VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALYTICKÝ OVLÁDACÍ PANEL PRO LOKAČNÍ SYSTÉM

ANALYTICAL DASHBOARD FOR INDOOR LOCATING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ PETROVIČ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR VESELÝ, Ph.D.
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19349/2016/xpetro11

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Petrovič Lukáš**
Obor: Informační technologie
Téma: **Analytický ovládací panel pro lokační systém**
Analytical Dashboard for Indoor Locating System
Kategorie: Počítačové sítě

Pokyny:

1. Vyhledejte vhodné knihovny pro realizaci webových dashboard aplikací. Prostudujte vhodné návrhové vzory a pokročilé vizualizační metody.
2. Prostudujte databázi a API real-time lokačního systému firmy Sewio. Analyzujte i jiná existující komerční a nekomerční řešení z volně dostupných materiálů.
3. Navrhněte generickou aplikaci, která bude vizualizovat vybrané metriky, ukazovat závislosti a poskytovat analytiku a pokročilou správu RTLS.
4. Navrženou aplikaci implementujte a otestujte její chování na reálných datech.
5. Zhodnoťte dosažené výsledky a možnosti dalšího vývoje.

Literatura:

- Jose A. Gutierrez. IEEE 802.15.4 Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensor Networks: Institute of Electrical & Electronics Enginee, 2003. 155p. ISBN 0738135577
- J. Duckett, JavaScript and JQuery: Interactive Front-End Web Development Paperback, Wiley, June 30, 2014,ISBN 1118531647

Pro udělení zápočtu za první semestr je požadováno:
- Body 1 až 3 včetně.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Konzultant: Šimek Milan, Ing., FEKT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cieľom tejto práce je vytvoriť aplikáciu, ktorá umožňuje vizualizovať historické dáta z polohových senzorov. Dáta sú získavané z uzavrených priestorov, akými sú napríklad obchody, telocvične alebo haly. Tieto dáta sú cenné, no bez ľahko použiteľnej vizualizácie strácajú svoj potenciál. Správna vizualizácia umožňuje používateľovi lepšie vyhodnotiť situáciu, ktorú tieto dáta reprezentujú. Výsledkom tejto práce je nástroj, s ktorým je používateľ schopný jednoducho analyzovať dáta, získané v určitom časovom intervale. Pre potreby analýzy poskytuje aplikácia viacero metrík, ktoré priraďujú týmto dátam špecifický účel. Dáta môžu byť reprezentované v grafoch, napríklad za účelom reprezentovať celkovú prejdenú vzdialenosť lokalizovaného objektu. Taktiež môžu byť použité v mapách, ktoré poskytujú dobrú reprezentáciu polôh vzhľadom k monitorovanej miestnosti. Všetky vytvorené vizualizačné prvky sú vložené do interaktívnych blokov. Tieto bloky tvoria takmer celý ovládací panel. S týmito blokmi je možné jednoducho manipulovať a s ich pomocou upravovať vytvorenú vizualizáciu.

Abstract

The main purpose of this work is to create an application which will provide visualisation of historical data from position sensors. The data are being obtained from closed spaces, for example shops, gymnasiums or halls. These data are valuable but without easily usable visualisation they will lose their potential. Right visualisation allows user to better evaluate situation which the data represents. Result of this work is a tool with which the user is able to easily analyse data gained in certain time interval. For the needs of analysis, the application provides various kinds of data representations, which assign the data specific purpose. Data might be presented in graphs for example for the purpose of representation of the total overcomed distance of monitored object. They can also be used in maps which provide good representation of positions due to monitored area. All created visualization elements are put into interactive blocks. These blocks make up almost whole control panel. It is easy to manipulate with these blocks and to edit created visualisation with their help.

Kľúčové slová

Ovládací panel, Lokalizácia, Analýza, Vizualizácia

Keywords

Dashboard, RTLS, Localization, Analytics, Visualization

Citácia

PETROVIČ, Lukáš. *Analytický ovládací panel pro lokační systém.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Veselý, Ph.D.

Analytický ovládací panel pro lokační systém

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Vladimíra Veselého, Ph.D. Ďalšie informácie mi poskytli zamestnanci spoločnosti Sewio Networks s.r.o. Uviedol som všetky literárne pramene, z ktorých som čerpal.

. . . . . .
Lukáš Petrovič
16. mája 2017

Poďakovanie

Chcel by som poďakovať Ing. Ľubomírovi Mrázovi a Ing. Tomášovi Kočanovi za technické vedenie projektu. Poďakovanie patrí aj vedúcemu mojej práce Ing. Vladimírovi Veselému, Ph.D. za nasmerovanie a kontrolu postupu pri vypracovaní tejto práce a ktorému venujem nasledujúci recept. Mojím poďakovaním vo forme receptu, je recept na jednoduché domáce sushi. Základom každého sushi je dobre pripravená ryža. Použijeme 500g ryže, ktorú najprv 4-5 krát opláchneme pod prúdom vody. Po poslednom opláchnutí vylejeme všetku vodu, nalejeme do nádoby 600 ml vody a počkáme asi 30 min. Potom ryžu varíme 10 minút. Na marinádu použijeme 6 polievkových lyžíc ryžového octu, 2 až 3 polievkové lyžice cukru a 0,5 kávovej lyžičky soli. Ocot, soľ a cukor zmiešame v menšej nádobe, privedieme do varu a nádobu odložíme na chladné miesto. Po vychladnutí ryže a marinády ich zmiešame. Následne môžme vytvárať vynikajúce varianty sushi, podľa vlastnej chuti. Medzi najchutnejšie, podľa mňa patria klasické lososové maki. Ich príprava spočíva v zavinutí kúskov lososa do sušenej riasy a pripravenej ryže.

Obsah

1 Úvod **3**

2 Lokalizácia v reálnom čase **4**
2.1 Architektúra lokalizačného systému 4
2.2 Metóda lokalizácie objektu 5
2.3 Aplikačné rozhranie lokalizačného systému 7
2.4 Existujúce riešenia 7

3 Architektúra **8**
3.1 Úlohy ovládacieho panelu 8
3.2 Klientská časť ovládacieho panelu 10
3.3 Serverová časť ovládacieho panelu 12
3.4 Možnosti zabezpečenia 14
3.5 Implementácia 15

4 Možnosti vizualizácie dát **19**
4.1 Združovanie objektov a delenie monitorovanej oblasti 19
4.2 Jednoduché vizualizačné prvky 20
4.2.1 Prvý a posledný výskyt v zóne 20
4.3 Reprezentácia dát pomocou grafov 20
4.3.1 Vzdialenosť 21
4.3.2 Aktivita 22
4.3.3 Obsadenie 22
4.3.4 Rýchlosť 23
4.4 Reprezentácia dát pomocou máp 24
4.4.1 Teplotná mapa 24
4.4.2 Mapa ciest 25
4.4.3 Mapa zón 25

5 Databáza a redukcia dát **27**
5.1 Špecifikácia databázy 27
5.1.1 Úprava využívanej databázy 28
5.2 Uložené dáta a požiadavky na ich vyhľadávanie 29
5.3 Optimalizačný algoritmus pre mapové vizualizácie 31

6 Interakcia s ovládacím panelom **34**
6.1 Možnosti manipulácie s ovládacím panelom 35
6.2 Možnosti manipulácie s prvkami vizualizácie 36

6.3 Špecifikácia požiadavky pre vytvorenie prvku vizualizácie 38

7 Testovanie **39**
7.1 Podporované zariadenia a webové prehliadače 39
7.2 Porovnanie vizualizácie s reálnym pohybom monitorového objektu 40
7.2.1 Množina testovacích dát 40
7.2.2 Výsledky testovania vizualizácií 40

8 Záver **46**

Literatúra **47**

Kapitola 1

Úvod

Presné lokalizačné dáta majú v dnešnej dobe veľkú hodnotu. Lokalizácia v reálnom čase, teda RTLS (z angl. Real-time locating system) spoločnosti Sewio Network s.r.o, ďalej len Sewio, sa zameriava na zber dát z objektov a spracovanie týchto dát. Tieto objekty môžu reprezentovať ľudí, stroje, alebo tovar. Reálne je tento systém používaný napríklad pre monitorovanie hráčov basketbalu. V situácii kedy sú k dispozícii presné a spoľahlivé polohy, je ďalším prirodzeným štádiom analýza týchto dát. Cieľom aplikácie je teda spoľahlivo reprezentovať zozbierané dáta.

V terajšej fáze systému je zákazníkovi poskytnutý kompletný systém pre zber polohových dát a taktiež aj aplikácie pre ovládanie tohto systému. Zákazník si tieto dáta následne spracovával sám. S vyvinutou aplikáciou budú môcť zákazníci jednoduchšie sledovať spracované dáta. Princíp fungovania aktuálne nasadeného lokalizačného systému je opísaný v kapitole 2.

Návrh aplikácie spočíval v správnom výbere aplikačnej platformy. Dôraz bol kladený na rýchlu odozvu na požiadavky používateľa, ako aj na široké možnosti konfigurácie ovládacieho panelu. Dôvody výberu webovej platformy a popis jednotlivých vrstiev systému sa nachádzajú v kapitole 3.

Hlavnou úlohou aplikácie je poskytnúť používateľovi možnosť vytvárať grafické reprezentácie dát. V kapitole 4 sú popísané možnosti, ktoré používateľ môže využiť. Aplikácia je navrhnutá tak, aby bolo jednoduché pridávať nové druhy vizualizácie dát. Vizualizáciou sa myslia rôzne grafy, mapy, prípadne tabuľky.

Kľúčovou časťou celého systému je databáza, ktorá uchováva všetky polohové dáta. Systém do tejto databázy vkladá údaje o polohe a vlastnostiach zariadení. Tieto uložené dáta sú neskôr zdrojom informácií pre ovládací panel. Výber databázy a návrh úprav stávajúcej databázy je popísaný v kapitole 5.

Kapitola 6 približuje možnosti manipulácie s ovládacím panelom. Kapitola taktiež popisuje spôsob, akým je používateľ schopný zadávať požiadavky na výslednú vizualizáciu dát.

Kapitola 2

Lokalizácia v reálnom čase

V nasledujúcej kapitole bude popísaná architektúra systému pre lokalizáciu objektov v reálnom čase spoločnosti Sewio. V prvej sekcií budú popísané jednotlivé zariadenia, ktoré tvoria lokalizačný systém. Nasledovne bude v krátkosti popísaná metóda získavania polohy lokalizovaného objektu. Záver kapitoly obsahuje popis už existujúcich riešení a motiváciu prečo je potrebné vlastné riešenie ovládacieho panelu.

2.1 Architektúra lokalizačného systému

Architektúra RTLS systému sa skladá z niekoľkých hlavných prvkov. Týmito prvkami sú:

1. Tag - zariadenie typu lokátor. Toto zariadenie sa pohybuje v priestore a pomocou bezdrôtového prenosu vysiela signál. Pomocou tohto signálu je vypočítavaná jeho poloha.

2. Kotva - zariadenie využívané na lokalizáciu tagov v priestore. Názov kotva naznačuje, že toto zariadenie je statické so známou polohou.

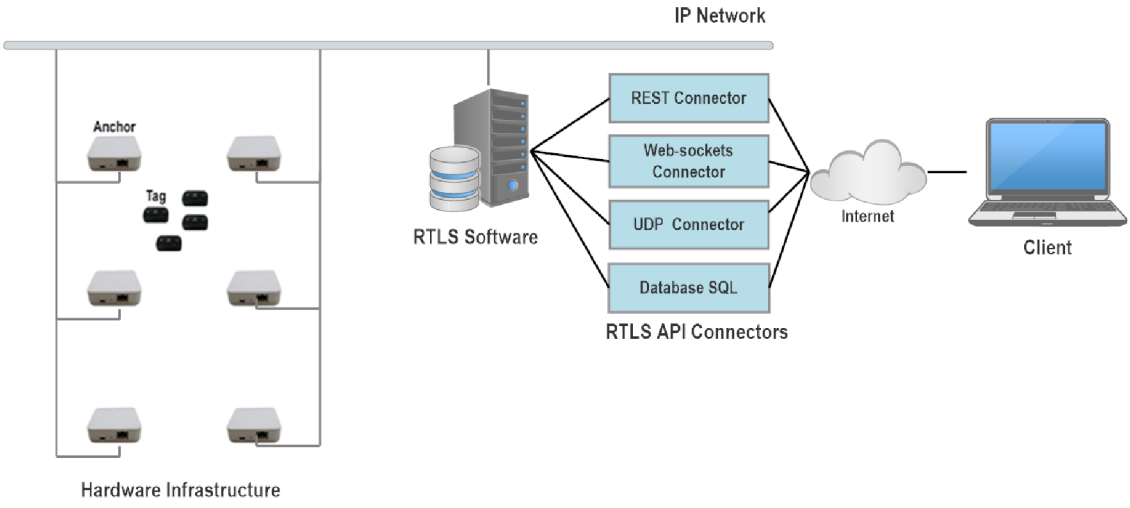
3. Jadro lokalizácie - software, ktorý komunikuje s tagmi a kotvami. Na základe získaných informácií vypočíta polohu tagu.

4. Middleware software spájajúci lokalizačný systém a aplikáciu. Umožňuje tým komunikáciu medzi zariadeniami, ktoré zbierajú dáta a aplikáciou, ktorá tieto dáta ďalej spracováva.

5. Aplikácia software, ktorý komunikuje s middlewarom a je schopný tieto dáta interpretovať, prípadne ich upravovať. Jednou z takýchto aplikácií je aj vyvíjaný ovládací panel.

Obrázok 2.1 popisuje prvky aktuálne používaného RTLS spoločnosti Sewio. Na ľavej strane obrázku sa nachádza pomyselná miestnosť, v ktorej sú rozmiestnené kotvy. Cez sieť sú získané dáta posielané sa server, kde sú následne vypočítavané pozície tagov. Obrázok taktiež približuje možnosti komunikácie s RTLS. Kapitola 2.3 približuje možnosti komunikácie medzi RTLS a vyvíjaným ovládacím panelom.

Aby bol systém schopný získať polohu jednotlivých tagov, je potrebné aby boli kotvy zosynchronizované. Metóda pomocou ktorej je vypočítavaná poloha tagu v priestore, bude popísaná v kapitole 2.2.



Obr. 2.1: Architektura RTLS.

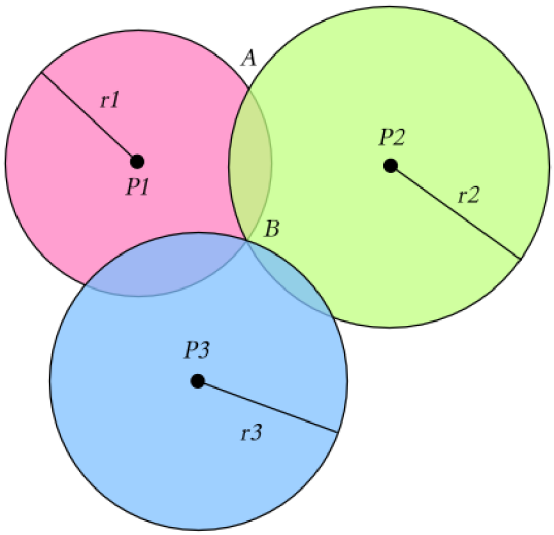
Správna synchronizácia jednotlivých kotiev je teda základná požiadavka správne fungujúceho lokalizačného systému. Pre účely synchronizácie sa vyberie z monitorovanej miestnosti jedna kotva tzv. master. S pomocou tejto kotvy si ostatné kotvy zosynchronizujú čas. Master kotva vysiela v určitých intervaloch synchronizačný signál. Ostatné kotvy tento signál príjmu a čas tohto prijatia odošlú na server. Server pozná rozmiestnenie kotiev a teda aj vzdialenosť medzi kotvami. V momente kedy vieme určiť vzdialenosť, vypočítame dobu letu signálu medzi kotvami. Server teda pozná informáciu, kedy jednotlivé kotvy prijali správu a pozná aj dobu letu signálu medzi mastrom a ostatnými kotvami. Je tak schopný synchronizovať čas na kotvách. Takáto synchronizácia sa uskutočňuje v pravidelných, niekoľko sekundových intervaloch.

2.2 Metóda lokalizácie objektu

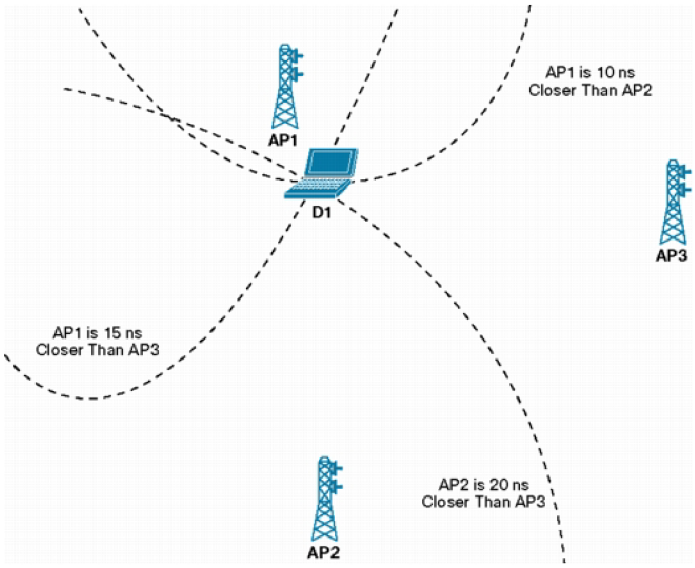
Jedným zo zástupcov lokalizačnej techniky je laterácia, niekedy označovaná aj ako trilaterácia alebo multilaterácia. Laterácia je proces, kde sa na základe nameraných vzdialeností z referenčných bodov určí pozícia tagu. Pre výpočet polohy v dvojdimenzionálnom priestore, sú potrebné dáta aspoň z troch referenčných bodov, teda kotiev. Obrázok 2.2 znázorňuje tri prijímače (body P1, P2, P3) a jeden vysielač (bod B), ktorého poloha je zisťovaná. Okolo prijímačov sú znázornené kružnice, ktorých polomer naznačuje vzdialenosť od tagu. Výsledná informácia o polohe sa vypočíta ako prienik týchto kružníc.

Na tomto princípe funguje napríklad metóda metóda TOA (z angl. Time of Arrival). Rozdiel času medzi odoslaním signálu a jeho prijatím, určí vzdialenosť tagu od kotvy. Táto metóda so sebou nesie nevýhody, vďaka ktorým nebola v RTLS použitá. Hlavnou nevýhodou je synchronizácia. Rovnako ako kotvy, tak aj tag musí byť zosynchronizovaný. To by prinieslo vysoké nároky na batériu tagu. Pri reálnom použití sa aktuálne neaktívne tagy navyše vypínajú, aby ušetrili energiu batérie. Tag by navyše musel posielať v každej správe časovú značku, čo by navýšilo objem prenášanej správy.

Riešením je využitie metódy TDOA (z angl Time Difference of Arrival). Táto metóda nepotrebuje poznať v akom čase tag vysiela svoju polohu aby vypočítala vzdialenosť od



Obr. 2.2: Princíp vyhľadania pozície objektu pomocou metódy TOA.



Obr. 2.3: Princíp vyhľadania pozície objektu pomocou metódy TDOA.

kotvy. Kotvy zachytávajú signály od jednotlivých tagov. Každá kotva, ktorá tento signál prijala, pošle časovú informáciu kedy ho prijala na server. Server následne vypočíta rozdiely časov medzi jednotlivými dvojicami kotiev. Pomocou týchto rozdielov je schopný spočítať hyperboly, na ktorých sa monitorovaný tag môže nachádzať. Priesečník hyberbol následne predstavuje vypočítanú polohu tagu. Pre správne určenie pozície tagu sú potrebné aspoň tri takéto hyperboly.

Výhodou je že tag nepotrebuje komunikovať s kotvami, ale len neustále vysiela tzv. blink signál. Táto správa je vysielaná v prednastavených intervaloch podľa aktuálneho využívania tagu. Pri monitorovaní športovcov je tento interval veľmi krátky, naopak pre sledovanie balíkov v sklade, môže tento interval predstavovať aj niekoľko sekúnd. Hodnota signálového intervalu je priamo úmerná výdrži batérie.

Obrázok 2.3 predstavuje tri kotvy (AP1, AP2, AP3) a tag (D1). Kotvy predstavujú ohniská jednotlivých hyperbol. V priesečníku vytvorených hyperbol sa nachádza lokalizovaný tag. Pri každej hyperbole je zaznamenané oneskorenie medzi dvoma kotvami.

2.3 Aplikačné rozhranie lokalizačného systému

Aplikačne rozhranie RTLS systému podporuje nasledujúce druhy pripojení:

1. REST Connector - REST[6] aplikačné rozhranie, pomocou ktorého je možné spravovať statické informácie RTLS, akými sú napríklad budovy, plány monitorovaných priestorov a iné. Toto rozhranie je ideálne pre požiadavky na historické lokalizačné dáta.

2. Web-socket Connector - rozhranie slúžiace pre výmenu dát v reálnom čase. Zasielanie dát prebieha cez spoľahlivý TCP/IP[1] protokol. Cez toto rozhranie je možné získavať pozície tagov v reálnom čase s obnovovacou frekvenciou pomalšou než 300ms.

3. UDP Connector - rozhranie, ktoré stavia na UDP[2] protokole. Medzi výhody patrí nízka odozva. Dokáže prenášať pozície tagu pri obnovovacej frekvencií do 10ms.

4. Databáza - priami prístup k databáze. Tabuľky potrebné pre ovládací panel budú popísané kapitole 5.1.

Pre potreby ovládacieho panelu bolo implementované nové aplikačné rozhranie, využívajúce REST architektúru. Toto rozhranie čerpá dáta priamo z databázy a následne ich upravuje pre potreby ovládacieho panelu. Kapitola 3.3 popisuje túto novú súčasť RTLS.

Ovládací panel sa ale postupom času rozširuje a nastáva potreba upravovať statické dáta, akými sú budovy, zóny alebo plány monitorovaných miestností. Pre túto úpravu je využívaný prvý typ pripojenia. Rozšírenia o tieto nástroje už ale nie sú obsahom tejto práce.

Hlavným zdrojom historických dát je teda priamo databáza. Podrobný popis databázy a jej úprav sa nachádza v kapitole 5.

2.4 Existujúce riešenia

Existujúcich analytických riešení je hneď niekoľko. Zväčša sa ale viažu na vlastný lokalizačný systém. Podobné riešenia sú často príliš komplexné. Tieto aplikácie majú často pevnú štruktúru a tým môžu obmedzovať používateľa. Dôvod prečo nie sú použité už fungujúce generické riešenia ovládacieho panelu je aj skutočnosť, že aplikácia bude rozširovaná o ďalšie nástroje. Medzi ne patrí napríklad nástroj pre správu skupín alebo editor zón v monitorovaných priestoroch. Podporu pre takto špecifické nástroje, neposkytuje žiadny už predpripravený ovládací panel.

Niektoré riešenia analýzy historických lokalizačných dát sa zameriavajú na špecifickú oblasť použitia. Ako príklad je možné uviesť ovládací panel spoločnosti Locatible[3]. Ten cieli na monitorovanie pacientov v zdravotníckych zariadeniach. Je teda špecificky zameraný na obmedzenú oblasť použitia.

[1] http://searchnetworking.techtarget.com/definition/TCP-IP
[2] https://www.ietf.org/rfc/rfc768.txt
[3] http://locatible.com/

Kapitola 3

Architektúra

Pri návrhu ovládacieho panelu bolo dôležité zvoliť vhodnú aplikačnú platformu. Webová aplikácia v dnešnej dobe poskytuje rozsiahle možnosti. Pri vhodnej implementácii dokáže dokonale nahradiť aplikácie, ktoré sú priamo určené pre konkrétny systém. Taktiež sa rozširuje škála zariadení, na ktorých je možné našu aplikáciu ovládať. Výber taktiež zohľadňoval terajšie aplikácie, ktoré rovnako fungujú na tejto platforme. Každá webová aplikácia stojí na správnom výbere technológií. Narozdiel od natívnych aplikácií, kde sa primárne používa jeden hlavný framework, je webové prostredie oveľa rozsiahlejšie. Táto kapitola preto popisuje výber technológií, ktoré sú vhodné pre túto aplikáciu. Kapitola približuje členenie aplikácie na dve hlavné časti.

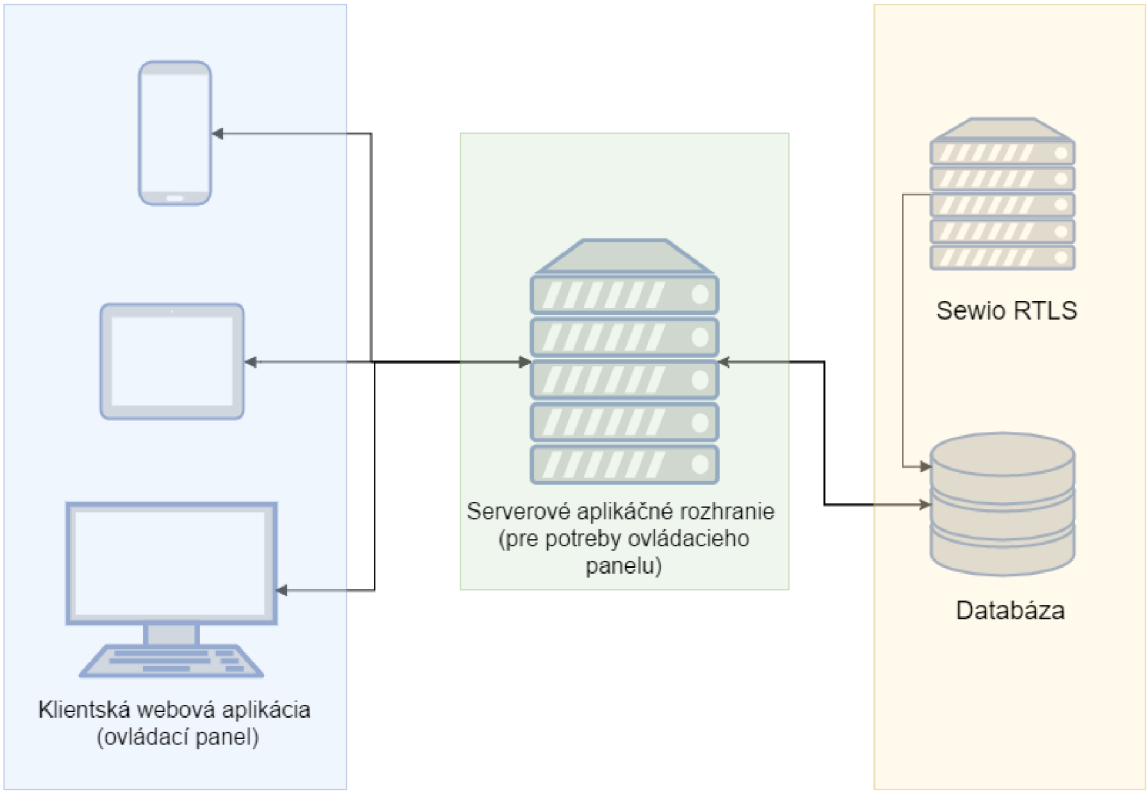
3.1 Úlohy ovládacieho panelu

Aplikácia ovládacieho panelu pozostáva z dvoch častí. Prvá časť je klientská aplikácia, ktorá má za úlohu prezentovať výsledné polohy. Táto časť práce bude bližšie špecifikovaná v kapitole 3.2. Táto kapitola v krátkosti uvedie úlohy klientskej aplikácie, týmito úlohami sú:

1. Poskytnúť používateľovi nástroj pre špecifikáciu požiadaviek na vizualizáciu dát. S týmto nástrojom je používateľ schopný meniť druh reprezentácie polohových dát. Bližšie je táto súčasť popísaná v kapitole 6.3.

2. Vizualizovať jednotlivé spôsoby reprezentácie polohových dát. Jednotlivým reprezentáciám sa venuje kapitola 4. Prvky vizualizácie sú rozšírené o interaktívnu legendu, s ktorej pomocou je používateľ schopný meniť výslednú reprezentáciu v reálnom čase. Týmto rozširujúcim možnostiam sa venuje kapitola 6.2.

3. Poskytovať používateľovi voľnosť pri vytváraní prvkov vizualizácie. Používateľ si je schopný vytvoriť množinu vizualizačných prvkov podľa jeho preferencií a potrieb. Vytvorené prvky je následne schopný presúvať, meniť ich veľkosť prípadne upravovať ich vlastnosti. Po vytvorení želanej konfigurácie, si je ju používateľ schopný uložiť a neskôr znovu zobraziť. Podrobnejšie informácie na nachádzajú v kapitole 6.1.



Obr. 3.1: Architektúra RTLS rozšírená o ovládací panel.

Druhou časťou ovládacieho panelu je serverové rozhranie. Jeho úlohami sú:

1. Spracovanie požiadavky na výslednú vizualizáciu. Toto spracovanie spočíva v dialógu medzi klientskou aplikáciou a serverom. Klientovi sú ponúkané možnosti vizualizácie dát do máp alebo grafov. Následne môže vyberať tagy a iné parametre. Možnosti výberu mu dynamicky generuje práve server. Bližšie informácie poskytuje kapitola 6.3.

2. Vyžiadanie dát z databázy. Po spracovaní požiadavky je server schopný vyžiadať správne dáta z databázy. Špecifikuje o ktoré tagy, prípadne zóny má klient záujem, na akom pláne dáta zobraziť a v akom časovom intervale. Spomenuté vlastnosti zapracuje do SQL požiadavky.

3. Transformácia pozičných dát pre potreby vizualizácie. Transformácia v zmysle získavania informácií z polohových dát. Ako príklad je možne uviesť výpočet priemernej rýchlosti v určitých časových intervaloch. Server podporuje celkovo 9 rôznych spôsobov reprezentácie dát. Tieto reprezentácie sú navyše rozšírené o ďalšie možnosti a budú popísané v kapitole 4.

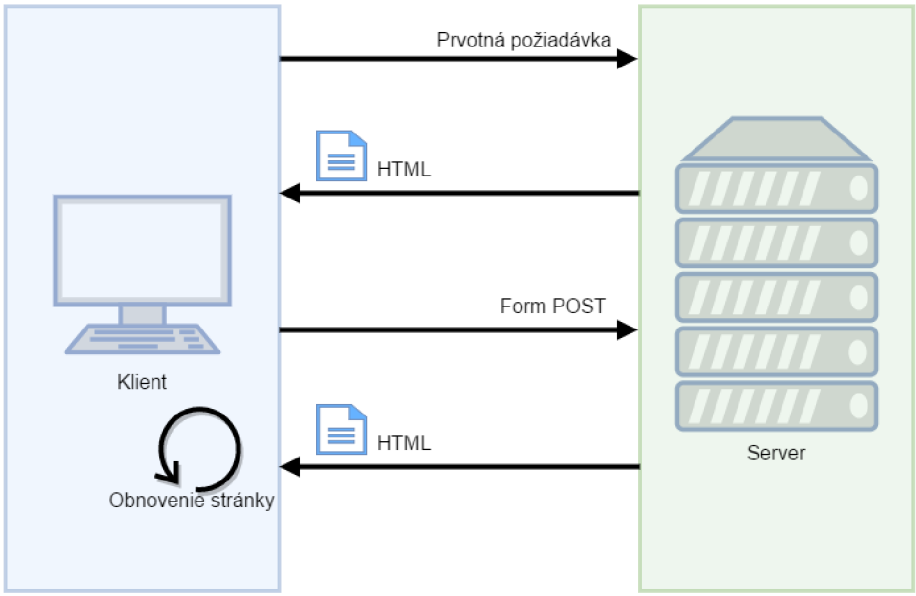
4. Kontrola a filtrovanie objemu pozičných informácií. Keďže ovládací panel pracuje s historickými dátami, je nevyhnutné, aby server dokázal v predstihu rozhodnúť, či je schopný požiadavku spracovať v rozumnom čase. Historické dáta môžu jednoducho naberať na objeme dát. Po splnení požiadaviek a získaní dát z databázy, je tiež vhodné

filtrovať tie dáta, ktoré nemajú na vizualizáciu žiadny vplyv. Týmto problémami sa bližšie venuje kapitola 5.

Obrázok 3.1 vizualizuje jednotlivé prvky systému. Zariadenia v modrej oblasti predstavujú klientskú časť ovládacieho panelu. Aj keď sa vyvíjaný ovládací panel primárne zameriava na podporu desktopu a tabletu, je možné napríklad vytvoriť natívneho klienta pre mobilný telefón. Takýto klient môže následne rovnako ako webová aplikácia využívať aplikačné rozhranie serveru. V zelenej oblasti sa nachádza serverová časť ovládacieho panelu. Táto časť bude pripojená k RTLS systému Sewio. Žltá oblasť prestavuje prvky systému, ktoré už sú reálne nasadené a ktorým sa ovládací panel musí prispôsobiť.

3.2 Klientská časť ovládacieho panelu

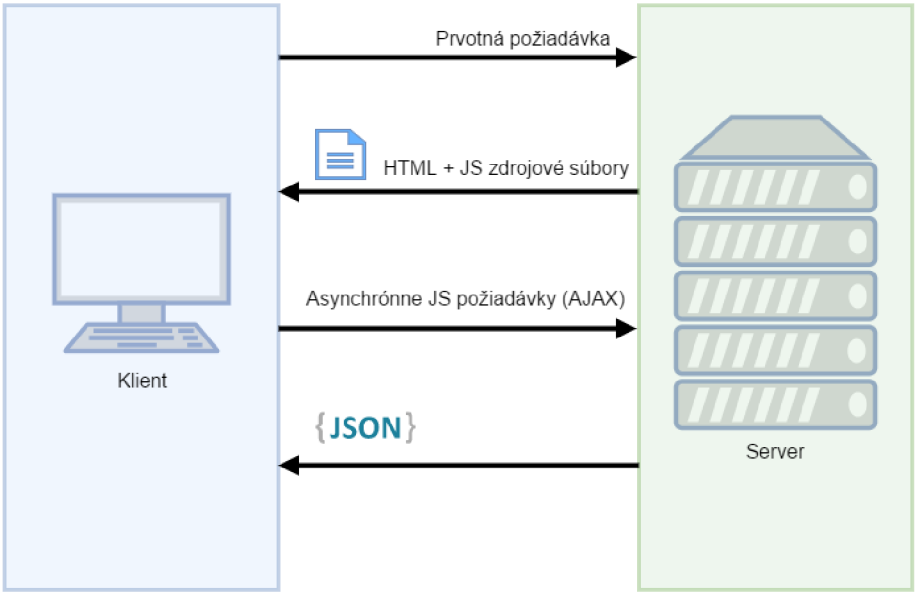
Webové aplikácie je možno rozdeliť na základe ich životného cyklu. Obrázok 3.2 popisuje životný cyklus tradičnej webovej aplikácie.



Obr. 3.2: Tradičný životný cyklus webovej aplikácie.

Základom tohto prístupu je server, ktorý sa stará o spracovanie html stránky, ktorá sa posiela klientovi. Nevýhoda tohto prístupu je, že server musí vyžiadanú stránku neustále znovu vypracovať. Je teda potrebné neustále zasielať veľmi podobné dáta. Za ďalšiu nevýhodu je možné považovať bezstavovosť protokolu HTTP[1]. Keďže protokol HTTP je bezstavový a teda jednotlivé žiadosti a odpovede na sebe prirodzene nezávisia, je nutné využívať sedenia. Tieto sedenia udržiavajú informácie o konkrétnom klientovi na strane serveru.

[1] https://tools.ietf.org/html/rfc2616



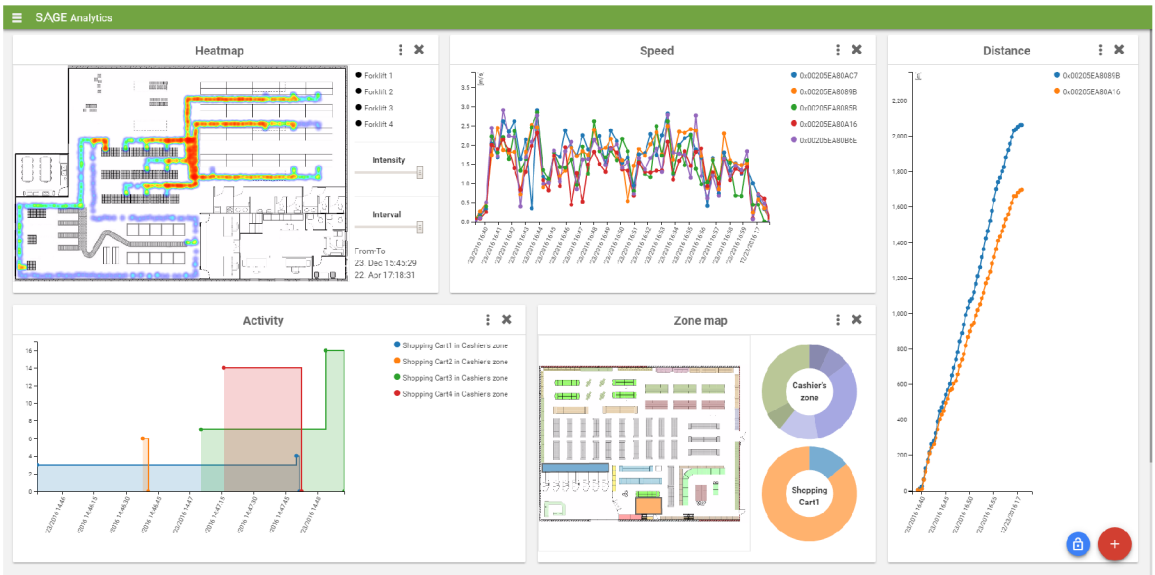
Obr. 3.3: Životný cyklus jednostránkovej webovej aplikácie.

Obrázok 3.3 popisuje prístup, ktorý je založený na tzv. SPA (z angl. Single-Page Application) princípe [8]. V tomto prípade je stav aplikácie uložený sa strane klienta. Tento prístup je použitý aj pre náš ovládací panel. Jeho základom je, že po prvotnej požiadavke na aplikáciu sa klientovi pošlú všetky potrebné súbory pre vytvorenie aplikácie v prehliadači. Tieto dáta sú spravidla optimalizované, aby zaberali minimum pamäte. Zdrojové súbory interpretovaného jazyka sú zbavené prebytočných znakov a následne sú komprimované. Týmto spôsobom môžeme dosiahnuť až 75 percentné zmenšenie výstupného súboru [4]. Takéto dáta sú potom udržiavané v cache pamäti webového prehliadača. Po prijatí na strane klienta začne spracovávanie zdrojového kódu. Tento kód je spravidla v jazyku JavaScript a teda je nevyhnutné, aby ho klientský webový prehliadač podporoval. Hlavná výhoda tohto prístupu je veľmi dobrá odozva na žiadosti používateľa. Aplikácia je schopná v reálnom čase meniť jednotlivé prvky ovládacieho panelu. Taktiež samotný vývoj aplikácie sa viac podobá klasickej, či už mobilnej alebo desktopovej aplikácii.

Výber frameworku prípadne knižnice, ktorý podporuje vyššie spomenutý životný cyklus webovej aplikácie, bol zúžený na dve možnosti. Prvou možnosťou je knižnica React[2] a druhou je framework Angular 2[3]. Ako vhodnejší bol zvolený framework Angular 2. Prevážili výhody akými sú napríklad jazyk, v ktorom sú písané zdrojové kódy, tým je jazyk TypeScript[7]. Pre prácu s knižnicou React je taktiež možné využívať tento jazyk, ale nie je to prirodzený a odporúčaný postup. Zvlášť pre nováčikov v tejto oblasti. Ďalšou výhodou je tzv. "two-way binding". To znamená že komponenty, z ktorých je zložená webová aplikácia vedia prijímať dáta, ale aj produkovať výstup. Tento výstup je v podobe udalostí, na ktoré sa môže nadradený komponent naviazať.

TypeScript je open-source jazyk udržiavaný firmou Microsoft. Do prostredia používateľských rozhraní vnáša veľkú silu objektovo orientovaného návrhu. Taktiež pridáva statické

[2] https://facebook.github.io/react/
[3] https://angular.io/docs/ts/latest/



Obr. 3.4: Ovládací panel s nakonfigurovanými vizualizačnými prvkami.

typovanie. Keďže tento jazyk nie je podporovaný v prehliadačoch, je potrebné ho transkompilovať, to znamená, že tieto zdrojové kódy sa prepíšu do jazyku JavaScript.

Ako hlavná knižnica pre potreby zobrazenia dát bola zvolená knižnica D3.js. Táto knižnica poskytuje rozhranie pre efektívnu manipuláciu s elementami na základe vstupných dát. Hlavnými oblasťami s ktorými knižnica D3 pracuje, je HTML, CSS a SVG [3]. Voľba tejto knižnice priniesla výhody v podobe flexibility, čo umožňuje prispôsobiť každý detail výsledného grafu. Ďalšou výhodou použitia knižnice D3 je vysoká rýchlosť. Vďaka týmto vlastnostiam je možné s grafom manipulovať v reálnom čase a výsledné grafy sú interaktívne. Knižnica používa funkcionálny zápis kódu, čo prináša dobrú čitateľnosť kódu.

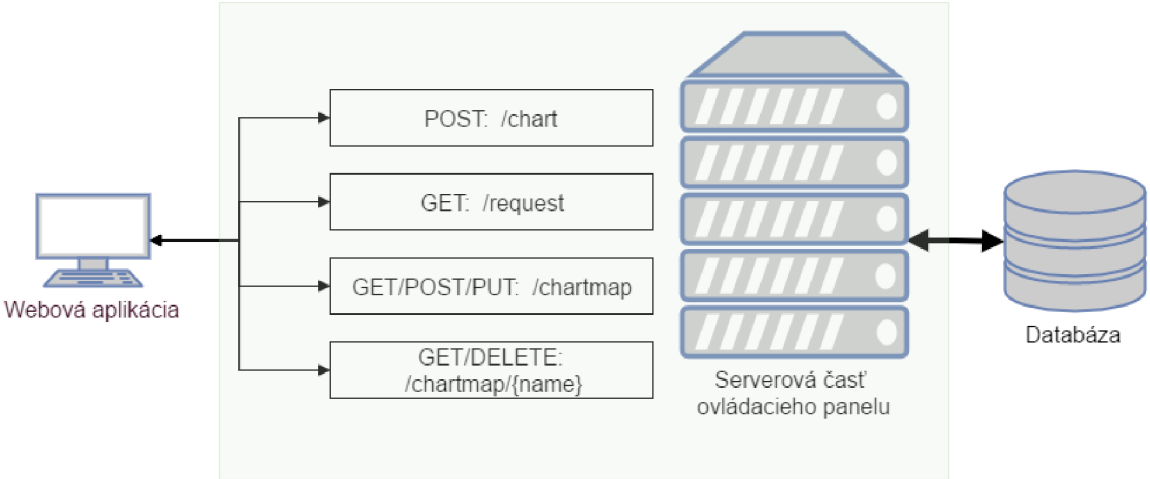
Používateľ pri prvom kontakte s aplikáciou uvidí takmer prázdnu plochu. Tým je naznačené, že aplikácia sa prispôsobuje používateľovi. Následne si používateľ vytvára jednotlivé prvky ovládacieho panelu a týmto prvkom priraďuje vlastnosti. Obrázok 3.4 približuje výslednú podobu webovej aplikácie. Obrázok zachytáva aplikáciu s plne zaplnenou plochou. Ovládací panel má možnosť sa ďalej vertikálne rozširovať. Ďalšie súčasti aplikácie budú popísané v kapitole 6.

3.3 Serverová časť ovládacieho panelu

Táto časť systému je bežnému používateľovi skrytá. Sú to tie prvky, ktoré ležia na strane servera. V tomto prípade je to webové rozhranie, teda aplikácia, ktorá zverejňuje prístupové body, s ktorými môžeme pracovať. K týmto bodom je možné pristupovať pomocou URL[4] adries. Toto rozhranie sa snaží splňovať zásady REST návrhu. Obrázok 3.5 približuje aktuálne možnosti práce s aplikačným rozhraním serveru. Na obrázku sú prístupové body delené do štyroch skupín. Tieto skupiny predstavujú:

1. /chart - Prijíma JSON objekt, ktorý špecifikuje vlastnosti požadovanej vizualizácie. Na základe týchto vlastností sformuje požiadavku na databázu a získané dáta ďalej

[4] https://tools.ietf.org/html/rfc1738



Obr. 3.5: Prezentácia serverového aplikačného rozhrania pre potreby ovládacieho panelu.

spracováva. Po spracovaní sú klientovi poskytnuté všetky potrebné dáta pre tvorbu vizualizácie.

2. /request - Pomocou tohto prístupového bodu klient špecifikuje vlastnosti požadovanej vizualizácie. Pre vytvorenie jedného vizualizačného prvku, si je potrebné so serverom vymeniť tri až štyri správy. Stav komunikácie sa prenáša v parametroch požiadavky.

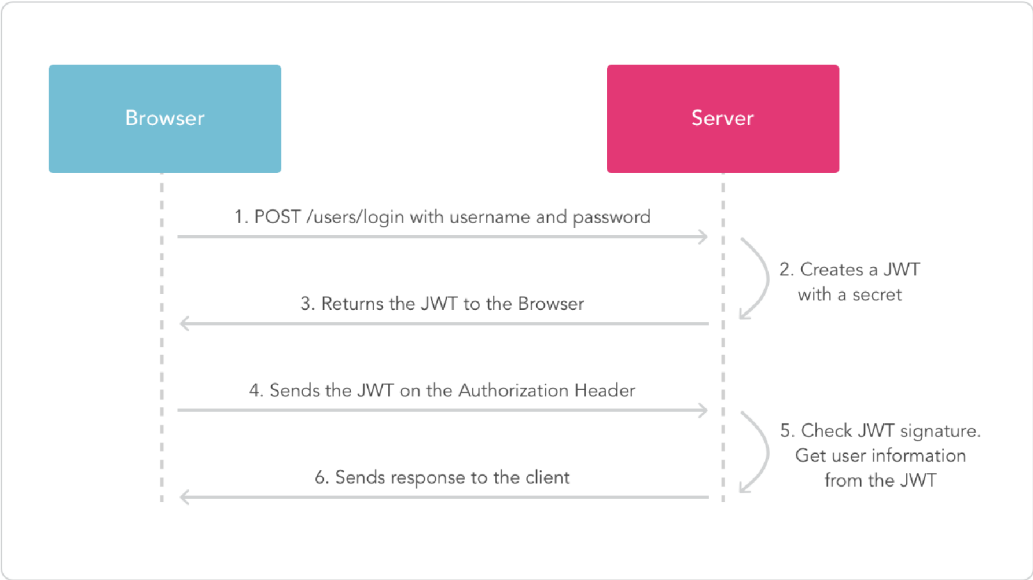
3. /chartmap a /chartmap/name - Slúži pre správu uloženej konfigurácie ovládacieho panelu. Rozmiestnenia a vlastnosti prvkov vizualizácie sa ukladajú do databázy.

Výber technológií pre túto časť ovládacieho panelu bol ovplyvnení súčasnou infraštruktúrou RTLS spoločnosti Sewio. Primárnym jazykom bol jazyk PHP[5]. Bolo teda prirodzene, že výber technológií sa bude zameriavať na jednoduché aplikačné rozhranie implementované v tomto jazyku.

Medzi najpopulárnejšie frameworky zamerané na túto oblasť použitia patria Silex[6], Slim[7] alebo Lumen[8]. Pre spracovanie komunikácie na strane serveru bol vybraný framework Slim, vďaka jeho rozsiahlej používateľskej komunite. Výber ovplyvnili taktiež predošlé pozitívne skúsenosti s týmto frameworkom. Medzi hlavné výhody tohto frameworku spadá jeho rýchlosť, dobrá dokumentácia a pokrytie všetkých potrieb ovládacieho panelu.

Pre účely automatických testov je používaný framework PHPUnit[9]. S pomocou tohto frameworku je možné vytvárať unit testy, pripadne integračné testy. Framework umožňuje aj testovanie kódu, ktorý pracuje s databázou. To prebieha vytvorením schémy databázy a následným spustením testu. Testovaný kód upraví túto databázu a jej stav sa automaticky porovná s požadovaným stavom databázy, ktorý je pripravený vo forme dokumentu XML[10].

[5] https://secure.php.net/
[6] http://silex.sensiolabs.org/
[7] https://www.slimframework.com/
[8] https://lumen.laravel.com/
[9] https://phpunit.de/index.html
[10] https://www.w3.org/XML/



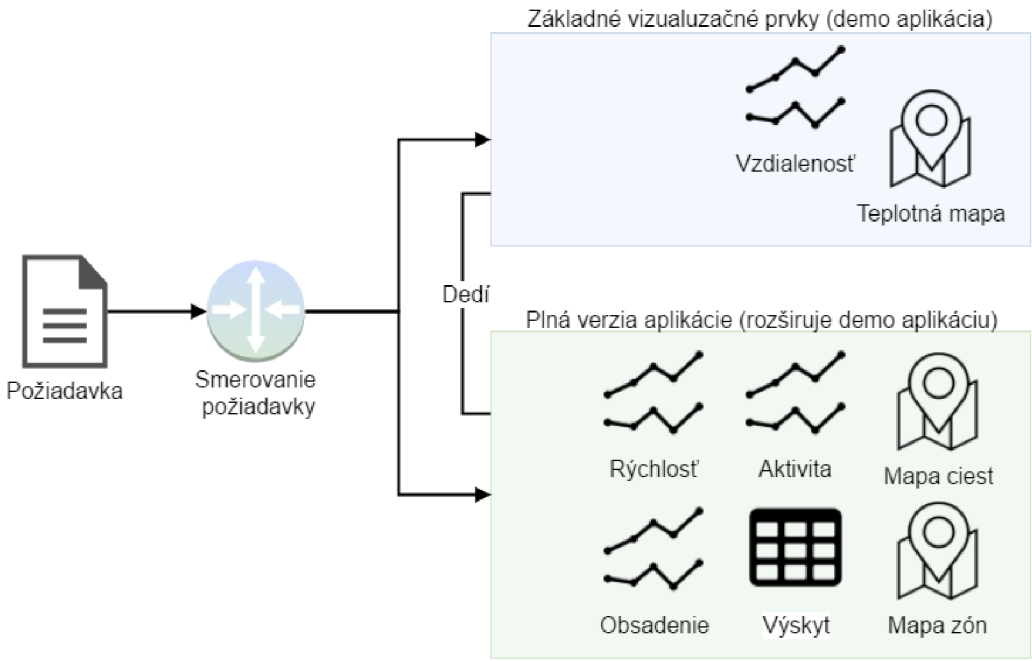
Obr. 3.6: Prezentácia vytvorenia a následného použitia tokenu, pre účely bezpečnej komunikácie medzi klientom a serverom.

3.4 Možnosti zabezpečenia

Aplikácia môže byť dostupná širokému okruhu ľudí. Keďže ide o jedno stránkovú aplikáciu (SPA), je vhodné, aby používateľ zadal prístupové údaje a až následne získal prístup k tejto stránke. Pre potreby autorizácie či autentizácie sa naskytajú dve riešenia. Prvým je jednoduchá autentifikácia, kedy si aplikácia v sebe udržiava prístupové údaje používateľa a pri každej požiadavke na dáta sa znovu autentifikuje. Druhá možnosť je autorizácia a autentifikácia pomocou tokenov. Táto možnosť je preferovaná, je bezpečnejšia, konkrétne tým, že klientská aplikácia si neukladá konkrétne prihlasovacie údaje, ale má uložený iba token.

Keďže aplikácia bude využívaná v prostredí intranetu, nároky na zabezpečenie sa líšia od aplikácií pôsobiacich na internete. Aktuálna implementácia spolieha na autentifikáciu pomocou API kľúčov súčasného RTLS [5]. Napriek tomu táto kapitola približuje možnosti rozsiahlejšieho zabezpečenia pomocou tokenov. Token má formu hashu. Ten je rozdelený do troch častí, ktoré sú oddelené bodkou. Prvá časť je hlavička. Tá sa zvyčajne skladá z typu hashovacieho algoritmu, ktorý bol použitý, napríklad HS256 (HMAC SHA256) a typu tokenu, napríklad JWT (JSON Web Token). Druhou časťou sú dáta, ktoré obsahujú metadáta o tokene a informácie, ktoré token prenáša. Sú to napríklad dodatočné informácie o používateľovi. Treťou časťou je podpis, ktorý potvrdzuje, že tento token nebol zmenený cestou k cieľu [2]. Výhoda tokenu je aj to, že je sebestačný, teda obsahuje všetky informácie v sebe, tým pádom sa zmenší počet požiadaviek na databázu.

Obrázok 3.6 popisuje situáciu, v ktorej klient pomocou zaslaných prihlasovacích údajov požiada o vytvorenie tokenu. Server overí prihlasovacie údaje a vytvorí mu príslušný token. Následne s jeho pomocou je používateľ schopný žiadať o ďalší chránený obsah bez toho, aby sám vedel svoje prihlasovacie údaje. Ovládací panel pracuje s citlivými informáciami, ako napríklad pohyby osôb. Takéto dáta je preto potrebné zabezpečiť. Týmto spôsobom je možné chrániť určité dáta a sprístupniť ich len tým používateľom, ktorí majú na tieto dáta právo. Tokenom je tiež možné časovo obmedziť schopnosť prístupu k týmto dátam.

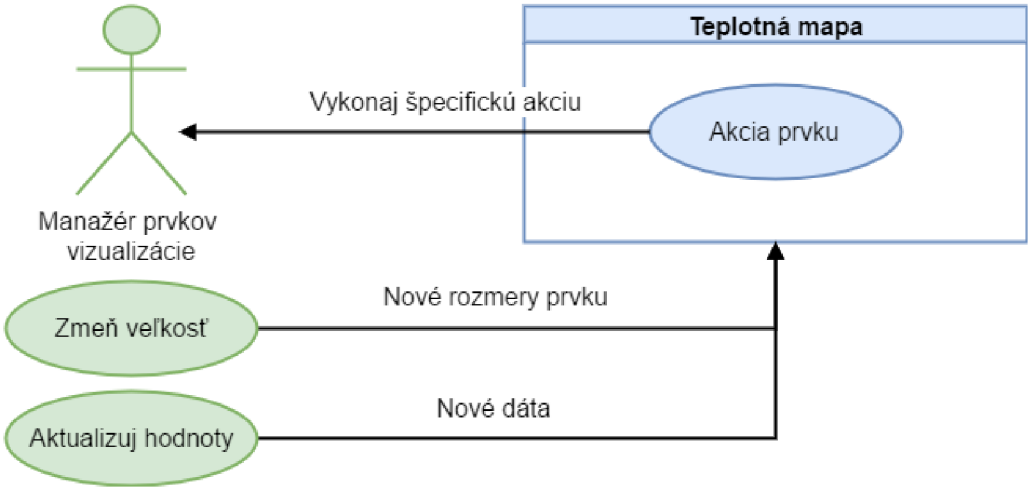


Obr. 3.7: Prezentácia smerovania požiadavky v prípade demo a plnohodnotnej aplikácie.

3.5 Implementácia

Kapitola približuje niektoré zaujímavé detaily aplikácie, ktoré bolo nutné implementovať pre špecifické požiadavky tejto práce. V nasledujúcom texte bude popísaný dôvod a spôsob rozdelenia aplikácie na demo a plnohodnotnú verziu. V závere sú popisované nástroje použité pre vývoj popisovanej aplikácie a pre samotnú tvorbu jednotlivých vizualizácií.

Práca je stavaná ako webová aplikácia, ktorá je implementovaná na strane serveru v jazyku PHP. Tento jazyk je interpretovaný a keďže sa takáto aplikácia nachádza v prostredí intranetu, tak zákazník má prístup k zdrojovým kódom programu. To môže byť problém v prípade, že zákazník má mať k dispozícií iba ukážkovú aplikáciu, ktorá podporuje len dva druhy reprezentácie dát. Obrázok 3.7 prezentuje rozdelenie aplikácie na demo a plnohodnotnú aplikáciu. Ako je vidieť plnohodnotná aplikácia obsahuje rozšírené možnosti reprezentácie dát, ale zároveň má prístup aj k reprezentáciám v demo aplikácií. Implementačne je takéhoto správania dosiahnuté tak, že kód pre jednotlivé reprezentácie je rozdelený do dvoch tried. Prvá je základná a je reprezentovaná v modrej oblasti na obrázku 3.7. Druhá, rozširovacia trieda dedí chovanie základnej triedy a je znázornená v zelenej oblasti. Tieto triedy majú rovnaké rozhranie a na začiatku programu je rozhodnuté, ktorá trieda sa použije. Takéto rozdelenie umožnilo rozdeliť zdrojové kódy programu na dve časti a v prípade, že zákazník má nárok len na demo aplikáciu, zdrojové súbory rozšírených funkcií mu nie sú poskytnuté. Klientskej časti aplikácie je následne poskytnutá informácia o verzií aplikácie a v prípade demo aplikácie, vie používateľovi naznačiť, ku akým funkciám nemá prístup.



Obr. 3.8: Prezentácia vzťahu medzi manažérom vizualizačných prvkov a samotným prvkom.

Na strane klienta bude popísaná komunikácia medzi časťou aplikácie, ktorá sa stará o správu vizualizačných prvkov a samotnými prvkami. Táto správa je nutná z dôvodu, že návrh aplikácie počítal s rozdelením jednotlivých dátových reprezentácií do oddelených blokov. Tieto bloky vytvára používateľ a následne s nimi manipuluje. Obrázok 3.8 zobrazuje tzv. manažéra prvkov (zelenou farbou) a jeho udalosti, ktorými ovplyvňuje konkrétny prvok. Udalosť, ktorá prikáže konkrétnemu prvku, aby sa prispôsobil zmene jeho veľkosti nastáva v prípade, kedy používateľ explicitne zmení veľkosť prvku alebo v prípade, kedy musia prvky reagovať na zmenu veľkosti okna prehliadača. Táto udalosť v sebe nesie nové rozmery, ktorým sa má prvok prispôsobiť a teda zmeniť svoju vnútornú reprezentáciu polohových dát. Druhou udalosťou, ktorá môže nastať týmto smerom je udalosť, kedy sú vizualizačnému prvku poskytnuté nové dáta. Táto udalosť nastáva v prípade úpravy vlastností už vytvorených prvkov. Tieto dve udalosti tvoria rozhranie každého prvku vizualizácie a tieto prvky naň musia správne reagovať.

Aj samotné prvky ale musia byť schopné komunikovať s popisovaným manažérom. Táto komunikácia je založená na špeciálnom formáte správy. Táto správa obsahuje:

1. Názov akcie - každá akcia má unikátny názov, podľa ktorého manažér akciu rozpozná a správne na ňu reaguje.

2. ID prvku - identifikácia prvku, ktorý konkrétnu akciu požaduje.

3. Dáta - niektoré akcie môžu obsahovať dáta, ktoré súvisia s daným typom akcie.

Spôsob komunikácie pomocou akcií bol zvolený z dôvodu, že jednotlivé prvky vizualizácie môžu mať k dispozícii rozdielne akcie. Všeobecné akcie sú popísané v kapitole 6.2.

Nasledujúci text sa venuje použitým nástrojom pre tvorbu webovej aplikácie. Následne bude popísaný spôsob použitia vybraných knižníc pre tvorbu dátových reprezentácií.

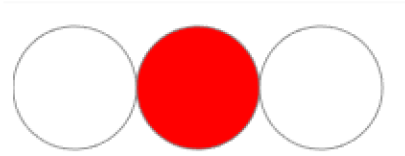
V rámci vývoja bol použitý nástroj angular-seed[11] pre automatický preklad zdrojových súborov. Tento nástroj poskytuje podporu pre prípravu zdrojových súborov do vývojového a produkčného stavu. Ďalej poskytuje nenáročný vývojový server a nástroje pre automatické testovanie webovej aplikácie.

Hlavná knižnica pre tvorbu interaktívnych vizualizácií je vyššie spomínaná knižnica D3.js. Výsledky použitia tejto knižnice budú popísané v kapitole 4. Knižnica slúži pre transformáciu dát do podoby objektového modelu dokumentu, teda DOM(z angl. Document Object Model). Prakticky to znamená napríklad transformáciu jednoduchého poľa hodnôt do podoby tabuľky. V momente, kedy sú hodnoty poľa prevedené do tejto reprezentácie, nastáva pri akejkoľvek zmene vstupných dát k príslušnej aktualizácií tejto tabuľky. K tejto funkcionalite pridáva knižnica aj možnosti naviazania sa na jednotlivé udalosti webového prehliadača alebo vytváranie prechodov medzi stavmi pomocou animácií. Popisované transformácie a udalosti sú potom využité pre manipuláciu s vizualizáciou v reálnom čase. Taktiež je možné kompletne zmeniť dáta prvku, ktorý je už vykreslený. Pomocou tejto knižnice bola vytvorená väčšina podporovaných vizualizácií. Výnimku tvorí teplotná mapa, pre ktorú bola použitá knižnica SimpleHeat.js[1]. Táto knižnica bola zvolená z dôvodu, že teplotnú mapu je výkonnostne náročne vykresľovať formou SVG, teda vektorovo. Ukážka 3.1 reprezentuje kód, ktorý na základe vstupného poľa (prvý riadok) generuje príslušný počet kruhov.

```
1  var vstupneData = [1,2,3];
2
3  var elementSVG = d3.select("#htmlKontainer")
4          .append("svg")
5          .attr("width", 300)
6          .attr("height", 100);
7
8  elementSVG.selectAll("circle")
9      .data(vstupneData)
10     .enter().append("circle")
11     .style("stroke", "gray")
12     .style("fill", "white")
13     .attr("r", 40)
14     .attr("cx", function(data, index){return index*80 + 40})
15     .attr("cy", 50)
16     .on("mouseover", function(){d3.select(this).style("fill", "red");});
```

Ukážka 3.1: Kód, ktorý na základe vstupného poľa generuje príslušný počet kruhov.

[11] https://github.com/mgechev/angular-seed



Obr. 3.9: Výsledná vizualizácia, vytvorená kódom z ukážky 3.1. Stredný kruh zmenil farbu po udalosti, kedy používateľov kurzor prechádzal cez tento prvok.

Kód predvádza vizualizáciu, riadenú vstupnými dátami. Na riadku 3 je definovaný SVG element, ktorý je vložený do HTML elementu s identifikátorom *htmlKontainer*. Definovaná je jeho šírka a výška. Následne na riadku 8 sú do tohto elementu pripájané ďalšie, už SVG špecifické elementy. Počet elementov *circle* je definovaný na základe vstupného poľa, ktoré je predané na riadku 9. Následne sa každému vytvorenému elementu definuje farba a veľkosť (riadky 11 až 13). Na riadku 14 je nutné predanie funkcie, ktorá počíta pozíciu vytvoreného kruhu. Funkcia dostáva prvým parametrom dáta, v našom prípade je to hodnota zo vstupného poľa a index, určujúci poradie kruhu. Z tohto poradia je možne vypočítať posun jednotlivých kruhov. Na záver je definovaná udalosť, ktorá nastane v momente kedy používateľ prejde kurzorom cez konkrétny kruh. Obrázok 3.9 reprezentuje výsledok popisovaného kódu v stave kedy používateľ prešiel kurzorom cez prostredný kruh.

Kapitola 4

Možnosti vizualizácie dát

Vizualizácia zozbieraných dát je hlavnou úlohou vyvíjanej aplikácie. Súčasný systém spoľahlivo pracuje a jeho funkcionality ako je zber dát, spresnenie zozbieraných dát, ako aj iné funkcionality sú reálne využívané. Koncoví užívatelia si ale tieto dáta museli spracovať vo vlastnej réžii. Ovládací panel sa snaží tento problém riešiť a vytvoriť jednotné rozhranie pre vytváranie vlastných zobrazovacích panelov. Výstup aplikácie sa nezameriava na konkrétnu oblasť, ale snaží sa zameriavať na široké možnosti použitia. Druhy vizualizácie sa delia do troch skupín. Jednoduché, kedy je používateľovi poskytnutá jednoduchá informácia, napríklad v podobe časového údaju prvého vstupu do špecifickej oblasti. Nasledujú grafové vizualizácie, ktoré sa vťahujú na určitý čas a zobrazujú k tomuto času informácie, napríklad celkovú prejdenú vzdialenosť, prípadne priemernú rýchlosť. Posledným druhom sú mapové vizualizácie, ktoré poskytujú dobrú reprezentáciu polôh, vzhľadom k monitorovanej miestnosti. Všetky implementované vizualizácie boli navrhnuté na základe požiadavok od zákazníkov z troch vertikál, a tými sú logistika, obchod a šport.

4.1 Združovanie objektov a delenie monitorovanej oblasti

Lokalizačný systém poskytuje ovládaciemu panelu pre každý tag rovnakú formu polohových dáta. Vzťahy medzi jednotlivými monitorovanými objektami nerozlišuje. Ako príklad môžme uviesť basketbalový zápas, v ktorom chceme porovnať výkonnosť jednotlivých tímov. Tagy z jedného tímu preto potrebujeme združiť do vlastnej skupiny. Vizualizácia musí brať túto funkcionalitu na vedomie a rozdielne reagovať pri tvorbe vizualizácie. V nasledujúcich druhoch zobrazenia bude vdžy popísaná zmena reprezentácie dát medzi vizualizáciou tagu a vizualizáciou skupiny. Ako ďalší príklad využitia združovania tagov do skupín, môže byť poslúžiť prostredie obchodu. V takomto prípade je potrebné, aby bol tagom pomocou skupín špecifikovaný účel, napríklad skupina bezpečnostnej služby, skupina zamestnancov alebo skupina nákupných vozíkov. Zamestnanci môžu byť následne delení do špecifickejších podskupín. Po takomto zoskupení je analýza špecifických polohových dát lepšie využiteľná.

Podobne ako v predošlom prípade, pri zbere polohových dát sa RTLS systém nezameriava ani na informáciu, v akej časti monitorovacej oblasti sa tag nachádza. Táto informácia sa prideľuje až v následnom spracovaní dát. Existujúca infraštruktúra už obsahuje nástroj pre tvorbu zón. Ovládací panel je ale plne pripravený túto funkciu prebrať a následne rozšíriť. Zóna rozdeľuje monitorovanú oblasť, vizualizácia je následne schopná s týmito zónami pracovať. Zóny nie sú tvorené jednoduchými štvoruholníkmi, ale sú tvorené polygónmi.

Tag/Category	Main Corridor	Side Corridor
Forklift 1	Fri Dec 23 2016 15:45:40	Fri Dec 23 2016 15:46:30
Forklift 2	Fri Dec 23 2016 15:45:36	
Forklift 3	Fri Dec 23 2016 15:45:33	
Forklift 4		
Forklifts_1-4	Fri Dec 23 2016 15:45:33	Fri Dec 23 2016 15:46:30

Obr. 4.1: Tabuľka poskytujúca informáciu o prvom výskyte vo vybraných zónach

4.2 Jednoduché vizualizačné prvky

V súčasnej dobe do kategórie jednoduchých vizualizácií patrí jedine tabuľka. Konkrétne sa jedná o metriku prvého vstupu do zóny a metriku posledného opustenia zóny. Do tejto kategórie ale spadajú napríklad aj jednoduché ukazovatele hodnôt. Ako príklad by bolo možne uviesť ukazovateľ hodnoty výdrže batérie v jednotlivých tagoch. V súčasnej dobe sa ale ovládací panel zameriava výhradne na vizualizáciu polohových dát.

4.2.1 Prvý a posledný výskyt v zóne

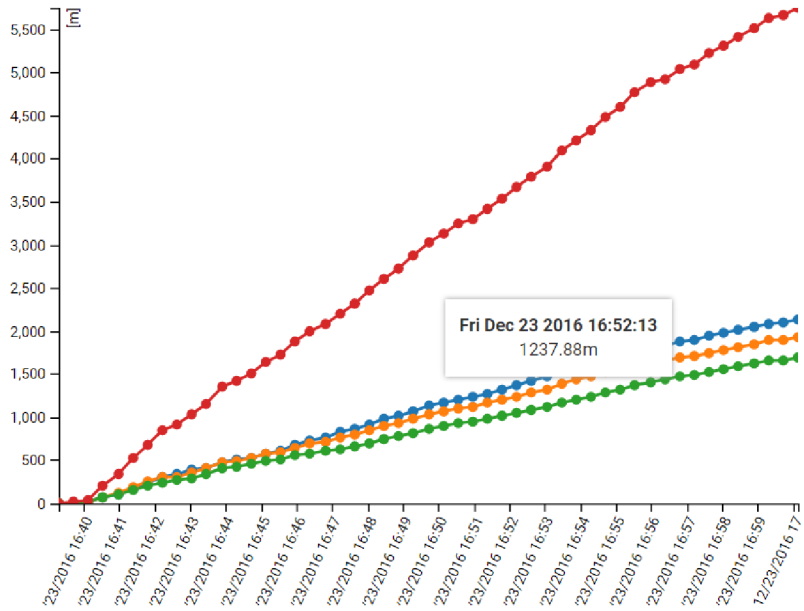
Na obrázku 4.1 je zachytená tabuľka, ktorá v hlavičke definuje množinu zón, pre ktoré poskytuje čas prvého vstupu do tejto zóny. Táto množina nie je obmedzená. V prvom stĺpci sú definované tagy a skupiny, pre ktoré sa informácia o prvom vstupe do zóny vyžaduje. V prípade že tag vo vyhradenom časovom intervale do tejto zóny nevstúpil, bude toto miesto v tabuľke prázdne.

Výsledné časy pre skupiny predstavujú združenú informáciu. Skupina Forklifts_1-4 na obrázku 4.1 zahŕňa 4 vyššie zobrazené objekty. V prípade metriky prvého vstupu do zóny sa pre jednotlivé zóny zobrazí čas toho tagu, ktorý ako prvý vstúpil do danej zóny. V prípade metriky posledného opustenia zóny sa zobrazí čas posledného tagu, ktorý opustil túto zónu.

4.3 Reprezentácia dát pomocou grafov

Tieto reprezentácie sa z pravidla viažu na čas. V súčasnej dobe sú podporované štyri grafové reprezentácie dát, a to:

1. Vzdialenosť - výpočet celkovej prejdenej vzdialenosti.
2. Aktivita - zaznamenanie aktivity tagu v danej zóne.
3. Obsadenie - zaznamenanie prítomnosti tagu v špecifickej zóne.
4. Rýchlosť - výpočet priemernej rýchlosti.

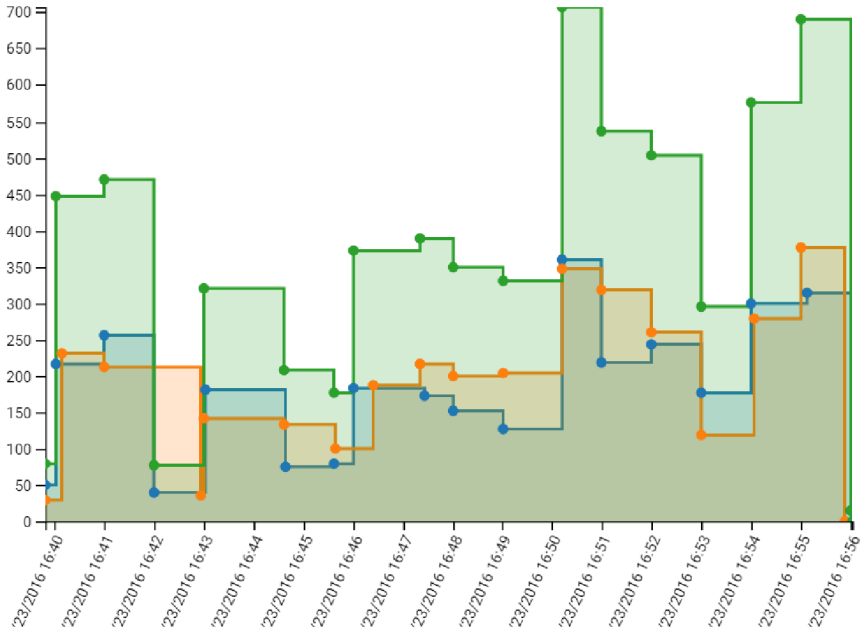


Obr. 4.2: Vzdialenosť - graf reprezentujúci rovnomerný nárast prejdenej vzdialenosti v čase.

4.3.1 Vzdialenosť

V situácii, kedy potrebujeme určiť celkovú vzdialenosť, ktorú monitorovaný objekt prekonal, nastáva nasledujúci problém. Približná prejdená vzdialenosť sa obvykle počíta ako súčet vzdialeností medzi jednotlivými bodmi z lokalizačného zariadenia. Problém ale nastáva v prípade, keď tag nemení svoju pozíciu. V tomto prípade, získané polohy nie sú rovnaké, ale vplyvom šumu sa generujú v okolí jedného bodu. Po uplynutí niekoľkých sekúnd sú rozdiely medzi týmito chybami sčítané a môžu deklarovať prejdenú vzdialenosť. Za niekoľko sekúnd môže táto chyba dosiahnuť aj jeden meter. Existuje ale aj druhá chybná situácia, tou je skoková chyba v dátach. Takáto chyba dokáže znehodnotiť výslednú vzdialenosť. Toto môže byť spôsobené chybou merania, alebo pri strate signálu, kedy nie sú k dispozícii informácie o polohe tagu. Vtedy sa táto vzdialenosť nezapočíta do prejdenej vzdialenosti. O toto spracovanie sa stará serverová časť práce a musí na takéto situácie správne reagovať.

Obrázok 4.2 prezentuje vizualizáciu prejdenej vzdialenosti. V spodnej časti grafu sa nachádza vizualizácia prejdenej vzdialenosti pre tri tagy. Ich dosiahnutá vzdialenosť sa postupom času odlišuje. Červenou farbou je následne zobrazená skupina obsahujúca tieto tri tagy a jej hodnota predstavuje sčítanie jednotlivých vzdialeností jej členov.



Obr. 4.3: Aktivita - graf reprezentujúci aktivitu tagu v závislosti na čase. Vertikálna os popisuje počet zozbieraných pozícií.

4.3.2 Aktivita

Metrika monitorujúca aktivitu tagu. Táto metrika rozdelí zadaný interval na minútové intervaly. V týchto intervaloch spočíta počet získaných pozícií. Pomocou tejto metriky je možné v čase vidieť jeho aktivitu. Tag nemusí meniť svoju pozíciu, stačí že akcelerometer, ktorý je zabudovaný v tagu zaznamená pohyb. Následne je možné porovnávať aktivitu jednotlivých tagov.

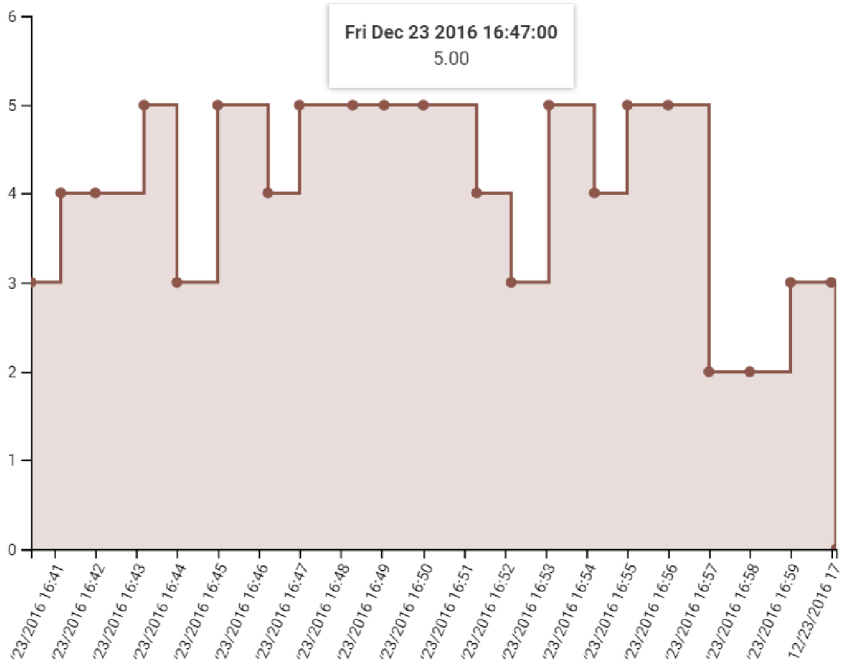
Metrika taktiež podporuje prácu zo zónami a je teda možné v jednom grafe zobraziť aktivitu tagov v jednotlivých zónach. V prípade využitia združovania tagov do skupín je výsledkom v jednotlivých intervaloch súčet hodnôt tagov vo vybranej skupine.

Obrázok 4.3 predstavuje vizualizáciu dvoch tagov v špecifickej zóne. Tieto tagy sú znázornené modrou a oranžovou farbou. Zelená farba predstavuje skupinu, ktorej členmi sú tieto dva tagy. Je možné vidieť že táto skupina mapuje súčet modrého a oranžového tagu.

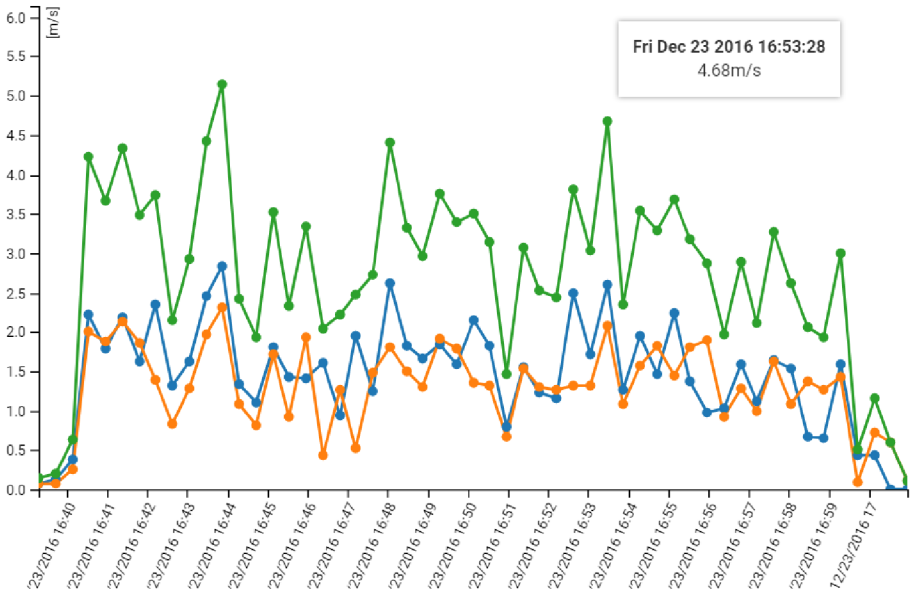
4.3.3 Obsadenie

Metrika aktuálneho obsadenia zóny zobrazuje, či je vybraná zóna obsadená tagom. V jednom grafe je opäť možné vybrať viacero tagov a zón. Výsledný graf v takomto prípade zobrazí hodnotu 1 v prípade prítomnosti tagu v určitom intervale a 0 v prípade neprítomnosti. Celkový vybraný časový interval je opäť rozdelený na menšie minútové úseky. Zaujímavejší výstup tohoto grafu uvidíme pri združovaní tagov do skupín. V takomto prípade vidíme číslo, reprezentujúce počet tagov z vybranej skupiny, ktoré sa nachádzajú v konkrétnej zóne. Takto je možné efektívne porovnávať zaplnenie jednotlivých zón na jednom grafe.

Obrázok 4.4 obsahuje skupinu piatich tagov. V minútových intervaloch následne poskytuje aktuálne zaplnenie zóny.



Obr. 4.4: Obsadenie - zobrazenie počtu jednotlivých monitorovaných objektov v zónach. Pomocou tohto grafu môžme zistiť obsadenie danej zóny v konkrétnom čase.



Obr. 4.5: Rýchlosť - graf popisuje priemernú rýchlosť monitorovaných tagov.

4.3.4 Rýchlosť

Metrika výpočtu priemernej rýchlosti rozdeľuje vybraný časový interval na špecifický počet rovnakých časových úsekov. Počet týchto úsekov sa pohybuje okolo hodnoty 60. Takýto počet je dobre vizualizovateľný pri všetkých veľkostiach prvku. Veľkosť tohto úseku je priamo úmerná veľkosti vybraného časového intervalu. V každom tomto úseku je vypočítaná prie-

merná rýchlosť. Tieto úseky sú následne vykreslené do grafu, kde užívateľ sleduje zmenu tejto priemernej rýchlosti v čase. V prípade výberu celej skupiny je výsledkom vizualizácie súčet jednotlivých rýchlosti tagov v tejto skupine.

Graf na obrázku 4.5 popisuje priemernú rýchlosť modrého a oranžového tagu. Hodnoty znázornené zelenou farbou predstavujú skupinu. Táto skupina obsahuje oba skôr spomínané tagy a jej hodnota v grafe predstavuje súčet rýchlostí týchto tagov.

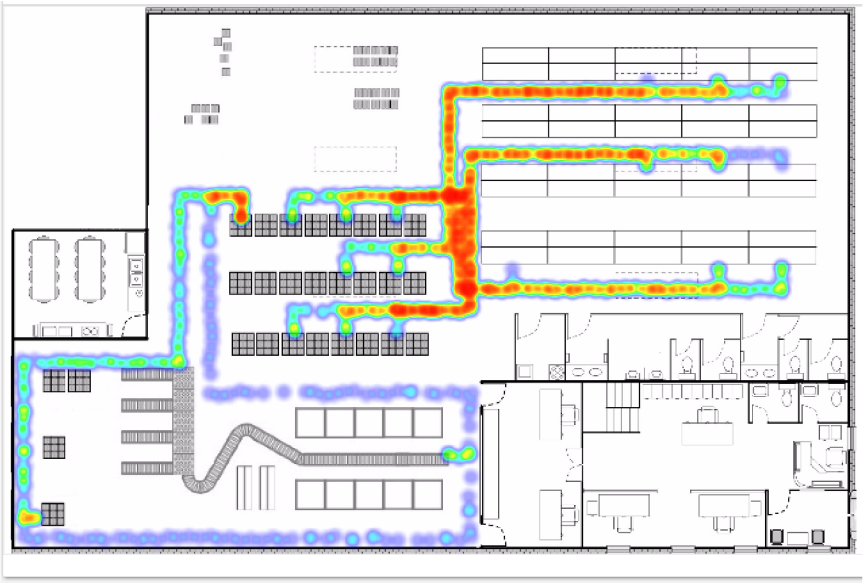
4.4 Reprezentácia dát pomocou máp

Metriky používajúce mapy poskytujú dobrú reprezentáciu polôh vzhľadom k monitorovanej miestnosti. S týmito mapami je rovnako ako s grafmi možné interaktívne manipulovať.

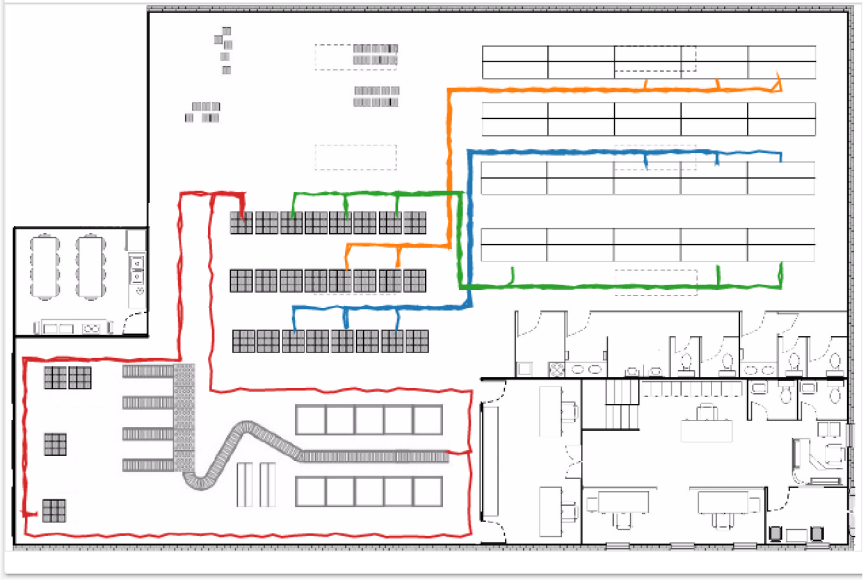
1. Teplotná mapa - vizualizuje miesta, v ktorých sa jednotlivé tagy pohybovali najčastejšie.
2. Mapa ciest - vizualizuje cesty, po ktorých sa jednotlivé tagy pohybovali.
3. Mapa zón - pre každý vybraný tag, poskytuje informáciu, koľko času strávil v danej zóne.

4.4.1 Teplotná mapa

Teplotná mapa sa používa pre zobrazenie polôh na dvojrozmernej mape. Polohy sú reprezentované pomocou bodov. Čím viac bodov je umiestnených na jednom mieste, alebo v tesnej blízkosti, tým viac sa mení farba v oblasti tohto miesta. Postupný prechod farieb z modrej až k červenej znázorňuje, kde sa jednotlivý tag najčastejšie pohyboval. Priamo integrované ovládacie rozhranie prvku umožňuje určovať veľkosť oblasti, v ktorej sa jednotlivé body ovplyvňujú. Ďalším možnostiam interakcie s mapovými vizualizáciami sa venuje kapitola 6.2.



Obr. 4.6: Teplotná mapa - pomocou zmeny farby, je mapa schopná vizualizovať miesta, v ktorých sa jednotlivé tagy pohybovali najčastejšie.



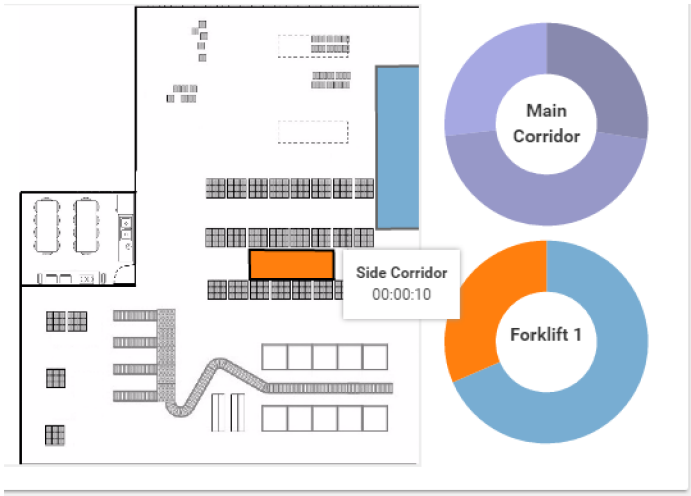
Obr. 4.7: Mapa ciest - vizualizuje cesty, po ktorých sa jednotlivé tagy pohybovali. Každému tagu je priradená špecifická farba cesty.

4.4.2 Mapa ciest

Mapa ciest, nazývaná aj ako špagety diagram (angl. spaghetti graph), vizualizuje cesty, po ktorých sa jednotlivé tagy pohybovali. Tagy je opäť možné združovať do skupín. Po združení viacerých tagov do skupiny, je táto skupina reprezentovaná jednou farbou. Tento druh vizualizácie je zobrazený na obrázku 4.7.

4.4.3 Mapa zón

Mapa zón slúži na porovnanie výskytu jednotlivých tagov v zónach. Používateľ vidí jednotlivé zóny na mape. Po kliknutí na zónu sa mu zobrazí koláčový graf, reprezentujúci čas všetkých tagov, ktoré sa vo vybranom časovom intervale nachádzali v tejto zóne. Popis obrázku 4.8 bližšie špecifikuje možnosti použitia tejto vizualizácie.



Obr. 4.8: Mapa zón - Interaktívna mapa zobrazujúca zóny. Po kliknutí na zónu sa zobrazí koláčový graf zachytávajúci strávený čas jednotlivých tagov v tejto zóne. Po kliknutí na výseč reprezentujúcu tag, sa vytvorí druhý graf pre tento konkrétny tag. Tento druhý koláčový graf znázorňuje v akých zónach sa vybraný tag pohyboval.

Kapitola 5

Databáza a redukcia dát

Databáza je hlavným prostriedkom pre ukladanie lokalizačných dát, metadát a ďalších konfiguračných informácií. Lokalizačný systém v reálnom čase zbiera dáta, reprezentujúce polohy jednotlivých tagov a ukladá ich do databázy. Tieto dáta sú neskôr vyžiadané pre potreby ovládacieho panelu. Po spracovaní vyžiadaných dát na strane servera, pre potreby vizualizačného prvku, sa tieto spracované informácie znovu neukladajú do databázy. Pre znovu načítanie prvkov vizualizácie je potrebné dáta znovu vyžiadať a spracovať.

V tejto kapitole sa nachádzajú informácie o výbere a úpravách databázy. Následne sú predstavené požiadavky pre vyhľadávanie dát z databázy. Popísané budu indexy, ktoré bolo nutné vytvoriť pre rýchle vyhľadávanie polohových dát. V závere bude predstavený algoritmus, ktorý pomáha redukovať objem pozícií s dôrazom na čo najmenší vplyv na výslednú vizualizáciu.

5.1 Špecifikácia databázy

Aplikácia využíva stávajúcu databázu, s ktorou pracuje lokalizačný systém. Dôvody výberu tohto riešenia budú vysvetlené. Aplikácia teda využíva MySQL relačný databázový server verzie 5.6, bežiaci na operačnom systéme Ubuntu 14.04. Pre potreby komunikácie s databázou je využívaná abstrakcia v podobe PDO (PHP data objects)[1].

Pri návrhu databázy bolo zvažované kompletné oddelenie databázy, s ktorou pracuje lokalizačný systém, od databázy, ktorá bude slúžiť ako zdroj historických dát. Takéto oddelenie by so sebou nieslo výhody v podobe možného nasadenia databáz na dve rozdielne serverové stanice. V prípade použitia jednej databázy by totiž mohol nastať problém v momente vyžiadania rozsiahlych historických dát. V takomto prípade by sa schopnosť databázy vybavovať požiadavky v reálnom čase mohla znížiť, čo sa ale pri testovaní nepotvrdilo. Nevýhodou tohto riešenia je aj potreba neustálej synchronizácie databáz. Táto synchronizácia by spôsobila duplicitné dáta v oboch databázach. Napríklad informácie o jednotlivých zónach by sa nachádzali v oboch databázach a bolo by zložitejšie ich aktualizovať. Taktiež by to znamenalo rozsiahly zásah do už existujúceho riešenia RTLS.

Jednou z možností odľahčenia záťaže pre databázu môže byť spôsob ukladania vytvoreného ovládacieho panelu. Ten by sa uložil aj so všetkými dátami, ktoré sú už prispôsobené a filtrované pre konkrétne prvky ovládacieho panelu. Týmto spôsobom by databázový server nemusel potrebné pozície vyhľadávať. Nevýhoda takéhoto riešenia je, že uložené dáta by neboli aktualizované novými dátami. Druhou nevýhodou by bolo omedzenie škálovateľnosti

[1] http://php.net/manual/en/book.pdo.php

výberu týchto dát. No najhlavnejším problémom takéhoto riešenia je objem uložených dát. Pri ukladaní výsledku každého prvku ovládacieho panelu by neustále vznikali duplicitné dáta.

Výsledný návrh používa jednu databázu. Pri ukladaní ovládacieho panelu pre potreby opätovného načítania, sa samotné lokalizačné dáta neukladajú. Ukladajú sa iba metadáta o rozložení jednotlivých prvkov a informácie aké dáta jednotlivé prvky požadujú. Do databázy sa teda uloží konfiguračný súbor, ktorý popisuje všetky požiadavky na dáta. Pri načítaní ovládacieho panelu sa tento konfiguračný súbor rozdelí na jednotlivé požiadavky pre každý prvok. Postupne sa tieto požiadavky spracúvajú a dáta sú opäť poskytnuté klientovi.

5.1.1 Úprava využívanej databázy

Kvôli vyššie spomenutým dôvodom výberu databáze bolo potrebné upraviť existujúcu databázu pre potreby ovládacieho panelu. Táto úprava spočívala v pridaní tabuliek pre ukladanie historických pozícií a pre ukladanie vytvorených konfigurácií.

Obrázok 5.1 popisuje tabuľky databázy, ktoré sú využívané ovládacím panelom. Na pravej strane obrázku sú predstavené tie tabuľky, ktoré zavedený systém už používa. Sú to tabuľky:

1. category - reprezentuje skupiny, do ktorých je možné tagy združovať. Taktiež obsahuje odkaz na rodičovskú skupinu, aby bolo možné vytvoriť strom skupín.

2. feed - tabuľka reprezentuje informácie o tagu, napríklad jeho alias alebo popis. Mimo to tabuľka obsahuje informácie o budovách, pomocou ktorých je možné na základe výberu tagu odvodiť plán, ktorý sa má zobraziť pri mapových vizualizáciách.

3. zones - táto tabuľka obsahuje informácie o jednotlivých zónach. Udržiava meno, typ a definuje polygón, z ktorého je tvorená. Medzi dôležité informácie patrí mierka, pomocou ktorej je možné určiť veľkosť zóny vzhľadom k plánu, ku ktorému patrí. Ďalším dôležitým parametrom je počiatočný bod, od ktorého sú pozície na pláne počítané. Od tohto bodu sa odvíja aj samotné vykreslenie zón na plán.

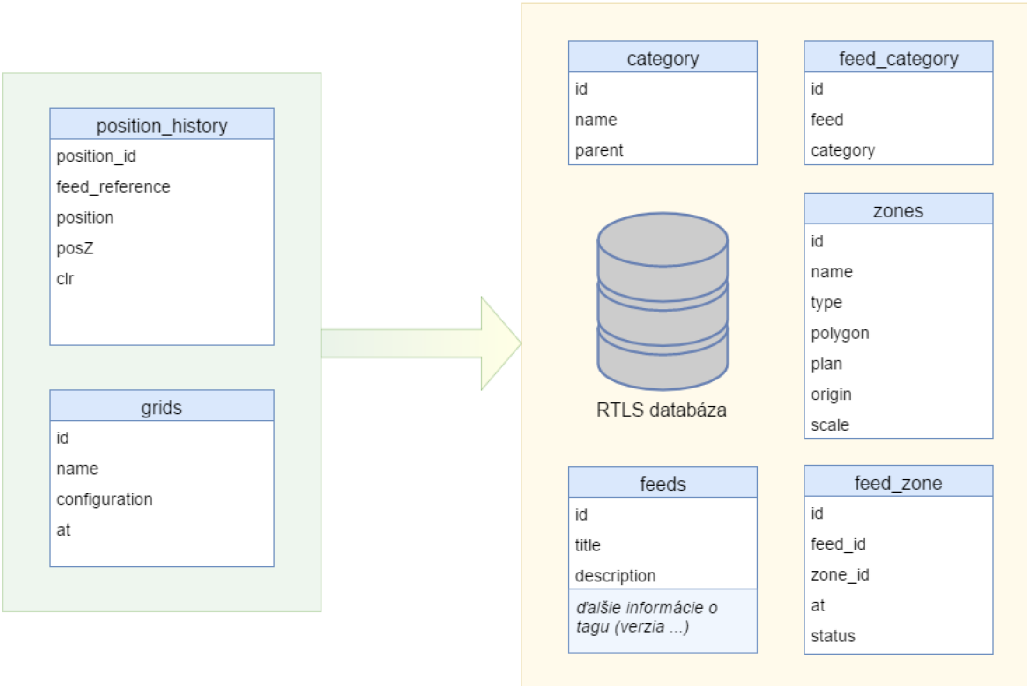
4. feed_category - tabuľka spája tabuľku feed a category pre dosiahnutie M:N vzťahu.

5. feed_zone - tabuľka opäť spája tabuľku feed, tentokrát s tabuľkou zone pre dosiahnutie M:N vzťahu. Do tohto vzťahu ale vkladá ďalšie informácie ako napríklad čas, kedy bolo toto spojenie vytvorené.

Na ľavej strane, v zelenej oblasti sa nachádzajú nove tabuľky potrebne pre ukladanie historických dát a uložených konfigurácií. Týmito tabuľkami sú:

1. position_history - reprezentuje jednotlivé získané pozície. Každá pozícia sa viaže na tabuľku feed, ktorá identifikuje tag, ku ktorému táto pozícia patrí. Pozícia je vyjadrená špeciálnym MySQL dátovým typom pre vyjadrenie geometrie. Pozícia obsahuje hodnoty na osách X a Y. Pozícia na osi Z je ukladaná osobitne a v súčasnej dobe nie je využívaná.

2. grids - táto tabuľka obsahuje informácie o uložených konfiguráciách. Každá konfigurácia má svoje unikátne meno a čas, v ktorom bola naposledy upravená. Samotná konfigurácia je uložená v JSON reprezentácií.



Obr. 5.1: Prehľad tabuliek, ktoré sú využívané ovládacím panelom.

5.2 Uložené dáta a požiadavky na ich vyhľadávanie

Dôležitým kritériom pre možnosť ukladania historických polohových dát, je mať k dispozícií dostatočný ukladací priestor. Objem výsledných polohových dát závisí na obnovovacej frekvencií jednotlivých tagov. Ako príklad je možné uviesť objem nazbieraných hodnôt, získaných pri monitorovaní hry hráča basketbalu. Obnovovacia frekvencia bola nastavená na hodnotu 100ms a čas hry predstavoval 20 minút. Teoreticky by sa preto v databázy malo nachádzať 12000 hodnôt. Tagy ale obsahujú akcelerometer a v prípade že sa hráč nehýbe, nevysielajú informáciu o pozícií. V databázy sa teda nachádza 9700 hodnôt čo predstavuje približne 0.9MiB dát v databázy. Za jediný deň by v takýchto podmienkach tento tag vyprodukoval približne 65MiB dát. To je preto potrebné brať na vedomie a prispôsobiť tomu zdroje serveru, na ktorom bude systém pracovať.

Nielen objem ale aj počet záznamov v tabuľke pozícií môže byť problém. Tieto pozície sa najčastejšie filtrujú na základe identifikácie tagu a času, v ktorom bola pozícia získaná. Preto bolo nutné vytvoriť indexy na jednotlivé stĺpce tabuľky. S použitím indexov sa objem dát v tejto tabuľke taktiež zväčšuje.

Po vyriešení vyššie spomínaných problémov môže aj napriek tomu nastať situácia, kedy používateľ vyberie taký interval, v ktorom sa nachádza až príliš veľké množstvo pozícií. Preto je vhodné používateľovi v čo najkratšom čase oznámiť, že vybraný interval nie je možné vizualizovať. Preto sa jednotlivým prvkom stanovili podmienky vizualizácie a pri vyžiadaní konkrétneho prvku, sa špeciálnou požiadavkou na databázu overi, či výsledný počet pozícií nepresahuje hodnotu, ktorú by ovládací panel nedokázal vizualizovať. Experimentálne bolo overené, že pri počte pozícií nad 100 000, dochádza pri mapových vizualizáciach v oneskoreniu reakcií. Toto oneskorenie nastáva hlavne pri manipulácií s mapou v reálnom čase. Informácie, ktoré má takýto prvok reprezentovať je ale stále dobre čitateľná.

Hlavný limit pre všetky ostatné dátové vizualizácie bol stanovený na jeden milión polôh. Do tohto počtu pozícií je doba vytvorenia vizualizácie na strane serveru prijateľná. Jedná sa o dobu v rozmedzí niekoľkých sekúnd, podľa druhu reprezentácie. Pre zistenie počtu pozícií sa používa kód podobný tomu, ktorý je v ukážke 5.1. Prezentovaná požiadavka na databázu vráti jednu pozíciu v prípade, že je vyhovujúcich pozícií aspoň jeden milión. V takomto prípade vieme, že by server musel spracovávať milión, prípadne niekoľko miliónov pozícií a v tomto momente môžme dať správu používateľovi, aby zmenil vybraný časoví interval, pripadne obmedzil množinu vybraných tagov.

```
1 SELECT 1 FROM positions P WHERE P.tagId IN ('$vybrane_tagy') AND
2                                P.at >= '$od' AND
3                                P.at <= '$do'
4 LIMIT 1000000, 1;
```

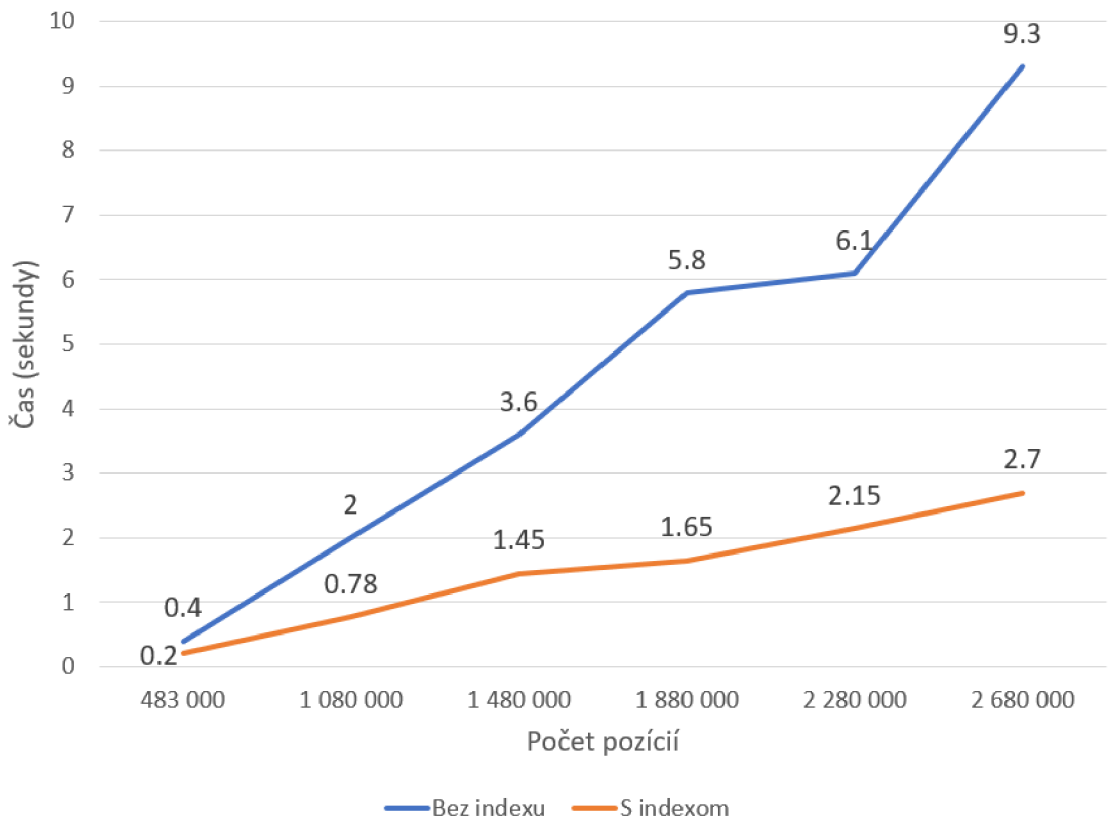
Ukážka 5.1: SQL kód pre zistenie, či počet pozícií, s ktorými bude server pracovať nepresahuje stanovený limit.

Ako už bolo spomenuté, polohy v databáze rýchlo naberajú na svojom objeme. Aktuálne sú všetky polohy uložené v jednej tabuľke. To je problém hlavne v prípade, že požadované polohy majú byť vybrané na základe časového obmedzenia. Nasledujúci zoznam popisuje vlastnosti pozícií, podľa ktorých sa výsledné pozície filtrujú:

1. Tag - množina tagov, pre ktoré sú pozície vyžadované.
2. Čas - časové ohraničenie.
3. CLR (z angl. Confidence Location Radius) - špeciálna hodnota, ktorá dáva pozícií určitú váhu. Popisuje ako si je RTLS systém s touto hodnotou istý. V našom prípade berieme do úvahy len tie najdôveryhodnejšie polohy.To znamená, že aj túto hodnotu treba pri tvorbe SQL požiadavky rozlišovať.

Hlavným nástrojom pre optimalizáciu filtrovania pozícií na základe uvedených kritérií, sú databázové indexy. Graf na obrázku 5.2 popisuje výsledky použitých indexov. Graf znázorňuje čas, ktorý je potrebný pre vyžiadanie špecifických pozícií a celkový počet pozícií v databáze. Index pozostáva z identifikácie tagu, hodnoty CLR a časovej stopy, ktorá je uložená pri každej pozícií. Časové hodnoty, ktoré sa v grafe objavujú vyjadrujú priemerný čas, ktorý bol vypočítaný z troch pokusov o tieto dáta. Z grafu je zrejme, že pri počte pozícií do 500 000 je čas, ktorý je potrebný pre vyžiadanie špecifických pozícií v oboch prípadoch takmer zanedbateľný. Zo zvyšovaním celkového počtu pozícií v databáze sa tento rozdiel zväčšuje. Z toho vyplýva nutnosť použitia týchto indexov.

Cena za použitie indexov je nárast objemu dát, ktoré tabuľka obsahuje práve o tieto indexy. Pri maximálnom počte pozícií, ktoré boli pri testovaní použité, teda 2 680 000 pozícií bolo potrebné 185.7MiB priestoru. Použité indexy zaberali v tomto prípade 40.6MiB priestoru, čo predstavuje 21,5% z celkového využitého priestoru.



Obr. 5.2: Výsledky použitia indexov na vyhľadávanie špecifických pozícií.

5.3 Optimalizačný algoritmus pre mapové vizualizácie

Mapové vizualizácie, akými sú teplotná mapa 4.4.1 alebo mapa ciest 4.4.2, sú veľmi závislé na počte pozícií, ktoré musia vykresliť. Pri manipulácií s týmito mapami totiž dochádza ku kompletnému prekresleniu všetkých zobrazených pozícií. Preto bol navrhnutý algoritmus, ktorý optimalizuje počet pozícií s čo najmenším dopadom na výslednú vizualizáciu. Algoritmus využíva dva parametre, s ktorými ho je možné optimálne nastaviť.

Prvým parametrom je citlivosť na zmenu umiestnenia tagu v priestore. Táto citlivosť je popísaná číslom, ktoré predstavuje hraničnú vzdialenosť medzi aktuálnym bodom a bodom nasledujúcim. V prípade, že táto vzdialenosť nie je prekročená, je nasledujúci bod vynechaný z výsledku. Tým sa odfiltrujú pozície, ktoré sú príliš blízko pri sebe a nebudú mať pri vizualizácií výpovednú hodnotu. Súradnice takto vynechaných bodov sa priemerujú. Tento priemer je neskôr použitý pre zohľadnenie menších posunov v priestore.

Druhým parametrom je interval v sekundách. Tento interval zaisťuje prítomnosť polohových dát vo výsledku v prípade, že tag nemení svoju pozíciu. V prípade, že tag nemení svoju pozíciu, sa po každom uplynutí tohto intervalu zapíše vyššie spomenutá priemerná pozícia do výsledku. Taktiež by bolo možné do tohto algoritmu vložiť tretí parameter, ktorý sa snaží odfiltrovať pozície, ktoré sú neprirodzené. Neprirodzenou pozíciou môže byť náhly odskok spôsobený chybou pri zbere polohových dát. Kód v ukážke 5.2 reprezentuje popisovaný algoritmus.

```
$param1 = 0.4 //meter
$param2 = 0.5 //sekunda
$actualPosition = $positions[0];

foreach( $positions + 1 as $nextPosition ){
        if( distanceBetween($actualPosition, $nextPosition) > param1 ){
                $actualPosition = $nextPosition;
                $averagePosition = $actualPosition;
                $resultPositions.push($actualPosition);
                continue;
        }

        $averagePosition = ($averagePosition + $actualPosition) / 2;

        if( timeBetween($actualPosition, $nextPosition) > param2 ){
                actualPosition = $averagePosition;
                $resultPositions.push($actualPosition);
                $averagePosition = 0;
        }
}

return $resultPositions;
```

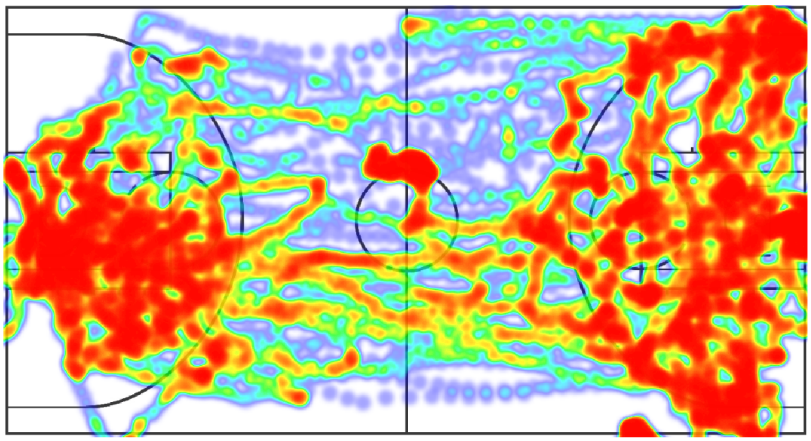
Ukážka 5.2: Pseudokód optimalizačného algoritmu.

Tabuľka 5.1 predstavuje výsledky redukcie dát pri basketbalovom zápase. Test bol vykonávaný na vzorke dát z 20 minútového zápasu. Vizualizácia zachytáva pohyb jedného hráča a ako metrika vizualizácie bola zvolená teplotná mapa. Prvý parameter algoritmu, určujúci citlivosť na zmenu polohy je nastavený na 0.4 metra. Druhý parameter určujúci interval, ktorý zaisťuje dáta v prípade, že tag nemení svoju pozíciu, je nastavený na 0.5 sekundy.

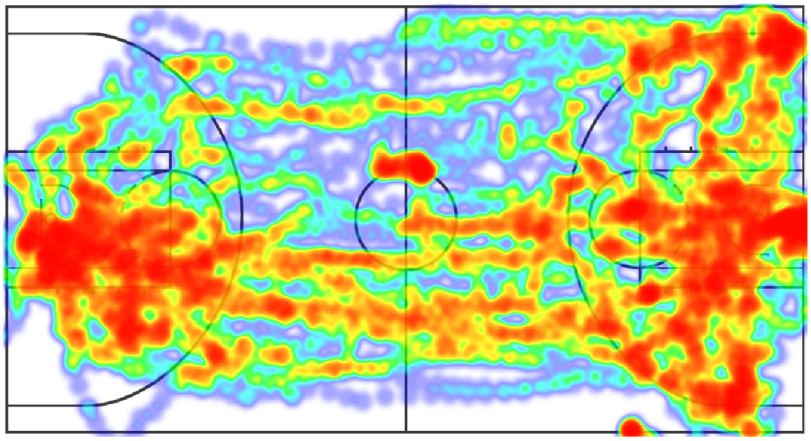
Na obrázku 5.3 je vidieť vizualizáciu zloženú z 9700 pozícií. Na obrázku 5.4 je mapa využívajúca len 42,13% z dostupných pozícií. Poskytuje ale veľmi podobné výsledky vizualizácie. Vďaka argumentom algoritmu sa úroveň filtrácie dá jednoducho meniť, prípadne vypnúť. Takáto filtrácia ale vo väčšine prípadov umožňuje zdvojnásobiť maximálny počet pozícií, ktoré dokáže vybraná vizualizácia zobraziť.

Tabuľka 5.1: Výsledky aplikovania algoritmu pre zníženie počtu pozícií pre mapové vizualizácie.

Bez použitia algoritmu	9700 pozícií
S použitím algoritmu	4087 pozícií



Obr. 5.3: Teplotná mapa zobrazujúca hráča pri basketbalovej hre. Pozície nie sú filtrované navrhnutým algoritmom.



Obr. 5.4: Teplotná mapa zobrazujúca hráča pri basketbalovej hre. Pozície sú tentokrát filtrované navrhnutým algoritmom.

Kapitola 6

Interakcia s ovládacím panelom

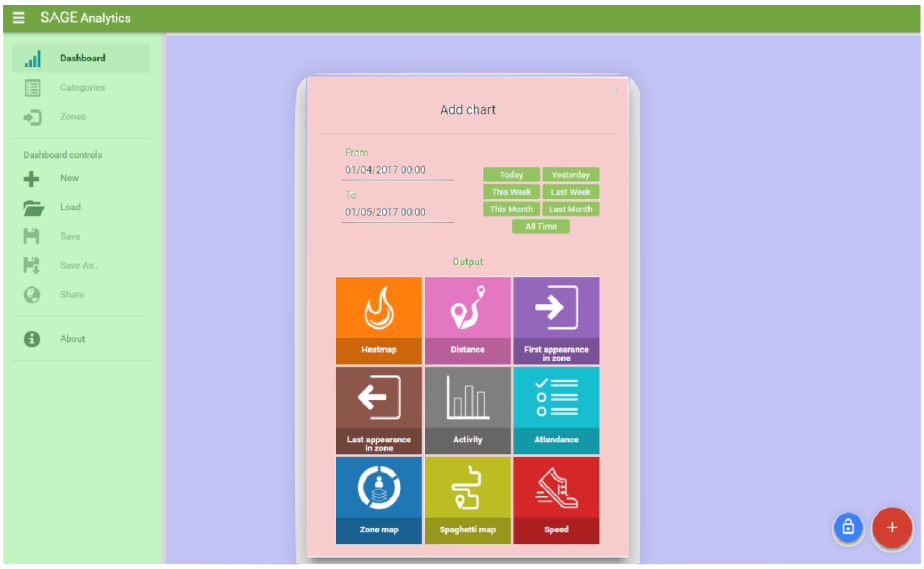
Táto kapitola popisuje možnosti definovania vizualizácie. Následné sú popísané všetky možnosti manipulácie s ovládacím panelom. To zahŕňa manipuláciu z rozložením prvkov vizualizácie, ale aj manipuláciu so samotnými grafmi a mapami.

Cely ovládací panel, znázornený na obrázku 6.1, sa skladá z troch hlavných častí:

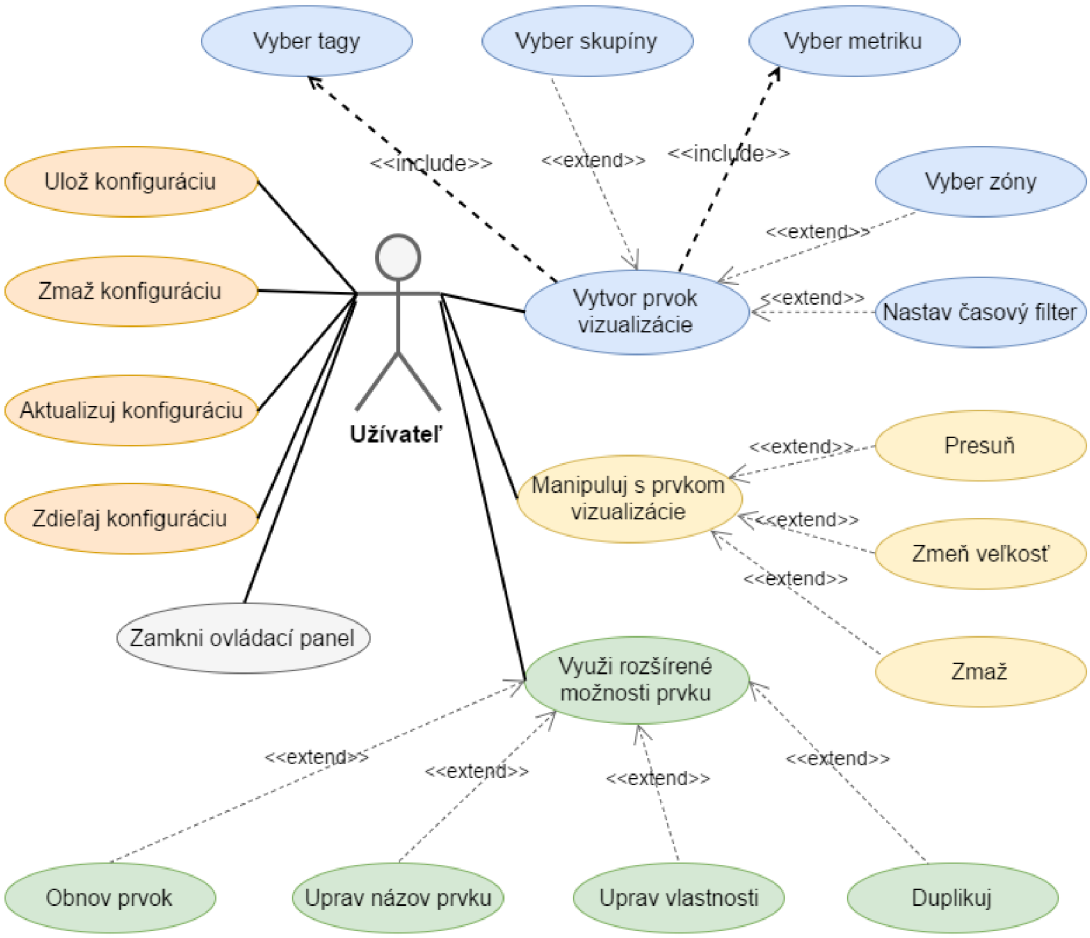
1. Hlavná plocha, označená modrou farbou - je to priestor, na ktorý sa umiestňujú prvky vizualizácie. Podrobnejší popis sa nachádza v kapitole 6.1.

2. Menu, označené zelenou farbou - poskytuje ďalšie rozšírenia ovládacieho panelu. Medzi tieto rozšírenia patrí možnosť ukladania konfigurácia, nástroj pre správu skupín a nástroj pre správu zón. Posledné dva menované nástroje prebiehajú vývojom a nie sú súčasťou tejto práce.

3. Panel pre požiadavky, označený červenou farbou - s jeho pomocou si používateľ definuje požadovanú vizualizáciu, tento panel je popísaný v kapitole 6.3.



Obr. 6.1: Rozdelenie ovládacieho panelu na tri časti reprezentované červenou, modrou a zelenou farbou.



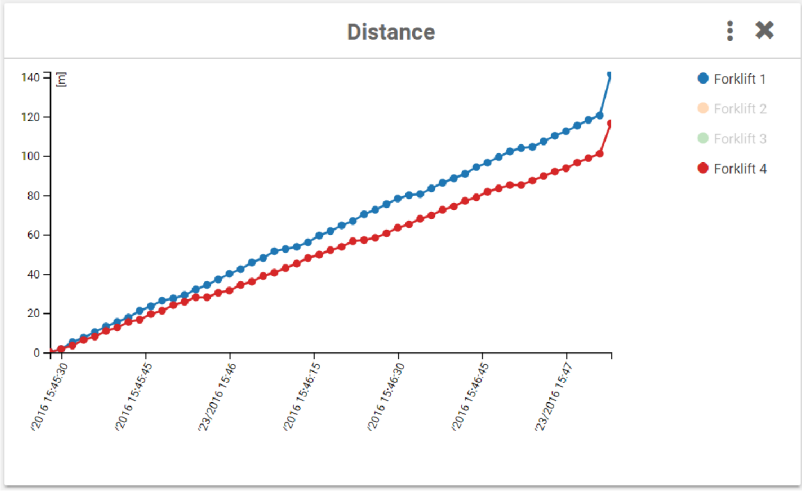
Obr. 6.2: Diagram prípadov použitia.

6.1 Možnosti manipulácie s ovládacím panelom

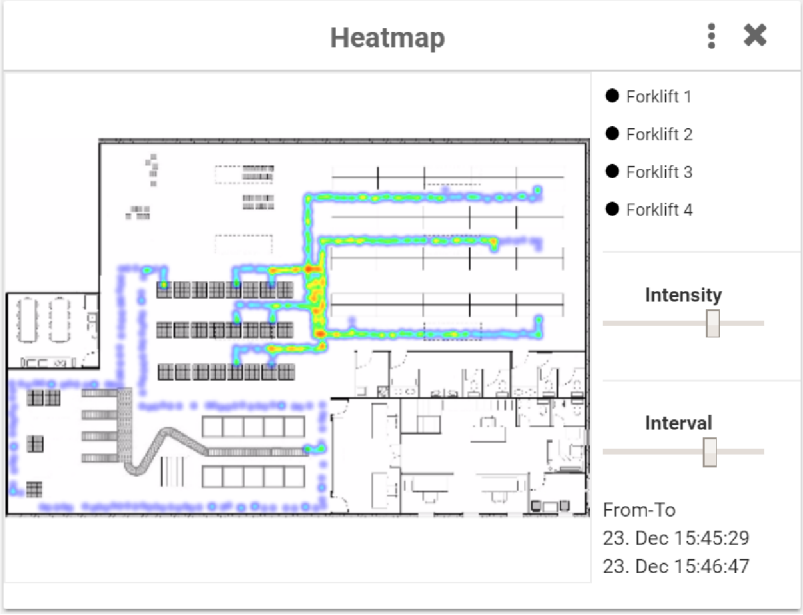
Hlavnú plochu ovládacieho panelu tvorí neviditeľná mriežka. Táto mriežka je použitá pre logické členenie priestoru, v ktorom sa nachádzajú jednotlivé prvky. To umožňuje používateľovi jednoducho manipulovať s jednotlivými prvkami panelu. Používateľ je schopný meniť ich pozíciu a veľkosť. Jednotlivé grafy a mapy, sú schopné reagovať na udalosti, ktoré sú spúšťané pri zmene veľkosti prvku. Ovládací panel tak získal používateľsky priateľské ovládanie.

Počet stĺpcov sa mení v závislosti na šírke prehliadača. Naopak počet riadkov sa dynamicky zväčšuje. Aplikácia by mala byť použiteľná ako na desktope, tak aj na tablete. To znamená že musí vedieť správne reagovať na zmeny veľkosti obrazovky. Pri zmene veľkosti prehliadača sa jednotlivým prvkom pošle signál na prepočítanie svojej veľkosti. Po potrebnom zmenšení sa prvky preusporiadajú.

Do oblasti manipulácie s panelom spadá aj ukladanie si aktuálnej konfigurácie. Spôsobu akým je konfigurácia uložená v databázy sa venovala kapitola 5.1. Konfiguráciu je navyše možné zdieľať. Aplikácia poskytne používateľovi odkaz, v ktorom je špecifikované meno aktuálne uloženej konfigurácie. V momente kedy sa na túto URL adresu dostane používateľ, ktorému bol tento odkaz poskytnutý, sa zaháji vytváranie panelu z uloženej konfigurácie. Možnosti práce s ovládacím panelom popisuje diagram prípadov použitia na obrázku 6.2.



Obr. 6.3: Interaktívna legenda integrovaná do vizualizačných prvkov ovládacieho panelu.



Obr. 6.4: Rozšírené rozhranie ovládania mapových vizualizácií.

6.2 Možnosti manipulácie s prvkami vizualizácie

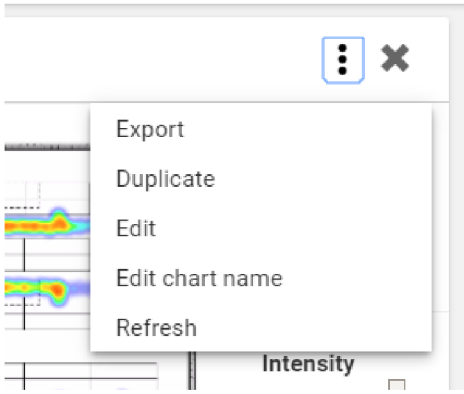
Každá metrika podporuje určitú manipuláciu so zobrazenou reprezentáciou dát. V prípade grafových a mapových reprezentácií ide najmä o podporu približovania, či už grafu alebo mapy. Takéhoto správania bolo možné dosiahnuť hlavne z dôvodu využívania knižnice D3.js, bližšie špecifikovanej v kapitole 3.2. Obrázok 6.3 zobrazuje prvok ovládacieho panelu spolu s interaktívnou legendou. Legenda slúži na filtrovanie vizualizovaných informácií v reálnom čase, podľa zvolenej množiny tagov. Legenda na obrázku je v stave, kedy sú tagy reprezentované oranžovou a zelenou farbou vynechá zo spracovania. Takéto chovanie podporuje väčšina vizualizácií.

Teplotná mapa a mapa ciest naviac pridávajú podporu pre filtrovanie pozícií v čase. Používateľ si je schopný posúvať čas, v ktorom sa konkrétne pozície získali a tým simuluje pohyb sledovaného tagu. Zároveň mu je poskytnutá informácia, v akom časovom intervale sa aktuálne nachádza. Teplotná mapa pridáva ešte rozhranie pre nastavenie intenzity, teda vzájomného ovplyvňovania sa jednotlivých zobrazených pozícií. Toto rozšírené rozhranie prvku prezentuje obrázok 6.4.

Pre lepšiu orientáciu a rýchlejšiu tvorbu prvkov ovládacieho panelu boli vytvorené rozšírené možnosti prvku ovládacieho panelu. Tieto možnosti je možné jednoducho rozširovať o možnosti, ktoré sú špecifické pre jednotlivé druhy vizualizácií. Medzi možnosti, ktoré podporujú všetky druhy vizualizačných metrík patrí:

1. Export - export dát do formátu XML.
2. Duplikácia - kópia duplikovaného prvku je vytvorená v spodnej časti panelu a obsahuje rovnaké dáta.
3. Úprava vybraných vlastností prvku - okno pomocou ktorého si používateľ volil prvok vizualizácie sa znovu sprístupní pre úpravu vybraného grafu. Tentokrát je už ale predvyplnené hodnotami upravovaného prvku, čo urýchľuje jeho zmenu.
4. Zmena mena prvku - používateľ si môže jednotlivé prvky pomenovať.
5. Obnovenie prvku - požiadavka, ktorá bola pri vytváraní prvku poslaná na server sa zopakuje. V prípade že vlastnosť časového omedzenia nebola nastavená, môže byť výsledná vizualizácia zostavená z aktuálnejších dát.

Kombinácia duplikovania a následnej úpravy prvku významne urýchľuje tvorbu obsahu ovládacieho panelu. Obrázok 6.5 vizualizuje formu rozšírených možností prvku.



Obr. 6.5: Rozšírené možnosti prvku.

Obr. 6.6: Obrázok predstavuje panel, do ktorého používateľ zadá požadované vlastnosti pre vizualizačný prvok.

6.3 Špecifikácia požiadavky pre vytvorenie prvku vizualizácie

Jednou z hlavných vlastností ovládacieho panelu je schopnosť správne reagovať na používateľove vstupy a byť flexibilný k jeho požiadavkám. V tomto prípade ide najmä o schopnosť navrhnúť vlastnú podobu ovládacieho panelu. Používateľ zadá druh zobrazenia dát a vyberie, ktoré lokalizačné zariadenia majú byť zobrazené. Okrem týchto zariadení je možné vyberať aj zóny alebo skupiny. Aplikácia pracuje s historickými dátami, a preto je nevyhnutné, aby bola schopná tieto dáta filtrovať na základe času. O to sa stará filter, do ktorého používateľ zadá počiatočný a koncový čas. K dispozícií ma na výber niekoľko predvolených časových rozmedzí. Následne sú všetky dáta, ktoré sa zobrazia do grafu, filtrované na základe tohto obmedzenia. Obrázok 6.6 predstavuje rozhranie, s pomocou ktorého používateľ zadáva svoje požiadavky aplikácií. Hlavná prednosť tohto okna je, že po výbere metriky sa používateľovi dynamicky generujú ďalšie možnosti. V prípade že si zvolí určitý tag, server sám rozpozná ku akému plánu je tento tag priradený. Následný zoznam zón z ktorých používateľ vyberá je prispôsobený tomuto plánu a tým odpadá nárok na to, aby používateľ vedel, ktorá zóna patrí na aký plán.

Kapitola 7

Testovanie

Testovanie výslednej aplikácia sa celkovo delí na dve hlavné časti. Prvou časťou je testovanie aplikácie vzhľadom na používané zariadenie a webový prehliadač. Používanie aplikácie na rozdielnych zariadeniach a prehliadačoch je mimoriadne náchylné na chybové, prípadne nežiadúce rozdielne správanie. Toto testovanie teda cieli hlavne na klientskú časť aplikácie.

Druhou časťou je testovanie výsledných vizualizácií vzhľadom na reálny pohyb po miestnosti. V tomto prípade poznáme presné vzdialenosti, po ktorých sa monitorovaný objekt pohyboval a porovnávame ich s výsledkom vizualizácie. Keďže presnosť výsledných hodnôt silne závisí na presnosti samotného lokalizačného systému, v tomto testovaní ide hlavne o schopnosť reprezentáciu polôh správne identifikovať, a získať z nej potrebné hodnoty.

7.1 Podporované zariadenia a webové prehliadače

Medzi zariadenia ktorým je aplikácia primárne určená patri stolný počítač a tablet. Rozlíšenia obrazoviek nehrajú dôležitú rolu a aplikácia je stále dobre čitateľná. V prípade priblíženia obrazovky na stolnom počítači, aplikácia správne reaguje a dokáže prispôsobiť svoj obsah pre aktuálnu veľkosť plochy prehliadača. Ovládanie pomocou gest na dotykovej obrazovke funguje spoľahlivo, no je tu stále priestor pre zlepšenie a to najmä v prípade zväčšovania veľkosti vizualizačného prvku. Klasické mobilné zariadenia majú príliš malý display pre praktické využitie, ale aplikácia je použiteľná aj na týchto zariadeniach.

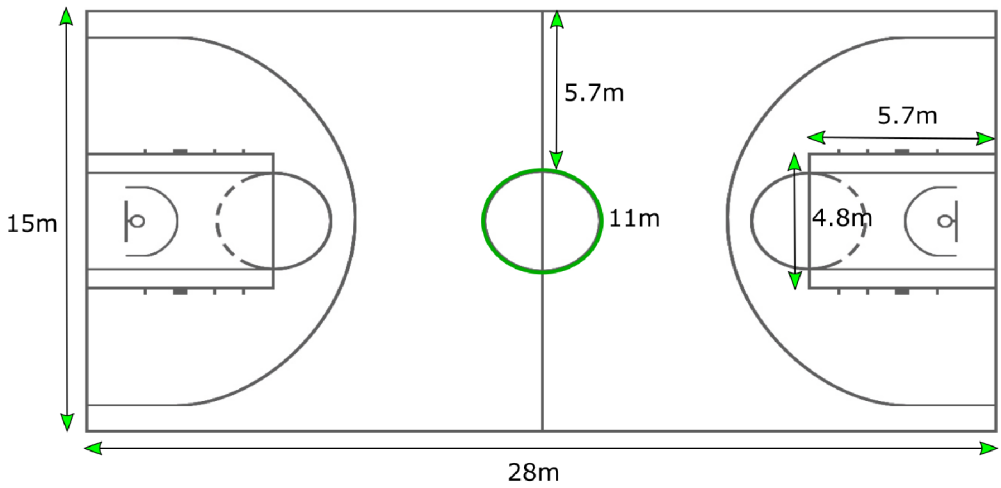
Lokalizačné dáta sú chránenou informáciou a nemali byť dostupné z prostredia internetu. Aplikácia bude teda vo väčšine prípadov nasadená do prostredia intranetu. Z toho vyplýva, že je možné si dovoliť zamerať vývoj na užší výber webových prehliadačov, pre ktoré je aplikácia primárne ladená. Počas implementácie to boli primárne prehliadač Chrome a prehliadač Firefox. Všetky prvky aplikácie boli pre tieto prehliadače testované a ich správanie sa nijak výrazne neodlišovalo. Pri použití prehliadača Safari nastali komplikácie pri používaní mapových reprezentácií.

7.2 Porovnanie vizualizácie s reálnym pohybom monitorového objektu

7.2.1 Množina testovacích dát

Pre testovanie boli použite umelo vytvorené testovacie dáta, ale aj reálne dáta. Výhoda umelo vytvorených pozícií je tá, že vieme simulovať také situácie, ktoré by sme reálne nevytvorili. Medzi takéto situácie môže patriť veľmi vysoká rýchlosť pohybu alebo simulácia chybne získaných dát. Testovanie pomocou takýchto dát prebiehalo hlavne pri vývoji ovládacieho panelu.

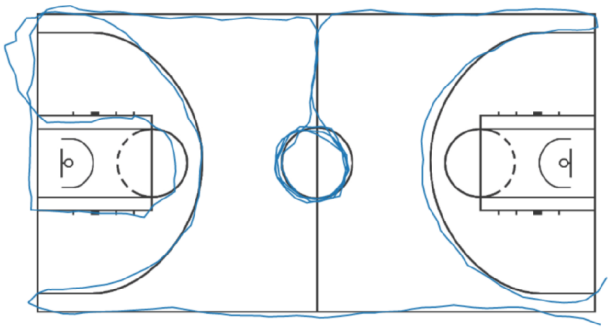
Pre finálne testovanie boli použité prevažne dáta získané v prostredí basketbalového ihriska. Obrázok 7.1 približuje rozmery ihriska, ktoré bolo použité pri testovaní prvkov vizualizácie. Pri testovaní v tomto prostredí je monitorovaný pohyb jedného tagu.



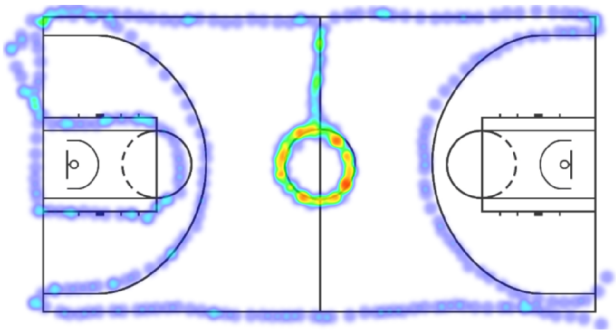
Obr. 7.1: Miery ihriska

7.2.2 Výsledky testovania vizualizácií

Hlavnou úlohou testovania je odhaliť, či sú zozbierané pozície správne reprezentované do zvolenej formy zobrazenia. Preto bude ku každému výsledku testu priložený obrázok s vizualizáciou. Prvkom je možné nastavovať časový filter najmenej v rozmedzí sekúnd. V niektorých výsledkoch testovania budú zvýraznené časti, na ktoré je špecifický test zameraný. Keďže presnosť výsledných hodnôt silne závisí na presnosti samotného lokalizačného systému, v tomto testovaní ide hlavne o schopnosť správne vypočítať informácie získané z týchto dát, prípadne správne zobraziť tieto pozície za rozličných okolností. Týmito okolnosťami sa myslí manipulácia s prvkom, prípadne s vykresleným grafom alebo mapou. ďalej sa v týchto testoch odhalí čí jednotlivé prvky správne vyhodnocujú situáciu, kedy tag prechádza, prípadne sa zdržuje v konkrétnej zóne.



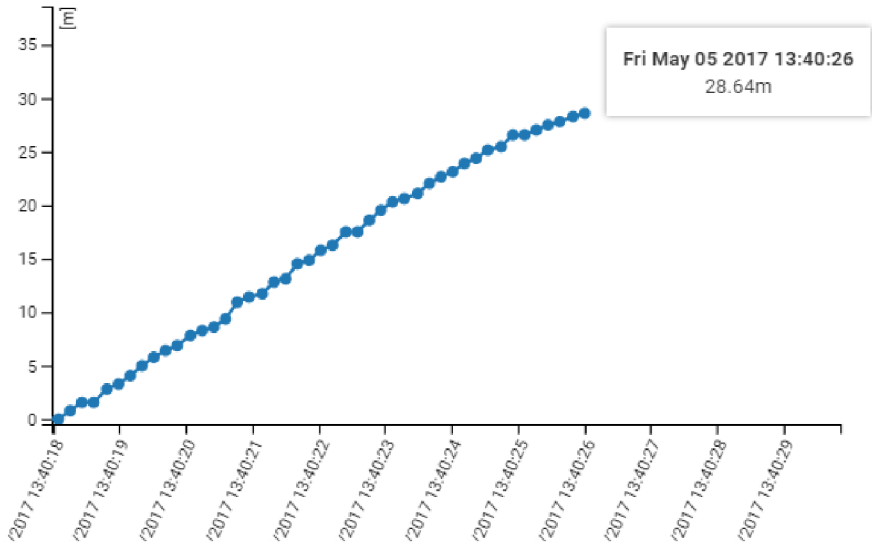
Obr. 7.2: Obrázok reprezentuje pohyb po basketbalovom ihrisku. Cieľ testu je dokázať, že poskytnuté a následne optimalizované pozície vytvárajú správne vykreslené cesty, na príslušnom podklade.



Obr. 7.3: Obrázok reprezentuje pohyb po basketbalovom ihrisku. Cieľ testu je dokázať, že poskytnuté a následne optimalizované pozície sú správne reprezentované pomocou teplotnej mapy, na príslušnom podklade.

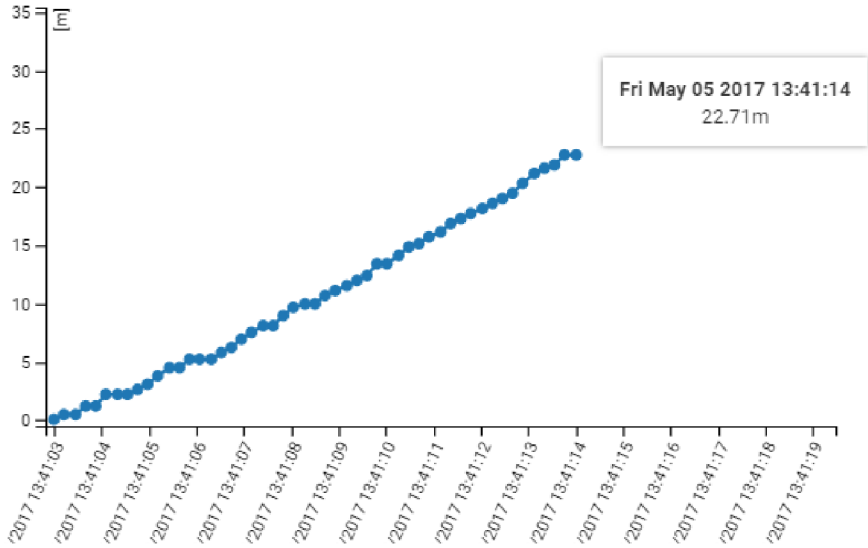
Dvojica obrázkov 7.2 a 7.3 predstavujú výsledok testovania správnej vizualizácie pohybu po ihrisku. Pohyb kopíruje čiary ihriska a je viditeľné, že vizualizované polohy správne priliehajú na tieto čiary. Výsledkom testu je teda fakt, že aplikácia dokáže poskytnuté polohy správne vykresliť na plán. Toto vykreslenie správne funguje aj pri optimalizácií pozícií na strane serveru. Ďalej správne funguje aj pri manipulácií s prvkom a samotnou vizualizáciou. Taktiež je možné vidieť, že teplotná mapa správne vizualizuje viacnásobný prechod kruhom, ktorý s nachádza v strede ihriska.

Obrázky 7.4 a 7.5 predstavujú otestovanie správnej vizualizácie prejdenej vzdialenosti. Táto vzdialenosť je vypočítavaná na strane serveru. Prvým vstupom testu je prejdenie jednej dĺžky ihriska. Druhý vstup predstavuje pohyb po kružnici nachádzajúcej sa v strede ihriska. Tento pohyb po kružnici je vykonaný dva krát pre utlmenie náväznosti kruhového pohybu. Výsledkom výpočtu prejdenej vzdialenosti pri priamom pohybe je hodnota 28.64 metrov. Táto nepresnosť o 0.64 metra môže byť spôsobená chybou merania alebo skutočnosťou, že vzdialenosť je vypočítavaná v rozmedzí sekúnd a tak je celková prejdená vzdialenosť zaokrúhlená na najbližšiu sekundu. To znamená že bola braná do úvahy aj vzdialenosť tesne po prejdení celkovej dĺžky ihriska.



Obr. 7.4: Obrázok reprezentuje výpočet a vizualizáciu prejdenej vzdialenosti v teste, v ktorom bola prejdená jedna dĺžka ihriska. Reálna dĺžka tohto ihriska je 28 metrov.

Druhý test celkovej prejdenej vzdialenosti testuje jej výpočet pri pohybe po kružnici. Hodnota 22.71 metrov sa líši od reálnej o 0.71 metrov. Výsledný graf bol zoskupený pre potreby dokumentácie. V reálnom použití zaberá celu šírku horizontálnej osi.



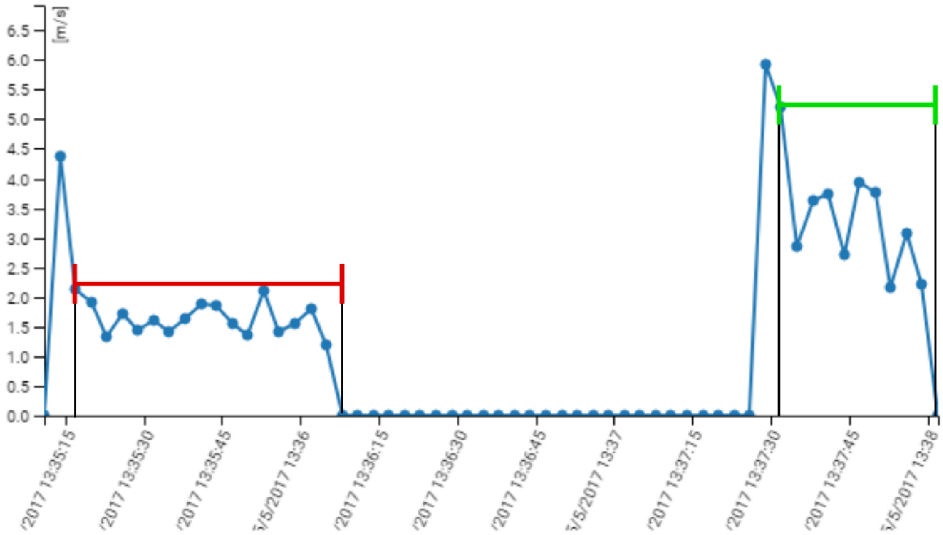
Obr. 7.5: Obrázok reprezentuje výpočet a vizualizáciu prejdenej vzdialenosti v teste, v ktorom bolo dva krát prejdené po obvode kruhu v strede ihriska. Tento obvod meria 11 metrov.

Nasleduje test, ktorý má odhaliť správnu vizualizáciu priemernej rýchlosti. V teste sú vykonané dva pohyby a to pohyb krokom a pohyb behom. Tento test spolieha na správny výpočet priemernej rýchlosti tagu a je vizualizovaný obrázkom 7.6. Ná obrázku sú označené dva úseky a to:

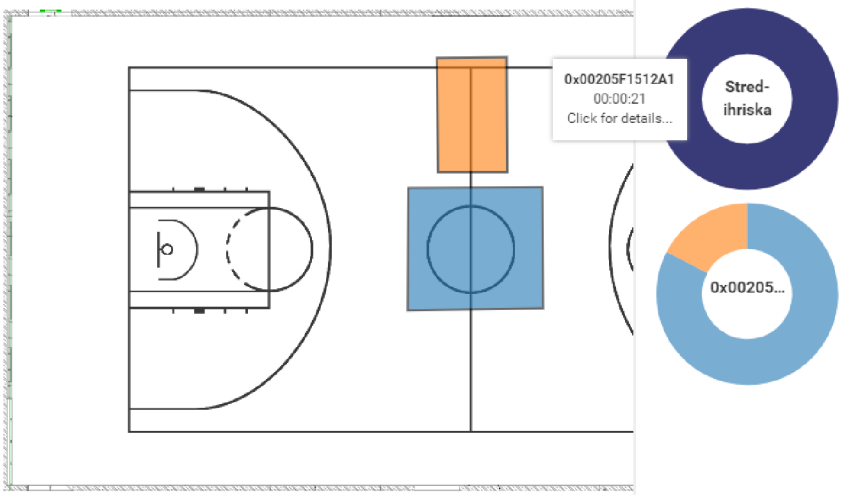
1. Červený úsek - popisuje pohyb krokom okolo ihriska. Pohyb predstavoval prejdenie 97 metrov za 51 sekúnd, čo znamená priemernú rýchlosť približne 1.9 metrov za sekundu.

2. Zelený úsek - popisuje podobný pohyb, tentokrát behom okolo ihriska. Pohyb predstavoval prejdenie 100 metrov za 29 sekúnd, čo znamená priemernú rýchlosť približne 3.45 metrov za sekundu.

Graf poskytuje čiastočné priemerné rýchlosti v jednotlivých úsekoch. Rýchlosti v červenom úseku sa pohybujú okolo hodnoty 1.9 metrov za sekundu, čo je aj vypočítaná priemerná rýchlosť celkového pohybu krokom. Rýchlosti v zelenom úseku vykazujú väčšie rozdiely no taktiež sa pohybujú v správnej oblasti a to 3.5 metra za sekundu. Reprezentované skokové rozdiely mohli byť spôsobené širokým rozsahom v ktorom sa polohy prepočítavali. Je treba brať do úvahy aj fakt, že pri behu do štvorca sa rýchlosť pri pravouhlej zmene smeru zmení.



Obr. 7.6: Vizualizácia pohybu krokom (červený interval) a pohybu v behu (zelený interval).



Obr. 7.7: Reprezentácia vytvorených zón pomocou zónovej mapy.

Nasledujúca skupina testov bude overovať funkcionalitu s použitím zón. Medzi metriky, ktoré využívajú zóny patrí metrika prvého a posledného výskytu v zóne, sledovanie aktivity tagu, sledovanie obsadenia zóny a mapa zón. V sade testov budú vytvorené dve zóny. Prvá bude zaznamenávať pohyb medzi krajom ihriska a stredovým kruhom. Druhá zóna ohraničuje kruh, ktorý je v strede ihriska. Testovaný pohyb je pohyb od kraja ihriska do stredového kruhu. V tomto kruhu je spravených niekoľko otočiek a nasleduje pohyb späť ku kraju ihriska. Tento pohyb je možné vidieť aj na obrázku 7.2.

Obrázok 7.7 predstavuje vytvorené zóny a zároveň zaznamenáva čas strávený v týchto zónach po prejdení popisovaného pohybu. Zaznamenaný čas, teda 21 sekúnd v stredovej zóne odpovedá reálne strávenému času v tejto zóne. Prípustná odchýlka bola stanovená na jednu sekundu. Vizualizácia taktiež vykresľuje vytvorené zóny na správnom mieste pri akejkoľvek manipulácií v prvkom. Druhý koláčový graf znázorňuje čas strávený v jednotlivých zónach.

Na obrázku 7.8 je výsledok výpočtu prvého a posledného výskytu v definovaných zónach. Údaje v tabuľkách sú presné a čitateľné.

Obrázok 7.9 predstavuje graf aktivity v jednotlivých zónach. Aktivita v stredovej zóne je správne vizualizovaná a predstavuje 21 sekúnd. Naopak vstupná zóna je vykreslená chybne. Táto chyba je zapríčinená tým, že rozdiel medzi jednotlivými aktivitami sa spája v prípade, že je medzi mini časový odstup pod jednu minútu. To umožňuje prehľadnejšiu vizualizáciu pri častom krátkom poklese aktivity.

Pri teste obsadenosti zóny je časový interval rozdelený na polovicu. Tým je docielené, že v prípade že druhý výskyt v oranžovej zóne ignorujeme, dostaneme tento krát správny výsledok. Výsledok tohto testu reprezentuje obrázok 7.10.

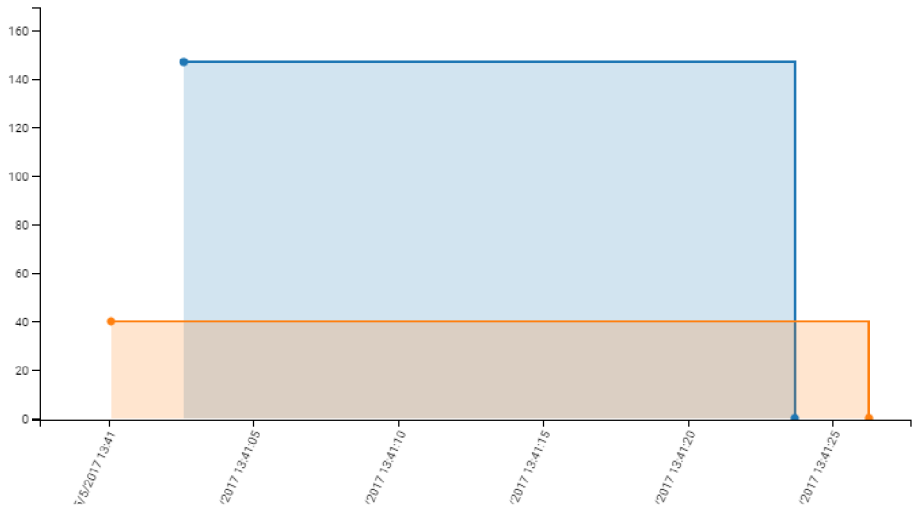
Last Appearance in Zone

Tag/Category	Stred-ihriska	Vstup
0x00205F1512A1	Fri May 05 2017 13:41:23	Fri May 05 2017 13:41:26

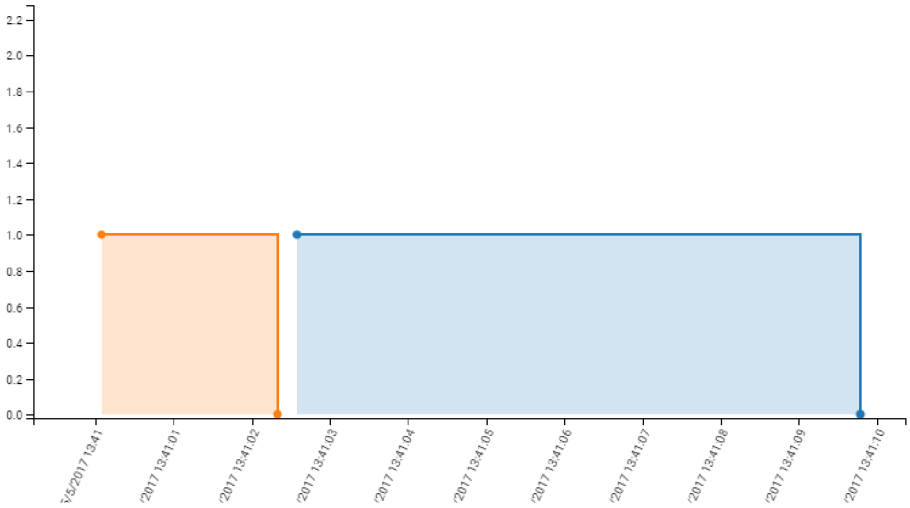
First Appearance in Zone

Tag/Category	Stred-ihriska	Vstup
0x00205F1512A1	Fri May 05 2017 13:41:02	Fri May 05 2017 13:41:00

Obr. 7.8: Reprezentácia výsledkov z tabuliek posledného a prvého výskytu v zónach.



Obr. 7.9: Reprezentácia aktivity v jednotlivých zónach.



Obr. 7.10: Reprezentácia obsadenia v jednotlivých zónach.

Kapitola 8

Záver

Lokalizácia v reálnom čase sa nezaobíde bez kvalitnej analýzy jej výsledkov. Výsledný ovládací panel má nasmerované byť univerzálnym nástrojom pre analýzu dát v rôznych prostrediach. Aplikácia úzko spolupracuje s presným RTLS spoločnosti Sewio. V prvotnej fáze projektu som sa s týmto systémom dobre oboznámil. Následne prebiehal výber vhodnej kombinácie knižníc, pre vytvorenie kvalitnej generickej webovej aplikácie. Implementácii predchádzalo podrobnejšie oboznámenie sa s vybranými knižnicami. Neskôr som ovládací panel v spolupráci s firmou Sewio začal rozvíjať až do popisovanej podoby.

Aplikácia je schopná bez problémov vizualizovať dáta z desiatok lokalizačných zariadení. Výkon aplikácie ale ovplyvňuje celkový počet polôh, s ktorými je potrebné v reálnom čase manipulovať. Prekresľovanie s určitým oneskorením nastáva u mapových vizualizácií. Toto oneskorenie nastáva pri práci s viac než 100 000 polohami. Výkonnostné obmedzenie nastáva aj v aplikácií na strane servera. Pri spracovaní rozsiahlych dát sa predlžuje čakacia doba na vyhotovenie žiadanej reprezentácie. Optimalizácia je teda nevyhnutná a do úvahy prichádza posielanie vypočítaných dát po častiach. Používateľovi sa tak bude výsledná vizualizácia postupne rozširovať.

Aplikácia sa v budúcnosti plánuje ďalej rozširovať. V pláne je pridávanie nových metrík, vylepšenie existujúcich či vytvorenie nástroja pre komplexnejšiu úpravu zón. Žiadanou funkcionalitou je aj možnosť automatického porovnávania informácií medzi jednotlivými prvkami vizualizácie.

Osobne som si skúsil reálne testovanie lokalizačného systému. Pohyboval som sa v miestnosti s lokalizačným senzorom a zaznamenával som si svoj pohyb vzhľadom ku miestnosti. Informácie o pohybe a prejdenej vzdialenosti som následne porovnával s vizualizáciou ovládacieho panelu. Ovládací panel spĺňal všetky požiadavky, ktoré boli naň kladené.

Literatúra

[1] Agafonkin, V.: *SimpleHeat.js* . 2017, [Online; navštívené 05.05.2017].
URL https://github.com/mourner/simpleheat

[2] Auth0: *Introduction to JSON Web Tokens*. 2017, [Online; navštívené 05.05.2017].
URL https://jwt.io/introduction/

[3] Bostock, M.: *D3.js*. library, Prosinec 1994, [Online; navštívené 05.05.2017].
URL https://d3js.org/

[4] Gechev, M.: *Building an Angular Application for Production*. blogpost, 2017, [Online; navštívené 05.05.2017].
URL http://blog.mgechev.com/2016/06/26/tree-shaking-angular2-production-build-rollup-javascript/

[5] Hazlewood, L.: *Top Six Reasons to Use API Keys* . blogpost, Prosinec 2016, [Online; navštívené 05.05.2017].
URL https://stormpath.com/blog/top-six-reasons-use-api-keys-and-how

[6] Rodriguez, A.: *RESTful Web services: The basics*. blogpost, Listopad 2008, [Online; navštívené 05.05.2017].
URL https://www.ibm.com/developerworks/webservices/library/ws-restful

[7] Wahlin, D.: *Getting Started with TypeScript – Classes, Types and Interfaces*. blogpost, Březen 2015, [Online; navštívené 05.05.2017].
URL https://weblogs.asp.net/dwahlin/getting-started-with-typescript-classes-static-types-and-interfaces

[8] Wasson, M.: *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET*. blogpost, Listopad 2013, [Online; navštívené 05.05.2017].
URL https://msdn.microsoft.com/en-us/magazine/dn463786.aspx