Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Měření kvality mobilních datových sítí - Vizualizace

Bc. Lukáš Sokol

Vedoucí práce: Ing. Jan Kubr

25. června 2015

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Janu Kubrovi za pomoc a cenné rady, které mi poskytl při zpracování mé diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 25. června 2015 .....

Odkaz na tuto práci

Sokol, Lukáš. *Měření kvality mobilních datových sítí - Vizualizace.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Obsahem této diplomové práce je návrh a implementace modulů webové aplikace určených pro prezentaci dat naměřených aplikací MBTest, pro administraci webové aplikace a pro export naměřených dat do různých formátů. Webová aplikace je naprogramována v jazyce PHP za použití frameworku Nette. Data jsou zobrazena pomocí tabulek a grafů, se kterými může uživatel interaktivně pracovat pomocí různých filtrovacích prvků.

Klíčová slova MBTest, PHP, Nette, MySQL, jQuery, grafy, tabulky, export dat, návrh, implementace

Abstract

The content of this thesis is the design and implementation of Web application modules for presentation of data measured by MBTest application, administration of Web application and export of measured data into various formats. The Web application is programmed in PHP using Nette framework. Data is displayed in tables and graphs, with which user can interact using different filter elements.

Keywords MBTest, PHP, Nette, MySQL, jQuery, graphs, tables, data export, design, implementation

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Projekt MBTest vznikl již v roce 2013, kdy na jeho jednotlivých částech pracovali kolegové jako na svých závěrečných bakalářských pracích. Hlavním cílem projektu bylo vytvořit systém, který získává od uživatelů informace o datovém připojení jejich mobilních zařízení. Takto získaná data systém ukládá a umožňuje zobrazení statistických údajů z naměřených hodnot. Vznikly tedy jednotlivé části tohoto systému. v prvé řadě to byly mobilní aplikace pro platformu android a iOs, které měří kvalitu datového připojení a výsledné údaje o jednotlivých měření zasílají serverové části systému. Server data ukládá do databáze a poté je zprostředkovává skrze své aplikační rozhraní. Poslední součástí systému je webová aplikace, která uživateli zobrazuje jeho naměřená data, ale i globální statistiky měření všech ostatních uživatelů.

Projekt se podařilo úspěšně dokončit, avšak poslední část systému (webová aplikace pro zobrazení dat) nesplňovala kladené požadavky, tak se vedoucí projektu shodli na její kompletní rekonstrukci. Protože se jedná o časově náročnou část, rozhodli se ji vedoucí práce rozdělit a vznikla tak dvě nová zadání.

Vzniklá témata nás s kolegou zaujala a po první konzultaci jsme si je rozdělili následovně. Mým úkolem je vytvořit administrativní část webové aplikace a dále pak modul, který zprostředkovává statistické údaje z naměřených dat a údaje poskytuje uživateli ve formě grafů. Kolegova část zahrnuje zobrazení takto získaných statistik do map a dále celkový frontendový design aplikace.

Kapitola 1

Popis problému, cíl práce

Primárním cílem projektu je zhotovit následující části webové aplikace. Modul, který bude zpracovávat statistiky naměřených dat ze serverové databáze pro různé uživatelské dotazy. Dále modul, který bude takto získaná data vhodným způsobem zobrazovat v tabulkách a grafech. A nakonec modul administrační části systému, ve které budou moci systémoví administrátoři měnit základní nastavení aplikace, nastavovat práva uživatelů a exportovat data ze serveru do různých formátů.

Ke globálním statistikám bude aplikace využívat naměřená data ze serverové databáze a také volně dostupná data z webového serveru netmetr.cz. O využití volně dostupných dat pojednává kapitola Analýza v sekci Požadavky na systém.

1.1 Specifikace cíle

Cílem diplomové práce je tedy provést analýzu projektu MBTest a na jejím základě navrhnout a implementovat jednotlivé moduly popsané výše a následně je otestovat.

1.1.1 Výstup práce

Výstupem budou následující části webové aplikace:

- modul sloužící jako databázové API
  - výběr dat podle uživatelského dotazu
  - zpracování statistických údajů
  - vrácení relevantních výsledků
- modul zobrazující vybraná data v tabulkách a grafech

  - zobrazení statistik uživatelova měření v tabulce umožňující přehledné řazení a vyhledávání dat
  - zobrazení statistik uživatelova měření v interaktivních grafech
  - zobrazení statistik globálních měření
  - zobrazení statistik měření na zadané trase
- administrační modul
  - možnost změny globálního nastavení aplikace
  - možnost změny uživatelských práv
  - export naměřených dat

Kapitola 2

Analýza

V kapitole věnované analýze jsou nejprve představeny jednotlivé části projektu MBTest. Kapitola pokračuje rešeršemi podobných aplikací a jejich vzájemným srovnáním. Poté jsou vymezeny funkční a nefunkční požadavky na systém. Následně kapitola rozebírá jednotlivé role a práva uživatelů. Na konec jsou představeny technologie, které by se daly pro implementaci systému použít. Všechny jsou zhodnoceny a je zde zdůvodněn výběr těch, které byly při vytváření aplikace použity.

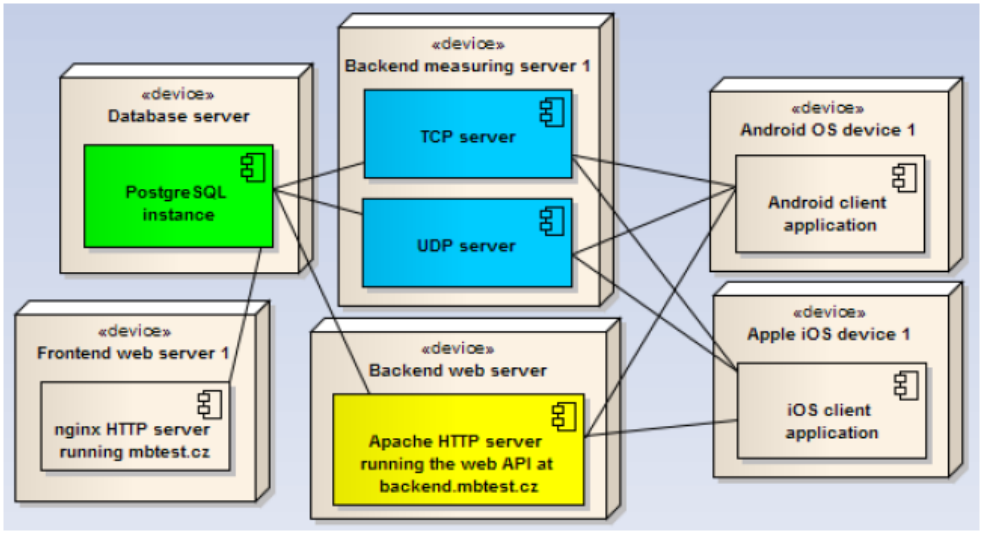
2.1 Projekt MBTest

V této sekci budou stručně popsány jednotlivé části projektu MBTest. Analýza určitých částí bude sloužit při návrhu webové aplikace. Jedná se o server, se kterým komunikují klienti (Android a iOS aplikace). Komunikaci jednotlivých komponent vystihuje obrázek 2.1. [1]

2.1.1 Server

Autor serverové části projektu navrhl a implementoval následující komponenty:

- měřící servery
  - TCP server – Protokol TCP je primárně zaměřený na spolehlivost spojení a má mnohem vyšší odezvy než protokol UDP (čas na potvrzení přijetí paketu). Data ovšem dorazí vždy v celku.
  - UDP server – Je výrazně rychlejší než TCP, ale data jsou rozdělena na pakety, ty jsou posílány bez potvrzení. Proto se používá spíše pro přenos objemnějších dat (streamování, online hry, atp.).
- databázový server (PostgreSQL)



Obrázek 2.1: Diagram nasazení

- backendový HTTP server

  Měřící servery komunikují s mobilními klienty a získávají od nich naměřená data, která následně ukládají do databáze. HTTP server s API pro získávání naměřených dat využívají mobilní klienti k zobrazení měření uživateli. K vývoji mé části aplikace bylo nezbytné analyzovat jaké parametry připojení jsou ukládány a jak vypadá schéma databáze.

2.1.1.1 Struktura měřených dat

Parametry mobilního připojení bychom mohli rozdělit do několika skupin.

- povinné údaje
  - typ připojení (GPRS, EDGE, HSPA+, atd.)
  - síla signálu
  - rychlost přenosu dat – TCP server
    - download – rychlost stahování dat
    - upload – rychlost nahrávání dat
  - odezva a ztrátovost dat – UDP server
  - model klientského zařízení
  -
- nepovinné údaje

- stav baterie
- poloha zařízení na začátku měření (GPS souřadnice)
- poloha zařízení na konci měření (GPS souřadnice)

2.1.1.2 Databáze

Autor projektu zvolil relační PostgreSQL databázi z důvodu lepšího API pro jazyk C++, který použil pro implementaci měřících serverů. Obrázek 2.2 zachycuje vytvořené databázové schéma[1]. Pro účely mé části webové aplikace je nejpodstatnější tabulka `measure` představující měření.

2.1.2 Klienti pro Android a iOS

Hlavním cílem mobilních klientů je naměřit parametry mobilního připojení (pomocí komunikace s měřícími servery) a zobrazit uživateli okamžitý výsledek měření. Dále umožňují zobrazení všech provedených měření uživatele. K implementaci mé části práce je přímo nevyužiji, pouze budu pracovat s již naměřenými daty uloženými serverové databázi, kam se postupně ukládají.

Důležitou otázkou je však, jakým způsobem se uživatelé do aplikací přihlašují. Autoři se nakonec shodli na použití standardního uživatelského jména a hesla. Email je pouze volitelným parametrem při vytváření účtu.

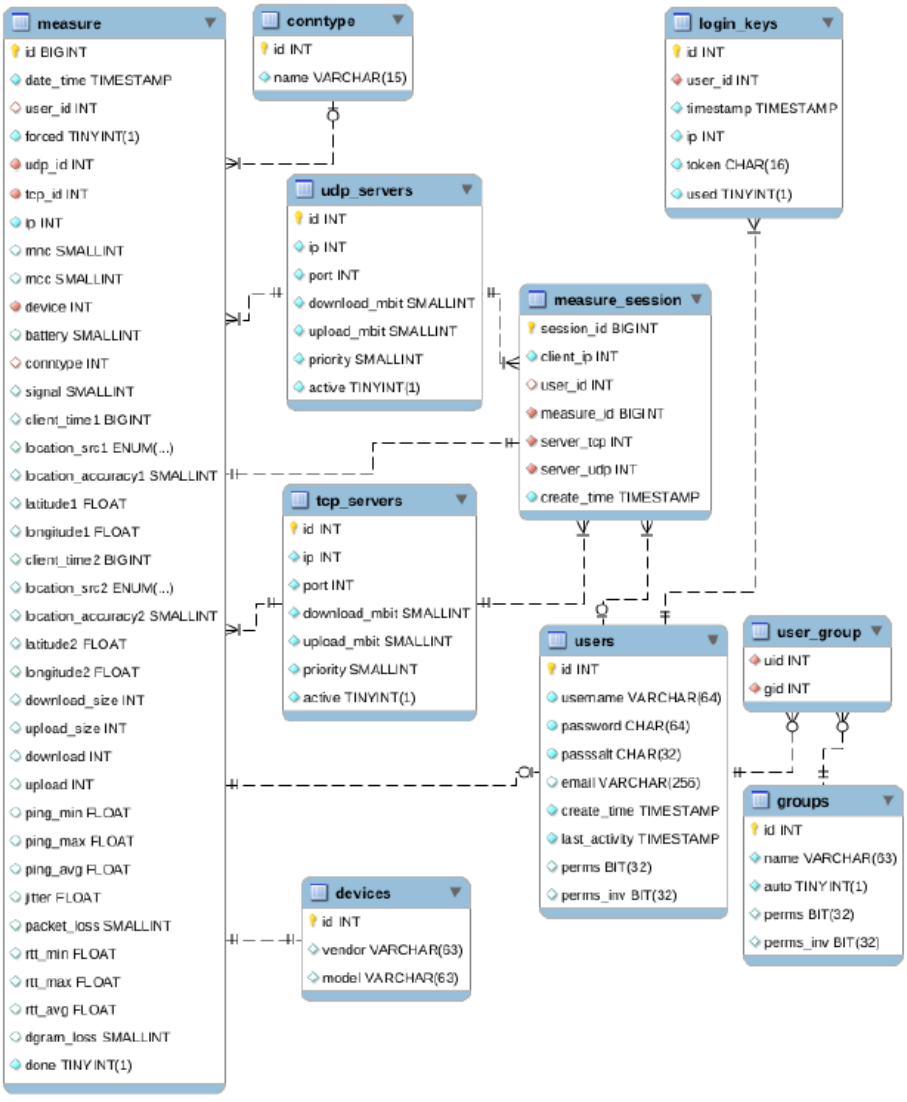
2.1.3 Webový server

Poslední částí je webový server. Za hlavní příčinu neúspěchu implementace tohoto projektu bychom mohli považovat autorovu snahu o sepsání vlastního PHP frameworku a nedostatek času pro lepší zobrazení uživatelských dat v přívětivé formě. I díky tomu se vedoucí projektu rozhodli tuto část rozdělit na dvě a poskytnout tak uživateli více možností při zobrazení svých měření a celkových statistik.

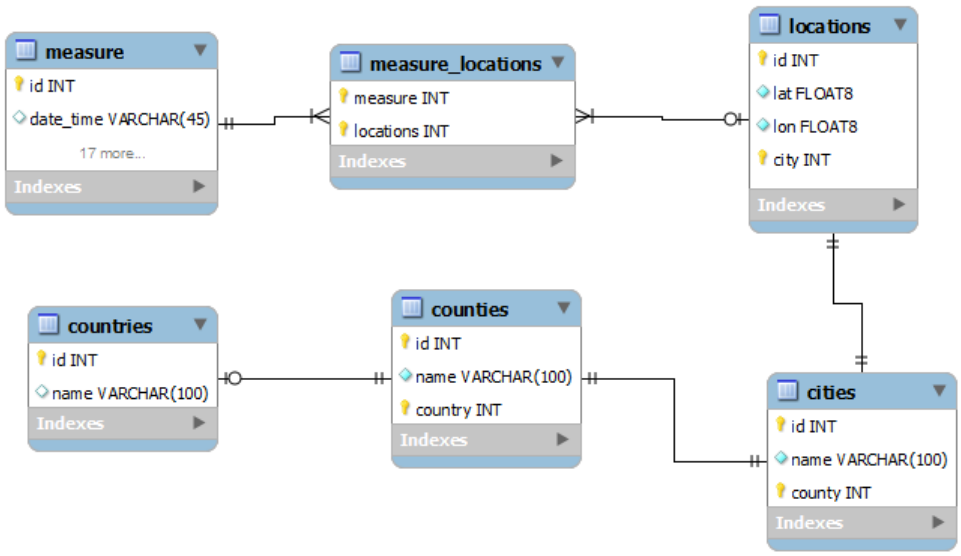
Tvůrce webové aplikace přidal do původní databáze další tabulky přesněji určující polohu místa, ze kterého bylo měření prováděno. Přiřazují tak polohu měření k městu, kraji a státu. Přidané tabulky jsou znázorněny na schématu 2.3. Jelikož již takto upravené databázové schéma bylo součástí testovacího provozu, rozhodl jsem se ho v této podobě ponechat. Způsob přiřazování GPS souřadnic k určitému celku (městu, kraji, státu) ovšem není zdokumentován. Bude tedy mým úkolem udržovat tyto tabulky aktuální.

2.2 Podobné existující aplikace

Při tvorbě aplikace je rozumné poohlédnout se po podobných konkurenčních produktech. Porovnat je mezi sebou a následně zhodnotit jejich funkčnost s očekávanými vlastnostmi mého projektu.



Obrázek 2.2: Databázové schéma systému MBTest



Obrázek 2.3: Databázové schéma dodatečné

Konkurenční projekty by se daly rozdělit do dvou skupin. Na skupinu projektů existujících před existencí systému MBTest a dále pak projekty, které vznikly po jeho dokončení.

2.2.1 Aplikace před existencí MBTestu

2.2.1.1 rychlost.cz [2]

Tato webová aplikace měří pouze rychlost stahování a nahrávání dat, webovou odezvu a stabilitu. Při měření používá protokol HTTP a měří pouze pomocí "webové odezvy"a neřeší tedy vůbec ztrátovost paketů (protokol UDP), viz sekce o serveru 2.1.1.

Jelikož se jedná pouze o webovou aplikace, tak není divu, že pro měření používá právě jen protokol HTTP. Měření pomocí jiných protokolů by vyžadovalo zapojení jiných technologií.

Výstupy portálu jsou zobrazeny v tabulce pro přihlášeného uživatele a nebo na základě IP adresy. Dále je k dispozici export měření do formátu CSV a histogram rychlostí.

2.2.1.2 dsl.cz [3]

Dsl.cz nabízí měření rychlosti a odezvy pomocí webové aplikace využívající Adobe Flash. Dále nabízejí i aplikaci pro iOS a Android, tudíž se velmi podobá

projektu MBTest. v době vzniku MBTestu byl dsl.cz nejpodobnější z konkurenčních aplikací.

Oproti MBTestu však zobrazuje uživateli pouze výsledek posledního měření a nebo jednoduchou mapu pokrytí, což není nic jiného než Google mapa se zanesenými body uživatelských měření s možností filtrování měření podle typu operátora a podle rychlostních kategorií. Další odlišností je neperiodičnost měření. Uživatel měření provádí samostatně, což má za následek poměrně zkreslené statistiky, protože uživatel měří většinou pouze v případech, kdy síť neběží podle jeho představ, a nebo v případě, kdy se chce pochlubit mimořádnou rychlostí.

Další vadou na kráse je podle uživatelů, kteří hodnotili Android aplikaci v Play storu, velký objem přenášených dat. Při častém používání pak hrozí vyčerpání kapacity mobilních dat. Tento problém řeší MBTest možným nastavením velikosti přenášených dat.

2.2.1.3 MobiPerf [4]

Za zmínku také stojí opensource mobilní aplikace MobiPerf pro operační systém Android. Jedná se pouze o klientskou aplikaci, která nemá žádné veřejně dostupné statistiky.

Otázkou tedy je, proč tvůrci MBTestu nepoužili již existujícího klienta pro platformu Android? Hlavním problémem bylo, že je tato aplikace svázaná s Google účtem a to se neshodovalo s návrhem přihlašování uživatelů, viz sekce 2.1.2. Dalším významným faktorem bylo, že k účelům projektu MBTest by se aplikace musela stejně hodně upravovat, aby splňovala všechny kladené nároky. A předělávat cizí kód zabere někdy více času a úsilí, než program vytvořit.

2.2.2 Aplikace po realizaci MBTestu

Jsou to již téměř dva roky od dokončení systému MBTest. Od té doby se objevila jedna výrazná konkurenční aplikace. Jedná se o projekt netmetr.cz.

2.2.2.1 netmetr.cz [5]

Tento projekt nabízí ke stažení aplikace pro iOS i Android, pomocí kterých uživatel provádí jednotlivá měření. Webová aplikace nabízí poměrně přehledné zobrazení globálních naměřených dat v tabulkách s možnostmi filtrování podle států. U každého státu pak srovnává všechny operátory podle rychlosti stahování, nahrávání, odezvy a síly signálu. Dále srovnává různé typy zařízení podle stejných kritérií. A nakonec je možné filtrovat výsledky z posledních 400 měření.

Uživatel si na webu může zobrazit své statistiky měření pomocí synchronizačního kódu, který mu vygeneruje jeho mobilní aplikace.

Poměrně dobře je zpracovaná mapa, která nabízí mnoho filtrů. Uživatel si může například vybrat mezi Google mapou (klasická, foto mapa, mapa terénu) nebo OSM (Open Street Map). Do mapy se zakreslují jednotlivá měření, která jsou zbarvená podle kvality testu měření (zelená, žlutá, červená). Filtry zobrazovaných měření jsou následující:

- výběr parametru – rychlost stahování, nahrávání nebo síla signálu
- výběr času – jak staré měření vracet (1 týden, 1 měsíc, atd.)
- výběr operátora
- výběr sítě

Významným faktem je, že aplikace nabízí veškerá měření jako Open Data. Proto můj kolega pověřen implementací pravidelného importu těchto dat do naší databáze. Strukturu těchto dat popisuje kapitola Návrh v sekci Databáze 3.1.

2.2.3 Shrnutí

Po srovnání konkurenčních projektů je potřeba určit, kdo bude moji aplikaci používat a co vše bude aplikace umožňovat.

2.3 Uživatelské role

v první řadě je potřeba rozlišit typy uživatelů, které bude aplikace obsluhovat a co vše jim umožní provádět.

2.3.1 Anonymní uživatel

Anonymní (nepřihlášený) uživatel si bude moci pouze prohlížet globální stránky aplikace (úvod, o nás, atd.) a bude se moci přihlásit. Není mu tedy ani povoleno prohlížení globálních statistik a map.

2.3.2 Přihlášený uživatel

Po zadání platných přihlašovacích údajů se z nepřihlášeného uživatele stane přihlášený. Pro tento typ uživatele bude ještě vymyšlen systém založený na hodnostech. Ten bude fungovat následovně: čím vyšší má uživatel hodnost, tím se mu zpřístupní další funkce aplikace. Přičemž zvyšování hodnosti bude záviset na počtu provedených měření uživatelem a jiných pravidlech (např. měření musí být vzdálena alespoň X metrů od sebe, aby se zamezilo spamování měření z jednoho jediného místa tak, aby uživatel nabyl vyšší hodnosti co nejrychleji).

2.3.3 Administrátor systému

Správce aplikace bude mít přístup do všech částí aplikace, včetně administrativní části systému, do které se běžný uživatel nedostane. Tento typ uživatele jsme zvolili primárně pro účely změny hodností jednotlivých uživatelů, pro určité VIP uživatele nebo například testovací uživatelské účty. Dále jsem povolil pouze této roli možnost exportu dat, viz kapitola Návrh, sekce export dat 4.2.1.

2.4 Požadavky na systém

2.4.1 Funkční požadavky

F1 **Identifikace uživatele**

- 1.1 Autentizace uživatele (ověření identity)
- 1.2 Autorizace uživatele (určení role)
- 1.3 Odhlášení uživatele

F2 **Zobrazení statistických dat**

- 2.1 Zobrazení uživatelských měření
- 2.2 Zobrazení globálních statistik
- 2.3 Zobrazení měření na trase

F3 **Filtrování statistických dat**

- 3.1 Filtrování uživatelských měření
- 3.2 Filtrování globálních statistik
- 3.3 Filtrování měření na trase

F4 **Administrační funkce**

- 4.1 Změna globálního nastavení aplikace
- 4.2 Změna práv uživatele
  - 4.2.1 Vyhledání uživatele
- 4.3 Export dat
  - 4.3.1 Export měření do formátu CSV
  - 4.3.2 Export měření do formátu XML
  - 4.3.3 Export měření pro zařízení umožňující simulaci připojení

2.4.2 Nefunkční požadavky

- Aplikace bude napsána v programovacím jazyku PHP s použitím frameworku Nette.
- Aplikace bude plně funkční ve všech hlavních vyhledávačích.
- Aplikace bude mít přehledné uživatelské rozhraní, které zajistí snadné provádění všech hlavních úkonů.

2.5 Výběr technologií

Na základě nefunkčních požadavků použiji pro implementaci aplikace programovací jazyk PHP s frameworkem Nette. Tato sekce představí jazyk PHP a některé další programovací jazyky, které by mohly být k implementaci použity. Následně popíše výběr datového úložiště. Dále představí možné JavaScriptové knihovny, které by se daly použít k implementaci datových tabulek a statistických grafů. Nakonec všechny možnosti rozebere a odůvodní jejich volbu.

2.5.1 Programovací jazyky pro tvorbu webu

Jedny z hlavních jazyků pro tvorbu webových aplikací jsou PHP, Java a Ruby.

2.5.1.1 PHP

Skriptovací jazyk PHP je jedním z nejrozšířenějších prostředku pro tvorbu dynamických webových aplikací. Je unikátní platformní nezávislostí a v dnešní době ho podporuje většina webových serveru. Nejznámější a nejpoužívanější z nich je server Apache. PHP umožňuje procedurální i objektově orientované programování. Jeho výhodou je jednoduchost, a tak není náročný ani pro programátorské nováčky.

PHP obsahuje mnoho knihoven zjednodušujících práci s texty, databázemi i grafikou. Nejčastěji se používá současně s MySQL databázemi, se kterými komunikuje pomocí dialektu jazyka SQL. Ten obsahuje příkazy pro definici dat, pro jejich manipulaci, dále příkazy pro řízení přístupových práv a nebo pro řízení datových transakcí.

Nejdůležitější vlastnost jazyka PHP je však jeho provázanost s technologiemi používanými pro tvorbu webových stránek. Tedy:

- HTML (Hypertext markup language) – značkovací jazyk pro formátování textu zobrazovaného na webových stránkách. Dříve sloužil i ke grafické úpravě stránek, nyní tuto činnost zastávají kaskádové styly.
- CSS (Cascading style sheets) – jazyk používající se k popisu způsobu zobrazení stránek vytvořených v jazycích HTML, XHTML nebo XML

- JavaScript – skriptovací jazyk, který se používá k vytváření dynamických prvků webové stránky. Na rozdíl od PHP je vyhodnocován na klientské straně. Umožňuje například vytváření různých animací a efektů na stránce.

2.5.1.2 Java

OOP jazyk Java je jedním z nejpoužívanějších a velmi oblíbených programovacích jazyku na světě. Díky své přenositelnosti je používán pro programy, které mají pracovat na různých systémech, počínaje čipovými kartami, přes mobilní telefony, aplikace pro desktopové počítače, až po rozsáhlé systémy pracující na řade spolupracujících počítačů. Je to jazyk volně šiřitelný a je nezávislý na operačním systému.

pro vytváření webové stránky se používá Java EE (enterprise Java)[6], jejíž základní kameny jsou:

- Servlet – javovská třída, která se stará o vyřizování HTTP požadavků (metody doGet(), doPost() atd.). Servlet je namapovaný na určitou cestu v URL (např. /upload). Dá se použít v roli controlleru nebo třeba pro nízkoúrovňové věci typu upload/download souborů.

- JSP (Java Server Pages) – dokument založený na XML. Představuje šablonovací systém (podobně jako Smarty nebo Latte pro PHP). Při nasazení na server se JSP překládají do jazyka Java a následně se kompilují – vznikne z nich normální servlet. JSP se nemusí používat jen pro vytváření HTML či XHTML stránek, můžeme z nich generovat třeba SVG nebo RSS/Atom, jakýkoliv XML formát nebo prostý text.

- Filtr – mezi zdrojem obsahu (JSP, servlet) a klientem může být řetězec filtrů, který si můžeme představit jako unixové roury, skrz které prochází proud dat. Filtry mají různé využití (transformace dat, autorizace, komprese, atd.)

- JavaBean – třída, která se používá jako prostředník mezi JSP a logikou aplikace.

- EJB (Enterprise JavaBean) – komponenta běžící na serveru, která poskytuje určitou službu (implementuje rozhraní) a je vhodná pro budování modulární architektury. EJB zapouzdřují byznys logiku aplikace a je možné k nim přistupovat i přes síť (např. pomocí protokolu CORBA/IIOP).

2.5.1.3 Ruby

Ruby je interpretovaný skriptovací jazyk, který je přísně objektově orientovaný. Programovacích jazyků nabízejících objektově orientované programování

je celá řada, ovšem Ruby patří mezi čistě objektové jazyky, kde výpočet probíhá výhradně interakcí objektů, a to vzájemným zasíláním zpráv.

pro vývoj webových aplikací se používá framework Ruby on Rails[7]. Je navržen tak, aby usnadnil programování webových aplikací na základě obecných předpokladů vývoje pro web. Umožňuje psát méně kódu a zároveň dosáhnout více, než v mnoha jiných jazycích a platformách. Ve svém jádru obsahuje architekturu MVC (Model, View, Controller), která odděluje aplikační logiku od uživatelského rozhraní, zajišťuje znovupoužitelnost kódu a má přehlednou strukturu usnadňující údržbu aplikace.

2.5.1.4 Zhodnocení jazyků

Všechny představené jazyky splňují nefunkční požadavky na aplikaci. Nefunkční požadavek na programovací jazyk PHP je tedy vhodná volba pro implementaci práce. Navíc kolega i já máme nejvíce zkušeností právě s jazykem PHP.

2.5.2 PHP frameworky

Jedno z doporučení vedoucího naší práce bylo použití frameworku, který výrazně ušetří čas při vytváření aplikace a urychlí tak její vývoj. Dalšími výhodami většiny PHP frameworků jsou:

- přehledné rozdělení aplikace do MVC architektury
- jednodušší práce s databází a databázovými objekty
- řešení autorizaci uživatelů, správa sessions a routování adres
- podpora moderní technologie (např. AJAX - Asynchronous Javascript and XML)

Hlavními představiteli PHP frameworků v Čechách jsou Symfony a Nette. Oba tyto opensource frameworky si jsou velice podobné. Největší rozdíl mezi nimi je v komunitě a dokumentaci. Symfony nabízí obrovskou komunitu a dobře propracovanou dokumentaci a je používané po celém světě. Nette je český framework s poměrně velkou tuzemskou komunitou a ucházející dokumentací. Co se týče použití, Symfony je poněkud robustnější a vhodnější pro středně velké, spíše větší projekty. Nette je vhodné pro malé až středně velké projekty.

Jelikož náš projekt představuje poměrně malou aplikaci, nefunkční požadavek na framework Nette je naprosto v pořádku. Navíc máme s kolegou opět oba nejvíce zkušeností právě z frameworkem Nette, tudíž bude úspora času mnohem větší, než kdybychom studovali nový framework.

2.5.2.1 Nette [8]

Framework Nette nabízí mimo urychlení práce a znovupoužitelnosti kódu také další funkce a mimořádné vlastnosti. Jsou to:

- dokonalé zabezpečení – Nette používá technologie, které eliminují výskyt bezpečnostních děr v aplikaci a jejich následné zneužití. Jsou to útoky typu XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), session hijacking atd.
- objektový návrh – Používá čistý objektový návrh využívající nových vlastností PHP 5.
- volnost – Nette se dá kombinovat s jinými frameworky.
- moderní framework – Využívá a podporuje nové technologie (např AJAX, Dependenci Injection, SEO, atd.).
- MVP – Nette odděluje logickou část (Presenter) aplikace od datové (Model) a prezentační (View). Tvoří tak třívrstvou architekturu.
- podpora připojení k různým typům databází – Mimo jiné i k PostgreSQL databázi, což znamená snadné připojení databázového serveru (viz sekce 2.1.1.2).

Jednotlivé části architektury frameworku budou dále představeny v kapitole 3.

2.5.3 Výběr datového úložiště

Protože jsme nechtěli výrazně měnit schéma existující serverové databáze, rozhodli jsme se s kolegou pro vytvoření databáze pro náš webový server, která bude oddělená od serverové. Zvolili jsme MySQL databázi.

2.5.4 Výběr JavaScriptové knihovny

Součástí mé práce je zobrazení statistických měření uživateli do tabulek a grafů. Je tedy vhodné poohlédnout se po nějakých JavaScriptových knihovnách.

2.5.4.1 Výběr tabulek

při výběru tabulkové knihovny jsem si stanovil několik funkcí, které tabulky umožňují. Jsou to:

- možnost filtrování
- řazení ve sloupcích tabulky
- stránkování výsledků

Kritérium	jsGrid	DataTables	tablesorter
možnost filtrování	ano	ano	ne
řazení ve sloupcích tabulky	ano	ano	ano
stránkování výsledků	ano	ano	ne
přizpůsobení tabulky	ano	ano	ano

Tabulka 2.1: Srovnání JavaScriptových knihoven pro tabulky

- přizpůsobení tabulky

Vyhledal jsem několik zajímavých rozšíření pro knihovnu jQuery, mezi které patří:

- jsGrid [9]
- DataTables [10]
- tablesorter [11]

Srovnání jednotlivých rozšíření shrnuje tabulka 2.1
JsGrid a DataTables splňují kladené požadavky. Nakonec jsem zvolil uživatelsky přívětivější a s lepší dokumentací rozšíření DataTables.

2.5.4.2 Výběr grafové knihovny

U grafových knihoven jsem sledoval tato kritéria:

- zobrazení popisků dat
- zachycení události při kliknutí na bod v grafu
- možnost úpravy vzhledu mapy
- opensource

Vyhledal jsem několik zajímavých rozšíření pro knihovnu jQuery, mezi které patří:

- Chart.js [12]
- CanvasJS [13]
- jqPlot [14]
- Highcharts [15]

Kritérium	Chart.js	CanvasJS	jqPlot	Highcharts
zobrazení popisků dat	ano	ano	ne	ano
událost klik na bod v grafu	ano	ne	ne	ano
možnost úpravy vzhledu mapy	ano	ano	ano	ano
opensource	ano	ne	ano	ne

Tabulka 2.2: Srovnání grafových JavaScriptových knihoven

Srovnání jednotlivých rozšíření shrnuje tabulka 2.2

Moji volbu výrazně ovlivnilo, zda se jedná o opensource knihovnu. Tím se mi výběr zúžil na Chart.js a jqPlot. Graf v naší aplikaci by měl být co nejvíce interaktivní, a proto jsem si vybral knihovnu Chart.js. Dalším významným plusem je i její velmi přehledná dokumentace, je založena na HTML5 a podporuje všechny prohlížeče.

2.5.5 Další technologie

pro snadnou práci grafickou úpravou stránek jsme se s kolegou rozhodli využít CSS frameworku Bootstrap[16], který nabízí snadnou tvorbu stránek pro různé typy zařízení. Dále obsahuje HTML a CSS komponenty a jQuery rozšíření pro ještě efektivnější práci s designem aplikace.

Dále jsme se rozhodli využít CSS preprocesor LESS. Tedy nástroj, který usnadňuje psaní CSS stylů a má několik výhod.

- Vytváření proměnných – Vlastnost, kterou ocení každý programátor. Při tvorbě stylů můžeme definovat různé proměnné (např. pro barvu, odsazení v pixelech, atd.).
- Tzv. mixins – představují sloučení několika CSS vlastností (tedy něco jako objekty v programování)
- Vnořená pravidla – si nejužitečnější vlastnost LESSů. Umožňují totiž vnoření elementů do sebe, což výrazně snižuje délku kódu a urychluje vývoj aplikace.
- Aritmetické operace – například délka prvku bude dva krát nějaká definovaná proměnná
- Vytváření funkcí

Kapitola 3

Návrh

Tato kapitola se zabývá návrhem řešení systému, jehož požadavky byly určeny v kapitole předchozí. Je zde představeno databázové schéma aplikace a představen návrh uživatelského rozhraní částí aplikace, které mám implementovat. Na konci této kapitoly bude jasné, jak má systém fungovat, jaké důležité části obsahuje a jakým způsobem bude implementován.

3.1 Databázové schéma

Před návrhem databázového schématu jsme s kolegou řešili problém, kam budeme ukládat open data ze systému netmetr.cz. Nabízeli se dvě možnosti. Buď bychom tyto data importovali do již existující serverové databáze a nebo si pro tyto data vytvořili tabulky v naší databázi.

Problém první možnosti, tedy nahrání open dat do serverové databáze, je ve struktuře open dat. Kvůli jejich nedostatku některých parametrů měření bychom museli předělávat serverovou databázi a to by mohlo poškodit stávající mechanismus ukládání dat z mobilních klientů.

Oproti tomu vytvoření nových tabulek pro měření ze serveru netmetr.cz znamená, že budeme mít dvě databáze. Veškeré dotazy vybírající data se tedy budou muset provádět jak nad serverovou databází, tak nad databází webové aplikace.

Obě možnosti mají své vady, ale při rozhodování jsme kladli větší důraz na zachování stávající serverové databáze a zvolili možnost druhou.

Návrh databázového schématu popisuje obrázek 3.1. Je zde zobrazeno schéma databázových tabulek, které budou v databázi použity. Jedná se o tabulky:

- `User` – řádek tabulky představuje jednoho uživatele aplikace. Ukládá práva uživatele a další autorizační data.
- `General_settings` – řádek tabulky představuje jedno globální nastavení aplikace. Sloupec `checkbox` slouží k odlišení textových polí a checkboxů.

- `Measure` – tabulka představující jedno měření ze serveru netmetr.cz.
- `Tecnology` – váže se k měření a představuje typ technologie, jakou bylo měření prováděno.
- `IP` – váže se k měření a představuje IP adresu, ze které bylo měřeno.
- `Operator` – váže se k měření a představuje typ operátora.
- `Platform` – váže se k měření a představuje typ platformy zařízení (Nokia, Samsung, atd.).
- `Model` – váže se k měření a představuje typ model zařízení (Galaxy S5, Nexus, atd).
- `Autonomous_system` – váže se k měření a představuje typ autonomního systému.
- `Network` – váže se k měření a představuje druh sítě, přes kterou se měřilo.
- `Measure_location` – vztahová tabulka pro tabulky `Measure` a `Location`.
- `Location` – řádek tabulky představuje přiřazení města k GPS souřadnicím.
- `City` – řádek tabulky představuje přiřazení kraje k městu.
- `County` – řádek tabulky představuje přiřazení kraje k státu.
- `Country` – řádek tabulky představuje stát.

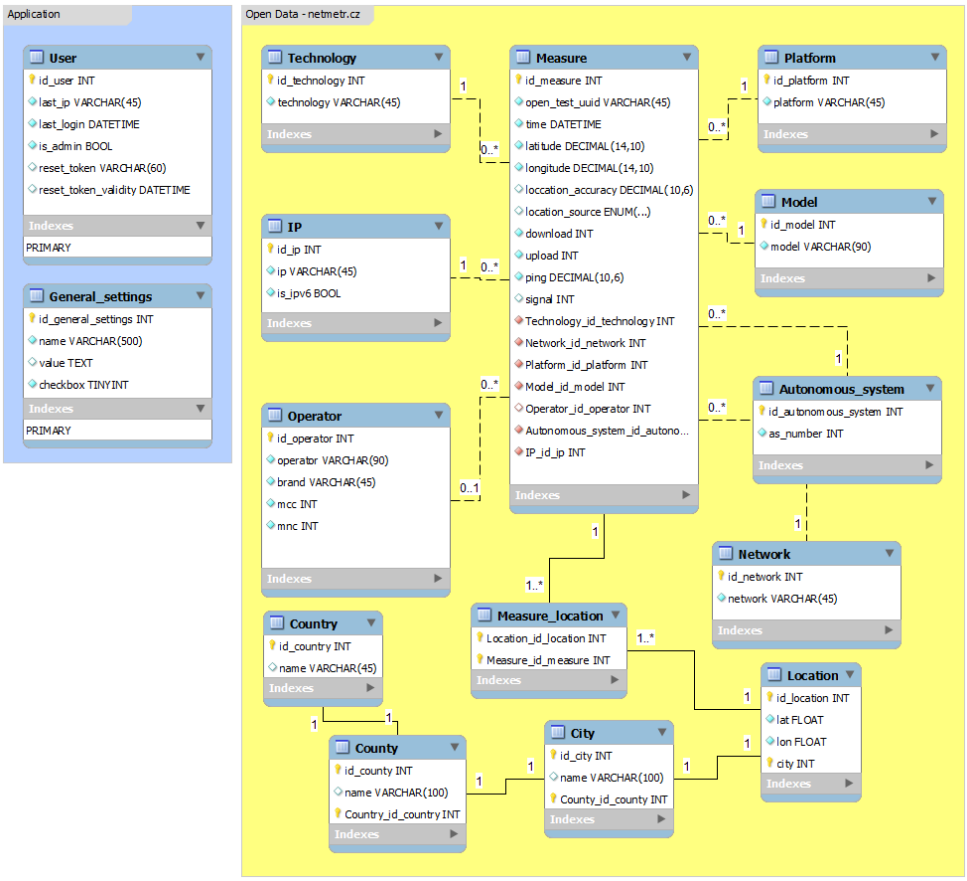
3.2 Případy užití

Seznam všech požadavků na aplikaci nejlépe shrnují UC (use case) diagramy, tedy diagramy případů užití modelované ve struktuře UML. Znázorňují, jaké funkce bude uživatelům s určitou rolí aplikace poskytovat.

3.2.0.1 Anonymní uživatel

Z diagramu nepřihlášeného uživatele 3.2 vyplývají následující scénáře:

- Zobrazení domovské stránky
  1. Uživatel v prohlížeči zadá url adresu projektu MBTest.
  2. Aplikace vrátí prohlížeči tělo odpovědi s HTML domovské stránky, kterou prohlížeč uživateli zobrazí.
- Přihlášení



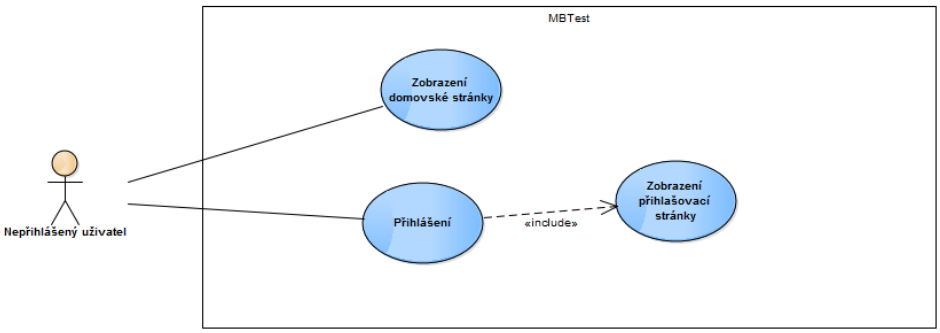
Obrázek 3.1: Databázové schéma webové aplikace

1. Uživatel v prohlížeči zadá url adresu přihlašovací stránky projektu MBTest nebo klikne na odkaz v horním menu stránky.

2. Aplikace vrátí prohlížeči tělo odpovědi s HTML přihlašovací stránky, kterou prohlížeč uživateli zobrazí.

3. Uživatel zadá své přihlašovací údaje a potvrdí přihlašovací formulář.

4. Aplikace ověří přihlašovací údaje a nastávají dvě situace. Přihlašovací údaje jsou neplatné a v tom případě aplikace vrátí chybovou zprávu, které je prohlzzížečem zobrazena. Nebo jsou údaje v pořádku a v tom případě aplikace uživatele přihlásí a přesměruje ho na stránku s jeho statistikami.

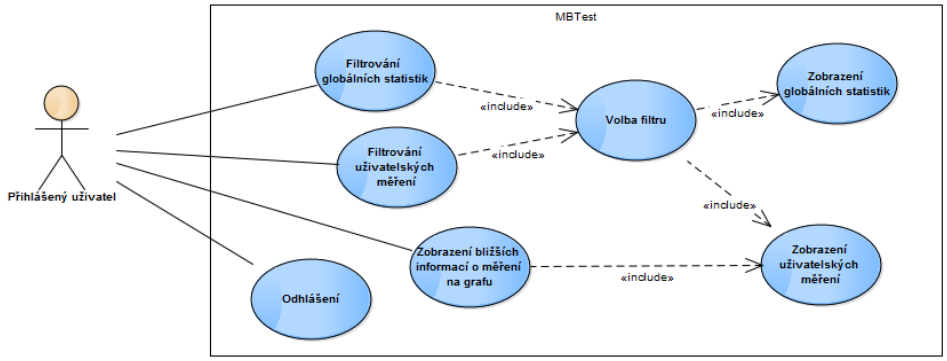


Obrázek 3.2: Diagram případů užití pro anonymního uživatele

3.2.0.2 Přihlášený uživatel

Z diagramu přihlášeného uživatele 3.3 vyplývají následující scénáře:

- Filtrování globálních statistik
  1. Uživatel v prohlížeči zadá url adresu stránky s globálními statistikami projektu MBTest nebo klikne na odkaz v horním menu stránky.
  2. Aplikace vrátí prohlížeči tělo odpovědi s HTML stránky globálních statistik, kterou prohlížeč uživateli zobrazí.
  3. Uživatel zvolí typ filtru, podle kterého bude chtít třídit globální statistiky projektu. Například vybere možnost ze select boxu.
  4. Prohlížeč zachytí změnu možnosti v select boxu a pošle aplikaci požadavek na obnovení dat s nově nastavenými možnostmi filtrů.
  5. Aplikace odešle zpět obnovená data prohlížeči, který je uživateli zobrazí.
- Filtrování uživatelských statistik
  1. Uživatel v prohlížeči zadá url adresu stránky se svými statistikami projektu MBTest nebo klikne na odkaz v horním menu stránky.
  2. Aplikace vrátí prohlížeči tělo odpovědi s HTML stránky uživatelských statistik pro přihlášeného uživatele, kterou prohlížeč uživateli zobrazí.
  3. Uživatel zvolí typ filtru, podle kterého bude chtít třídit své statistiky. Například vybere možnost ze select boxu.
  4. Prohlížeč zachytí změnu možnosti v select boxu a pošle aplikaci požadavek na obnovení dat s nově nastavenými možnostmi filtrů.



Obrázek 3.3: Diagram případů užití pro přihlášeného uživatele

5. Aplikace odešle zpět obnovená data prohlížeči, který je uživateli zobrazí.

- Zobrazení bližších informací o měření na grafu

1. Uživatel v prohlížeči zadá url adresu stránky se svými statistikami projektu MBTest nebo klikne na odkaz v horním menu stránky.
2. Aplikace vrátí prohlížeči tělo odpovědi s HTML stránky uživatelských statistik pro přihlášeného uživatele, kterou prohlížeč uživateli zobrazí.
3. Uživatel klikne na bod v grafu statistik.
4. Prohlížeč zachytí klik na určitý bod grafu a zobrazí uživateli detailní informace o měření.

3.2.0.3 Administrátor

Z diagramu administrátora systému 3.4 vyplývají následující scénáře:

- Zobrazení administrace

1. Administrátor v prohlížeči zadá url adresu administrační stránky projektu MBTest nebo klikne na odkaz v horním menu administrační stránky.
2. Aplikace vrátí prohlížeči tělo odpovědi s HTML administrační stránky, kterou prohlížeč uživateli zobrazí.

- Změna základních nastavení

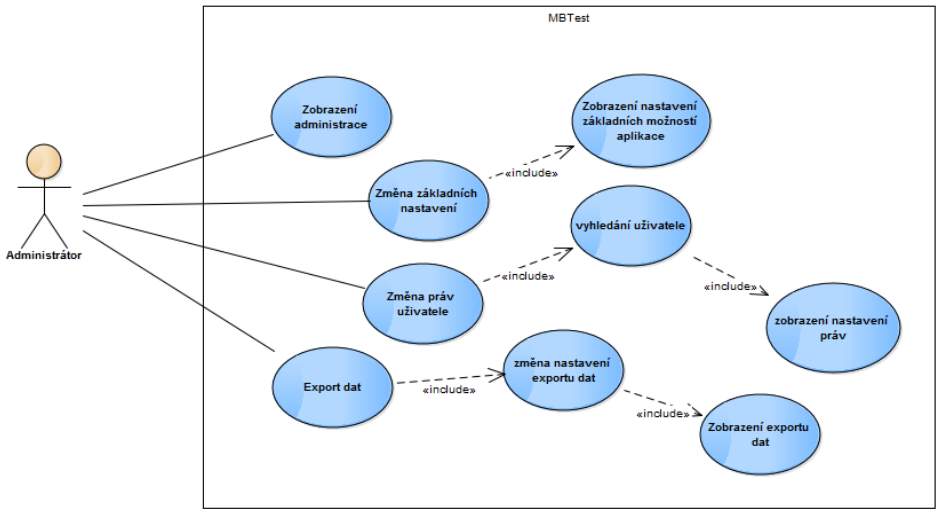
1. Administrátor v prohlížeči zadá url adresu administrační stránky projektu MBTest nebo klikne na odkaz v horním menu aplikace.
2. Aplikace vrátí prohlížeči tělo odpovědi s HTML administrační stránky, kterou prohlížeč administrátorovi zobrazí.
3. Administrátor změní základní nastavení aplikace změnou textů ve formuláři a potvrdí změny tlačítkem.
4. Aplikace odešle prohlížeči zprávu s odpovědí. Pokud se změna provedla v pořádku, pak zobrazí informační hlášku. Pokud nastala chyba při změně nastavení, pak zobrazí chybovou hlášku.

- Změna práv uživatele

1. Administrátor v prohlížeči zadá url adresu administrační stránky pro změnu práv uživatelů projektu MBTest nebo klikne na odkaz v horním menu administrační stránky.
2. Aplikace vrátí prohlížeči tělo odpovědi s HTML administrační stránky pro změnu práv uživatelů, kterou prohlížeč administrátorovi zobrazí.
3. Administrátor napíše část jména nebo emailu uživatele do textového pole pro vyhledávání a odešle tlačítkem.
4. Aplikace vrátí prohlížeči seznam uživatelů odpovídajících vyhledávanému výrazu. Pokud je seznam prázdný, informuje administrátora hláškou. v opačném případě mu seznam uživatelů zobrazí.
5. Administrátor vybere uživatele, u kterého chce provést změnu práv. v select boxu vybere nový typ práv a potvrdí tlačítkem.
6. Aplikace odešle prohlížeči zprávu s odpovědí. Pokud se změna práv provedla v pořádku, pak zobrazí informační hlášku. Pokud nastala chyba při změně práv, pak zobrazí chybovou hlášku.

- Export dat

1. Administrátor v prohlížeči zadá url adresu administrační stránky pro export dat projektu MBTest nebo klikne na odkaz v horním menu administrační stránky.
2. Aplikace vrátí prohlížeči tělo odpovědi s HTML administrační stránky exportu dat, kterou prohlížeč administrátorovi zobrazí.
3. Administrátor provede nastavení exportu dat tak, že nastaví jednotlivé prvky formuláře. Dále administrátor vybere formát exportovaných dat a formulář odešle tlačítkem.
4. Aplikace vybere naměřená data podle nastavení administrátorem a převede je do vybraného formátu a odešle je prohlížeči, který je administrátorovi zobrazí.



Obrázek 3.4: Diagram případů užití pro administrátora

3.3 Návrh uživatelského rozhraní

Rozložení interaktivních a grafických prvků na stránce velmi ovlivňuje funkčnost aplikace a uživatelovi možnosti při interakci s aplikací. Při tvorbě webové aplikace je tedy vhodné využít technologií, které umožní co nejlepší user experience, tedy celkový uživatelův dojem z aplikace. Konkrétně u webových aplikací je nejvýznamnější technologií AJAX (Asynchronous javascript and XML). Je to technologie, která umožňuje JavaScriptu komunikovat se serverem. Toho se využívá k obnovování pouze částí aplikace a ne celé webové stránky.

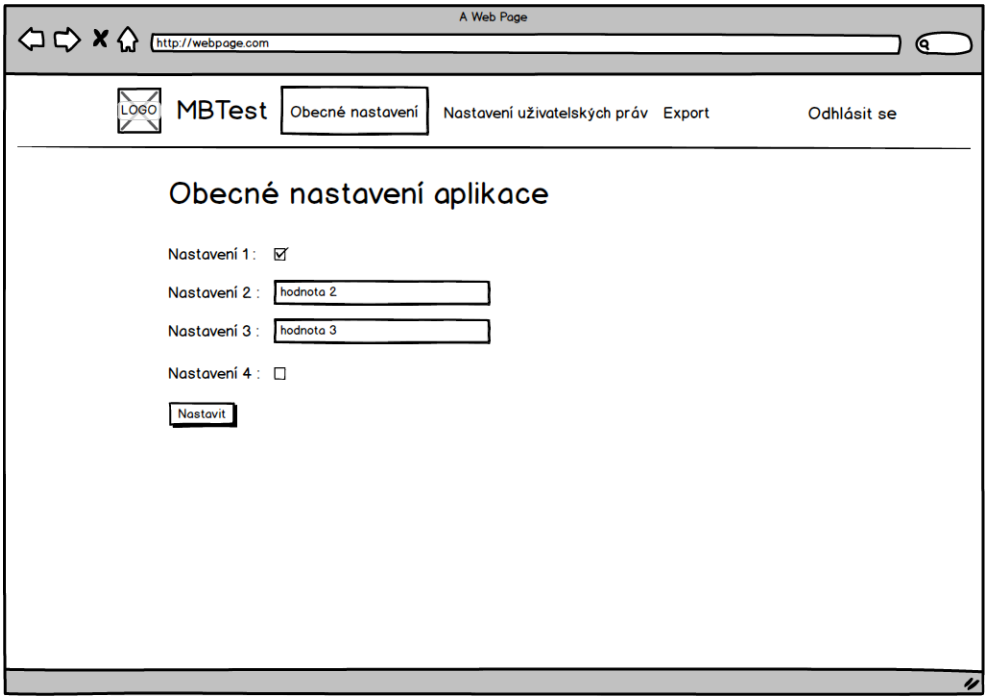
Abych nezačal uživatelské rozhraní vytvářet až při implementaci systému, vytvořil jsem jednoduché wireframy pro nastínění rozložení prvků na stránce a následně popsal, jakým způsobem by měl uživatel s aplikací pracovat.

Wireframe je nákres aplikace, který nezahrnuje žádné grafické prvky, barvy ani typografický styl. Pomocí čar a vzorových textů definuje, co bude kde umístěno a jak budou stránky fungovat. Dá se tvořit jednoduše ručně na papír a nebo se dá využít některého existujícího software pro jeho tvorbu.

Já jsem pro tvorbu použil aplikaci Balsamic moccups, se kterou mám již zkušeností a je velice intuitivní.

3.3.1 Administrační část

Administrační část aplikace bude využívat pouze administrátor. Měla by tedy umožňovat rychlou a intuitivní práci.



Obrázek 3.5: Wireframe globálního nastavení aplikace

Administrátor má na každé stránce administrační části aplikace možnost přejít na jinou administrační stránku pomocí horního menu. Dále se může kdykoliv odhlásit kliknutím na příslušné tlačítko nebo přejít do uživatelské části kliknutím na logo nebo název projektu.

3.3.1.1 Stránka pro změnu globálního nastavení aplikace

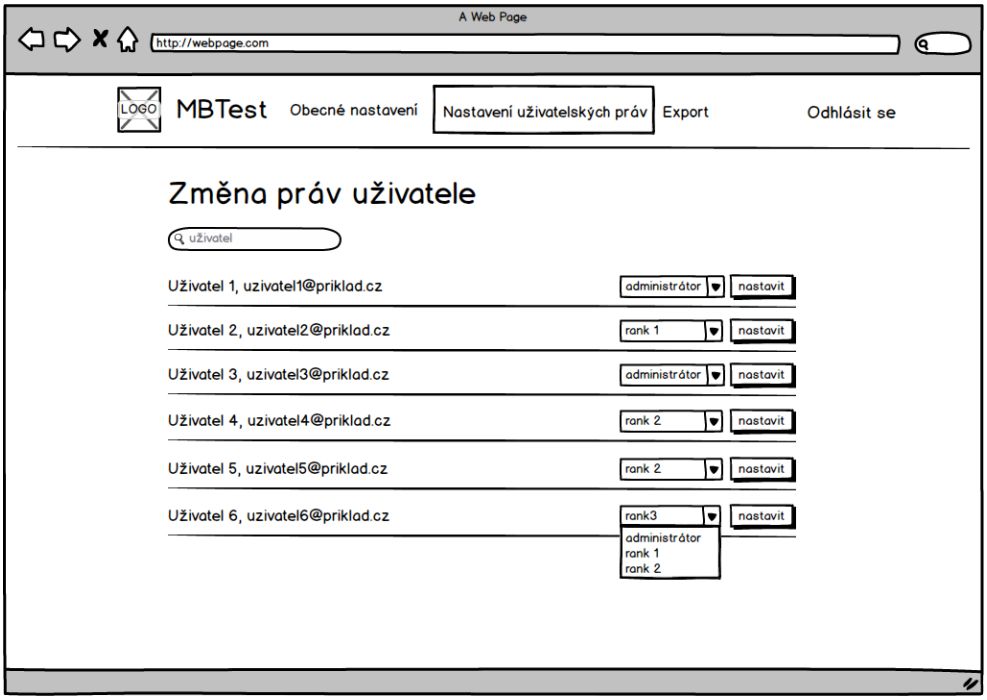
Wireframe stránky je zobrazen na obrázku 3.5.

Pro změnu nastavení administrátor pouze vyplní nebo zaškrtá/odškrtá formulářové prvky a změny potvrdí tlačítkem na konci formuláře.

3.3.1.2 Stránka pro změnu práv uživatelů

Wireframe stránky je zobrazen na obrázku 3.6.

Pro změnu práv uživatele si administrátor vybere ze seznamu uživatelů. Pokud hledaný uživatel v seznamu není, pak administrátor využije vyhledávací textbox. Zde začne psát uživatelův username nebo jeho email. Seznam uživatelů se bude dynamicky měnit podle textu ve vyhledávacím textboxu. Selectboxem vybere nové nastavení práv a potvrdí tlačítkem.



Obrázek 3.6: Wireframe nastavení uživatelských práv

3.3.1.3 Stránka pro export dat

Wireframe stránky je zobrazen na obrázku 3.7.

Pro provedení exportu dat administrátor nastaví pomocí checkboxů jednotlivé parametry měření, které budou zahrnuty v exportu. Dále si vybere formát exportovaných dat označení příslušného radiobuttonu a potvrdí tlačítkem.

3.3.2 Uživatelská část

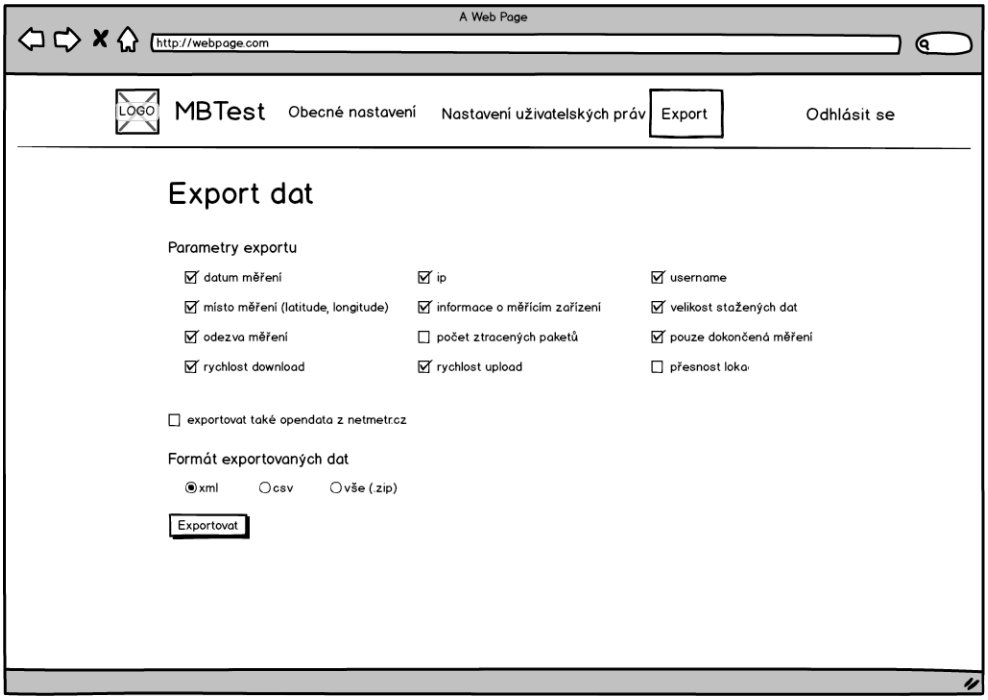
Jelikož celkový design a rozhraní uživatelské části aplikace má na starosti kolega, jsou na wireframech zobrazeny pouze části stránek, které souvisejí s mým zadáním práce.

Uživatel bude mít na každé stránce možnost přejít do jiného typu statistik pomocí menu v horní části obsahu stránky.

3.3.2.1 Stránka s globálními statistikami

Wireframe stránky je zobrazen na obrázku 3.8.

Na této stránce má uživatel možnost zobrazit a filtrovat globální statistiky měření. Uživatel vybere v horním selectboxu stát. Po jeho vybrání se



Obrázek 3.7: Wireframe exportu dat

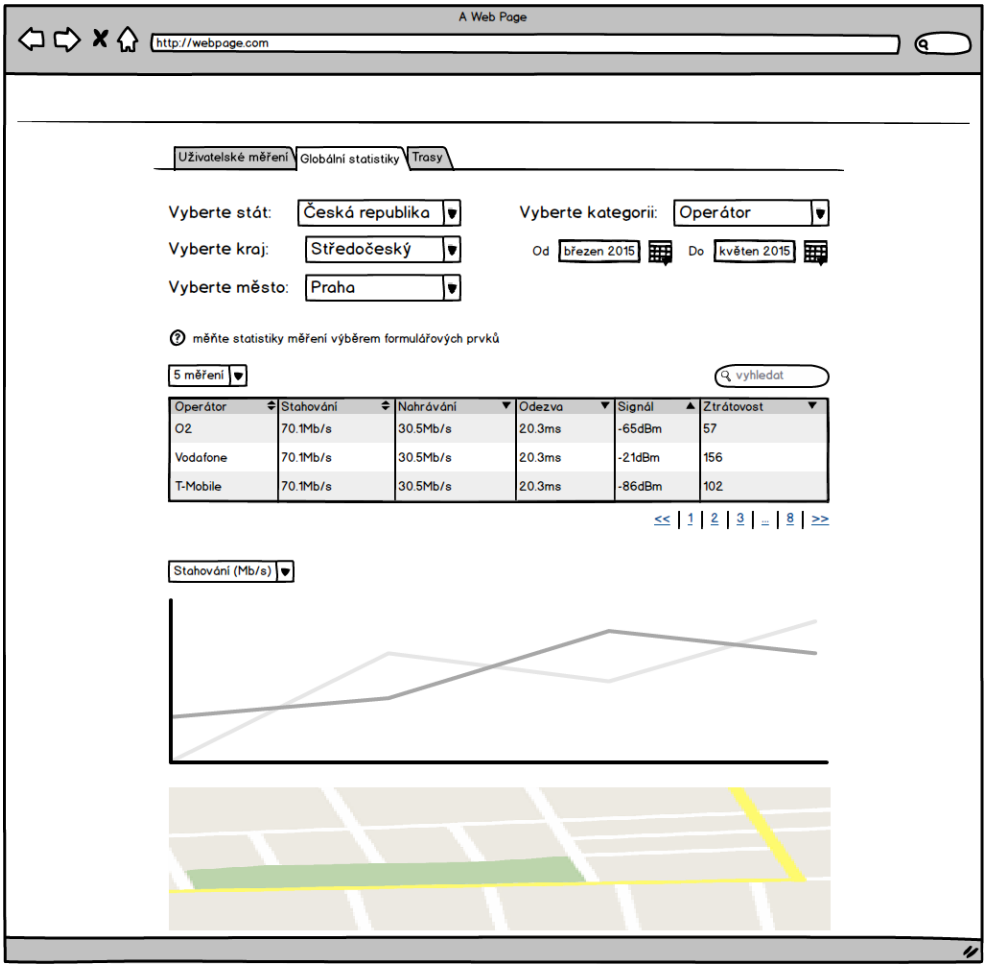
mu zpřístupní výběr kraje. Jakmile uživatel vybere kraj, tak se mu zobrazí selectbox pro výběr města. Kromě toho si ještě uživatel volí kategorii statistik příslušným selectboxem. Při každé změně se dynamicky změní tabulka se statistikami, graf i mapa.

Tabulka dále podporuje stránkování, řazení podle sloupců, vyhledávání a změnu počtu záznamů v tabulce. U grafu a mapy bude uživatel moci měnit parametr měření příslušným selectboxem. Při jeho změně se aktualizují data v grafu i mapě.

3.3.2.2 Stránka s uživatelskými měřeními

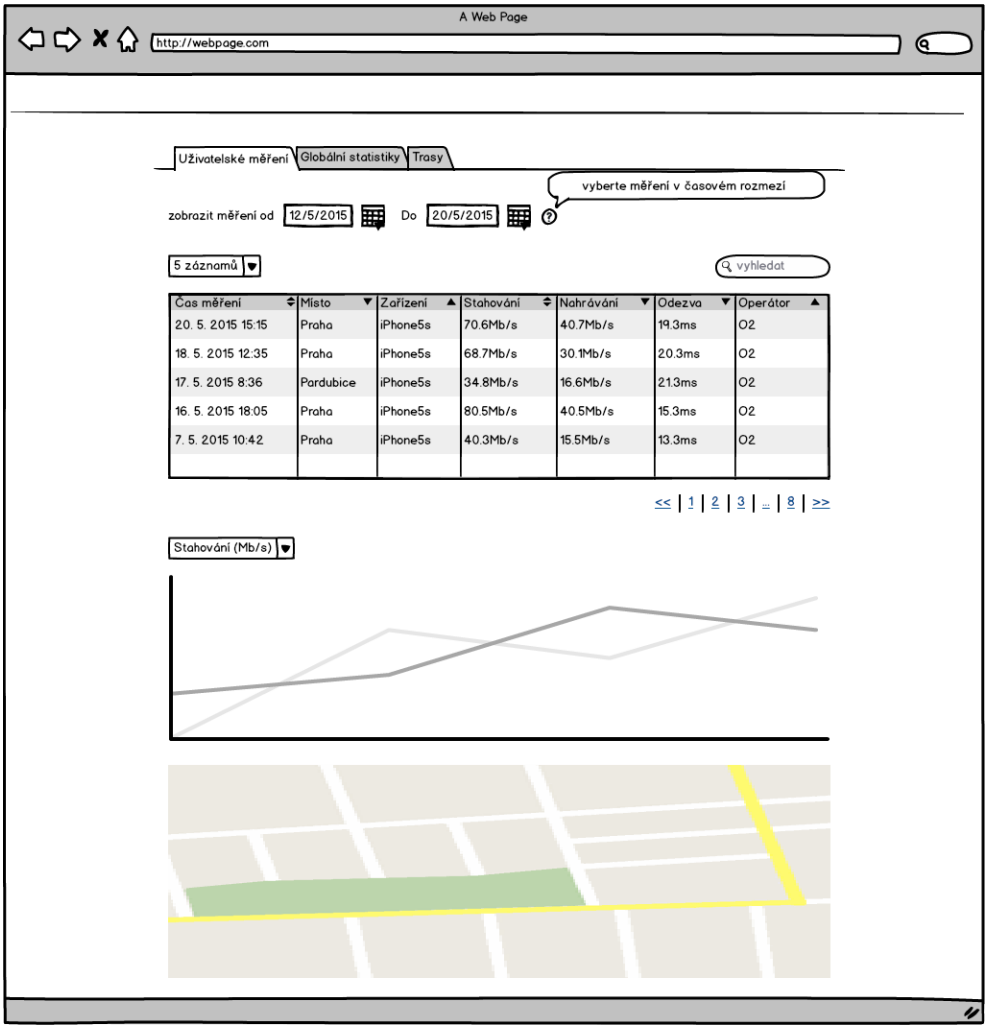
Wireframe stránky je zobrazen na obrázku 3.9.

Na této stránce má uživatel možnost zobrazit a filtrovat svá provedená měření. v tabulce jsou zobrazena uživatelova měření z určitého období vybraného pomocí filtrů nacházejících se nad tabulkou. Samotná tabulka podporuje stránkování, řazení podle sloupců, vyhledávání a změnu počtu záznamů. Při změně se data dynamicky změní data v tabulce, grafu i mapě. U grafu a mapy bude uživatel moci měnit parametr měření příslušným selectboxem. Při jeho změně se aktualizují data v grafu i mapě.



Obrázek 3.8: Wireframe globálních statistik

V grafu budou na vodorovné ose časy měření a na svislé ose hodnoty zvoleného parametru.



Obrázek 3.9: Wireframe uživatelských měření

Obrázek 3.10: Wireframe statistik trasy

3.3.2.3 Stránka s se statistikami trasy

Wireframe stránky je zobrazen na obrázku 3.10.

Na této stránce má uživatel možnost zobrazit statistiky zvolené trasy. Uživatel vybere z jakého města a kam trasa povede. do příslušných textových polí zadává názvy měst, přičemž mu aplikace bude s výběrem pomáhat napovídáním měst podle již napsaného textu. Jakmile budou obě města vybrána, uživatel potvrdí výběr tlačítkem. Uživateli se zobrazí graf s mapou, které budou zobrazovat statistiky měření v průběhu trasy. V grafu na vodorovné ose budou jednotlivé orientační body trasy a na svislé ose hodnoty zvoleného parametru měření. Ten bude uživatel moci měnit příslušným selectboxem a při jeho změně se aktualizují data v grafu i mapě.

Kapitola 4

Realizace

Tato kapitola je věnována vlastní implementaci navrženého systému za použití zvolených implementačních technologií. Nejprve je zde popsána architektura frameworku Nette a popis jeho důležitých částí. Poté je v této kapitole upřesněna tvorba administračního modulu a modulu pro vybírání dat z databáze. Nakonec bude představena implementace exportu statistických dat.

4.1 Nette

Pro framework Nette je charakteristicky návrhový vzor MVP, který je odvozen od vzoru MVC. MVP se skládá ze tří základních vrstev, kterými jsou `Model`, `View` a `Presenter`. Typický případ požadavku na server probíhá podle následujícího scénáře:

1. Klient (webový prohlížeč) odešle požadavek na server.
2. Požadavek se dostane do `Presenteru`, který chce vrátit uživateli odpověď. Nejdříve však potřebuje data, takže začne komunikovat s `Modelem`.
3. `Model` získá data pomocí SQL dotazů z databáze a pošle je `Presenteru`.
4. `Presenter` předá data do `View`.
5. `View` zobrazí data klientovi.

Další významnou charakteristikou je použití šablonovacího systému `Latte`. `Latte` zjednodušuje vytváření jednotlivých `View` pomocí své speciální syntaxe.

Rozdělení složek frameworku popisuje 4.1.

Po sepsání funkčních požadavků bylo jasné, že systém je třeba rozdělit na dvě části. Jedna část přístupná pouze administrátorovi – administrační část. Druhá část, kterou z části uvidí nepřihlášený uživatel a celou ji uvidí přihlášený uživatel a administrátor. Z tohoto důvodu jsme s kolegou přizpůsobili rozložení adresářových složek aplikace. Obměnou oproti klasickému rozdělení

```
www/ ..... kořenový adresář webu
├── dipmbtest/ ..... název projektu
│   ├── app/ ..... adresář webové aplikace
│   │   ├── config/ ..... konfigurační soubory
│   │   ├── forms/ ..... třídy formulářů
│   │   ├── model/ ..... třídy modelové vrstvy
│   │   ├── presenters/ ..... třídy presenterů
│   │   │   └── templates/ ..... šablony
│   │   ├── router/ ..... konfigurace URL adres
│   │   └── bootstrap.php ..... spouštěcí soubor
│   ├── log/ ..... chybové logy
│   ├── temp/ ..... dočasné soubory (cache, session, atd.)
│   ├── vendor/ ..... knihovny
│   │   └── nette/ ..... samotný framework
│   └── www/ ..... místní kořenový adresář webu - pouze ten je přístupný
│       z internetu
```

Obrázek 4.1: Stromový adresář Nette

```
app/ ..... adresář webové aplikace
├── AdminModule/ ..... administrační část
│   ├── presenters/ ..... třídy presenterů administrační části
│   └── templates/ ..... šablony administrační části
├── FrontModule/ ..... uživatelská část
│   ├── presenters/ ..... třídy presenterů uživatelské části
│   └── templates/ ..... šablony uživatelské části
├── forms/ ..... třídy formulářů
├── helper/ ..... pomocné třídy
├── model/ ..... třídy modelové vrstvy
├── presenters/ ..... třídy presenterů
├── router/ ..... konfigurace URL adres
├── bootstrap.php/ ..... spouštěcí soubor
└── config.neon/ ..... konfigurační soubor
```

Obrázek 4.2: Předělaný stromový adresář app

je jiná struktura složky `app/`, které popisuje následující stromová struktura 4.2.

4.1.1 Architektura

Jak bylo zmíněno v předchozí kapitole, základní architektura aplikace je rozdělena na tři vrstvy. Tyto vrstvy budou následně popsány i se seznamem implementovaných tříd.

```
15          database:
16                  default:
17                          dsn: 'mysql:host=localhost;dbname=dip_mbtest'
18                          user: root
19                          password:
20                          options:
21                                  lazy: yes
22                          autowired: true
23
24                  db2conn:
25                          dsn: 'pgsql:host=localhost;dbname=mbtest'
26                          user: postgres
27                          password:
28                          autowired: false
29
```

Obrázek 4.3: Připojení k databázím v app/config.neon

4.1.1.1 Model

První vrstvu tvoří `Model`, tedy datová vrstva aplikace. Konfigurace připojení k databázi se v Nette provádí v souboru `app/config.neon` (viz obrázek 4.3). Na obrázku pracuji s testovacími databázemi na lokálním serveru. `Default` představuje MySQL databázi webové aplikace a `db2conn` je PostgreSQL databáze serverové části aplikace MBTest.

Třídy modelů jsou umístěny ve složce `model/`. Je zde implementováno několik tříd, které dědí od základní třídy `BaseRepository` (rozšiřující třídu `Nette\Object`). Protože máme dvě databáze, bylo třeba určit, jakou databázi bude určitý model používat. Proto jsme do konstruktoru třídy `BaseRepository` přidali jako parametr databázový kontext a při vytváření modelů tedy volíme s jakou databází objekt pracuje. Třídu `BaseRepository` s konstruktorem popisuje následující kód.

```
1  class BaseRepository extends Nette\Object {
2
3    /** @var Nette\Database\Context */
4    protected $database;
5
6      /**
7     * Constructor.
8     * @param Nette\Database\Context $database
9     */
10   public function __construct(Nette\Database\Context $database) {
11       $this->database = $database;
12   }
13
14 }
```

V třídách, které dědí od třídy `BaseRepository` jsou implementovány veškeré metody zajišťující potřebné SQL dotazy. Význam těchto tříd je následující:

- `Authenticator` – Třída zabezpečující autentizace uživatelů. Tedy přihlašování uživatelů do aplikace.
- Třídy připojující se k databázi serveru MBTest.
  - `CitiesRepository` – Tato třída zajišťuje požadavky na data týkající se měst (tabulka `cities`).
  - `CountiesRepository` – Třída, která pracuje s daty kraje (tabulka `counties`).
  - `ConntypeRepository` – Třída, která pracuje s druhy připojení měření (tabulka `conntype`).
  - `CountriesRepository` – Třída pracující s daty států (tabulka `countries`).
  - `LocationsRepository` – Třída, která pracuje s daty lokací (tabulky `locations` a `measure_locations`).
  - `MeasureRepository` – Tato třída zajišťuje požadavky na data týkající měření (tabulka `measure`).
  - `UsersRepository` – Třída zajišťující požadavky na data týkající se uživatelů (tabulka `users`).
- Třídy připojující se k databázi webové aplikace.
  - `Autonomous_systemRepository` – Třída zajišťující požadavky na data týkající se autonomních systémů měření (tabulka `Autonomous_system`).
  - `CityRepository` – Tato třída zajišťuje požadavky na data týkající se měst (tabulka `City`).
  - `CountyRepository` – Třída, která pracuje s daty kraje (tabulka `County`).
  - `CountryRepository` – Třída pracující s daty států (tabulka `Country`).
  - `General_settingsRepository` – Tato třída zajišťuje požadavky na data týkající se obecného nastavení (tabulka `General_settings`).
  - `IPRepository` – Třída, která pracuje s IP adresami měření (tabulka `IP`).
  - `LocationRepository` – Třída, která pracuje s daty lokací (tabulky `Location` a `Measure_location`).
  - `MeasureLocRepository` – Tato třída zajišťuje požadavky na data týkající se měření (tabulka `Measure`).
  - `ModelRepository` – Třída pracující s daty modelů zařízení měření (tabulka `Model`).

- `NetworkRepository` – Třída, která pracuje se sítěmi měření (tabulka `Network`).
- `OperatorRepository` – Tato třída zajišťuje požadavky na data týkající se operátorů zařízení měření (tabulka `Operator`).
- `PlatformRepository` – Třída, která pracuje s platformou zařízení (tabulka `Platform`).
- `TechnologyRepository` – Třída pracující s technologiemi měření (tabulka `Technology`).
- `UserRepository` – Třída zajišťující požadavky na data týkající se uživatelů (tabulka `User`).

Ve složce `model/` se dále nachází třída, která od `BaseRepository` nedědí. Jedná se o třídu `Authorizator`. `Authorizator` dědí od třídy `Nette\Security\Permission` nastavuje jednotlivým rolím (anonymní uživatel, přihlášený uživatel, administrátor) přístup na určité stránky. Je zde tedy například přidělen přístup administrátorům do administrativních stránek a naopak ostatním rolím zakázán. Činí tak funkcí `setRules()`, jejíž obsah je znázorněn následujícím kusem kódu.

```
1 private function setRules() {
2   $this->allow(static::ROLE_GUEST, 'Front:Default');
3   $this->allow(static::ROLE_GUEST, 'Front:Sign');
4   $this->allow(static::ROLE_USER, 'Front:Dashboard');
5   $this->allow(static::ROLE_USER, 'Front:Statistics');
6   $this->allow(static::ROLE_USER, 'Front:Measure');
7   $this->allow(static::ROLE_USER, 'Front:Route');
8   $this->allow(static::ROLE_ADMIN);
9 }
```

4.1.1.2 Presenter

`Presentery` tvoří druhou vrstvu architektury. Adresářová struktura 4.2 ukazuje, že v naší aplikaci existují dva typy `Presenterů`. Jedná se o `Presentery` administrační části (`AdminModule/presenters/`) a `Presentery` uživatelské části (`FrontModule/presenters/`).

Všechny třídy `Presenterů` dědí od třídy `BasePresenter`, která se nachází ve složce `presenters/`. `BasePresenter` zajišťuje celkový chod aplikace a jsou zde implementovány věci společné pro všechny `Presentery`. Jsou to:

- globální proměnné použité v celé aplikaci
- autentizační funkce, které kontrolují práva uživatele při vstupu na konkrétní stránku

- věci, které je důležité vykonat před předání kontroly do `View` (metoda `beforeRender()`)

Seznam tříd `Presenteru` je uveden zde:

- Třídy administrační části (`AdminModule/presenters/`)
  - `DefaultPresenter` – Tato třída zajišťuje požadavky týkající se změny globálního nastavení aplikace.
  - `ExportPresenter` – Zde se jedná o třídu, která zpracovává data měření z databází a exportuje je do souborů zvoleného formátu.
  - `RightsPresenter` – Jedná se o třídu zpracovávající požadavky na změnu práv uživatelů.
- Třídy připojující se k databázi webové aplikace.
  - `DefaultPresenter` – Třída zajišťující zobrazení hlavní stránky aplikace.
  - `MeasurePresenter` – Jedná se o třídu zpracovávající požadavky uživatelských měření.
  - `RoutePresenter` – Tato třída zajišťuje požadavky týkající se statistik trasy.
  - `SignPresenter` – Třída zabezpečující autentizace uživatelů a jejich odhlášení.
  - `StatisticsPresenter` – Jedná se o třídu zpracovávající požadavky na globální statistiky měření.

Pro získání tříd `Modelu`, které chtějí `Presentery` používat k získání dat s databáze využívá Nette tzv. Dependency injection.

Dependency injection – Podstatou Dependency injection (DI) je odebrat třídám zodpovědnost za získávání objektů, které potřebují ke své činnosti (tzv. služeb) a místo toho jim služby předávat při vytváření.

Použití DI popíši na třídě `DefaultPresenter`. v této třídě je potřeba vybrat všechna globální nastavení aplikace (model `General_settingsRepository`). Abych mohl třídu `General_settingsRepository` v presenteru používat, je nutné ji nejprve zaregistrovat jako službu. To se provede v souboru `config.neon` nacházejícího se ve složce `app/`. Následně implementujeme v presenteru metodu `injectRepositories()` jejímž parametrem je modelová třída viz následující kus kódu.

```
templates/ ..... složka s jednotlivými šablonami
├─ Default/ ..... šablony DefaultPresenteru
├─ Export/ ..... šablony ExportPresenteru
├─ Rights/ ..... šablony RightsPresenteru
├─ @layout.latte ..... společná část pro všechny šablony
└─ @navbar.latte ..... horní menu stránky
```

Obrázek 4.4: Adresář šablon administrační části

```
1  class DefaultPresenter extends \App\Presenters\BasePresenter {
2
3      private $settings;
4
5      /**
6       * injects database objects needed
7       * @param \App\Model\General_settingsRepository $settings
8       */
9      public function injectRepositories(\App\Model\
           General_settingsRepository $settings) {
10         $this->settings = $settings;
11     }
12
13     ...
```

4.1.1.3 View

Poslední vrstvou je `View`. Tato vrstva se stará o vykreslení výsledků uživatelských požadavků. Data k vykreslení přijímá od `Presenterů` a následně pomocí HTML kódu doplněného o logiku šablonovacího systému `Latte` data zobrazuje.

Šablonové soubory mají příponu `.latte` a jsou opět rozděleny do administrační a uživatelské složky (viz 4.2). Jelikož jeden `Presenter` může pracovat s více šablonami, jsou následně vytvořený složky s názvy `Presenterů`. Tuto skutečnost popisuje stromový adresář 4.4 pro šablony administrační části. Můžeme na něm vypozorovat významnou vlastnost šablonovacího systému `Latte`. Jedná se o možnost vnořovat jednotlivé šablony do sebe.

4.2 Helpery

Mimo standardních tříd architektury MVP jsem dále implementoval pomocné třídy tzv. `Helpery`. Ty se nacházejí ve složce `app/helper/` a jedná se o třídy `ExportHelper` a `LocationQueryHelper`. Použití těchto tříd v `Presenterech` je stejné jako u `Modelů`. Musejí se tedy opět registrovat v konfiguračním souboru `config.neon` a poté injektovat v potřebném `Presenteru`.

4.2.1 Export dat

Třída `ExportHelper` implementuje možnost exportu dat. Nabízí čtyři možnosti exportu:

- export do formátu CSV – metoda `saveCSV()`
- export do formátu XML – metoda `saveXML()`
- export do formátů CSV a XML – vrátí .zip soubor. Metoda `saveZIP()`
- export pro zařízení simulující konkrétní připojení – metoda `saveDevice()`

4.2.1.1 Příprava nastavení

Celý proces exportu zpracovává presenter `ExportPresenter`, který v momentě vzniku požadavku na export načte jednotlivá nastavení. Nastavení exportu zadává administrátor a toto nastavení je odesíláno jako pole parametrů v POST požadavku. `ExportPresenter` nejprve převede jednotlivé parametry exportu na názvy sloupců tabulky `measure`. Dále získá jednotlivá měření odpovídající nastavení exportu pomocí injektovaného modelu `MeasureRepository`. Poté zjistí, o jaký export dat uživatel požádal, vytvoří dočasný soubor s příslušnou koncovkou a zavolá příslušnou metodu `ExportHelperu`.

4.2.1.2 CSV export

Pro vytváření souborů ve formátu CSV má programovací jazyk PHP funkci `fputcsv()`, která má dva parametry. Prvním je soubor, do kterého budeme zapisovat (získaný funkcí `fopen()`). Druhým parametrem je pole hodnot, které funkce zapíše do souboru jako jeden řádek dat v CSV formátu.

Při CSV exportu se zavolá funkce `saveCSV`, které se předají jako parametry název souboru, pole názvů databázových sloupců, které mají být vybrány a pole vybraných dat z databáze. Funkce následně postupně vkládá řádky dat do souboru pomocí funkce `fputcsv()`.

4.2.1.3 XML export

Pro vytváření XML struktury má programovací jazyk PHP třídu `DOMDocument`. Metody této třídy využiji ke konstrukci XML dokumentu, který následně uložím do souboru. Následující zdrojový kód ukazuje tvorbu XML dokumentu pomocí metody `saveXML()` třídy `ExportHelper`.

```
1  public function saveXML($filename, $headers, $data, $to_file =
       true) {
2          if (empty($data)) {
3              return false;
4          }
5
6          $xml = new \DOMDocument();
7          $xml_export = $xml->createElement("export");
8
9          foreach ($data as $d) {
10             $xml_measure = $xml->createElement("measure");
11             foreach ($headers as $h) {
12                 switch ($h) {
13                 // určité databázové sloupce jsou jen cizí klíče
                       jiné tabulky, proto je potřeba je napojit
                       zvlášť
14                     case "user_id":
15                         $xml_attr = $xml->createElement("username"
                               );
16                         $user = $d->ref("users", "user_id");
17                         if ($user) {
18                             $xml_attr->nodeValue = $user->username
                                   ;
19                         } else {
20                             $xml_attr->nodeValue = '';
21                         }
22                         $xml_measure->appendChild($xml_attr);
23                         break;
24                     case "device":
25                         ...
26                     case "conntype":
27                         ...
28                     default:
29                         $xml_attr = $xml->createElement($h);
30                         $xml_attr->nodeValue = $d->$h;
31                         $xml_measure->appendChild($xml_attr);
32                 }
33             }
34             $xml_export->appendChild($xml_measure);
35         }
36         $xml->appendChild($xml_export);
37
38         if ($to_file) {
39             $xml->save($this->dir . '/' . $filename);
40         } else {
41             return $xml->saveXML();
42         }
43     }
```

4.2.1.4 .zip export

K vytváření archivů v jazyku PHP slouží třída `ZipArchive()`. Její použití v metodě `saveZIP` představuje následují kód.

```
1   public function saveZIP($filename, $headers, $data) {
2
3         $this->saveCSV($filename . ".csv", $headers, $data);
4         $xml = $this->saveXML($filename . ".xml", $headers, $data,
                false);
5
6         $zip = new \ZipArchive();
7
8         // otevření archivu pro~zápis, pokud se nepovedlo vrátíme
              false
9         if ($zip->open($this->dir . '/' . $filename, \ZipArchive::
              CREATE) !== TRUE) {
10            return false;
11        }
12
13        $zip->addFile($this->dir . "/" . $filename . ".csv",
              $filename . ".csv");
14        $zip->addFromString($filename . ".xml", $xml);
15        $zip->close();
16        return true;
17    }
```

4.2.1.5 Export pro simulující zařízení

Jedním z funkčních požadavků na export je možnost exportu pro speciální zařízení pro simulaci konkrétního připojení. Specifikovaným formátem je XML a přesná struktura dat byla dána následovně:

```
1  <sce>
2    <name></name>
3    <desc></desc>
4    <rule>
5      <dlrate></dlrate>
6      <dldelay></dldelay>
7      <dljitter></dljitter>
8      <dlloss></dlloss>
9      <uprate></uprate>
10     <updelay></updelay>
11     <upjitter></upjitter>
12     <uploss></uploss>
13     <time></time>
14   </rule>
15   <rule>
16    ...
17   </rule>
18 </sce>
```

Implementace metody `saveDevice()` je tedy velmi podobná metodě `saveXML()`, jenom se dodržuje daná struktura dat.

4.2.2 Modul pro výběr dat

Druhá část práce ze zabývá výběrem měření a jejich statistik pro určité typy dotazů. Pro tyto potřeby jsem vytvořil helper třídu `LocationQueryHelper`. Tato třída implementuje několik metod pro získání statistických dat, nebo dat související s polohou. Jednotlivé metody budou dále představeny.

4.2.2.1 Statistické metody

Všechny funkce, které vybírají některé statistiky mají vždy dva stejné parametry. Obsahy obou parametrů jsou přesně dané a kontrolují se v každé statistické metodě. Jedná se o:

- `$option` – Název databázového sloupce, ze kterého statistiku děláme. Mezi kontrolované možnosti patří `download, upload, ping_avg, jitter, packet_loss`.
- `$filters` – Pole filtrů, které omezují výběr měření. Mezi kontrolované možnosti patří `device, location_src1, conntype, done, forced`.

Tyto metody vracejí několik statistických ukazatelů pro zvolený parametr měření (`$option`). Jedná se o aritmetický průměr, rozptyl, směrodatnou odchylku a počtem dat, nad kterými byly statistiky prováděny. Tyto čtyři ukazatele budu dále nazývat statistická data.

Výčet statistických metod je následující:

- `getCityStatistics()` – Metoda pro získání statistických dat daného města. Parametrem je identifikátor města.
- `getCountyStatistics()` – Metoda pro získání statistických dat daného kraje. Parametrem je identifikátor kraje.
- `getCountryStatistics()` – Metoda pro získání statistických dat daného státu. Parametrem je identifikátor státu.
- `getLocationStatistics()` – Metoda pro získání statistických dat místa zadaného GPS souřadnicemi. Parametrem jsou GPS souřadnice místa a rozsah. Rozsah je udaný ve stupních (jako jsou GPS souřadnice) přičemž platí, že `0.001° ~ 110m`.

4.2.2.2 Metody využívající OpenStreetMap API

Protože jedním z požadavků na aplikaci je získání statistik podél trasy mezi městy, je potřeba nějak získat informace o poloze měst a hlavně informace o samotné trase. K těmto úkolům jsem využil serverů s OpenStreetMap API (dále jen OSRM). Jazyk PHP používá pro zasílání HTTP požadavků technologii `curl`. `curl` umožňuje nastavit jednotlivé hlavičky HTTP požadavku, odeslat požadavek na server a vrátit z něho odpověď.

V metodách byli použity dva servery.

- `http://router.project-osrm.org/` – Vlastní server OSRM, který jsem využil k získání trasy mezi zadanými městy.

- `http://overpass-api.de/api/interpreter` – Server, na kterém jsou OSRM API s rozšířenými možnostmi. Tento server jsem využil k určení města pro zadanou GPS lokaci.

Implementoval jsem následující metody využívající tyto servery:

- `getRouteCoordinates()` – Metoda pro získání pole GPS souřadnic bodů na trase. Parametrem jsou identifikátory dvou měst.

- `getNearestCity()` – Metoda pro získání informací o nejbližším městě pro zadané GPS souřadnice. Parametrem jsou GPS souřadnice místa a rozsah. Rozsah je udaný ve stupních (jako jsou GPS souřadnice) přičemž platí, že `0.001° ~ 110m`.

Oba servery vracejí data ve formátu JSON a odpověď je komprimovaná metodou GZIP. Proces odeslání, přijetí a rozparsování odpovědi v metodě `getRouteCoordinates()` demonstruje následující zdrojový kód.

```
1  // inicializace curl
2  $ch = curl_init();
3  curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
4  curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);
5  curl_setopt($ch, CURLOPT_HEADER, 1);
6  curl_setopt($ch, CURLOPT_URL, $request);
7  curl_setopt($ch, CURLOPT_MAXREDIRS, 10);
8  curl_setopt($ch, CURLOPT_HTTPHEADER, array("Accept: text/html,
       application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q
       =0.8", "Connection: keep-alive", "Accept-Encoding: gzip,
       deflate, sdch", "Accept-Language: cs-CZ,cs;q=0.8,en;q=0.6", "
       User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit
       /537.36 (KHTML, like Gecko) Chrome/43.0.2357.124 Safari/537.36
       ", "Cache-Control: max-age=0"));
9
10 // odeslání požadavku
11 $response = curl_exec($ch);
12
13 if (!$response) {
14     return NULL;
15 }
16
17 $header_size = curl_getinfo($ch, CURLINFO_HEADER_SIZE);
18 $body = substr($response, $header_size);
19
20 // dekóduje string odpovědi na~JSON objekt
21 $json = json_decode(gzdecode($body));
22
23 if ($json == NULL) {
24     return NULL;
25 }
26
27 $coords = array();
28
29 // dekoduje polyline na~GPS souřednice
30 $decoded = Polyline::Decode($json->route_geometry);
31
32 for ($i = 0; $i < count($decoded); $i += 2) {
33     $coord = array();
34     $coord["latitude"] = $decoded[$i];
35     $coord["longitude"] = $decoded[$i + 1];
36     $coords[] = $coord;
37 }
```

Dalším zádrhelem byla forma přijatých GPS souřadnic. OSRM totiž využívá zakódování souřadnic do tzv. Polyline řetězce [17]. K jeho odkódování jsem použil opensource knihovnu `google-map-polyline-encoding-tool`[18]. Metoda `Decode()` dekóduje řetězec polyline na pole GPS souřadnic. Získání jednotlivých souřadnic je znázorněn v předchozí ukázce zdrojového kódu.

4.3 Implementace uživatelského rozhraní

Uživatelské rozhraní aplikace vycházelo z návrhu (3.3). Při implementaci za použití technologií výrazně ulehčujících práci (Bootstrap, LESS, viz 2.5.5) jsem nenarazil na žádný problém.

Použití zvolených knihoven pro zobrazování grafů a tabulek bylo snadné díky jejich dobré dokumentaci.

4.4 Shrnutí implementace

Implementace požadovaných částí webové aplikace, která probíhala v návaznosti na vytvořený návrh systému, byla úspěšná. Byly naprogramovány veškeré kladené funkční požadavky.

Časově nejnáročnější bylo nastudování OSRM API pro dotazování se na GPS souřadnice trasy a následné vybrání technologií pro zpracování nalezeného výsledku.

Tvorbu uživatelského rozhraní výrazně urychlily vybrané technologie. O veškeré funkční vlastnosti a základní logiku aplikace se postaral framework Nette, který se mi opět úspěšně osvědčil.

Kapitola 5

Testování

Pro zaručení správné funkčnosti a spolehlivosti aplikace je otestování nezbytnou součástí každého systému. Hlavním cílem je snížení možnosti a také pravděpodobnosti výskytu chyb, které se mohou objevit při běžném používání systému.

5.1 Testování při vývoji

Průběžné testování funkčních požadavku specifikovaných v analytické části práce, viz sekce 2.4.1, probíhalo již ve fázi realizace. Veškerá implementace byla ručně otestována ihned po vytvoření nebo po jakékoliv provedené změně. Využití tohoto způsobu testování umožnilo včasné nalezení vznikajících chyb, které by bylo mnohem složitější dohledat v době, kdy projekt začínal nabývat větších rozměrů.

Pro účely tohoto testování byl použit nástroj Laděnka, který je přímo součástí frameworku Nette. Tento nástroj umožňuje rychle odhalit a opravit chyby, dále také tyto chyby popřípadě logovat a důležitá vlastnost je také možnost vypisovat proměnné systému.

Pro testování použití javascriptových knihoven byl využit zabudovaný nástroj v prohlížeči `Google Chrome`. Jedná se o javascriptovou konzoli, do které lze vypisovat jakékoliv proměnné příkazem `Console.log()`.

5.2 Localhost

Testování na lokálním serveru je nedílnou součástí vývoje každé webové aplikace. Localhost je softwarový server, který běží přímo na osobním počítači. Jelikož jsem práci vyvíjel v operačním systému Windows, zvolil jsem XAMPP server. Ten obsahuje vše potřebné pro vývoj aplikace, tedy Apache, MySQL, PostgreSQL, phpMyAdmin a spoustu dalších rozšíření.

Na tomto serveru běžela námi vytvořená MySQL databáze s importovanými daty ze serveru netmetr.cz. Dále jsme měli k dispozici dump serverové databáze MBTest naplněné reálnými daty z mobilních aplikací.

Testování s reálnými daty odhalilo několik zásadních chyb aplikace, které byly následně opraveny.

5.3 Ostrý server

Nasazení na ostrý server proběhne ke konci roku 2015. Od té chvíle bude aplikace dostupná na doméně *www.mbtest.cz*.

Závěr

Cílem této diplomové práce byl návrh a implementace modulů webové aplikace systému MBTest. Tyto moduly by umožňovali uživatelům zobrazovat statistická data v grafech a tabulkách. Dále by administrátorům umožňovaly měnit nastavení webové aplikace společně s možností změny práv uživatelů a také s možností exportu dat v různých formátech.

Před započetím realizace byla provedena analýza existujících části systému MBTest a také průzkum konkurenčních aplikací. v rámci analýzy byly rozlišeny uživatelské role a vytyčeny hlavní požadavky na moduly webové aplikace. Dalším bodem analýzy byl přehled technologií využívajících se pro tvorbu webových aplikací zakončený jejich odůvodněním výběrem.

Jakmile byly známy veškeré požadavky na aplikaci a byly vybrány potřebné technologie, začalo se s návrhem webové aplikace. Nejprve bylo navrženo databázové schéma webové aplikace. Poté byly rozebrány všechny případy na jejichž základě bylo navrženo uživatelské rozhraní modulů webové aplikace, u kterého se kladlo co nejvíce na přehlednost hlavních ovládacích prvků aplikace a efektivitu práce s ní.

Realizace vycházela z návrhu a také z architektury použitého frameworku Nette. Všechny části architektury frameworku byly implementovány poměrně rychle. Více času však spotřebovala realizace pomocných tříd neboli očekávaných modulů. Uživatelské rozhraní bylo implementováno podle detailně zpracovaného návrhu.

Aplikace byla testována průběžně při implementaci s využitím testovacích technologií typických pro vývoj webových aplikací.

Za osobní přínos této práce považuji v první řadě možnost vyzkoušet si nové technologie v praxi. v neposlední řadě je to zkušenost s vývojem aplikace, která bude mít využití v praxi.

Realizace aplikace splnila všechny požadavky určené v analýze. s nově nabytými zkušenostmi bych některé části aplikace již dnes řešil jinak.

Další vývoj projektu

Ostré nasazení systému proběhne až na konci roku 2015, a proto je spousta času na další zlepšování projektu. Mezi možné vylepšení bych zařadil například možnost historie exportovaných dat. Také mám v plánu průběžně pracovat na dalších vylepšeních uživatelského rozhraní.

Literatura

[1] WATZKE, D.: *David Watzke / 2013 / Archiv diplomových pracĂ-ČVUT [online]*. Květen 2013, [cit. 2015-06-22]. Dostupné z: `https://dip.felk.cvut.cz/browse/pdfcache/watzkdav_2013bach.pdf`

[2] Adam Haken: *Test rychlosti připojenĂ k internetu [online]*. [cit. 2015-06-22]. Dostupné z: `http://rychlost.cz/`

[3] DSL.cz: *Výsledky mýeření rychlosti internetového připojení [online]*. [cit. 2015-06-22]. Dostupné z: `http://www.dsl.cz/test-mereni-rychlosti`

[4] Univerzity of Michigan, Northeastern Univerzity, University of Washington, M-lab: *MobiPerf [online]*. [cit. 2015-06-22]. Dostupné z: `https://sites.google.com/site/mobiperfdev/`

[5] cz.nic: *Úvod: NetMetr*. [cit. 2015-06-22]. Dostupné z: `https://www.netmetr.cz/`

[6] Kučera, F.: *Java na webovém serveru: první web / Zdroják [online]*. [cit. 2015-06-22]. Dostupné z: `http://www.zdrojak.cz/clanky/java-na-webovem-serveru-prvni-web/`

[7] rubyonrails.cz: *Ruby on Rails Guides: Začínáme s Rails [online]*. 2015, [cit. 2015-06-22]. Dostupné z: `http://guides.rubyonrails.cz/`

[8] Nette framework: *Nette framework [online]*. 2015, [cit. 2015-06-22]. Dostupné z: `http://doc.nette.org/cs/`

[9] jsGrid: *Lightweight Grid jQuery Plugin [online]*. 2015, [cit. 2015-06-22]. Dostupné z: `http://js-grid.com/`

[10] DataTables: *DataTables / Table plug-in for jQuery [online]*. 2015, [cit. 2015-06-22]. Dostupné z: `https://www.datatables.net/`

[11] Bach, C.: *jQuery plugin: Tablesorter 2.0 [online]*. Polyester, 2015, [cit. 2015-06-22]. Dostupné z: http://tablesorter.com/docs/

[12] Downie, N.: *Chart.js | Open source HTML5 Charts for your website [online]*. Chart.js, 2015, [cit. 2015-06-22]. Dostupné z: http://www.chartjs.org/

[13] CanvasJS: *Beautiful jQuery Charts; Graphs | CanvasJS [online]*. 2015, [cit. 2015-06-22]. Dostupné z: http://canvasjs.com/jquery-charts/

[14] jqPlot: *jqPlot Charts and Graphs for jQuery [online]*. 2015, [cit. 2015-06-22]. Dostupné z: http://www.jqplot.com/index.php

[15] Highcharts: *Interactive JavaScript charts for your webpage [online]*. 2015, [cit. 2015-06-22]. Dostupné z: http://www.highcharts.com/

[16] Bootstrap: *Bootstrap; The world's most popular mobile-first and responsive front-end framework. [online]*. 2015, [cit. 2015-06-22]. Dostupné z: http://getbootstrap.com/

[17] Google: *Encoded Polyline Algorithm Format. [online]*. 2015, [cit. 2015-06-22]. Dostupné z: https://developers.google.com/maps/documentation/utilities/polylinealgorithm

[18] emmconville: *google-map-polyline-encoding-tool/src at master Â· emcconville/google-map-polyline-encoding-tool Â· GitHub [online]*. 2015, [cit. 2015-06-22]. Dostupné z: https://github.com/emcconville/google-map-polyline-encoding-tool/tree/master/src

Příloha A

Seznam použitých zkratek

API Application Programming Interface

CSS Cascading Style Sheets

GUI Graphical user interface

Java EE Java Enterprise Edition

MySQL My Structured Query Language

PHP Hypertext preprocesor

SQL Structured Query Language

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF