Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Riadenie projektu

Dokumentácia k tímovému projektu

Tímový projekt

Tím č. 21

Vedúci: Ing. Ivan Srba, PhD.

tim21.2018.fiit@gmail.com

Obsah

1 Úvod

Tento dokument opisuje postupy spojené s riadením tímu Traffic Watch v rámci predmetu Tímový projekt. V nasledujúcich kapitolách sa detailne venuje členom tímu a rolám ktoré napĺňali, aplikácií manažmentov, sumarizácií šprintov a globálnej retrospektíve z dokončených šprintov.

2 Role členov a podiel práce na dokumentácií

2.1 Role členov tímu

Meno	Rola
Adam Kňaze	Počítačové videnie a algoritmy
Jozef Kamenský	Full-stack developer
Matej Horváth	Computer vision developer
Kristína Macková	Front-end a back-end web developer Manažér komunikácie
Jakub Sedlář	Back-end developer Manažér testovania a verziovania
Lenka Pejchalová	Grafik, Front-end developer
Matej Groma	Manažér nasadzovania
Peter Jurkáček	Scrum master Manažér riadenia

2.2 Podiel práce na dokumentácií

Meno	Percentuálny podiel
Adam Kňaze	5.44
Jozef Kamenský	18
Matej Horváth	11
Kristína Macková	13
Jakub Sedlář	9.5
Lenka Pejchalová	9.5
Matej Groma	12,56
Peter Jurkáček	15

3 Aplikácie manažmentov

3.1 Manažment úloh

V našom tíme pri riadení úloh používame systém Jira. Jednotlivé úlohy sa evidujú ako issues. Issues evidujeme na troch úrovniach granularity (Epic, Story, Sub-task).

Scrum master vytvára Epicy a Storky a členovia tímu si k nim vytvárajú Sub-tasky. Každý člen tímu si zaznamenáva čas strávený prácou na tímovom projekte na príslušné Sub-tasky. Čo nie je v Jire to sa nerobí.

Storky a sub-tasky majú definované definition of done. Storky sú akceptované produktovými vlastníkmi a sub-tasky sú akceptované členmi tímu.

Všetky procesy sojené s manažmentom úloh sú opísané v Metodike úloh.

3.2 Manažment komunikácie

Komunikácia je dôležitá súčasť fungovania tímu. Najpoužívanejším komunikačným kanálom tímu je Slack, ktorý je organizovaný do kanálov podľa tém. Tieto kanály slúžia na formálnu aj neformálnu komunikáciu medzi členmi tímu, komunikáciu s vedúcim tímového projektu a product ownermi. To, aký obsah sa rieši v akých kanáloch, je definované v metodike komunikácie. Postup pri komunikácií a zdielaní obsahu s product ownermi je tiež vyhradený v tomto dokumente.

Do Slacku sme integrovali Jiru, Github a Trevis, takže každý člen tímu dostáva notifikácie o zmenách na projekte.

Na zdieľanie dokumentov v tíme používame výhradne Confluence. Na Confluence tiež zbierame všetky intelektuálne výstupy ktoré vznikli v rámci tímového projektu.

3.3 Automatizácia nasadzovania

Produkčne použiteľné nasadenie je zabezpečené pomocou Ansible playbooku - týmto spôsobom je aj štandardne dokumentovaný celkový postup nasadenia (požadované balíčky, konfigurácia systému, databázy a pod.), keďže funguje aj na úplne čistom stroji. Nasadenie funguje na Linuxe, predpokladom je nainštalované Ansible na klientovi z ktorého nasadenie vykonávame (napr. apt install ansible) a funkčný stroj, na ktorý chceme aplikáciu (backend a frontend) nasadiť (otestované na Ubuntu 18.04, malo by fungovať aj na inej distribúcií, prípadne po menších úpravách). Nasadená je verzia v špecifikovanej vetve, ktorej repozitár je automaticky naklonovaný. Automatizované je:

- nastavenie základných konfiguračných parametrov OS
- nainštalovanie web servera nginx pre FE, jeho konfigurácia
- vyžiadanie certifikátu cez Letsencrypt
- nainštalovanie a nakonfigurovanie PostgreSQL a TimescaleDB
- nainštalovanie a nakonfigurovanie MQTT servera
- nakonfigurovanie, zostavenie a nasadenie backendu
- nakonfigurovanie, zostavenie a nasadenie frontendu

V súbore hosts pod [servers] nakonfigurujeme FQDN servera, na ktorý chceme aplikáciu nasadiť. Parametrom ansible_user špecifikujeme meno používateľa, prostredníctvom ktorého sa na server pripájame (root alebo používateľ so sudo oprávneniami). Pod [servers:vars] môžeme upraviť niektoré konfiguračné parametre servera (napr. porty, na ktorých BE/FE bežia). Pod [all:vars] upravujeme nastavenia, ktoré sa aplikujú lokálne

(ako cesta k repozitáru, alebo vetva ktorú nasadiť). Následne skopírujeme súbor host_vars/example.com do host_vars/FQDN a definujeme používateľské mená a heslá ktoré chceme použiť pre MQTT (kvôli citlivosti obsahu takéto súbory nie sú verziované).

Nasadenie spustíme cez ./play.sh (stiahne podporné roly a vykoná samotný playbook). Po vykonaní aplikácia beží na serveri bez potreby ďalšieho manuálneho zásahu. Na školskom serveri aplikácia beží na porte 8123 (backend) a 8124 (frontend).

Menej závažným problémom bolo, že Java využíva vlastný systém na infraštruktúru verejného kľúča – potrebné je prekonvertovať používaný certifikát do keystore. Z tohto dôvodu bol najskôr nasadený backend nezabezpečene. Frontend musel byť nasadený nezabezpečene tiež, kvôli zakázanému načítavaniu mixed content na zabezpečenej stránke v súčasných prehliadačoch.

V súvislosti s nasadzovaním existuje metodika nasadzovania, ktorou sa v rámci tímu riadime za účelom zabezpečenia bezproblémového chodu procesu.

3.4 Manažment písania kódu

Systém, ktorý v rámci tímového projektu vyvíjame sa skladá z viacerých častí, pričom každá časť je písaná v inom programovacom jazyku. Preto sme potrebovali v pomerne krátkom čase zaviesť pravidlá písania kódu tak, aby bol kód jednotný a ľahšie pochopiteľný.

Ako základ pravidiel pre písanie kódu používame dostupné pravidlá písania kódu zavedené veľkými spoločnosťami. Pre kód v jazyku JAVA používame pravidlá písania kódu od Googlu, Google JAVA Style Guide. Pri frontendovom vývoji používame príručku Airbnb React/JSX Style Guide. Obidve tieto príručky sú dosť rozsiahle a podrobne popisujú všetky aspekty spojené s formátovaním či štruktúrovaním kódu.

V rámci tímu máme zavedené používanie rôznych nástrojov na kontrolu kódu, ktoré sa využívajú pri tvorbe čiastkových riešení alebo skriptov.

3.5 Manažment verziovania

Na verziovanie sme použili nástroj git a služby bitbucket a github. Na všetky moduly projektu je vytvorený samostatný repozitár, a všetci členovia tímu pracovali s týmito nástrojmi. Pravidlá a postupy vytvárania vetiev, vytvárania pull requestov a číslovaniu verzií je opísaný v metodike verziovania.

4 Sumarizácie šprintov

4.1 Šprint Alomomola

Cieľ: Stretnúť sa partnermi projektu, configurácia serverov a IDEs.

Trvanie šprintu: 05.10.2018 - 19.10.2018

Odpracovaný čas: 115h 38m

Názov story	Podúlohy	Počet nedokončených podúloh	Počet dokončených podúloh	Commitnuté Story pointy	Dodané Story pointy	Stav
Analyzovanie a prototypovanie spracovania obrazu	TP-27, TP-28, TP-29,TP-30, TP-31, TP-32	0	2	13	13	Accepted
Analyzovanie hardwaru	TP-8	0	0	5	5	Accepted
Vytvorenie 1.návrhu architektúry	TP-45, TP-46, TP-47,TP-48, TP-49	0	3	5	5	Accepted
Analyzovanie mapových služieb	TP-33, TP-34, TP-35	0	0	3	3	Accepted
Analyzovanie CI atlassianu		0	0	2	2	Accepted
Vytváranie 1. verzie projektových metodík	TP-14, TP-36, TP-43	0	5	0	0	Accepted
Vytvorenie zápisníc		0	3	0	0	Accepted
Prihlásenie do súťaže TP CUP	TP-55, TP-56	0	6	0	0	Accepted
Vytvorenie statickej web stránky	TP-39, TP-40, TP-41	0	1	0	0	Accepted
Total	20	0	20	28	28	
Velocity	28					
Počet úloh	9					

Tabuľka 1: Vyhodnotenie šprintu Alomomola

Člen tímu	Pridaná hodnota	**Pridaná hodnota (%)**
Matej Groma	18	**13%**
Matej Hortváth	15	**11%**
Peter Jurkáček	15	**11%**
Kristína Macková	16	**12%**
Lenka Pejchalová	16	**12%**
Jozef Kamensky	24	**18%**
Adam Kňaze	18	**13%**
Jakub Sedlář	15	**11%**
Total	137	100%

Tabuľka 2: Pridaná hodnota na projekte členov tímu za šprint Alomomola

4.2 Šprint Bulbasaur

Cieľ: Rozbehaný Full stack (BE, FE, Kamera klient)

Trvanie šprintu: 19.10.2018 - 09.11.2018

Odpracovaný čas: 124h 40m

Názov story	Podúlohy	Počet nedokončených podúloh	Počet dokončených podúloh	Commitnuté Story pointy	Dodané Story pointy	Stav
Aplikovanie algoritmu KNN z OpenCV	TP-119, TP-120, TP-121,TP-134	0	5	13	13	Accepted
Detekovanie prechodu cez zónu	TP-142	0	1	8	8	Accepted
Vytvorenie BE Kostry	TP-100, TP-110, TP-116,TP-117	0	4	8	8	Accepted
Vytvorenie prototypu pre MQTT komunikáciu	TP-111, TP-112, TP-114,TP-115, TP-122, TP-123,TP-141	0	7	5	5	Accepted
Zobrazenie údajov o kamere na FE	TP-102, TP-103, TP-104,TP-105, TP-106, TP-107,TP-137	0	7	5	5	Accepted
Vytvorenie FE Kostry	TP-130, TP-131, TP-132,TP-143	0	4	5	5	Accepted
Analýzovanie FE technológií		0	0	1	1	Accepted
Dummy Bulbasaur	TP-101, TP-108, TP-109,TP-125, TP-126, TP-127,TP-128, TP-129, TP-136,TP-138, TP-144	0	11	0	0	Accepted
Vytváranie 2. verzie projektových metodík	TP-63, TP-64, TP-65, TP-99	0	4	0	0	Accepted
Total	43	0	43	45	45	
Velocity	45					
Počet úloh	9					

Tabuľka 3: Vyhodnotenie šprintu Bulbasaur

Člen tímu	Pridaná hodnota	**Pridaná hodnota (%)**
Matej Groma	18	**13%**
Matej Hortváth	17	**13%**
Peter Jurkáček	14	**10%**
Kristína Macková	18	**13%**
Lenka Pejchalová	15	**11%**
Jozef Kamensky	20	**15%**
Adam Kňaze	19	**14%**
Jakub Sedlář	13	**10%**
Total	134	100%

Tabuľka 4: Pridaná hodnota na projekte členov tímu za šprint Bulbasaur

4.3 Šprint Caterpie

Cieľ: Zobrazenie počtu prejazdov zo sledovanej oblasti na stránke na základe generovaných udalostí z kamery.

Trvanie šprintu: 9.11.2018 - 23.11.2018

Odpracovaný čas: 220h

Názov story	Podúlohy	Počet nedokončených podúloh	Počet dokončených podúloh	Commitnuté Story pointy	Dodané Story pointy	Stav
Uloženie prejazdu a aktualizovanie štatistík	TP-175, TP-187, TP-188, TP-189, TP-190,TP-194, TP-197, TP-202, TP-203, TP-204, TP-206, TP-207, TP-208, TP-209,TP-211	0	15	8	8	Accepted
Konfigurovanie parametrov sledovanej oblasti pre detekciu objektu	TP-172, TP-174, TP-179, TP-180, TP-184,TP-195	0	6	8	8	Accepted
Generovanie eventu pre prejazd	TP-169, TP-171, TP-193	0	3	8	8	Accepted
Zobrazenie tabuľky prejazdov	TP-161, TP-164, TP-166, TP-167, TP-168,TP-185, TP-186	1	6	5	2	Rejected
Prijímanie a aktualizovanie parametrov sledovanej oblasti pre detekciu objektu	TP-102, TP-103, TP-104,TP-105, TP-106, TP-107,TP-137	0	7	3	3	Accepted
Automatizovanie nasadení a testovaní	TP-173, TP-176, TP-178, TP-238	0	4	0	0	Accepted
Dummy Caterpie	TP-157, TP-158, TP-170, TP-181, TP-182,TP-196, TP-198, TP-199, TP-200, TP-201,TP-205, TP-210, TP-212, TP-235, TP-236, TP-237	0	16	0	0	Accepted
Total	58	1	57	32	29	
Velocity	29					
Počet úloh	7					

Tabuľka 5: Vyhodnotenie šprintu Caterpie

Člen tímu	Pridaná hodnota	**Pridaná hodnota (%)**
Matej Groma	17	**13%**
Matej Hortváth	13	**10%**
Peter Jurkáček	14	**11%**
Kristína Macková	18	**14%**
Lenka Pejchalová	15	**11%**
Jozef Kamensky	23	**18%**
Adam Kňaze	17	**13%**
Jakub Sedlář	14	**11%**
Total	131	100%

Tabuľka 6: Pridaná hodnota na projekte členov tímu za šprint Caterpie

5 Globálna retrospektíva

5.1 Zimný semester

Členovia nášho tímu sa pred začiatkom projektu nepoznali a aj napriek tomu pracovali ako skutočný tím. Každý člen tímu prispel svojimi vedomosťami pri inicializácií projektu.

Počas šprintov sme sa potýkali s problémami, na ktoré sme poukázali počas retrospekívy v každom šprinte. Tieto problémy sme sa snažili vyriešiť v nasledujúcich šprintoch. Medzi hlavné problémy považujeme komunikáciu, rôznu časovú dostupnosť jednotlivých členov tímu a nové technológie.

Musíme poznamenať, že na konci tretieho šprintu sa nám podarilo vytvoriť infraštruktúru systému, ktorý dokáže monitorovať zaznamenávať počty aút, ktoré prešli cez vyznačené oblasti na videu a zobrazovať k nim štatistky prostredníctvom webového rozhrania.

5.1.1 Čo robíme dobre?

- Vytvárame si program na každé stretnutie.
- Snažíme sa riešiť problémy osobne na tímových stretnutiach.
- Máme povedomie o pocitoch tímu vďaka online dotazníku.
- Máme poriadok v gite.

5.1.2 V čom sa snažíme zlepšiť?

- Pravidelnejšie aktualizovať dokumenty na stránke tímu.
- Najprv vyvíjať Api a mockovať dáta
- Lepšie zadefinovať zodpovednosti členov timu.
- Skúsiť pair programming.

A Motivačný list

Predstavenie tímu č.21

Nas tim sa sklada z 8 studentov, ktori na FIIT absolvovali aj bakalarske studium a momentalne studuju jeden zo studijnych programov ISS alebo IT.

V oblasti tvorby backendu ma kazdy clen timu zakladne skusenosti s programovacim jazykom Java, vacsie skusenosti s nim maju Jozef Kamensky, Peter Jurkacek a Matej Horvath, ktori ich ziskali v ramci staze alebo zamestnania. Dalsimi jazykom, s ktorym dokaze nas tim pracovat je C (Kristina Mackova, Jakub Sedlar, Lenka Pejchalova, Matej Horvath). V pripade spravy Linux serverov sa mozeme spolahnut na Mateja Gromu.

Pri ukladani dat sa kazdy clen timu stretol s relacnymi databazami PostgreSQL a MySQL. Peter Jurkacek pri mobilnom vyvoji casto pracuje s databazami SQLite alebo Realm. Z nerelacnych databaz tim pozna a vie pouzivat Redis (Matej Horvath) ci grafovu databazu Neo4j (Jozef Kamensky).

Pri spracovavani dat vieme vyuzit programovaci jazyk Python spolu s kniznicami na analyzu dat NumPy, SciPy a scikit-learn (Peter Jurkacek, Adam Knaze, Jozef Kamensky). Adam Knaze sa navyse v ramci bakalarskej prace venoval hlbokym neuronovym sietam a spracovaniu textu, pricom pouzival kniznicu Tensorflow.

Pri tvorbe frontendu sa mozeme opriet predovsetkym o Lenku Pejchalovu, ktora sa zaobera dizajnom pouzivatelskych rozhrani. Pri neskorsom tvoreni webovych stranok vie pomoct kazdy clen timu - kazdy ovlada HTML, CSS a zaklady Javascriptu. Ti pokrocilejsi ovladaju tiez TypeScript (Lenka Pejchalova) a rozne Javascriptove frameworky, napr. Angular2+ alebo Polymer (Jozef Kamensky).

Na vyvoj mobilnych aplikacii sa sustredi Peter Jurkacek, ktory je schopny vytvarat nativne aplikacie pre platformy Android a iOS. S vyvojom na platforme Android mu pomahaju Jakub Sedlar a Matej Horvath.

Medzi clenov nasho timu so specifickymi schopnostami patri Kristina Mackova, ktora sa sa zaobera vyvojom pocitacovych hier a virtualnou realitou. Tieto schopnosti ziskala a rozvijala v ramci bakalarskej prace ci priemyselnej staze. Navyse sa zucastnila Letnej skoly tvorby hier Summer Game Dev 2018.

Takisto treba spomenut Mateja Gromu, ktory sa venuje digitalnej elektronike a vnorenym systemom. Svoje zrucnosti zdokonalil pri bakalarskej praci, ktora sa venovala teme IoT. Spolu s Adamom Knaze ovladaju jazyk C++.

Clenovia timu svoje vedomosti a prakticke zrucnosti neustale rozsiruju – pracuju alebo sa zucastnuju stazi, napr. v ramci predmetu Priemyselna staz – 3 z 8 stazistov su v nasom time. Myslime, ze to dostatocne znazornuje ochotu ucit sa nove veci a celit novym vyzvam. Tymto sposobom tiez ziskavame skusenosti s pracou v time.

Motivácia 1: Analyza spravania sa vozidiel v meste [SmartMobility]

Tato tema nas tim zaujala hlavne vyuzitim mnohych modernych technologii, o ktore sa nas tim aktivne zaujima, a spaja ich zaujimavym a pre prax vyuzitelnym sposobom. Nas tim ma schopnosti aktivne prispiet ku vsetkym castiam riesenia daneho problemu. Matej Groma sa dokaze venovat produktu po hardverovej stranke, o spracovanie dat a strojove ucenie v Pythone sa zaujimaju viaceri clenovia timu. Spracovane data je nutne zobrazit vhodnym sposobom. Nas tim je schopny tvorit moderne webove rozhranie aj nativne mobilne aplikacie. Jedinym ciastocnym nedostatkom je, ze nas tim ma zatial male skusenosti so spracovavanim obrazu. Myslime, ze tento nedostatok dokazeme lahko prekonat absolvovanim predmetov pocas semestra (SOGAM) a dodatocnym studiom problemovej oblasti.

Motivácia 2: Podpora výskumu behaviorálnej biometrie [Behametrics - learn]

Behavioralna biometria sa v sucasnosti dostava do popredia. Prebieha vyskum novych moznosti jej vyuzitia, ktore ma nas tim zaujem podporit. V ramci timu sme schopni pracovat a zaznamenavat data z roznych

platforiem – web, smartfony (Android aj iOS) a virtualna ci rozsirena realita (senzor Leap Motion). Tiez mame skusenosti s analyzou dat – hlavne v Pythone s pouzitim kniznic na analyzu dat (NumPy, SciPy, scikit-learn). Skusenosti s tvorbou webovych a mobilnych aplikacii mozeme vyuzit pri tvorbe grafickeho rozhrania, ktore umozni lahky pristup, filtrovanie a vizualizaciu ziskanych a spracovanych udajov.

Motivácia 3: Prostredie pre inteligentnú analýzu textov [TextEnv]

V oblasti spracovavania textu ma v ramci timu najvacsie skusenosti Adam Knaze, ktory sa venuje spracovaniu textu pomocou hlbokych neuronovych sieti (kniznica Tensorflow).. Tiez mame skusenosti s analyzou dat - hlavne v Pythone s pouzitim kniznic na analyzu dat (NumPy, SciPy, scikit-learn). Skusenosti s tvorbou webovych a mobilnych aplikacii mozeme vyuzit pri tvorbe grafickeho rozhrania, ktore umozni pouzivatelovi lahko pracovat s databazou text a prehladnym sposobom vizualizuje dolezite informacie.

Príloha A: Zoradenie tém podľa priority

1. **Analyza spravania sa vozidiel v meste [SmartMobility]**
2. **Podpora vyskumu behavioralnej biometrie [behametrics-learn]**
3. **Prostredie pre inteligentnu analyzu textov [TxtEnv]**
4. **Identifikacia entit – spracovanie textu [SK-CZ-TEXT]**
5. Vnimanie neviditelneho [Holographic Eyes]
6. Monitoring antisocialneho spravania [MonAnt]
7. Inteligentny importer verejnych dat [Importer]
8. IoT system monitorovania osob [Breyslet]
9. Prostredie na vizualizaciu mikrogridu [GridBox]
10. Skola hrou vo virtualnej realite [VREducation]
11. Vyhladavanie pomocou obrazkov [ImageSearch]
12. Monitorovanie a vyhodnocovanie fyziologickych procesov cloveka [BioMonitor] 13.Generator 3D priestoru [3DSpaceGen]
13. Databanka otazok a uloh [FIIT - DU]
14. 3D simulovany roboticky futbal [3D futbal]
15. Automaticke testovanie v prostredi Internetu veci [IoTesting]
16. Vizualizacia softveru vo virtualnej a rozsirenej realite (Remake) [VizReal] 18.WiFi Funtoro [WFuntoro]
17. In-memory databaza s vyuzitim GPU [In-memory-DB]
18. 3D UML, improved version [3D-UML]

Rozvrh členov tímu

Pondelok	**07–08**	**08–09**	**09–10**	**10–11**	**11–12**	**12–13**	**13–14**	**14–15**	**15–16**	**16–17**	**17–18**	**18–19**	**19–20**	**20–21**
Kristina Mackova														
Adam Kňaze						SJ				SJ				
Jakub Sedlář														
Jozef Kamenský														
Lenka Pejchalová		VINF												
Peťo Jurkáček										IPIKT		IPIKT		
Matej Horváth				KSS										
Matej Groma														
Voľný čas								Spoločná práca						
Utorok	**07–08**	**08–09**	**09–10**	**10–11**	**11–12**	**12–13**	**13–14**	**14–15**	**15–16**	**16–17**	**17–18**	**18–19**	**19–20**	**20–21**
Kristina Mackova								ASS		VISS	TP1		ASS	
Adam Kňaze														
Jakub Sedlář														
Jozef Kamenský					SOGAM									
Lenka Pejchalová			ZKGRA										ASS	
Peťo Jurkáček														
Matej Horváth									VSPI				APS	
Matej Groma														
Voľný čas														
Streda	**07–08**	**08–09**	**09–10**	**10–11**	**11–12**	**12–13**	**13–14**	**14–15**	**15–16**	**16–17**	**17–18**	**18–19**	**19–20**	**20–21**
Kristina Mackova									MTS		MTS (každý druhý týždeň)			
Adam Kňaze		NSIETE												
Jakub Sedlář														
Jozef Kamenský														
Lenka Pejchalová				ZKGRA										
Peťo Jurkáček														
Matej Horváth							DOVI		APS		DOVI			
Matej Groma														
Voľný čas														
Štvrtok	**07–08**	**08–09**	**09–10**	**10–11**	**11–12**	**12–13**	**13–14**	**14–15**	**15–16**	**16–17**	**17–18**	**18–19**	**19–20**	**20–21**
Kristina Mackova				PS					NAVPH		NAVPH			
Adam Kňaze			ASS		NSIETE									
Jakub Sedlář					ASS									
Jozef Kamenský				PS					SOGAM		ASS			
Lenka Pejchalová							VINF							
Peťo Jurkáček														
Matej Horváth		KSS												
Matej Groma						KSS			VNOS				VNOS	
Voľný čas														
Piatok	**07–08**	**08–09**	**09–10**	**10–11**	**11–12**	**12–13**	**13–14**	**14–15**	**15–16**	**16–17**	**17–18**	**18–19**	**19–20**	**20–21**
Kristina Mackova														
Adam Kňaze														
Jakub Sedlář														
Jozef Kamenský														
Lenka Pejchalová														
Peťo Jurkáček														
Matej Horváth														
Matej Groma														
Voľný čas		Spoločná práca / stretnutie s vedúcim												

Legenda	Prednáška	Cvičenie	Voľný čas

Obr. 1: Spoločný rozvrh členov tímu

B Prihláška na TP CUP 2018

Predstavenie tímu

Nas tim sa sklada z 8 studentov, ktori na FIIT absolvovali bakalarske studium. Šiesti členovia nášho tímu študujú odbor ISS a dvaja odbor IT. Clenovia timu majú vedomosti a prakticke zrucnosti, či už z absolvovaných stazi alebo zamestnania. Každý člen nášho tímu má skúsenosti s vývojom webových aplikácií, či už v rámci Back-endu alebo Front-endu.

1. Matej Groma - Špecialista na hardware a správu serverov. Má užitočné skúsenosti z bakalárskej práce, ktorá bola z oblasti IoT.
2. Matej Horváth - Back-end programátor. Zaujíma sa o počítačové videnie. Má skúsenosti so sieťami, a v práci sa zaoberá implementáciou projektov a DevOps.
3. Peter Jurkáček - Scrum master, bol sám zvolený pri demokratickom hlasovaní. Má skúsenosti s metodikou scrumu, plánovaním, analýzou dát a vývojom mobilných aplikácií.
4. Jozef Kamenský - Softvérový architekt. Má užitočné skúsenosti z návrhu a implementácie projektov zo stáže a z práce, kde pracuje ako full-stack developer.
5. Adam Kňaze - Špecialista na počítačové videnie a algoritmy. Má bohaté skúsenosti z bakalárskeho projektu s analýzou údajov a spracovaním dát.
6. Kristína Macková - Špecialista na počítačovú grafiku. Vo voľnom čase sa venuje vývoju hier. Absolvovala veľa stáží. Má bohaté skúsenosti s VR, AR, tvorbou mobilných, či webových aplikácií.
7. Lenka Pejchalová - Grafický dizajnér. Stojí za logom a stránkou nášho tímu. Vo voľnom čase sa venuje práci vo Photoshop-e a má bohaté pracovné skúsenosti v oblasti Front-endu.
8. Jakub Sedlář - Git master. Dohliada na správny gitflow nášho tímu. Má užitočné skúsenosti zo stáže, z analýzy údajov a vývoja mobilných aplikácií.

Motivácia

Na projekte spolupracujeme so spoločnosťami Orange a Unicorn a rámci 12. ročníka súťaže TP Cup by sme chceli preukázať svoje schopnosti tvorbou komplexného softvérového systému, ktorý rieši problematiku z oblasti SmartCity so zameraním na dopravu. TP Cup pre nás predstavuje výzvu a motiváciu na vytvorenie použiteľného a užitočného riešenia, ktoré pomôže zlepšiť dopravu.

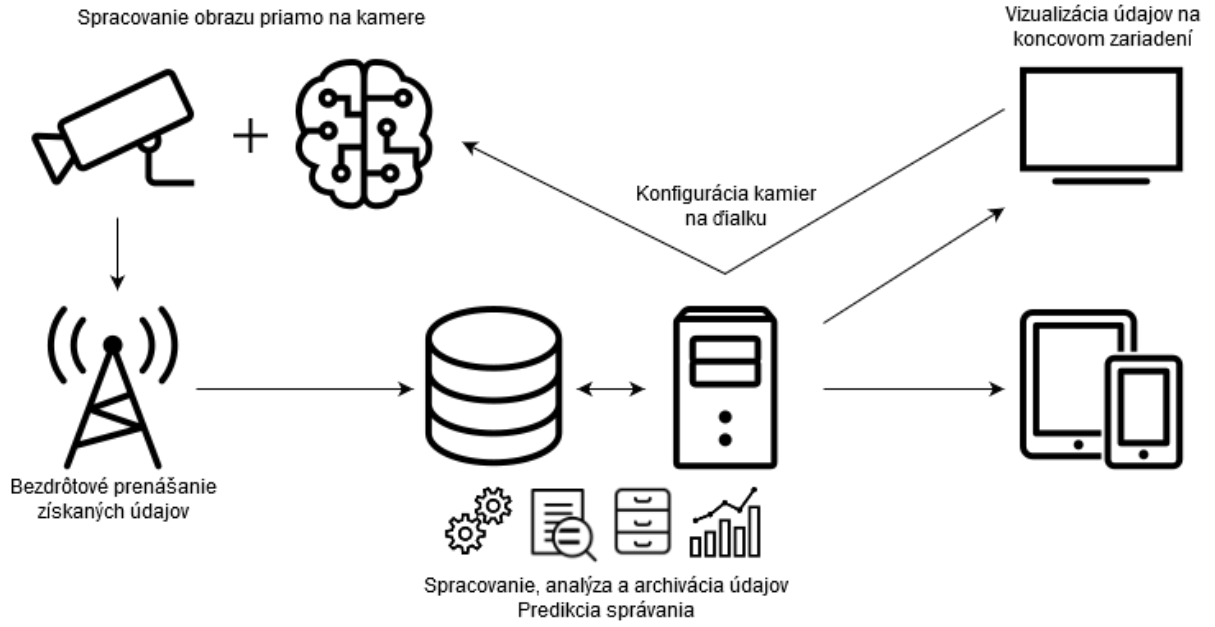
Čomu sa venujeme?

Pomocou kamier zaznamenávame a vyhodnocujeme informacie o spravani sa vozidiel, cyklistov a chodcov v sledovanom useku. Našim cieľom je optimalizovat dopravu napriklad sledovaním vývoja dopravy v rámci jednej križovatky alebo identifikaciou „busy hours“ na zaklade dlzky statia vozidiel. Vyhodnocovaním informácií z viacerých kamier chceme sledovať vývoj dopravy v rámci určených oblastí či postupné plnenie sa mestských častí vozidlami.

Kedy a ako sa to dá použiť?

V prípade, ak mesto plánuje odklon dopravy alebo prestavbu cestných komunikácií, vie použiť naše informácie o vyťaženosti jednotlivých križovatiek aby zvolilo to najoptimálnejšie riešenie na preznačenie hlavných a vedľajších ciest. V prípade, ak rozmýšľate nad kúpou alebo prenájmom nehnuteľnosti a zaujíma vás rentabilita obytných alebo firemných priestorov z pohľadu dopravnej situácie, viete použiť naše riešenie na sledovanie dopravy vo vybraných oblastiach.

Ako to dosiahneme ?



Obr. 2: Big picture navrhovaného riešenia

Kamera

Každá použitá kamera bude vybavená modulom, na ktorom bude prebiehať spracovanie obrazu a následné posielanie udalostí ďalej na server. Pri spracovaní obrazu na kamere použijeme kombináciu technológií OpenCV a Tensorflow, pričom OpenCV sa primárne zameriava na prvotné spracovanie obrazu a Tensorflow na následné rozpoznávanie jednotlivých vozidiel, bicyklov či chodcov a zaznamenávanie ich pohybu. Od takéhoto modelu si sľubujeme zvýšenú mieru presnosti pri určovaní výskytu a pohybu vozidiel a tiež zmenšenie výpočtových nárokov. Vzniknuté udalosti sa budú posielať cez RESTové služby priamo na náš server alebo sa môžu využiť už existujúce platformy na ukladanie údajov z IoT sietí, napr. Live Objects od Orange-u. Naše riešenie počíta tiež s konfiguráciou kamier na diaľku, napríklad na zmenu parametrov potrebných pre korektné spracovanie obrazu, napr. výška a poloha kamery či úprava parametrov pre spracovanie obrazu v závislosti od prostredia.

Ukladanie a spracovanie dát

Keďže kamery generujú veľké množstvo udalostí, bolo potrebné vybrať vhodnú databázu. Z počiatočnej analýzy a porovnávania rôznych používaných databáz sme sa rozhodli použiť databázu TimescaleDB. Táto databáza vznikla ako rozšírenie databázy PostgreSQL za účelom uchovávania dát z IoT zariadení a efektívnu prácu s časovými pečiatkami.

Server

Na základe analýzy a skúseností členov nášho tímu sme sa rozhodli použiť jazyk JAVA spolu s frameworkom Spring. Server sme rozdelili na 4 primárne komponenty:

1. komponent komunikujúci s rozličnými databázami

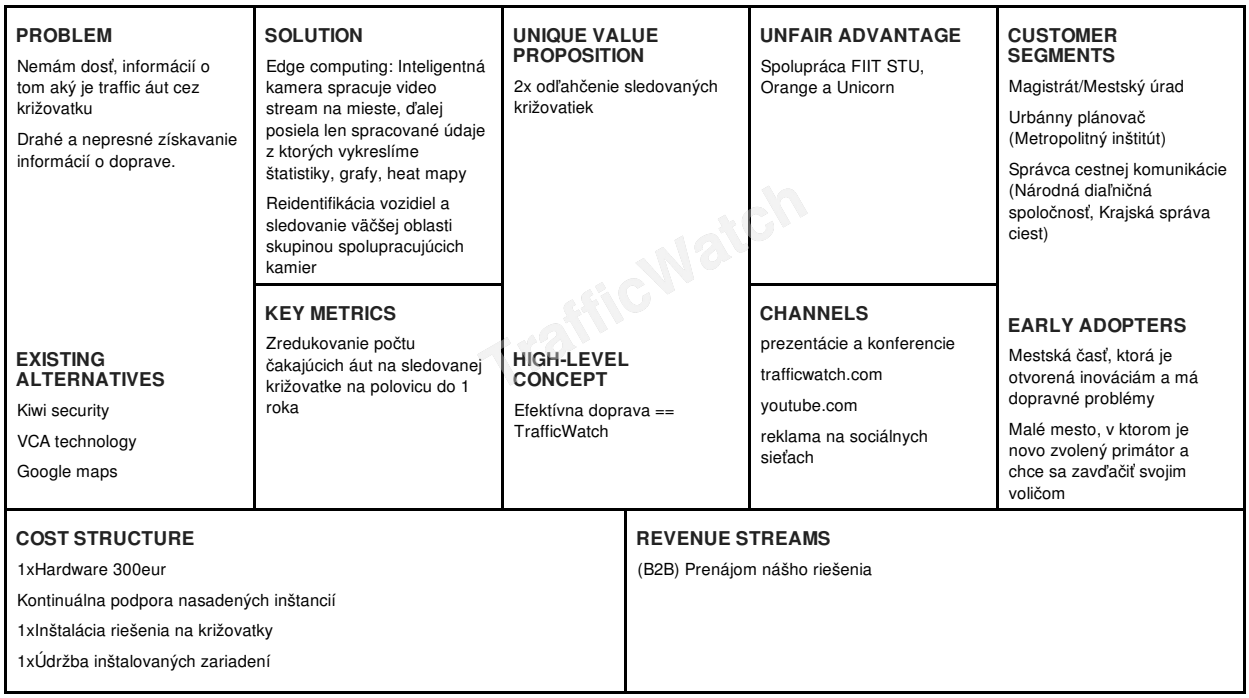
2. komponent zaoberajúci sa spracovaním dát
3. rozhrania s vystavenými službami, napr. pre front-end alebo ukladanie dát z kamery
4. komponent zaoberajúcu sa správou kamier

Webová aplikácia

Webová aplikácia bude vytvorená pomocou frameworku Angular. V spojení s knižnicami ako napr. Bootstrap nám umožní rýchlo vytvárať vizuálne príťažlivé a responzívne single-page aplikácie na profesionálnej úrovni. Údaje o doprave budeme zobrazovať predovšetkým na mape s využitím farebnej škály a základných geometrických tvarov ako body, krivky či polygóny. V rámci analýzy sme identifikovali ako vhodné riešenia Google Maps a HERE Maps. Obe riešenia poskytujú potrebné základné služby ktoré plánujeme využiť. Konečné rozhodnutie bude vykonané hlavne na základe obtiažnosti integrácie a ich cenového modelu. Tiež ho ovplyní rozhodnutie o tom, či chceme využívať pokročilé možnosti, ktoré poskytuje HERE Maps.

C Príprava na mentoring

Predstavenie nápadu



Obr. 3: Lean canvas

Otázky

1. Agilný vývoj a deadline - čo sa prvé prestane dodržovať?
2. Ako máme získať klienta ?
3. Aké veľkosti tímov sa používajú v praxi ?
4. Má zmysel pracovať podľa metodiky SCRUM keď sa tím skladá z expertov na rôzne problematiky ?

   Problém, ktorý v tomto prípade nastáva je, že počas standupu nikto nerozumie tomu, kto práve rozpráva.
5. Ako presvedčiť šéfa že scrum je cesta?
6. Ako začať so scrumom v priebehu projektu? Je to možné ?
7. Ako vyzerá fungujúce continuous delivery v praxi?
8. Ako často ste schopní dodávat softvér a ako často dodávate?
9. Ako sa vo vašom tíme/tímoch vyvíjal scrum (vlastné pravidlá, zrušenie niektorých a pod.)?
10. Ako formou napísať metodiku aby si ju niekto aj prečítal?
11. Z akých ľudí by sa mal skladať tím aby bola docielená "správna chémia"?

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Inžinierske dielo

Dokumentácia k tímovému projektu

Tímový projekt

Tím č. 21

Vedúci: Ing. Ivan Srba, PhD.

tim21.2018.fiit@gmail.com

Obsah

1 Úvod

Dokument opisuje stav projektu SmartMobility tímu TrafficWatch po ukončení 3 šprintov.

V prvej časti opisujeme globálne ciele projektu, následne zobrazuje clkový pohľad ny systém - uvádza "big picture"koncept projektu a to, z akých častí sa skladá.

2 Globálne ciele

2.1 Globálne ciele za zimný semester

Počas zimného semestra sme sa pustili do implementácie systému horizontálne. Kládli sme dôraz na kvalitnú analýzu technológií a stratégií, predtým než sme začali písať akýkoľvek kód. Následne, experimentálne aspekty projektu sa prototypovali, zatiaľ čo druhá časť tímu pracovala na rozbehaní infraštruktúry projektu.

Počas zimného semestra chceme dosiahnuť nasledujúcu funkcionalitu:

- detekcia áut, sledovanie prejazdov cez vyznačené zóny
- mať prepojený systém s fungujúcou infraštruktúrou
- UI na konfiguráciu kamier
- vizualizovať štatistiky o prejazdoch

3 Celkový pohľad na systém

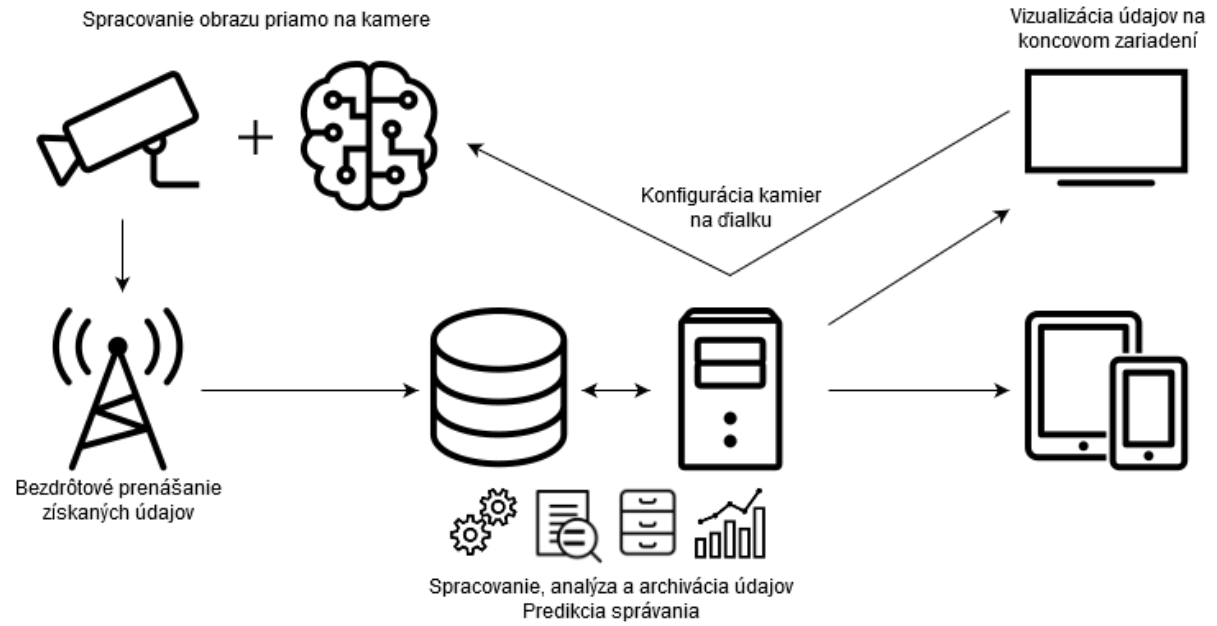
Obrázok 1 predstavuje zjednodušený pohľad na celkovú architektúru a rozloženie systému. Celkovo možno systém rozdeliť na 3 hlavné časti:

1. Inteligentná kamera
2. Server a úložisko dát
3. Webové rozhranie

Monitorovanie oblasti zabezpečuje kamera spojená s výpočtovým modulom, ktorý zaznamenaný obraz spracúva a rozpoznáva v ňom objekty záujmu, napr. vozidlá, cyklistov či chodcov. Po rozpoznaní objektu ho kamera sleduje a zaznamenáva o ňom informácie. Medzi základné informácie, ktoré určujeme, patrí pohyb objektu medzi stanovenými zónami, jeho rýchlosť, doba státia a podobne. Jednotlivé objekty kamera taktiež vo vhodnom momente klasifikuje. Cieľom je spracovať video záznam priamo na mieste a neprenášať ho zbytočne cez sieť. Cez sieť sa na server prenášajú len zistené informácie o pohybe a správaní sledovaných objektov.

Serverová časť zbiera informácie z jednotlivých kamier a priebežne aktualizuje štatistiky. Taktiež sa zaoberá dátovou analýzou, pričom využíva kombinované dáta získané z viacerých kamier. Takýmto spôsobom dokáže zisťovať informácie o toku dopravy či sledovať počet áut v danej oblasti. Server zároveň poskytuje služby potrebné na správne fungovanie webovej aplikácie.

Webová aplikácia sa stará o vhodné zobrazenie získaných informácií koncovým používateľom. Informácie o doprave možno zobrazovať ako text, graf alebo priamo na mape. Výzvou je nájsť spôsob, ktorý umožní zobraziť komplexné informácie o správaní sa dopravy jednoduchým a zrozumiteľným spôsobom.

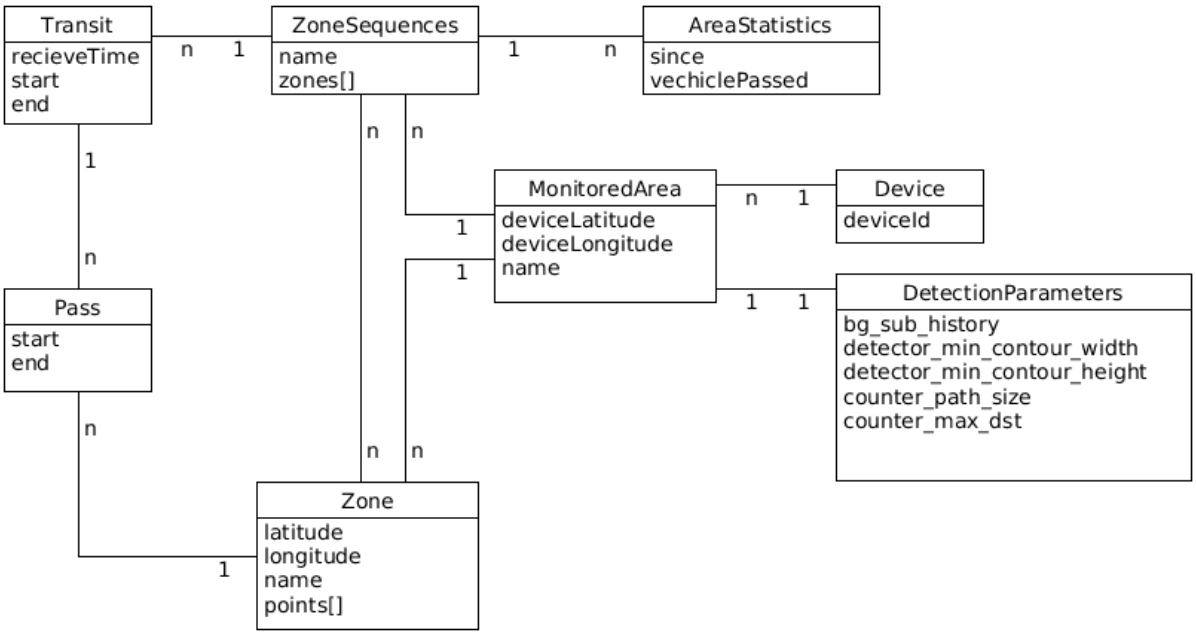


Obr. 1: Schéma riešenia

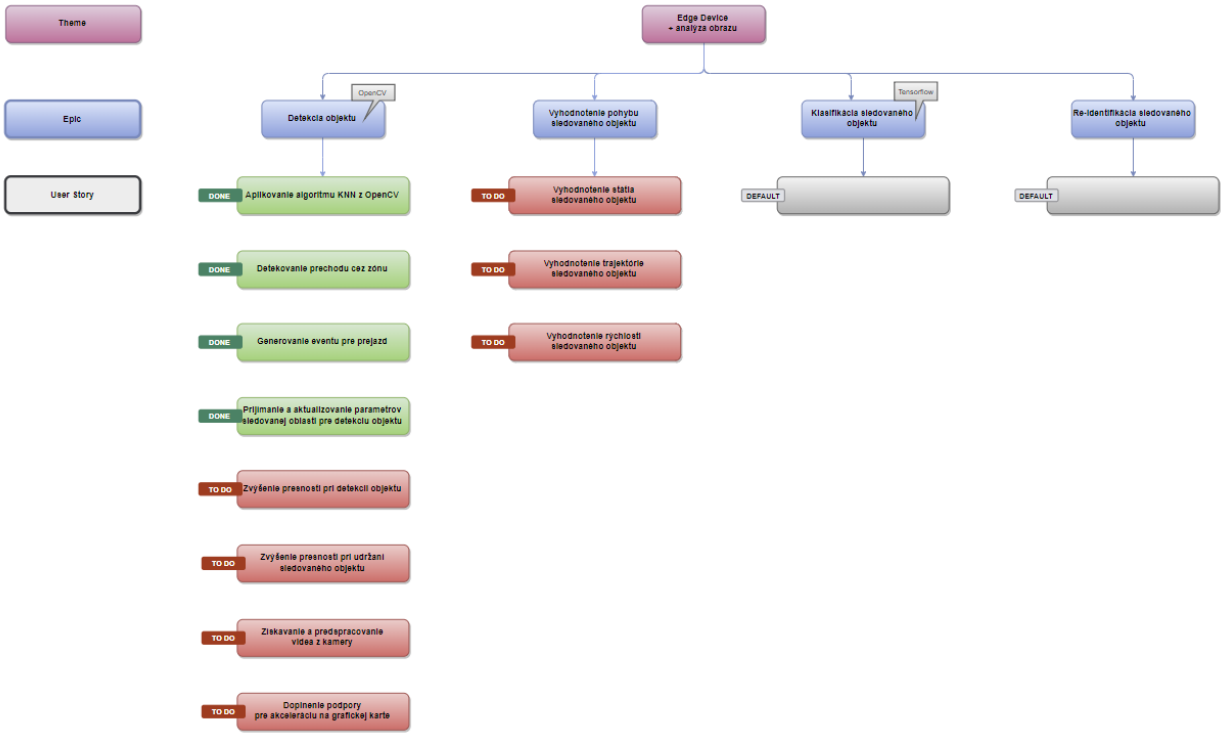
3.1 Model údajov

Obrázok 2 znázorňuje model údajov, ktorý používame na serveri a v databáze. Opis jednotlivých entít: **Device** - Fyzické zariadenie, kamera.
MonitoredArea - Oblasť sledovaná kamerou
Zone - Mnohouholník definovaný pomocou súradníc vrcholov/pixelov/bodov (x,y), používaný pri sledovaní prechodu.
Pass - Vstup sledovaného objektu do zóny a výstup zo zóny
DetectionParameters - Všetky konfigurovateľné hodnoty použité pri detekcií objektov kamerou (zóny, veľkosť sledovaného objektu)
Transit - Predstavuje vektor prechodov sledovaného objektu cez zóny
ZoneSequence - Predstavuje možnú kombináciu prechodov cez zóny, kvôli rýchlejšej práci s databázou
AreaStatistics - Agreguje informácie o počte áut, ktoré prešli cez prejazd za 1 hodinu

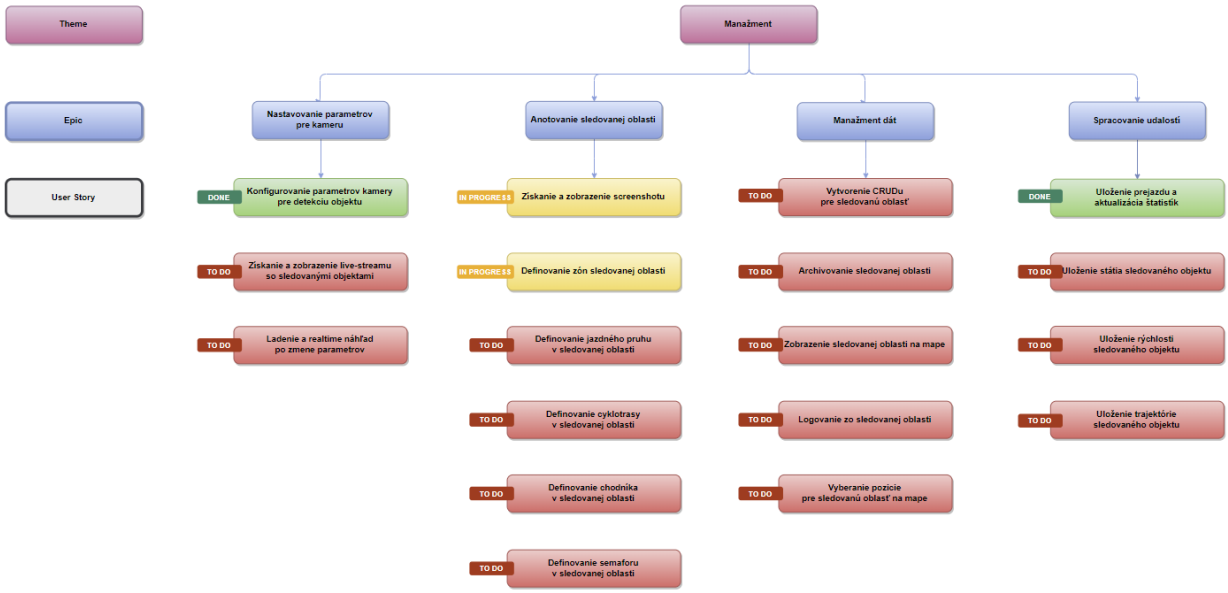
3.2 Reprezentácia pomocou WBS diagramu



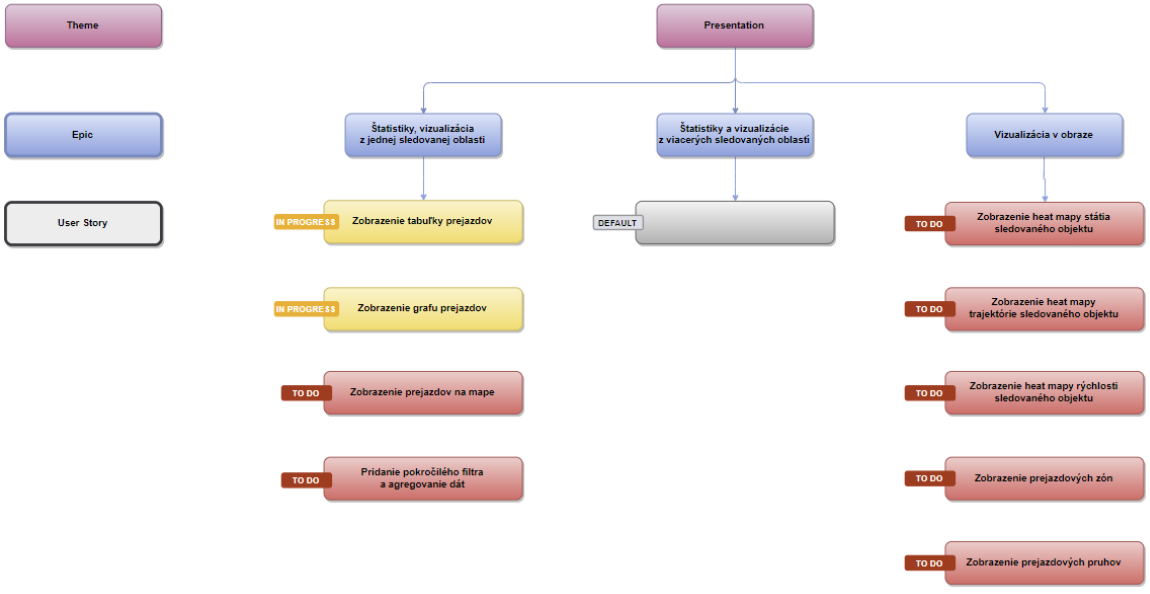
Obr. 2: Model údajov



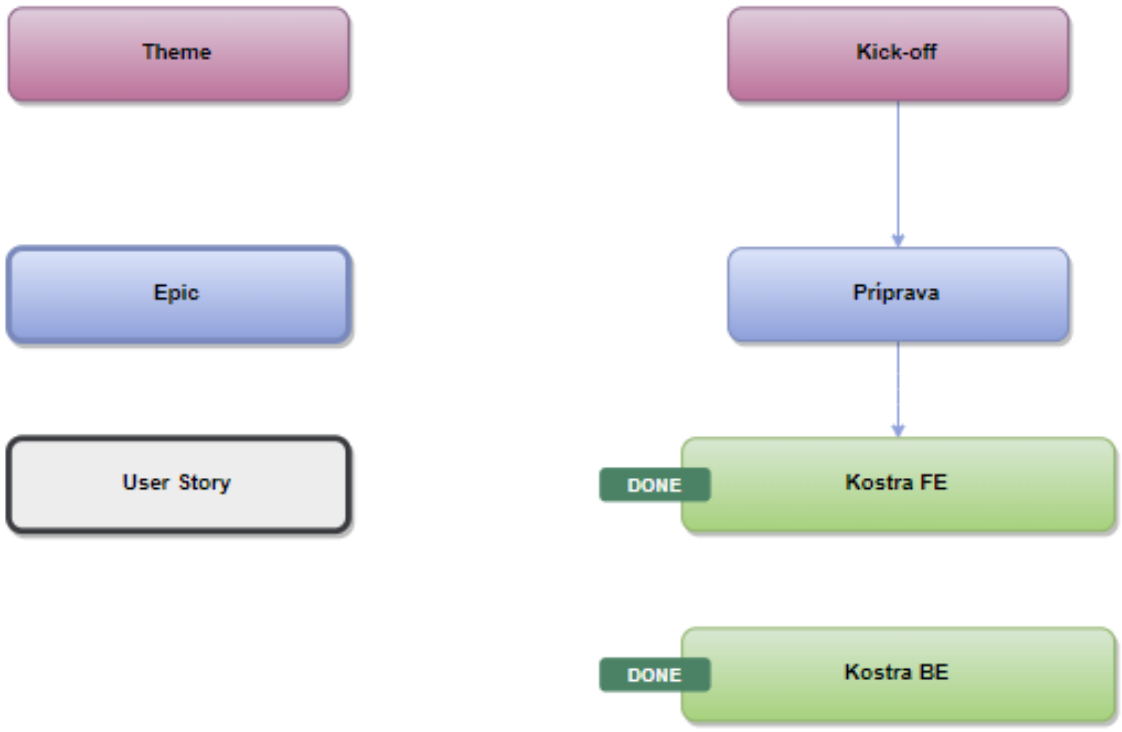
Tabuľka 1: EdgeDevice



Tabuľka 2: Metronic



Tabuľka 3: Metronic



Tabuľka 4: Metronic

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Moduly systému

Dokumentácia k tímovému projektu

Tímový projekt

Tím č. 21

Vedúci: Ing. Ivan Srba, PhD.

tim21.2018.fiit@gmail.com

Obsah

1 Analýza

1.1 Hardvér

1.1.1 Hardvér pre kamerový systém

Od spolupracujúcich firiem máme alokovaný neobmedzený budget, ktorý bude použitý na preplatenie nákladov na hardvér. Na zvolenú dosku bude potrebné pripojiť komunikačný modul a kameru. Hardvér je vhodné voliť tak, že sa budú neskôr môcť uchovávať vzorky videa. Napájanie hardvéru elektrickou sieťou nie je problémom.

Všeobecne sme zistili, že lacné dosky a dosky bez výkonnejšieho GPU sú výkonovo nedostatočné (kvôli potrebe akcelerácie neurónovej siete).

Po zvážení sme nakoniec zvolili nasledovný hardvér (vyznačený tučným, zvyšok tvoria alternatívy):

- **JETSON TX1 DEVELOPER KIT SE** – https://www.nvidia.co.uk/object/JetsonTX1DeveloperKitSE-cz.html?nvid=nv-int-jnt1drktsnt-21728
- Jetson TX2 – https://www.nvidia.co.uk/object/jetsontx2-edu-discount-cz.html
- **Huawei E5577C**, prípadne ľubovoľný LTE modem s Wi-Fi AP (alebo bez Wi-Fi AP ak vyskúšame a bude fungovať)
- **Logitech C920 PRO HD**, prípadne ľubovoľná kamera s aspoň 720p
- **USB micro B - USB A OTG** (napr. https://www.alza.sk/akasa-usb-micro-b-usb-a-otg-d4254841.htm)
- **SIM karta s dátami**

1.1.2 Požiadavky na HW akceleráciu

- TensorFlow - na GPU akceleráciu CUDA® Compute Capability >= 3.5 – Jetson TX1, Jetson TX2, inak nevhodné dedikované grafické karty
- Darknet neurónová sieť na YOLO – CUDA, Compute Capability neudávaná
- OpenCV – NVIDIA CUDA, Compute Capability prakticky ľubovoľná a OpenCL 1.2 (1.1 obmedzene)

1.1.3 Porovnanie použiteľného hardvéru

Nvidia JETSON TX1 DEVELOPER KIT SE Ponuka pre ČR je 5868Kč. Jetsony TX1/TX2 robí Nvidia priamo pre AI na embedded systémoch, výhoda oproti ostatným možnostiam je GPU akcelerácia – oproti čisto CPU násobné zrýchlenie (5 až 10 krát). Aj čistý Tensorflow object detection model tu zaručene pobeží aspoň 15 FPS. TX1 by určite postačoval, dá sa neskôr optimalizovať nižšie, ale situácia že nevieme zrealizovať náš usecase lebo nemáme dosť dobrý HW by tu nehrozila. Procesor má až 10x väčší výkon ako populárne single board dosky. Problémom je len jeden USB port, riešenie je kúpiť OTG redukciu na druhý (https://www.alza.sk/akasa-usb-micro-b-usb-a-otg-d4254841.htm).

Starší model Jetson TK1 sa už nedá veľmi zohnať, neakceleruje TensorFlow a len kvôli CPU sa neoplatí kupovať.

LATTEPANDA Výkon v benchmarku Geekbench je 3343 single-core a 6323 multi-core. Linux nemusí byť až tak dobre podporovaný. Neurónová sieť YOLO beží stále len na 0.62 FPS (https://www.youtube.com/watch?v=ic8zDDOIjbQ).

Firefly - AIO-3399J Výhodou je slot na LTE modul. Výkon v benchmarku Geekbench je 1326 single-core a 2935 multi-core. Nevýhodou je slabá podora v Linuxe (ovládače).

MinnowBoard Výkon v benchmarku Geekbench je 1004 single-core a 2874 multi-core.

HiKey 960 Výkon v benchmarku Geekbench je 1710 single-core a 4384 multi-core. Veľmi drahý.

ROCK64 Výkon v benchmarku Geekbench je 530 single-core a 1422 multi-core.

Intel Joule Výkon v benchmarku Geekbench je 671 single-core a 2318 multi-core.

1.1.4 Porovnanie Nvidia TX1 vs LATTEPANDA (UP-board)

Problém bol nájsť benchmark na LATTEPANDA, výkon má porovnateľný s UP-board (rovnaký procesor a frekvencia), bol teda v benchmarkoch zastupovaný. Výsledky sú dostupné na https://openbenchmarking.org/result/1608034-HA-1603058GA17 a https://openbenchmarking.org/result/1703199-RI-ARMYARM4104 (iný benchmark, nezahŕňa Z8350). Niektoré výsledky má Intel Atom x5-Z8350 (LATTEPANDA) výrazne lepšie (pravdepodobne optimalizované v x86), inak sú výsledky lepšie pre Nvidia (tesne) alebo porovnateľné.

Zhodnotenie: ak bez GPU akcelerácie, tak výkonnejšie zariadenie (v rovnakej cenovej kategórií by sa asi dal už kúpiť aj TX1), GPU akceleráciu by sme mohli chcieť (veľmi slabý výkon YOLO na CPU).

1.1.5 Sieťové moduly

Existujú špecializované produkty pre rôzne scenáre (odpočty), komunikačný modul však v našom prípade musí byť schopný preniesť vlastnú správu kvôli flexibilite. Sigfox má nedostatočný počet správ, kity stoja od 20 . Lora je drahšia s cenou 78,65 za Orange Starter Kit. Technológia NB IoT je aj na slovensku, je skôr ale v testovacej prevádzke. Na začiatok na prototypovanie tak vyberáme LTE modem, neskôr budeme môcť produkt doplniť o IoT komunikačnú technológiu. Ako protokol použijeme MQTT, ktorý bol definovaný product ownermi.

HUAWEI E3372 Mal by fungovať aj na Linuxe. Niektoré verzie aj emulujú ethernet cez usb (NAT), záleží od FW/distribútora (http://www.draisberghof.de/usb_modeswitch/bb/viewtopic.php?f=3&t=2261). Sú postupy, ako pridať ovládač (https://devtalk.nvidia.com/default/topic/1006033/jetson-tx2/lte-usb-module-on-tx2/, https://www.jetsonhacks.com/2018/04/21/build-kernel-and-modules-nvidia-jetson-tx
Cena je 44,49 . Podobný model E3372h má cenu 44,26 .

Alcatel Link Key 4G Používa RNDIS (MS proprietárny, implikuje NAT) – predvolene bez ovládača, mal by sa dať pridať. Cena je 38,15 .

E5577C Jedná sa o Wi-Fi modem, teda je isté, že bude fungovať. Cena je 68,05

1.1.6 Výber kamery

Kamera by mala mať podporu na Linuxe, dostupnosť v EÚ/Slovensko a podporu 1080p. Zatiaľ na prototypovanie si môžeme vystačiť z ľubovoľnou kamerou. Kameru je možné pripojiť aj cez špeciálny kamera modul (lepší výkon bez USB, znížené nároky na CPU), na prototypovanie je však vhodnejšia USB kamera (napr. kvôli dlhšiemu káblu alebo nižšej cene). Podporované formáty obrazu v prípade USB kamery môžu byť dôležité (môže byť väčšie zaťažovanie pri nekomprimovaných dátach). Aj menšie FPS kamery je dostatočné,

pri 50 km/h rýchlosti auta je posun pri 12 FPS 1,67 m medzi snímkami. Záleží aj na tom, v akej výške bude kamera umiestnená. 640 x 480 (320 x 240) by malo byť dostatočné na rozpoznanie druhu vozidla (auto, nákladné, ...) podľa existujúcich komerčných riešení, máme sa však orientovať na väčšie rozlíšenie.

C920 Overená na TX1, cena je 72,90 . Obľúbená je na počítačové videnie (https://www.chiefdelphi.com/forums/showthread.php?t=154360, https://www.reddit.com/r/computervision/comments/7x59de/inexpensive_5075_camera_for_machine_vision/du6hbm9/, https://www.quora.com/Which-camera-would-be-the-best-low-cost-solution-for-small-OpenCV-projects-like-Face-Tracking-and-Obje Má veľmi dobre manuálne nastaviteľné parametre (napr. zaostrenie), dala by sa tak nastaviť vzdialene cez konfiguráciu (https://www.kurokesu.com/main/2016/01/16/manual-usb-camera-settings-in-linux/). Je to 2. najpopulárnejší model (heuréka).

C922 Overená na TX1, cena je 85,90 . Na počítačové videnie je rovnako dobre použiteľná ako C920 (https://devtalk.nvidia.com/default/topic/1002483/jetson-tx1/reliable-usb-3-0-camera/). Je to 1. najpopulárnejší model (heuréka).

Iné vhodné kamery Neboli porovnávané, keďže sú obsolete (napr. 720p modely), alebo sú to podobné Logitech kamery v rovnakej cenovej kategórií.

1.1.7 Iné poznámky k hardvéru

Nebudeme mať verejnú IP adresu cez modem (niektoré modemy priamo robia NAT) – potrebné je vyriešiť vzdialené pripojenie (zariadenie sa bude aktívne pripájať/udržiavať komunikačný kanál). Modem cez PCIe nie je vhodný, väčšina modulov je cez mini PCIe a adaptér je položka navyše k aj tak drahšiemu modulu. Potrebujeme LTE na prenos 1080p streamu. Wi-fi modem je isté riešenie z hľadiska spoľahlivosti a je jednoduchší na prototypovanie.

1.2 Komunikácia medzi kamerou a backendom

1.2.1 Voľba MQTT servera

V tabuľke 1 sú popísané použiteľné MQTT servery. Neuvádzané sú neudržiavané, uzatvorené, platené, prípadne inak nevyhovujúce. Definovanými požiadavkami boli:

- dá sa jednoducho spustiť standalone (až spadne server, nemusia sa stratiť správy)
- robí len to čo má (nie nejaký väčší framework alebo pre nás zbytočné funkcionality navyše)
- podporuje čo najviac MQTT funkcionality
- obstojný v benchmarkoch

Z trojice vhodných bol dalej kvôli slabému výkonu (https://github.com/home-assistant/home-assistant/issues/12117, https://community.home-assistant.io/t/hbmqtt-default-mqtt-broker-doesnt-support-heavy-flow 37333) vyradený HBMQTT. Zvolený bol najskôr VerneMQ, ktorý sa úspešne používa aj pri vyšších záťažiach. Tento sa však nepodarilo nastaviť (problémy so zabezpečením), prešli sme tak na Mosquitto.

Server beží na adresách tls://team21-18.studenti.fiit.stuba.sk:8883 (secure TCP) a wss://team21-18.studenti.fiit.stuba.sk:8083 (secure websocket), na otestovanie je možné sa pripojiť napr. cez MQTTBox.

Meno	Standalone	Zbytočnosti	Funkcionalita
ActiveMQ (Artemis)	áno	áno	áno
emitter	áno	nie	nie
HBMQTT	áno	nie	áno
Mongoose	nie	áno	nie
Moquette	áno	nie	nie
mosca	áno	áno	nie
Mosquitto	áno	nie	áno
tstack	áno	nie	nie
VerneMQ	áno	nie	áno

Tabuľka 1: Porovnanie MQTT serverov

1.2.2 Voľba MQTT knižníc

Java

- Eclipse Paho Java
- XenQTT - neaktualizované od 2015
- MeQanTT - neaktualizované od 2012
- mqtt-client - neaktualizované od 2016
- Qatja - málo funkcionality

Zvolené Paho je udržiavané a nestane sa, že v budúcnosti bude potrebné vymeniť knižnicu kvôli nepodporovanej funkcionalite. MQTT knižnica bola zakomponovaná do build procesu servera.

Na systéme MacOS sa vyskytli tradičné problémy – zlyhanie bezpečného pripojenia z dôvodu nerozpoznania dôveryhodnosti použitého certifikátu. Na vyriešenie vykonáme nasledovné príkazy, ktoré používaný certifikát pridajú napevno do keystore (po vymenení certifikátu je potrebné zopakovať): openssl x509 -in <(openssl $s_client - connectteam21 - 18.studenti.fiit.stuba.sk : 8883 : 443 - prexit2 > /dev/null) - out /certificate.crtsudokeytool−importcert−file /certificate.crt−aliasexample−keystore(/usr/libexec/java_home)/jre/lib storepasschangeit

Python

- Eclipse Paho Python
- gmqtt
- Nyamuk - neaktualizované od 2016
- hbmqtt

Paho Python bol zvolený kvôli rovnakej knižnici použitej pre jazyk Java.

1.3 Zápisnica zo stretnutia s Martinom Tamajkom

1.3.1 Detekcia a rozpoznanie auta

základný postup: detekcia vozidla z opencv, pridanie marginu a rozpoznanie typu v neurónovej sieti

(dense) optical flow - zistenie smeru pohybu medzi obrázkami konkrétne Gunnar Farneback's algorithm

ďalej je možné klastrovať na základe uhlových šípok, napr. pomocou DBSCAN

Background modeling/subtraction odpočítanie pozadia medzi dvoma po sebe idúcimi snímkami, získanie tak masky pozadie/popredie

Kontúrová analýza analýza, či je auto stále to isté medzi snímkami (sledovanie) - zobrať kontúru a porovnať ju s kontúrami z predošlých snímkov

porovnanie na základe vzdialenosti, alebo farby (lepšie, porovnanie histogramov)

Odstránenie šumu

- treshold
- mediánový filter
- zahodiť najmenšie kontúry

Zistenie typu vozidla

- neurónová sieť - napr. YOLO na rozpoznávanie objektov (https://pjreddie.com/darknet/yolo/, https://www.youtube.com/watch?v=g5BECgk9AEw, https://www.youtube.com/watch?v=PncSIx8AHTs, https://github.com/tejaslodaya/car-detection-yolo)
- pomocou detekcie ŠPZ - malo by byť dostupné API, ktoré vracia aj typ vozidla

Riešenia problému detekcie prekrývajúcich sa áut Je to dosť závažný problém, mohli by sme chcieť ho riešiť.

Opatrenia:

- umiestnenie kamery zvislo dolu, minimalizujeme tak perspektívne skreslenie
- detekcia pruhov a obmedzenie kontúry na jeden pruh

1.3.2 Detekcia auta medzi kamerami

dôležité je zabezpečenie rovnakého nastavenia kamier (pohľad a i.)

Prístup 1: kľúčové body opis auta pomocou bodov, pri ktorých je pravdepodobnosť, že budú rozpoznané aj na inom zábere

algoritmy SIFT (pomalý, lepší, lepšie na GPU), SURF, FAST

spárovanie kľúčových bodov

RANSAC - zrušenie falošných párovaní – https://vgg.fiit.stuba.sk/2013-07/object-recognition-ransac-verificat

Prístup 2: siamské neurónové siete využíva dve rovnako nakonfigurované neurónové siete

na výstupe je, či je to rovnaký objekt

tzv. one-shot learning

Prístup 3: template matching veľmi nefunguje (študenti testovali na cvičeniach)

1.3.3 Iné poznámky

prototypovanie v pythone, do produkcie C++

odporúčaná kombinácia prístupov, z viacerých výstupov rozhodnúť (hlasovanie):

- klasické opencv
- neurónové siete
- iné klasifikátory

"Rozhodnúť sa, čo je horšie - false positive alebo false negative? Môže mať vplyv napr. na rozhodnutie, ako spojiť rôzne paralelne použité algoritmy"

1.3.4 Odkazy

https://docs.google.com/document/d/1XSlcn4ZJxyspVy0X-_YLgvbUgA52AQibret-weNZE2g/edit

https://www.youtube.com/watch?v=inCUJ0JM5ng

https://github.com/topics/vehicle-counting

https://medium.com/machine-learning-world/tutorial-making-road-traffic-counting-app-based-on-computer-

https://chatbotslife.com/vehicle-detection-and-tracking-using-computer-vision-baea4df65906

1.4 Softvérové vybavenie kamery

1.4.1 Modely a implementácie (Neurónové siete)

Detekcia objektov (aka objavenie a vyznačenie objektu na obrázku) neurónkami je výrazne náročnejšia úloha ako obyčajná klasifikácia. Samostatná kapitola je detekcia na videu realtime, ešte donedávne neexistoval model schopný túto úlohu splniť. Tri roky dozadu vznikli dva prístupy (modely), ktoré sa vyvíjajú a stále sú v zásade jediné dve možnost

Sú to:

- YOLO (you only look once) - https://pjreddie.com/darknet/yolo/
- SSD (single shot multibox detector) - https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

Toľko modely, nás však zaujíma aj implementácia, chceli by sme niečo štandardné, odtestované, s dobrou podporou. Pri detekcií máme v podstate dve možnosti.

Štandardom pre YOLO je originálna implementácia od autora modelu vo frameworku darknet (jeho custom framework napísaný v C a CUDA) a spĺňa všetky vyššie podmienky. Nevýhodou je ten darknet, tažšie by sme v ňom implementovali prípadné modifikácie, na druhú stranu nič extra by sme nemali potrebovať. https://pjreddie.com/darknet/yolo/

V Tensorflowe máme od Googlu oficiálne Object Detection API, udržiavané repo ktorého najväčšia výhoda je množstvo predimplementovaných modelov na rôzne účely, okrem iného aj SSD (okrem iného aj mobilenet variáciu SSD (to je neurónkový model od Googlu špeciálne robený s tým aby rýchlo bežal na mobilných zariadeniach)).

1.4.2 Performance

Skúšal som niekoľko implementácií využívajúcich TF Object Detection API, plus som pozeral čo poskúšali iní.

- čisto CPU na mojej staršej i5ke som dal max 7 FPS, pekný výsledok ale nie dosť. https://github.com/datitran/object_detector_app
- s použitím GPU (NVIDIA computation capability len trochu menej ako Tegra X1 z Jetsonu) som na inej implementácií dostal 15 FPS čo už je použiteľná hodnota. Potvrdzuje to, že na na zariadení s GPU vieme použiť aj čisto neurónkový detekčný model.
- pohľadal som pár ľudí čo robili detekciu priamo na Jetsone TX1, SSD aj YOLO, reportujú od 10 do 20 FPS

Poučenie - čisto na CPU, čisto neurónkový detekčný realtime model nikdy nerozbeháme. S GPU áno viacerým sa to už experimentálne podarilo. Prakticky jediné embedded zariadenie ktoré to umožňuje je NVIDIA Jetson TX1/2

1.4.3 Presnosť

Jedna vec je, že máme model s dostatočným FPS, druhá vec je presnosť. V skúšaných modeloch ktoré mali dostatočné FPS bola kvalita detekcie vo všeobecnosti neuspokojivá. Samozrejme, model bude ešte možné dotrénovať (bude to nutné aby rozoznal triedy ktoré chceme) čím zlepšíme kvalitu, osobne však pochybujem že takto dosiahneme dostatočnú kvalitu. Ak má byť náš nástroj použiteľný musíme dosiahnúť aspoň nejakú minimálnu presnosť, a čisto neurónkový model na embedded zariadení bežiaci realtime to podľa mňa nedáva.

Pravdepodobne sa tým pádom nevyhneme nejakej kombinácií/spolupráci s tradičnými computer vision algoritmami (cez OpenCV).

1.4.4 Čo ďalej

Bez neurónok to nepôjde, chceme rozoznávať typy áut a to klasickými CV algoritmami (OpenCV) nedávame. Čisto neurónkami to tiež nepôjde, lebo naraz realtime aj kvalitu nezvládajú. Tu sú nejaké nápady:

- Nezávisle bežia OpenCV a TF modely, kde sa zhodnú je auto
  - ktovie či máme dosť výpočtovej kapacity na takéto srandy
- OpenCV detekuje možné autá, TF klasifikuje (nie detekuje) či naozaj
  - klasifikácia je rýchlejšia ako detekcia, vieme ju robiť rýchlo a kvalitne
  - Ak ale OpenCV auto nenájde už nám nič nepomôže
  - Ak OpenCV dá viac áut do jedného boxu musíme nejako vymyslieť splitovanie
- niečo iné?

OpenCV background substraction beží rýchlo a takmer nemá false negatives. Ak však auto zastane zmizne.

1.4.5 Datasety

Nech to bude akokoľvek, budeme musieť neurónku trénovať na nejakých dátach. Časom si môžme niečo oanotovať sami, ale to je veľa roboty. Čo existujúce som našiel:

- Detrarc challenge - anotovane data z cinskych ciest, zabery z dopravnych kamier http://detrac-db.rit.albany.edu/download
- Nvidia Ai city cahllenge - k dátam som sa zatiaľ nedokopal http://smart-city-sjsu.net/AICityChallenge/data.html
- Aicitychallenge - riešia rýchlosť, detekciu anomálií a znovuobjavenie auta, datasety nemusia byť verejne dostupné https://www.aicitychallenge.org/2018-data-sets/

Bonus, podľa všetkého sú nejaké datasety aj na znovuobjavenie, aka reidentifikáciu

https://github.com/knwng/awesome-vehicle-re-identification

1.5 OpenCV

Open Source Computer Vision Library - knižnica ponúkajúca optimalizované funkcie využívajúce metódy počítačového videnia.

Analyzované prístupy detekcie vozidiel pre náš projekt:

- **Haar** - otestované, tento prístup nie je vhodný z dôvodu časovej náročnosti algoritmu
- **HOG** - nepodarilo sa otestovať
- **MOG background subtraction + Kontúrová analýza** odporúčaný prístup, otestovaný, vhodný

Testovné kvality videozáznamu pri detekcii zvoleným algoritmom:

- Viac ako 720p - detekcia bola príliš pomalá
- 240p až 720p - detekcia v reálnom čase
- Zmena FPS neovplyvnila zásadným spôsobom úroveň detekcie (4 vs 15)

Pri akcelerácii OpenCV na GPU sa detekcia pravdepobodne zrýchly, testovanie prebiehalo na CPU.

Postrehy:

- Parametre do OpenCV funkcií (pre rôzne videá treba nastavit rôzne tresholdy)
- Spájanie detekovaných vozidiel do jedného detekovaného vozidla, ak sú blízko seba - riešiteľné uhlom, pod ktorým je umiestnená kamera a detekciou pruhov
- Detekované sú len pohybujúce sa vozidlá

1.6 Backend

1.6.1 Databáza

Klasické SQL databázy nepripadajú do úvahy, porovnávali sme hlavne NoSQL databázy a rôzne rozšírenia SQL databáz:

1. TimescaleDB
2. InfluxDB
3. CrateDB
4. MongoDB
5. Apache Cassandra

Na internete nie sú k dipozícii presné benchmarky, väčšinou sa dajú nájsť len články/blogy, často propagujúce konkrétnu technológiu. Z dostupných zdrojov sa ako najlepšia javí TimescaleDB, knižnica rozširujúca PostgreSQL.

Dôvody a výhody:

1. Benchmarkovo poráža všetky ostatné - hlavne pri náročných dopytoch typu GROUP BY + ORDER BY + LIMIT
2. SQL syntax - jednoduchá a rozšírená
3. Rozšírenie PostgreSQL, ktorý je rokmi overený, otestovaný, dobrá kompatibilita
4. Škálovateľnosť - https://docs.timescale.com/v0.12/faqscaling
5. Hypertables
6. Indexy na časových údajoch
7. Custom funkcie na prácu s časom, napr. time_bucket()

1.6.2 Porovnanie manažérov závislostí

Pre jednoduchšiu prácu s externými knižnicami je vhodné použiť manažéra závislostí (angl. depenency manager). Zamerali sme sa dva populárne nástroje, Maven a Gradle. Maven

- založené na XML
- jednoduchá validácia, v skriptoch (zvyčajne) nevznikajú bugy
- lepšia podpora pluginov, menej problémov pri používaní, viac dostupných pluginov
- lepší tooling pre písanie skriptov

Gradle

- vlastný jazyk založený na Groovy
- oveľa kratšie súbory
- plnohodnotný jazyk, dá sa v ňom nakódiť čokoľvek, častejšie môžu vzniknúť bugy
- ťažšie na naučenie sa
- podporuje inkrementálny build
- rýchlejšie buildy

1.6.3 Záver

Gradle je silnejší nástroj, dá s as ním toho viac robiť, ale je ťažší na naučenie, je tiež rýchlejší, obzvlášť pri veľkých projektoch. Maven je jednoduchší, je ťažšie na tom niečo pokaziť, lepšia podpora toolov a pluginov.

Pre naše účely bola rozhodujúca rýchlosť, nakoľko žiadne zložité skripty alebo pluginy sme používať neplánovali. Rozhodli sme sa preto pre Gradle.

1.7 Frontend

1.7.1 Mapové riešenia

V rámci analýzy sme porovnávali 2 mapové riešnia - Google Maps a HERE Maps. Google Maps

API:

1. JavaScript
2. Android SDK
3. iOS SDK

Ponúkané služby, ktoré môžme využiť:

1. Heatmapy
2. Custom štýlovanie
3. Traffic Layer

Možnosti vykresľovania:

1. markers, popups, info panels
2. polylines, polygons, circles, rectangles
3. používateľom zadané/editované body a oblasti(polygons, rectangles...)

Pricing:

- nutný API klúč (účet a aktívny billing)
- platba za každý prístup
- kredit v hodnote 200$ zadarmo každý mesiac - up to 28,000 loads(dynamic maps)
- 0.007$ za každý load naviac (7$ za 1000 loadov) do 100 000 loadov, 5.6$ do 500 000 loadov... potom custom plány

Podpora prehliadačov:

1. Microsoft Edge
2. IE 10 a 11
3. Firefox
4. Chrome

5. Safari

HERE Maps

Ponúkané služby, ktoré môžme využiť:

1. Map Satellite Tiles
2. Custom Location Extension - Store, manage and retrieve custom POIs and polygons
3. Platform Data Extension - Get additional HERE Map Content, including height and slope values, curvature, speed limits and traffic lights

Možnosti vykresľovania:

1. Markers - normal a DOM
2. Polylines
3. Polygons, Circles, Rectangles

Pricing (mesačne):

1. FREEMIUM - free, 250k tranzakcií + 5k aktívnych userov,
2. PRO - 449$, 1M tranzakcií + 5k aktívnych userov
3. CUSTOM - dá sa dohodnúť custom plán

Podpora prehliadačov:

1. Internet Explorer 10+
2. Firefox (latest)
3. Google Chrome (latest)
4. Apple Safari 6+
5. Internet Explorer 9

Rozhodnutie V službách veľký rozdiel nie je - obidve riešenia poskytujú potrebnú funkcionalitu, HERE Maps ale ponúkajú aj služby navyše (napríklad vytváranie vlastných bodov či vrstiev mapy). Obidve riešenia majú určitý druh free plánu, ktorý nám pri vývoji postačí. Čo sa týka podpory, obidve riešenia poskytujú integráciu formou Javascriptovej knižnice.
Pri riešení HERE Maps sa ale javia pricing plány ako výhodnejšie. Preto sme sa rozhodli využiť na vizualizáciu získaných informácií HERE Maps.

1.7.2 Frontendové frameworky

Angular
Výhody:

- dobrá dokumentácia
- RXJS, HttpClient
- two-way data binding

- dependency injection
- TypeScript

Nevýhody:

- problémy s migráciou (release každých 6 mesiacov)
- strmá krivka učenia
- TypeScript

ReactJS
Výhody:

- ľahký na naučenie
- flexibilita
- virtual DOM
- Downward data binding
- 100% open source
- ľahká migrácia

Nevýhody:

- problémy s dokumentáciou - product owneri nám pomôžu
- príliš voľnosti
- potrebný dlhý čas na tzv. "master"level

VueJS
Výhody:

- detailná dokumentácia
- prispôsobivosť
- integrácia
- škálovateľnosť
- veľkosť

Nevýhody:

- malé zastúpenie na trhu
- over flexibility
- problém s dokumentáciou v angličtine

Kvôli jednoduchosti učenia a preferencii product ownerov sme sa rozhodli frontend vyvíjať v Reacte. Umožní nám rýchlo začať vyvíjať webové rozhranie a prípadné problémy môžme riešiť aj s product ownermi, ktorí nám sľúbili pomoc.

1.7.3 Porovnanie Metronic vs. AdminLTE

Metronic
Výhody:

- má vysoké rozlíšenie
- má veľké množstvo widgetov, diagramov, atď.

Nevýhody:

- nie je priamo integrovateľný s React-om
- spoplatnený (35$), neponúka žiadne voľné šablóny

Podpora prehliadačov:

1. Internet Explorer 10+
2. Firefox (latest)
3. Google Chrome (latest)
4. Apple Safari 6+
5. Microsoft Edge (latest)
6. Opera (latest)

Kompatibilita:

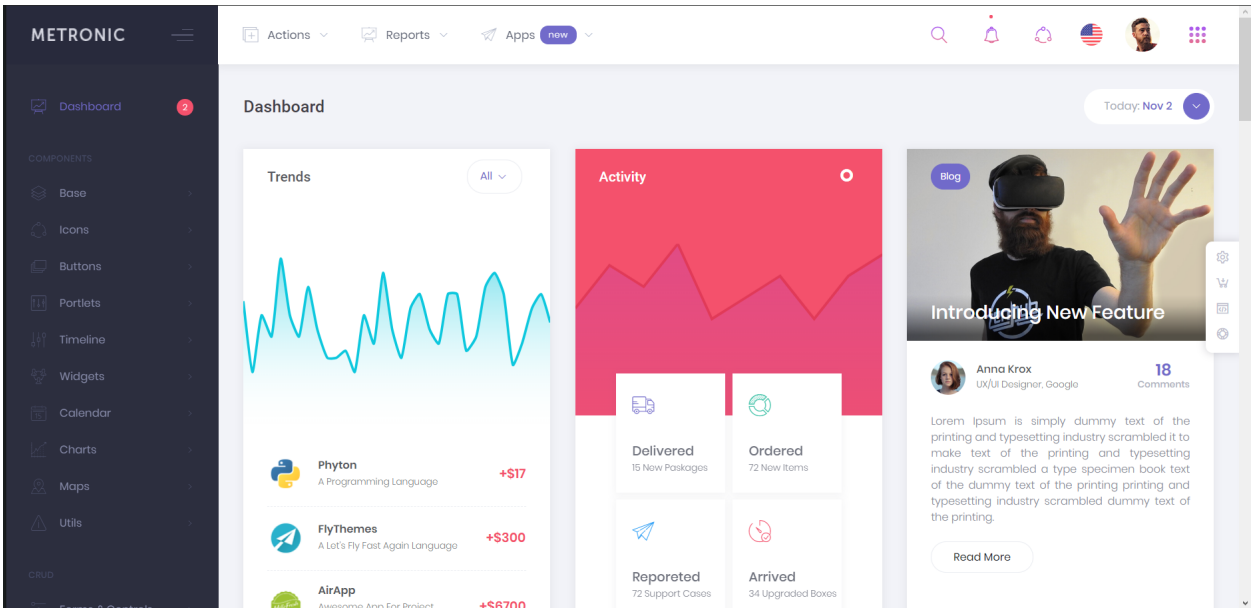
1. AngularJS 2
2. Bootstrap 4.x
3. Bootstrap 3.x

AdminLTE
Výhody:

- má vysoké rozlíšenie
- má veľké množstvo widgetov, diagramov, atď.
- ponúka aj šablóny zadarmo (študentská MIT licencia)

Podpora prehliadačov:

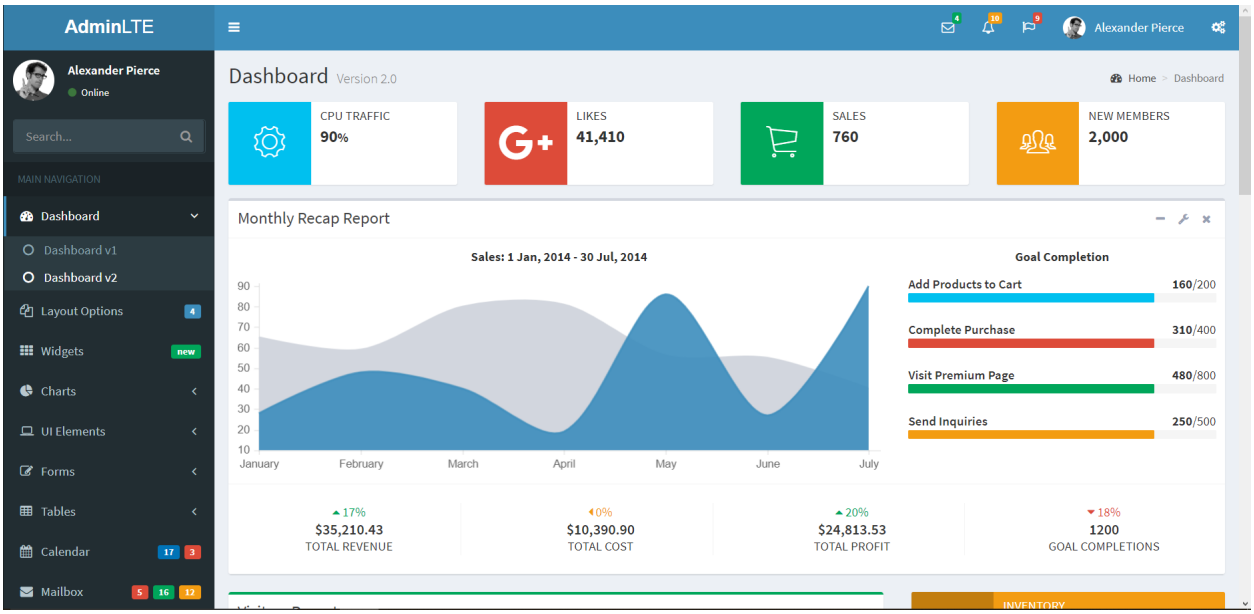
1. Internet Explorer 9+
2. Firefox (latest)
3. Google Chrome (latest)
4. Apple Safari 6+
5. Microsoft Edge (latest)
6. Opera (latest)



Tabuľka 2: Metronic

Kompatibilita:

1. React
2. Laravel
3. AngularJS 2
4. Bootstrap 3.x+



Tabuľka 3: AdminLTE

Vzhľadom na fakt, že Metronic nemá priamu integráciu s Reactom, rozhodli sme sa použiť AdminLTE, ktorý ponúka podobnú funkcionalitu aj v šablónach, ktoré sú zadarmo.

2 Návrh

2.1 Kamera

Kamera je zodpovedná za tvorbu videozáznamu a následne spracovanie tohto záznamu, teda detekovanie objektov, vyhodnotenie ich pohybu a odoslanie výsledkov na server.

Kamera je implementovaná primárne v jazyku Python, kvôli využitiu CUDA akcelerácie a optimalizácií sú časti implementované v C++.

2.1.1 Organizácia

- **trafficwatch** - Hlavný modul, z ktorého sa spúšťa celá camera app. Slúži na inicializáciu detekčných modulov, MQTT modulu a tiež modulu spravujúceho konfiguráciu. Niekoľko prvých snímkov je spracovaných výhradne na počiatočné natrénovanie algoritmu KNN. Následne sú v cykle spracúvané jednotlivé snímky vstupného videozáznamu (pomocou nainicializovaných modulov) a výsledky sú odosielané v podobe eventov na server.
- **detector** - Modul zabezpečujúci detekciu pohyblivých objektov. Pri vytvorení triedy VehicleDetector prebehne inicializácia detekčných parametrov z konfiguračného súboru. Tieto parametre je tiež možné za behu programu aktualizovať. Pri volaní triedy sa aktuálna snímka spracuje background-subtraction algoritmom (v našom prípade je to algoritmus KNN). Výsledkom je maska obsahujúca detekované pohybujúce sa objekty. Táto je následne očistená od šumu, vyhladená a odoslaná do metódy, ktorá implementuje OpenCV algoritmus na nájdenie kontúr. Výstupom je zoznam detekovaných objektov.
- **counter** - Modul zabezpečuje počítanie prejazdov áut cez zóny. Pri vytvorení triedy VehicleCounter prebehne inicializácia detekčných parametrov z konfiguračného súboru. Tieto parametre je tiež možné za behu programu aktualizovať. Pri volaní triedy je pre objekty detekované triedou VehicleDetector zapamätávaná ich poloha v jednotlivých snímkach, pričom množiny týchto polôh tvoria trajektórie jednotlivých detekovaných objektov. Na základe týchto trajektórií sú následne identifikované prejazdy, ktoré sú počítané.
- **visualizer** - Modul slúžiaci na zobrazenie priebehu detekcie vo vstupnom videozázname.Do každej snímky videozáznamu sú vyznačené detekované objekty, ich trajektórie, zóny a tiež počítadlo prechodov, ktoré je súhrnné pre všetky zóny.
- **config** - Modul obsahuje triedu ConfigManager, ktorá slúži na správu konfiguračných súborov. Sú podporované 3 typy konfiguračných súborov:
  - defaultný - obsahuje všetky parametre, ktoré je možné nastavovať a ich východzie hodnoty.
  - používateľský - hodnoty parametrov vložené do tohto konfiguračného súboru majú vyššiu prioritu ako východzie. Tento konfiguračný súbor má praktické využitie napr. pri vývoji, keďže pri experimentovaní so zmenami parametrov nie je potrebné prepisovať východzie hodnoty.
  - prijatý zo serveru - má najvyššiu prioritu. Hodnoty parametrov prijaté zo serveru prostredníctvom MQTT sú ukladané do tohto typu konfiguračného súboru
- **mqtt** - Modul spravujúci MQTT komunikáciu na kamere. Použitie MQTT modulu umožňuje pripojenie sa do komunikácie, posielanie správ a prijímanie správ.
- **utils** - Modul obsahuje pomocné funkcie používané pri detekcii objektov, počítaní prejazdov a vizualizácii v ostatných moduloch

2.2 Backend

Backend (alebo server) je implementovaný v jazyku Java. Je realizovaný frameworkom Spring Web MVC.

2.2.1 Architektúra

Pre backend sme zvolili architektonický štýl MVC. Controller pozostáva z REST API, pristupuje k entitám, ktoré zodpovedajú entitám v databáze.

2.2.2 Organizácia

Projekt je organizovaný do nasledujúcich balíkov:

- **controller** - obsahuje REST API pre aplikáciu
- **entity** - objekty, ktoré predstavujú dátové modely
- **service** - služby pre kamery a frontend, obsahuje logiku systému
- **storage** - logika spojená s prístupom do databázy (rozšírenie JPA)
- **util** - pre všeobecné pomocné funkcionality (logger, exceptions)

3 Implementácia

3.1 Kamera

Zo serveru je pomocou aktualizácie konfiguračného súboru možné nastavovať tieto parametre:

- **video_source** - cesta k súboru s videom, na ktorom bude program spúšťať detekciu vozidiel
- **bg_sub_history** - počet framov, ktoré sa uchovávajú pre výpočet background modelu algoritmom MOG2. Čím je číslo menšie, tým ľahšie sa objekty stávajú súčasťou pozadia a nie sú detekované ako pohybujúce sa.
- **detector_min_contour_width** - minimálna šírka detekovanej kontúry.
- **detector_min_contour_height** - minimálna výška detekovanej kontúry. Spolu s minimálnou šírkou detekovanej kontúry určujú, akú minimálnu veľkosť na obraze musí mať objekt aby bol detekciou označený za pohybujúce sa vozidlo. Ak je napr. kamera ďalej od vozovky, treba tieto parametre zmenšiť, keďže vozidlá sa budú na obraze javiť ako menšie.
- **zones** - oblasti, v ktorých je detekovaný prejazd počítaných vozidiel. Každá oblasť ma svoje ID a je definovaná poľom vrcholov (dvojice X a Y súradníc), ktoré v obraze túto oblasť tvoria.
- **transit_table** - slovník obsahujúci dvojice ID a typov prejazdov.
- **counter_path_size** - počet bodov zobrazovanej trasy vozidla. Čím viac bodov, tým budú zobrazované trasy dlhšie.
- **counter_max_dst** - maximálna vzdialenosť novopridávaného bodu trasy od existujúcej trasy. Ak je príliš veľká, bude trasa nepresná - bude obsahovať vzdialené body, ktoré do nej nepatria. Ak je príliš malá, relevantné body nebudú do trasy pridané

3.2 Backend

Dokumentácia implementácie backendu je poskytnutá vo forme exportu z nástroja Javadoc ako príloha A.

4 Testovanie

Zo všetkých modulov systému sú zatiaľ testami pokryté len niektoré moduly na strane serveru. Na strane kamere o testoch zatiaľ neuvažujeme. V neskorších fázach riešenia projektu sa zavedú automatické testy pre frontend, napr. pomocou nástroja Selenium.

4.1 Backend

Na strane serveru sú dostupné jednotkové testy, ktoré sa zameriavajú na overenie správnosti fungovanie CRUD operácií nad entitami z modelu údajov.
Pri testovaní používame knižnicu JUnit, ktorú sme spojili s knižnicou DBUnit.
Každý test si pred spustením inicializuje testovaciu databázu a naplní ju údajmi zo zadaného XML súboru. Po úspešnom alebo neúspešnom vykonaní testu sa dáta z databázy odstránia. Výsledkom sú testy, ktoré sú samostatne spustiteľné a nezáleží na poradí ich vykonávania.

5 Continuous integration

Po troch žiadostiach na Github a Travis bola schválená študentská organizácia s prístupom k Travis CI. VCS bolo následne premigrované z Bitbucketu na Github. Repozitár servera bol obohatený o Travis skript, ktorý automaticky nakonfiguruje systém s ohľadom na podmienky spustenia testov (databáza) a vykoná testovanie. CI bolo integrované do Githubu, merge pull requestu nie je možné vykonať, pokiaľ daná vetva neprešla úspešne testami. Github a Travis boli integrované do Slacku.

A Dokumentácia implementácie backendu

Dokumentácia vo formáte HTML je priložená ako zip archív.