláře *filterN1ql* jako výcho-

zí hodnota v textovém poli) a do lokální proměnné *usedN1qlText* se uloží text dotazu jako escapovaný textový řetězec (escapování proběhne zavoláním metody *escapeQuotes*).

2. Hodnota usedN1qlText je NULL -> uživatel použil filtrovací formulář:

   Formulář *filterData* vrátil uživatelem vyplněné hodnoty formuláře a uložil je do persistentních proměnných třídy, které se použijí při skládání dotazu ve třídě *N1qlTextService*.

   Pomocí *setGlobalVarsByFilter* se zjistí, jestli je na uživatelem zvolené mapě (přes formulář *filterData*) potřeba používat přepočítané souřadnice či nikoliv a zároveň zjistí číslo patra.

   GetActiveFilters vrátí asociativní pole s hodnotami vyplněnými ve formuláři a dalšími potřebnými informacemi, které se následně použijí při volání metody služby v *N1qlTextService getTextForPointsFiltering*. Vrácený textový řetězec bude uložen do *queryText* a lokální proměnná *usedN1qlText* se nastaví na NULL (toho bude využito v šabloně, resp. v javascriptovém souboru při hledání nejbližších bodů, viz níže část Javascriptový soubor).

Po skončení podmínky je v každém případě v proměnné *queryText* uložen textový řetězec, který se má použít pro vyhledání fingerprintů v Couchbase.

Pokud byl text dotazu shledán nevhodným, tedy hodnota proměnné *banQueryText* je TRUE, je zavolána metoda *handleQueryBan*, která zajistí zobrazení chybové hlášky uživateli. V opačném případě je zavolána metoda *proceedPictureDisplay*. Poslední funkcí zavolanou v *renderDefault* je *setTemplateVariables*, která nastaví proměnné v šabloně na potřebné hodnoty k zobrazení detailů o fingerprintech kliknutím do mapy a zjišťování polohy kurzoru na mapě.

```
public function renderDefault()
{
	$this->deleteTemps();
	$this->setDefaultBuilding();
	$this->template->incorrectQuery = false;
	$this->template->emptyResult = false;
	$banQueryText = false;
	if($this->usedN1qlText)
	{
		$queryText = $this->usedN1qlText;
		$banQueryText = $this->n1qlTextService->checkQueryText($queryText);
		$this->displayedQueryText = $this->usedN1qlText;
		$usedN1qlText = $this->escapeQuotes($this->usedN1qlText);
	} else
	{
		$this->setGlobalVarsByFilter();
		$activeFilters = $this->getActiveFilters();
		$queryText = $this->n1qlTextService
			->getTextForPointsFiltering($activeFilters, $this->recalculation);
		$this->displayedQueryText = $queryText;
		$usedN1qlText = null;
	}
	switch ($banQueryText)
	{
		case false:
			$this->proceedPictureDisplay($queryText);
			break;
		case true:
			$this->handleQueryBan();
			break;
	}
	$this->setTemplateVariables($usedN1qlText, $level_label);
}
```

Ukázka kódu 54 – Homepage Presenter, Renderovací metoda

V *ProceedPictureDisplay* s parametrem volání *queryText* obsahujícím text dotazu pro vyhledání záznamů se rozhoduje, jestli se použije již existující obrázek mapy nebo jestli se vytvoří nový. Hledání již vytvořeného obrázku proběhne voláním funkce v *GraphicsService checkForCache* (viz kapitola 4.3.2.4 Graphics Service, ukázka kódu 36). Parametry volání jsou text dotazu, informace, jestli se k odeslání použila konzole („yes“ nebo „no“) a filtrovaná MAC adresa (relevantní pouze při filtrování přes konzoli). Pokud se obrázek nenašel, pokračuje se voláním metody *selectFingerprints*. Pokud ale ano, naplní se hodnota proměnné picturePath v šabloně cestou k tomuto souboru (cesta se nalezne voláním funkce v *GraphicsService getPicturePath*).

V případech, kdy se k filtrování bodů použil formulář filterN1ql (konzole), bude potřeba zjistit, která mapa podlaží se použila, aby bylo možné zjišťovat souřadnice na mapě (problematika vysvětlena v kapitole 3.1.5 *Filtrování fingerprintů* v části „N1QL konzole“). Použitým řešením je získání názvu podlaží ze souboru, který byl vytvořen zároveň s obrázkem, zavoláním funkce v *GraphicsService getLevelLabelFromFile* s parametrem volání *queryText* (text dotazu).

Jestliže se nenašel vyhovující „cachovaný“ obrázek, pokračuje se zavoláním metody *selectFingerprints*, která přijímá text dotazu. Proběhne vyhledání dat v Couchbase, ošetření případných problémů (vrátilo se prázdné pole nebo výjimka vzniklá chybou v dotazu) a kontrola vráceného pole, jestli obsahuje všechny potřebné informace (souřadnice, označení podlaží apod.).

Fingerprinty se z databáze získají zavoláním metody v *CouchbaseService getData*, které se předá text dotazu *queryText* a metoda vrátí pole nalezených dokumentů. Pokud byl použit dotaz odeslaný z konzole (persistentní proměnná *usedN1qlText* není prázdná), překontrolují se obdržené záznamy v metodě *checkRows*, která přijímá pole s fingerprinty a v případě nesprávného tvaru (např. chybějící souřadnice) vrací pole s dvojicemi „correct – FALSE“ a „error – string určující typ chyby“. Podle typu chyby se v šabloně zobrazí chybová hláška, která uživateli napoví, jak upravit text dotazu pro úspěšné zobrazení bodů. Pokud záznamy obsahují vše potřebné, je hodnota „correct“ ve vráceném poli TRUE. Během procházení záznamů se také kontroluje atribut *level* u každého z nich a podle toho se vybere prázdná mapa, do které se fingerprinty zobrazí. Například pokud každý vrácený záznam má hodnotu atributu level rovnu „J3NP“, použije se mapa třetího patra pro budovu „J“ apod. Název patra se vrátí v poli pod klíčem *level* spolu s dvojicí „correct – TRUE“. V případě, že bylo nalezeno vícero hodnot (fingerprinty jsou z různých pater), hodnota *level* ve vráceném poli je NULL. Nalezený název se uloží do persistentní proměnné *levelLabel* a následně se nalezne číslo tohoto podlaží zavoláním metody v *MysqlDatabaseService getLevelNumber*. Pokud se text dotazu získal přes filtrovací formulář *filterData*, je číslo podlaží obsaženo v persistentní proměnné *levelNumber*. Pokračuje se voláním metody *createNewPicture*.

CreateNewPicture přijímá pole záznamů z Couchbase (*rows*), číslo podlaží (*level*) a text dotazu (*queryText*). Před vytvořením nového obrázku proběhne kontrola velikosti složky, kam se nově vytvořené soubory ukládají, voláním metody *preventOverfilling* a případně se vymaže několik nejstarších souborů. Název mapy, do které se budou zakreslovat body, se nalezne v metodě *getEmptyMapName*, která přijímá číslo

patra (může být i NULL) a vrací název souboru s prázdnou mapou (získaný z MySQL databáze). Zde je nastaveno, že pokud se zobrazují fingerprinty z několika poschodí zaráz, budou se vždy zakreslovat do mapy prvního patra. Pomocí metody v *MysqlDatabeseService getAllLevelNumbersAndLevelLabels* se získá pole všech pater v budově a jejich označení v Couchbase. To se následně použije při volání metody *getColorsForPoints*, která následně vrátí pole s barvami, které se použijí pro obarvení bodů jednotlivých pater. V případě, že jedna z persistentních proměnných *wifi* nebo *ble* není prázdná, vrací metoda hodnotu NULL, což bude později značit, že se k obarvení bodů má použít barva vypočítaná podle síly signálu dané MAC adresy. Nakreslení nového obrázku proběhne ve funkci *GraphicsService drawPicture*. Metoda přijímá označení patra, název souboru s prázdnou mapou, pole fingerprintů, text dotazu, hodnoty persistentních proměnných *wifi* a *ble*, pole s dvojicemi „číslo podlaží – označení v Couchbase“, pole s barvami a informaci o nutnosti použití přepočítaných souřadnic (hodnota persistentní proměnné *recalculation*). *DrawPicture* vrací cestu k vytvořenému obrázku, která se uloží do proměnné *picturePath* v šabloně.

Nastavení proměnných šabloně

SetTemplateVariables nastavuje hodnoty proměnných v šabloně. Přijímá parametr usedN1qlText s hodnotou NULL, pokud byl k filtrování fingerprintů použit formulář, nebo escapovaný text dotazu do Couchbase, pokud byla použita konzole.

Většina proměnných předaných do šablony v této metodě bude využita při akci zobrazování detailů o fingerprintech po kliknutí do mapy. Předají se hodnoty uložené v persistentních proměnných, které souvisí s formuláře *filterData*, přijatý text dotazu, nalezené aliasy pro filtrované hodnoty (zavoláním metody setAliasesForForm), informace o obrázku mapy s fingerprinty (výška, šířka, případně posunutí a přiblížené oproti mapě prvního patra) apod.

Chybové hlášky v šabloně

HandleCouchbaseException, *handleQueryBan, handleEmptyCouchbaseResult* a *handleIncorrectArrayError* jsou metody volané v případě problému s výsledkem vráceného z Couchbase. Každá z těchto funkcí vede k vypsání zprávy s informací o problému, která se v šabloně zobrazí uživateli. Při vrácení prázdného pole z Couchbase je potřeba poslat do šablony i cestu k souboru s prázdnou mapou získanou voláním metody *getEmptyMapName* a metody v *GraphicsService getEmptyMapPath*. *HandleIncorrectArrayError* se má pokusit uživateli napovědět, jaké chyby se v textu dotazu dopustil, což se zjišťuje podle hodnoty přijaté v parametru *error* (viz ukázka kódu 55).

```
$this->template->incorrectQuery = true;
$text1 = "Problem with returned data occured. SELECT statement must contain";
$text2 = "None of the returned documents contains array with signals. Make sure SELECT
statement contains";
switch ($error)
{
        case "level":
                $missing = " level";
                $errorText = $text1 . $missing;
                break;
        case "coordinates":
                $missing = " coordinates - intX and intY";
                $errorText = $text1 . $missing;
                break;
        …
}
$this->template->errorText = $errorText;
```

Ukázka kódu 55 – Homepage Presenter, Chybové hlášky

Funkce pro mazání generovaných souborů

Vygenerované mapy a soubory s texty dotazů budou smazány, pokud jejich stáří přesahuje hodnotu uloženou v proměnné *cacheFileExpirationTime* v konfiguračním souboru, nebo pokud velikost všech souborů ve složce, kam se ukládají generované soubory, přesáhla limit definovaný v parametru *imageFolderLimit* (opět v konfiguračním souboru).

Vymazání starých souborů probíhá v metodě *deleteTemps*. Ze služby *ConfigParametersService* se voláním metod *getCacheFileExpirationTime* a *getCacheFolderPath* získají potřebné údaje, PHP funkcí *glob* se načtou všechny soubory ve složce a v cyklu FOREACH se kontroluje, jestli stáří souboru (*time* - *filemtime*) nepřesahuje limit. Pokud ano, soubor se smaže pomocí *unlink*.

Metoda *preventOverfilling* zajišťující kontrolu velikosti složky s generovanými soubory se volá vždy před vytvořeným nového obrázku. Pro zjištění velikosti složky byla vytvořena metoda *getFolderSize*, která přijímá argument s cestou k souboru, sečte velikost všech obsažených souborů, převede zjištěné číslo v bajtech na megabajty a to pak odešle zpět jako návratovou hodnotu.

Pokud je velikost složky v pořádku, pokračuje se k vytvoření nového obrázku a textového souboru s dotazem. Pokud je ale velikost složky větší než povolený limit, voláním metody *deleteOldestFiles* se v ní vymaže x nejstarších souborů. *deleteOldest-*

Files přijímá dva argumenty – cestu ke složce a počet souborů, které se mají smazat. Soubory jsou nejprve seřazeny od nejstaršího po největší (*usort* na základě *filetime*) a následně se prvních x souborů vymaže (*unlink*).

V *deleteTemps* se použitím PHP funkce *glob* najdou všechny soubory ve složce vygenerovaných map a souborů s N1QL dotazem a následně se v cyklu foreach kontroluje, jestli je stáří dokumentu vyšší než hodnota definovaná v konfiguračním souboru. Pokud ano, soubor se smaže přes *unlink*.

```
$files = glob($cacheFolderPath . '*');
foreach($files as $file)
{
	if(is_file($file) && ((time() - filemtime($file)) > $cacheFileExpirationTime))
	{
		unlink($file);
	}
}
```

Ukázka kódu 56 – Homepage Presenter, Smazání „cachovaných" souborů

Formuláře

Formuláře jsou v Nette Frameworku vytvářeny pomocí tzv. „továrniček". Jednotlivé komponenty se vytvářejí v presenteru ve funkci „createComponent" a v šabloně se pak zobrazují použitím makra (např. {**control** editForm}). Pro každý formulář jsou v presenteru definovány dvě funkce – jedna pro vytvoření a druhá, která se volá po odeslání formuláře („formSucceeded").

První formulář *filterData* se vytváří v metodě *createComponentFilterData* (viz ukázka kódu 57), která nepřijímá žádné parametry a vrací komponentu formuláře. Nejprve se do proměnné *form* vytvoří nová instance nettovské třídy „Form". Textový input se do formuláře přidá metodou *addText*, přičemž parametry volání jsou název inputu (např. „wifi") a popisek (např. „Wi-Fi address: ").

Lze přidat CSS atributy *id* a *class* přes metody *setAttribute*. U textových polí bude v jQuery přidán našeptávač nebo DateTime Picker navázáním funkce autocomplete na input s určitým id atributem.

SetDefaultValue nastaví výchozí hodnotu inputu (rovna hodnotě v persistentní proměnné), což může být NULL nebo hodnota aktuálně použitého filtru, vytvořeného předchozím odesláním formuláře *filterData* nebo přesměrováním stránky odkazem

(např. ze stránky Document, kliknutím na kterýkoliv odkaz v zobrazeném stromě – kapitola 4.3.3.4 *Document – JSON Dokumenty*).

Metoda *addSelect* vyžaduje jako parametry volání kromě názvu inputu a popisu ještě pole dvojic „klíč – hodnota“, kde hodnoty budou nabízeny v inputu a klíč patřící k vybrané hodnotě se po odeslání formuláře přijme v metodě *succeed*.

Pole s existujícími značkami zařízení v Couchbase se získá voláním funkce *getBrandsArray*, která nepřijímá žádné parametry a vrací pole již upravené do požadovaného formátu.

U výběru patra se budou uživateli zobrazovat čísla podlaží aktuální budovy od nejnižšího po nejvyšší, přičemž klíč bude stejný jako hodnota (např. 1 => 1). Budovy mohou mít různý počet pater, proto se tento seznam získá z MySQL databáze voláním funkce *getAllLevels* s parametrem obsahujícím název aktuálně vybrané budovy. Na začátek vráceného pole se připojí dvojice 'all' => 'All' pro přidání možnosti zobrazení fingerprintů ze všech pater najednou.

AddSubmit přidá tlačítko pro odeslání formuláře a nakonec se do handleru události *onSuccess* nadefinuje, která metoda se má zavolat po odeslání formuláře (zde *filterDataFormSucceeded*).

```
$form = new Nette\Application\UI\Form;
...
$form->addSelect('level', 'Level: ', $levels)
        ->setAttribute('id', 'levelInput')
        ->setAttribute('class', 'levelLabel form-control selectInput')
        ->setDefaultValue($level);
$form->addSubmit('send', 'Filter');
$form->onSuccess[] = $this->filterDataFormSucceeded;
return $form;
```

Ukázka kódu 57 – Homepage Presenter, Formulář *filterData*

FilterDataFormSucceeded se, jak již bylo zmíněno dříve, volá automaticky po odeslání formuláře. Vyplněné údaje se uloží do persistentních proměnných (viz ukázka kódu 58), podle kterých se po znovunačtení stránky se v renderovací metodě poskládá N1QL dotaz pro vyhledání vyhovujících fingerprintů.

Odeslané hodnoty se získají pomocí Nette funkce *getValues*, která vrátí asociativní pole, kde klíčem je název inputu a hodnotou je vstup vyplněný uživatelem. Tato získaná hodnota může být buď shodná s hodnotou z Couchbase, nebo jde o alias ulo-

žený v MySQL databázi, který je potřeba převést zpět na originál. K tomu byla vytvořena metoda *findPairAliasOriginal* s parametry *alias* a *type*, která vrací buď dvojici „original - alias“ nebo NULL v případě, že nebyla nalezena shoda.

```
$wifi = $values['wifi'];
$aliasWifi = $this->mysqlDatabaseService->findPairAliasOriginal($wifi, 'wifi');
if($aliasWifi != null)
{
	$this->wifi = $aliasWifi[$wifi];
} else {
	$this->wifi = $wifi;
}
```

Ukázka kódu 58 – Homepage Presenter, Nastavení persistentních proměnných 1

Persistentní proměnná *usedN1qlText* bude vždy v metodě f*ilterDataFormSucceeded* nastavena na NULL, čímž bude naznačeno, že uživatel nepoužil vlastní dotaz, ale filtrovací formulář.

Zvláštní případ je způsob filtrování patra (viz ukázka kódu 59). Váží se k němu dvě persistentní proměnné – *levelNumber* a *levelLabel*. Hodnota *levelNumber* bude vždy shodná s hodnotou vrácenou z formuláře. To může být buď vybrané číslo nebo textový řetězec „all“. Pokud je to číslo, vyhledá se označení v couchbase přes funkci služby *MysqlDatabaseService getCouchbaseLabelForLevel* s parametry volání číslo podlaží a název budovy a nalezený řetězec se uloží do *levelLabel*. V případě „all“ se *levelLabel* nastaví na NULL, což bude při vytváření obrázku ukazatel, že se má použít mapa prvního patra a případně přepočítané souřadnice.

```
if($values['level'] == 'all')
{
	$this->levelLabel = null;
} else {
	$this->levelLabel = $this->mysqlDatabaseService
		->getCouchbaseLabelForLevel($values['level'], $this->building);
}
$this->levelNumber = $values['level'];
```

Ukázka kódu 59 – Homepage Presenter, Nastavení persistentních proměnných 2

Na konci metody *filterDataFormSucceeded* se provede znovunačtení stránky (tzn., spustí se funkce *renderDefault* s nastavenými persistentními proměnnými).

Metody *createComponentFilterN1ql* a *filterN1qlFormSucceeded* fungují podobně jako předchozí dvě metody. *FilterN1ql* bude obsahovat jeden input typu „text area“,

kde se bude jako výchozí hodnota zobrazovat text dotazu použitý pro aktuálně zobrazené fingerprinty, textový input pro zadání MAC adresy, dva radio buttony („wifi“ a „ble“), *checkbox*, který bude indikovat, že se mají vykreslené body obarvit podle síly signálu a tlačítko pro odeslání. Název a typ MAC adresy bude vyplněn jako výchozí hodnota, pokud právě jedna z persistentních proměnných *wifi* nebo *ble* není rovna NULL.

Po odeslání se persistentní proměnné *levelNumber*, *levelLabel*, *user*, *deviceId*, *brand*, *createdFrom* a *createdTo* nastaví na NULL zavoláním metody *setPersistentVarsToNull*. Text dotazu uživatele se uloží do persistentní proměnné *usedN1qlText*. Pokud byl zaškrtnut checkbox *signal*, vyplněn název MAC adresy a vybrán typ (*wifi* nebo *ble*), uloží se originální název adresy do persistentní proměnné wifi nebo ble (podle vyplněného typu), a hodnota druhé persistentní proměnné se nastaví na NULL. V opačném případě se nastaví na hodnotu NULL obě dvě persistentní proměnné – *wifi* i *ble*.

Funkce pro obsluhu AJAXu

HandleGetInfo se zavolá po kliknutí do obrázku s mapou. Obdržená data (odeslaná metodou *POST*) obsahují souřadnice kliknutí, text dotazu (NULL pokud byla k filtrování použita konzole), hodnoty odeslané přes formulář *filterData* a název budovy.

Algoritmus se dělí do dvou větví:

1. uživatel použil k filtrování fingerprintů, které jsou zobrazeny na mapě, formulář *filterData*
2. uživatel odeslal dotaz přes konzoli (formulář *filterN1ql*)

Ad 1) Filtrované hodnoty (předány do šablony v presenteru v metodě *setTemplateVariables*) se uloží do pole *activeFilters*, které se spolu s informací o nutnosti použití přepočítaných souřadnic a souřadnicemi kliknutí do mapy odešlou do funkce třídy *N1qlTextService getTextForNearestPoints*, která vrátí text dotazu pro nalezení pěti nejbližších bodů. Odesláním dotazu do Couchbase se získá pole s pěti fingerprinty, které se ještě upraví v metodě *prepareData* (přidání aliasů, vypočítání vzdálenosti bodu od místa kliknutí apod.) a následně se pošle zpět jako odpověď.

Ad 2) Nejprve se zjistí, jestli je potřeba použít přepočítané souřadnice či nikoliv zavoláním funkce *decideRecalculation*, která přijímá parametry název budovy a označení patra (obojí získáno z šablony metodou POST). Funkce vrací TRUE, pokud je název budovy v seznamu budov vyžadujících přepočítávání souřadnic (uloženo v konfi-

guračním souboru) a zároveň filtrované podlaží je rovno NULL nebo „all“ (zobrazují se fingerprinty z více pater najednou do mapy přízemí), nebo FALSE. Vrácená hodnota se uloží do proměnné *recalculation*.

Přes funkci v *N1qlTextSevice modifyUsedQueryText* s parametry textu dotazu použitého pro filtrování, souřadnicemi kliknutí a *recalculation* se sestaví dotaz pro nalezení ID pěti dokumentů obsahujících souřadnice, které jsou nejblíže souřadnicím kliknutí. Následně se opět zavolá metoda v *N1qlTextService*, tentokrát *getQueryForPointsById*. Ta přijímá pole s identifikátory dokumentů, které se mají najít, souřadnice kliknutí (použijí se pro seřazení dokumentů od nejbližšího po nejvzdálenější) a *recalculation*. Získaný text dotazu se použije pro dotaz do Couchbase. Tímto postupem se zaručí, že vrácené pole záznamů bude obsahovat všechny potřebné atributy (*wifiScans*, *bleScans*, *deviceId*, *brand*… atd.). Pole se upraví v metodě *prepareData* obdobně jako v předchozím případě a odešle se zpět do šablony.

Šablona

Veškerý HTML obsah bude vložen do bloku *content* a pod ním do, bloku "javascript", se vloží javascriptový kód ze souboru, který se nachází ve stejné složce jako šablona, použitím makra *include* (viz ukázka kódu 60). Stejným způsobem jsou organizovány i ostatní šablony.

```
{block content}
        …
{/block}
{block javascript}
        {include 'javascript.js'}
{/block}
```

Ukázka kódu 60 – Homepage Latte

Layout stránky je rozdělen do čtyř částí:

- navigace
- mapa s fingerprinty
- pravý panel pro filtrování a zobrazení informací o fingerprintech
- dolní N1QL konzole pro odesílání dotazů

Panel s navigací (umístěn v horní části stránky) byl vytvořen pomocí šablony frameworku Bootstrap (jednotlivým HTML elementům byly přidány příslušné třídy, čímž se docílilo požadovaného CSS stylování). Stránka Homepage je v menu obsažena

pod jednou z možností „dropdown“ položky „Map“, jenž nabízí seznam implementovaných budov (viz ukázka kódu 61). Seznam budov byl do šablony předán ve třídě *BasePresenter* (viz kapitola 4.3.3.1 *Base Presenter*). Při procházení pole se ke každé dvojici „klíč - hodnota“ vytvoří odkaz vedoucí na Homepage s přednastaveným filtrem, čehož se docílí nastavením persistentních proměnných na NULL kromě proměnné building, která bude mít hodnotu klíče položky pole *menuBuildings*. Takto bude zajištěno, že filtrovací formulář po kliknutí na odkaz a nahrání stránky prázdný, a zároveň je tímto způsobem umožněna změna výběru aktuální budovy (v presenteru se podle hodnoty v persistentní proměnné building určují názvy map, do kterých se vykreslují fingerprinty, text dotazu do Couchbase se automaticky omezí na nalezení fingerprintů naměřených v této budově a do *select* inputu ve formuláři se dynamicky dosadí čísla pater budovy).

```
{foreach $menuBuildings as $key => $value}
        <li><a href="{link Homepage:default, 'building' => $key, 'usedN1qlText' => null,
                        'wifi' => null, 'ble' => null, 'levelLabel' => null, 'levelNumber' => null,
                        'user' => null, 'deviceId' => null, 'brand' => null, 'createdFrom' => null,
                        'createdTo' => null}">
                {$value}</a></li>
{/foreach}
```

Ukázka kódu 61 – Homepage Latte, Výběr budovy v navigaci

Další položkou menu je odkaz na stránku s aliasy. Ten se vygeneruje pomocí *plink*, což je zkrácený název pro metodu „$presenter->link“.

```
<li><a href='{plink //Aliases:default}'>Aliases</a></li>
```

Ukázka kódu 62 – Homepage Latte, Generování odkazů

Odkaz na stránku se stromem dokumentu v Couchbase se do menu přidávat nebude. Uživatel se na stránku dostane kliknutím na ID dokumentu v „Informačním panelu“, které je potřeba k nalezení záznamu v databázi (stránka Document vždy zobrazuje jeden konkrétní záznam).

HTML element div s názvem atributu *class* „rightDiv“obsahuje:

1. panel pro zobrazování aktuálních souřadnic kurzoru na mapě
2. filtrovací formulář
3. „informační panel“, kam se budou vypisovat informace o nejbližších bodech po kliknutí do mapy

Ad 1) Atribut ID *divu* je pojmenován „coordinates“, v JavaScriptu na něj bude navázána funkce pro zjištění a zobrazení souřadnic.

Ad 2) Samotný formulář byl vytvořen v presenteru v metodě *createComponentFilterData*. Do šablony může být celý vygenerován jednoduchým makrem *{control filterData}*, ale kvůli potřebě přidání aliasů existujících k případným předvyplněným hodnotám v inputech a také větší svobodě při CSS stylování budou komponenty formuláře přidávány jednotlivě. Stylování bude vytvořeno použitím Bootstrapu a to obalením elementů formuláře do vlastních divů a přiřazením požadovaných tříd těmto divům, formuláři a jeho prvkům. Začátek a konec je ohraničen tagy *form*, kde název formuláře je „filterData“ (viz ukázka kódu 63).

```
{form filterData class => "form-horizontal"}
        …
{/form}
```

Ukázka kódu 63 – Homepage Latte, Formulář

Každý prvek, například textový input *wifi*, obsahuje popis a vstup. Ukázka kódu 64 popisuje přidání inputu *wifi* a symbolického názvu pro případnou vyplněnou hodnotu do formuláře (existující symbolické názvy k vyplněným hodnotám se zobrazí pod textovými inputy po odeslání formuláře).

```
<div class="form-group">
        <div class="col-sm-5">{label wifi class => "label" /}</div>
        <div class="col-sm-7">{input wifi}</div>
</div>
<div class="form-group">
        <div class="col-sm-5"></div>
        <div class="col-sm-7 aliasFormDescription">
                {if $wifiAlias != null}{$wifiAlias}{/if}
        </div>
</div>
```

Ukázka kódu 64 – Homepage Latte, Vstup ve formuláři

Ostatní položky filtru jsou přidávány obdobně. Na konci je přidáno tlačítko pro odeslání a navíc odkaz pro resetování formuláře (resp. vymazání všech hodnot v inputech, řešeno v JavaScriptu).

Ad 3) „Informační panel“ je po nahrání stránky prázdný, informace o fingerprintech se sem vkládají dynamicky přes JavaScript. Funkce je navázána na HTML element *div* s ID *#information.* Pro vyšší uživatelskou přívětivost byla přidána hlavička

s popisem a nápovědou a vzhledem k tomu, že je panel relativně malý, bude mu v JavaScriptu přidána možnost změny umístění a velikosti.

Ad 4) „N1QL konzole“ je ve skutečnosti HTML div obsahující formulář s nastylovaným textovým polem, checkboxem, skrytou částí obsahující vstup pro název MAC adresy a jejího typu (Wi-Fi nebo Bluetooth) a tlačítkem pro odeslání. Možnost odeslání přes stisknutí klávesy enter a zobrazování a schovávání skryté části formuláře přes zaškrtávání checkboxu bude přidána v JavaScriptu.

Mapa s fingerprinty se zobrazuje v HTML elementu div *#imgDiv*, přičemž cesta k souboru byla do šablony poslána z presenteru (viz ukázka kódu). Pokud během vyhledávání a zakreslování bodů nastal nějaký problém (například chybějící souřadnice v příkazu SELECT apod.), bude místo obrázku vložena chybová hláška informující uživatele o problému.

```
<div id="imgdiv">
	{if $emptyResult}
		<div class="alert alert-info">
			<strong>Info</strong> {$infoText}
		</div>
	{/if}
	{if !$incorrectQuery}
		<img id="image" class='zoom' src={$picturePath}>
	{else}
		<div class="alert alert-danger homepageAllert">
			<strong>Error!</strong> {$errorText}
		</div>
	{/if}
</div>
```

Ukázka kódu 65 – Homepage Latte, Zobrazení mapy fingerprintů

Javascriptový soubor

Přes JavaScript se na stránce Homepage mj. přibližuje, oddaluje a posouvá mapa (použitím pluginu Wheelzoom) nebo se zobrazují detailní informace o fingrerprintech.

Navázání rozšíření *Wheelzoom* na obrázek mapy je k vidění v ukázce kódu 66, přičemž byl kód tohoto nástroje pozměněn, aby vracel souřadnice kliknutí – v souboru *wheelzoom.js* byla upravena funkce *onmouseup*. Výsledný kód je uveden v ukázce kódu 67.

```
wheelzoom(document.querySelector( 'img.zoom' ), { onmouseup:onmouseup });
```

Ukázka kódu 66 – Homepage JavaScript, navázání Whellzoom na obrázek

```
function onmouseup(e) {
                newPos = [e.pageX, e.pageY];
                var isEqual = compareArrays(newPos, currentPos);
                 if(isEqual) {
                        var rect = img.getBoundingClientRect();
                        var offsetX = e.pageX - rect.left - window.pageXOffset;
                        var offsetY = e.pageY - rect.top - window.pageYOffset;
                         if(settings.onmouseup) {
                         settings.onmouseup(offsetX, offsetY);
                         }
                }
        }
```

Ukázka kódu 67 – Wheelzoom.js, úprava funkce *onmouseup*

Při přejíždění kurzorem myši přes mapu fingerprintů se budou v HTML *divu #coordinates* zobrazovat pixelové souřadnice. Na element s ID *#image* se naváže jQuery funkce *.mousemove*. Při každém pohybu se vyprázdní *div #coordinates* kam se pak vloží nové údaje.

Souřadnice na aktuálně zobrazené mapě se vypočítají jako rozdíl *.pageX*(nebo *Y*) a *.offsetLeft*(nebo *Top*). Jelikož však může být obrázek přiblížený, oddálený nebo i různě posunutý (obvykle použitím rozšíření Wheelzoom), byla vytvořena funkce pro přepočítání *getCurrentCoordinates* (viz ukázka kódu 68), která tyto možnosti zohlední, podle přijatých souřadnic vypočítá a vrátí skutečné souřadnice na obrázku v původním stavu. Pomocí jQuery funkce *.css('background-size')* se zjistí aktuální výška a šířka obrázku (nástroj *Wheelzoom* převedl element *img* na obrázek pozadí elementu div *#image*). Posunutí mapy po x-ové a y-ové ose se zjistí přes hodnoty CSS vlastnosti *background-position*. Vydělením šířky pozadí hodnotou získanou z presenteru a uloženou v proměnné *mapWidth*, která obsahuje skutečnou šířku obrázku na serveru, se vypočítá zvětšení na x-ové ose (obdobně y-ová osa přes výšku pozadí a hodnotu v proměnné *mapHeight*). Následně se souřadnice zvětšené o hodnotu posunutí a vydělené hodnotou přiblížení vrátí zpět jako návratová hodnota.

```
function getCurrentCoordinates(x1, y1) {
        var bgSize = $('#image').css('background-size').split(" ");
        var width = bgSize[0].replace(/[^-\d\.]/g, '');
        var height = bgSize[1].replace(/[^-\d\.]/g, '');
        var bgPosition = $('#image').css('background-position').split(" ");
        var bgPositionX = bgPosition[0].replace(/[^-\d\.]/g, '');
        var bgPositionY = bgPosition[1].replace(/[^-\d\.]/g, '');
        bgPositionX = Math.abs(bgPositionX);
        bgPositionY = Math.abs(bgPositionY);
        var shiftedX = x1 + bgPositionX;
        var shiftedY = y1 + bgPositionY;
        var ratioX = (width/{$mapWidth}).toFixed(5);
        var ratioY = (height/{$mapHeight}).toFixed(5);
        var x = Math.round(shiftedX/ratioX);
        var y = Math.round(shiftedY/ratioY);
        return [x, y];
        }
```

Ukázka kódu 68 – Homepage JavaScript, Přepočítávání souřadnic na obrázku

Podle požadavků (viz kapitola 3.1.2 *Detailní informace o fingerprintech*) se v případě, že budova vyžaduje používání přepočítaných souřadnic a aktuální filtr je nastaven na jedno patro, kde se pro vykreslování bodů použily souřadnice původní, budou zobrazovat souřadnice dvoje – poloha na současné mapě a zároveň na mapě, podle které proběhlo přepočítání souřadnic (v případě budovy „J“ to byla mapa prvního podlaží). Jestli se mají zobrazovat dvoje souřadnice nebo stačí jedny, bylo řešeno v presenteru uložením hodnoty TRUE nebo FALSE do proměnné v šabloně *recalculation*.

Pro zobrazení pouze přepočítávaných souřadnice je *recalculation* nastaveno na FALSE a hodnoty vrácené z funkce *getCurrentCoordinates* a uložené v poli *coordinates* se přes funkci *.append* vloží do *div#coordinates*.

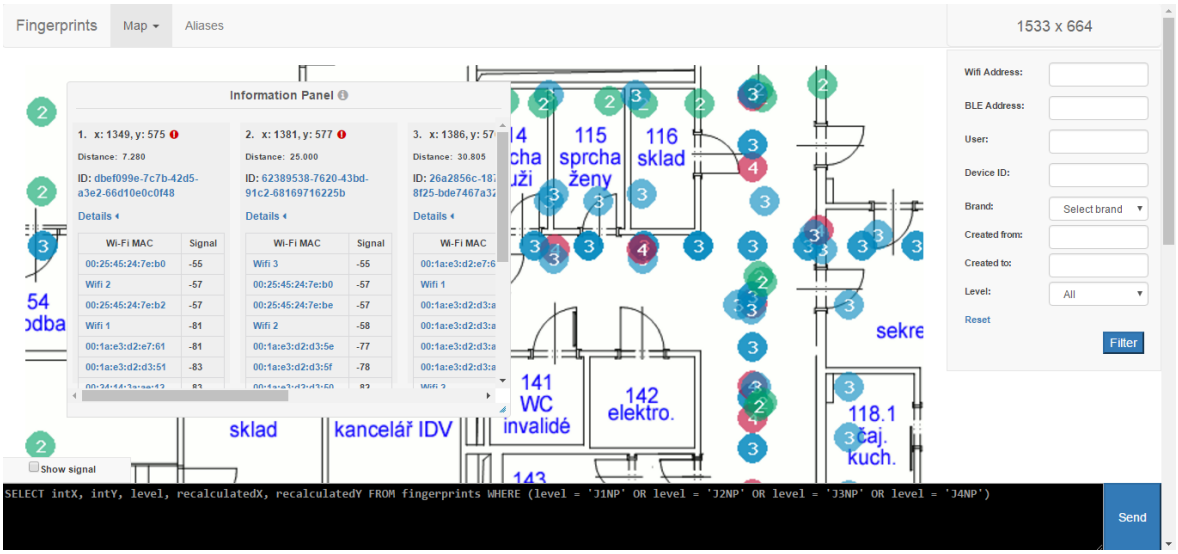
V druhém případě bylo z presenteru obdrženo pole *levelZoomAndShiftArray* s údaji o rozdílu v měřítku a posunutí a čísla v poli *coordinates* se podle nich upraví a uloží do nových proměnných (*rec_x* a *rec_y*). Nakonec se všechny souřadnice vypíší do *div#coordinates*.

Vyhledání a zobrazení detailů o fingerprintech je inicializováno kliknutím do mapy. Souřadnice získané z *Wheelzoom* se upraví v metodě *getCurrentCoordinates*. Pole s informacemi o souřadnicích kliknutí a nastavení formulářů se sestaví zavoláním metody *getClickInfo* s parametry volání x a y (souřadnice). Ajaxová funkce callAjax

odešle pole s informacemi do presenteru (funkce *handleGetInfo*) a z něj se do metody *handleData*, která se volá při úspěšné odpovědi ze serveru, vrátí výsledky (pole pěti nejbližších fingerprintů s detaily). Každý vrácený záznam se projde jQuery funkcí *.each* a vytvoří se HTML struktura, která se zobrazí v „Informačním panelu“. Vypsaným hodnotám, které lze vyplnit ve formuláři *filterData*, bude přidán falešný odkaz (na něj bude navázána jQuery funkce *.on(„click“)*, která po kliknutí na odkaz nastaví hodnotu v příslušném inputu ve formuláři *filterData* a v případě názvu Wi-Fi a Bluetooth adresy navíc i ve formuláři *filterN1ql*).

Další funkce obsažené v javascriptovém souboru slouží například k vytvoření našeptávačů, zobrazování nápověd, resetování formuláře *filterForm* do původního stavu atp.

Výsledný vzhled stránky Homepage je vidět na obrázku 18.



Obrázek 18 – Stránka Homepage

4.3.3.3 Aliases - symbolické názvy

Manipulace se symbolickými názvy (editace, vytváření a mazání) probíhá na stránce „Aliases“. Zde jsou vypsány všechny existující aliasy a jejich původní názvy (ve čtyřech tabulkách rozdělených podle typu aliasu). Typem se myslí atribut *type* v MySQL tabulce *level*, který může nabývat celkem čtyř hodnot:

- ***wifi*** – aliasy pro Wi-Fi MAC adresy
- ***ble*** – aliasy pro Bluetooth MAC adresy (stejný formát jako u *wifi*)
- ***user*** – aliasy pro uživatele

- ***deviceId*** – aliasy pro ID zařízení

Presenter

Ze služeb v modelu se budou využívat *CouchbaseService* a *MysqlDatabaseService* a renderovací funkce má za úkol pouze najít existující dvojice „originál - alias“ a poslat je do šablony, kde se vypíší do tabulek.

Textová pole v šabloně používaná k vytváření nových aliasů budou mít připojenou jQuery funkci našeptávače, který se naplní voláním metody v presenteru *getAutocomplete*, jenž podle přijaté hodnoty *type* vyhledá dosud nepřejmenované originální názvy v Couchbase.

GetOriginalsFromCouchbase rozhoduje podle hodnoty přijaté v *type*, která funkce v *CouchbaseService* se má zavolat. Vrací seznam všech různých hodnot daného atributu.

Všechny operace nad MySQL databází budou probíhat z šablony pomocí AJAXu.

HandleDeleteAlias se použije ke smazání aliasu. Jednoduše zavolá předdefinovanou funkci z *MysqlDatabaseService* s parametry alias (název aliasu) a *type* (typ aliasu) a na stránce zobrazí flashovou zprávu o úspěchu operace. Potvrzení, jestli si uživatel opravdu přeje záznam smazat, probíhá na straně klienta.

HandleSaveNewAlias se volá při pokusu uživatele vytvořit nový alias. Je potřeba zamezit duplicitám v MySQL tabulce levels, zabránit vytvoření symbolického názvu pro neexistující originál a zároveň upozornit uživatele, pokud k uložení nedošlo a proč se tak stalo. Návratová hodnota této funkce je tedy JSON response obsahující pole s klíčem reset a hodnotou string s textem indikujícím úspěch operace nebo typ chyby.

HandleShowSaveSuccess zobrazí flashovou zprávu a volá se z šablony, pokud obdržela odpověď o úspěšném uložení aliasu.

CheckOriginalInCb podle přijaté hodnoty parametru *type* vybere pomocí výrazu SWITCH text dotazu do couchbase, který má za úkol najít dokumenty s obsahující hodnotu shodnou s hodnotou přijatou v druhém argumentu – *original*. Pokud se vrátí prázdné pole, znamená to, že se hodnota *original* v Couchbase nenachází.

HandleSaveEditedAlias ukládá změny v již existujících symbolických názvech.

```
$checkAlias = $this->mysqlDatabaseService->checkIfAliasExists($alias);
if($checkAlias)
{
        return $this->sendResponse(new \Nette\Application\Responses
                                        \JsonResponse(array(['result' => 'aliasExists'])));
} else
{
        $this->mysqlDatabaseService->updateAlias($original, $alias);
        return $this->sendResponse(new \Nette\Application\Responses
                                        \JsonResponse(array(['result' => 'success'])));
}
$this->redirect('Aliases:default');
```

Ukázka kódu 69 – Aliases Presenter, Obsluha AJAXu

Šablona

V HTML kódu šablony je obsažen snippet *flashMessage* sloužící k zobrazení flashových zpráv poslaných z presenteru. Zároveň byl přidán *div* s ID *#flashes*, kam se budou přidávat zprávy vytvořené přes funkce jQuery (viz ukázka kódu 70).

```
{snippet flashMessage}
        <div id="flashes">
                {foreach $flashes as $flash}
                        <div class="flash {$flash->type}">{$flash->message}</div>
                {/foreach}
        </div>
{/snippet}
```

Ukázka kódu 70 – Aliases Latte, Flashové zprávy

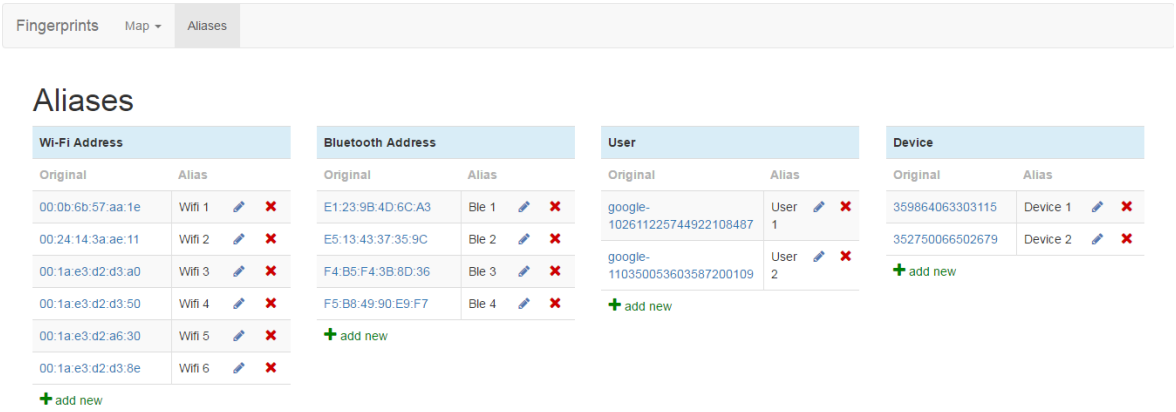
Kromě navigace a divu pro zprávy jsou v HTML stránce pouze čtyři tabulky (viz obr. 19), jejichž kód je velmi podobný, proto bude v ukázkách kódu popsána pouze jedna tabulka a následně budou nastíněny rozdíly.

Každá tabulka má sloupec s originálním názvem, symbolickým názvem, odkazem pro editaci a odkazem pro smazání. Jednotlivé řádky jsou do tabulek přidávány procházením pole s dvojicemi „originál - alias“ (posláno z presenteru) cyklem FOR-EACH. Každé projití cyklem vytvoří jeden řádek tabulky. HTML elementům budou přidány třídy, které budou využívány javascriptovými funkcemi:

- **td. original** - buňka s originálním názvem
- **td.alias** – buňka se symbolickým názvem

- **td.wifi / td.ble / td.user / td.deviceId** – přidáno navíc k buňkám **td.alias** podle typu symbolického názvu
- **a.editAlias** - odkaz pro spuštění akce editace aliasu
- **a.fakeLink** – bude přidána funkce odstranění přesměrování na jinou stránku při kliknutí
- **a.ajax** – umožní připojení Nette potvrzovacích dialogů

Odkaz pro smazání aliasu odkazuje přímo na funkci presenteru *deleteAlias* s parametry volání *alias* a typ aliasu (zde „wifi“). Po vygenerování stránky obsahuje tedy každý z těchto odkazů cestu k funkci v presenteru zahrnující název aliasu a typ aliasu, který má být smazán. Při rozkliknutí se použije Nette makro pro potvrzovací dialog a teprve po potvrzení se provede akce smazání. Po skončení cyklu bude na konci každé z tabulek přidán odkaz pro vytvoření nového aliasu, resp. nového řádku v tabulce s přidanými textovými inputy (řešeno v javascriptovém souboru).



Obrázek 19 Vzhled stránky Aliases

Javascriptový soubor

Přes jQuery budou volány funkce presenteru pro vytvoření, smazání nebo editaci symbolického názvu, dynamické přidávání inputů, navázání našeptávačů na inputy, vytváření dialogových oken a zobrazování flashových zpráv.

Podle požadavků musí být uživatel upozorněn předtím, než opustí stránku, na neuložené změny. Pro tento účel byly vytvořeny proměnné *somethingChanged* a *sendingForm* s výchozím nastavením na hodnotu FALSE, která se bude měnit podle provedených akcí na stránce, a hodnoty obou proměnných se budou kontrolovat vždy před opuštěním stránky (viz ukázka kódu 71).

```
var somethingChanged = false;
var sendingForm = false;
$(window).bind('beforeunload', function(e) {
      if(somethingChanged && !sendingForm)
        return "Do you want to leave without saving changes?";
      else
        e = null;
   });
```

Ukázka kódu 71 – Aliases Javascript, Upozornění před opuštěním stránky

Proměnné *minLength* a *maxLength* budou určovat minimální a maximální povolenou délku textu v inputu.

Smazání aliasu

Ukázce kódu 72 uvádí příklad odkazu vedoucího na smazání řádku v tabulce.

```
<a n:href='deleteAlias! $alias, "wifi"'
       data-confirm='Are you sure you want to delete this alias?' class='ajax'>
```

Ukázka kódu 72 – Aliases Javascript, odkaz na smazání symbolického názvu

V kódu javascriptového souboru bude přidána funkce *$.nette.ext* (viz ukázka kódu 73), která zajistí, že se před každým posláním ajaxového requestu zavolá event *before*, jenž v parametru *settings* obsahuje informace o zdroji requestu. Zkontroluje se atribut *data*, a pokud se shoduje s „confirm“, vytvoří se dialogové okno vyžadující potvrzení. Teprve po odsouhlasení se zavolá funkce v presenteru *handleDeleteAlias*, na kterou vedl odkaz.

```
$.nette.ext({
                before: function (xhr, settings) {
                        if (!settings.nette) {
                                return;
                        }
                        var question = settings.nette.el.data('confirm');
                        if (question) {
                                return confirm(question);
                        }
                }
        });
```

Ukázka kódu 73 – Aliases Javascript, potvrzení před smazáním

Vytvoření nového aliasu

Pro kliknutí na tlačítko přidání nového aliasu bude spuštěna funkce navázaná na odkaz, která podle ID elementu zjistí typ symbolického názvu, který se bude vytvářet, a zavolá metodu *addRow*, jenž vytvoří nový řádek tabulky obsahující buňky s inputy a tlačítky pro uložení nového symbolického názvu a odstranění tohoto řádku. Dynamicky vytvořeným textovým inputům budou přidány našeptávače (přes funkci *createAutocomplete* a jQuery funkci *autocomplete*).

Odkaz pro uložení nové dvojice „original - alias" má třídu *.saveNewAlias*, na kterou je navázána funkce, jenž se po kliknutí na odkaz spustí. Zde se zkontrolují vstupy od uživatele a zajistí se, aby byly oba vyplněné, měly alespoň minimální délku a zároveň nepřesahovaly délku maximální[12]. To vše se děje při volání funkce *checkInputValues,* která vrací hodnotu TRUE při chybě (*error* = TRUE) nebo FALSE, pokud je vše v pořádku.

Ajaxová funkce se volá pouze v případě, že *checkInputValues* vrátila hodnotu FALSE. Do pole f*ormData* se uloží text s názvem originálu, aliasu a typem. Proměnná *sendingForm* bude nastavena na FALSE, aby uživatel nemusel potvrzovat opuštění stránky při znovunačtení.

CheckInputValues přijímá tři argumenty:

- ***originalOrOld*** – Originální název nebo NULL
- ***newAlias*** – Nový symbolický název, který chce uživatel vytvořit
- ***action*** – indikátor, jestli se jedná o akci vytvoření nové dvojice „originál - alias" nebo o editaci existujícího názvu

Podle *action* se přes příkaz SWITCH zkontrolují přijaté hodnoty a podle definovaných podmínek se buď zobrazí flashová zpráva o chybě v divu *#flashes* a vrátí hodnota TRUE, nebo se jen vrátí hodnota FALSE znamenající, že je vše v pořádku.

Kontrolování délky textových řetězců probíhá ve funkcích *checkMinLength* a *checkMaxLength*. Obě metody přijímají textový řetězec jako argument, porovnají jeho délku s hodnotou limitu uloženou v globální proměnné a vrátí TRUE nebo FALSE.

[12] U obou vstupů je nastavena stejná minimální i stejná maximální délka (lze případně snadno upravit přidáním nových globálních proměnných)

```
function checkMinLength(text) {
        if(text.length < minLength) {
                return false;
        } else {
                return true;
        }
}
function checkMaxLength(text) {
        if(text.length > maxLength) {
                return false;
        } else {
                return true;
        }
}
```

Jelikož je většina flashových zpráv na této stránce vytvářena přidáváním HTML kódu do určitého elementu *div*, byla pro každý používaný text flashové zprávy vytvořena zvláštní funkce, přičemž každá z nich vytvoří HTML kód a zavolá metodu *createFlashMessage*, která přijatý kód vloží do elementu *div#flashes* (stylován pomocí doplňku Bootstrap).

Ajaxová funkce, která má na starosti komunikaci s presenterem je v případě vytváření nových symbolických názvů *callAjaxSaveNew*. Přijatá data posílá do presenteru, kde volá funkci *handleSaveNewAlias*. Podle hodnoty v poli *result* se buď zobrazí informace o úspěchu operace, nebo chybová hláška[13].

Editace existujícího aliasu

Změna již vytvořeného symbolického názvu je umožněna kliknutím na odkaz vedle buňky obsahující tento název (ikona tužky). Spuštěná funkce vytvoří nový textový input s předvyplněným textem (starý název aliasu) a změní odkaz, přes který uživatel změnu inicializoval, na odkaz pro uložení změny do databáze. Po kliknutí na odkaz pro uložení editovaného aliasu se spustí funkce, která nejprve zjistí hodnotu nového symbolického názvu a originálního názvu, poté zkontroluje, jestli nový název splňuje požadavky zavoláním metody *checkInputValues* a v případě úspěchu zavolá ajaxovou funkci *callAjaxEdit*, kam pošle zjištěné hodnoty.

[13] Zprávy o úspěšných operacích jsou vytvářeny v presenteru pomocí funkcí Nette (flashMessage()), zatímco chybové zprávy v javascriptovém souboru (především kvůli možnosti CSS stylování zpráv)

CallAjaxEdit posílá data do metody v presenteru *handleSaveEditedAlias*. V odpovědi se v poli result hledá hodnota „success“ značící úspěch uložení do databáze a „aliasExists“, která je způsobena pokusem o uložení již existujícího aliasu. V obou případech se zobrazí zpráva stejným způsobem, jako bylo vysvětleno u metody *callAjaxSaveNew*.

4.3.3.4 Document - JSON dokumenty

Zobrazení jednoho konkrétního záznamu z Couchbase probíhá na stránce „Document“. Podle přijaté hodnoty ID dokumentu se vyhledají potřebná data a výsledek se zobrazí stylovaný do podoby JSON stromu se „zabalovacími“ listy.

Presenter

Třída DocumentPresenter využívá služby *CouchbaseService* a *MysqlDatabaseService*. Jediná PHP funkce, kterou je zde potřeba nadefinovat, je renderovací funkce *renderShow*, jejímž parametrem volání je ID dokumentu v Couchbase, podle kterého se záznam v databázi vyhledá. Každý odkaz, který má vést na stránku s dokumentem, musí tuto hodnotu v cestě k cíli obsahovat.

Do *latte* šablony se přidají tři proměnné:

1. *doc* – dokument zakódovaný do formátu JSON
2. *documentId*- ID dokumentu
3. *aliases* – pole všech vytvořených aliasů zakódované do formátu JSON

Šablona

Název stránky „Document“ obsahuje ID dokumentu získané z presenteru v proměnné *documentId*. Na HTML element *div#tree* se v jQuery naváží funkce rozšíření JSON.VIEW (viz ukázka kódu 75).

Javascriptový soubor

Jak bylo zmíněno výše, do elementu *div#tree* se zobrazí dokument z Couchbase v podobě JSON stromu. Toho se docílí zavoláním funkce *jsonView* (viz ukázka kódu 75). Parametrem volání je zde JSON pole obsahující záznam z Couchbase, které se sem bylo posláno z presenteru v proměnné *doc*.

```
$( '#tree' ).jsonView( {$doc} );
```

Ukázka kódu 74 – Document Javascript, navázání jsonView na HTML element

Tento jeden řádek stačí k zobrazení celého stromu fingerprintu. Podle požadavků se ale ještě musí provést dva úkoly:

1. navázat na vybrané atributy odkazy vedoucí na Homepage s nastaveným filtrem na hodnotu tohoto atributu
2. přidat k atributům vytvořené aliasy

Strom vypsaný na stránce je vlastně jeden velký HTML seznam obsahující další vnořené seznamy (viz ukázka kódu 75). Jednotlivé elementy *li* jsou tvořeny elementy *span*, kterým byla kvůli stylování, které používá JSON.VIEW, přiřazena třída podle obsahu *spanu* (např. „.q“ pro uvozovky, „.str“ pro hodnotu typu string z pole *doc*, „.num“ pro hodnotu typu *int* apod.).

```
<ul class="obj collapsible level0">
        <li>…</li>
        <li>
                <span class="q">"</span>
                "couchbase_sync_gateway_id "
                <span class="q">"</span>:
                <span>
                        <span class="q">"</span>
                        <span class="str">"google-110350053603587200109"</span>
                        <span class="q">"</span>
                </span>
                "  „"
        </li>
        <li>…</li>
```

Ukázka kódu 75 – Document Javascript, HTML kód vytvořený nástrojem JSON.VEW

Toho bylo využito při procházení listů a manipulací s nalezenými texty (viz ukázka kódu 76). Nejprve se pomocí jQuery funkce *.each* každé položce seznamu, která obsahuje vnořený seznam, přidá třída *.parent* a každé koncové položce třída *.child*. Všechny texty, které je potřeba změnit se budou nacházet v listech s třídou *.child* a budou ohraničeny tagy *span* s třídou *.str*. Následně se nad všemi elementy *li.child* zavolá další *.each* funkce, ve které se kontroluje přítomnost klíčových slov – názvů atributů v Couchbase:

- „mac“

- „address“
- „couchbase_sync_gateway_id“
- „deviceId“
- „brand“
- „level“
- „createdAt“

Při shodě se text z dalšího elementu s třídou *.str* uloží do proměnné *text* a buď se hned nahradí odkazem na Homepage, který v cestě obsahuje nastavení persistentní proměnné na hodnotu v *text,* nebo se v případech, kdy lze přidat alias, zavolá funkce *findAlias*.

```
$( 'li.child' ).each( function () {
        var element;
        var text;
        var quotes;
        if($(this).is(':contains("mac")')) {
                element = $(this).find('.str');
                text = element.text();
                quotes = $(this).find('.q:last');
                findAlias(element, text, quotes, 'wifi');
        } else if($(this).is(':contains("address")')) {
                        element = $(this).find('.str');
                        text = element.text();
                        quotes = $(this).find('.q:last');
                        findAlias(element, text, quotes, 'ble');
        } else if ($(this).is(':contains("level")')) {
                element = $(this).find('.str');
                text = element.text();
                $(this).find('.str').replaceWith( '<a class="link" href="../../?level='+text+'">'
                                                                        + text +'</a>' );
        } else …
```

Ukázka kódu 76 – Document Javascript, Procházení listů JSON stromu

Metoda pro přidání aliasů *findAlias* (viz ukázka kódu 77) přijímá jako argumenty:

- *element* – prvek, který se má nahradit aliasem
- *original* – původní text
- *quotes* – prvek obsahující uvozovky[14]
- *type* – typ aliasu, použije se při generování URL adresy

[14] Přidáno kvůli stylování (nahrazování tagem *mark*)

V poli *aliases*, které bylo do šablony přidáno z presenteru v renderovací metodě se funkce pokusí vyhledat alias podle originálního názvu přijatého v *original*.

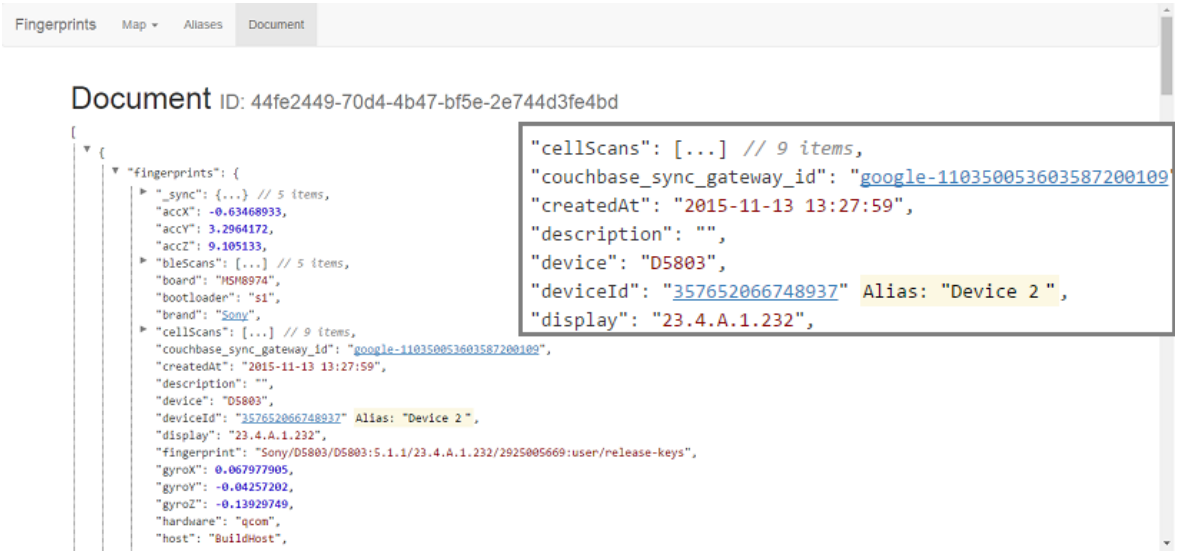
```
function findAlias(element, original, quotes, type) {
                        var aliases = {$aliases};
                        var alias = aliases[original];
                        if(alias != null) {
                                element.replaceWith( '<a class="link" href="../../?' + type + '='
                                + original + '">' + original + '</a>"
                                <mark>Alias: "' + alias + '</mark>' );
                        quotes.replaceWith( '<mark>"</mark>' );
                        } else {
                                element.replaceWith( '<a class="link" href="../../?' + type + '='
                                        + original + '">' + original + '</a>' );
                        }
                }
```

Ukázka kódu 77 – Document Javascript, Vytváření odkazů a hledání aliasů

Ke všem vytvořeným odkazům byla přidána třída *.link*, aby se nakonec ještě mohlo přidat CSS stylování „underline“ (text odkazu bude podtržený, aby bylo více zřejmé, že se jedná o odkaz). Výsledný HTML kód je naznačen v ukázce kódu 78 a na obrázku 21 je předveden vzhled stránky.

```
<li class="child">
        <span class="q">"</span>
        "couchbase_sync_gateway_id"
        <span class="q">"</span>
        :
        <span>
        <span class="q">"</span>
                <a class="link" href="../../?user=google-110350053603587200109"
                style="text-decoration: underline;">"google-110350053603587200109"
                </a>
                " <mark>Alias: "User 2</mark><mark>"</mark>
        </span>
        ","
</li>
```

Ukázka kódu 78– Document Javascript, JSON strom po úpravě



Obrázek 20: Vzhled stránky Document

5 Nasazení a testování

Pro nasazení na školní server beacon.uhk.cz (přístupný z virtuálních učeben) bylo potřeba zkopírovat složku s Nette frameworkem (nazvanou „fingerprints“) do adresáře „C:\xampp\htdocs\“ a nainstalovala se PHP rozšíření pro práci s Couchbase. Postup je detailně vysvětlen v kapitole 4.1.1 *Zavedení databáze*.

V MySQL byla vytvořena databáze „fingerprints“ podle popisu v kapitole 4.2 *MySQL*[15]

Testování probíhalo v průběhu vývoje aplikace i po nasazení na školní server na PC se systémem Windows 7 a 8 v prohlížečích:

- Google Chrome (verze 57.0.2)
- Mozilla Firefox (verze 52.0.2)
- Internet Explorer (verze 11.0.9)

Níže jsou uvedeny příklady překážek, na které se během testování narazilo.

Couchbase:

1. Souřadnice fingerprintů jsou v Couchbase uloženy ve formátu *string*

[15] Skript pro vytvoření MySQL databáze je obsažen v příloze

2. Při vyhledávání fingerprintů podle složitého dotazu (např. vyhledání hodnot v hnízděném poli) u verze Couchbase nižší než *4.5* trvá odezva z databáze příliš dlouho
3. Nepodařilo se implementovat přístup aplikace Fingerprints do databáze pod uživatelskou rolí „pouze pro čtení“

Přidaná rozšíření:

4. Přidávání značek fingerprintů do mapy je v rozšíření *imgNote* příliš pomalé při velkém počtu bodů
5. *Wheelzoom* způsobuje, že při nahrání stránky ve zmenšeném okně a následném maximalizování se zobrazená mapa s fingerprinty neroztáhne podle aktuální velikosti (nemá to vliv na počítání souřadnic kliknutí)

Stránka „Aliases“:

6. Lze přejmenovávat i originální názvy, které se v Couchbase nenacházejí

Stránka „Homepage“:

7. Uživatel odeslal vlastní N1QL dotaz, kde nevyhledal pole důležitá pro správné vykreslení bodů do mapy, což vedlo k zobrazení chyby z Couchbase
8. V prohlížeči IE nefunguje získání vlastnosti *background-position-x* přes jQuery funkci *.css()*

Všechny odhalené chyby kromě 3. a 5. byly odstraněny.

Ad 3) Podle zdroje (30) je v balíčcích PHP SDK od verze 2.3.0 implementována autentizace na úrovni clusteru, tzv. *ClassicAuthenticator*. Tento balíček se však pro nepodařilo úspěšně nainstalovat. Byla přidána alespoň kontrola textu dotazu vytvořeným uživatelem.

Ad 5) Pokud nastane chyba č. 5, postačí reload stránky.

Testovací verze aplikace Fingerprints je k nahlédnutí na adrese:

beacon.uhk.cz/fingerprints

6 Závěry a doporučení

V aplikaci „Nástroj pro databázi rádiových fingerprintů“ se podařilo implementovat všechny stanovené požadavky.

Na hlavní stránce jsou zobrazeny všechny body měření, které lze filtrovat podle specifikovaných atributů. Body jsou buď barevně rozlišeny podle podlaží, kde proběhlo měření, nebo podle síly signálu určité MAC adresy, podle které byly body filtrovány. Barvy podlaží jsou specifikovány v MySQL databázi, spolu s dvojicemi „číslo podlaží - označení v Couchbase“. Mapu lze přibližovat, oddalovat a posouvat pomocí myši díky implementovaným funkcím jQuery pluginu *Wheelzoom*.

Dole na stránce je umístěn formulář s textovým polem, kde může uživatel provádět vlastní N1QL dotazy. Vstup uživatele je kontrolován, aby bylo minimalizováno riziko zásahu do Couchbase.

Kvůli možnosti zobrazení všech bodů z různých pater budovy a faktu, že obrázky s mapami jsou různě velké a pixelově posunuté, bylo potřeba k obdrženým JSON dokumentům přidat nové atributy, aby se ulehčilo vyhledávání nejbližších bodů od určitého místa v mapě.

Výpis, úprava a mazání symbolických názvů probíhá na stránce „Aliases“. Všechny uživatelem vytvořené názvy jsou rozděleny podle typu do čtyř tabulek jako dvojice „originální název – alias“. Manipulace se záznamy probíhá intuitivně pomocí přidaných jQuery funkcí. Záznamy s aliasy se neukládají do stejné databáze jako dokumenty, ale do samostatné tabulky v MySQL databázi.

Byla přidána možnost zobrazit konkrétní JSON dokument z databáze ve formě stromu. Pro formátování textu a přidání funkce „zabalení větví stromu“ byl použit jQuery plugin *JSON.VIEW*. Na stránku s dokumentem se lze dostat kliknutím na odkaz s ID dokumentu v databázi, který se objeví při kliknutí do mapy na hlavní stránce v boxu pod levým horním filtrem (*Informační panel*).

Přestože byly v době vývoje k dispozici pouze mapy a fingerprinty z budovy FIM („J“) UHK, byla aplikace navržena tak, aby bylo snadné přidat zobrazení fingerprintů zachycených na dalších budovách.

Seznam použité literatury

1. **Kříž, Pavel, Malý, Filip a Kozel, Tomáš.** Improving Indoor Localization Using Bluetooth Low Energy Beacons. *Hindawi.* [Online] 27. březen 2016. [Citace: 20. březen 2017.] https://www.hindawi.com/journals/misy/2016/2083094/.

2. **MOGG, Wayne.** jQuery Plugin For Adding Notes and Markers To An Image – imgNotes. *www.jqueryscript.net.* [Online] 2013. [Citace: 10. srpen 2016.] http://www.jqueryscript.net/zoom/jQuery-Plugin-For-Adding-Notes-Markers-To-An-Image-imgNotes.html.

3. **BRADWELL, Joe.** Converting Signal Strength Percentage to dBm Values. *madwifi-project.org.* [Online] 2002. [Citace: 12. srpen 2016.] http://madwifi-project.org/attachment/wiki/UserDocs/RSSI/Converting_Signal_Strength.pdf?format=raw.

4. **MOORE, Jack.** Wheelzoom, A script to zoom IMG elements on mousewheel or touchpad scrol. *www.jacklmoore.com.* [Online] 2012. [Citace: 15. leden 2016.] http://www.jacklmoore.com/wheelzoom/.

5. **Couchbase, Oficiální dokumentace.** Why NoSql Database. *couchbase.com.* [Online] Couchbase, 2016i. [Citace: 6. červen 2016.] http://www.couchbase.com/nosql-resources/why-nosql.

6. **Couchbase, Oficiální dokumentace.** Nodes and Clusters. *couchbase.com.* [Online] Couchbase, 2015. [Citace: 15. červenec 2016.] http://docs.couchbase.com/admin/admin/Concepts/concept-nodesCluster.html.

7. **Grentz, Mimi a Rabeler, Carl.** NoSQL vs SQL. *Microsoft.* [Online] 14. březen 2017. [Citace: 6. duben 2017.] https://docs.microsoft.com/en-us/azure/documentdb/documentdb-nosql-vs-sql#what-are-the-microsoft-sql-offerings.

8. **Couchbase, Oficiální dokumentace.** Document. *developer.couchbase.com.* [Online] Couchbase, 2016. [Citace: 15. červenec 2016.] http://developer.couchbase.com/documentation/mobile/current/develop/guides/couchbase-lite/native-api/document/index.html.

9. **Couchbase, Oficiální dokumentace**. Bucket. *developer.couchbase.com.* [Online] Couchbase, 2016. [Citace: 15. červenec 2016.]

http://developer.couchbase.com/documentation/server/current/architecture/core-data-access-buckets.html.

10. **Couchbase, Oficiální dokumentace**. View. *developer.couchbase.com.* [Online] Couchbase, 2016. [Citace: 11. srpen 2016.] http://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/index.html.

11. **Couchbase, Oficiální dokumentace**. N1QL Language Reference. *developer.couchbase.com.* [Online] 2016. [Citace: 11. srpen 2016.] http://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/index.html.

12. **Couchbase, Oficiální dokumentace**. Nested Operators. *developer.couchbase.com.* [Online] [Citace: 6. duben 2017.] https://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/nestedops.html.

13. **Couchbase, Oficiální dokumentace**. N1QL Queries and Results. *developer.couchbase.com.* [Online] Couchbase, 11. srpen 2016. http://developer.couchbase.com/documentation/server/4.5/n1ql/n1ql-intro/queriesandresults.html.

14. **Couchbase, Oficiální dokumentace**. Collection Operators. *developer.couchbase.com.* [Online] Couchbase, 2017. [Citace: 26. březen 2017.] https://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/collectionops.html.

15. **Couchbase, Oficiální dokumentace**. Indexing. *developer.couchbase.com.* [Online] Couchbase, 2016. [Citace: 12. srpen 2016.] http://developer.couchbase.com/documentation/server/4.5/concepts/indexing.html.

16. **Couchbase, Oficiální dokumentace**. Array Indexing. *developer.couchbase.com.* [Online] Couchbase, 2017. [Citace: 10. duben 2017.] https://developer.couchbase.com/documentation/server/current/n1ql/n1ql-language-reference/indexing-arrays.html.

17. **Nette, Oficiální dokumentace.** Seznámení se s Nette Frameworkem. *nette.org.* [Online] 2016. [Citace: 3. srpen 2016.] https://doc.nette.org/cs/2.4/getting-started.

18. **POTEL, Mike.** MVP: Model-View-Presenter. *www.wildcrest.com.* [Online] 1996. [Citace: 3. srpen 2016.] http://www.wildcrest.com/Potel/Portfolio/mvp.pdf.

19. **CHANDEL, Sumit.** Testing, Gwtproject. *www.gwtproject.org.* [Online] 2009. [Citace: 4. srpen 2016.] http://www.gwtproject.org/articles/testing_methodologies_using_gwt.html.

20. **Nette, Oficiální dokumentace.** Model-View-Presenter (MVP). *nette.org.* [Online] 2016. [Citace: 3. srpen 2016.] https://doc.nette.org/cs/0.9/model-view-presenter.

21. **Nette, Oficiální dokumentace**. Latte. *nette.org.* [Online] 2016. [Citace: 3. srpen 2016.] https://latte.nette.org/cs/.

22. **Nette, Oficiální dokumentace**. Konfigurace. *nette.org.* [Online] 2016. [Citace: 5. srpen 2016.] https://doc.nette.org/cs/2.4/configuring.

23. **Nette, Oficiální dokumentace**. AJAX & snippety. *doc.nette.org.* [Online] [Citace: 6. duben 2017.] https://doc.nette.org/cs/2.4/ajax.

24. **Dobeš, Vojtěch.** Nette.ajax.js. *Componette.com.* [Online] [Citace: 4. duben 2017.] https://componette.com/vojtech-dobes/nette.ajax.js/.

25. **The jQuery foundation.** jQuery. *jquery.com.* [Online] [Citace: 15. březen 2017.] https://jquery.com/.

26. **BAZHENOV, Anton.** jQuery Based Pretty Collapsible JSON Tree. *jqueryscript.net.* [Online] 2014. [Citace: 6. červen 2016.] http://www.jqueryscript.net/other/jQuery-Based-Pretty-Collapsible-JSON-Tree-Viewer.html.

27. **Couchbase, Oficiální stránky.** Couchbase. *couchbase.com.* [Online] 2016. [Citace: 11. srpen 2016.] http://www.couchbase.com/.

28. **PECL.** PECL – repository for PHP group. [Online] [Citace: 11. srpen 2016.] http://pecl.php.net/.

29. **Couchbase, Oficiální dokumentace.** Running Your First N1QL Query. *couchbase.com.* [Online] 2016. [Citace: 10. srpen 2016.] http://developer.couchbase.com/documentation/server/4.1/getting-started/first-n1ql-query.html.

30. **Couchbase, Oficiální dokumentace**. Couchbase PHP Release Notes and Archives. *developer.couchbase.com.* [Online] [Citace: 18. duben 2017.] https://developer.couchbase.com/server/other-products/release-notes-archives/php-sdk.

Přílohy

Přiložené CD ROM obsahuje:

1. *fingerprints* – složka se zdrojovými kódy serverové aplikace
2. *load.sql* - skript pro vytvoření databázového schématu v MySQL