Sem vložte zadání Vaší práce.

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

Bakalářská práce

Systém na komplexní monitorování pohybu a technického stavu vozidel

Ondřej Texler

Vedoucí práce: Ing. Viktor Černý

11. května 2015

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce Ing. Viktoru Černému za pomoc při tvorbě této práce. Dále bych chtěl poděkovat firmě Telematix za to, že mi umožnila tento projekt realizovat a mohl jsem ve firmě konzultovat technické a implementační problémy. A v neposlední řadě bych rád poděkoval mé přítelkyni a rodině za psychickou podporu.

Prohlášení

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

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

V Praze dne 11. května 2015 .....

Odkaz na tuto práci

Texler, Ondřej. *Systém na komplexní monitorování pohybu a technického stavu vozidel*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

V rámci této bakalářské práce jsem analyzoval a navrhl systém na komplexní monitorování technického stavu a pohybu vozidel – Fleet Management. Dále jsem implementoval prototyp serverové části systému. Zadavatelem tématu této práce je firma Telematix. Moje bakalářská práce se zabývá analýzou konkurenčních systémů, dále robustním návrhem celého systému, který vyhovuje všem požadavkům zadavatele a následně prototypovou implementací serverové části systému. Výsledek mojí práce bude sloužit jako podklad pro vytvoření rozsáhlého systému na monitorování vozidel ve zmiňované firmě.

Klíčová slova Monitorovací systém, pohyb vozidel, kniha jízd, technický stav vozidel, fleet management, C# server.

Abstract

In this thesis I analyzed and designed a complex system for monitoring the technical condition and the movement of vehicles - Fleet Management. Furthermore, I have implemented a prototype server part of the system. The project owner of the topic of this work is Telematix company. My thesis deals with the analysis of competitive systems, further robust design of the entire system which meets all requirements of the project owner and then a prototype

implementation of the server. The result of my work will serve as a basis for the creation of an extensive system for monitoring vehicles in the mentioned company.

Keywords Monitoring system, vehicle movement, log book, technical condition of vehicles, fleet management, C# server

Obsah

Seznam obrázků

Úvod

Mnoho firem i soukromníků má potřebu sledovat technický stav a pohyb vozidel. Průběžné monitorování technického stavu vozidel slouží jako prevence vzniku vážnějších technických závad. Funkce sledování pohybu a technického stavu vozidel najde uplatnění nejen při sledování služebních nebo půjčených vozů, ale i mimo komerční oblast. Zákazník díky monitorovacímu systému bude mít přehled například o reálných nákladech na provoz vozidla, stylu jízdy řidičů a bude si moci optimalizovat trasy tak, aby co nejvíce ušetřil. Má bakalářská práce se tedy bude zabývat vytvořením systému pro správu vozového parku. Problematika správy vozového parku je známá pod anglickým názvem Fleet-Management, což v doslovném překladu znamená správa flotily.

Kapitola 1

Cíl práce

Cílem této bakalářské práce je analyzovat současná řešení monitoringu vozidel. Na základě této analýzy navrhnu architekturu celého systému, jednotlivé části systému a jejich vzájemnou komunikaci. Rozhodnu, jaké funkce bude systém nabízet v první verzi a jaké v budoucích verzích. A implementuji prototyp hlavní části systému, server psaný v jazyce C#. Implementace serveru nebude příliš rozsáhlá, bude hodně abstraktní a jejím hlavním cílem bude definovat strukturu kódu serveru a umožnit tak lehkou rozšiřitelnost. Server dokáže sbírat data o provozu vozidel v dohodnutém formátu, vyhodnocovat je, ukládat je do databáze a na požádání poskytne veškeré dostupné informace. Tyto informace pak budou předány některé z prezentačních aplikací, mobilní aplikaci nebo webové aplikaci. V těchto prezentačních aplikacích bude uživateli umožněn grafický, či textový náhled na nasbíraná data a jejich následná editace. Cílem celého projektu je tedy:

- Analýza existujících Fleet-Management systémů.
- Návrh systému jako celku.
- Návrh stylu komunikace mezi jednotlivými částmi systému.
- Alespoň základní návrh jednotlivých částí systému.
- Prototyp serveru v jazyce C#

Motivace

Téma své bakalářské práce jsem si vybral z několika důvodů. Prvním důvodem je přínos pro společnost, respektive pro komunitu řidičů. Rád pracuji na projektu, který vím, že někomu přinese užitek, usnadní práci nebo ušetří čas. Připadá mi pak, že moje práce má smysl a to mě velice motivuje. Dalším důvod je, že já sám jsem aktivní řidič, tudíž budu moci mnou vyvíjený systém

testovat a používat. A posledním důvodem je příležitost. Já sám bych v současné době absolutně nemohl takto velký systém vyvinout. K vývoji systému tohoto rozsahu je potřeba mnoho programátorů, drahé servery a počítače, databázové servery, licence pro software a mnoho dalšího. Firma Telematix mi poskytla veškeré potřebné hardwarové a softwarové nástroje a programátory, kteří se podílí na vývoji jednotlivých částí systému podle mého návrhu.

Kapitola 2

Analýza

V této kapitole budu rozebírat možnosti, kterými by bylo možné moji práci řešit. Nejprve analyzuji, jak daný problém řeší konkurence. Od této analýzy konkurence si slibuji, že mě inspiruje a pomůže mi rozhodnout, jaké funkce a možnosti chci, aby můj systém implementoval. A dále definuji požadavky, které jsou na systém kladeny.

2.1 Současný stav řešení problematiky

2.1.1 Přehled konkurenčních firem

Na českém a zahraničním trhu existuje mnoho malých a středních firem nabízejících monitorování vozidel. Analyzuji čtyři české firmy a jednu americkou, jejichž služby mi přišly zajímavé a inspirativní. Jedná se o české společnosti Sherlog – Secar Bohemia, CCS Carnet a Car – Monitor, Lokátory a zahraniční Automatic. Těchto pět firem podrobím důkladné analýze a na jejím základě rozhodnu, jaké služby a funkce a v jaké míře bude implementovat můj systém.

2.1.2 Sherlog – Secar Bohemia

Sherlog – Secar Bohemia [1]. Tato firma se specializuje především na hledání odcizených vozidel. Dále nabízí online sledování pohybu vozidla, sledování chování řidičů, vyhodnocování stylu jízdy a knihu jízd. Zajímavou věcí je aplikace Sherlog vision demo. Jedná se o demoverzi aplikace Sherlog vision, zákazník si díky této aplikaci může zdarma vyzkoušet nabízené služby.

2.1.3 CSS Carnet

CSS Carnet [2] staví na propojení jejich monitorovacího systému s palivovými platebními kartami CSS. Dále nabízí sledování vozidel, knihu jízd, přehled nákladů na provoz vozidla, kontrolu dodržování pracovní doby, plánování a optimalizaci využití vozového parku. CSS Carnet nabízí i zabezpečení vozidel,

ale zdaleka ne na takové úrovni, jako Sherlog. CSS Carnet rovněž nabízí demo aplikaci.

2.1.4 Car - Monitor

Car – Monitor [3] je relativně mladá firma. Její předností je spolupráce se společností T-Mobile. Společnost T-Mobile poskytuje společnosti Car – Monitor zabezpečení v oblasti síťové komunikace. Car – Monitor nabízí podobné služby jako CSS Carnet, až na palivovou kartu CSS.

2.1.5 Lokátory

Z analyzovaných firem nabízí Lokátory [4] zcela jistě největší množství funkcí. Krom klasické knihy jízd, zaznamenávání provozních výdajů, tankování a real-time sledování, nabízí společnost Lokatory.cz například zónové hlídání. Pro konkrétní vozidlo lze vymezit prostor, kde se smí pohybovat a pokud vozidlo daný prostor opustí, systém pošle SMS nebo e-mail upozornění a zaznamená opuštění prostoru do databáze. Dále nabízí upozornění při překročení státních hranic nebo hlídání dodržování povolené rychlosti. Krom funkcí vycházejících z polohy GPS společnost nabízí i sledování provozních veličin vozidla, otáček motoru, stav paliva v nádrži a mnoho dalšího.

2.1.6 Automatic

Jedná se o společnost sídlící v San Franciscu, působící převážně v USA. Informace o této společnosti jsem zjišťoval na jejich webových stránkách[5]. Nabízí klasické služby, knihu jízd, diagnostiku technického stavu vozidla, dodržování povolené rychlosti a další funkce podobné jako konkurenční společnosti. Krom těchto běžných funkcí mě na Automatic zaujaly následující dvě funkce. První je funkce, kterou Automatic nazývá "Dude... here's your car". Tato funkce uživateli řekne, kde je vozidlo zaparkované, najde tedy uplatnění v situacích, kdy více uživatelů sdílí jedno vozidlo, popřípadě, když uživatel zapomene, kde zaparkoval. Druhou funkcí je "Coach your teen driver", tuto funkci využijí typicky rodiče ke sledování způsobu a stylu jízdy svých potomků. Tuto funkcionalitu pojal Automatic velice zeširoka.

2.1.7 Shrnutí

Pro návrh mého systému mi byla největší inspirací firma Lokátory. Tato firma nabízí v podstatě všechny služby, o které by mohl mít zákazník zájem. Na svých webových stránkách má veškeré funkce systému detailně popsané se spoustou screen-shotů přímo z aplikace. Dále velice inspirativní byla firma Automatic a to hlavně díky funkci "Dude... here's your car"a "Coach your teen driver".

2.2 Funkční požadavky

Zadavatel měl několik požadavků na systém. První požadavek byl, abych systém zaměřil převážně na osobní automobily. Systém by měl být určen pro soukromníky s jedním automobilem i pro firmy s mnoha desítkami automobilů. Jelikož bude určen i pro firmy, musí existovat dvě skupiny uživatelských účtů. Běžný účet a administrátorský účet. Administrátorský účet bude mít kontrolu nad jedním nebo více běžnými účty. Co se týče toho, jaké bude systém nabízet funkce, tak mi zadavatel nechal hodně volnou ruku. Jediný požadavek byl, aby systém nabízel buď velmi širokou škálu funkcí a nebo nabízel funkcí jen pár, zato velice propracovaných a byl tak konkurenceschopný.

2.3 Nefunkční požadavky

Zadavatel by rád, abych jako databázi použil nějakou neplacenou databázi, například SQLite, ale pokud shledám použití jiné databáze vhodnějším, nemusím tento nefunkční požadavek splnit.

2.4 Uživatelské role

V systému budu rozlišovat mezi dvěma uživatelskými rolemi, uživatelem a administrátorem. Uživatel bude vlastnit uživatelský účet, který bude jednoznačně spojen s jednou konkrétní vozidlovou jednotkou. Administrátor bude vlastnit administrátorský účet, který bude spravovat jeden nebo více uživatelských účtů. Typicky šéf firmy bude vlastnit administrátorský účet, který bude spravovat účty jeho zaměstnanců.

2.4.1 Administrátor

Administrátor bude mít plnou kontrolu nad všemi spravovanými účty. Bude moci prohlížet, mazat a editovat data nasbíraná z vozidlových jednotek. Vytvářet služební cesty a tyto cesty přiřazovat uživatelským účtům. Normální uživatel bude také moci vytvořit služební cestu, ale již si ji nebude moci schválit, tuto pravomoc má pouze administrátor.

2.4.2 Uživatel

Uživatel nebude mít skoro žádné právo editace a mazání dat. Bude moci nasbíraná data pouze prohlížet. Bude moci vytvářet služební cesty, ale nebude si je moci schvalovat.

Kapitola 3

Možnosti řešení

3.1 Nabízené funkce systému

Konkurenční společnosti nabízejí převážně tyto funkce: knihu jízd, online sledování vozidla, plánování jízd, sledování spotřeby paliva, zabezpečení vozidel, kontrolu dodržování pracovní doby, sledování chování řidičů, vyhodnocování jízdního stylu, přehled nákladů na provoz vozidla. Jedná se tedy o širokou škálu funkcí. Jeden možný směr, kterým se může ubírat moje bakalářská práce, je, zaměřit se na jednu problematiku a věnovat se převážně jí. Například jako Sherlog se zaměřuje na zabezpečení. Nebo mohu problém uchopit komplexněji a implementovat větší množství funkcí, avšak ne v tak dokonalém provedení, v jakém bych byl schopen implementovat užší výběr funkcí.

3.2 Způsob monitorování

K monitorování vozidla lze použít mobilní telefon nebo speciální GPS/GSM vozidlovou jednotku. Nevýhodou mobilního telefonu je, že nebude schopen monitorovat technický stav. Bude pouze zaznamenávat informace o poloze, rychlosti a zrychlení. Na druhou stranu výhodou mobilního telefonu je, že uživatel může použít svůj vlastní telefon, tedy si k monitorovacímu systému nemusí dokupovat vozidlovou jednotku ani žádné další zařízení. Vozidlová jednotka je díky napojení na sběrnici CAN nebo FMS Cotel schopna získávat provozní informace o vozidle.

3.3 Možnosti databází

Rozhodnout, jaká databáze bude pro tento systém nejlepší je velice obtížné. Možná se ani nedá jednoznačně zodpovědět a záleží na preferencích a ostatních hardwarových prostředcích. Jedna databáze bude třeba rychlá na přístup, vyhledávání, vkládání záznamů a mazání záznamů, ale bude zabírat hodně

místa na disku. A jelikož na databázi tohoto typu bude určitě potřeba SSD disk, každých několik set GB navíc je ohromný cenový rozdíl. Jiná databáze, třeba databáze, která bude komprimovaná, bude zabírat výrazně méně místa na disku, bude tedy stačit menší a levnější SSD disk, ale při práci s touto databází se bude dekomprimace provádět na procesoru a bude potřeba výkonný a drahý procesor. Já jsem se váhy zodpovědnosti za rozhodování, jakou použiji databázi, částečně vyhnul tím, že pro práci s databází použiji Entity Framework. Díky Entity Frameworku bude relativně jednoduché kdykoliv vyměnit jeden druh databáze za jiný, aniž by se musel provádět větší zásah na serveru.

3.3.1 SQL nebo NoSQL

Při výběru databáze musím rozhodnout, jestli chci použít relační SQL databázi nebo NoSQL databázi. A následně rozhodnout, jaká databáze bude co nejvíce vyhovovat mým požadavkům.

3.3.2 Relační SQL databáze

Informace o této problematice jsem čerpal z výukového materiálu Greendot [6]. Relační databáze jsou založeny na práci s tabulkami, které jsou propojeny nastavenými vztahy – relacemi. Tabulky obsahují jednotlivé sloupce – atributy a řádky – záznamy. Atributy tabulky určují vlastnosti objektů, které se do tabulky ukládají. Do jedné tabulky se ukládají objekty stejného druhu. Vztahy mezi tabulkami jsou realizovány pomocí takzvaných cizích klíčů. SQL je zkratka Structured Query Language – strukturovaný dotazovací jazyk, který je používán pro práci s daty v relačních databázích. V jazyce SQL jsou příkazy pro manipulaci s daty. Select – příkaz pro vyhledání záznamů v tabulce, insert – příkaz pro vložení záznamů do tabulky, delete – pro mazání záznamů z tabulky, create – pro vytváření tabulek a další.

3.3.3 NoSQL databáze

NoSQL je databázový koncept, ve kterém datové úložiště i zpracování dat používají jiné prostředky, než tabulková schémata, jako je tomu u klasických relačních databází. Motivací k tomuto netabulkovému přístupu může být jednoduchost designu, horizontální i vertikální škálovatelnost a další. Databáze bez SQL mohou být vysoce optimalizovaná úložiště typu klíč hodnota nebo úložiště ohromného množství dat. NoSQL databáze často používají například stromovou reprezentaci ukládání dat, proto může být algoritmická složitost pro různé operace odlišná. V současné době použití NoSQL databází značně roste a to hlavně v oblasti big-data a real-time. Předpona „No“ neznamená anglický zápor „ne“, znamená „not-only“. NoSQL databáze totiž často podporují i SQL dotazovací jazyk.

3.3.4 MySQL

Jak již název napovídá, MySQL [7] je SQL databáze, která podporuje klasické SQL dotazy a nějaké dotazy navíc. Tato databáze je multiplatformní, což pro mě není příliš podstatné, jelikož server je psán v jazyce C# na platformě. NET na Windows, takže výhodu multiplatformní databáze stejně nevyužiji. MySQL je optimalizováno na rychlost za cenu některých zjednodušení. Je šířeno jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

3.3.5 SQLite

SQLite [8] je relační databázový systém psaný v jazyce C. SQLite je velmi zvláštní, ale zajímavá databáze, tato databáze totiž nemá server jako samostatný proces. Ostatní relační databáze fungují převážně na principu klient-server. Nastartuje se server, který běží a čeká na připojení a následný dotaz klienta. SQLite je pouze malá knihovna, která se přilinkuje k aplikaci a pomocí daného rozhraní s ní lze pracovat. Každá SQLite databáze je uložena v souboru s příponou .dbm – Database Manager. K ukládání dat se používá technika hashování, aby byl zajištěn rychlý přístup k datům při vyhledávání podle klíče. SQLite je šířena pod licencí public domain, je tedy k použití zdarma.

3.4 MS SQL

MS SQL [9] je zkratka Microsoft SQL. Je tedy zřejmé, že za vývojem této databáze stojí firma Microsoft. Na rozdíl od SQLite je MS SQL velice rozsáhlý projekt se spoustou analytických nástrojů, podporou pro e-shop a dalších business řešení. Z názvu je patrné, že tato databáze podporuje dotazování pomocí jazyku SQL. Je založený na principu klient – server. Tato databáze byla navržena pro zvládnutí velkého objemu transakcí, což je pro mě podstatná informace, jelikož můj server bude hodně transakčně vytížen. Každá aktivní vozidlová jednotka bude v řádech sekund posílat informace o svojí poloze na server, a když těchto vozidlových jednotek bude víc, potřebuji server, který takový nápor zvládne. MS SQL je placený komerční produkt.

3.5 Cassandra

Databáze Cassandra [10] je první NoSQL databáze, o které se zmiňuji. Tuto databázi používá například Facebook nebo Twitter. Cassandra je velice dobře škálovatelná a je optimalizovaná na opravdu velká data, ve kterých se dá rychle vyhledávat a vkládat nové záznamy. Jako datový model používá strom. V současné době je Cassandra stále aktivně vyvíjena, poslední stabilní verze Cassandry je verze 2.1.5 vydaná 29.4.2015. Cassandra nepoužívá dotazovací jazyk SQL, ale má vlastní dotazovací jazyk zvaný CQL – Cassandra Query

Language. Jedná se o projekt licencovaný jako Apache verze 2, tedy lze použít i bezplatně.

3.6 MongoDB

Jedná se o velmi rozsáhlý a hojně využívaný projekt. MongoDB [11] nepoužívá tabulky, na které jsme zvyklí u relačních databází. Pro uložení dat používá BSON, což je obdoba formátu JSON, akorát že není textová, ale binární, proto B místo J. MongoDB nepodporuje dotazovací jazyk SQL, ale má vlastní dotazovací jazyk, který mi přijde, že dotazovacímu jazyku SQL moc podobný není. Nejen, že mu není podobný syntaxí, ale není mu ani podobný stylem použití. Dotazovací jazyk v MoongoDB vypadá spíše jako volání metod nad objekty. Já jsem zastánce objektově orientovaného programování, takže mi je tento styl sympatický. Pravdou ale je, že většina programátorů je zvyklá na dotazovací jazyk SQL nebo na některý dialekt tohoto jazyka a naučit se pracovat s takto odlišným dotazovacím jazykem, jako je dotazovací jazyk používaný v MongoDB, může být zdlouhavé a náročné. MongoDB je opensource projekt, takže se dá bezplatně použít.

Kapitola 4

Zvolené řešení

Na základě podrobné analýzy konkurenčních společností, finančních, časových a technologických prostředků, jsem se rozhodl pro následující řešení.

Poznámka: Kdekoliv se v této a následujících kapitolách zmíním o serveru a nespecifikuji, jestli se jedná o data nebo account server, myslím data server.

4.1 Zvolené řešení systému

Systém pojmu komplexně, tedy nezaměřím se pouze na jednu oblast problematiky, ale navrhnu a později i implementuji většinu zmiňovaných funkcí. Funkci hledání odcizeného vozidla a zabezpečení vozidla nebudu navrhovat ani implementovat vůbec. Tuto problematiku má velice dobře vyřešenou společnost Sherlog a bylo by velice obtížné jim konkurovat. Navíc kvalitní zabezpečení vozidel vyžaduje rozsáhlejší servisní úpravy vozidla, čímž by došlo k výraznému zvýšení ceny systému. V rámci své bakalářské práce nebudu implementovat ani navrhovat kompletní seznam všech funkcí, které bude systém v budoucnu poskytovat, ale navrhnu a implementuji robustní jádro systému, které bude velmi snadno rozšiřitelné a navrhnu komunikaci mezi jednotlivými komponentami systému. Do databáze budu ukládat taková data a v takové formě, aby na jejich základě šlo postavit v podstatě libovolnou funkcionalitu systému.

4.2 Administrace systému

Základní částí systému bude vozidlová jednotka (OBU – on board unit). Každá vozidlová jednotka bude jednoznačně přiřazena k vozidlu. Nebude tedy možnost přenášet vozidlovou jednotku mezi různými vozidly. Vozidlovou jednotku bude spravovat jeden administrátorský účet a jeden uživatelský účet. Typicky - přístup k administrátorskému účtu bude mít šéf firmy. Přístup na uživatelské účty budou mít zaměstnanci. Přes jeden administrátorský účet lze spravovat

více vozidlových jednotek. Na uživatelský účet bude připadat pouze jedna vozidlová jednotka.

4.3 Vytváření účtů

Dlouho jsem přemýšlel, jaký bude nejlepší způsob vytváření uživatelských a administrátorských účtů. A došel jsem k následujícím závěrům.

4.3.1 Vytváření uživatelských účtů

Co se týče účtů uživatelských, tak je řešení, vzhledem k tomu, jak jsem definoval vztah vozidlových jednotek a uživatelských účtů, poměrně jednoduché a přímočaré. Jelikož základní částí systému bude vozidlová jednotka, která bude jednoznačně spárovaná s uživatelským účtem, nejlepší možnost, jak řešit vytváření uživatelských účtů, bude následující. Při vytvoření nové vozidlové jednotky bude vytvořen i uživatelský účet. Než firma Telematix dá vozidlovou jednotku koncovému uživateli, opatří tuto jednotku štítkem, na kterém bude uvedeno přihlašovací jméno a vygenerované heslo. Toto vygenerované náhodné přihlašovací jméno i heslo půjde samozřejmě změnit. Podobný systém v dnešní době funguje u většiny software zakoupeného v kamenném obchodě. Když si zákazník koupí software v takovémto obchodě, často je na krabici od produktu nalepen štítek s registračním klíčem. Uživatel si posléze vytvoří nový účet a spáruje tento vytvořený účet s registračním klíčem. Varianta, že uživatel místo registračního klíče rovnou dostane přístupové informace k již vytvořenému uživatelskému účtu, mi přijde pro uživatele jednodušší a pohodlnější. Uživatel má možnost heslo a jméno změnit, pokud chce, ale když se jedná o méně zkušeného uživatele, nemusí se o nic starat a může začít účet hned používat.

4.3.2 Vytváření administrátorských účtů a párování s uživatelskými účty

Vyřešení vytváření administrátorských účtů je o něco složitější. A hlavně systém párování administrátorských účtů s účty uživatelskými je třeba dobře promyslet. Definoval jsem, že každá vozidlová jednotka je jednoznačně spárována s právě jedním uživatelským účtem a právě jedním administrátorským účtem. Ve chvíli, kdy bude vozidlové jednotce přiřazen uživatelský účet, by bylo ideální ji přiřadit i k nějakému administrátorskému účtu. Abych zachoval konzistenci systému, nedovolím tedy prodat zákazníkovi vozidlovou jednotku, aniž by měla přiřazený jak uživatelský, tak administrátorský účet.

Pokud přijde nový zákazník a bude si chtít koupit pouze jednu vozidlovou jednotku. Firma Telematix dá zákazníkovi vozidlovou jednotku s přiděleným uživatelským účtem (viz kapitola 4.3.1) a vygeneruje zákazníkovi i administrátorský účet, který bude spravovat pouze tuto jednu vozidlovou jednotku.

Zde jsem přemýšlel nad otázkou, jestli není zbytečně složité, že k jedné samostatné vozidlové jednotce budou dva účty, uživatelský a administrátorský a zda by nebylo lepší, udělat pro tyto případy výjimku a zrušit administrátorský účet a ponechat pouze uživatelský účet, ale s plnými pravomocemi. Nebo lepší řešení, zrušit uživatelský účet a ponechat jen administrátorský účet. Pokud bych chtěl tuto situaci vyřešit jedním z těchto dvou způsobů, zahrnovalo by to návrhové a implementační změny. Když jsem přemýšlel, jak tuto situaci návrhově vyřešit, napadlo mě, že současný stav řešení dvou účtů k jedné samostatné vozidlové jednotce je sice na první pohled složitý, ale zato nabízí novou možnost využití. V kapitole s analýzou jsem se zmiňoval o funkci s názvem „Coach your teen driver“, která mě zaujala u firmy Automatic. Jedná se o funkci, která umožňuje rodičům sledovat vozidlo, jež svěřili svým potomkům. A jaké je tedy propojení s problémem, který právě diskutuji? Nabízí se využití, že přístup k administrátorskému účtu si nechá rodič a potomkovi dá přístup pouze k uživatelskému účtu. Potomek se tedy bude moci podívat, kolik ujel, kde jezdil, vyplňovat kolik utratil za tankování a další, ale nebude moci mazat nebo upravovat záznamy. Tudíž rodič bude mít přehled o všech řidičských aktivitách potomka. Tedy zákazníci, kteří budou vozidlovou jednotku chtít použít pouze sami pro svoje osobní potřeby, budou mít dva účty, ale budou používat pouze administrátorský účet a s uživatelským účtem nebudou nic dělat. A zákazníci, kteří budou vozidlo někomu svěřovat, například potomkovi, budou mít možnost využít i uživatelský účet.

V případě, že přijde zákazník, který bude chtít novou vozidlovou jednotku ke stávajícímu administrátorskému účtu, pracovník Telematixu mu vydá jednotku a manuálně tuto jednotku přiřadí k zákazníkovu stávajícímu administrátorskému účtu. Pokud přijde zákazník, který ještě nemá žádný administrátorský účet nebo bude chtít nový administrátorský účet, pracovník Telematixu vytvoří nový administrátorský účet a s tímto účtem spáruje všechny prodávané vozidlové jednotky.

Pokud bude mít jeden zákazník více administrátorských účtů a bude mezi nimi chtít přeřazovat vozidlové jednotky, bude tento požadavek muset vyřizovat přes technickou podporu firmy Telematix. Toto by však neměla být komplikace, předpokládám, že požadavky na přeřazení vozidlové jednotky z jednoho administrátorského účtu na druhý budou velmi řídké. Pokud by se ukázalo, že je těchto požadavků více, než jsem předpokládal, mám vymyšlené řešení, jak tento proces automatizovat, tím však v současné době nebudu systém zbytečně komplikovat.

Ve všech případech tedy zůstane zachována podmínka, že každá vozidlová jednotka má právě jeden administrátorský a jeden uživatelský účet. Na jednu stranu je škoda, že zákazník nebude mít možnost přiřazovat vozidlové jednotky k administrátorským účtům a přeřazovat vozidlové jednotky mezi uživatelskými účty a v těchto případech bude muset kontaktovat technickou podporu, na druhou stranu se zákazník při koupi nové vozidlové jednotky nebo založení nového administrátorského účtu nebude muset starat o párování s

vozidlovými jednotkami. Moje zkušenosti se zákazníky jsou takové, že i když zákazník dostane jasný a stručný návod, jak něco udělat a zdá se, že není možné, aby v tomto postupu udělal chybu, vždycky se najde spousta zákazníků, kteří nejsou schopni tento návod dodržet a dosáhnout cíle a tyto zákazníci pak stejně musejí kontaktovat technickou podporu a ještě jsou rozmrzelí, protože si myslí, že chyba je na straně produktu. Kdyby si zákazník sám pároval vozidlové jednotky s administrátorskými účty a mohl přesouvat vozidlové jednotky mezi administrátorskými účty, myslím si, že by to způsobovalo více problémů, než užitku. Navíc jeden z mých funkčních požadavků je, aby byl systém jednoduchý na administraci a používání.

4.4 Zvolené funkce systému

Momentálně je velmi obtížné určit, jaké všechny funkce bude systém nabízet. Přesný seznam a rozsah funkcí určím na základě zpětné vazby od uživatelů a testerů první verze systému. Hlavní podmínkou na systém je navrhnout jej tak robustně, aby nebylo náročné upravovat a přidávat novou funkcionalitu během vývoje. První verze systému, kterou budou testovat vybraní uživatelé a testeři, bude implementovat základní funkce, jejich seznam je následující:

- Aktuální poloha – aktuální poloha vozidla na mapě a informace o tom, jestli je vozidlo v pohybu, popřípadě, jak rychle se pohybuje, nebo jestli stojí na místě se zapnutým nebo vypnutým zapalováním. Bude zde i možnost sledovat zároveň více vozidel v jedné mapě.
- Kniha jízd – podrobná elektronická kniha jízd. Uživatel vozidlové jednotky bude mít určitou možnost knihu jízd editovat. Plnou možnost editovat knihu jízd bude však mít pouze administrátor vozidlové jednotky. V rámci této funkce bude možné zobrazit všechny trasy v mapě a dále statistický přehled o ujeté vzdálenosti za daný časový úsek, průměrné rychlosti, cenu za trasu a mnoho dalších statistických údajů.
- Typy jízd – systém bude umožňovat rozlišovat služební a soukromé jízdy.
- Tankování – zaznamenávání tankování pohonných hmot. Informace o místě tankování, typu, ceně a množství paliva.
- Provozní výdaje – kompletní přehled o výdajích spojených s provozem vozidla. Tankování, náklady na servis, pojištění, cena mýtného a další.
- Zónové hlídání – hlídání, jestli vozidlo neopustilo danou oblast. Tyto oblasti lze definovat z administrátorského účtu pro konkrétní vozidlovou jednotku. Při opuštění vymezené oblasti systém zapíše záznam do databáze, pošle e-mail nebo SMS.
- Překročení hranic – zaznamenávání o překročení státních hranic

- Hlídání rychlosti – zaznamenávání překročení maximální povolené rychlosti, tato informace bude zaznamenána v knize jízd.

- Provozní veličiny – sledování a zaznamenávání provozních veličin ze sběrnice CAN nebo FMS v závislosti na typu vozidla.

4.5 Zvolený způsob monitorování

Rozhodl jsem, že k vlastnímu monitorování vozidla nepůjde použít mobilní telefon, ale bude použita vozidlová jednotka 4.3 4.4 připojená do sběrnice CAN nebo FMS Cotel. Kdyby byl na monitorování použitý mobilní telefon, nebylo by možné zaznamenávat technické informace, což by nebyl až takový problém - pokud by uživatel chtěl mít i technické informace, musel by si dokoupit vozidlovou jednotku, pokud by uživateli stačily informace z mobilního telefonu, což jsou pouze poloha, rychlost, zrychlení a další informace vypočítané z těchto třech základních informací, mohl by uživatel používat tuto levnější variantu. Problém však vzniká tím, že mobilní telefon je přenositelný mezi více automobily. Výše jsem definoval, že každá vozidlová jednotka bude přiřazena právě k jednomu vozidlu. Kdyby vozidlová jednotka byla přenositelná, musel by systém pro každou vozidlovou jednotku, v tomto případě mobilní telefon, počítat s tím, že vozidlová jednotka může být používána v různých vozidlech a při přesunutí z jednoho vozidla do druhého by uživatel musel systému říci, že byla jednotka přesunuta. Uživatel by mohl zapomenout a mohlo by dojít k nekonzistenci nasbíraných dat. Dále by uživatel musel v mobilním telefonu před každou jízdou zapínat aplikaci a po ukončení jízdy aplikaci zase vypínat. Kdyby uživatel zapomněl, došlo by opět k nekonzistenci dat. U vozidlové jednotky připojované do sběrnice CAN lze poznat, do jakého vozidla je jednotka připojena. Ze sběrnice CAN lze mimo jiné získat číslo VIN (Vehicle Identification Number), česky identifikační číslo vozidla, pomocí něhož lze ověřit, jestli je vozidlová jednotka připojena do správného vozidla.

4.6 Zvolená databáze

Dlouho jsem přemýšlel, jakou databázi bude nejvhodnější použít. Můj projekt bude pracovat s velkým množstvím dat, nabízelo by se tedy použít některou z NoSQL databází, které jsou zpravidla optimalizovány pro big-data. Na druhou stranu s NoSQL databází nemám žádné zkušenosti a ve firmě, kde bude tento projekt nasazen, se většinou používají klasické relační SQL databáze, konkrétně MS SQL. Databáze MS SQL je hodně komplexní a nabízí možnost komprese dat. Data, která budu do databáze ukládat, by měla být velmi dobře komprimovatelná, tudíž při zvolení databáze MS SQL by neměl být problém s nedostatkem místa na disku. Ze zmiňovaných SQL databází se mi tedy nejvíc líbí MS SQL.

Co se týče NoSQL databází, velmi mě zaujala databáze Cassandra a to hlavně díky jejímu dotazovacímu jazyku CQL. MongoDB má zajímavý dotazovací jazyk, ale jeho použití mi připadá velice složité. Ze zmiňovaných NoSQL databází se mi tedy více než MongoDB zamlouvá databáze Cassandra.

Vybral jsem kandidáta ze skupiny SQL databází i skupiny NoSQL databází, MS SQL a Cassandra. Faktem, díky kterému jsem se rozhodl, jakou z těchto dvou databází použít, mi nakonec bylo zachování konzistence s ostatními projekty ve firmě Telematix. V této firmě většina projektů používá SQL databáze, tudíž programátoři ve firmě jsou na SQL databáze zvyklí. Kdybych zvolil NoSQL databázi, programátoři by se museli s touto novou databází učit pracovat, což by bylo zbytečně složité a zdlouhavé. Nehledě na to, že já sám nemám s NoSQL databázemi žádné praktické zkušenosti, takže na SQL databázi se mi bude projekt určitě lépe vyvíjet. Na strojích, kde jsem systém vyvíjel, testoval a kde bude spuštěna první verze systému, běží MS SQL databáze, takže použití MS SQL pro mě bylo jednodušší a příjemnější. Jeden z nefunkčních požadavků zadavatele sice je, abych použil databázi, která není placená. MS SQL placená je. Ale pokud bude dobrý důvod použít placenou databázi, není nutné tento nefunkční požadavek splnit. Navíc díky tomu, že na práci s databází použiji Entity Framework, bude možné i později, a to bez většího zásahu do serveru, zaměnit MS SQL databázi za jinou databázi. Ve svém projektu tedy použiji databázi MS SQL.

Obrázek 4.1 ilustruje, jak bude databáze vypadat.

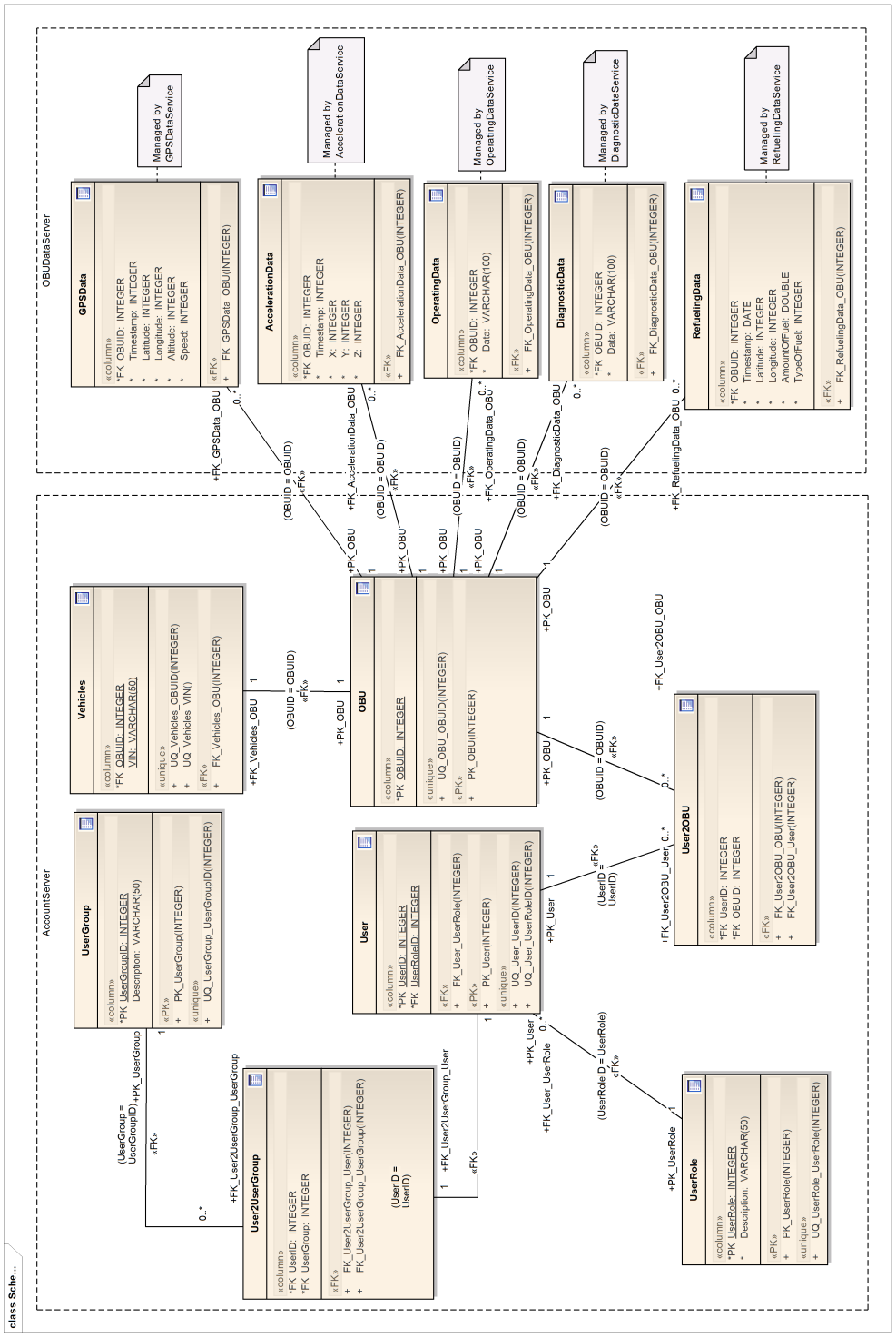
4.7 Části systému

- Server
  - Data server
  - Account server (jeho implementace není součástí práce)
- Databáze
- Vozidlová jednotka
- Mobilní prezentační aplikace
- Webová prezentační aplikace

Situaci znázorňuje vývojový diagram 4.2 nacházející se na konci této kapitoly.

Centrální část systému bude server, který bude rozdělen na dvě části - data server a account server. Data server bude sloužit k přijímání, vyhodnocování a ukládání dat. Bude psán v jazyce C#. S databází bude komunikovat pomocí Entity Frameworku. Data server bude s vozidlovou jednotkou a prezentačními

Obrázek 4.1: Doménový model databáze



aplikacemi komunikovat přes HTTP s využitím ServiceStacku. Bude poskytovat prezentačním aplikacím nasbíraná data v dohodnutém formátu. Data server sám nebude mít žádný grafický výstup, veškerá data bude poskytovat v textové formě přes HTTP prezentačním aplikacím. Account server není součástí této práce, bude sloužit k přihlašování uživatelů a správě jejich účtů.

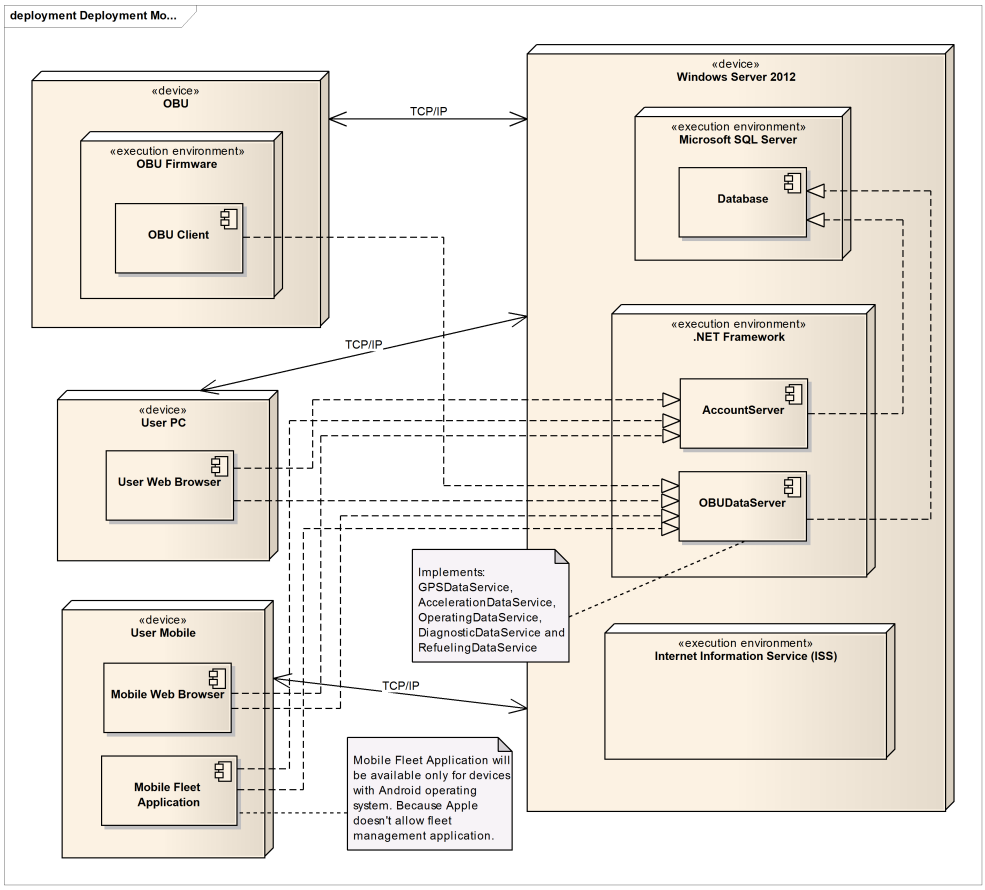
Jako databázi použiji MS SQL (podrobněji diskutováno v kapitole 4.6). Pro komunikaci s databází použiji Entity Framework. Tento framework vytváří abstraktní vrstvu nad databází a je velice jednoduché vyměnit databázi za jinou. Jednoduše lze zaměnit SQL databázi za jinou SQL databázi i SQL databázi za NoSQL databázi.

Vozidlová jednotka (OBU) bude odesílat GPS polohu, aktuální rychlost, zrychlení a veškeré dostupné technické informace. Dostupnost informací o technickém stavu vozidla se bude lišit v závislosti na konkrétním vozidle. Vozidlová jednotka bude se serverem komunikovat přes GSM. Bude umět posílat a zpracovávat HTTP požadavky dotazovací metodou POST.

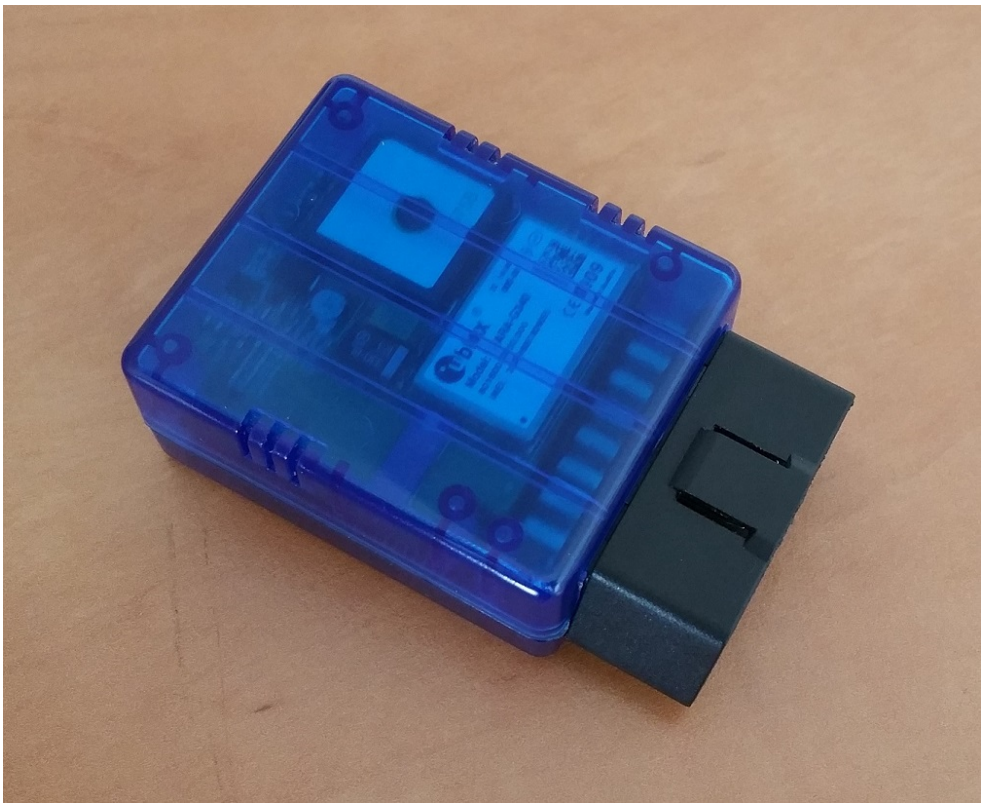
Prezentační aplikace pro mobilní zařízení - aplikace bude k dispozici pro operační systémy iOS a Android. Bude sloužit k editaci a prohlížení dat. Data bude možné zobrazit jak v textové, tak v grafické podobě (například graf spotřeby paliva nebo vyhodnocení dat z knihy jízd bude zobrazeno přímo v mapě). Tato aplikace bude podporovat přihlášení administrátorských i uživatelských účtů.

Webová prezentační aplikace - bude mít stejné funkce jako mobilní aplikace.

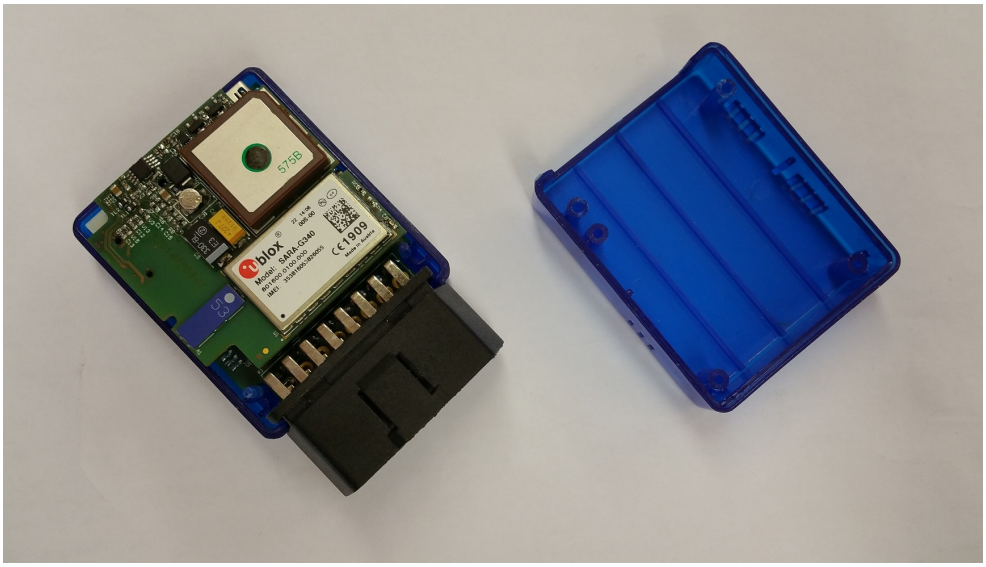
Obrázek 4.2: Diagram nasazení



Obrázek 4.3: Prototyp vozidlové jednotky



Obrázek 4.4: Prototyp vozidlové jednotky - detail



Kapitola 5

Zvolené technologie

Technologie, které mojí práci velice ovlivnily, jsou ServiceStack a EntityFramework. Pokud bych se nerozhodl pro použití těchto technologií, implementační část mé práce by pravděpodobně vypadala diametrálně odlišně, než jak vypadá teď.

5.1 ServiceStack

Komunikace se serverem bude realizovaná přes HTTP. Potřebuji tedy najít vhodný způsob, jak posílat data na server a data ze serveru získávat. Pro tento účel definuji na serveru REST API a komunikovat se serverem budu právě přes toto rozhraní. REST je architektura, která umožňuje přistupovat k datům přes protokol HTTP. V jazyce C# jsou tři základní možnosti, jak vytvořit webovou službu s REST rozhraním. ServiceStack, WebAPI a WCF. WebAPI i WCF jsou přímo od Microsoftu, WebAPI je součástí ASP.NET MVC Frameworku a WCF je tvořeno sadou knihoven v .NET frameworku. ServiceStack je komerční projekt.

Proč jsem se rozhodl pro ServiceStack

Hlavním důvodem je možnost vytváření nových a upravování starých služeb. V rámci ServiceStacku je to velmi snadné a implementačně rychlé. Při použití ServiceStacku bude tedy splněna podmínka na snadnou rozšiřitelnost systému. Na ServiceStacku se mi líbí, že podporuje několik formátů přenosu dat a mezi nimi je i formát JSON, který hodlám použít. ServiceStack je sice placený komerční produkt, ale má i starší verzi, která je zdarma a tato verze plně pokrývá moje nároky. ServiceStack je velmi oblíbený v komunitě programátorů, dá se tedy na internetu najít spousta návodů, příkladů a řešených problémů. Zadavatel projektu používá ve většině svých webových aplikací právě ServiceStack, bude tedy dobré zachovat konzistenci a použít stejný systém.

5.2 Entity Framework

Co je Entity Framework

Entity Framework je framework v jazyce C#, který podporuje takzvané ORM (objektově relační mapování). ORM je v softwarovém inženýrství technika, která zajišťuje konverzi dat mezi objektově orientovaným programovacím jazykem a databází. Dalo by se říci, že tato technika odstiňuje programátora od vlastní databáze. Programátor pracuje pouze s objekty a tyto objekty jsou přes ORM mapovány na jednotlivé záznamy v databázi. Například při ukládání dat do databáze programátor nepíše žádný SQL dotaz (ani jiný dotaz), ale pouze vytvoří instanci dané třídy a tuto instanci uloží pomocí Entity Frameworku do databáze. A až Entity Framework interně pracuje s databází například přes SQL dotazy.

Proč jsem se rozhodl pro Entity Framework

Důvodů jsem měl několik, pokusím se je sestupně seřadit podle váhy pro rozhodování.

Pomocí Entity Frameworku se dá velice rychle a efektivně programovat. Když má programátor s Entity Frameworkem nebo podobným systémem zkušenost, je schopen velice rychle upravovat nebo rozšiřovat systém. Například přidat tabulku do databáze a napsat k příslušné tabulce metody pro práci s ní lze velice jednoduše. Jeden z mých hlavních požadavků na systém je jeho modifikovatelnost a hlavně rozšiřitelnost a k tomuto účelu je Entity Framework ideální.

Dalším důvodem pro zvolení Entity Frameworku je skutečnost, že zadavatel tohoto projektu využívá Entity Framework ve svých ostatních projektech, tudíž jeho programátoři s Entity Frameworkem umí a budou tedy schopni efektivně provádět případné modifikace systému.

Entity Framework se dá chápat jako interface mezi programem a databází. Podporuje práci nejen se SQL databázemi, ale i s NoSQL databázemi, jako například MongoDB nebo Cassandra. Pro MongoDB lze použít modul MongoDB.Driver.Linq. Entity Framework komunikuje s databázemi interně a navenek používá univerzální rozhraní, je tedy relativně triviální zaměnit například MS SQL databázi za MongoDB, což vyhovuje mému požadavku na robustnost a rozšiřitelnost systému. Rozhodl jsem se pro použití SQL databáze, ale připouštím možnost, že bude v budoucnu lepším řešením (možná dokonce nutným řešením) použít nějakou NoSQL databázi, hlavně pokud se systém rozšíří, naroste počet uživatelů a množství dat ukládaných do databáze.

Jako předposlední, pro mě také velice důležitý fakt při rozhodování, je ten, že nedávno jsem se s Entity Frameworkem setkal na jiném projektu.

Celá myšlenka ORM a Entity Framework mě velice zaujala a rád se o této problematice dozvím něco víc.

Entity Framework je moderní a stále se vyvíjející open source projekt. Tyto aspekty byly pro moje rozhodování také velice důležité.

Kapitola 6

Implementace a testování

Moje bakalářská práce je z převážné většiny návrhová. V zadání práce jsem si určil, že implementuji pouze serverovou část systému, ale v průběhu psaní práce a implementování serverové části jsem začal mít chuť zkusit si nad rámec práce implementovat i další části systému, jako například webovou prezentační aplikaci nebo aplikaci pro Android. Tak jsem začal přemýšlet, jak tyto aplikace začnu implementovat a uvědomil jsem si, že bude daleko lepší se dál a podrobněji věnovat návrhu systému jako celku, protože dokud nemám robustní a komplexní návrh systému, nemohu začít implementovat, takže nakonec jsem implementoval opravdu jen část serveru, která se stará o ukládání a zprostředkování dat z vozidlových jednotek a většina práce je tedy analýza a návrh.

Při implementaci serveru jsem použil jazyk C# a s ním spojený framework .NET. Dále jsem použil webovou službu ServiceStack a framework pro práci s databází Entity Framework.

V prototypové implementaci serveru jsem implementoval pouze několik základních funkcí. Celý data server jsem implementoval velice abstraktně a to proto, aby bylo možné jednoduše přidat další funkcionalitu.

6.1 Implementované služby na serveru

Server implementuje celkem pět služeb na posílání dat na server a získávání dat ze serveru. Tyto služby jsou: GPSDataService, AccelerationDataService, OperatingDataService DiagnosticDataService a RefuelingDataService. Každá z těchto služeb má momentálně jednu metodu sloužící k posílání dat na server. Služba GPSDataService má i metodu pro dotazování se na nasbíraná data. Všechny služby jsou navrženy a implementovány tak, aby bylo jednoduché přidat jim další metody. Přidání nové metody je časově náročné pouze v řádech několika málo minut.

6.2 Popis jednotlivých služeb běžících na serveru

Krátký popis všech pěti služeb implementovaných na serveru. Jaká data tyto služby přijímají, jaké provádějí zásahy do databáze a jaká data vracejí.

Data na server a ze serveru přenáším přes protokol HTTP v textové formě, a to konkrétně ve formátování JSON. ServiceStack umožňuje pracovat s daty ve formátu JSON, takže jsem nemusel data nijak složitě parsovat. Jedinou výjimku tvoří GPS data posílaná z vozidlové jednotky. Binární reprezentaci těchto dat jsem zakódoval do řetězce formátovaného jako Base64, neposílám je tedy jako JSON, ale jako obyčejný Base64 řetězec, a to z toho důvodu, že GPS data budu na server posílat z vozidlové jednotky velice často a díky tomuto kódování nebude muset vozidlová jednotka posílat na server tak velký objem dat.

6.2.1 GPSDataService

Jedná se o službu pracující s daty o poloze vozidla. Tato služba jako jediná nepřijímá data ve formátu JSON, ale řetězec dat v kódování Base64. Tento řetězec na serveru rozparsuji a dostanu informaci o tom, jaké je identifikační číslo jednotky, která data posílá, jaký je čas pořízení dat, zeměpisné souřadnice včetně nadmořské výšky a aktuální rychlost vozidla. Kódování Base64 místo formátu JSON jsem použil, abych snížil velikost posílaných packetů. Informace o poloze se budou posílat velmi často a bylo by zbytečné posílat velká data, když se tato data dají velice snadno zakódovat a zmenšit.

6.2.2 AccelerationDataService

Tato služba přijímá a ukládá do databáze informace z akcelerometru, identifikační číslo vozidlové jednotky, čas pořízení záznamu a zrychlení ve všech třech osách x, y a z.

6.2.3 OperatingDataService

Služba OperatingDataService se stará o zpracovávání provozních dat,což jsou například teplota motoru, otáčky motoru, teplota pod kapotou, teplota v kabině, stav hladiny oleje v motoru a mnohé další, v závislosti na typu vozidla. Jelikož zatím nevím a možná ani nikdy přesně vědět nebudu, jaká všechna operační data mi budou z vozidlové jednotky posílána, budu operační data do databáze ukládat jako string, který bude ve formátu JSON. Tím zajistím efektivní možnost rozšiřitelnosti. Služba OperatingDataService tedy přijímá pouze dva parametry, jeden je identifikační číslo vozidlové jednotky a druhým parametrem je string ve formátu JSOU nesoucí širokou škálu provozních dat.

6.2.4 DiagnosticDataService

V rámci služby DiagnosticDataService jsou zpracovávány diagnostická data vozidla, což můžou být různé chybové kódy. Varování o nefunkčnosti air-bagů, úbytku oleje v motoru a další, opět v závislosti na typu a možnostech vozidla. Stejně jako u operačních dat nevím, jaká všechna data mi na server budou posílána, zachovám se k těmto datům stejně jako u OperatingDataService, to znamená, že do databáze uložím pouze identifikační číslo vozidla a string ve formátu JSON, ve kterém budou všechna data obsažena.

6.2.5 RefuelingDataService

Poslední je služba zaznamenávající informace o tankování. Jedná se o trochu jinou službu, než služby předchozí. Dlouho jsem váhal, jestli ji mám také implementovat na datovém serveru nebo spíše na account serveru. Jde o službu pracující s daty, ale tato data nebudou posílána z vozidlové jednotky, nýbrž z prezentační aplikace, kam uživatel manuálně zapíše záznam o tankování. Služba RefuelingDataService přijímá a zpracovává následující data: identifikační číslo vozidla, čas tankování, polohu čerpací stanice, množství natankované pohonné hmoty a informaci, o jaké palivo se jedná.

6.3 Rozdělení zdrojových kódů serveru

Zdrojové soubory na datovém serveru lze rozdělit podle typu a funkce do pěti základních kategorií.

- Services
- Dto
- Model
- Mapping
- Repositories

Toto rozdělení odpovídá adresářové struktuře zdrojových souborů na serveru. Celý návrh je opravdu velmi abstraktní a může se zdát těžký na pochopení a zbytečný. Pouze u takovéto míry abstrakce však lze dosáhnout snadné rozšiřitelnosti serveru. Jako první se zmíním o kategorii Services, tato kategorie se mi jeví jako nejvíce přímočará.

Services

Service je ta část serveru, která zprostředkovává komunikaci přes HTTP protokol. Každá service je třída, která dědí od třídy s názvem Service, jež je

implementovaná v ServiceStacku. Jako názornou ukázku použiji service třídu ze svého projektu AccelerationDataService.

Kód 6.1: Ukázka třídy dědící od třídy Service

```
public class AccelerationDataService : Service
```

V následujícím textu se vědomě dopustím programátorské nepřesnosti, když řeknu, že přes HTTP zavolám metodu, přes HTTP se samozřejmě žádné metody nevolají, ale představit si danou situaci jako volání metody, mi přijde na pochopení principu ServiceStacku vhodné. V každé service třídě je metoda Any, která představuje tu metodu, kterou voláme přes HTTP. Celá práce se ServiceStackem se dá představit jako volání metod přes HTTP protokol. Do HTTP zadám adresu serveru, dále název metody a pomocí dotazovací metody POST pošlu data ve formátu JSON (nebo jiném formátu, záleží na konkrétním nastavení). ServiceStack tyto data správně rozparsuje a vytvoří z nich objekt, který jde jako parametr metodě Any.

Kód 6.2: Příklad deklarace metody Any

```
public object Any(AccelerationDataSet request)
```

V metodě Any zpracuji data, v tomto případě data uložím do databáze a vrátím stav operace. False, pokud se uložení do databáze nepovedlo a True, pokud vše dopadlo v pořádku.

Kód 6.3: Práce s daty v metodě Any

```
Model.AccelerationData accelerationDataModel
    = new Model.AccelerationData();
accelerationDataModel.OBUID = request.OBUID;
accelerationDataModel.x = request.x;
accelerationDataModel.y = request.y;
accelerationDataModel.z = request.z;
accelerationDataModel.Timestamp = request.Timestamp;
```

Objekt request typu AccelerationDataSet obsahuje data, která byla poslána přes HTTP. AccelerationDataSet tedy reprezentuje objekt, který přenáším internetem. Data z tohoto objektu uložím do nového objektu typu AccelerationData, který reprezentuje jeden záznam v databázi.

Kód 6.4: Vlastní uložení dat do databáze

```
m_AccelerationDataRepository.
        Create(accelerationDataModel);
m_unitOfWork.SaveChanges();
```

A následně uložím objekt typu AccelerationData do databáze.

Dto

Dto je velice jednoduchá třída. Tato třída pouze definuje, jak vypadá objekt, který je přenášen přes HTTP. Dto je zkratka anglických slov Data Transfer Object, volně přeloženo jako přenášený objekt s daty. Na ukázku zase použiji část svého kódu, který se stará o ukládání dat z akcelerometru.

Kód 6.5: Ukázka třídy Dto

```
public class AccelerationDataSet : IReturn<string>
    {
        public int OBUID { get; set; }
        public int Timestamp { get; set; }
        public int x { get; set; }
        public int y { get; set; }
        public int z { get; set; }
    }
```

Tato třída říká, jaká data se mají posílat metodou POST, v tomto případě je to identifikační číslo vozidlové jednotky, čas pořízení záznamu a zrychlení v ose x, y a z. Pro správné fungování je nutné všechny proměnné v této třídě deklarovat jako public a vygenerovat jim setter a getter.

Model

Třída model je velice podobná třídě Dto. Třída Dto definuje, jak vypadá objekt, který je přenášený přes HTTP. Třída Model definuje, jak vypadá objekt ukládaný do databáze. V příkladu s daty z akcelerometru vypadá Dto a Model, až na jednu proměnnou, stejně. Model má navíc proměnnou Id, což je unikátní identifikátor daného záznamu v databázi.

Kód 6.6: Ukázka třídy Model

```
public class AccelerationData
{
        public long Id { get; set; }
        public int OBUID { get; set; }
        public int Timestamp { get; set; }
        public int x { get; set; }
        public int y { get; set; }
        public int z { get; set; }
}
```

Mapping

Mapping je třída, která definuje, jak bude objekt typu Model ukládán do databáze. Definuje, co bude primární klíč, jaké položky jsou povinné a další.

Kód 6.7: Ukázka mapování objektu typu Model do databáze

```
Property(p => p.Id).
    HasDatabaseGeneratedOption(
    DatabaseGeneratedOption.Identity);
Property(p => p.OBUID).IsRequired();
Property(p => p.Timestamp).IsRequired();
Property(p => p.x).IsRequired();
Property(p => p.y).IsRequired();
Property(p => p.z).IsRequired();
```

Volání metody HasDatabaseGeneratedOption(DatabaseGeneratedOption.Identity) zajišťuje to, aby sloupec v databázi s názvem Id byl unikátní automaticky inkrementovaný identifikátor. Metoda IsRequired() zajišťuje, aby byla daná položka povinná.

Repositories

V adresáři Repositories jsou ke každé service dva zdrojové soubory. Jeden zdrojový soubor s názvem AddelerationDataRepository, v příkladu s daty z akcelerometr a soubor definující interface s názvem IAccelerationDataRepository. Oba tyto zdrojové soubory jsou velice abstraktní a v případě dat z akcelerometru jsou takřka prázdné. Jediná část mého projektu, která nemá implementaci těchto zdrojových souborů prázdnou, je část starající se o ukládání a zprostředkování dat o poloze, GPSDataService. V zdrojovém souboru GPSDataService je implementace metody pro filtrování GPS dat. A v souboru IGPSDataRepository je deklarace této metody.

Kód 6.8: Deklarace zmiňované filtrovací metody

```
public IEnumerable<GPSData> FindAllInTimeRange
            (int OBUID, int timeFrom, int timeTo)
```

Metoda FindAllInTimeRange v případě GPSDataService vrací pro danou vozidlovou jednotku veškeré záznamy o její poloze ve vymezeném časovém úseku.

6.4 Testování data serveru

Veškeré testování jsem prováděl já sám. Jelikož jsem server psal já, tak vím, jaké má nedostatky a jaká mezní data by mohla způsobovat problémy. Výhodou je, že většinu ošetření mezních a nevalidních dat za mě provádí přímo ServiceStack - pokud mu pošlu nevalidní data, tak je ignoruje.

Jelikož jsem v době testování serveru neměl k dispozici funkční vozidlovou jednotku, musel jsem chování a posílání dat z vozidlové jednotky nasimulovat. K tomu mi posloužil plugin Postman [12] do prohlížeče Google Chrome, který

umožňuje komunikaci přes HTTP přes rozhraní REST. Do tohoto pluginu jsem zadával různá data a zkoušel, jestli se server zachová tak, jak má.

Během testování jsem nenarazil na žádný zásadnější problém.

6.5 Shrnutí

Implementace mi zabrala opravdu velmi mnoho času, což bylo hlavně způsobeno tím, že jsem doposud nikdy nepracoval s EntityFrameworkem ani ServiceStackem. S programovacím jazykem C# a platformou .NET jsem již nějaké zkušenosti měl. Nakonec se mi ale podařilo implementovat vše, co jsem chtěl a co jsem si na začátku práce stanovil. Naučil jsem se mnoho nových věcí, programátorských technik a také ovládat EntityFramework a ServiceStack.

Závěr

Míra naplnění cíle

Výsledkem mé práce je podrobný návrh systému na komplexní monitorování pohybu a technického stavu vozidel. Jedná se o systém nabízející širokou škálu funkcí. Od elektronické knihy jízd, až po sledování úbytku oleje nebo chladící kapaliny v motoru. Dále v rámci mé bakalářské práce vznikl prototyp serveru psaný v jazyce C#.

Původním záměrem bylo navrhnout monitorovací systém, který bude nabízet velké množství funkcí anebo systém, který se zaměří na jednu funkcionalitu, avšak velmi podrobně. Současně bylo úkolem implementovat prototyp serveru, který bude sloužit jako předloha pro další rozvoj systému. V rámci mé bakalářské práce takový návrh na systém a prototyp serveru vznikl. Cíl mé bakalářské práce byl tedy naplněn.

Výhled do budoucna

Do budoucna plánuji s kolegy z firmy Telematix implementovat systém navržený v této bakalářské práci a tento systém uvést do provozu. Dále plánuji přidat do systému více funkcí.

Osobní přínos

Za hlavní osobní přínos považuji to, že jsem se zdokonalil v programování v jazyce C# a v práci s platformou .NET. Naučil jsem se také pracovat s Entity Frameworkem a ServiceStackem, oba tyto frameworky mě velmi zaujaly a určitě je v budoucnu použiji v dalších projektech. Rovněž pro mě byla velkým přínosem možnost pracovat na tak rozsáhlém projektu. Při práci jsem se setkával s reálnými problémy, které jsem se pokoušel řešit. Moje bakalářská

práce mi tedy zcela jistě rozšířila obzory v oblasti softwarového inženýrství, za což jsem velmi rád.

Literatura

[1] WWW Consorcium: *Sherlog Secar Bohemia group [online]*. [cit. 2015-06-02]. Dostupné z: `http://www.sherlog.cz`

[2] WWW Consorcium: *Monitoring vozidel CSS Carnet [online]*. [cit. 2015-06-02]. Dostupné z: `http://www.ccs.cz`

[3] WWW Consorcium: *Car Monitor [online]*. [cit. 2015-06-02]. Dostupné z: `http://www.car-monitor.cz`

[4] WWW Consorcium: *Lokátory, sledování vozidel a kniha jízd [online]*. [cit. 2015-05-02]. Dostupné z: `http://www.lokatory.cz`

[5] WWW Consorcium: *Automatic [online]*. [cit. 2015-05-02]. Dostupné z: `https://www.automatic.com`

[6] WWW Consorcium: *Výuka Greendot [online]*. [cit. 2015-05-05]. Dostupné z: `http://vyuka.greendot.cz/materialy/material-4.pdf`

[7] WWW Consorcium: *Internetová encyklopedie Wikipedia článek o MySQL [online]*. [cit. 2015-05-05]. Dostupné z: `http://cs.wikipedia.org/wiki/MySQL`

[8] WWW Consorcium: *Internetová encyklopedie Wikipedia článek o SQLite [online]*. [cit. 2015-05-05]. Dostupné z: `http://cs.wikipedia.org/wiki/SQLite`

[9] WWW Consorcium: *Internetová encyklopedie Wikipedia článek o MS SQL [online]*. [cit. 2015-05-05]. Dostupné z: `http://cs.wikipedia.org/wiki/Microsoft_SQL_Server`

[10] WWW Consorcium: *Oficiální webové stránky databáze Cassandra [online]*. [cit. 2015-05-05]. Dostupné z: `http://cassandra.apache.org`

[11] WWW Consorcium: *Internetová encyklopedie Wikipedia článek o databázi MongoDB [online]*. [cit. 2015-05-05]. Dostupné z: `http://en.wikipedia.org/wiki/MongoDB`

[12] WWW Consorcium: *Postman - REST Client [online]*. [cit. 2015-05-06]. Dostupné z: `https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm`

Příloha **A**

Seznam použitých zkratek

GPS Geo Positioning System

GSM Global System for Mobile

SSD Solid State Drive

SQL Structured Query Language

NoSQL Not only Structures Query Language

GPL General Public License

OBU On Board Unit

VIN Vehicle Identification Number

HTTP HyperText Transfer Protocol

API Application Programming Interface

WCF Windows Communication Foundation

MVC Model View Controller

JSON JavaScript Object Notation

ORM Object Relation Mapping

DTO Data Transfer Object

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe`
  - `OBUDataServer.zip` ..... balíček k nahrání na server
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `Solution` ..... adresář s Visual Studio projektem
    - `Sources` ..... adresář se zdrojovými kódy k projektu
    - `GPSToBase64Converter.cs` ..... konvertor GPS dat
  - `thesis` ..... zdrojová forma práce ve formátu LateX
- `text`
  - `BP_Texler_Ondřej_2015.pdf` ..... text práce ve formátu PDF
- `diagrams`
  - `DatabaseDiagram.eap` .... databázový diagram ve zdrojovém formátu
  - `DatabaseDiagram.png` ..... databázový diagram ve formátu PNG
  - `DeploymentDiagram.eap` ..... diagram nasazení ve zdrojovém formátu
  - `DeploymentDiagram.png` ..... diagram nasazení ve formátu PNG