

Moduly systému

Dokumentácia k tímovému projektu

Tímový projekt

Tím č. 21

Vedúci: Ing. Ivan Srba, PhD.

Členovia tímu:

Matej Groma
Matej Horváth
Peter Jurkáček
Jozef Kamenský
Adam Kňaze
Kristína Macková
Lenka Pejchalová
Jakub Sedlár

tim21.2018.fiit@gmail.com

Akademický rok: 2018/2019

Posledná zmena: 9. mája 2019

Obsah

1	Úvod	1
2	Kamera	1
2.1	Analýza	1
2.1.1	Hardvér	1
2.1.2	Komunikácia medzi kamerou a backendom	4
2.1.3	Zápisnica zo stretnutia s Martinom Tamajkom	5
2.1.4	Softvérové vybavenie kamery	7
2.1.5	OpenCV	9
2.2	Implementácia	9
2.2.1	Organizácia	10
2.2.2	Konfigurácia	10
3	Backend	12
3.1	Analýza	12
3.1.1	Databáza	12
3.1.2	Maven	12
3.1.3	Gradle	13
3.1.4	Záver	13
3.2	Návrh	13
3.2.1	Architektúra	13
3.2.2	Organizácia	13
3.3	Implementácia	13
4	Frontend	14
4.1	Analýza	14
4.1.1	Mapové riešenia	14
4.1.2	Frontendové frameworky	15
4.1.3	Porovnanie Metronic vs. AdminLTE	17
4.2	Návrh	19
4.2.1	Organizácia	19
4.3	Implementácia	20
4.3.1	Komponenty	20
4.3.2	Obrazovky	21
4.3.3	Lokalizácia	36
5	Testovanie	37
5.1	Testovanie backendu	37

6	Continuous integration	37
A	Zoznam príloh	38

1 Úvod

V tomto dokumente uvádzame dokumentáciu k modulom ktoré vznikli ako riešenie projektu TrafficWatch. Dokument je rozdelený na sekcie pre každý modul, a pre modul uvádzame separátne ich analýzy, návrh a implementáciu. Rozsah týchto častí sa odvíja od charakteru modulu. Následne uvádzame spis o testovaní a kontinuálnom nasadení projektu.

2 Kamera

2.1 Analýza

2.1.1 Hardvér

Hardvér pre kamerový systém

Od spolupracujúcich firiem máme alokovaný neobmedzený budget, ktorý bude použitý na preplatenie nákladov na hardvér. Na zvolení dosky 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**

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)

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 dost 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=ic8zDD0IjbQ>).

Firefly - AIO-3399J

Výhodou je slot na LTE modul. Výkon v benchmarku Geekbench je 1326 single-core a 2935 multi-core. Nevýhodou je slabá podpora 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.

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 kategórii by sa asi dal už kúpiť aj TX1), GPU akceleráciu by sme mohli chcieť (veľmi slabý výkon YOLO na CPU).

Sieťové moduly

Existujú špecializované produkty pre rôzne scenáre (odpočty), komunikačný modul však v našom prípade musí byť schopný preniesť vlastnú správu kvôli flexibilitě. 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-tx1/>). 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

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 . Oblíbená 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-Object-Detection-etc>). 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 (heurka).

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 (heurka).

Iné vhodné kamery

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

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.

2.1.2 Komunikácia medzi kamerou a backendom

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

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

Z trojice vhodných bol ďalej 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-of-data/37333>) vyradený HBMQTT. Zvolený bol najskôr VerneMQ, ktorý sa úspešne používa aj pri vyšších záťažach. 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.

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 -connect team21-18.studenti.fiit.stuba.sk
:8883:443 -prexit 2>/dev/null) -out ~/certificate.crt
sudo keytool -importcert -file ~/certificate.crt -alias example -keystore
$(/usr/libexec/java_home)/jre/lib/security/cacerts -storepass changeit
```

Python

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

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

2.1.3 Zápisnica zo stretnutia s Martinom Tamajkom

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

- threshold
- 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

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-verification/>

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)

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"

Odkazy

https://docs.google.com/document/d/1XS1cn4ZJxyspVyOX-_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-vision-and-opencv-166937911660>

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

2.1.4 Softvérové vybavenie kamery

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 možnosti

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>

Tolko modely, nás však zaujíma aj implementácia, chceli by sme niečo štandardné, otestované, 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, ťažšie 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)).

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 dost. https://github.com/datitran/object_detector_app
- s použitím GPU (NVIDIA computation capability len trochu menej ako Tegra X1 z Jetsonu) som na inej implementácii dostal 15 FPS čo už je použiteľná hodnota. Potvrďuje to, že na na zariadení s GPU vieme použiť aj čisto neurónkový detekčný model.
- pohľadal som pár ľudí čo robili detekciu priamo na Jetsonu 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

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 zlepšime kvalitu, osobne však pochybujem že takto dosiahneme dostatočnú kvalitu. Ak má byť náš nástroj použiteľný musíme dosiahnuť aspoň nejakú minimálnu presnosť, a čisto neurónkový model na embedded zariadení bežiaci realtime to podľa mňa nedáva.

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

Čo ďalej

Bez neuróniek to nepôjde, chceme rozoznávať typy áut a to klasickými CV algoritmi (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 dost 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 nejakým spôsobom vymyslieť splitovanie
- niečo iné?

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

Datasety

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

- Detrarc challenge - anotované data z čínskych ciest, zábery z dopravných kamier <http://detrac-db.rit.albany.edu/download>
- Nvidia Ai city challenge - 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/knwnng/awesome-vehicle-re-identification>

2.1.5 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 pravdepodobne zrýchly, testovanie prebiehalo na CPU.

Postrehy:

- Parametre do OpenCV funkcií (pre rôzne videá treba nastaviť 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

2.2 Implementácia

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++.

2.2.1 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 počítačové 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. Optionally môže byť zapnuté aj trackovanie kľúčových bodov pomocou algoritmu Lucas-Kanade sparse optical-flow. To umožňuje zlepšenie trackovania vozidiel pri prekryvoch.
- **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.
- **livestream** - Modul zabezpečujúci funkčnosť živého prenosu detekovaného obrazu z kamery na frontend.
- **utils** - Modul obsahuje pomocné funkcie používané pri detekcii objektov, počítaní prejazdov a vizualizácii v ostatných moduloch
- **CudaWatch** Pre optimalizáciu kódu a akceleráciu na grafickej karte beží časť kódu v C++. Komunikáciu medzi Pythonom a C++ zabezpečuje wrapper pbcvt (využívajúci knižnicu Boost.Python). Momentálne je v tomto module implementovaná celá funkcionálna detektoru a používa OpenCV funkcie akcelerované na grafickej karte pomocou CUDA.

2.2.2 Konfigurácia

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, respektíve číselná hodnota pri použití kamery (id nahrávacieho zariadenia).
- **video_resize** - Umožňuje zmeniť rozlíšenie videa ktoré prichádza na vstupe na akýkoľvek rozmer hodnotou [sirka, vyska]. Hodnota null zabezpečí, že video ostane nezmenené. Zmena rozlíšenia v programe je pomerne neefektívna, tento parameter je preto lepšie nepoužiť ak to nie je potrebné.
- **bg_sub_history** - počet framov, ktoré sa uchováajú 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 polom vrcholov (dvojice X a Y súradníc), ktoré v obraze túto oblasť tvoria.
- **counter_path_size** - počet bodov uloženej trasy vozidla. Čím viac bodov, tým budú uložené a 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
- **sampling_delay** - Interval v ktorom sa sampluje trajektória vozidla. S transitom sa odosiela aj trajektória vozidla, tento parameter určuje v sekundách ako často sa uloží jeho poloha.
- **keypoint_tracking** - zapína/vypína trackovanie kľúčových bodov pomocou algoritmu LK optical flow. Zlepšuje sledovanie vozidla pri prekryvoch, avšak pre svoju náročnosť citeľne znižuje FPS.
- **mqtt_...** - skupina parametrov spravujúcich komunikáciu kamery so serverom prostredníctvom mqtt. Nastavujú adresy, porty...
- **mqtt_prefix** - identifikuje inštanciu backendu (pre rozlíšenie ak beží viacej); zhodné s parametrom `deployment_prefix` pri nasadzovaní servera
- **camera_id** - id zariadenia na serveri, potrebné pre komunikáciu (na serveri je to `device_id`)
- **area_id** - id sledovanej oblasti na serveri
- **vis_save_vid** - umožňuje uložiť nahrané a anotované video
- **headless** - pri nasadení zariadenia v teréne funguje v režime bez používateľského rozhrania (nebeží X server), toto nastavenie vypína všetky takéto rozhrania
- **CUDA** - Kód momentálne obsahuje tri možnosti akcelerácie kódu na grafickej karte. Pri nastavení tohto parametra na null žiadna akcelerácia neprebíha, všetko beží v Pythone. Pri hodnote "native" sa použijú native Python cuda bindigy. Tie ale ešte nie sú oficiálne vydané, neexistuje k nim dokumentácia a implementovali sme s nimi len background substracion. Pri nastavení "pbcvt" sa využíva wrapper na volanie C++ kódu v ktorom je možné používať cuda verzie niektorých funkcií a tak celkovo bežať v C++. Momentálne je v ňom implementovaný celý detector (viď. súbor `detector_cuda.py`). Toto je podporovaný a ďalej vyvíjaný detector modul.

3 Backend

3.1 Analýza

3.1.1 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 dispozícii 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()`

Porovnanie manažérov závislostí

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

3.1.2 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

3.1.3 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

3.1.4 Záver

Gradle je silnejší nástroj, dá sa s 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.

3.2 Návrh

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

3.2.1 Architektúra

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

3.2.2 Organizácia

Projekt je organizovaný do nasledujúcich balíkov:

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

3.3 Implementácia

Dokumentácia pre implementáciu backendu bola automaticky vygenerovaná nástrojom Javadoc. Dokumentácia pre API rozhrania bola vygenerovaná nástrojom Swagger. Výstupy týchto nástrojov sú priložené v samostatných dokumentoch.

4 Frontend

4.1 Analýza

4.1.1 Mapové riešenia

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

Google Maps

API:

1. JavaScript
2. Android SDK
3. iOS SDK

Ponúkané služby, ktoré môžeme 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 kľúč (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 navyše (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 môžeme 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 funkcionálnosť, 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.

4.1.2 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 môžeme riešiť aj s product ownermi, ktorí nám slúbili pomoc.

4.1.3 Porovnanie Metronic vs. AdminLTE

Metronic

Výhody:

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

Nevýhody:

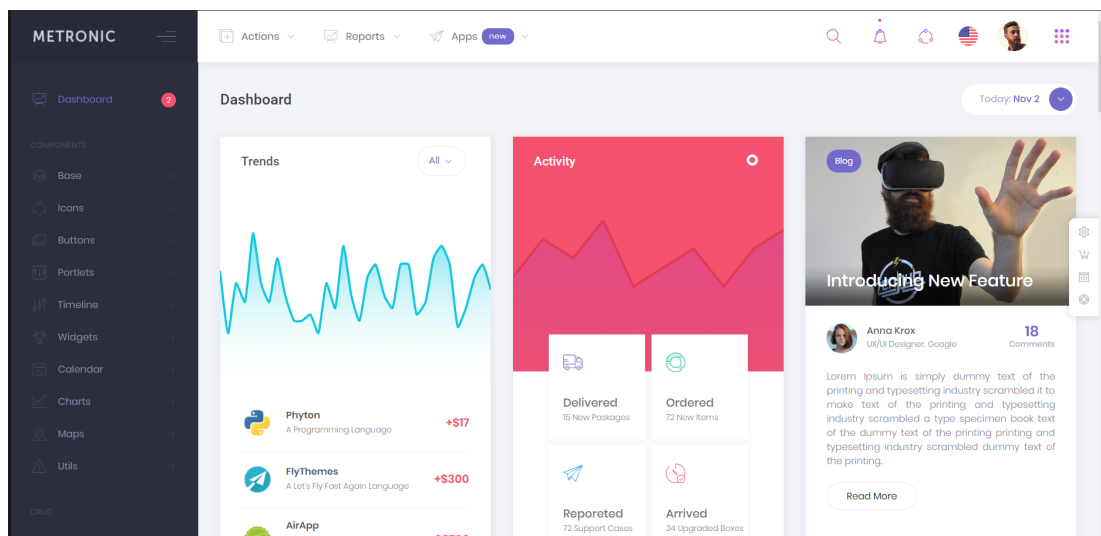
- nie je priamo integrovateľný s React-om
- spoplatnený (35\$), neponúka žiadne voľné šablóny

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



Obr. 1: Metronic.

AdminLTE

Výhody:

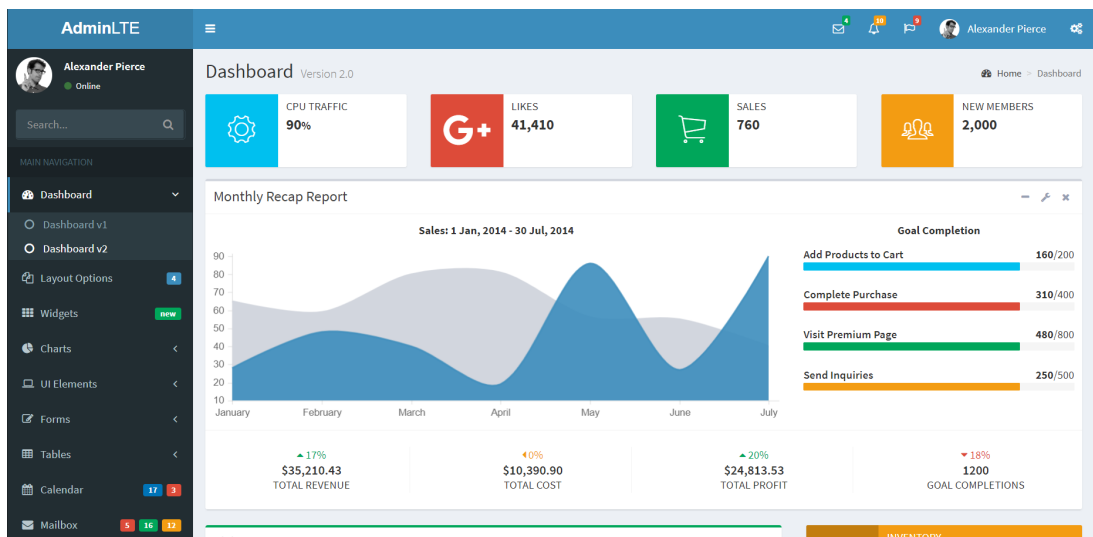
- má vysoké rozlíšenie
- má veľké množstvo widgetov, diagramov, atď.
- ponúka aj šablóny zadarmo (študentská MIT licencia)

Podpora prehliadačov:

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

Kompatibilita:

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



Obr. 2: AdminLTE.

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

4.2 Návrh

4.2.1 Organizácia

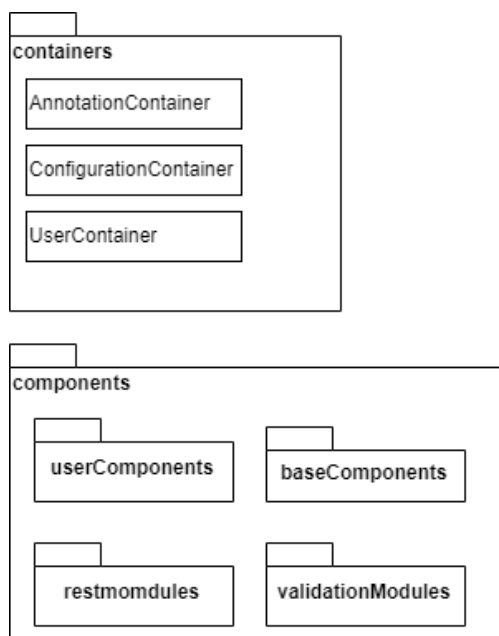
V tejto sekcii priblížime organizáciu frontendovej aplikácie projektu TrafficWatch.

Funkcionalita aplikácie je rozdelená na znovupoužiteľné komponenty. Univerzálna šablóna pre celú stránku je:

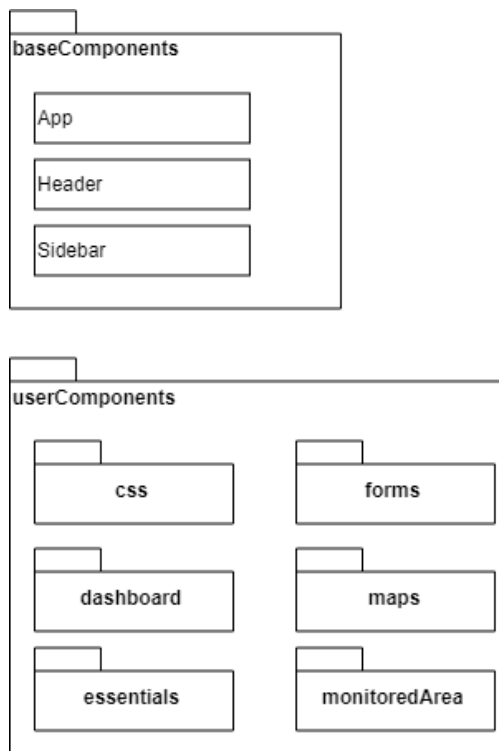
```
<div>  
  <Header/>  
  <SideBar/>  
  <UserContainer/>  
</div>
```

Listing 1: App.js

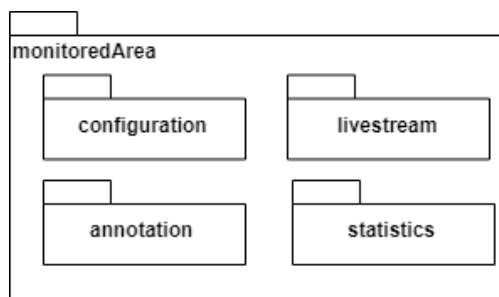
, kde containers zabezpečujú presmerovanie a načítanie potrebných komponentov. Organizácia komponentov je naznačená v diagramoch 3, 4 a 5.



Obr. 3: Primárna organizácia projektu je rozdelená do balíkov containers a components.



Obr. 4: Components sa ďalej delia na základné časti stránky a používateľské komponenty.



Obr. 5: Ďalšie balíky sú rozdelené logicky podľa obsahu sekcií stránky a sub-komponentov.

4.3 Implementácia

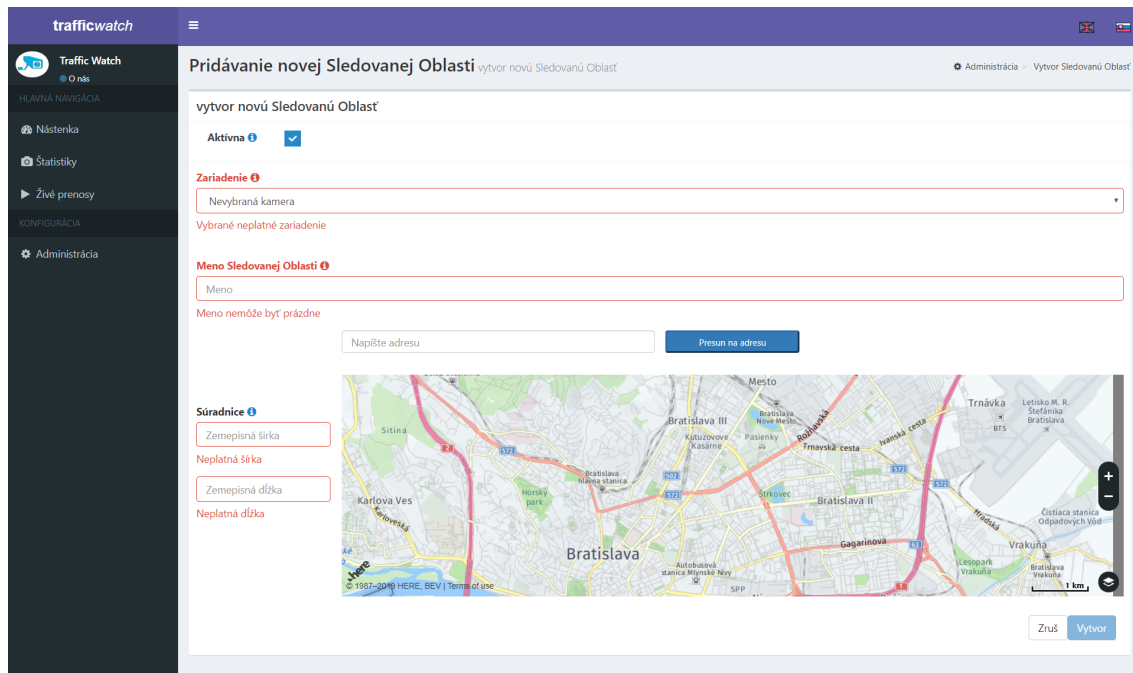
Frontend je v rámci nášho projektu oddelený od backend-u a je vytvorený pomocou aplikačného rámca React v programovacom jazyku javascript.

4.3.1 Komponenty

Dokumentácia komponentov bola automaticky vygenerovaná nástrojom JSDoc a je priložená ako .zip balík HTML súborov.

4.3.2 Obrazovky

V tejto časti predstavíme obrazovky webovej stránky nášho projektu:



Obr. 6: Obrazovka pre pridanie novej Sledovanej oblasti.

trafficwatch

Traffic Watch

HLAVNÁ NAVIGÁCIA

- Nástenka
- Štatistiky
- Živé prenosy
- KONFIGURÁCIA
- Administrácia

Anotácia Sledovanej Oblasti uprav & vytvor & nastav zóny Sledovanej Oblasti

Administrácia > Anotácia Sledovanej Oblasti

Anotácia Trnavske Myto

Ukáž mapu Ukáž pohľad kamery

© 1987–2019 HERE, BEVI | Terms of use

20 m

Tabuľka zón

Vytvor zónu

Číslo zóny	Meno zóny	Uprav
0	Smer Hlavna stanica	Uprav
1	Vjazd od Zimneho stadionu	Uprav
2	Vjazd z Trnavskeho myta	Uprav
3	Smer Zimny stadion	Uprav
4	Smer Mileticka	Uprav

Previous Page 1 of 1 5 rows Next

Zruš

Obr. 7: Obrázok anotácie Sledovanej oblasti - mapa.

trafficwatch

Traffic Watch
O nás

HLAVNÁ NAVIGÁCIA

- Nástenka
- Štatistiky
- Živé prenosy

KONFIGURÁCIA

- Administrácia

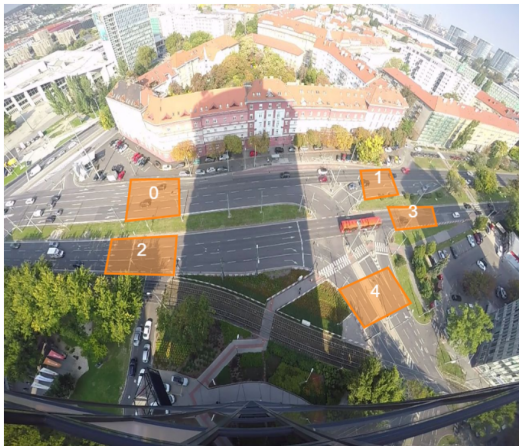
Anotácia Sledovanej Oblasti uprav & vytvor & nastav zóny Sledovanej Oblasti

Administrácia > Anotácia Sledovanej Oblasti

Anotácia Trnavske Myto

Ukáž mapu Ukáž pohľad kamery

Nová snímka



Tabuľka zón

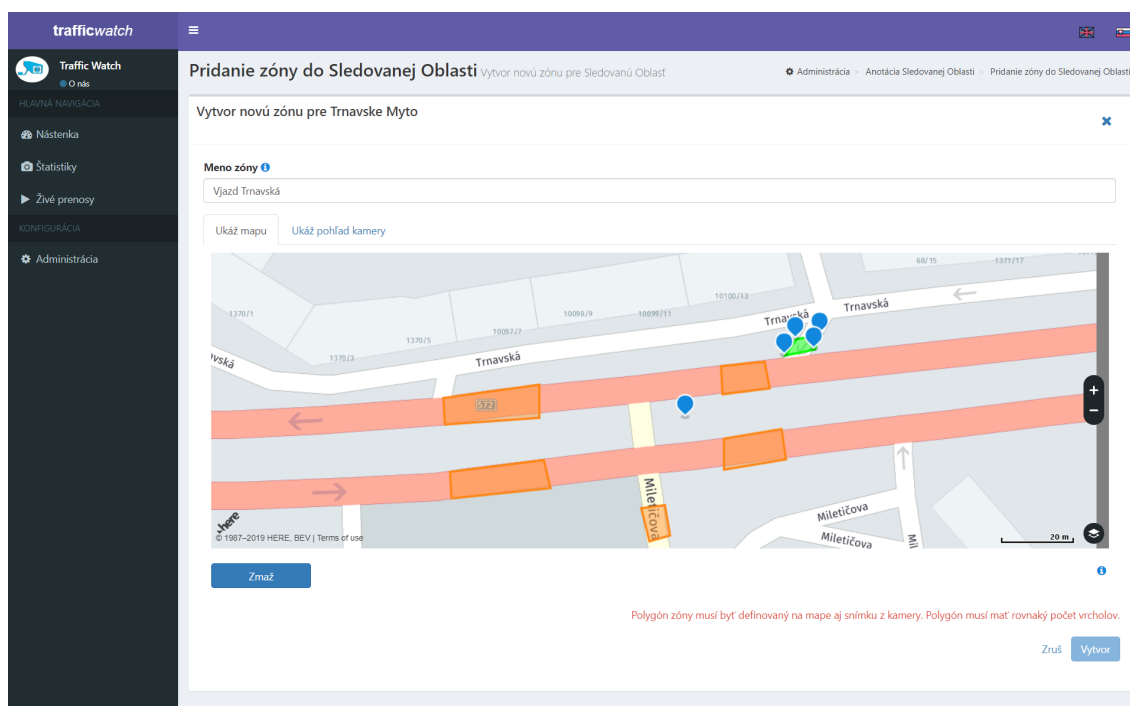
Vytvor zónu

Číslo zóny	Meno zóny	Uprav
0	Smer Hlavná stanica	Uprav
1	Vjazd od Zimneho stadionu	Uprav
2	Vjazd z Trnavskeho myta	Uprav
3	Smer Zimny stadion	Uprav
4	Smer Mileticka	Uprav

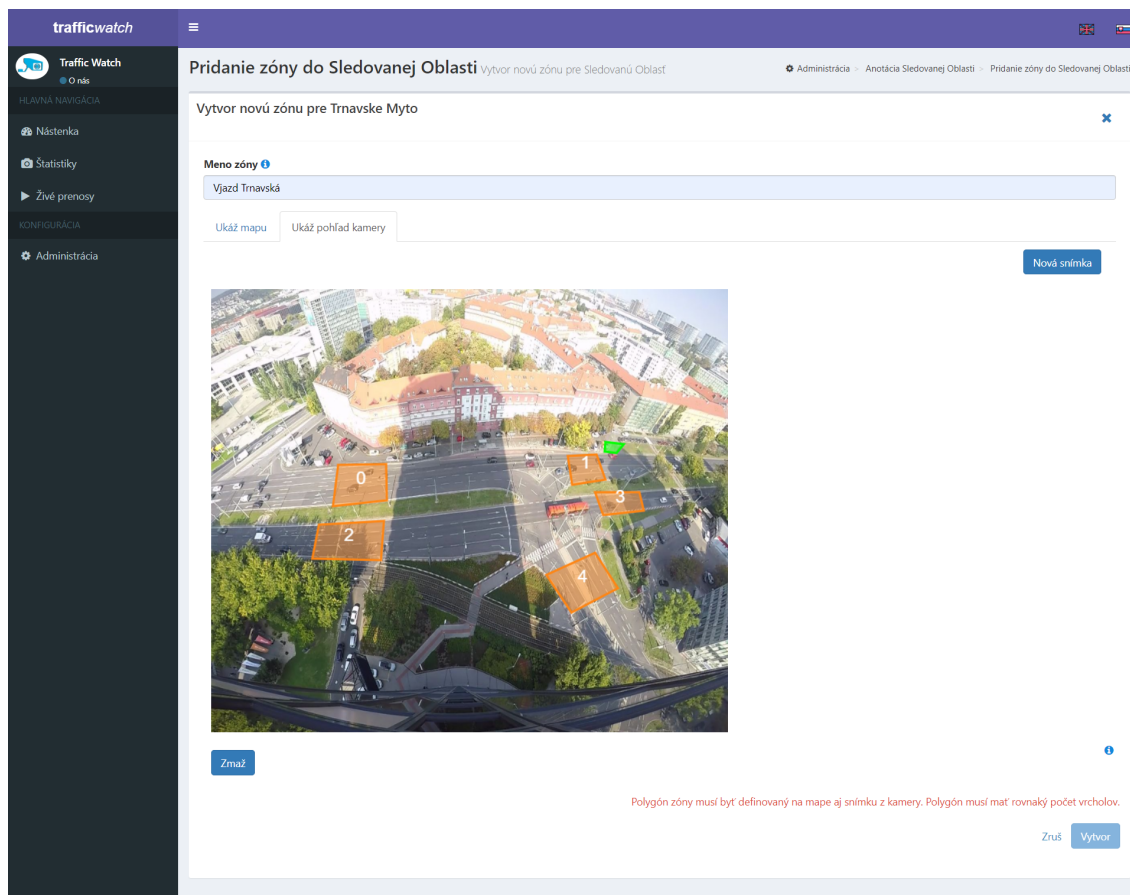
Previous Page 1 of 1 5 rows Next

Zruš

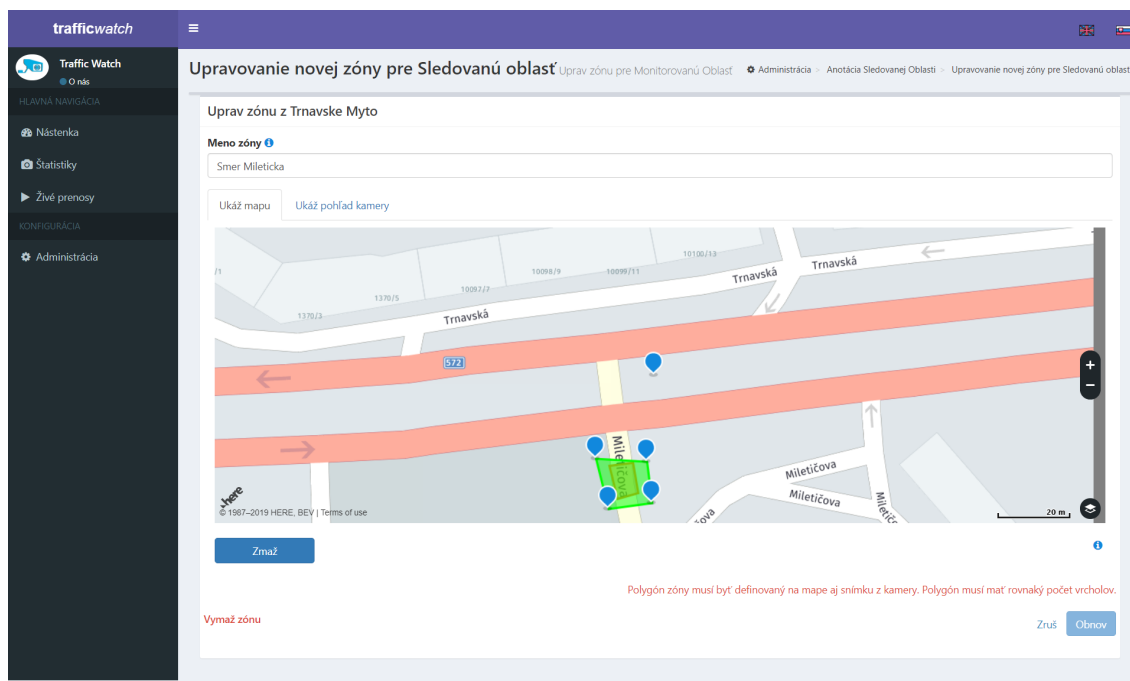
Obr. 8: Obrazovka anotácie Sledovanej oblasti - pohľad kamery.



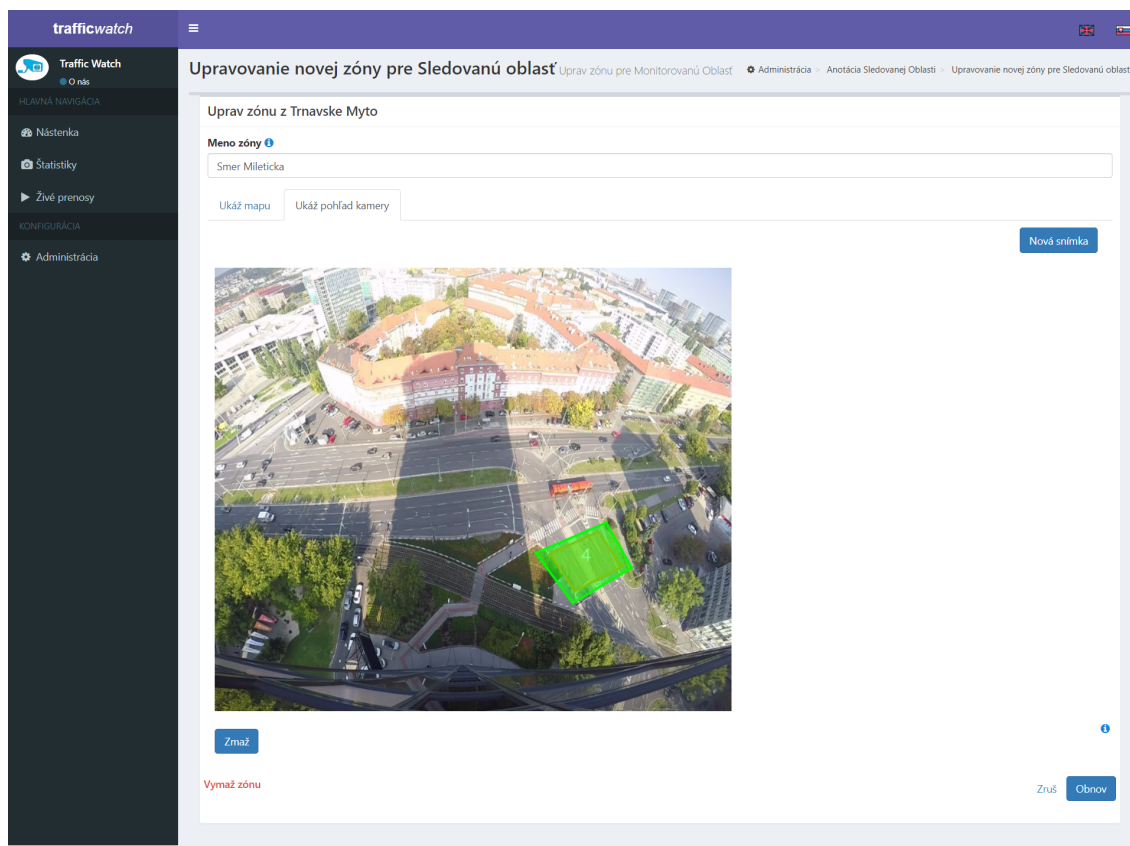
Obr. 9: Obrazovka na pridanie novej zóny do Sledovanej oblasti - mapa.



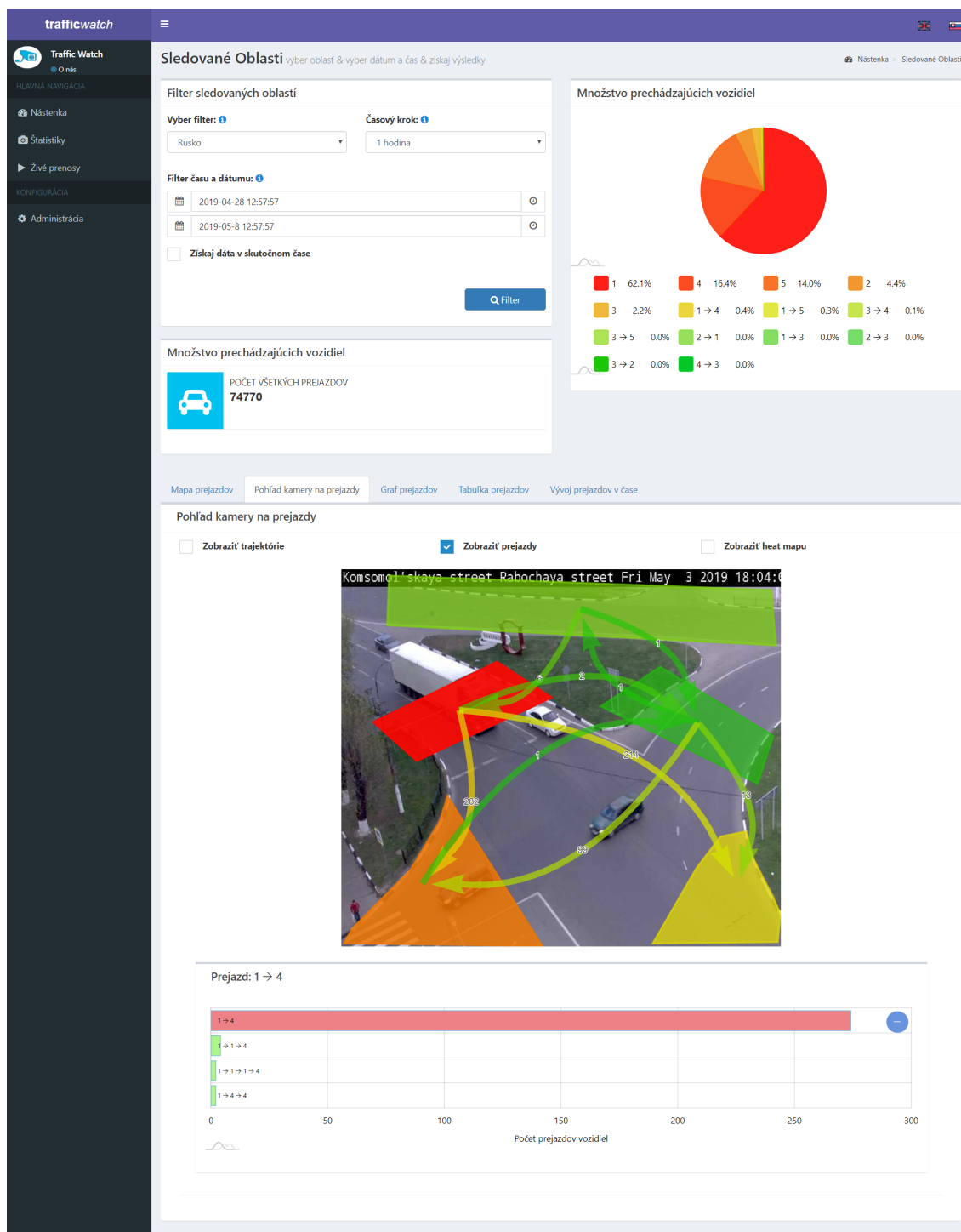
Obr. 10: Obrazovka na pridanie novej zóny do Sledovanej oblasti - pohľad kamery.



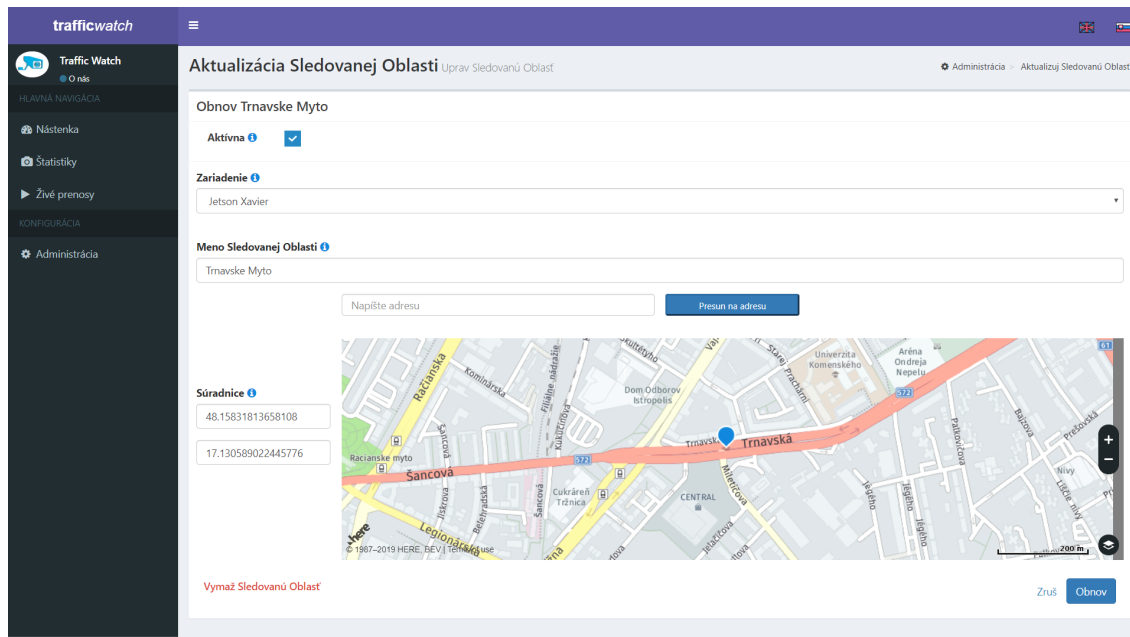
Obr. 11: Obrazovka na upravovanie novej zóny pre Sledovanú oblasť - mapa.



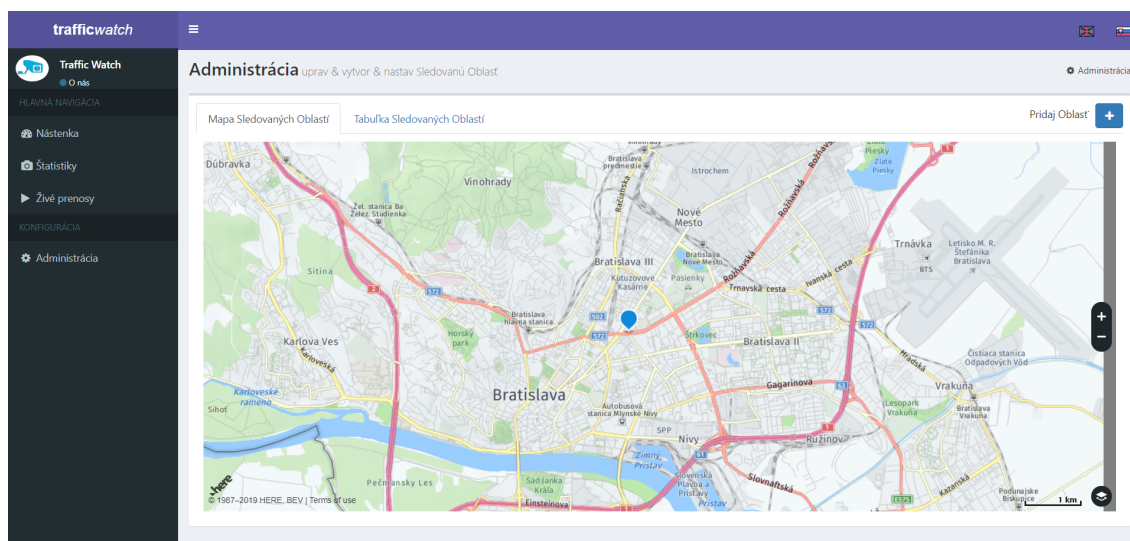
Obr. 12: Obrazovka na upravovanie novej zóny pre Sledovanú oblasť - pohľad kamery.



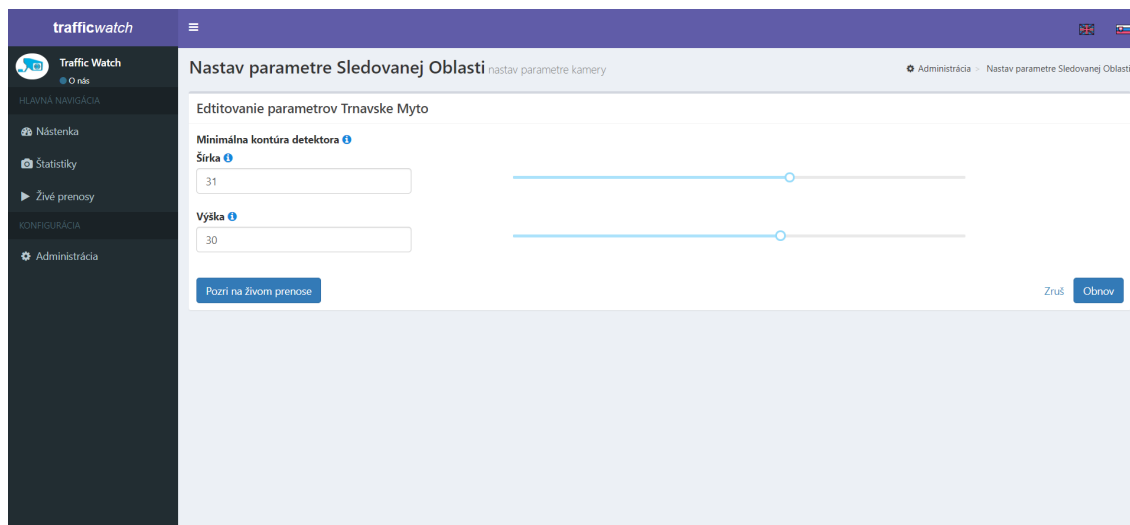
Obr. 13: Obrazovka zobrazujúca štatistiky zvolenej sledovanej oblasti. Zvolené štatistiky, v tomto prípade prejazdy, sú vizualizované pomocou Pohľadu kamery.



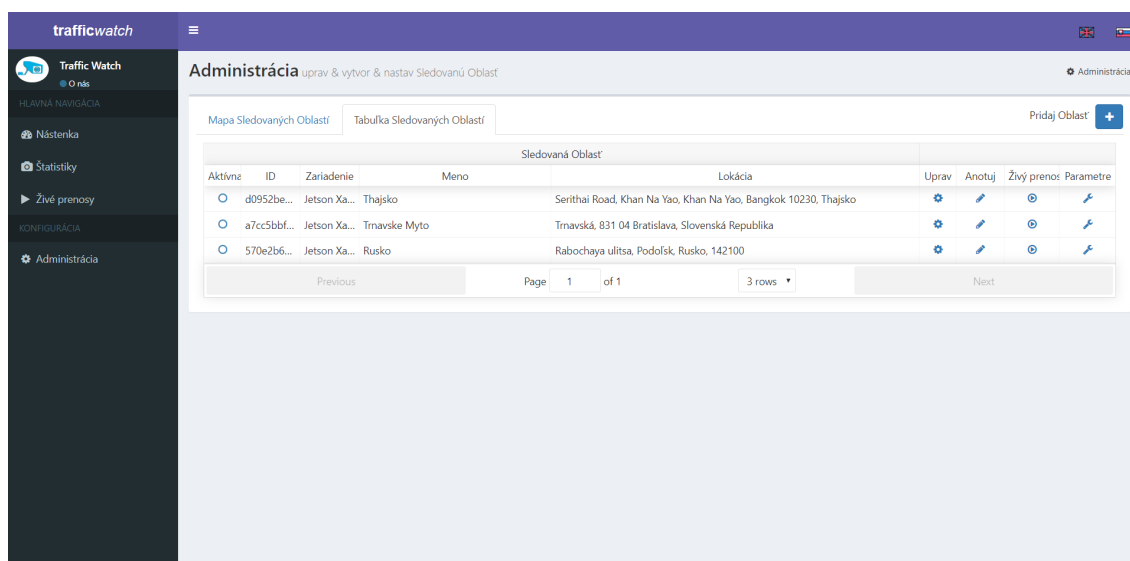
Obr. 14: Obrazovka na aktualizovanie, resp. editáciu informácií Sledovanej oblasti.



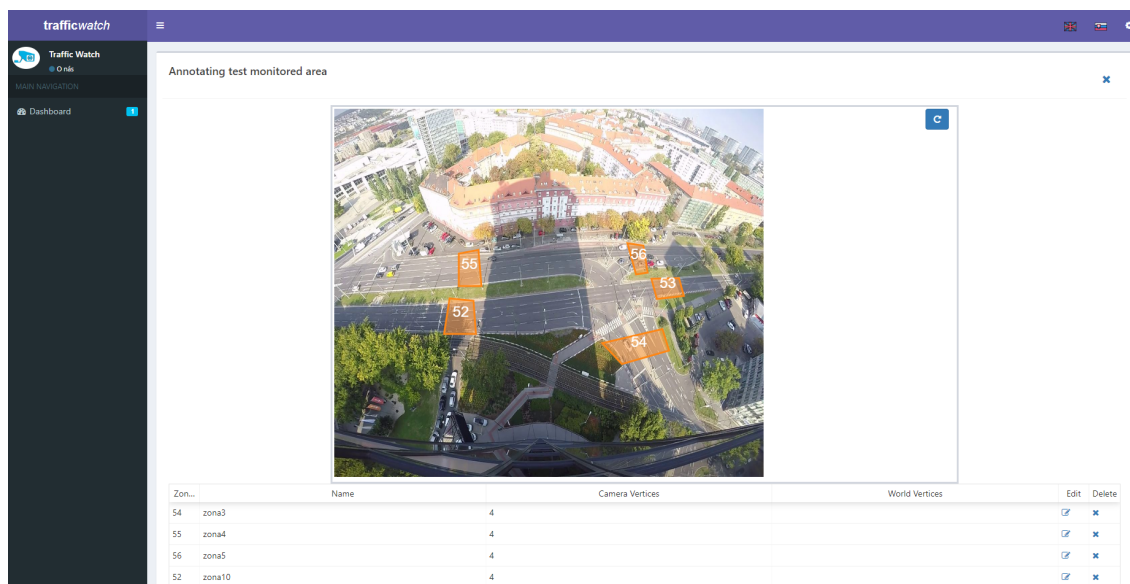
Obr. 15: Obrazovka na pridanie oblasti - mapa.



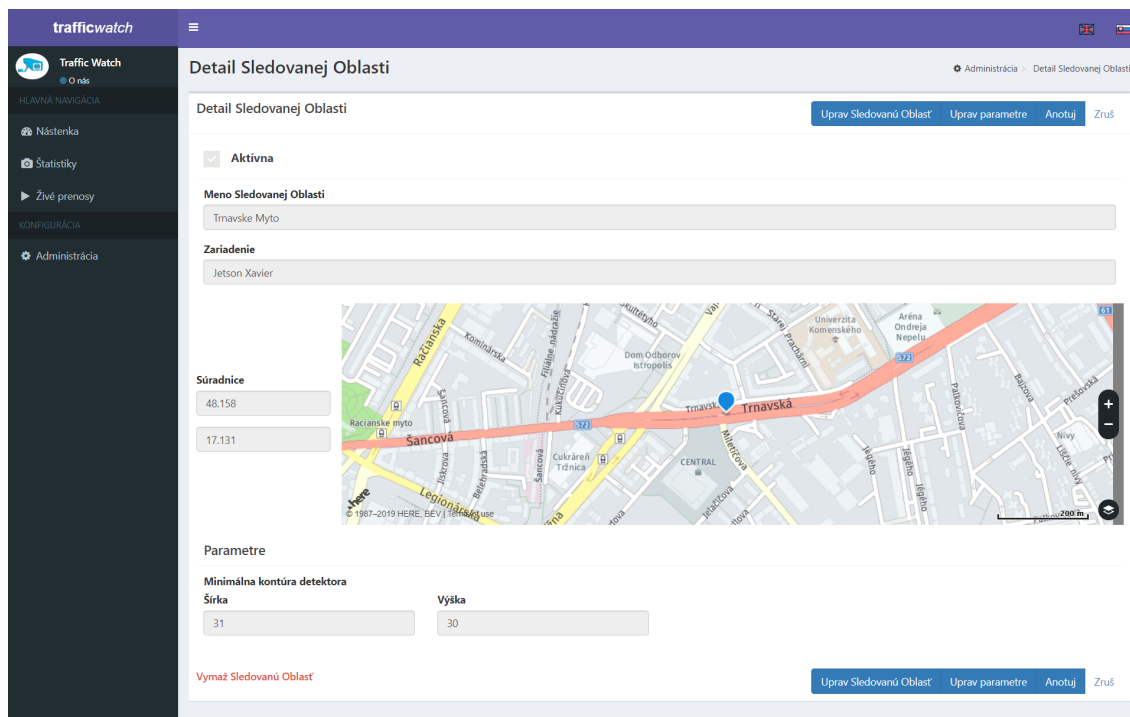
Obr. 16: Obrazovka slúžiaca na úpravu parametrov Sledovanej oblasti.



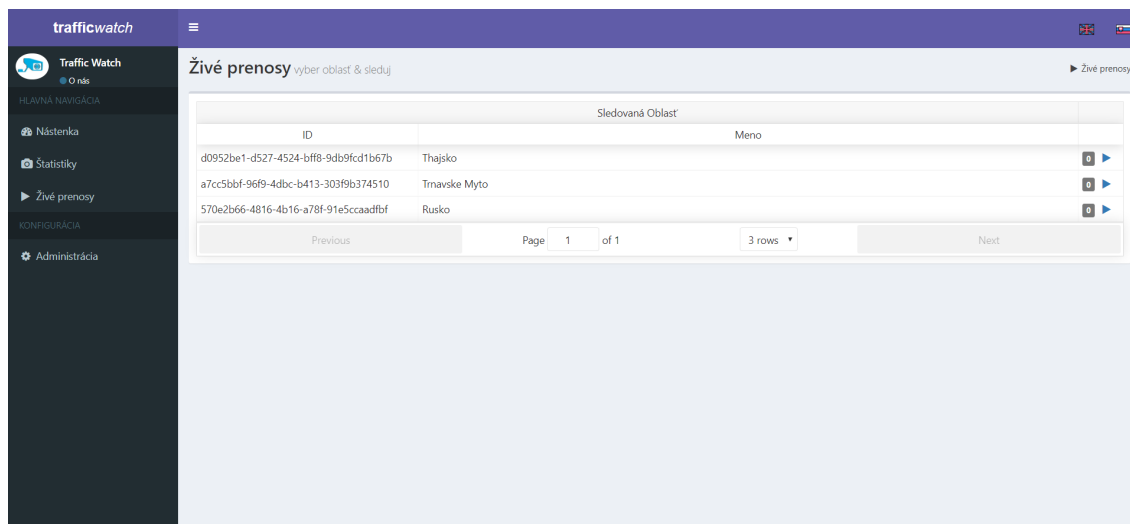
Obr. 17: Komponent pre konfiguráciu sledovaných oblastí, umožňuje oblasti pridávať, editovať, anotovať a vymazávať.



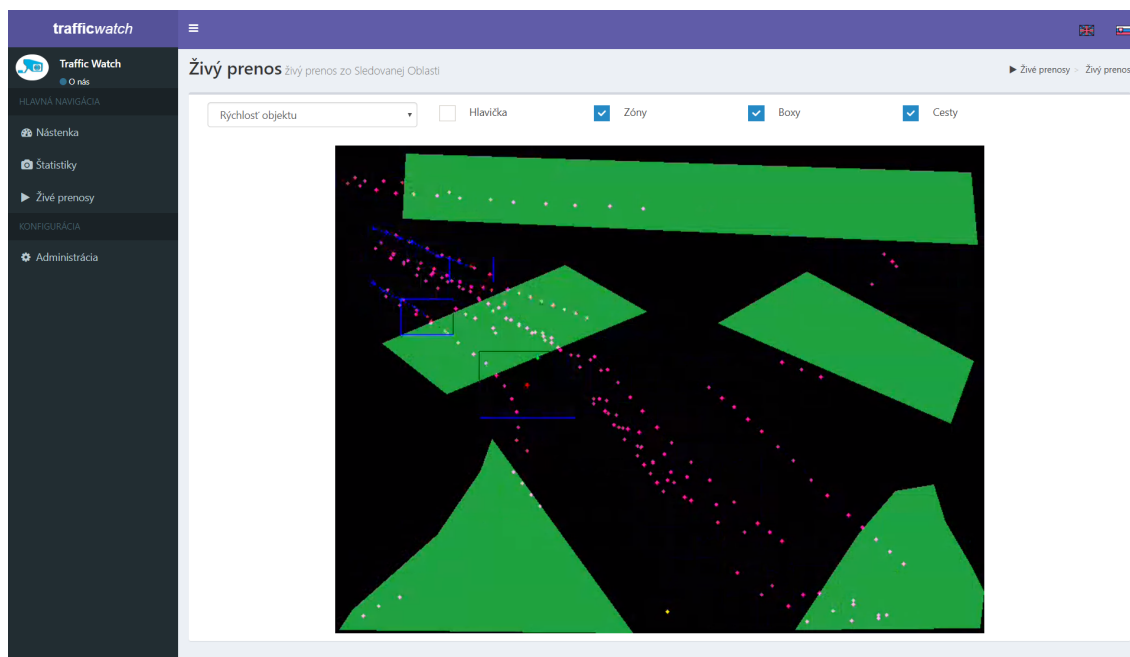
Obr. 18: Tento komponent slúži na konfiguráciu zón pre sledované oblasti, dostaneme sa sem kliknutím na ikonku "konfigurovať"(klúč) v riadku tabuľky sledovaných oblastí.



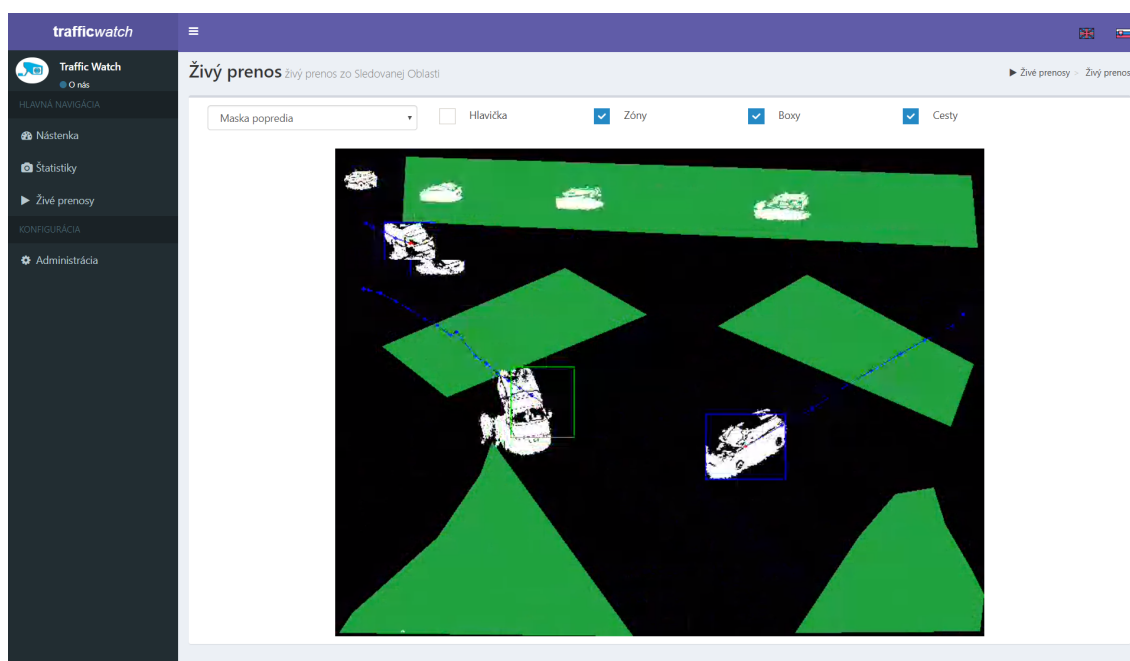
Obr. 19: Obrazovka zobrazujúca detaily Sledovanej oblasti, avšak bez priamej možnosti editovania.



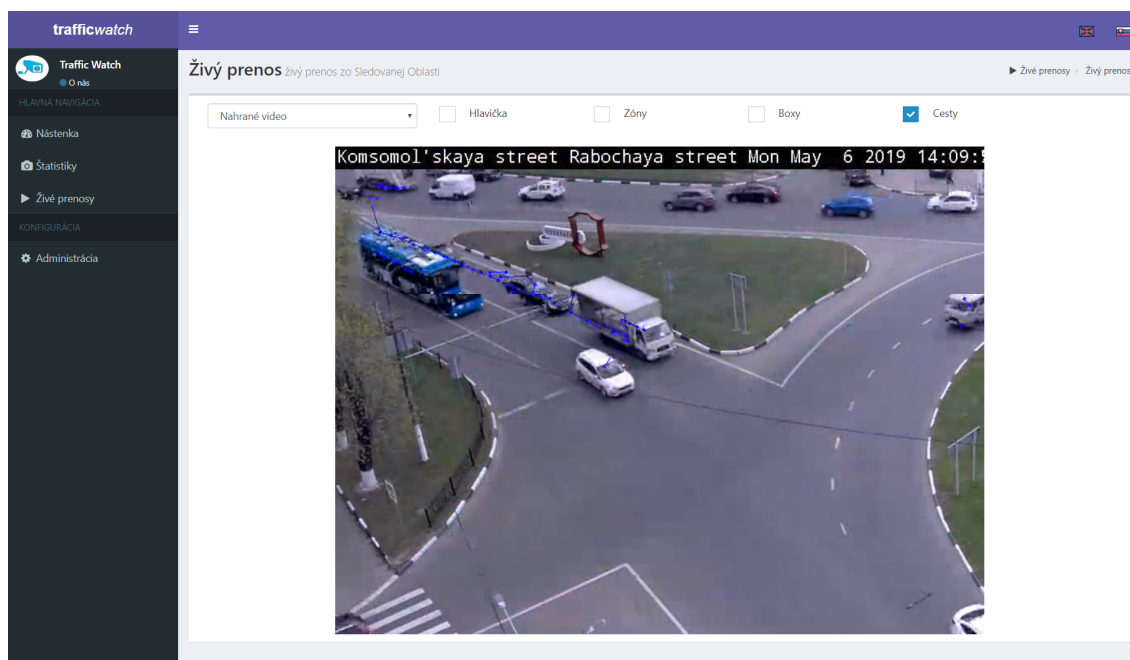
Obr. 20: Obrazovka ponúka zoznam živých prenosov z jednotlivých Sledovaných oblastí.



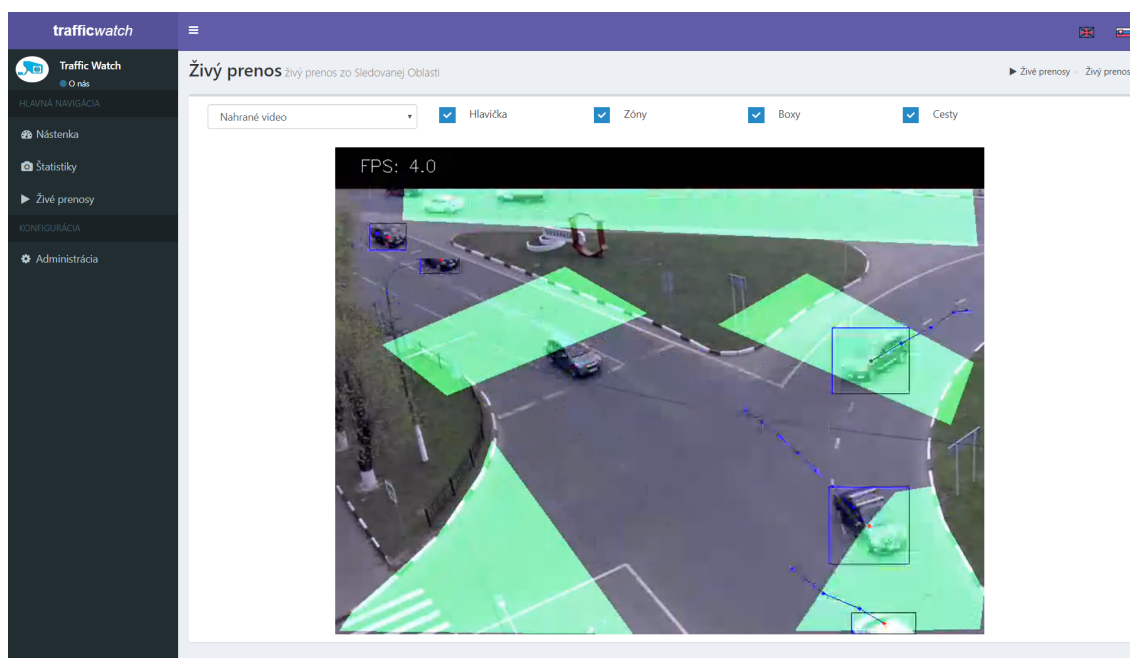
Obr. 21: Samotný livestream ponúka viaceré možnosti zobrazenia - ukážka rýchlosti objektu v kombinácií so zobrazenými zónami, boxmi a cestami.



Obr. 22: Samotný livestream ponúka viaceré možnosti zobrazenia - ukážka masky popredia v kombinácií so zobrazenými zónami, boxmi a cestami.



Obr. 23: Samotný livestream ponúka viaceré možnosti zobrazenia - ukážka skutočného pohľadu kamery v kombinácií so zobrazenými cestami.



Obr. 24: Samotný livestream ponúka viaceré možnosti zobrazenia - ukážka skutočného pohľadu kamery v kombinácii so zobrazenými zónami, boxmi, cestami a hlavičkou (informuje o FPS).

trafficwatch
☰
🇸🇰 🇨🇪

Traffic Watch

© O n&s

Sledované Oblasti vyber oblasť & vyber dátum a čas & získaj výsledky

🏠 Nástenka

📊 Sledované Oblasti

Filter sledovaných oblastí

Vyber filter: Trnavske Myto Získaj dáta v skutočnom čase

Filter času a dátumu: 2018-12-30 14:34:47 2019-05-3 14:34:47

[Filter](#)

Množstvo prechádzajúcich vozidiel

POČET VŠETKÝCH PREJAZDOV

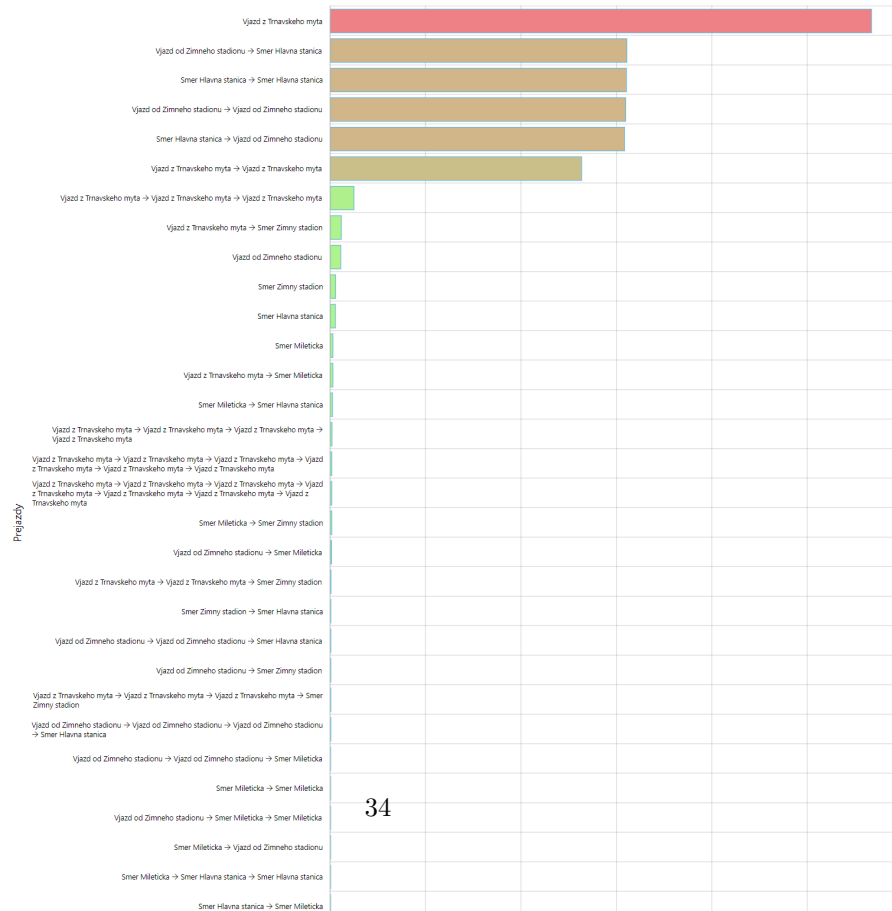
10725

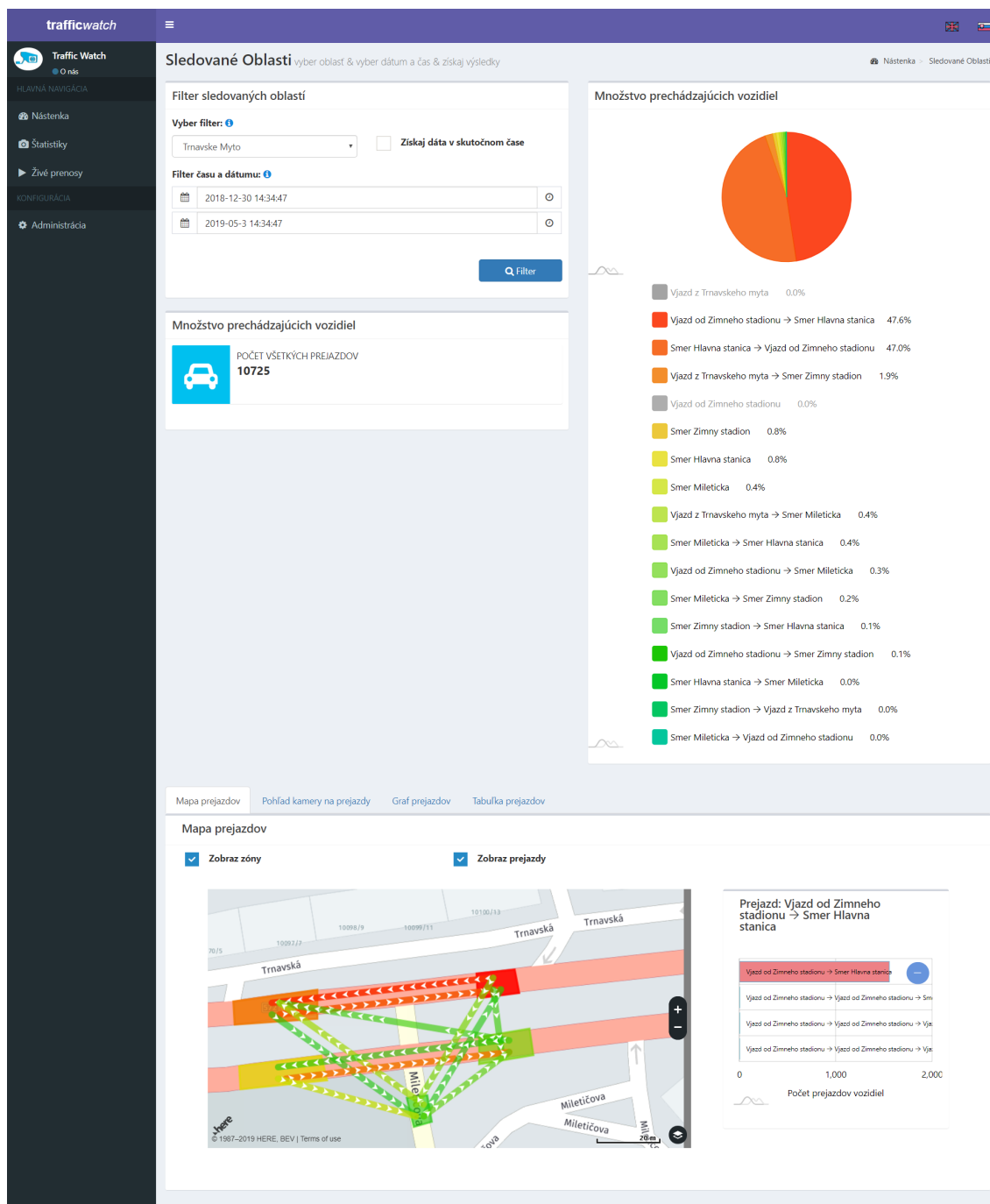
Množstvo prechádzajúcich vozidiel

Smery	Podiel (%)
Vjazd z Trnavského myta	0.0%
Vjazd od Zimného štadiónu → Smer Hlavná stanica	47.6%
Smer Hlavná stanica → Vjazd od Zimného štadiónu	47.0%
Vjazd z Trnavského myta → Smer Zimný štadión	1.9%
Vjazd od Zimného štadiónu	0.0%
Smer Zimný štadión	0.8%
Smer Hlavná stanica	0.8%
Smer Miletická	0.4%
Vjazd z Trnavského myta → Smer Miletická	0.4%
Smer Miletická → Smer Hlavná stanica	0.4%
Vjazd od Zimného štadiónu → Smer Miletická	0.3%
Smer Miletická → Smer Zimný štadión	0.2%
Smer Zimný štadión → Smer Hlavná stanica	0.1%
Vjazd od Zimného štadiónu → Smer Zimný štadión	0.1%
Smer Hlavná stanica → Smer Miletická	0.0%
Smer Zimný štadión → Vjazd z Trnavského myta	0.0%
Smer Miletická → Vjazd od Zimného štadiónu	0.0%

Mapa prejazdov
Pohľad kamery na prejazdy
Graf prejazdov
Tabuľka prejazdov

Graf prejazdov vozidiel





Obr. 26: Obrazovka zobrazujúca štatistiky zvolenej sledovanej oblasti. Zvolené štatistiky, v tomto prípade prejazdy, sú vizualizované pomocou Mapy prejazdov.

4.3.3 Lokalizácia

Na lokalizáciu bola integrovaná knižnica `react-i18next` (zvolená kvôli univerzálnosti knižnice `i18next` v prostredí webových aplikácií v rámci rozvoja schopností členov tímu do budúcnosti). Jazyky sú načítavané dynamicky (XHR) na základe voľby používateľa podľa pridaného selektora jazyka, prípadne predvolene podľa nakonfigurovanej autodetekcie podľa jazykového nastavenia webového prehliadača. Ďalej bol integrovaný nástroj `i18next-scanner` na automatickú extrakciu kľúčov z kódu pomocou príkazu `npm run translations` pre jednoduchšiu lokalizáciu. V rámci CI bola implementovaná kontrola, či je každý textový reťazec (kľúč) preložený.

5 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.

5.1 Testovanie backendu

Na strane serveru sú dostupné jednotkové testy, ktoré sa zameriavajú na overenie správneho fungovania 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.

6 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 Zoznam príloh

1. Dokumentácia Javadoc (backend)
2. Dokumentácia Swagger (backend)
3. Dokumentácia SchemaSpy (databáza)
4. Dokumentácia JSDoc (frontend)