

Vysoká škola ekonomická v Praze
Fakulta informatiky a statistiky



Vývoj webového rozhraní pro aplikaci Pocket Pilot

DIPLOMOVÁ PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Podniková informatika

Autor: Bc. Andrej Souček

Vedoucí práce: Ing. Jarmila Pavlíčková, Ph.D.

Praha, červen 2019

Prohlášení

Prohlašuji, že jsem diplomovou práci *Vývoj webového rozhraní pro aplikaci Pocket Pilot* vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne 26. června 2019

.....

Podpis studenta

Poděkování

Tímto bych chtěl poděkovat své vedoucí Ing. Jarmile Pavlíčkové, Ph.D. za vstřícný přístup a cenné rady při zpracování této diplomové práce.

Abstrakt

Tato diplomová práce se zabývá rozšířením mobilní aplikace Pocket Pilot. Ta je rozšířena o webovou aplikaci, která slouží pilotům pro plánování letů. Naplánované trasy se automaticky synchronizují s mobilním zařízením, ze kterého tak vzniká letecký navigační přístroj.

V první části práce je provedena analýza současných aplikací. Z té vyplývá, že neexistuje podobná neplacená alternativa. Před samotným vývojem je provedena analýza aplikace dle metodiky MMSP, ze které vychází detailní seznam požadavků a případy užití. Na základě provedené analýzy jsou vybrány vhodné technologie pro backendovou i frontendovou část. Ty jsou dále použity při implementaci. V kapitole Implementace jsou popsány zajímavosti a problémy, které bylo nutné během samotného vývoje aplikace řešit. Součástí této kapitoly jsou pro lepší orientaci uvedeny výpisy zdrojového kódu. Práce je zakončena kapitolou Testování, která obsahuje testovací nápady a ukázkové testovací případy dle metodiky MMSP.

Hlavním přínosem práce je rozšíření neplacené aplikace Pocket Pilot o funkci, kterou ostatní podobné aplikace zpoplatňují. Zároveň je v práci popsáno několik moderních přístupů a technologií pro vývoj webových aplikací.

Klíčová slova

PHP, Nette, PostGIS, Webpack, Leaflet

Abstract

This thesis deals with an extension of the Pocket Pilot mobile application. It is extended by a web application which is used by pilots for flight planning. Planned routes are automatically synchronised with the mobile device.

In the first part of the thesis there is an analysis of currently available applications. The result is that there is no similar free alternative. Before the actual implementation the analysis of the application based on the MMSP methodology is done. The analysis produces a detailed list of requirements and uses cases as a result. Based on this result the backend and frontend technologies are chosen and described. These are later used for the actual implementation. Problems which occurred during the implementation are described in the Implementation chapter. They are also documented by extracts of the source code for better orientation. The thesis is ended by the Testing chapter which includes testing ideas and test cases based on the MMSP methodology.

The main benefit of the thesis is that the free Pocket Pilot application is extended by a function which is paid for in other applications. Also the thesis shows the modern approaches to web and web applications development.

Keywords

PHP, Nette, PostGIS, Webpack, Leaflet

Obsah

Úvod	17
1 Analýza současných dostupných aplikací	19
1.1 XCSoar	19
1.2 AirMate	19
1.3 SkyDemon	19
1.4 Fly is Fun	20
1.5 Zhodnocení dostupných aplikací	20
2 Analýza a návrh aplikace	21
2.1 Zadání aplikace	21
2.2 Základní charakteristika potřeb	21
2.3 Detailní specifikace požadavků	21
2.3.1 Volba jazyka	21
2.3.2 Registrace a přihlášení	22
2.3.3 Obnovení hesla	22
2.3.4 Nástěnka	23
2.3.5 Plánování tras	23
2.3.6 Zobrazení návodů	25
2.3.7 Spárování s mobilní aplikací	25
2.3.8 Kontaktování autora	25
2.3.9 Přístup z mobilní aplikace	25
2.4 Analýza	25
2.4.1 Seznam požadavků	25
2.5 Model případů užití	28
2.5.1 Vytvoření nové trasy	29
3 Výběr technologií - backend	31
3.1 PHP	31
3.2 Nette	32
3.2.1 Novinky v Nette 3	32
3.2.2 Latte	32
3.2.3 DI Container	33
3.2.4 Database	33
3.2.5 Forms	34
3.2.6 Mail	34
3.2.7 Tester	34
3.2.8 Tracy	35
3.2.9 Další komponenty	35
3.3 Databáze	36

3.3.1	MySQL	36
3.3.2	PostgreSQL	37
3.3.3	SQLite 3	37
3.3.4	Shrnutí	38
3.4	Synchronizace s mobilní aplikací	38
3.4.1	Autentizace uživatele	39
3.4.2	Získání naplánovaných tratí	40
4	Výběr technologií - frontend	41
4.1	Správce balíčků	41
4.1.1	Yarn vs. NPM	41
4.2	Výběr knihoven	42
4.2.1	Designový framework	42
4.2.2	Knihovna pro práci s mapou	43
4.2.3	Knihovna pro AJAX	44
4.2.4	Knihovna pro generování QR kódů	45
4.3	Podpora prohlížečů	45
4.3.1	Babel	45
4.4	Nástroje pro sestavení	46
4.4.1	GruntJS	46
4.4.2	GulpJS	47
4.4.3	Webpack	47
4.4.4	Shrnutí	48
5	Implementace	49
5.1	Struktura projektu	49
5.2	Databáze	50
5.3	Model	51
5.4	Webpack	53
5.4.1	Konfigurace	53
5.4.2	Webpack a Nette	56
5.4.3	Babel	57
5.5	REST API	58
5.5.1	Autentizace	58
6	Testování	61
6.1	Testovací nápady	61
6.1.1	Registrace a přihlašování	61
6.1.2	Nástěnka	61
6.1.3	Plánování tras	61
6.1.4	Návody	62
6.1.5	Spárovat	62
6.1.6	Kontaktní formulář	62
6.1.7	API	63
6.2	Testovací případy	64

6.2.1	Smazání naplánované trasy	64
6.2.2	Odeslání e-mailu z kontaktního formuláře	65
	Závěr	67
	Seznam použité literatury	69
	A SQL skript pro vytvoření tabulek	73
	B Výpis lokální konfigurace	75

Seznam obrázků

2.1	Návrh přihlašovací obrazovky (vlastní zpracování)	22
2.2	Návrh obrazovky Nástěnka (vlastní zpracování)	23
2.3	Návrh obrazovky Plánování tras - seznam (vlastní zpracování)	24
2.4	Návrh obrazovky Plánování tras - mapa (vlastní zpracování)	24
2.5	Diagram případů užití (vlastní zpracování)	28
3.1	Zobrazení chyby v provedení Tracy (Nette Foundation, 2019e)	35
5.1	Diagram databáze (vlastní zpracování)	50
5.2	UML diagram tříd modelu (vlastní zpracování)	52

Seznam tabulek

2.1	Funkční požadavky (vlastní zpracování)	26
2.2	Nefunkční požadavky (vlastní zpracování)	27
2.3	Systémová omezení (vlastní zpracování)	27
2.4	Bezpečnost (vlastní zpracování)	28
2.5	Specifikace případu užití (vlastní zpracování)	29
2.6	Základní scénář (vlastní zpracování)	29
2.7	Alternativní scénář 1 (vlastní zpracování)	29
2.8	Alternativní scénář 2 (vlastní zpracování)	30
2.9	Alternativní scénář 3 (vlastní zpracování)	30
3.1	Aktuální verze PHP (The PHP Group, 2019)	32
3.2	Srovnání databází (vlastní zpracování)	36
4.1	Srovnání designových frameworků (vlastní zpracování)	43
4.2	Srovnání nástrojů pro sestavení (vlastní zpracování)	48
5.1	Množství stažených dat před optimalizací (Gilbertson, 2018)	54
5.2	Přenesená data po první optimalizaci (Gilbertson, 2018)	55
5.3	Přenesená data po druhé optimalizaci (Gilbertson, 2018)	55
5.4	Přenesená data po kompletní optimalizaci (Gilbertson, 2018)	56
6.1	Specifikace testovacího případu 1 (vlastní zpracování)	64
6.2	Scénář testovacího případu 1 (vlastní zpracování)	64
6.3	Specifikace testovacího případu 2 (vlastní zpracování)	65
6.4	Scénář testovacího případu 2 (vlastní zpracování)	65

Úvod

Vymezení tématu a cíle práce

Tato diplomová práce rozšiřuje mobilní aplikaci Pocket Pilot, která vznikla v rámci mé bakalářské práce.

Smyslem tohoto rozšíření je, aby si pilot na počítači nebo tabletu mohl naplánovat trasu, kterou chce letět. Aplikace spočítá na základě deklarované rychlosti časy jednotlivých otočných bodů, kurzy mezi nimi a další data potřebná pro let. Díky zobrazení rozdělení vzdušného prostoru na mapě bude snadné vyhnout se uzavřeným nebo řízeným prostorům. Takto naplánované trasy se synchronizují s mobilní aplikací, čímž odpadne složité zadávání otočných bodů nebo export a následný import tratě do GPS navigace apod.

V práci popisuji kompletní vývoj webové aplikace od definování požadavků, výběru technologií a frameworků, přes samotnou realizaci implementace backendu a frontendu, až po zprovoznění aplikačního rozhraní pro komunikaci s mobilní aplikací. Při výběru nástrojů je kladen důraz na open source. Samotné jádro aplikace tvoří český framework Nette, který je představen ve vlastní kapitole. Aby mezi sebou aplikace mohly komunikovat, je vytvořeno rozhraní postavené na architektuře REST s výměnou dat ve formátu JSON. Ostatní vybrané nástroje jsou v průběhu práce také popsány a použity při implementaci vlastní aplikace.

Cílem práce je vytvoření webového rozhraní pro navigační přípravu pilotů tak, aby fungovalo jako rozšíření mobilní aplikace Pocket Pilot. Součástí práce je popis požadavků na funkčnost aplikace, výběr vhodných technologií a jejich implementace. Přílohu tvoří samotná webová aplikace splňující předem definovaná kritéria.

Předpoklady a omezení práce

Pro úplné porozumění textu se předpokládá znalost webových technologií, znalost jazyka PHP s povědomím o Nette, základní znalost databází a SQL, HTML a AJAXu.

Za omezení práce by se dalo považovat to, že je použita nejnovější verze frameworku Nette. Není možné tedy v práci odkazovat na jiné práce, které by do hloubky popisovaly problematiku této poslední verze.

Rešerše prací na podobné téma

Do rešerše jsem zařadil ty práce, které popisují samotný framework Nette, nebo jej porovnávají s ostatními frameworky. Vzhledem k tomu, že ve své práci používám poslední verzi Nette, žádná ze současných prací se této verzi nevěnuje. Je proto nutné brát ohled na to, že starší práce se zaobírají zastaralou verzí Nette a nemusí tak zcela odpovídat verzi nové.

Nette framework

Tato bakalářská práce z roku 2012 popisuje framework Nette jako celek. Seznamuje čtenáře se základními vlastnostmi a pojmy, s použitými návrhovými vzory a s dalšími specifiky ve frameworku. Část práce je věnována demonstraci Nette při psaní webové aplikace. (Tölg, 2012) I když se Nette od roku 2012 změnilo, základní principy zůstávají stejné, a tak může být práce nápomocná při prvním seznámení s frameworkem.

Porovnání PHP frameworků pro tvorbu internetové aplikace

Autor v rámci své bakalářské práce porovnává dva frameworky - Nette a Laravel. Obě porovnávané verze jsou ze dne 4. 3. 2016. Jsou zkoumány kvalitativní požadavky na framework, měřeny rychlosti, posuzována bezpečnost a další veličiny. Součástí je měření na základě vytvořené aplikace společně s porovnáním na základě zjištěných informací z veřejných zdrojů. (Haška, 2016) Co se týče výsledků, autor prokázal, že pro tvorbu malé až středně velké aplikace je lepší framework Nette.

Srovnání vývoje webových aplikací v Nette frameworku (PHP) a Node.JS

Tato diplomová práce se zabývá srovnáním platforem pro vývoj webových aplikací, konkrétně frameworku Nette pro platformu PHP a platformy Node.JS. Cílem práce je poskytnout komplexní obraz o rozdílech mezi zkoumanými platformami. Součástí práce jsou ukázkové webové aplikace. (Kočárek, 2013) Autor v závěru práce uvádí, že mu při vývoji v Node.JS často chyběla funkcionality, kterou v Nette našel. I z toho důvodu by pro vývoj webové aplikace sáhl radši po Nette.

1. Analýza současných dostupných aplikací

V této části práce jsou analyzovány některé vybrané aplikace, které lze v současné době použít při létání. Jelikož aplikace Pocket Pilot je neplacená a volně dostupná ke stažení v obchodě Google Play, zabývám se nejdříve aplikacemi, které jsou zdarma a bez nákupů v aplikaci. Na Google Play jsem našel takové aplikace pouze dvě - XCSOar a AirMate.

1.1 XCSOar

Aplikace XCSOar je velmi oblíbená hlavně mezi piloty větroňů. Několikaletý vývoj umožnil vytvořit velmi schopný a spolehlivý nástroj do kokpitu letadla. Na trhu s aplikacemi podobného zaměření byste marně hledali nějakou s lepšími funkcemi, navíc zadarmo. Jelikož ale vývoj započal před více než deseti lety, kdy byla funkce multi-touch u dotykových displejů tehdejších kapesních počítačů budoucností, ovládání není pro nováčka jednoduché, už vůbec ne intuitivní. Proto se novým uživatelům nezamlouvá. Naopak těm, co už se s ní sžili a ovládání jim nedělá problém, se z důvodu bezkonkurenčních funkcí nechce přejít na žádnou jinou. (Souček, 2016) Nyní je možné ji neoficiálně nainstalovat také do některých čteček elektronických knih, což přináší lepší čitelnost na slunci. Tam ale vzniká problém s GPS modulem, který čtečky nemají, proto to nepovažuji za velkou výhodu.

Aplikace nemá žádné další rozhraní, kde je možné pohodlně plánovat trasu.

1.2 AirMate

Nově vzniklá neplacená aplikace, která je zaměřená na plánování VFR letů (letů za vidu). Nabízí mapové podklady celého světa včetně mapek letišť. Umí zaznamenávat uletěnou trasu a zpětně zobrazit její průběh. Dle recenzí na Google Play mají uživatelé problémy s pády aplikace. Instalační balíček má velikost 101 MB a aplikace vyžaduje registraci pro její používání. Plánování letu je nutné dělat v aplikaci, což na mobilu není pohodlné.

1.3 SkyDemon

Skydemon má na Google Play přes 100 000 stažení. Jedná se o mezi piloty populární aplikaci, která je proslulá pro svou spolehlivost. Nabízí všechny důležité funkce pro plánování letů. Kromě mobilního rozhraní nabízí také desktopovou aplikaci pro snažší plánování. Z hlediska

funkčnosti není aplikaci co vytknout. Jediným mínusem je cena - v přepočtu cca 350 Kč měsíčně nebo 4000 Kč ročně.

1.4 Fly is Fun

GPS navigace od českého vývojáře s velmi pokročilými funkcemi. Nabízí netradiční řešení mapového podkladu – uživatel si jej musí importovat sám. Tím pádem vývojář nemá povinnost aktualizovat aplikaci při každé změně ve vzdušných prostorech atd. Také odpadá limit použitelnosti – navigaci může použít uživatel kdekoliv na světě, jen si musí předem stáhnout a importovat potřebná data. Aplikace umožňuje všechno to, co klasická letecká GPS navigace. Je jí podobná i vzhledem a ovládáním. Počítá vzdálenost do cíle, zobrazuje mapu s leteckými prostory, letišti a jinými důležitými body. Dále je možné uložit odletěnou trasu i vést zápisník letů. (Souček, 2016) Aplikaci lze používat 30 dní zdarma, poté je nutné zakoupit roční licenci za 650 Kč. Plánovat lety je ovšem možné pouze přímo v aplikaci.

1.5 Zhodnocení dostupných aplikací

Z předchozích informací vyplývá, že jediná aplikace, která je zdarma, umí plánovat lety a navigovat, je AirMate. Oproti současné verzi aplikace Pocket Pilot ovšem postrádá zápisník letů, logger a správce dokladů. Za nevýhodu dále považuji plánování letů pouze přímo v aplikaci.

2. Analýza a návrh aplikace

Tato část práce popisuje návrh aplikace, který bude použit k samotné implementaci. Analýzy a návrh jsou zpracovány dle metodiky MMSP, která byla vytvořena jako lokalizace a přizpůsobení metodiky OpenUP. Je doplněna o další prvky, které vycházejí z jiných metodik, především agilních. (Buchalceková et al., 2013)

2.1 Zadání aplikace

Úkolem je realizovat webové rozhraní pro navigační přípravu pilotů. To musí komunikovat s mobilní aplikací pro Android - Pocket Pilot.

2.2 Základní charakteristika potřeb

Základní princip fungování aplikace je takový, že se uživatel (pilot) registruje nebo přihlásí přes Facebook účet. Po přihlášení si na mapě definuje trasu, kterou chce letět. Aplikace sama na základě deklarované rychlosti spočítá vzdálenosti a časy mezi otočnými body. Po uložení je trasa dostupná pro synchronizaci s mobilním zařízením.

Kromě této stěžejní funkce musí být v aplikaci zobrazen seznam změn v aplikaci - jak mobilní, tak webové. Další sekci jsou návody a kontakt na autora aplikace.

2.3 Detailní specifikace požadavků

Aplikace musí mít jednoduché a intuitivní ovládání. Zároveň by design webové aplikace měl být blízko designu aplikace Pocket Pilot pro Android, kde je využit material design. Vzhled jednotlivých částí aplikace popisují návrhy rozvržení ve formě drátěných modelů. Jelikož se k webové aplikaci dá přistupovat ze zařízení s různými velikostmi displeje, každý návrh má dvě části - rozvržení pro mobilní telefon a pro počítač.

2.3.1 Volba jazyka

Aplikace bude dvojjazyčná. Výchozím jazykem bude čeština s možností přepnutí na angličtinu. Zvolený jazyk se zohlední v URL adrese při používání aplikace.

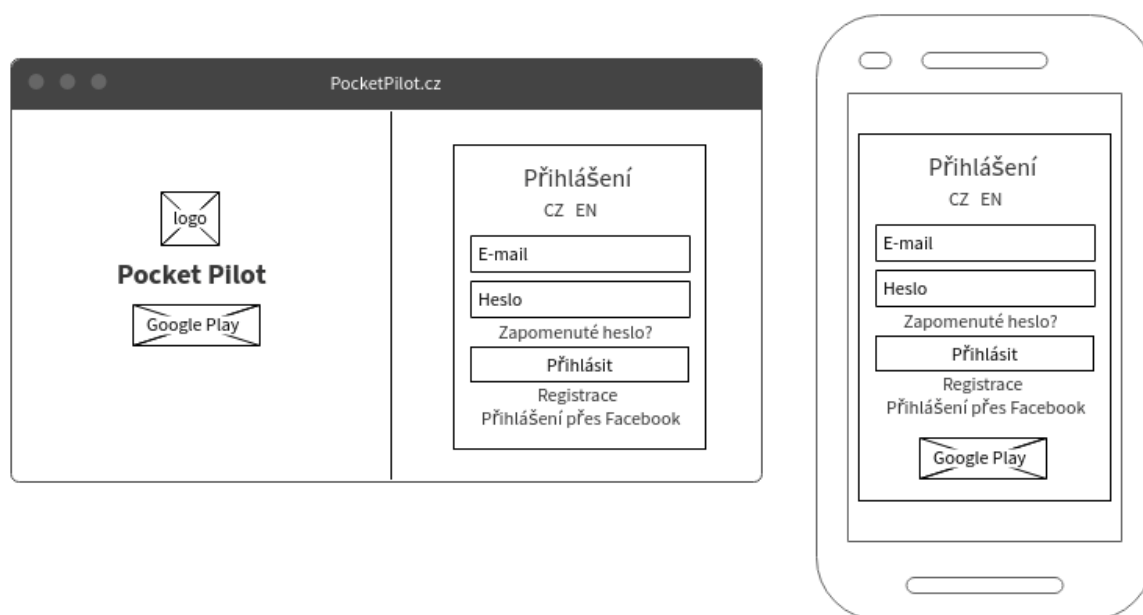
2.3.2 Registrace a přihlášení

Uživatel bude mít dvě možnosti, jak se zaregistrovat do aplikace.

První možnost: Klasická registrace pomocí e-mailu a hesla. Tyto přístupové údaje budou použity pro přihlašování do systému. Heslo bude ukládáno pouze jako hash a bude použita kryptografická sůl.

Druhá možnost: Externí přihlášení pomocí Facebooku. Uživatel si nebude muset pamatovat svoje přihlašovací údaje, při každém přístupu do aplikace využije svůj Facebookový účet.

Kromě přihlášení se na přihlašovací obrazovce zobrazí odkazy pro přepnutí jazyka, resetování hesla a odkaz ke stažení mobilní aplikace Pocket Pilot, který povede do obchodu Google Play.



Obrázek 2.1: Návrh přihlašovací obrazovky (vlastní zpracování)

2.3.3 Obnovení hesla

Obnovení hesla musí být součástí každé aplikace, kde je pro přihlášení heslo vyžadováno. Z hlediska bezpečnosti není možné uživateli poslat jeho heslo. Navíc díky tomu, že je heslo uloženo jen jako hash, není to ani technicky možné. Obnovení hesla tedy využije následující funkčnost:

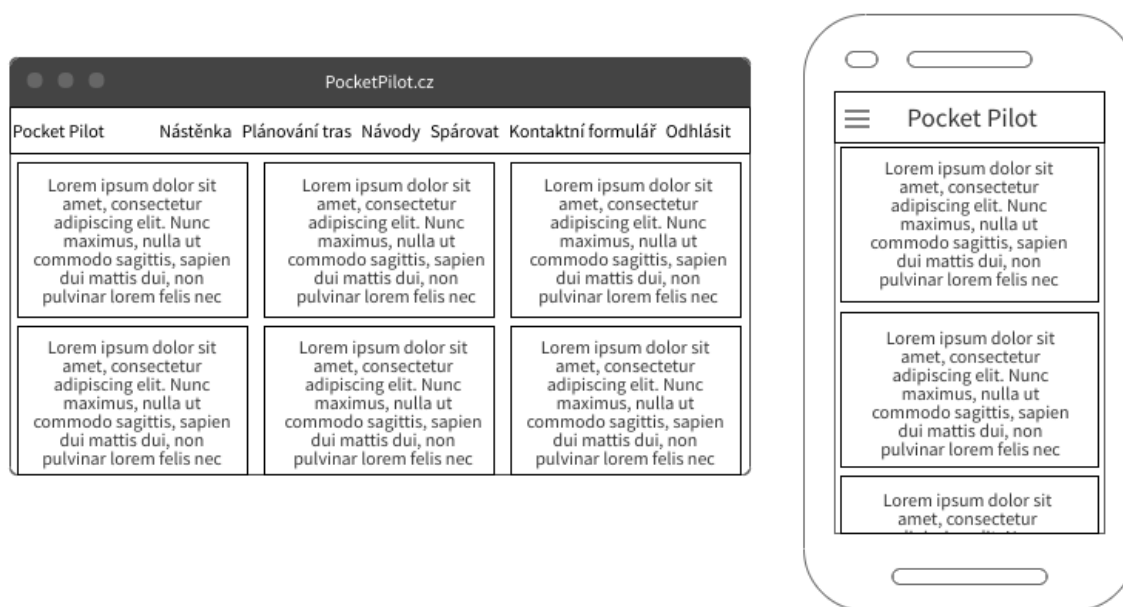
1. Uživatel přejde na stránku pro obnovu hesla.
2. Zadá e-mail, pod kterým je registrován.
3. Server vygeneruje odkaz, jehož součástí bude náhodný řetězec, který se odešle na zadaný e-mail. Tento řetězec bude platný jen po předem určený časový interval.
4. Kliknutím na odkaz e-mailu se uživatel dostane na stránku, kde si v případě, že je odkaz platný, vytvoří nové heslo.

2.3.4 Nástěnka

Po přihlášení se uživateli zobrazí nástěnka. Na ní budou oznámeny případné nové verze mobilní aplikace, novinky a changelog. Tato část nebude nijak interaktivní, poslouží pouze jako informační kanál ve formě nástěnky.

Informace budou zobrazeny na kartách. Počet karet bude záviset na velikosti displeje.

Stejné rozvržení karet využije i sekce „Návody“.

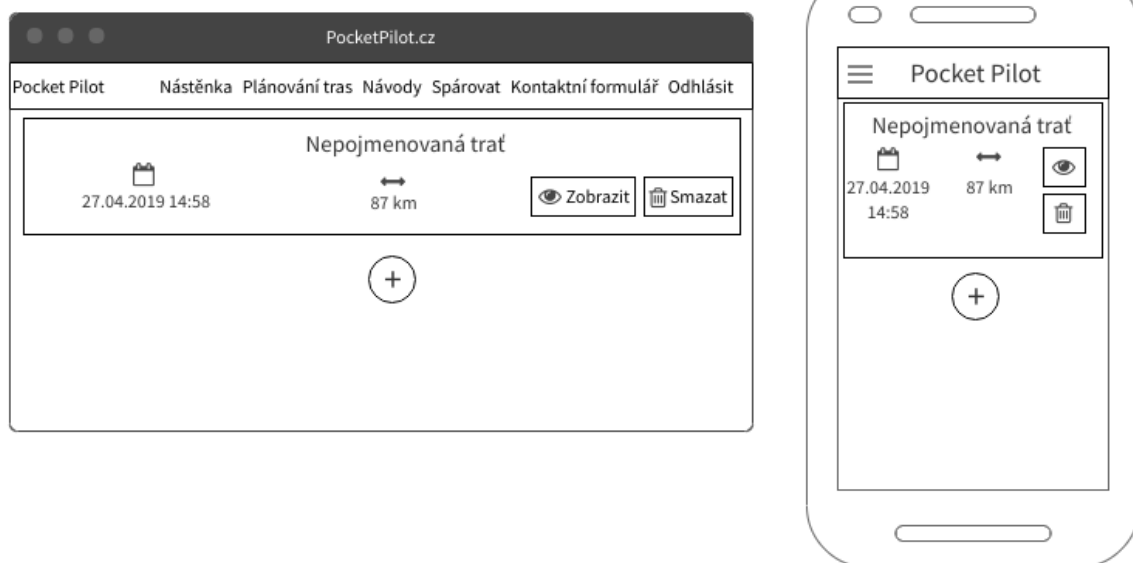


Obrázek 2.2: Návrh obrazovky Nástěnka (vlastní zpracování)

2.3.5 Plánování tras

Plánování tras bude stěžejní funkcí aplikace. Uživateli se zobrazí mapa včetně mapy vzdušných prostorů, na které bude mít možnost zkonstruovat trať pomocí otočných bodů podobně, jako by to dělal s papírovou mapou. Zadáním plánované rychlosti letu se automaticky vypočítá vzdálenost a doba letu mezi jednotlivými body na trati. Naplánovanou trať bude možné pojmenovat a uložit. Uložené tratě bude možné znovu otevřít a upravit, případně smazat. Sekce plánování tras bude rozdělena na dvě části - v první části se vypíše seznam naplánovaných tras s možností přejít do editace, případně vytvoření nové trasy. Kromě názvu tratě se zobrazí i datum vytvoření tratě a její délka.

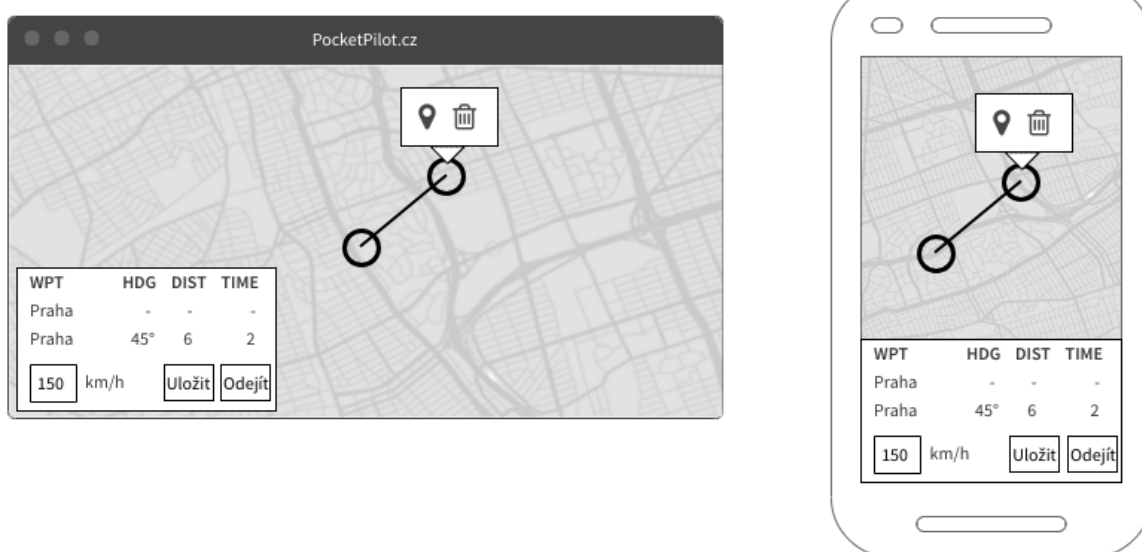
Druhá část bude sloužit k samotnému plánování. Pro využití co největšího prostoru zmizí z horní části navigační lišta a mapa se zobrazí přes celou obrazovku. Vytvářet otočné body, tzn. vkládat jednotlivé body do mapy, půjde přes ovládání u každého bodu. Kliknutím na ikonu značky se nový bod vloží za bod právě zvolený. V případě, že se má bod vložit mezi dva body, vloží se doprostřed. Pokud půjde o prodloužení trasy, vloží se za poslední bod ve stejném směru, jako má poslední rameno trasy.



Obrázek 2.3: Návrh obrazovky Plánování tras - seznam (vlastní zpracování)

V levém dolním rohu bude seznam všech otočných bodů včetně spočítaných časů a vzdáleností ve formě tabulky. Také zde bude možné měnit plánovanou rychlost letu, což způsobí přepočítání tabulky. Tlačítka „Uložit“ a „Odejít“ mluví samy za sebe. Před uložením se aplikace zeptá, jak má trasu pojmenovat. Po uložení bude uživatel přesměrován do předchozí části - na seznam tras.

Mobilní návrh se liší pouze tím, že tabulka s otočnými body se roztáhne na 100 % šířky a maximální výška této tabulky bude polovina výšky displeje.



Obrázek 2.4: Návrh obrazovky Plánování tras - mapa (vlastní zpracování)

2.3.6 Zobrazení návodů

I když je při vývoji mobilní aplikace kladen důraz na jednoduchost a intuitivnost ovládání, existují funkce, které zjednodušit nejdou. Právě proto vznikne podrobný manuál. Jeho součástí bude postup plánování tras a spárování zařízení, popis exportu a importu dat do mobilní aplikace a návod, jak manuálně stáhnout a nahrát mapy do mobilního zařízení.

2.3.7 Spárování s mobilní aplikací

Jak bylo řečeno v úvodu práce, naplánované tratě se synchronizují s mobilním zařízením. Aby to bylo možné, musí se jednotlivé části aplikace spárovat. To bude provedeno pomocí klíče, který bude nutné do mobilní aplikace zadat.

2.3.8 Kontaktování autora

Jednoduchý formulář, pomocí kterého odešle uživatel aplikace zprávu autorovi. Může sloužit buď k nahlášení chyb, nebo k předání poznatků při používání aplikace, případně návrhů na novou funkčnost.

2.3.9 Přístup z mobilní aplikace

Z diagramu případů užití vyplývá, že do aplikace bude přistupovat kromě člověka (uživatele) i stroj (mobilní aplikace). Z toho důvodu bude kromě webového uživatelského rozhraní vytvořeno i rozhraní, které bude data poskytovat v JSON formátu pomocí tzv. REST API. Detailně je návrh a popis rozebrán v kapitole 3.4.

2.4 Analýza

Analýza aplikace zahrnuje definici funkčních i nefunkčních požadavků. Informace pro definování požadavků byly získány díky zpětné vazbě od uživatelů mobilní aplikace a z vlastní zkušenosti autora.

2.4.1 Seznam požadavků

Funkční požadavky

Konkrétní požadavky na funkce jsou sepsány v tabulce 2.1.

Tabulka 2.1: Funkční požadavky (vlastní zpracování)

ID	Název požadavku	Podrobný popis	Priorita 1 nejvyš. 5 nejniž.
F01	Registrace uživatele	Uživatel se může registrovat pomocí e-mailu a zvoleného hesla.	1
F02	Přihlášení pomocí e-mailu a hesla	Po registraci může použít e-mail a heslo pro přihlášení.	1
F03	Přihlášení pomocí Facebook účtu	Uživatel se nemusí registrovat, stačí se přihlásit pomocí Facebook loginu.	2
F04	Obnovení hesla	Pokud uživatel zapomene heslo, je mu poslán na e-mail odkaz s dočasným klíčem, pomocí kterého si může nastavit nové heslo.	1
F05	Zobrazení seznamu změn	V seznamu změn bude uvedeno v bodech, co se změnilo v aplikaci. Změny se budou vztahovat na mobilní i webovou aplikaci.	3
F06	Zobrazení naplánovaných tras	Zobrazení seznamu naplánovaných tras včetně základních informací o trati - název, délka, datum naplánování.	1
F07	Mazání tras	Naplánované trasy bude možné smazat. K tomu poslouží tlačítko u každé trasy v seznamu.	1
F08	Vytvoření nové trasy	Uživatel bude přesměrován na mapu, kde bude moci nakreslit mapu. V tabulce budou zobrazeny informace o trase. Trasu bude možné pojmenovat a uložit.	1
F09	Zobrazení a editace trasy	Uživatel bude přesměrován na mapu, kde bude moci prohlížet a editovat mapu. Editovaná mapa půjde uložit.	1
F10	Návody	Zobrazení návodů pro složitější funkce.	4
F11	Spárování s aplikací Pocket Pilot	Vygenerování API klíče a QR kódu.	1
F12	Kontaktování autora	Možnost odeslat e-mail autorovi. E-mail odesílatele bude nastaven dle přihlášeného uživatele.	5
F13	Volba jazyka	Součástí přihlašovací obrazovky bude možnost přepnutí jazyka. Jazyky budou dva - angličtina a čeština.	4
F14	Synchronizace s aplikací Pocket Pilot	REST API pro synchronizaci tras.	1

Nefunkční požadavky (požadavky na kvalitu)

Nefunkční požadavky popisuje tabulka 2.2.

Tabulka 2.2: Nefunkční požadavky (vlastní zpracování)

ID	Název požadavku	Podrobný popis	Priorita 1 nejvyš. 5 nejniž.
N01	Jazyková lokalizace	Aplikace musí být lokalizována v ČJ a AJ.	4
N02	Intuitivnost ovládání	Ovládání musí být řešeno co nejjednodušeji.	1
N03	Dostupnost	Systém musí být dostupný 24/7. Roční dostupnost min. 99 %.	1
N04	Rozšiřitelnost	Předpokládá se, že aplikaci bude možné rozšířit o další funkce.	1
N05	Design aplikace	Použití material designu. Primární barva aplikace je modrá. Responsivní webdesign.	1

Byznys pravidla

Běžný uživatel může mít uložených maximálně pět naplánovaných tras současně.

Systémová omezení

Tabulka 2.3: Systémová omezení (vlastní zpracování)

ID	Název požadavku	Podrobný popis	Priorita 1 nejvyš. 5 nejniž.
S01	Programovací jazyk	Aplikace bude vyvíjena převážně v PHP a JS.	1
S02	Databázový systém	Viz kapitola 3.3.	1
S03	Podpora prohlížečů	Viz kapitola 5.4.3.	1

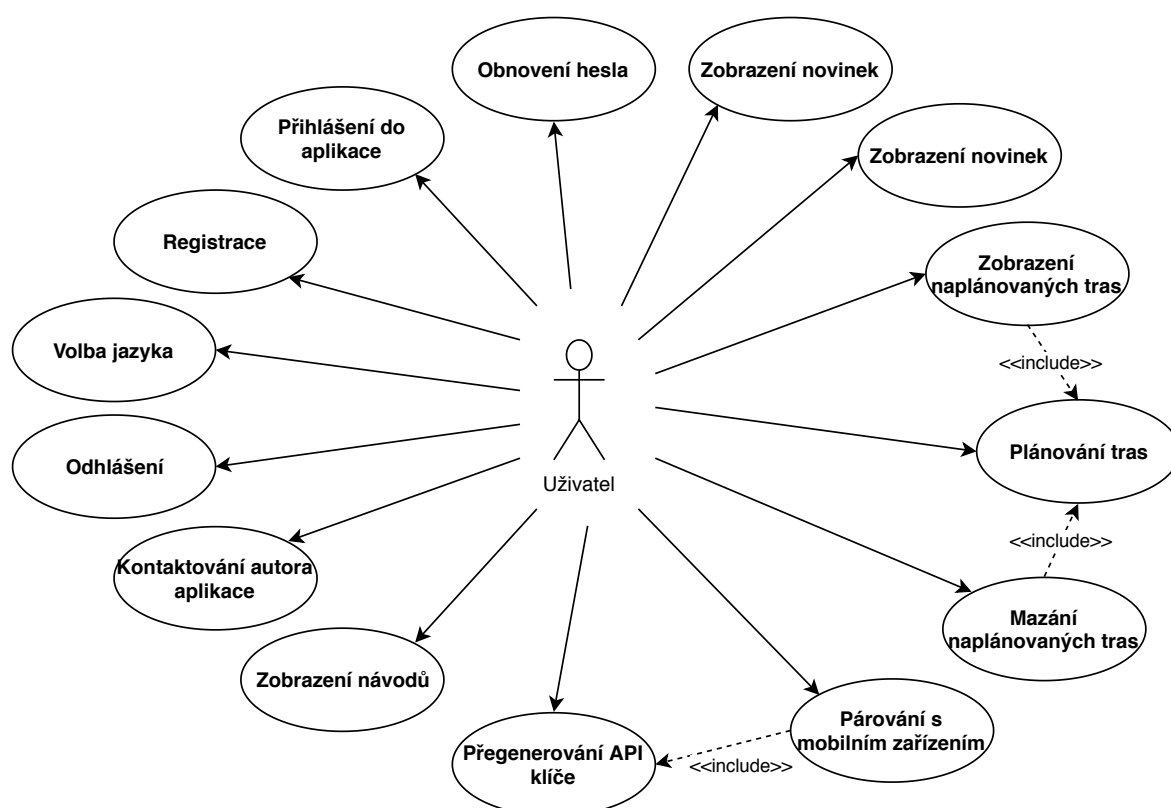
Bezpečnost

Tabulka 2.4: Bezpečnost (vlastní zpracování)

ID	Název požadavku	Podrobný popis	Priorita 1 nejvyš. 5 nejniž.
B01	Šifrované připojení	Komunikace bude probíhat přes HTTPS.	1
B02	Ochrana před XSS	Bude použit šablonovací systém Latte.	1
B02	Ochrana před SQL injection	Bude použita knihovna Nette\Database.	1
B03	Hashování hesel	Pro hashování hesel bude použit algoritmus bcrypt.	1

2.5 Model případů užití

V této kapitole je zachycen komplexní model případů užití. V aplikaci je jediný aktér, kterým je uživatel.



Obrázek 2.5: Diagram případů užití (vlastní zpracování)

2.5.1 Vytvoření nové trasy

Specifikace případu užití

Tabulka 2.5: Specifikace případu užití (vlastní zpracování)

Název	Vytvoření nové trasy
ID funkčního požadavku	F08
Identifikace obrazovek	Obr. 2.3, obr. 2.4
Byznys pravidla	Max. 5 uložených tras
Cíl případu užití	Naplánování trasy letu
Primární aktér	Uživatel
Vstupní podmínky	Přihlášený uživatel v sekci Plánování tras
Výstupní podmínky	Trasa se uloží do systému

Základní scénář

Tabulka 2.6: Základní scénář (vlastní zpracování)

Krok	Role	Akce
1	Aktér	Klepnutím na ikonu „plus“ zahájí plánování.
2	Systém	Přesměruje aktéra na mapu.
3	Systém	Vloží první bod do mapy.
4	Aktér	Přesune bod na požadovaný vstupní bod tratě.
5	Systém	Přepíše název vstupního bodu tratě v tabulce otočných bodů.
6	Aktér	Přidává nové otočné body.
7	Systém	Přepočítává tabulku otočných bodů.
8	Aktér	Uloží trať klepnutím na ikonu diskety.
9	Systém	Zobrazí modální okno pro pojmenování tratě.
10	Aktér	Zadá jméno tratě a uloží ji.
11	Systém	Přesměruje aktéra na seznam naplánovaných tras.

Alternativní scénář - aktér nezadá název tratě

Tabulka 2.7: Alternativní scénář 1 (vlastní zpracování)

Krok	Role	Akce
11a	Systém	Upozorní na chybějící název.
12	Aktér	Doplní název a uloží trať.
13	Systém	Přesměruje aktéra na mapu.

Alternativní scénář - trať nemá žádné otočné body

Tabulka 2.8: Alternativní scénář 2 (vlastní zpracování)

Krok	Role	Akce
11a	Systém	Upozorní na chybějící otočné body.
12	Aktér	Pokračuje dle kroku 6 základního scénáře.

Alternativní scénář - trasu se nepovede uložit

Tabulka 2.9: Alternativní scénář 3 (vlastní zpracování)

Krok	Role	Akce
11a	Systém	Upozorní na nezdařené uložení.
12	Aktér	Pokračuje dle kroku 8 základního scénáře.

3. Výběr technologií - backend

Technologie, které budou použity pro vývoj aplikace lze rozdělit podle klasického členění na část backendovou a frontendovou. Pojem backend vychází z anglických slov „back“ a „end“, což by se dalo přeložit jako „zadní část“. Backend je serverová část webové aplikace, která slouží ke zpracování, uchovávání a čtení dat. Uživateli je backend skrytý. Pro backend jsem se na základě vlastních zkušeností rozhodl použít programovací jazyk PHP a framework Nette.

3.1 PHP

PHP je v současné době nejrozšířenějším programovacím jazykem pro vývoj dynamických webů. (W3Techs, 2019) Počátky jazyka se datují rokem 1995, kdy Rasmus Lerdorf vyvinul jistý skript, který mu umožňoval zjistit, kolik návštěvníků četlo jeho online résumé. Tento čítač vyvolal záplavu e-mailů se žádostmi o tento skript. Lerdorf začal dávat svou sadu nástrojů k dispozici pod názvem Personal Home Page. Od svého založení je PHP bez jakýchkoliv restrikcí ohledně jeho používání, modifikace a redistribuce.

Po letech úprav a nových vydání došlo v roce 1997 ke změně slov tvořících zkratku PHP na Hypertext Preprocessor. V červnu roku 1998, kdy byla vydána verze 3.0, využívalo PHP přes 50 000 uživatelů. O šest měsíců později, na začátku roku 1999, oznámila společnost Netcraft, že odhad uživatelské základny přesahuje 1 000 000. PHP 4 z května 2002 přineslo stovky nových funkcí a Netcraft odhadoval, že je PHP nainstalováno na více než 3,6 miliónech domén. Pátá verze jazyka nesmírně zdokonalila objektově orientovanou architekturu PHP - konstruktory, klonování objektů, abstraktní třídy, rozhraní. (Gilmore, 2007)

Kvůli oblíbenosti verze 5.6.x byla její podpora prodloužena do 31. prosince 2018 a řada webů ji stále využívá. Vydání PHP 6 se plánovalo s nativní podporou Unicode, ale vývoj nedosáhl bodu, kdy by bylo schváleno jeho vydání. Následovala tedy sedmá verze, která je v době psaní této práce verzí poslední.

Verze 7.0 přinesla kromě dvojnásobné rychlosti i mnoho novinek. Největšími lákadly byly podpora typové kontroly pro skalární datové typy v parametrech a definice návratových hodnot funkcí. Ve verzi 7.1 přibyl návratový typ void a definice viditelnosti konstant. Také je možné definovat parametry jako nullable a v jednom catch bloku zachytávat více výjimek. Každá podverze je podporována po dobu tří let od vydání. Behem prvních dvou let jsou opravovány všechny chyby a bezpečnostní díry nahlášené vývojáři, kteří používají jazyk PHP. Třetí rok se podpora vztahuje jen na kritické bezpečnostní nedostatky. Uplynutím tří let je podpora větve nadobro ukončena.

Poslední stabilní verzí je verze 7.3, která byla představena v prosinci 2018. Všechny momentálně podporované verze shrnuje následující tabulka. Pro vývoj aplikace bude použita nejnovější větev, tj. 7.3.

Tabulka 3.1: Aktuální verze PHP (The PHP Group, 2019)

Verze	Datum vydání	Aktivní podpora do	Bezpečnostní podpora do
7.1	01.12.2016	01.12.2018	01.12.2019
7.2	30.11.2017	30.11.2019	30.11.2020
7.3	06.12.2018	06.12.2020	06.12.2021

3.2 Nette

Nette je rodina vyspělých a samostatně použitelných komponent pro PHP. Jedná se o svobodný software nabízený pod licencemi GNU GPL a licenci Nette, která vychází z původní BSD licence. Tyto komponenty dohromady vytváří MVP framework, který byl v roce 2015 vyhodnocený magazínem SitePoint jako 3. nejpopulárnější na světě. (Skvorc, 2015) Nette jako celek klade důraz na best practices, efektivitu a bezpečnost. Za více než 10 let vývoje prošel framework mnoha změnami a vylepšeními. Knihovny prochází neustálým zlepšováním, čímž dosahují výborné stability. Nette využívají známé weby jako csfd.cz, denik.cz, damejidlo.cz a mnoho dalších. V dubnu 2019 vyšla nová verze (3.0), která přinesla nové užitečné funkce. Nejdůležitější z nich jsou zmíněny níže.

3.2.1 Novinky v Nette 3

Na začátku dubna 2019 byla vydána nová verze frameworku, která přináší mnoho novinek. Mezi nejdůležitější změny patří:

- Podpora PHP od verze 7.1.
- Použití skalárních typehintů.
- Použití návratových typehintů.
- Funkce, které vracely **false** při chybě (např. funkce při práci s řetězci), vrací nově **null**.
- Odstranění starých (deprecated) tříd a metod (např. `Nette\Environment`).
- Možnost definovat persistentní parametr na trait, což zaručí přenos mezi všemi presen-tery, které trait používají.
- Dynamické parametry v konfiguraci.
- Bezpečnost - podpora pro CSP, SameSite cookie. Defaultně zapnuto.
- DI - Autowiring funguje i pro pole objektů.

3.2.2 Latte

Šablonovací systém Latte, který Nette využívá, se zaměřuje na jednoduchost zápisu a na bezpečnost aplikace. Syntaxe je úplně stejná jako syntaxe PHP. U všech webových aplikací je

nutné vstupy od uživatele tzv. escapovat, tj. očistit. Escapování je u webových aplikaci jedna z nejdůležitějších věcí z hlediska bezpečnosti. Pokud kodér na escapování někde zapomene, vzniká velká bezpečnostní díra náchylná na cross-site scripting útoky. Tato díra ale může vzniknout i pouhým zvolením špatné escapovací funkce. Latte je navrženo tak, aby escapování řešil samotný šablonovací systém a vývojář se jím nemusel zdržovat. Podívejme se na následující latte šablonu:

```
<p onclick="alert({$movie})">{$movie}</p>
<script>var movie = {$movie};</script>
```

Latte rozeznává, ve které části dokumentu se obsah nachází, a podle toho volí správné escapování. Jak je vidět, v Latte není třeba nic manuálně nastavovat, vše se děje automaticky. Pokud bude proměnná `$movie` obsahovat řetězec `'Amarcord & 8 1/2'`, vygeneruje se následující výstup.

```
<p onclick="alert(&quot;Amarcord &amp; 8 1\2&quot;)">Amarcord &amp; 8 1/2</p>
<script>var movie = "Amarcord & 8 1\2";</script>
```

Jak je možné vidět, uvnitř HTML se použije jiné escapování, než uvnitř JavaScriptu a ještě jiné v atributu `onclick`. (Nette Foundation, 2019b)

3.2.3 DI Container

Dependency injection patří mezi standardy používané v objektově orientovaném programování. Princip je v tom, že třídám odeberáme odpovědnost za získávání objektů (služeb), které potřebují ke své činnosti. Namísto toho jim službu předáme už při vytváření. Nette používá DI kontejner, který je prvotní továrnou na všechny objekty nutné k chodu aplikace. Nette umí tyto kontejnery generovat automaticky, stačí mu napovědět pomocí konfiguračních souborů. Stručným zápisem dostaneme skutečný kód DI kontejneru, který může mít ve větších aplikacích i tisíce řádků, u kterých je ruční údržba téměř nemožná. Nette navíc generuje tento kontejner pouze jednou, kód se zapíše do cache a dále se načítá odsud.

3.2.4 Database

Nette Database Core je základní vrstva pro přístup k databázi. Tvoří obálku nad PDO a poskytuje základní funkcionalitu pokládání dotazů. Podporuje databáze MySQL, PostgreSQL, SQLite3, SQLite2, Oracle, MS SQL, ODBC. Druhou související komponentou, která stojí za zmínku, je Nette Database Explorer, který funguje u databází MySQL, PostgreSQL, SQLite3 a MS SQL. Explorer umožňuje získávat data bez nutnosti psaní SQL dotazů (Nette Foundation, 2019a):

```
$books = $context->table('book');
foreach ($books as $book) {
    echo $book->title;
    echo $book->author_id;
}
```

3.2.5 Forms

Nette Forms slouží k tvorbě webových formulářů. Umí validovat data jak na straně klienta JavaScriptem, tak na straně serveru, čímž se vyhneme psaní dvojí validace a minimalizujeme tak pravděpodobnost vzniku chyb. Opět je kladen důraz na bezpečnost, chrání aplikaci před XSS, CSRF, ověří validitu UTF-8 kódování nebo jestli nejsou položky ve formuláři podvržené. Vše probíhá automaticky bez nutnosti manuálního nastavení.

3.2.6 Mail

Nedílnou součástí webových aplikací je odesílání emailů. Nette poskytuje nástroj Nette Mail, který umí e-mail vytvořit, použít v něm latte šablony, přidat přílohy, vytvářet odkazy a odeslat. Zejména použití zmíněných latte šablon usnadňuje vytváření dobře vypadajících emailů. Příklad vytvoření a odeslání e-mailu pomocí Mail\Message a Mail\SendmailMailer (Nette Foundation, 2019d):

```
use Nette\Mail\Message;
use Nette\Mail\SendmailMailer;

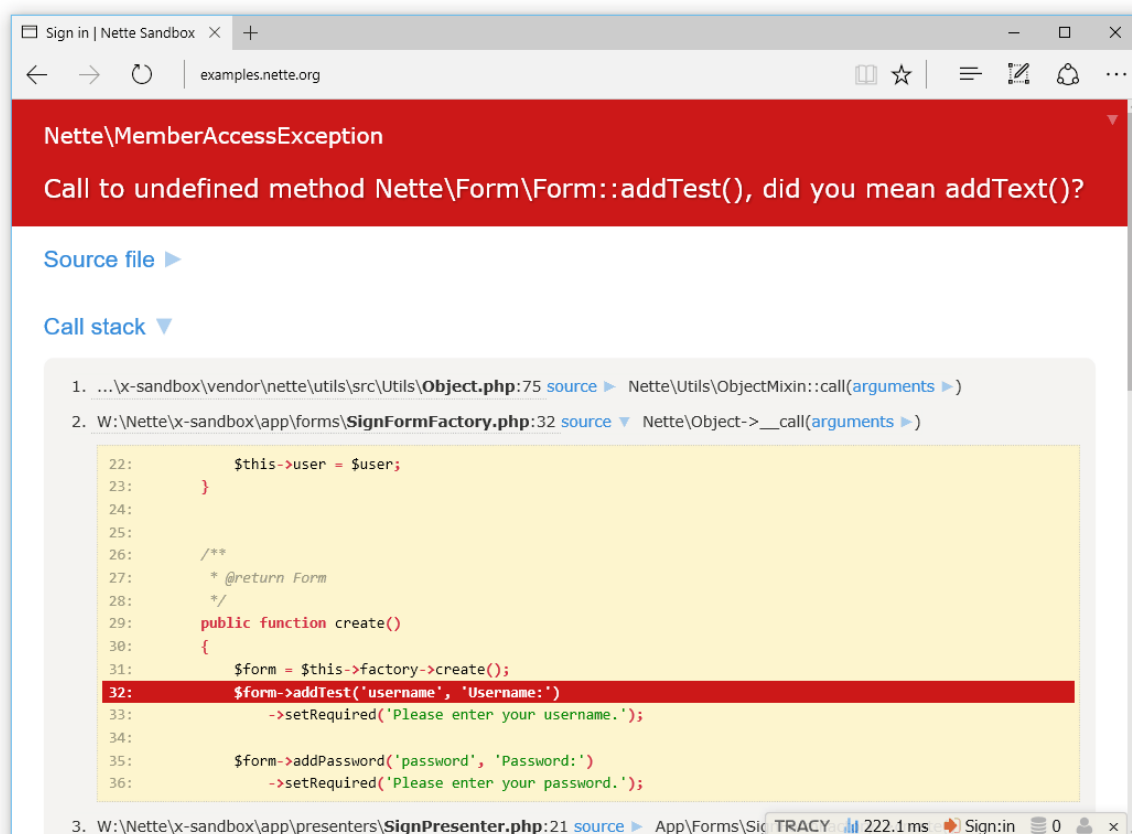
$mailer = new SendmailMailer;
$mail = new Message;
$mail->setFrom('Franta <franta@example.com>')
    ->addTo('petr@example.com')
    ->addTo('jirka@example.com')
    ->setSubject('Potvrzení objednávky')
    ->setBody("Dobrý den,\nvaše objednávka byla přijata.");
$mailer->send($mail);
```

3.2.7 Tester

Nette myslí i na testování kódu. Nette Tester konkuruje nejznámějšímu testovacímu frameworku PHPUnit, oproti kterému umí mnohem více druhů „assertů“, už v základu má možnost pouštět testy paralelně, případně umožňuje testovat HTML kód stylem procházení DOMu. Výhodou je, že testovací metody lze používat přímočaře, není potřeba vytvářet třídu, která dědí z jiné třídy a v ní teprve psát test.

3.2.8 Tracy

Tracy, někdy také laděnka, je nástroj pro ladění kódu. Pomáhá s vizualizací a logováním chyb a vypisováním proměnných. Základním prvkem je tzv. Debug bar. Malý panel na stránce informuje programátora o důležitých informacích - paměť, doba načtení, databázové dotazy, aktuální presenter a další. Umí i napovídat a opravovat chyby a lze rozšířit díky spoustě dostupných pluginů. Tracy je hojně používána i mimo Nette projekty.



Obrázek 3.1: Zobrazení chyby v provedení Tracy (Nette Foundation, 2019e)

3.2.9 Další komponenty

Kromě výše zmíněných komponent používá Nette mnoho dalších zajímavých věcí, jako jsou třeba Utils obsahující trait SmartObject. Ta dělá třídy striktnější, čímž umožňuje upozornit na překlapy. Má i tzv. syntaktická cukrátky, díky kterým není potřeba volat `$object->getFoo()`, ale stačí jen `$object->foo`. Jednoduché `$object->foo = 'val'` potom nahrazuje setter.

Za zmínku stojí i Neon - formát pro serializaci dat. Ve formátu Neon jsou psány konfigurační soubory Nette. Syntaxe vznikla zjednodušením JSONu - odstraněním závorek a uvozovek. Silnou stránkou Nette je i Cache. Framework si sám uloží data pro příští použití. Kromě automatické cache nabízí Nette Cache intuitivní API.

3.3 Databáze

Výběr databáze se řídí dle následujících hlavních kritérií:

- Podpora v Nette Database Explorer,
- licence,
- cena,
- práce s geografickými daty.

Nette Database Explorer, jak již bylo dříve zmíněno, podporuje databáze MySQL, PostgreSQL, SQLite 3 a MS SQL. Následující tabulka zobrazuje, jak tyto databáze splňují hlavní kritéria.

Tabulka 3.2: Srovnání databází (vlastní zpracování)

Databáze	Licence	Cena	Podpora geo. dat
MySQL	GNU GPL	Zdarma	Ano
PostgreSQL	Open Source	Zdarma	PostGIS
SQLite 3	Public domain	Zdarma	Spatialite
MS SQL	MS EULA	Zdarma do 10 GB	Ano

Z tabulky můžeme vyřadit databázi MS SQL, která jako jediná není svobodným softwarem. Využít ji zdarma sice lze do velikosti 10 GB, to však platí pouze pro vývoj. Pokud bychom pro produkční nasazení aplikace chtěli využít nějaký z webhostingů, cena hostingu by se navýšila právě o cenu licence pro MS SQL.

Ze zbylých tří zatím není možné jednoznačně vybrat tu databázi, která se pro projekt nejvíce hodí. Proto je nutné jednotlivé databáze popsat podrobněji a zaměřit se na výhody a nevýhody.

3.3.1 MySQL

MySQL je nejpopulárnější open-source databázový systém. (solid IT gmbh, 2019) Propracovaná dokumentace a velká komunita vývojářů přináší snadnou použitelnost a rychlé řešení případných problémů. MySQL bylo navrženo s důrazem na rychlost a spolehlivost i za cenu toho, že se někdy odchyluje od standardů SQL. (Drake, 2019)

Výhody

- Popularita a komunita – existuje nespočet návodů jak databázi zprovoznit, jak ji spravovat, případně řešit problémy.
- Bezpečnost – MySQL používá pro instalaci skript, který uživatele provede základním nastavením včetně zabezpečení přístupu.

- Rychlost – Tím, že vývojáři neimplementovali vše podle standardů SQL, dosáhli vyšších výkonů a rychlostí.

Nevýhody

- Omezení některých SQL funkcí – jak bylo zmíněno výše, kvůli upřednostnění rychlosti nejsou implementovány nějaké funkce. Patří mezi ně např. FULL JOIN.
- Pomalý vývoj – v roce 2008 došlo k prodeji společnosti Sun Microsystems a o rok později společnosti Oracle Corporation. Od té doby se internetem šíří stížnosti na zpomalení vývoje a neochotu řešit a opravovat chyby.

3.3.2 PostgreSQL

PostgreSQL, známý také jako Postgres, cílí na rozšiřitelnost a plnění standardů SQL. Je vysoce efektivní při multitaskingu. I když není tak masivně používán jako MySQL, existuje mnoho rozšíření třetích stran. (Drake, 2019)

Rozšíření PostGIS přidává podporu pro geografické objekty. To zahrnuje geometrické typy – POINT, LINESTRING, POLYGON, MULTIPOINT, MULTILINESTRING, MULTIPOLYGON, GEOMETRYCOLLECTION, prostorové funkce a indexy. PostGIS používá mnoho známých produktů jako databázový systém, jsou to např. ArcGIS, Mapnik nebo OpenStreetMap.

Výhody

- SQL standardy – dle oficiální dokumentace podporuje PostgreSQL 160 ze 179 prvků standardu SQL:2011 (ISO/IEC 9075:2011).
- Open source – systém je vyvíjen komunitou. Stejně tak komunita spravuje dokumentaci nebo PostgreSQL wiki.
- PostGIS – stabilní, hojně používané rozšíření s velmi aktivní komunitou a podrobnou dokumentací.

Nevýhody

- Paměť – pro každé připojení do databáze vytvoří PostgreSQL vlastní proces. Ten využívá okolo 10 MB paměti.
- Rychlost – není vhodný pro projekty zaměřené čistě na rychlost čtení a zápisu.

3.3.3 SQLite 3

SQLite je malá knihovna napsaná v jazyku C, která ukládá data do souboru. Většina databázových systémů funguje jako serverový proces. U SQLite je to jinak – kdokoli přistupuje k databázi, zapisuje rovnou do souboru na disku. Proto je konfigurace tohoto databázového

systému velmi jednoduchá. Stejně jako konfigurace případných aplikací – vše, co potřebují, je přístup k disku. (Drake, 2019)

Výhody

- Velikost – velikost SQLite se pohybuje ve stovkách kilobajtů.
- Uživatelsky přívětivá – není třeba žádná složitá konfigurace nebo instalace.
- Přenositelnost – všechna data jsou uložena do jednoho souboru. Ten je možné přenášet mezi zařízeními bez složitých importů a exportů.

Nevýhody

- Zámek databáze – k SQLite je možné přistoupit z více procesů v jeden okamžik pouze pro čtení. Změny může dělat vždy jen jeden proces.
- Bezpečnost – výhodná jednoduchá instalace s sebou nese i nevýhodu. SQLite nemá žádnou možnost, jak definovat jednotlivé uživatele databáze a jejich práva.

3.3.4 Shrnutí

Kvůli absenci správy uživatelů a použití zámku na celou databázi jsem dospěl k závěru, že nevyužiji databázi SQLite. Ze zbylých dvou – MySQL a PostgreSQL – jsem se rozhodl pro PostgreSQL s rozšířením PostGIS. Ten je používán a vyvíjen delší dobu a má větší základnu uživatelů. To lze ověřit zadáním klíčových slov do vyhledávání na fóru stackoverflow. Pro dotaz „mysql spatial“ bylo nalezeno 1439 výsledků, pro dotaz „postgis“ jich je 9037 (výsledky k červnu 2019). Pro „mysql geometry“ 719 oproti 1241 pro „postgis geometry“.

Pro vývoj bude použita poslední stabilní verze PostgreSQL, tj. 11.2 ze 14. února 2019 a PostGIS 2.5.

3.4 Synchronizace s mobilní aplikací

Pro synchronizaci s mobilní aplikací Pocket Pilot poslouží technologie REST API. REST je architektura rozhraní, která je použitelná pro snadný a jednotný přístup ke zdrojům. Každý zdroj má svůj vlastní identifikátor (tzv. URI). REST implementuje čtyři základní metody, pomocí kterých lze přistoupit ke zdrojům, a které jsou známe pod označením CRUD, tedy vytvoření dat (Create), získání požadovaných dat (Retrieve), změnu (Update) a smazání (Delete). Tyto metody jsou implementovány pomocí odpovídajících metod HTTP protokolu (Malý, 2009):

- GET (Retrieve) – získání entity. Příklad URI: www.example.com/articles/1.
- POST (Create) – vytvoření entity. Příklad URI: www.example.com/articles. V POST parametrech se posílají data o entitě, která se má vytvořit.

- DELETE (Delete) – smazání entity. Volání je stejné jako u metody GET. Příklad URI: `www.example.com/articles/1`.
- PUT (Update) – aktualizace existující entity. Na konkrétní URI posíláme v parametrech data, která se mají aktualizovat. Příklad URI: `www.example.com/articles/1`.

Jako na každý HTTP požadavek server musí nějak odpovědět. O úspěšnosti provedení požadavku server informuje pomocí standardních stavových kódů.

Z požadavků na funkčnost vychází, že je API potřeba pouze pro získání naplánovaných tratí. Již od začátku je dobré v URI uvádět verzi API. Pokud by někdy bylo třeba API aktualizovat, dá se tak odlišit nová verze od staré, a zároveň mohou fungovat obě verze současně. URI bude mít následující tvar: `/api/v1/<endpoint>`.

3.4.1 Autentizace uživatele

Způsobů jak provést autentizaci je u REST API více. Lze použít jednoduchou HTTP autentizaci (Basic Authentication), kdy se do hlavičky každého requestu přidá přihlašovací jméno a heslo. Pokud je komunikace zabezpečena pomocí HTTPS, nehrozí ukradení hesla při případném odposlouchávání komunikace.

Další možností je použití autentizačního protokolu OAuth. Ten umožňuje sdílení identity z jedné aplikace, aniž by uživatelé museli prozrazovat heslo dalším aplikacím. Na tomto principu funguje přihlašování pomocí účtů ze sociálních sítí. OAuth 2.0 vyžaduje kromě zdrojového serveru ještě autorizační server, který ověřuje identitu a generuje dočasné přístupové klíče, tzv. tokeny. Jelikož do webové aplikace bude možné se přihlásit přes facebookový účet bez použití hesla, musel by Facebook login fungovat i na telefonu a aplikace by si musely složitě vyměňovat několik tokenů, aby se opravdu ověřila identita uživatele.

Jednodušším řešením je využití klasického autentizačního procesu v Nette, který je potřeba jen nepatrně přizpůsobit. V procesu hrají role dva klíče - tajný a dočasný. Jelikož uživatel nemusí mít v databázi uloženo heslo, ověření identity bude probíhat pomocí kombinace e-mailu s vygenerovaným tajným klíčem. Tento tajný klíč se vygeneruje na serveru při registraci uživatele, resp. prvním přihlášení přes Facebook a bude možné jej zobrazit, případně přegenerovat, v části aplikace „Spárovat“. Klient pošle request na API endpoint, který ověří jeho identitu a vystaví mu dočasný klíč. Tento dočasný klíč bude nadále přidáván do všech requestů - tím bude zajištěno „podepsání“ requestů. Proces autentizace přes API tedy bude vypadat následovně:

1. Klient pošle POST request na endpoint `/login`. V parametrech bude e-mail a tajný klíč.
2. Server ověří kombinaci e-mailu a tajného klíče. Pokud je kombinace správná, vygeneruje dočasný klíč a odešle jej klientovi.
3. Klient obdrží dočasný klíč a uloží si jej pro podepsání všech dalších requestů. Jakmile platnost tohoto klíče vyprší, bude nutné provést autentizaci znovu.

3.4.2 Získání naplánovaných tratí

Stěžejním endpointem bude `/tracks`, který poskytne data o naplánovaných tratích. Díky dočasnému klíči v hlavičce requestu bude možné rozpoznat, od kterého uživatele požadavek pochází, a v odpovědi tak poslat jen tratě, které se ho týkají. Jelikož trať bude v databázi uložena jako geometrický útvar, bude nutné ji nejdříve nějak zpracovat. Existuje opět více způsobů, jak výsledný JSON pro response strukturovat.

Jedna z možností je převést otočné body na objekty, jehož atributy by byly GPS souřadnice, a z těchto objektů vytvořit pole.

Druhá možnost je použití tzv. GeoJSON formátu. Jedná se o otevřený formát založený na formátu JSON, který byl navržen pro reprezentaci geografických dat. Jelikož je GeoJSON standardizován (RFC 7946), rozhodl jsem se pro jeho použití.

4. Výběr technologií - frontend

Frontend je opak backendu. Jedná se o část, se kterou se uživatel baví, tedy o uživatelské rozhraní. To musí být jednoduché, intuitivní, mělo by dobře vypadat a designově se blížit vzhledu mobilní aplikace Pocket Pilot. Standardem ve vývoji frontendu je kombinace Javascriptu s CSS. Pro vytvoření dobré webové aplikace ale tyto dvě technologie nestačí. Některé starší prohlížeče totiž například neumí nejnovější JS funkce. Pro CSS to platí podobně, navíc je pro vývojáře jeho zápis zdlouhavý. Vznikly proto nástroje, které hlídají podporu v prohlížečích, kontrolují správnost kódu, usnadňují zápis apod. Další důležitou věcí před implementací frontendu je výběr vhodných knihoven a frameworků.

4.1 Správce balíčků

Správce Javascriptových balíčků usnadňuje jejich instalaci a údržbu. Jedná se o obdobu Composeru pro PHP. Dříve se pro frontendové balíčky používal nástroj zvaný Bower. Tento projekt skončil v roce 2017 – byl totiž duplicitní k NPM (Node Package Manager). NPM byl původně vytvořen pro správu balíčků pro systém Node.js, který slouží pro vytváření backendu. Časem se v NPM začaly objevovat i frontendové balíčky.

4.1.1 Yarn vs. NPM

Při oznámení ukončení Boweru viselo na webu doporučení, aby lidé začali používat nástroj Yarn. Nástroj Yarn vytvořili vývojáři Facebooku. Instalaci balíčků zvládal v době svého představení několikrát rychleji než starší NPM. Yarn také zajišťoval, že se všem vývojářům do složky `node_modules` nainstalují stejné balíčky, což byl někdy u NPM problém. Postupem času ale NPM přišlo s vylepšeními, kterými se na Yarn dotáhl. Verze 5 přinesla velký comeback, ve kterém se NPM rychlostně dotáhlo na Yarn a adresář `node_modules` začal být také konzistentní díky tzv. lock souboru. Oba důvody, proč lidé přešli na Yarn, byly tedy odstraněny a vývojáři se vrátili ke známému NPM, který navíc přinesl příkaz `npm audit`. Ten upozorňuje na nebezpečné verze balíčků, které mají bezpečnostní díry. Tato rizika rozdělí na nízkou, střední a vysokou míru nebezpečí a ve většině případů nabídne řešení. Pro vyřešení často stačí zadat příkaz `npm audit fix`, který vyhledá a nainstaluje opravené verze balíčků. Na Yarn se někteří nahlíželi skrz prsty i z toho důvodu, že odesílá statistická data Facebooku. Odesílání dat Facebooku vidím jako největší zápor nástroje Yarn. Jelikož mám několikaletou zkušenost s nástrojem NPM, rozhodl jsem se pro instalaci balíčků používat NPM.

4.2 Výběr knihoven

Při výběru knihoven je opět potřeba vycházet z požadavků na aplikaci:

1. Material design,
2. interaktivní mapa s vlastním podkladem,
3. AJAX,
4. QR kód.

Ze zmíněných bodů vyplývá, že budou potřeba minimálně čtyři knihovny či frameworky. Při výběru mají přednost open source projekty. Knihovny by měly podporovat moduly z ECMAScript 6 a musí být dostupné pro NPM.

4.2.1 Designový framework

Bootstrap

Nejznámějším designovým frameworkem je v současnosti Bootstrap, který byl v roce 2018 použit na více než 18% všech webů (Burge, 2018) a jejich repozitář na Githubu je na třetím místě v počtu udělených hvězdiček (GitHub, 2019). Jeho největší nevýhodu vidím právě v tom, že je použit tolikrát. Na první pohled lze poznat, že se jedná o Bootstrap a web nepůsobí originálně. Dalším záporem je závislost na knihovně jQuery. Ta byla v posledních letech velmi využívána právě kvůli zmíněné interoperabilitě mezi prohlížeči. V dnešní době lze tento problém řešit jinými způsoby, a tak není důvod zatěžovat web knihovnou navíc. Základní verze Bootstrapu neodpovídá material designu.

Bulma

Bulma je otevřený CSS framework postavený na CSS layoutu zvaném flexbox. Lze využít připravené modifikátory a helpery. Obsahuje mnoho hotových komponent, které se často používají v aplikacích. Nechybí grid systém a podpora ikon z Font Awesome. Tento framework jsem nezvolil z důvodu absence javascriptového kódu ovládajícího některé dynamické komponenty, jako je např. modální okno. V některých případech je to velká výhoda, pro mě by to znamenalo mnoho Javascriptu navíc.

Materialize

Dalším frameworkem je Materialize. Tento open source rámec je založen na CSS gridu, má předdefinovanou paletu barev, širokou škálu helperů a komponent. Narozdíl od frameworku Bulma lze použít i předpřipravený Javascript, který ovládá dynamické komponenty. Vše je

pečlivě zdokumentováno na webu materializecss.com. Velikost CSS souboru, v případě přímého nalinkování do stránky, je pouze 19 kB.

Shrnutí

Všechny uvedené frameworky jsou dostupné pro NPM. Pro aplikaci jsem vybral framework Materialize, který již má předpřipravené javascriptové ovládání dynamických komponent. Ty lze importovat modulově pomocí ES6 modulů.

Tabulka 4.1: Srovnání designových frameworků (vlastní zpracování)

Framework	Material design	Obsahuje JS	Závislost na ext. knihovně
Bootstrap	Ne	Ano	Ano – jQuery
Bulma	Ano	Ne	Ne
Materialize	Ano	Ano	Ne

4.2.2 Knihovna pro práci s mapou

Google Maps

Google mapy poskytují své API od roku 2005. Co se týče služeb, které poskytuje Google Maps API, tak nemají konkurenci. Při vývoji aplikace, kde jsou potřeba data o aktuální dopravě na silnicích, nelze sáhnout po ničem jiném. Menší nedostatky bych viděl v dokumentaci, kde je občas obtížné něco dohledat. Největší negativa tvoří to, že se nejedná o open source, od června 2018 je využití služby zdarma jen za určitých podmínek, a také potřeba vlastnit Google Cloud účet s povolenou měsíční fakturací. Jedna z podmínek pro neplacenou verzi je ta, že nelze použít na placených stránkách. Další podmínka je počet přístupů za den. Vývojář má přednabitý účet na 200 dolarů měsíčně, a pokud tento limit vyčerpá, musí za mapy platit. (Google, 2019)

Leaflet

Leaflet je open source alternativou Google Maps API. Leaflet je postaven na tom, že mapový podklad dodá sám vývojář. Poskytuje pouze základ pro zobrazení mapy a práci s ní (umísťování markerů do mapy, kreslení cest apod.). Ostatní věci, jako například získání názvu města ze souřadnic, hustota dopravy, clustering a další, jsou řešeny formou pluginů a externích služeb. Za zmínku stojí jednoduchá a přehledná dokumentace, malá velikost knihovny, rychlost a spolehlivost. Pro zobrazení své mapy používá tuto knihovnu např. web OpenStreetMap.org.

Shrnutí

Pro vývoj jsem Leaflet vybral, jelikož jde o malou a velmi dobře zdokumentovanou knihovnu, kterou lze rozšířit o pluginy dle vlastních potřeb. Na rozdíl od Google Maps API nehrozí zpoplatnění.

4.2.3 Knihovna pro AJAX

Při výběru knihovny, která by umožnila používat AJAX, je nutné pamatovat na Nette. Jednak musí být na AJAXovou obsluhu formulářů připraven presenter, a jednak je třeba v šabloně vytvořit tzv. snippet. Pro samotné odesílání requestů existuje několik knihoven, které byly vytvořeny přímo pro Nette.

nette.ajax.js

O nette.ajax.js se nedá mluvit jako o frameworku. Jedná se o jednoduchý skript, který zprovozní AJAX na Nette. Odkazům, formulářům a tlačítkům, které mají být obslouženy AJAXově, stačí připsat třídu `ajax`. Bohužel tento skript byl vytvořen již dávno a opět je závislý na jQuery.

Nitro

Nitro je opakem předchozího jednoduchého skriptu. Jedná se o robustní framework, který dokáže vytvořit aplikace kompletně řízené AJAXem. Mezi nejužitečnější funkce bych zařadil validace, práci s historií prohlížeče, dynamické snippety, routing, dialogová okna, flash messages a další. Pro potřeby aplikace, kde je AJAX potřeba jen u pár formulářů, je tato knihovna zbytečně složitá.

Naja

Naja je nástupce nette.ajax.js. V základu se váže na elementy s třídou `ajax` podobně jako nette.ajax.js. Přehledná jednoduchá dokumentace usnadňuje zprovoznění i práci. Vytváří životní cyklus snippetů, na který lze připínat vlastní funkce a vytvářet tak vlastní rozšíření knihovny. Pamatuje i na skripty ve snippetech, které při překreslení pustí znovu. Zjednodušeně by se dalo říct, že se jedná o zmodernizovaný skript nette.ajax.js, který nabízí ještě pár užitečných věcí navíc.

Shrnutí

Z výběru jsem vyloučil skript `nette.ajax.js` kvůli závislosti na jQuery. Pro tento projekt je velmi vhodná knihovna Naja, která přesně vystihuje potřeby projektu, tj. ajaxově obsloužit některé části aplikace, případně na události s obsluhou spojené navázat další funkce.

4.2.4 Knihovna pro generování QR kódů

Pro generování QR kódů jsem vybral knihovnu `qrcode-generator-es6` od vývojáře `rendaw`, která vychází z knihovny `qrcode-generator` od `kazuhikoarase`. Jak už název napovídá, tato knihovna umí generovat QR kód z libovolného textu, navíc podporuje ES6 moduly. Lze nastavit různé parametry, jako je třeba míra opravy chyb ve vygenerovaném QR kódu. Kód lze vygenerovat jako obrázek v HTML tagu `<img>` nebo ve křivkách v tagu `<svg>`.

4.3 Podpora prohlížečů

Několikrát byl v textu zmíněn pojem ECMAScript 6 (někdy zvaný jako ES6, ES2015 či Harmony). Jedná se o standard, na kterém je postaven jazyk Javascript. Standard pochází z roku 2015 a kromě Javascriptu z něj vychází i další jazyky jako ActionScript nebo JScript. Při vydání nového standardu by se měli vývojáři prohlížečů snažit zavést podporu tohoto standardu tak, aby byl prohlížeč schopný s novými funkcemi zacházet. Například poslední Internet Explorer 11 podporuje pouze 11 % funkcí ES6. Oproti tomu poslední verze prohlížečů Edge, Firefox a Chrome podporují ES6 z 98 %, Safari 12.1 dokonce z 99 % (kangax, 2019b). Jak zabezpečit, aby kód psaný v ES6 fungoval i na starších prohlížečích?

4.3.1 Babel

Babel je nástroj, který překompiluje kód psaný v ES6 do staršího standardu ES5, který má téměř 100% podporu napříč prohlížeči (kangax, 2019a). Babel nyní získává širokou podporu i nadcházejícího standardu ESNext (kangax, 2019c). Kód se při využití nových funkcí zavedených v ES6 přetvoří do ES5 kompatibilní podoby.

```
[1, 2, 3].map(n => n ** 2);

const name = "Guy Fieri";
const place = "Flavortown";
`Hello ${name}, ready for ${place}?`;
```

Tento kus kódu se přepíše na:

```
[1, 2, 3].map(function (n) {  
    return Math.pow(n, 2);  
});  
  
var name = "Guy Fieri";  
var place = "Flavortown";  
"Hello ".concat(name, "ready for ").concat(place, "?");
```

Na příkladu je vidět, jak si Babel poradil s tzv. arrow function a template literals. Ale k tomu, aby takhle fungoval, potřebuje mít nainstalované pluginy, zde konkrétně:

`@babel/plugin-transform-arrow-functions`

a `@babel/plugin-transform-template-literals`.

Pokud bychom měli pro každou funkcionalitu, kterou je potřeba překompilovat, instalovat další plugin, mohli bychom rovnou psát kód kompatibilní se standardem ES5. Proto existují tzv. Babel presets. Tyto přednastavené balíky obsahují nejčastěji používané pluginy. Jedním z oficiálních balíků je `@babel/preset-env`, který obsahuje všechny pluginy se základním nastavením pro převod ES6 kódu do ES5. Největší výhodou tohoto balíku je to, že mu lze říct, které prohlížeče mají umět výsledný kód pustit, a na které už se cílit nemá.

4.4 Nástroje pro sestavení

Po výběru balíčkovacího systému, knihoven a představení Babelu přichází na řadu nástroj, který zautomatizuje vše, co je potřeba udělat proto, aby bylo možné umístit do HTML kódu aplikace odkaz na výsledný JS soubor. Není možné nalinkovat přímo soubory z `node_modules`, proto se odsud potřebné knihovny musí vytáhnout a spojit do nového souboru na veřejně přístupném místě. Skripty je dobré zminifikovat, případně rozdělit na několik menších souborů kvůli zrychlení načítání. Celý tento proces sestavení je nutné dělat při každé změně aplikační logiky. Jelikož se jedná o několik po sobě jdoucích činností, které se při každém sestavení opakují, vznikly nástroje na jejich zautomatizování.

4.4.1 GruntJS

GruntJS je nástroj ovladatelný přes příkazový řádek (CLI). Vývojář definuje konfiguračním souborem, jaké činnosti se mají pouštět a v jakém pořadí. Nástroj funguje na principu pluginů, které se starají o jednotlivé kroky procesu - např. puštění testů, spojení JS souborů, překompilování LESS/SASS do CSS, spuštění Babel tasku, zminifikování a obnovení stránky v prohlížeči. Konfigurační soubor je možné psát v CoffeeScriptu, který je stručnější než klasický Javascript.

Výhody

- Velký počet pluginů, které pokrývají široké spektrum úloh,
- možnost psát konfiguraci v CoffeeScriptu.

Nevýhody

- V případě rozsáhlé aplikace může být konfigurace nepřehledná,
- není flexibilní k méně častým úlohám,
- rychle zastarává.

4.4.2 GulpJS

GulpJS byl vydán 18 měsíců po nástroji GruntJS. Jeho účel je stejný, rozdíl je v konfiguraci. Zatímco Grunt používá pro konfiguraci objekty, konfigurace Gulpu je definována funkcemi. Díky použití streamů je rychlejší než Grunt a poskytuje větší kontrolu nad procesem.

Výhody

- Vyšší rychlost jednotlivých úloh,
- jasnější pohled na celkový proces,
- oproti GruntJS je konfigurace stručnější.

Nevýhody

- Streamy mohou být ze začátku nepřehledné.

4.4.3 Webpack

Webpack přináší odlišný přístup. Nejedná se o task runner jako v předchozích dvou případech, ale o tzv. module bundler. Jádrem webpacku je opět doplňováno o pluginy. Kromě nich používá Webpack ještě tzv. loadery, které slouží ke zpracování souborů. Kód projde statistickou analýzou, ze které je vytvořen strom závislostí. S tímto stromem se poté dále pracuje a jsou podle něj vytvořeny výsledné JS soubory. Ty umí logicky rozdělit na menší, v případě HTTP/2 je tedy výrazně zrychleno načítání. Při opakovaném přístupu na stránku pak pracuje i s cachováním skriptů v prohlížeči.

Výhody

- Díky statistické analýze dokáže snadno rozdělovat kód,
- zabalí jen používané soubory,

- v současné době velice populární.

Nevýhody

- Náročnost první konfigurace.

4.4.4 Shrnutí

Nejpopulárnějším z trojice, jak je možné vidět v tabulce 4.2, je Webpack. Umí velice efektivně rozdělit kód a zrychlit načítání aplikace. Z těchto důvodů jsem jej vybral jako nejvhodnější, a to i přes náročnější konfiguraci, kterou popisuji v kapitole Implementace.

Tabulka 4.2: Srovnání nástrojů pro sestavení (vlastní zpracování)

Nástroj	Licence	Počet hvězdiček na GitHubu	Počet stažení za týden (NPM)
GruntJS	MIT	11 945	630 351
GulpJS	MIT	31 251	1 197 674
Webpack	MIT	49 293	7 441 978

5. Implementace

Samotná implementace probíhá za použití vybraných technologií na linuxové distribuci Ubuntu 18.04. Tato kapitola popisuje nejdůležitější a nejzajímavější kroky implementace včetně problémů a způsobů jak je řešit.

5.1 Struktura projektu

Struktura projektu vychází z tzv. Nette Sandboxu (Nette Foundation, 2019c). Jedná se o předpřipravenou a předkonfigurovanou aplikaci, kterou lze využít jako odrazový můstek pro další vývoj. Základní adresářová struktura Nette Sandboxu má následující podobu:

```
nette-sandbox/  
  app/  
  log/  
  temp/  
  vendor/  
  www/
```

Kromě těchto adresářů obsahuje kořenový adresář ještě konfigurační soubory.

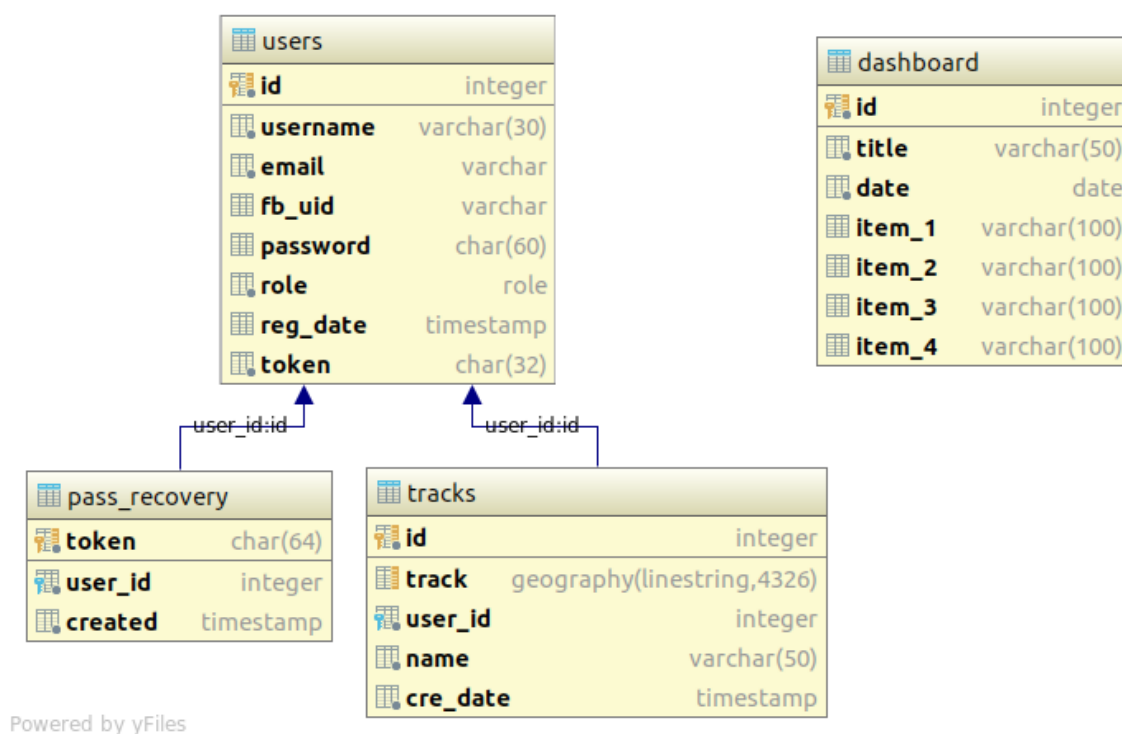
Stěžejním adresářem je adresář **app**, kde je veškerá logika aplikace. Adresář **log** obsahuje logy a informace o případných neodchycených výjimkách. Složka **temp** slouží pro dočasné soubory a cache. Do **vendor** se instalují externí knihovny pomocí nástroje Composer. Posledním adresářem je **www**, který obsahuje soubor **index.php**. Adresář tedy slouží jako brána do aplikace, obsahuje také složky **css** a **js**, odkud se do stránky načítají styly a skripty. Struktura adresáře **app** je také předpřipravená. Já jsem si ji upravit do následující podoby:

```
app/  
  api/  
  assets/  
    css/  
    js/  
  config/  
  i18n/  
  models/  
  public/  
    controls/  
    templates/  
  router/
```

Do adresáře **api** patří presentery obsluhující REST API. V **assets** jsou další dvě podsložky – **css** a **js** obsahující zdrojové soubory Javascriptu a stylů, ze kterých webpack vytvoří pro-

dukční soubory. Pro uložení konfiguračních souborů Nette slouží adresář **config**. Konfigurační soubory jsou dva - jeden obecný pro všechna vývojová prostředí, druhý lokální, pojmenovaný **config.local.neon**. Obecný konfigurační soubor je součástí zdrojového kódu v příloze. Lokální konfigurace je uvedena samostatně v příloze B. Pro překlady a lokalizační soubory je připraven adresář **i18n**. Aplikační logika, tzv. model, patří do složky **models**. Presentery obsluhující uživatelské rozhraní pak do složky **public**. Ta obsahuje ještě dva podadresáře – **controls** pro komponenty a **templates** pro latte šablony. Posledním adresářem je **router**. Tam je základní továrna pro Nette Router.

5.2 Databáze



Obrázek 5.1: Diagram databáze (vlastní zpracování)

Databázi tvoří 4 tabulky. Tabulka **users** uchovává následující data o uživateli:

- **id** – unikátní identifikátor uživatele
- **username** – zvolené uživatelské jméno nebo přezdívka
- **email**
- **fb_uid** – uživatelský identifikátor pro Facebook login
- **password** – heslo
- **role** – admin/premium/user
- **reg_date** – datum registrace uživatele
- **token** – API klíč pro autentizaci přes API

Na tuto tabulku se dále váže tabulka **pass_recovery**, která ukládá data pro přenastavení hesla v případě jeho zapomenutí:

- token – vygenerovaný jednorázový hash pro změnu hesla
- user_id – komu tento hash patří
- created – kdy byl hash vygenerován

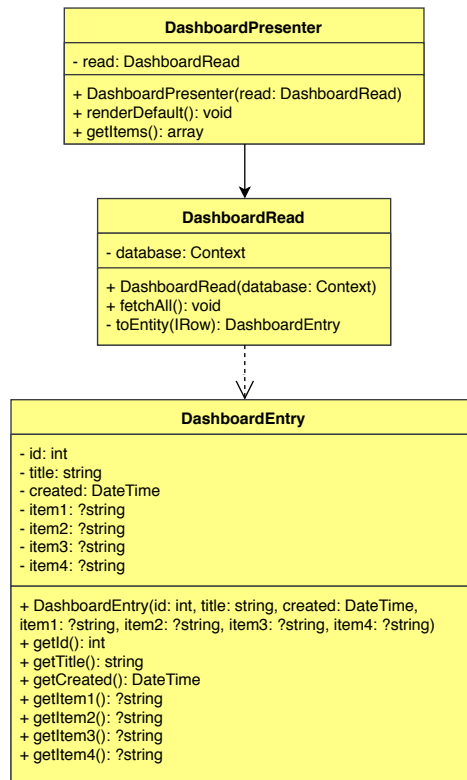
Druhá tabulka navázaná na **users** je tabulka **tracks**, která nese informace o tratích:

- id – unikátní identifikátor tratě
- track – naplánovaná trať
- user_id – komu trať patří
- name – název trati
- cre_date – datum vytvoření trati

Poslední tabulkou je tabulka **dashboard**, ze které se data načítají na nástěnku. Kromě identifikátoru, nadpisu a datumu jsou zde uloženy maximálně čtyři body pro načtení do changelogu. Kromě samotných tabulek je nutné vytvořit kontrolu vyplnění buď uživatelského hesla nebo Facebook ID, dále trigger, který při uložení nového uživatele rovnou vygeneruje token, a tzv. spatial index. Kompletní SQL skript obsahuje příloha A.

5.3 Model

Nette používá architekturu MVP, která rozděluje datový model, uživatelské rozhraní a samotnou logiku předávání dat mezi těmito dvěma vrstvami. Základní princip je oddělení logiky od výstupu. Model by vůbec neměl vědět o tom, jak budou data, která poskytuje, vypsána, případně odkud přišly parametry, se kterými má pracovat. Jeho jedinou funkcí je přijmout parametry a vydat data ven. Často se ale stává, že tento předpoklad není dodržen. Pro ukázkou, jak má takový model vypadat, jsem vybral nejjednodušší část aplikace - nástěnku. UML diagram je zobrazen na obrázku 5.2.



Obrázek 5.2: UML diagram tříd modelu (vlastní zpracování)

Logiku řídí **DashboardPresenter**. Ten díky DI containeru dostane v konstruktoru třídu modelu - **DashboardRead**, která na základě dat z databáze vytvoří instanci třídy **DashboardEntry** a vrátí ji zpět volajícímu, tj. presenteru.

Řetězec událostí vypadá tak, že uživatel načte stránku s nástěnkou, presenter se zeptá modelu na data a ten mu je předá. Díky view a šabloně se potom data vykreslí tak, aby vzhled odpovídal designovým požadavkům na aplikaci. Jak vyplývá z diagramu, je dodržen princip jedné odpovědnosti a zapouzdření. Pro úplné pochopení je nutné přidat ještě kód metody `fetchAll()`. Jedná se o jedinou veřejnou metodu třídy **DashboardRead**, která načte data, přetvoří je do žádané podoby a nakonec je vrátí.

```

public function fetchAll(): array {
    try {
        $rows = $this->database->table(DashboardDatabaseDef::TABLE_NAME)
            ->order('id DESC')
            ->fetchAll();
        $ret = [];
        foreach ($rows as $row) {
            $ret[$row[DashboardDatabaseDef::COLUMN_ID]] = $this->toEntity($row);
        }
        return $ret;
    } catch (\PDOException $e) {
        throw new \RuntimeException($e->getMessage(), $e->getCode(), $e);
    }
}

```

Pokud by mělo dojít k přidání funkčnosti, jako je například přidávání nových položek na nástěnku, vznikla by nová třída v modelu - `DashboardCreate`, která by opět řešila jen převzetí parametrů a jejich vložení do databáze. Stejně tak v případě mazání a editace. Kompletní CRUD model lze vidět v projektu ve složce `app/models/tracks/`.

5.4 Webpack

Webpack byl vybrán jako nástroj pro vytváření balíčků, tzv. bundles. Jak bylo zmíněno dříve, konfigurace je relativně složitá a udělat ji správně znamená zrychlení načítání aplikace a ušetření přenesených dat. Při používání Webpacku pracujeme s následujícími názvy:

- Entry point – začátek bundle procesu, obsahuje moduly, tzn. JS soubory, obrázky, fonty, atd.,
- loaders – slouží pro transformování modulů do výsledných balíčků,
- output bundle – konec bundle procesu, výsledek, který je poté načítán na stránce,
- chunks – output bundle je rozdělen na malé části - chunks. Chunk v překladu znamená poleno, lze si tedy představit output bundle jako kmen a chunks jako jednotlivá polena z něj nařezaná.

5.4.1 Konfigurace

Samotný konfigurační soubor `webpack.config.js` je umístěn v kořenovém adresáři aplikace. Jedná se o Javascriptový objekt, který má předem danou strukturu. Atribut `entry` určuje entry point – vstup do aplikace. Těch může být i více, jak je vidět na části konfiguračního souboru aplikace:

```
entry: {
  main: './app/assets/js/Main.js',
  map: './app/assets/js/Map.js',
  qr: './app/assets/js/QR.js'
}
```

Dalším atributem konfigurace je `output`. Ten určuje, jak se má jmenovat výsledný output bundle a kam se má uložit:

```
output: {
  path: path.join(path.resolve(), 'www/dist'),
  filename: '[name].[contenthash:8].js',
  publicPath: '/dist/'
}
```

Z ukázky je patrné, že se uloží do složky `www/dist` se jménem `[name].[contenthash:8].js`, tedy například `main.768e77df.js`. Hash se do názvu přidává proto, protože se při změně kódu změní i hash. Prohlížeč tak bude donucen stáhnout tento soubor znovu místo načtení z cache. Následuje část `module`, kde se na základě testovacích pravidel nastavují jednotlivé loadery a část `plugins`, kde jsou další pluginy jako například `CleanWebpackPlugin`. Ten má za úkol smazat soubory z output složky při jejich přegenerování.

Poslední sekci je `optimization`. Ta slouží k nastavení rozdělení výstupu do chunků. Důvod, proč se výstup krájí na jednotlivé části, je jednoduchý. Bez chunkování by při každé sebemenší změně musel prohlížeč stahovat celý soubor znovu. Kdyby byl takový výstup rozdělen na dvě části, stáhne prohlížeč jen tu změněnou a druhou načte z cache.

Následující příklad (Gilbertson, 2018) demonsturuje, jak je možné nastavit dělení balíčků:

- Uživatel navštíví stránku jednou za týden po dobu 10 týdnů,
- jednou za týden vývojář stránku aktualizuje,
- každý týden vývojář upravuje „seznam produktů“,
- existuje stránka „detail produktu“, ale na té se teď nepracuje,
- pátý týden přibude do stránky nová knihovna (přes NPM),
- osmý týden je jedna z knihoven aktualizována,
- celková velikost JS souboru, který se načítá do stránky je 400 kB, jeho název je `main.[hash].js`.

Množství stažených dat týdně před optimalizací popisuje následující tabulka.

Tabulka 5.1: Množství stažených dat před optimalizací (Gilbertson, 2018)

Soubor	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
main.[hash].js	400	400	400	400	420	420	420	420	420	420
Celkem v kB	400	400	400	400	420	420	420	420	420	420

V součtu za deset týdnů se stáhne 4,12 MB dat, při každé návštěvě 400, respektive 420 kB. Jedna z možností, kde ušetřit data, je oddělení knihoven do vlastního kódu. Toho lze dosáhnout následující konfigurací sekce `optimization`:

```
optimization: {
  splitChunks: {
    chunks: 'all',
  },
}
```

Taková konfigurace říká Webpacku, aby ze všeho, co potřebuje z `node_modules`, vytvořil samostatný bundle `vendors~main.js`. Množství přesunutých dat je opět shrnuto v následující tabulce.

Tabulka 5.2: Přenesená data po první optimalizaci (Gilbertson, 2018)

Soubor	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
main.[hash].js	200	200	200	200	200	200	200	200	200	200
vendors~main.[hash].js	200				220			220		
Celkem v kB	400	200	200	200	420	200	200	420	200	200

V součtu je za deset týdnů potřeba přenést 2,64 MB dat, tzn. ušetří se 36 %. To už je jistý pokrok, nicméně aktualizace či přidání knihovny opět zapříčiní stažení celého skriptu s knihovnami. Následující konfigurací lze nastavit zabalení každé knihovny do samostatného balíčku:

```
optimization: {
  runtimeChunk: true,
  splitChunks: {
    chunks: 'all',
    maxInitialRequests: Infinity,
    minSize: 0,
    cacheGroups: {
      vendor: {
        test: /[\\/]node_modules[\\/]/,
        name(module) {
          const packageName = module.context
            .match(/[\\/]node_modules[\\/] (.*) ([\\/]|$)/)[1] // název knihovny
          return `npm.${packageName.replace('@', '')}`
        }
      }
    }
  }
}
```

Následující tabulka shrnuje množství stažených dat po rozdělení knihoven do samostatných souborů.

Tabulka 5.3: Přenesená data po druhé optimalizaci (Gilbertson, 2018)

Soubor	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
main.[hash].js	200	200	200	200	200	200	200	200	200	200
npm-1.[hash].js	50									
npm-2.[hash].js	50							20		
npm-3.[hash].js	50									
npm-4.[hash].js	50									
npm-5.[hash].js					20					
Celkem v kB	400	200	200	200	220	200	200	220	200	200

2.24 MB dat za deset týdnů. To znamená téměř 46% úsporu. V úvodu jsem ukázal, že je možné definovat několik entry pointů. Tabulka 5.4 popisuje množství dat při rozdělení vlastního kódu na části rozdělené logicky dle zadání. Je zde navíc soubor Icon.js, který obsahuje SVG ikony o velikosti 25 kB, které byly dosud součástí main.js souboru. Tyto ikony se dosud neměnily.

Tabulka 5.4: Přenesená data po kompletní optimalizaci (Gilbertson, 2018)

Soubor	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
main.[hash].js	125	125	125	125	125	125	125	125	125	125
ProductPage.[hash].js	25									
ProductDetail.[hash].js	25	25	25	25	25	25	25	25	25	25
Icon.[hash].js	25									
npm-1.[hash].js	50									
npm-2.[hash].js	50							20		
npm-3.[hash].js	50									
npm-4.[hash].js	50									
npm-5.[hash].js					20					
Celkem v kB	400	150	150	150	170	150	150	170	150	150

V součtu za deset týdnů se při této konfiguraci stáhne 1,79 MB dat. To znamená snížení o 56,5 %. Předpokladem pro použití tohoto rozdělení je, že aplikace poběží na webovém serveru s podporou protokolu HTTP/2, který podporuje stahování více zdrojů najednou.

5.4.2 Webpack a Nette

Webpack je nyní nastaven, aby generoval jednotlivé soubory dle entry pointů a použitých knihoven. Nyní přichází na řadu problém s přidáním odkazů na tyto soubory do `<script>` tagů. Odkazy není možné staticky přidat do kódu, protože se při každé změně generuje nový hash. Je tedy potřeba zjišťovat vygenerované názvy souborů. K tomu jsem využil plugin Webpack Assets Manifest. Ten vygeneruje JSON, který mapuje vygenerované názvy na původní název entry pointu. Jak tato mapa vypadá, je možné vidět na následující ukázce.

```
...
"qr": {
  "js": [
    "runtime-qr.b29588ff.js",
    "npm.qrcode-generator-es6.567e3e7d.js",
    "qr.ce8fe57a.js"
  ]
}
...
```

Z ukázky je patrné, že Javascripty obsluhující generování QR kódu se vygenerovaly do tří

souborů.

Dalším krokem je vytvoření Nette komponenty, která tento soubor přečte a na jeho základě vytvoří cesty k jednotlivým souborům. Aby bylo možné generovat `<script>` tagy podle názvu entry pointu, má metoda `render()` v komponentě parametr `$entry`, kterým lze určit, co má komponenta vykreslit:

```
public function render(string $entry): void {  
    $this->template->paths = $this->resolvePaths($entry);  
    $this->template->render(__DIR__ . '/webpackControl.latte');  
}
```

Šablona komponenty projde pole s těmito cestami a vytvoří odkazy:

```
{foreach $paths as $path}  
    <script n:nonce src="{ $path }"></script>  
{/foreach}
```

Samotné použití komponenty pak může vypadat takto:

```
{control webpack 'qr'}
```

Tímto zápisem se vygeneruje následující HTML:

```
<script src="/dist/runtime-qr.b29588ff.js" nonce></script>  
<script src="/dist/npm.qrcode-generator-es6.567e3e7d.js" nonce></script>  
<script src="/dist/qr.ce8fe57a.js" nonce></script>
```

5.4.3 Babel

Babel byl v kapitole 4 zmíněn jako nástroj, který při použití balíku `@babel/preset-env` přetvoří kód do podoby, se kterou si poradí starší prohlížeče, resp. prohlížeče, které jsou nakonfigurovány jako podporované. Populární a hodně používaná konfigurace podporovaných prohlížečů je `last 2 versions`, která znamená, že budou podporovány poslední dvě verze všech prohlížečů. Mezi ně patří mimo jiné i prohlížeče, které mají i 0% podíl na trhu. Celkově tato konfigurace pokryje kompatibilitu prohlížečů používaných 86.62 % uživatelů. (Rohan, 2018a) Proč ale není dobré ji používat, pokud opravdu necílíme na staré prohlížeče? Například Internet Explorer 11 byl nahrazen prohlížečem Edge a jeho vývoj skončil. Použití direktivy `last 2 versions` by tedy znamenalo, že kód bude kompatibilní s IE 11 navždy, i kdyby jej nikdo nepoužíval. Kód se zbytečně prodlužuje a zvětšuje se jeho velikost. Lepší, než omezovat podporu prohlížečů podle verzí, je použít míru podílu na trhu. Mnou zvolená

konfigurace vypadá takto: `>0.25%, not ie 11, not op_mini all`. Ta pokrývá 88,59 % uživatelů (Rohan, 2018b) a říká, že kód bude kompatibilní s každým prohlížečem, který používá více než 0,25 % uživatelů kromě Internet Exploreru 11 a Opery Mini. Internet Explorer je dnes zastaralý prohlížeč, jehož podporou by se kód nafukoval. Opera Mini je mobilní prohlížeč, který dostává obsah přes proxy server. Tento server nejdříve zpracuje a zkomprimuje obsah, který odesílá uživateli. Kompresní poměr okolo 90 % výrazně šetří mobilní data. Nicméně se často stává, že interaktivní stránky a aplikace nefungují správně kvůli zpracování obsahu na serveru.

5.5 REST API

Vytvoření REST API v Nette zatím není nativně podporováno a chybí nějaký presenter, který by implementaci API usnadňoval. Vzniklo ale několik komunitních rozšíření, které lze při vývoji využít. Jelikož ale Nette 3.0 vyšlo teprve nedávno, nenašel jsem žádné, které by bylo na tuto verzi připraveno.

Pro směřování API requestů jsem tedy zvolil rozšíření `ublaboo/api-router` (Ublaboo, 2019), které jsem upravil tak, aby fungovalo s Nette 3 a PHP 7.3. Jednalo se především o povýšení Nette knihoven a úpravu části kódu v závislosti na změně rozhraní `Nette\Routing\Router`. Ublaboo router mapuje HTTP metody requestů na metody action v presenterech:

- GET – `actionRead()`,
- POST – `actionCreate()`,
- DELETE – `actionDelete()`,
- PUT – `actionUpdate()`.

Vše se zprovozní pouze přidáním nových cest v `RouterFactory`. Každý endpoint musí mít svou cestu:

```
$router->add(new ApiRoute('/api/v1/login', 'APIv1:Login'));  
$router->add(new ApiRoute('/api/v1/tracks', 'APIv1:Tracks'));
```

5.5.1 Autentizace

Jelikož přihlašovací údaje (e-mail a API klíč) odešle klient na server pomocí POST metody, `LoginPresenter` tedy bude implementovat metodu `actionCreate()`. V této metodě se provede autentizace uživatele ověřením přijatých parametrů `email` a `key`.

Pokud autentizace proběhne v pořádku, server odešle na klienta session ID se stavovým kódem 200. Toto session ID bude nadále klient využívat při každém dalším requestu, aby bylo možné ověřovat jeho identitu. Jedná se tedy o úplně stejný proces přihlášení, jako když se uživatel přihlašuje pomocí klasického formuláře na webu.

V případě, že kombinace e-mailu a klíče není správná, server vrátí odpověď se stavovým kódem

401 – Unauthorized. Kompletní podobu metody `actionCreate()` ve třídě `LoginPresenter` je možné vidět na následujícím výpisu.

```
public function actionCreate(): void {
    try {
        $email = $this->getRequest()->getPost('email');
        $key = $this->getRequest()->getPost('key');
        $this->getUser()->login(new TokenCredentials($email, $key));
    } catch (AuthenticationException $e) {
        $this->getHttpResponse()->setCode(403);
        $this->sendResponse(new JsonResponse(['error' => 'Incorrect e-mail or key.']));
    }
    if ($this->getSession()->isStarted()) {
        $this->getSession()->close();
        $this->getSession()->destroy();
    }
    $this->getSession()->setExpiration('4 hours');
    $this->getHttpResponse()->setCode(200);
    $this->sendResponse(new JsonResponse(
        ['session' => $this->getSession()->getId()]
    ));
}
```

Pro ověření identity v dalších requestech poslouží překrytá metoda `checkRequirements()`, která je volána frameworkem na začátku životního cyklu presenteru. Tím je zaručeno ověření identity na samém počátku zpracování requestu a lze v ní využít výhod, které přináší třída `Nette\Security\User`. Ta je k dispozici právě díky využití klasického autentizačního procesu Nette. V případě, že je s requestem posláno špatné session ID, klientovi se vrátí odpověď s kódem 403 – Forbidden. Finální podobu metody `checkRequirements()` ve třídě `TracksPresenter` zobrazuje následující výpis.

```
public function checkRequirements($element): void {
    parent::checkRequirements($this->getReflection());
    if (!$this->user->isLoggedIn()) {
        $this->getHttpResponse()->setCode(403);
        $this->sendResponse(new JsonResponse(['error' => 'User not authenticated']));
    }
}
```

Jak je vidět na obou předchozích výpisech, kromě samotného chybového kódu je odeslána i informace o tom, proč k chybě došlo. To je důležité pro práci s API při vývoji klienta.

6. Testování

V této kapitole uvádím testovací nápady a příklady testovacích případů dle metodiky MMSP.

6.1 Testovací nápady

Testovací nápady vychází z případů užití a seznamu požadavků.

6.1.1 Registrace a přihlašování

- Uživatelé se mohou registrovat.
- Uživatelé se mohou přihlásit pomocí Facebook účtu bez předchozí registrace.
- Uživatelé se mohou přihlásit pomocí hesla zvoleného při registraci.
- Uživatel je upozorněn na neplatné přihlašovací údaje v případě chybného zadání. Nesmí být specifikováno, který údaj byl špatně zadán.
- Při zapomenutí hesla lze heslo obnovit.
- Token pro obnovení hesla má omezenou platnost.
- Přihlašovací obrazovka se přizpůsobuje velikosti displeje.
- Po přihlášení se uživatel může odhlásit klepnutím na tlačítko Odhlásit v pravém horním rohu.

6.1.2 Nástěnka

- Po přihlášení do aplikace se zobrazí nástěnka se seznamem změn.
- Nástěnka se přizpůsobuje velikosti displeje.

6.1.3 Plánování tras

- Sekce Plánování tras zobrazuje seznam naplánovaných tras.
- U jednotlivých tras je uveden název, datum vytvoření a délka trasy.
- Trasu lze smazat.
- Trasu lze zobrazit.
- Trasu lze upravovat.
- Je zobrazeno tlačítko „plus“, které slouží pro plánování nové trasy.
- Není zobrazeno tlačítko „plus“, pokud je uloženo 5 tras.
- Otočné body jsou správně pojmenovány.
- Při plánování trasy se správně přepočítávají vzdálenosti a časy mezi otočnými body.
- Při změně rychlosti letu se správně přepočítají časy mezi otočnými body.
- Nově naplánovanou trasu lze uložit.

- Upravenou trasu lze uložit.
- Nelze uložit nepojmenovanou trasu.
- Nelze uložit trasu bez otočných bodů.
- Podkladová mapa se zobrazuje správně.
- Mapa vzdušných prostorů se zobrazuje správně.
- Při zobrazení naplánované trasy je mapa vycentrována tak, aby byla trasa uprostřed displeje.
- Nové otočné body lze vkládat za předchozí.
- Při vložení nového otočného bodu za poslední otočný bod je nový otočný bod vložen ve správné vzdálenosti a pod správným úhlem.
- Nové otočné body lze vkládat mezi existující otočné body.
- Při vložení nového otočného bodu mezi dva existující body je nový otočný bod vložen přesně do poloviny vzdálenosti mezi nimi.
- Při vložení nového otočného bodu se zobrazí nástroje pro další interakci.
- Při kliknutí na jakýkoliv otočný bod se zobrazí nástroje pro další interakci.
- Otočné body lze mazat.
- Při smazání otočného bodu mezi jinými dvěma body je trasa překreslena.
- Vstupní bod tratě nelze smazat.
- Ikona pro uzavření trasy je zobrazena až při vložení druhého otočného bodu.
- Trasa není změněná při opuštění plánování bez uložení.
- Plánování tras se přizpůsobuje velikosti displeje.

6.1.4 Návody

- Sekce Návody zobrazuje návody.
- Návody se přizpůsobují velikosti displeje.

6.1.5 Spárovat

- V sekci Spárovat je zobrazen e-mail a tajný klíč uživatele.
- Klíč lze přegenerovat.
- Generuje se QR kód z uvedených údajů.
- Sekce Spárovat se přizpůsobuje velikosti displeje.

6.1.6 Kontaktní formulář

- V sekci Kontaktní formulář je zobrazena textarea s tlačítkem pro odeslání.
- E-mail se odesílá.
- Odeslaný e-mail má správně uvedeného odesílatele.
- Sekce Kontaktní formulář se přizpůsobuje velikosti displeje.

6.1.7 API

- Funguje endpoint /login s aktuálním e-mailem a klíčem.
- Funguje endpoint /tracks s aktuálním tokenem (session ID).
- Po přegenerování klíče v sekci „Spárovat“ nefunguje starý klíč.
- Bez platného tokenu (session ID) nelze získat data.

6.2 Testovací případy

6.2.1 Smazání naplánované trasy

Tabulka 6.1: Specifikace testovacího případu 1 (vlastní zpracování)

Název	Smazání naplánované trasy
ID funkčního požadavku	F12
Identifikace obrazovek	Obr. 2.3
Popis	Smazání naplánované trasy
Vstupní podmínky	Uživatel je přihlášen do aplikace Ve výpisu tras je alespoň jedna uložená trasa
Poznámky	

Tabulka 6.2: Scénář testovacího případu 1 (vlastní zpracování)

Krok	Akce testera	Reakce systému	Výsledek	Pozn.
1	Klikne na položku „Plánování tras“ v menu aplikace.	Zobrazí sekci Plánování tras.		
2	Klikne na tlačítko s ikonou popelnice.	Zobrazí upozornění s dotazem, jestli trasu určitě smazat.		
3	Potvrdí smazání trasy	Vymaže trasu a vypíše zprávu o úspěšném smazání.		
Celkový výsledek				

6.2.2 Odeslání e-mailu z kontaktního formuláře

Tabulka 6.3: Specifikace testovacího případu 2 (vlastní zpracování)

Název	Odeslání e-mailu z kontaktního formuláře
ID funkčního požadavku	F12
Identifikace obrazovek	—
Popis	Odeslání e-mailu autorovi aplikace z kontaktního formuláře
Vstupní podmínky	Uživatel je přihlášen do aplikace
Poznámky	

Tabulka 6.4: Scénář testovacího případu 2 (vlastní zpracování)

Krok	Akce testera	Reakce systému	Výsledek	Pozn.
1	Klikne na položku „Kontaktní formulář“ v menu aplikace.	Zobrazí sekci Kontaktní formulář.		
3	Zkontroluje, že je uvedena správná e-mailová adresa v závorce nad formulářem.			
3	Do pole „Vaše zpráva“ napíše libovolný text.			
4	Klikne na tlačítko „Odeslat“.	Odešle e-mail a vypíše zprávu o úspěšném odeslání.		
5	Zkontroluje přijetí e-mailu.			
Celkový výsledek				

Závěr

Cílem práce bylo rozšířit mobilní aplikaci Pocket Pilot o webovou aplikaci pro plánování letů. Jak vyplývá z analýzy dosavadních aplikací, na trhu v současné době není dostupná žádná neplacená aplikace, která by umožňovala naplánovat trasu na počítači a následně ji zobrazit v mobilním zařízení.

Pro dosažení cíle musel být nejdříve vytvořen kompletní návrh aplikace s analýzou a definováním požadavků dle metodiky MMSP. Podle této analýzy byly vybrány nejvhodnější nástroje pro vývoj. Vybrané nástroje byly charakterizovány a použity při implementaci. Problémy, které nastaly při implementaci, byly popsány včetně způsobu jejich řešení s ukázkami zdrojového kódu. Samotná implementace aplikace využívá moderních přístupů k vývoji webových aplikací a dodržuje zásady objektově orientovaného programování, což je podpořeno použitím frameworku Nette. Hlavní důraz byl kladen na jednoduché používání a intuitivní ovládání tak, aby uživatel – pilot – mohl co nejpohodlněji použít např. svůj telefon jako navigační přístroj. Z toho důvodu bylo vytvořeno REST API, které slouží pro komunikaci mezi zmíněnými aplikacemi.

Hlavní přínos této práce vidím v tom, že propojením webové aplikace s mobilní vznikl první neplacený systém pro piloty, který umožňuje pohodlné plánování tratí jak na počítači, tak přímo na mobilním zařízení, a zároveň řeší synchronizaci naplánovaných tras automaticky bez zásahu uživatele. Díky tomu mohou využívat výhody GPS navigace i piloti, kteří nechtějí utrácet peníze za drahé letecké navigační přístroje.

Posledním krokem před zprovozněním aplikace je výběr hostingu, na kterém poběží. Také musí následovat úprava mobilní aplikace Pocket Pilot tak, aby s nově vzniklým webovým rozhraním uměla komunikovat. Touto diplomovou prací tudíž není ukončen vývoj ani webové, ani mobilní aplikace. Po zprovoznění bude možné webové rozhraní dále rozšiřovat, např. o:

- Dynamická úprava nástěnky,
- zavedení premium účtů,
- livetracking - přenášení aktuální polohy letadla na mapu v reálném čase,
- generování a tisk navigačního štítku,
- export naplánovaných tras do formátů KML a GPX.

Cíl práce definovaný v úvodu byl naplněn, aplikace byla dle stanovených požadavků vytvořena a tvoří přílohu práce. Aktuální verzi aplikace lze najít na GitHubu v repozitáři <https://github.com/andrejsoucek/pocketpilot-web/>.

Seznam použité literatury

BUCHALCEVOVÁ, Alena; STANOVSKÁ, Iva, 2013. *Příklady modelů analýzy a návrhu aplikace v UML*. Praha: Oeconomica. ISBN 978-80-245-1922-7.

BURGE, Steve, 2018. *How Popular is Bootstrap in 2018?* [online] [cit. 2019-06-08]. Dostupné z: <https://www.ostraining.com/blog/webdesign/bootstrap-popular/>.

DRAKE, Mark, 2019. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [online] [cit. 2019-05-02]. Dostupné z: <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>.

GILBERTSON, David, 2018. *The 100% correct way to split your chunks with Webpack* [online] [cit. 2019-06-17]. Dostupné z: <https://hackernoon.com/the-100-correct-way-to-split-your-chunks-with-webpack-f8a9df5b7758>.

GILMORE, W. J., 2007. *Velká kniha PHP a MySQL 5: Kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner Press. ISBN 80-86815-53-6.

GITHUB, 2019. *Search, stars:>1* [online] [cit. 2019-06-08]. Dostupné z: <https://github.com/search?q=stars:%3E1&s=stars&type=Repositories>.

GOOGLE, 2019. *Pricing Table / Google Maps Platform / Google Cloud* [online] [cit. 2019-06-23]. Dostupné z: <https://cloud.google.com/maps-platform/pricing/sheet/>.

HÁŠKA, David, 2016. *Porovnání PHP frameworků pro tvorbu internetové aplikace*. Praha. Bakalářská práce. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky.

KANGAX, 2019a. *ECMAScript 5 compatibility table* [online] [cit. 2019-06-09]. Dostupné z: <http://kangax.github.io/compat-table/es5/>.

KANGAX, 2019b. *ECMAScript 6 compatibility table* [online] [cit. 2019-06-09]. Dostupné z: <http://kangax.github.io/compat-table/es6/>.

KANGAX, 2019c. *ECMAScript Next compatibility table* [online] [cit. 2019-06-09]. Dostupné z: <http://kangax.github.io/compat-table/esnext/>.

KOČÁREK, Michal, 2013. *Srovnání vývoje webových aplikací v Nette frameworku (PHP) a Node.JS*. Praha. Diplomová práce. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky.

MALÝ, Martin, 2009. *REST: architektura pro webové API* [online] [cit. 2019-05-26]. Dostupné z: <https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>.

NETTE FOUNDATION, 2019a. *Database Explorer* [online] [cit. 2019-04-09]. Dostupné z: <https://doc.nette.org/cs/3.0/database-explorer>.

NETTE FOUNDATION, 2019b. *Latte* [online] [cit. 2019-04-09]. Dostupné z: <https://latte.nette.org/cs/guide#toc-kontextove-sensitivni-escapovani>.

NETTE FOUNDATION, 2019c. *Nette Framework sandbox project* [online] [cit. 2019-04-09]. Dostupné z: <https://github.com/nette/sandbox>.

- NETTE FOUNDATION, 2019d. *Odeslání e-mailů* [online] [cit. 2019-04-09]. Dostupné z: <https://doc.nette.org/cs/3.0/mailing>.
- NETTE FOUNDATION, 2019e. *Začínáme s Tracy* [online] [cit. 2019-04-09]. Dostupné z: <https://tracy.nette.org/cs/guide>.
- ROHAN, Jon, 2018a. *browserlist* [online] [cit. 2019-06-17]. Dostupné z: <https://browserlist/?q=last+2+versions>.
- ROHAN, Jon, 2018b. *browserlist* [online] [cit. 2019-06-17]. Dostupné z: https://browserlist/?q=%3E0.25%2C+not+ie+11%2C+not+op_mini+all.
- SKVORC, Bruno, 2015. *The Best PHP Framework for 2015: SitePoint Survey Results* [online] [cit. 2019-04-09]. Dostupné z: <https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/>.
- SOLID IT GMBH, 2019. *DB-Engines Ranking* [online] [cit. 2019-05-02]. Dostupné z: <https://db-engines.com/en/ranking>.
- SOUČEK, Andrej, 2016. *Android aplikace pro piloty ultralehkých letadel*. Praha. Bakalářská práce. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky.
- THE PHP GROUP, 2019. *PHP: Supported Versions* [online] [cit. 2019-06-17]. Dostupné z: <https://www.php.net/supported-versions.php>.
- TÖLG, Jan, 2012. *Nette framework*. Praha. Bakalářská práce. Vysoká škola ekonomická v Praze. Fakulta informatiky a statistiky.
- UBLABOO, 2019. *ApiRouter* [online] [cit. 2019-06-17]. Dostupné z: <https://ublaboo.org/api-router/>.
- W3TECHS, 2019. *Historical trends in the usage of server-side programming languages for websites, April 2019* [online] [cit. 2019-04-28]. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language.

Přílohy

A. SQL skript pro vytvoření tabulek

```
CREATE TYPE role as ENUM('admin', 'user', 'premium');

CREATE TABLE users (
  id SERIAL PRIMARY KEY,
  username VARCHAR(30) NOT NULL,
  email VARCHAR NOT NULL UNIQUE,
  fb_uid VARCHAR,
  password CHAR(60),
  role ROLE NOT NULL DEFAULT 'user',
  reg_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  token CHAR(32) NO NULL
);
ALTER TABLE users
  ADD CONSTRAINT FbOrPwFilled CHECK (
    ( CASE WHEN fb_uid IS NULL THEN 0 ELSE 1 END
    + CASE WHEN password IS NULL THEN 0 ELSE 1 END
    ) > 0
);
CREATE OR REPLACE FUNCTION generate_token() RETURNS trigger AS $$
BEGIN
  IF NEW.token IS NULL THEN
    NEW.token := md5(NEW.username || NOW());
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_insert_users_token
  BEFORE INSERT
  ON users
  FOR EACH ROW
  EXECUTE PROCEDURE generate_token();

CREATE TABLE pass_recovery (
  token CHAR(64) PRIMARY KEY,
  user_id INTEGER REFERENCES users(id) NOT NULL,
  created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE tracks (
  id SERIAL PRIMARY KEY,
  track GEOGRAPHY(LINESTRING, 4326) DEFAULT NULL,
  user_id INTEGER REFERENCES users(id) NOT NULL,
  name VARCHAR(50) NOT NULL DEFAULT 'Unnamed track',
  cre_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE INDEX "tracks_track_gix" ON tracks USING gist(track);
```

```
CREATE TABLE dashboard (  
  id SERIAL PRIMARY KEY,  
  title VARCHAR(50) NOT NULL,  
  date DATE NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  item_1 VARCHAR(100) DEFAULT NULL,  
  item_2 VARCHAR(100) DEFAULT NULL,  
  item_3 VARCHAR(100) DEFAULT NULL,  
  item_4 VARCHAR(100) DEFAULT NULL  
);
```

B. Výpis lokální konfigurace

```
parameters:
    facebook:
        app_id: '12345'
        app_secret: 'secret'

database:
    dsn: 'pgsql:host=127.0.0.1;dbname=pocketpilot'
    user:
    password:
    debugger: true
    options:
        lazy: yes

http:
    csp:
        script-src:
            - nonce
            - unsafe-eval

tracy:
    bar:
        - Nextras\MailPanel\MailPanel(%tempDir%/mail-panel-latte)

services:
    nette.mailer:
        class: Nette\Mail\IMailer
        factory: Nextras\MailPanel\FileMailer(%tempDir%/mail-panel-mails)
```