Projekt č. VI20172020068

Nástroje a metody zpracování videa a obrazu pro zvýšení efektivity operací bezpečnostních a záchranných složek (VRASSEO)

Architektura systému, ViAn Server, Snímací uzel a ViAn GUI

Technická zpráva - závěrečná

Jaroslav Zendulka, Vladimír Bartík, Tomáš Volf, Radim Kocman

Vysoké učení technické
Fakulta informačních technologií
Božetěchova 1
Brno, 612 66, Česká republika

Prosinec 2020

Obsah

Abstrakt

Tato technická zpráva popisuje celkovou architekturu systému pro zpracování videa vyvíjeného v rámci projektu a jedné z jejích komponent - ViAn serveru. Tato komponenta je zodpovědná za správu uložených video dat a metadat z videí extrahovaných. Je zde popsána struktura databáze a dvě aplikační programátorská rozhraní sloužící pro ukládání video dat a extrahovaných metadat (ViAn Sensing API) a pro manipulaci se spravovanými daty (ViAn API).

Dále je zde popsáno rozhraní snímacích uzlů (SeNoAPI), tedy uzlů obsahujících snímací moduly, které extrahují metadata z videa. Pro asynchronní přenos metadat k ViAn serveru je využívána fronta MQ. Popis její správy je rovněž součástí zprávy.

Ve zprávě je zahrnut i popis jednoduché operativní aplikace ViAn GUI, která slouží k základní konfiguraci sestavy pro konkrétní případy použití a pro dotazování nad obsahem databáze a některé online detekce.

V případě softwarových komponent je zahrnut i instalační manuál.

1 Architektura systému

Zařízení VRASSEO je systém pro zpracování obrazových záznamů z různých obrazových zdrojů. Architektura systému uvedená na Obr. 1 je tvořena především třemi základními typy nástrojů:

1. `ViAn Server` - poskytuje správu video dat a metadat z nich extrahovaných.

2. Snímací modul - slouží k extrakci metadat z videa pořizovaného snímacím zařízením. Metadata se týkají především událostí různého typu. Příkladem může být modul extrahující informaci o výskytech objektů (auto, osoba) ve videu. Snímací modul nebo sada snímacích modulů spolu s dalšími podpůrnými nástroji tvoří `Snímací uzel.`

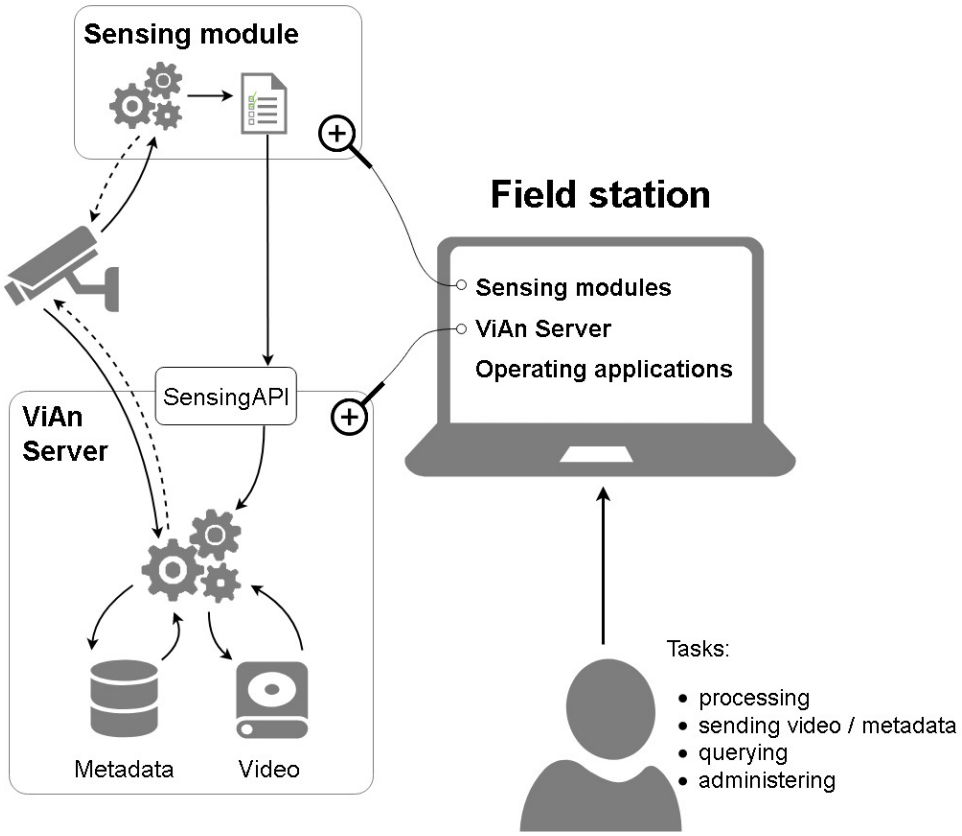
3. Aplikace - poskytuje vstup a výstup koncovému uživateli. Jednou z aplikací zahrnutých v systému je vždy tzv. operativní aplikace, která umožňuje základní správu a ovládání systému.

Systém vždy obsahuje jeden ViAn server a jednu Operativní aplikaci a může obsahovat několik snímacích modulů doplněných specifickými aplikacemi pro práci s extrahovanými metadaty. Snímací zařízení může být statické nebo mobilní. v případě mobilního zařízení jde o dron s kamerou a dalším vybavením, včetně pozemní stanice pro napojení k ViAn Serveru.

Systém je navržen s ohledem na potřebu jeho běhu na strojích s různým výpočetním výkonem. Architektura řešení umožňuje systém škálovat od instance s jednou kamerou a jedním snímacím modulem běžícím na běžném notebooku, až po distribuovaný systém analyzující data z více kamer na několika výpočetních strojích (od PC až po výpočetní server s GPU kartami). Systém lze např. využít ke sledování v náročnějších případech, kdy je potřeba analyzovat data jak z několika IP kamer (výpočetní server, vysoký datový přenos), tak z kamer v terénu bez možnosti rychlého přenosu velkých dat (výpočet na místě - kamera s vestavěným systémem nebo napojená na lokální počítač, přenos pouze metadat do systému) - viz Obr. 2.

Systém lze také libovolně nakombinovat na jakoukoliv kombinaci mezi řešením tzv. *"vše v jednom"* (obr. 1) a řešením, kdy je zvlášť operativní aplikace, zvlášť Snímací uzel a zvlášť ViAn Server. Lze tedy provozovat například operativní aplikaci s ViAn Serverem na stejném počítači a zvlášť Snímací uzel na výkonném počítači s GPU nebo kteroukoliv jinou kombinaci.

Koncept komunikace tří základních komponent (součástí / nástrojů) systému pro spuštění nímacího modulu a příjem dat na ViAn Server je zobrazena na Obr. 3. Nejdříve se přihlásí uživatel do applikace, která jej autentizuje vůči ViAn Serveru (1). Poté uživatel vybere stream pro zpracování, snímací uzel, na kterém se bude video zpracovávat a konkrétní snímací modul, kterým bude stream zpracováván. Tyto informace uloží aplikace

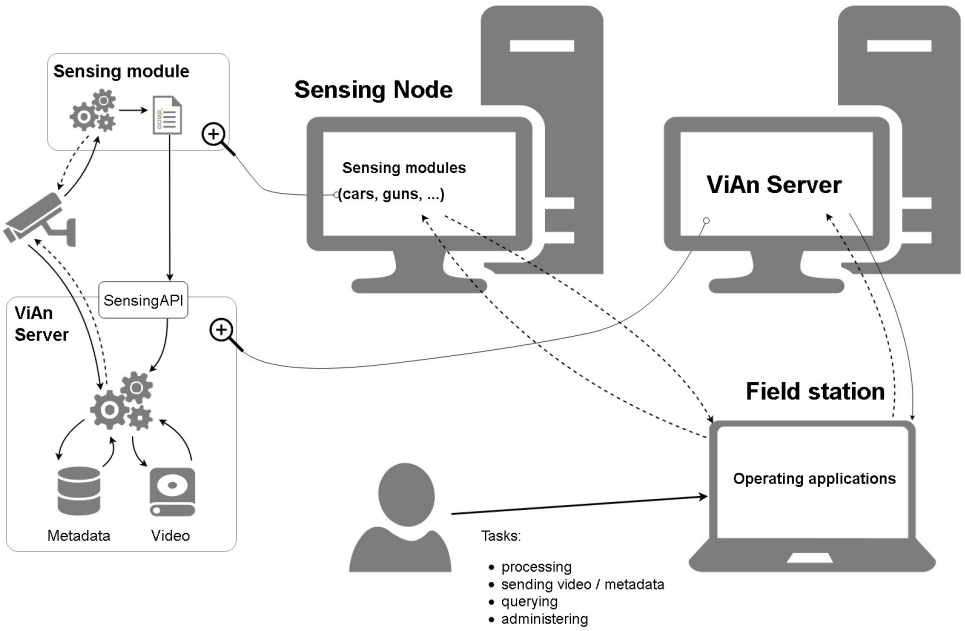


Obrázek 1: Základní architektura systému

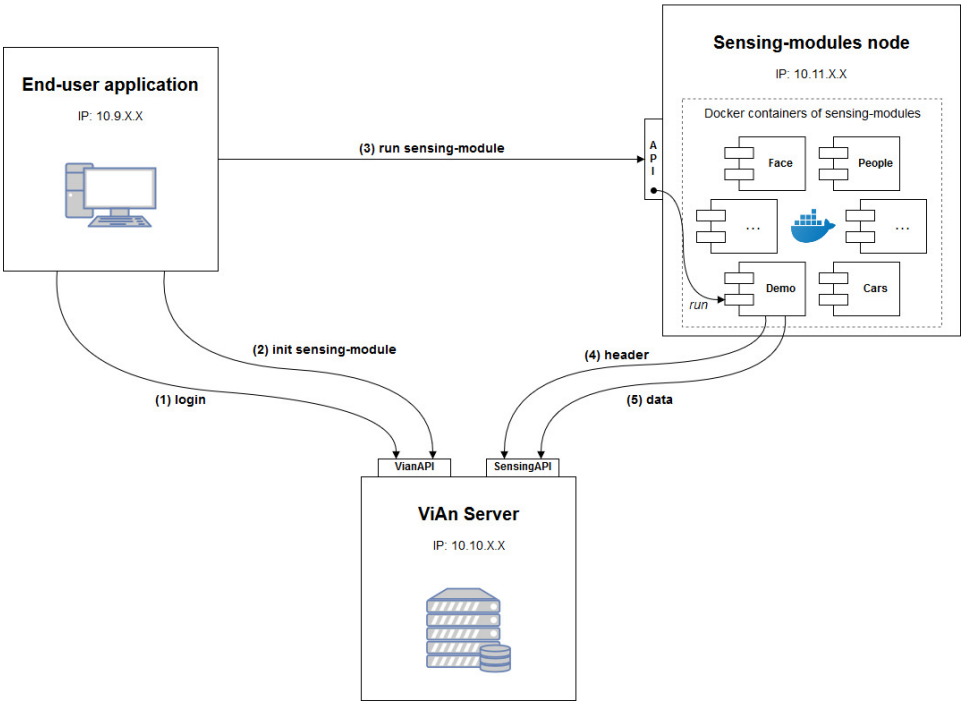
skrze API na ViAn Server a získá tak autentizační token pro snímací modul samotný (2). Následně pošle požadavek na spuštění daného modulu na vybraný Snímací uzel (3).

Snímací modul zasílá na ViAn Server nejprve hlavičku (4), v níž specifikuje strukturu dat (které vlastnosti bude ukládat a jakých datových typů jsou). Po odeslání hlavičky zasílá modul opakovaně data (5) detekovaných objektů.

Snímací modul může být zabalen do Docker kontejneru (preferovaná varianta), systém však dokáže pracovat i s modulem nacházejícím se fyzicky přímo na Snímacím uzlu (skript, binární soubor).



Obrázek 2: Architektura s distribuovaným zpracováním



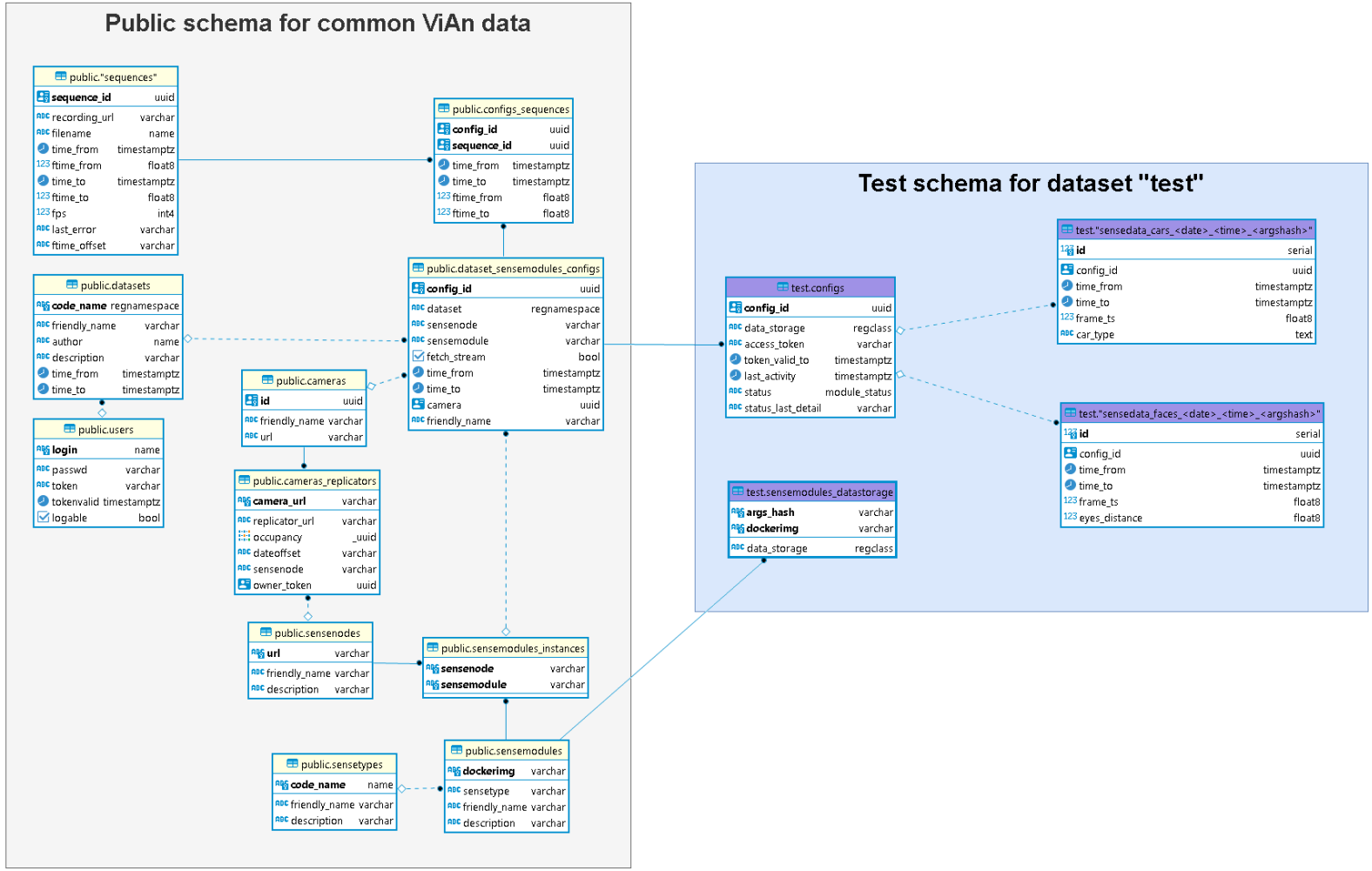
Obrázek 3: Diagram komunikace základních komponent systému

2 Databázové schéma

Podoba schématu databáze je zobrazena na Obrázku 4. Databáze umožňuje ukládání dat, které pocházejí z různých zdrojů poskytujících a zpracovávajících video data. Kromě toho obsahuje také tabulky určené pro ukládání informací o těchto zdrojích, tj. kamerách, snímacích modulech atd. It also contains tables designed for storage of information about the sources, e.g. cameras, sensing modules etc. k těmto účelům je využita databáze PostgreSQL.

ViAn databáze se skládá ze schématu `PUBLIC` a z dalších schémat, z nichž každé je určeno pro jeden dataset. Hlavní schéma (`PUBLIC`) obsahuje obecné informace o jednotlivých datasetech, uživatelích, kteří je vytvořili, kamerách a snímacích modulech. Informace o snímacích modulech zahrnují tabulky pro typy snímacích modulů (`Sensetypes`), snímací moduly samotné (`Sensemodules`) a instance jednotlivých snímacích modulů (`Sensemodules_instances`). Tyto instance jsou seskupeny v tzv. snímacích uzlech (`Sensenodes`). Informace o jednotlivých detekcích, které jsou prováděny snímacími moduly, jsou uloženy v tabulce `Dataset_sensemodules_configs`. Pro každou detekci je zde např. informace o kameře, která je zdrojem vstupních dat pro detekci, dataset, kam se výsledky detekce ukládají nebo čas začátku a konce detekce. Atribut `fetch_stream` indikuje, zda se společně s danou detekcí má na server ukládat i video sekvence z kamery. Pokud je video ukládáno, pak informace o uložených video sekvencích (např. název souboru atd.) je uložena v tabulce `Sequences`. Protože pro jednu detekci může být uloženo i více sekvencí, je ve schématu vložena vazební tabulka `configs_sequences`. Informace o replikátorech a jejich využití je pak uložena v tabulce `cameras_replicators`.

Pro každý dataset je v databázi PostrgeSQL vytvořeno nové schéma. Jeho název je odvozen od názvu dataset (atribut `code_name` v tabulce `Datasets`. Hlavní tabulkou v každém z těchto schémat je tabulka `Configs`, která je cizím klíčem propojena s tabulkou `Dataset_sensemodules_configs` ve schématu `PUBLIC`. Zde je uložena např. informace o přístupovém tokenu a jeho platnosti. Výstupní data ze snímacího modulu jsou uložena v tabulkách, které jsou v diagramu označeny jako `Sensedata_cars_XXX` a `Sensedata_face_XXX`. Názvy těchto tabulek jsou odvozeny od hodnoty atributu `data_storage`, který je také součástí tabulky `Configs` table. Jestliže dojde ke změně verze snímacího modulu a změní se tím struktura výstupních dat modulu, informace o struktuře dat staré verze, včetně atributů a datových typů, je uložena v tabulce `Sensemodule_datastorage`.



Obrázek 4: Návrh databáze na ViAn Serveru

3 Systém zpracování videa

Tato sekce popisuje způsob zpracování video dat v rámci celého systému VRASSEO. Jsou zde představeny vybrané důležité pojmy, používané technologie i využívané principy.

3.1 Základní pojmy

Implementovaný systém zpracování videa lze z hlediska popisu pojmů rozdělit na čtyři hlavní části. Každá část je zde vždy nejdříve ve zkratce představena a následně jsou doplněny podrobnější technické detaily.

Video streamy

Uvažujeme-li video data putující systémem, ať už se jedná o data z vnějšího zdroje (kamera, dron apod.) nebo data putující mezi různými částmi systému (ViAn server, snímací moduly apod.), jsou vždy přenášena ve formě video streamů.

Pro vnitřní komunikaci mezi jednotlivými částmi systému je využíván šifrovaný streamovací protokol RTMPS (Real-Time Messaging Protocol over a TLS/SSL connection). Video streamy přicházející z vnějšího zdroje mohou být obecně v libovolném zavedeném streamovacím formátu, který dokáže zpracovat knihovna ffmpeg, ale na vstupu do systému jsou vždy automaticky převáděny do formátu RTMPS. Testování převodu video streamů probíhalo primárně s RTSP H.264 streamy, které představují běžný výstupní formát IP kamer.

Video soubory

Pokud je nutné video streamy v systému archivovat pro pozdější použití, jsou ukládány ve formě video souborů na ViAn serveru. Obsah těchto souborů pak může být ViAn serverem zpětně přehráván (např. do operativní aplikace pro zobrazení detekovaných událostí).

Pro ukládání využívá systém standardní video formát FLV (Flash Video), který umožňuje přímé ukládání video streamů formátu RTMPS bez nutnosti dodatečné konverze dat.

Časové značky s reálným časem

Události, které snímací moduly detekují ve video streamu, je nutné propojit s konkrétním reálným časem i konkrétním snímkem videa. Pro tyto účely jsou v systému využívány časové značky s reálným časem, které se přenášejí jako součást video streamů.

Pro přesnou reprezentaci reálného času využívá systém zpracování videa unixové časové značky. RTMPS streamy ale pracují pouze s 32-bitovými

časovými značkami a nedokáží tak pojmout plné unixové časové značky zachycující reálný čas včetně milisekund. Časové značky přenášené uvnitř RTMPS jsou proto modifikované tak, aby reprezentovaly pouze hodnotu času počítanou od začátku dne, kdy byl daný stream spuštěn. V databázi je poté navíc uložena dodatečná informace o čase spuštění každého streamu a plné unixové časové značky jednotlivých snímků videa je tak možné zpětně přesně zrekonstruovat.

Video krabičky

Pro sjednocení práce s videem napříč celým systém byla vytvořena sada před-připravených funkčních jednotek, tzv. video krabiček, které se starají o provádění komplexnějších operací nad video daty (např. konverze formátu video streamu, doplnění časových značek s reálným časem apod.). ViAn server a snímací uzly následně využívají tyto krabičky pro zpracování potřebných úloh. Struktura video krabiček je detailněji rozebrána v navazující samostatné sekci.

3.2 Struktura video krabiček

Video krabičky jsou navrženy jako zapouzdřené Docker kontejnery, které dokáží samostatně plnit svoji požadovanou funkci. Obsah těchto Docker kontejnerů můžeme obecně rozdělit do následujících částí:

1. python skript řídící funkci krabičky,
2. ffmpeg příkaz zpracovávající video data,
3. nginx server distribuující výsledky dalším částem systému a
4. stunnel zajišťující odchozí šifrované spojení.

Ne všechny čtyři popsané části jsou u video krabiček vždy aktivní; jejich využití závisí na konkrétní funkci, kterou má daná krabička plnit (např. krabička zajišťující archivaci videa nevyužívá nginx server ani stunnel).

V systému jsou implementovány následující typy video krabiček:

- *Replicator* — zajišťuje standardizaci vstupního video streamu. Vstupní video stream je převeden do formátu RTMPS a jsou do něj doplněny časové značky s reálným časem. Výsledný standardizovaný stream je poté přes nginx a stunnel připraven k distribuci pro libovolný počet navazujících jednotek v systému. Vstupem této krabičky je obecně vnější zdroj (např. kamera). Výstup pak využívají snímací moduly a navazující video krabičky.
- *Recorder* — slouží pro nahrávání standardizovaného video streamu do video souboru. Používá se primárně na ViAn serveru pro archivaci video streamů pro pozdější využití.

- *Replay* — přehrává standardizované video streamy archivované ve video souborech pro použití v operativní aplikaci. Spouští se na ViAn serveru, kde jsou archivovány video soubory, a operativní aplikaci umožňuje procházet přehrávaný záznam libovolnou rychlostí.
- *Restreamer* — přehrává uložený záznam z video souboru takovým způsobem, jako by se jednalo o nový živý video stream. Tato video krabička se používá v případech, kdy je potřeba archivovaný video stream znovu využít jako vstupní stream do systému.

4 ViAn SensingAPI

ViAn SensingAPI je REST API rozhraní, které poskytuje ViAn Server. Toto rozhraní umožňuje snímacím (výpočetním) modulům zasílat data z detekcí a ukládat je na straně ViAn Serveru (a také další provozní činnosti jako správu registrace replikátoru, který video obohacuje o synchronizační značku, a správu zaznamenávání videa, které je modulem zpracováváno).

API endpoint pro ukládání dat je implementován jako obecný mechanismus pro uložení dat za běhu, což znamená, že na straně serveru není třeba dopředu vytvářet místo pro uložení dat, úložiště se vytváří za běhu až v době zavolání endpointu definujícího hlavičku dat. Další výhodou tohoto principu je zjednodušení pro vývojáře v případech, kdy se struktura dat v průběhu vývoje často mění (nebo se mění například datové typy dat); v takovém případě ViAn Server vytvoří nové úložiště pro tato nová data.

ViAn SensingAPI se skládá z poměrně malého množství API endpointů, které můžeme rozdělit do skupin dle toho, co zpracovávají nebo s čím pracují:

- **datové** endpointy *(pro snímací moduly)*:
  - `/data/header` - pro definici, jaké informace se budou ze snímacího modulu posílat a jakého datového typu budou
  - `/data/store` - pro zasílání (resp. ukládání) dat samotných
- endpoint pro **informování o stavu snímacího modulu** *(pro přenos informace o stavu, ve kterém se modul nachází, ze snímacího uzlu do centrální evidence na ViAn Serveru)*:
  - `/sensing-module/status` - pro nahlášení změny stavu modulu (volitelně lze přiložit i detailní informaci - typicky použitelné například pro zaznamenání chybové zprávy)
- endpointry **nahrávající video** *(pro ovládání nahrávání vudei streamu na strane ViAn Serveru)*:
  - `/video/capture/start` - k zahájení nahrávání předaného streamu
  - `/video/capture/stop` - k zastavení probíhajícího nahrávání
- endpointy pro **správu video replikátorů** *(pro registraci a uvolnění využívání replikátoru, resp. páru replikátor-kamera)*:
  - `/video/replicator/register` - registrace replikátoru pro danou kameru
  - `/video/replicator/release` - uvolnění replikátoru po skončení užívání snímacím modulem

Společné parametry požadavku (requestu)
Pro všechny požadavky posílané na ViAn SensingAPI jsou povinné nejméně tyto následující parametry:

- `dataset`:
  - umístění parametru: předáván jako součást tzv. ‘query‘ řetězce v URL (tj. součást URL adresy)
  - povinnost: povinný
  - datový typ: string (řetězec)
- `access-token`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)

4.1 Datové endpointy

Jak již bylo v úvodu zmíněno, endpointy této skupiny jsou navrženy pro účely snímacích (výpočetních) modulů, konkrétně pro zasílání dat (událostí detekovaných snímacími moduly) ze snímacích modulů do úložiště na straně ViAn serveru. U datových endpointů je velmi důležité dodržet nastavené pořadí jejich použití:

1. Každý modul nejprve zavolá `/data/header` endpoint pro definici datového úložiště.
2. Teprve potom může modul používat endpoint `/data/store`, a to i vícekrát po sobě, jak jen je potřebné pro přenos samotných detekovaných dat.

4.1.1 Odpovědi zasílané na požadavky datových endpointů:

V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": {
    "access_token": "***",
    "token_validity": "2021-01-10T22:35:27+01:00"
  }
}
```

Datová část odpovědi obsahuje následující položky:

- `access_token` - přístupový token potřebný pro příští požadavek (přístupový token může být jiný při dalším požadavku)
- `token_validity` - časová značka, do kdy token platí, v opačném případě expiruje

Poznámka: Odpověď byla navržena pro možnost změny přístupového tokenu, zatím se však výměna nepoužívá.

4.1.2 Endpoint pro definování / zaslání hlavičky: `/data/header`

Endpoint pro definování / zaslání hlavičky je navržen pro účely přípravy / vytvoření datového úložiště potřebného pro ukládání dat snímacího modulu. Tento endpoint, jak už bylo řečeno, musí být zavolán na počátku před endpointem, který data ukládá (`/data/store`). Endpoint se volá pouze jednou, další zavolání endpointu se stejnou hlavičkou v rámci jednoho běhu je zbytečné. Opakované zavolání endpointu pro zasílání hlavičky, avšak s jinou hlavičkou, v rámci jedné instance modulu není žádané, neboť může způsobit neuložení všech následně zasílaných dat.

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:
Tělo požadavku obsahuje řetězec ve formátu JSON, který je složen z párů klíč-hodnota, kde klíč udává jméno ukládané vlastnosti a hodnota definuje datový typ samotné vlastnosti.

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/data/header?da
↪  taset=test" -H  "accept: application/json" -H  "access-tok
↪  en: ***" -H  "Content-Type: application/json" -d "{\"fo
↪  o\":\"int\", \"bar\":\"string\"}"
```

4.1.3 Endpoint pro ukládání dat: `/data/store`

Endpoint pro ukládání dat je navržen pro zasílání (resp. ukládání) dat událostí extrahovaných příslušným detektorem snímacího modulu. Tento endpoint může být používán opakovaně v rámci instance snímacího modulu, ale vždy povinně až po zavolání endpointu pro definování hlavičky (`/data/header`). Endpoint nabízí použití jednoho ze dvou režimů zpracování dat v rámci každého požadavku:

- režim zasílání každé položky zvlášť
- režim zasílání více položek dohromady (dávkový režim).
  Rozlišení, který mód je použit, je založeno pouze na formátu těla požadavku, tj. zaslané JSON zprávy s daty; není tedy nutné v požadavku dále specifikovat, o jaký režim se jedná.

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:
Telo požadavku obsahuje řetězec ve formátu JSON, který je složen z párů klíč-hodnota, kde klíč udává jméno ukládané vlastnosti a hodnota udává samotnou hodnotu této vlastnosti detekované události.

Příklad požadavku v cURL (režim každá položka zvlášť):
V tomto režimu zasílání každé položky v novém požadavku je datový záznam reprezentován jako standardní objekt JSON (kolekce).

```
curl -X POST "https://localhost/vian_sensingapi/data/store?dat
↪  aset=test" -H  "accept: application/json" -H  "access-toke
↪  n: ***" -H  "Content-Type: application/json" -d "{\"foo\":
↪  1234, \"bar\":\"ViAn module test\"}"
```

Příklad požadavku v cURL (režim více položek dohromady / dávkový):
V tomto dávkovém režimu (zasílání více položek dohromady v rámci jednoho požadavku) jsou data reprezentována jako pole jednotlivých záznamů ve formátu JSON, kde každá jednotlivá položka pole je reprezentována jako standardní JSON objekt / kolekce (podobně jako v režimu každé položky zvlášť).

```
curl -X POST "https://localhost/vian_sensingapi/data/store?dat
↪  aset=test" -H  "accept: application/json" -H  "access-toke
↪  n: ***" -H  "Content-Type: application/json" -d "[{\"fo
↪  o\":1234, \"bar\":\"ViAn module test\"}, {\"foo\":5678,
↪  \"bar\":\"ViAn module test 2\"}]"
```

4.2 Endpoint pro informování o stavu snímacího modulu

4.2.1 Endpoint pro informování o stavu snímacího modulu: `/sensing-module/status`

Endpoint pro informování o stavu snímacího uzlu přináší snímacímu uzlu prostředek pro informování o stavu snímacího (výpočetního) modulu, který byl spuštěn na daném uzlu spuštěn, na centrální úrovni skrze ViAn Server.

Dostupné stavy

- `configured` - Detekce je nakonfigurovaná, ale ještě nebyla spuštěna (výchozí stav po vytvoření detekce).
- `launching` - Detekce byla spuštěna, zatím se spouští potřebné součásti k běhu modulu či zaznamenávání videa apod.
- `prestopped` - Byl zaznamenán požadavek na ukončení deteke (akce stop) v průběhu spouštění potřebných součástí, ještě před samotným spuštěním snímacího modulu.
- `prefailed` - Objevila se chyba v průběhu spouštění součástí, ještě před samotným spuštěním snímacího modulu.
- `running` - Snímací (výpočetní) modul byl spuštěn (= všechny potřebné součásti byly úspěšně spuštěny).
- `failed` - Snímací modul skončil chybou.
- `finished` - Snímací modul skončil úspěšně sám od sebe (například došel na konec videa).
- `stopped` - Snímací modul (a potřebné součásti) byly ukončeny požadavkem zastavení (akce stop).
- `killed` - Snímací modul (a potřebné součásti) byly ukončeny požadavkem okamžitého ukončení (akce kill).

Povolené přechody stavů
Průběh stavů modulu, kterých může nabývat, je omezen následujícími přechody mezi stavy - z::

- `configured` do:
  - `launching`
  - `prefailed`
  - `prestopped`
  - `killed`
- `prefailed` do:
  - `launching` *(pouze pro vývojové účely)*
  - `prestopped`

  - `killed`

- `launching` do:
  - `running`
  - `prefailed`
  - `prestopped`
  - `killed`

- `running` do:
  - `failed`
  - `finished`
  - `stopped`
  - `killed`

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:

- `status`:
  - povinnost: povinný
  - datový typ: string (řetězec, resp. výčet dostupných stavů zmíněných výše)
  - popis: Stav snímacího modulu.

- `detail`:
  - povinnost: volitelný
  - datový typ: string (řetězec)
  - popis: Detailnější informace ke stavu - například pro chybový stav `failed` typicky poslední chybová zpráva.

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/sensing-module
↪  /status?dataset=test" -H  "accept: application/json" -H  "
↪  access-token: ***" -H  "Content-Type: application/json" -d
↪  "{\"status\":\"failed\", \"detail\":\"Missing library xyz
↪  .\"}"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

4.3 Endpointy řídící záznam streamu

ViAn SensingAPI obsahuje rovněž endpointy řídící záznam streamu, které umožňují zaznamenávat stream na straně ViAn Serveru nebo naopak takto zaznamenávaný stream ukončit.

4.3.1 Endpoint pro zahájení záznamu: `/video/capture/start`

Thisendpoint zařídí, aby byly zaznamenány potřebné údaje o záznamu do úložiště a zároveň na pozadí spustí pomocný skript, který nahrává samotný stream do souboru na straně Vian Serveru. Tento pomocný skript načte důležité informace ze vstupního streamu (například synchronizační časovou značku) a takto získané informace přidá do databáze úložiště. Interně rovněž využívá `VideoBoxy` z kapitoly 3.2.

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:

- `replicator`:
  - povinnost: povinný
  - datový typ: string (řetězec, ve formátu url)
  - popis: URL replikátoru vstupního streamu, který bude zaznamenáván.

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/video/capture/
↪  start?dataset=test" -H  "accept: application/json" -H  "ac
↪  cess-token: ***" -H  "Content-Type: application/json" -d "
↪  {\"replicator\":\"rtmps://localhost/replicated/camera/liv
↪  e\"}"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": {
    "capture_id": "2e42a800-e81f-475c-a9ea-6b6c86536cab"
  }
}
```

Položka `capture_id` v datové části odpovědi je identifikátor spuštěného nahrávání; tento identifikátor je důležitý pro vypnutí nahrávání.

4.3.2 Endpoint pro ukončení záznamu: `/video/capture/stop`

Tento endpoint ukončí běh nahrávacího skriptu, který byl spuštěn za pomoci endpointu `video/capture/start`, což způsobí zároveň aktualizaci koncové značky záznamu apod.

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:

- `capture`:
  - povinnost: povinný
  - datový typ: string (řetězec)
  - description: Identifikátor nahrávání, které má být ukončeno.

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/video/capture/
↪  stop?dataset=test" -H  "accept: application/json" -H  "acc
↪  ess-token: ***" -H  "Content-Type: application/json" -d "
↪  {\"capture\":\"2e42a800-e81f-475c-a9ea-6b6c86536cab\"}"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

4.4 Endpointy pro správu replikátorů

Tyto endpointy jsou navrženy pro správu využívání služeb replikátoru.

4.4.1 Endpoint registrující replikátor: `/video/replicator/register`

Tímto endpointem snímací uzel nabízí pro danou kameru svůj vlastní replikátor, který přidá časovou značku do video streamu. Pokud kameru neovládá žádný replikátor, je nabízený replikátor zaregistrován a jako stvrzení jeho registrace je jeho URL adresa zaslána zpět v odpovědi. V opačném případě se pošle v odpovědi URL adresa tohoto již dříve zaregistrovaného replikátoru - takový replikátor je nutné používat při dalším zpracování, neboť endpoint zároveň registruje použití replikátoru snímacím modulem.

Další parametry požadavku:
Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:

- `camera`:
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: Zdroj kamery (streamu), který má být zpracováván (označkován) daným replikátorem.
- `replicator`:

- povinnost: povinný
- datový typ: string (řetězec ve formátu url)
- popis: Nabízená URL replikátoru, který by mohl zpracovávat (označkovávat) daný video vstup (stream) v případě, že tento vstup již není zpracováván (značkován) jiným replikátorem.

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/video/replicat
↪  or/register?dataset=test" -H  "accept: application/json"
↪  -H  "access-token: ***" -H  "Content-Type: application/jso
↪  n" -d "{\"camera\":\"rtsp://camera.demo\",\"replicato
↪  r\":\"rtmps://replicator2.demo\"}"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": {
    "replicator": "rtmps://replicator1.demo"
  }
}
```

Pokud vstupní zdroj videa není zpracováván (značkován) jiným replikátorem, nabízený replikátor je registrován pro toto zpracování (značkování), což je stvrzeno předáním jeho zasílané URL zpátky v odpovědi. V případě, že je zdroj videa zpracováván již jiným replikátorem, je adresa tohoto již dříve registrovaného replikátoru navrácena v odpovědi pro další použití (např. ve snímacím modulu). Parametr `replicator_owner_token` v datové části odpovědi je vrácen pouze v případě, že je nabízený replikátor akceptován; v opačném případě je v datové části obsažena pouze položka `replicator`.

4.4.2 Endpoint pro uvolnění registrace replikátoru: `/video/replicator/release`

Tento endpoint zruší registraci o využívání replikátoru snímacím modulem (identifikovaného jeho přístupovým tokenem).

Další parametry požadavku:

Nejsou požadovány žádné další parametry kromě společných zmíněných dříve.

Položky těla požadavku:

- `replicator`:
  - povinnost: povinný
  - datový typ: string (¨řetězec ve formátu url)

Příklad požadavku v cURL:

```
curl -X POST "https://localhost/vian_sensingapi/video/replicat
↪  or/release?dataset=test" -H  "accept: */*" -H  "access-tok
↪  en: ***" -H  "Content-Type: application/json" -d "{\"repli
↪  cator\":\"rtmps://replicator.demo\"}"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód 200 s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

5 ViAnAPI

VianAPI je rozhraní REST API poskytované Vian Serverem, které koncovým uživatelům a různým aplikacím umožňuje dotazování a další databázové i jiné operace s daty uloženými v databázi na ViAn Serveru. Umožňuje získávat různá metadata, např. o jednotlivých snímacích modulech, kamerách nebo datasetech. Poskytuje také funkcionalitu ke spuštění nebo zastavení detekce. Dále umožňuje umožňuje čtení a dotazování dat, která bylo uložena snímacími moduly jako výsledek detekce. Mezi pokročilejší funkce patří spouštění složených detekcí a podobnostního vyhledávání. Kromě toho také VianAPI umožňuje znovu spouštět video sekvence uložené na Vian Serveru.

Pro všechny požadavky na VianAPI je potřeba zadat:

- `access-token` - přístupový token v hlavičce požadavku. Zatím se tyto tokeny nevyužívají, jsou navrženy pro případné pozdější použití. Každý endpoint vrací HTTP kód `200`, jestliže je požadavek úspěšný.

5.1 Přehled endpointů ve VianAPI

ViAn VianAPI se skládá z následujících endpointů:

- **Dotazování na metadata**:
  - `/dataset/*` - dotazování na informace o datasetech v databázi
  - `/camera/*` - dotazování na informace o kamerách využívaných pro detekce
  - `/sensing-module/`, `/sensing-module/all`, `/sensing-module/id` - dotazování na informace o snímacích modulech využívaných pro detekce
  - `/sensing-module/all-sensenodes` - informace o všech snímacích uzlech
  - `/replicator` and `/replicator/all` - informace o využívaných replikátorech
- **Dotazování nad výstupními daty detekcí a jejich metadaty**
  - `/detection/running`, `/detection/past` - zjištění informací o detekcích, které právě běží nebo které byly v minulosti dokončeny
  - `/detection/data` - jednoduché dotazování - vrátí výstupní data dané detekce, které odpovídají zadané (pouze jedné) podmínce
  - `/detection/records` - vrátí informace o video sekvencích, které byly uloženy v rámci zadané detekce
  - `/detection/query` - pokročilejší dotazování nad výstupními daty z detekce
  - `/detection/clean` - odstraní z databáze veškerá data i metadata o zadané detekci a zároveň odstraní všechny video sekvence pro tuto detekci

  - `/detection/token` - zjistí přístupový token pro zadanou detekci

- **Spouštění a ukončování detekcí**
  - `/sensing-module/init` - spustí detekci
  - `/sensing-module/stop` - ukončí detekci

- **Metadata výstupů jednotlivých endpointů**
  - `metadata/*` - vrátí informace o všech atributech vrácených jednotlivými endpointy

- **Správa datasetů a kamer**
  - `/dataset/add`, `dataset/drop` - přidá/odstraní dataset
  - `/camera/add`, `camera/drop`, `camera/update` - přidá, odstraní nebo modifikuje informace o kameře

- **Složené detekce**
  - `/compound/get-all`, `compound/details` - endpointy pro zjišťování informací o dostupných složených detekcích
  - `/compound/query` - spustí složenou detekci s požadovanými parametry

- **Podobnostní vyhledávání**
  - `/similar/status`, `similar/results` - zjištění stavu zadané úlohy podobnostního vyhledávání / zobrazení jejích výsledků
  - `/similar/run` - spuštění úlohy podobnostního vyhledávání

- **Přehrání videa**
  - `/replay/start` - spustí přehrávání videa a vrátí URL
  - `/replay/stop` - ukončí přehrávání videa

5.2 Endpointy pro dotazování metadat

Tyto endpointy zahrnují dotazování metadat o datasetech, kamerách a snímacích modulech.

5.2.1 Dotazování metadat o datasetech: `/dataset/*`

Jsou zde dostupné tři možnosti, jak se dotazovat nad metadaty o datasetech:

Zjištění informace o datasetu na základě identifikátoru: `/dataset/`
Pro zadanou hodnotu identifikátoru (name) datasetu, který je jediným parametrem tohoto endpointu, jsou vráceny hodnoty všech atributů zadaného datasetu.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/dataset?dataset=tes
↪  t" -H  "accept: */*" -H  "user-token: 123"
```

Odpověď na úspěšný požadavek: `200`

```
{
  "status": "success",
  "data": {
    "friendly_name": "test",
    "author": "vb",
    "description": "testovaci",
    "time_from": "2019-05-05T00:00:00+02:00",
    "time_to": "2019-05-06T00:00:00+02:00"
  }
}
```

Zjištění informací o všech datasetech: `/dataset/all`
Tento endpoint nemá žádné vstupní parametry a jeho výsledkem jsou informace o všech dostupných datasetech v databázi. Výstupem je seznam všech datasetů, u každého datasetu jsou uvedeny hodnoty všech atributů.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/dataset/all" -H  "a
↪  ccept: */*" -H  "user-token: 123"
```

Pokročilé dotazování metadat o datasetech: `/dataset/id`
Tento endpoint vrací seznam identifikátorů datasetů. Dataset, jehož identifikátor je ve výsledku, musí splňovat podmínku z těla požadavku. Je zde možné specifikovat libovolné z následujících atributů datasetu a jejich příslušné hodnoty:

- `sensing-module` - snímací modul, jehož data jsou uloženy v datasetu
- `sensing-node` - snímací uzel, kde je umístěn daný snímací modul
- `sensetype` - požadovaný typ snímacího modulu
- `camera` - kamera, z níž jsou uložena data v datasetu
- `time` - časový okamžik, pro který jsou uložena data v datasetu

Příklad cURL požadavku pro podmínku: sensing-module = 'modul'

```
curl -X POST "http://localhost/vian_vianapi/dataset/id" -H  "a
↪  ccept: */*" -H  "user-token: 123" -H  "Content-Type: appli
↪  cation/json" -d "{\"sensing-module\":\"modul\"}"
```

Odpověď na úspěšný požadavek: 200

```
{
  "status": "success",
  "data": [
    "test"
  ]
}
```

5.2.2 Dotazování metadat o kamerách: /camera/*

Podobně jako u datasetů, jsou zde dostupné tři možnosti, jak se dotazovat nad metadaty o kamerách:

Zjištění informace o kameře na základě identifikátoru: /camera/
Pro zadaný identifikátor (UUID) kamery (jediný pamametr tohoto endpointu), hodnoty všech atributů této kamery jsou vráceny.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/camera?camera=cd0ab
↪  c34-9bdd-42fe-8fef-0d5f8d814d7" -H  "accept: */*" -H  "use
↪  r-token: 123"
```

Odpověď na úspěšný požadavek: 200

```
{
  "status": "success",
  "data": {
    "friendly_name": "kamera",
    "url": "1.11.1.1:2222"
  }
}
```

Zjištění informací o všech kamerách: /camera/all
Tento endpoint nemá žádné vstupní parametry a vrací informace o všech kamerách v databázi. Výstupem ke seznam položek se stejnými atributy jako u předchozího endpointu.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/camera/all" -H  "ac
↪  cept: */*" -H  "user-token: 13"
```

Pokročilé dotazování metadat o kamerách: `/camera/id`
Tento endpoint vrací seznam identifikátorů kamer. Kamera, jejíž identifikátor je součástí výsledku, musí splňovat podmínku zadanou v těle požadavku. Zde je možná specifikovat libovolné z následujících atributů, kterou mohou být součástí podmínky, a jejich hodnoty:

- `sensing-module` - snímací modul, který je připojen na danou kameru
- `sensing-node` - snímací uzel, kde je snímací modul připojený na kameru umístěn
- `sense-type` - požadovaný typ snímacího modulu
- `dataset` - identifikátor datasetu, kde jsou data z kamery uložena
- `time` - časový okamžik, ve kterém daná kamera zaznamenávala data

Příklad cURL požadavku odpovídající podmínce: dataset = 'test'

```
curl -X POST "http://localhost/vian_vianapi/camera/id" -H  "ac
↪  cept: */*" -H  "user-token: 123" -H  "Content-Type: applic
↪  ation/json" -d "{\"dataset\":\"test\"}"
```

Odpověď na úspěšný požadavek: `200`

```
{
  "status": "success",
  "data": [
    "cd0abc34-9bdd-42fe-8fef-0d5f8d814d77",
    "78c9d135-2803-4322-9f3e-f4a6330fd741"
  ]
}
```

5.2.3 Dotazování metadat o snímacích modulech: `/sensing-module/*`

Podobně jako u předchozích endpointů, jsou zde k dispozici tři možnosti dotazování nad metadaty o snímacích modulech:

Zjištění informací o snímacím modulu na základě identifikátoru: `/sensing-module/`
Pro identifikátor snímacího modulu (Docker image), zadaného jako jediný

parametr požadavku, jsou vráceny jako výsledek hodnoty všech atributů snímacího modulu a seznam snímacích uzlů, kde je tento snímací modul umístěn.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/sensing-module?sens
↪  emodule=modul" -H  "accept: */*" -H  "user-token: 123"
```

Odpověď na úspěšný požadavek: `200`

```
{
  "status": "success",
  "data": {
    "dockerimg": "modul",
    "sensetype": "face",
    "friendly_name": "face detection",
    "description": "face detection",
    "sensenodes": [
      "https://localhost/sensingnode2_api",
      "https://localhost/sensingnode_api"
    ]
  }
}
```

Zjištění informací o všech snímacích modulech: `/sensing-module/all`
Tento endpoint nemá žádné vstupní parametry a vrací informaci o všech snímacích modulech v databázi včetně snímacích uzlů, na kterých jsou jednotlivé snímací moduly umístěny. Výsledek je ve formě seznamu položek se stejnými atributy jako u předchozího endpointu.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/sensing-module/all"
↪  -H  "accept: */*" -H  "user-token: 123"
```

Pokročilé dotazování nad metadaty o snímacích modulech: `/sensing-module/id`
Tento endpoint vrací jako výsledek seznam identifikátorů snímacích modulů odpovídajících podmínce. Podmínka je zadána v těle požadavku, může obsahovat libovolné z následujících atributů a jejich požadované hodnoty:

- `camera` - kamera, jejíž data jsou zpracovávána snímacím modulem
- `sensing-node` - snímací uzel, kde je snímací modul umístěn
- `sense-type` - požadovaný typ snímacího modulu
- `dataset` - identifikátor datasetu, kam jsou data ze snímacího modulu ukládána
- `time` - čas, kdy byl snímací modul aktivní

Příklad cURL požadavku pro podmínku: dataset = 'test'

```
curl -X POST "http://localhost/vian_vianapi/sensing-module/id"
↪  -H  "accept: */*" -H  "user-token: 123" -H  "Content-Type:
↪  application/json" -d "{\"dataset\":\"test\"}"
```

Výsledek úspěšného požadavku: 200

```
{
  "status": "success",
  "data": [
    "demo",
    "modul"
  ]
}
```

Zjištění všech snímacích uzlů: `/sensing-module/all-sensenodes`
Tento endpoint nemá žádné vstupní parametry a pouze vrací informaci o všech snímacích uzlech v databázi. Výstupem je seznam položek obsahujících následující atributy:

- `uri`
- `friendly-name`
- `desciption`

Příklad cURL požadavku:

```
curl -X GET "http://localhost/vian_vianapi/sensing-module/all-
↪  sensenodes" -H  "accept: */*" -H  "user-token: 123"
```

5.2.4 Informace o replikátorech: `/replicator/*`

Tyto endpoint slouží k získávání potřebných informací o replikátorech.

Informace o právě využívaných replikátorech: `/replicator/all`
Endpoint poskytuje informace o všech replikátorech, které jsou právě využívány. Není vyžadován žádný vstupní parametr. Výsledná informace se skládá z URL kamery, snímacího modulu a seznamu příslušných detekcí.

Příklad cURL požadavku:

```
curl -X GET "https://vian-dev.fit.vutbr.cz/vian_vianapi/replic
↪  ator/all" -H  "accept: */*" -H  "user-token: 1323"
```

Získání informací o replikátorech pro danou kameru: `/replicator`
Pro kameru, jejíž identifikátor je jediným parametrem požadavku, tento endpoint poskytne informaci o jejích replikátorech, tj. URL replikátoru a seznam příslušných detekcí.

Příklad cURL požadavku pro podmínku: camera = ad2915ac...

```
https://localhost/vian_vianapi/replicator?camera=ad2915ac...
```

5.3 Endpointy pro spuštění a zastavení detekce

5.3.1 Spuštění nové detekce: `/sensing-module/init`

Tento endpoint spustí uloženou databázovou funkci, která uloží potřebnou informaci o nově spuštěné detekci do databáze a jestliže je tento proces úspěšný, vrátí přístupový token a čas jeho platnosti. Ke spuštění nové detekce je potřeba zadat následující parametry požadavku:

- `dataset` - identifikátor datasetu, kam se budou data ze spouštěné detekce ukládat
- `sensing-node` - snímací uzel, kde je umístěn snímací modul provádějící detekci
- `sensing-module` - snímací modul, který bude provádět detekci
- `camera` - indentifikátor kamery, nad jejíž daty bude prováděna detekce
- `friendly-name` - uživatelské pojmenování detekce
- `fetch-stream` - booleovská hodnota udávající, zda bude video z této detekce ukládáno na server

Příklad cURL požadavku:

```
curl -X POST "http://localhost/vian_vianapi/sensing-module/ini
↪  t" -H  "accept: */*" -H  "user-token: 123" -H  "Content-Ty
↪  pe: application/json" -d "{\"dataset\":\"test\",\"sensing-
↪  node\":\"https://localhost/sensingnode_api\",\"sensing-mod
↪  ule\":\"modul\",\"camera\":\"1.11.1.1:2222\",\"friendly-na
↪  me\":\"new detection\",\"fetch-stream\":true}"
```

Odpověď na úspěšný požadavek: `200`

```
{
  "status": "success",
  "data": {
    "access_token": "eb1a38382a15bf58bf4c82b86fa80a50",
    "token_validity": "2021-01-13 00:00:00+01"
  }
}
```

5.3.2 Zastavení běžící detekce: `/sensing-module/stop`

Pro zastavení běžící detekce je vyžadováno zadání snímacího uzlu, snímacího modulu a kamery. Význam těchto parametrů je stejný jako u předchozího endpointu. Jestliže jsou všechny hodnoty jednotlivých parametrů platné, je spuštěna uložená databázová funkce, která zaznamená čas ukončení detekce. Endpoint nemá žádné návratové hodnoty.

Příklad cURL požadavku:

```
curl -X POST "http://localhost/vian_vianapi/sensing-module/sto
↪  p" -H  "accept: */*" -H  "user-token: 123" -H  "Content-Ty
↪  pe: application/json" -d "{\"sensing-node\":\"https://loca
↪  lhost/sensingnode_api\",\"sensing-module\":\"modul\",\"cam
↪  era\":\"1.11.1.1:2222\"}"
```

5.4 Dotazování výstupních dat a metadat z detekcí

Tato skupina endpointů poskytuje možnost získat informace o běžících detekcích a detekcích, které byly ukončeny v minulosti. Dále umožňuje číst a dotazovat se ve výsledcích detekcí v příslušných výstupních tabulkách detekcí.

5.4.1 Získání metadat o běžících detekcích: `/detection/running`

Tento endpoint poskytuje hodnoty všech atributů právě běžících detekcí (tj. takových detekcí, u kterých je prázdná hodnota u atributu `time_to`). Detekce jsou zobrazeny pro dataset, jehož identifikátor je uveden jako parametr požadavku. Odpověď obsahuje množinu položek, z nichž každá představuje jednu detekci. Pro každou tuto detekci jsou vypsány hodnoty následujících atributů:

- `config_id`: identifikátor detekce (uuid)
- `friendly_name`: uživatelský název detekce
- `sensenode`: identifikátor snímacího uzlu, kde tato detekce běží

- `sensemodule`: snímací modul, který provádí detekci
- `camera`: identifikátor kamery (uuid), na níž je prováděna detekce
- `fetch_stream`: booleovská hodnota udávající, zda bude video z této detekce ukládáno na server
- `time_from`: počáteční čas detekce
- `data_storage`: cílová tabulka, kam se ukládají výsledky detekce
- `access_token`
- `token_valid_to`
- `last_activity`

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/detection/running?d
↪  ataset=test" -H  "accept: */*" -H  "user-token: 123"
```

5.4.2 Získání metadat o ukončených detekcích: `/detection/past`

Tento endpoint poskytuje hodnoty všech atributů detekcí, které byly v minulosti ukončeny. Je potřeba zadat jako parametr identifikátor datasetu, jehož detekce se mají vyhledat. Odpověď obsahuje tytéž atributy jako předchozí endpoint, navíc je ještě výsledkem atribut `time_to`, který udává čas, kdy byla daná detekce ukončena.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/detection/past?data
↪  set=test" -H  "accept: */*" -H  "user-token: 123"
```

5.4.3 Jednoduché dotazování dat detekcí: `/detection/data`

Pro zadaný dataset, detekci a podmínku tento endpoint vyhledá záznamy z tabulky výsledků detekce, které odpovídají zadané podmínce. Podmínka `attname = value` je vyjádřena pomocí dvou parametrů požadavku: `attname` and `value`. Je však možné specifikovat i jiné typy podmínek. Atributy výsledku závisí na schématu příslušné tabulky pro danou detekci. Zde je seznam parametrů tohoto endpointu:

- `dataset`: identifikátor datasetu, kde jsou data dané detekce uložena
- `config_id`: identifikátor detekce
- `offset`: číslo řádku výsledku, od kterého se mají začít vracet výsledky (užitečné především v případech, kdy počet záznamů na výstupu je velmi vysoký - implicitní hodnota je 0) 0)
- `row-count`: požadovaný počet řádků výsledku - implicitní hodnota je 10

- `attname`: název atributu, který je součástí podmínky
- `value`: požadovaná hodnota atributu v podmínce (pro podmínku ve tvaru `attname = value`, jinak musí být hodnota prázdná)
- `lower`: požadovaná hodnota atributu pro podmínku typu `attname > value`
- `upper`: požadovaná hodnota atributu pro podmínku typu `attname < value` (jestliže jsou specifikovány obě hodnoty (lower i upper), pak bude vyhodnocena podmínka `attname BETWEEN lower AND upper`)
- `order-by`: atribut, podle kterého se mají data seřadit
- `desc`: booleovský parametr, který udává, zda se má výsledek seřadit sestupně - implicitní hodnota je false, tj. vzestupné řazení)

Příklad cURL požadavku pro podmínku: veh_color_class = 'white'

```
curl -X GET "http://localhost/vian_vianapi/detection/data?data
↪  set=test&config-id=0fc639c8-9f63-4112-bc0a-bbb767a6e93d&of
↪  fset=5&row-count=1&attname=veh_color_class&value=white&ord
↪  er-by=frame_num&desc=false" -H  "accept: */*" -H  "user-to
↪  ken: 123"
```

Odpověď na úspěšný požadavek: 200


```
{
  "status": "success",
  "returned_from": 5,
  "returned_to": 5,
  "total_records": 11139,
  "data": [
    {
      "row_id": 7,
      "config_id": "0fc639c8-9f63-4112-bc0a-bbb767a6e93d",
      "time_from": "2019-10-07T14:00:20+02:00",
      "time_to": null,
      "frame_ts": 50420.68,
...
      "veh_color_class": "white",
      "veh_color_class_prob": 0.9505827,
      "veh_model": "fiat ducato van mk3",
      "veh_model_prob": 0.7821336,
      "frame_num": 1034,
      "id": 2346
    }
```

```
  ]
}
```

5.4.4 Zjištění informace o pořízených video sekvencích v rámci detekce: `/detection/records`

Tento endpoint poskytuje informace o všech video sekvencích pro detekci zadanou jako parametr požadavku. Identifikátor detekce (`config-id`) je zde jediným parametrem. Název souboru obsahujícího video sekvenci a informaci o časových značkách je vráce pro každou video sekvenci.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/detection/records?c
↪  onfig-id=65eb8d02-2d18-4eb0-88bb-00130c0e2cbb" -H  "accept
↪  : */*" -H  "user-token: 123"
```

Výsledek úspěšného požadavku: `200`

```
{
  "status": "success",
  "data": [
    {
      "filename": "20051210-w50t.flv",
      "ftime_offset": 1570399200,
      "ftime_from": 50400,
      "ftime_to": 52302.496
    }
  ]
}
```

5.4.5 Pokročilé dotazování dat z detekcí: `/detection/query`

Pokročilé dotazování umožňuje specifikovat komplexnější podmínky pro dotazy nad výstupními daty z detekcí. Význam parametrů požadavku (`dataset`, `config-id`, `offset`, `row-count`, `order-by`, `desc`) je stejné jako u endpointu `/detection/data`. Monožina podmínek je specifikována ve formě JSON v těle požadavku. Zde je příklad zápisu složené podmínky: `(veh_model LIKE '%ford%' OR veh_model LIKE '%renault%') AND frame_num > 128`

```
[
  {
    "attname": "veh_model",
```

```
    "operator": "like",
    "values": [
      "renault", "ford"
    ]
  },
  {
    "attname": "frame_num",
    "operator": ">",
    "values": [
      "128"
    ]
  }
]
```

Příklad odpovídajícího cURL požadavku:

```
curl -X POST "http://localhost/vian_vianapi/detection/query?da
↪  taset=test&config-id=65eb8d02-2d18-4eb0-88bb-00130c0e2cbb&
↪  offset=0&row-count=10&desc=false" -H  "accept: */*" -H  "u
↪  ser-token: 123" -H  "Content-Type: application/json" -d "[
↪  {\"attname\":\"veh_model\",\"operator\":\"like\",\"value
↪  s\":[\"renault\",\"ford\"]},{\"attname\":\"frame_num\",\"o
↪  perator\":\">\",\"values\":[\"128\"]}]"
```

Struktura výsledné odpovědi je obdobná tomu, co je výsledkem endpointu `/detection/data`.

5.4.6 Vyčištění dat o detekci: `/detection/clean`

Pro zadanou detekci (specifikována pomocí `config-id`) jsou smazány veškeré informace o této detekci ze všech relevantních tabulek v databázi a odstraní tabulku, kde jsou uloženy výsledky této detekce. Jestliže součástí této detekce jsou i uložené video sekvence, jsou také smazány z ViAn serveru.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/detection/clean?con
↪  fig-id=65eb8d02-2d18-4eb0-88bb-00130c0e22aa" -H  "accept:
↪  */*" -H  "user-token: 123"
```

5.4.7 Získání tokenu pro přístup k detekci: `/detection/token`

Tento endpoint poskytne uživateli token pro přístup k detekci, která je zadána pomocí jejího `config-id` (jediný parametr požadavku).

Příkad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/detection/token?con
↪  fig-id=0fc639c8-9f63-4112-bc0a-bbb767a6e93d" -H  "accept:
↪  */*" -H  "user-token: 123"
```

5.5 Přístup k metadatům odpovědí jednotlivých endpointů

Tyto endpointy poskytují informaci o všech atributech a jejich datových typech, které jsou výsledkem ostatních endpointů, které jsou součástí VianAPI. Jde např. o endpointy, které zpřístupňují informace o snímacích modulech, kamerách nebo datasetech atd. Tyto endpointy nemají žádné vstupní parametry. Zde je seznam všech endpointů poskytujících tato metadata:

- `/metadata/dataset-atts` - poskytuje seznam atributů datasetů, které jsou výsledkem endpointů `/dataset/*`
- `/metadata/sensemodule-atts` - poskytuje seznam atributů snímacích modulů které jsou výsledkem endpointů `/sensing-module/*`
- `/metadata/sensenode-atts` - poskytuje seznam atributů snímacích uzlů (výsledek endpointu `/sensing-module/all-sensenodes`)
- `/metadata/camera-atts` - poskytuje seznam atributů kamer, které jsou výsledkem endpointů `/camera/*`
- `/metadata/detection-atts` - poskytuje seznam atributů detekcí, které jsou výsledkem endpointů `/detection/current` a `detection/past`
- `/metadata/data-atts` - poskytuje seznam atributů výstupní tabulky, kam se ukládají informace ze zadané detekce, která je specifikována pomocí parametrů `dataset` a `config-id`.

Navíc je součástí této skupiny i endpoint `/metadata/table-atts`, který umožňuje zjistit množinu atributů a jejich datových typů pro zadanou konkrétní tabulku. Tu je potřeba specifikovat pomocí následujících dvou parametrů:

- `dataset`: název datasetu, jehož je tabulka součástí
- `tablename`: název samotné tabulky

Příklad cURL požadavku pro endpoint `metadata/camera-atts`

```
curl -X GET "http://localhost/vian_vianapi/metadata/camera-att
↪  s" -H  "accept: */*" -H  "user-token: 123"
```

Výsledek tohoto požadavku:
Odpověď na úspěšný požadavek: `200`

```
{
  "status": "success",
  "data": [
    {
      "attname": "friendly_name",
      "typname": "varchar"
    },
    {
      "attname": "url",
      "typname": "varchar"
    },
    {
      "attname": "id",
      "typname": "uuid"
    }
  ]
}
```


5.6 Správa datasetů

Správa datasetů zahrnuje endpointy pro přidání nového datasetu a pro odstranění existujícího datasetu.

5.6.1 Přidání nového datasetu: `/dataset/add`

Tento endpoint vytvoří nový dataset, což zahrnuje vložení informace o novém datasetu do databáze, vytvoření nového odpovídajícího databázového schématu, kam budou ukládány tabulky z detekcí patřící do tohoto datasetu a v tomto novém schématu vytvoří potřebné tabulky. Zde je seznam parametrů tohoto endpointu:

- `code-name`: unikátní název datasetu, který bude jeho identifikátorem
- `friendly-name`: uživatelské pojmenování datasetu
- `author`: autor datasetu, musí být v tabulce `authors`
- `descr`: stručný popis datasetu

Příklad cURL požadavku

```
curl -X POST "https://localhost/vian_vianapi/dataset/add?code-
↪  name=test&friendly-name=testovaci&author=user1&descr=testo
↪  vaci" -H  "accept: */*" -H  "user-token: 123"
```

5.6.2 Odstranění datasetu: `/dataset/drop`

Tento endpoint zajistí odstranění datasetu, který je potřeba specifikovat pomocí jeho unikátního názvu (code name), což je jediný parametr tohoto endpointu. Je odstraněno příslušné databázové schéma a veškeré informace o tomto datasetu jsou odstraněny z metadat.

Příklad cURL požadavku

```
curl -X POST "https://localhost/vian_vianapi/dataset/drop?code
↪  -name=test2" -H  "accept: */*" -H  "user-token: 123"
```

5.7 Správa kamer

Do správy kamer jsou zahrnuty endpointy pro přidání, odstranění a modifikaci kamery.

5.7.1 Přidání nové kamery: `/camera/add`

Tento endpoint vloží do databáze informace o nové kameře. Je potřeba zadat jako parametry požadavku její URL a její uživatelské pojmenování (friendly name).

Příklad cURL požadavku

```
curl -X POST "https://localhost/vian_vianapi/camera/add?url=1.
↪  1.1.1%2F1234&friendly-name=new%20camera" -H  "accept: */*"
↪  -H  "user-token: 123"
```

5.7.2 Přidání nové kamery a získání jejího identifikátoru (jestliže existuje, získání pouze existujícího identifikátoru): `/camera/add-get-id`

Tento endpoint přidá novou kameru do databáze, pokud již neexistuje, a vrátí její nově vytvořený identifikátor, pokud existuje, na základě URL získá pouze její identifikátor a vrátí ho. Je potřeba specifikovat URL kamery a nepovinně uživatelské pojmenování kamery (friendly name).

Příklad cURL požadavku

```
curl -X POST "https://localhost/vian_vianapi/camera/add-get-id
↪  ?url=1.1.1.1%2F1234&friendly-name=new%20camera" -H  "accep
↪  t: */*" -H  "user-token: 123"
```

Odpověď na úspěšný požadavek
Úspěšný požadavek vrátí HTTP kód `200` a následující tělo odpovědi:

```
{
  "status": "success",
  "data": {
    "id": "12345678-9999-abcd-0123-456789abcdef",
  }
}
```

5.7.3 Odstranění kamery: `/camera/delete`

Tento endpoint odstraní informaci o kameře z databáze. Je potřeba zadat identifikátor kamery jako parametr požadavku.

Příklad cURL požadavku

```
curl -X POST "https://localhost/vian_vianapi/camera/delete?id=
↪  0c32bcc6-76c9-4f61-ab07-7833b9bf3e6a" -H  "accept: */*" -H
↪  "user-token: 123"
```

5.7.4 Modifikace kamery: `/camera/update`

Tento endpoint modifikuje informace o kameře v databázi. Je potřeba specifikovat identifikátor kamery, kterou potřebujeme upravit. Dalšími dvěma parametry jsou nové hodnoty zbývajících atributů kamery (`url`, `friendly-name`).

Příkald cURL požadavku

```
curl -X POST "http://localhost/vian_vianapi/camera/update?id=c
↪  d0abc34-9bdd-42fe-8fef-0d5f8d814d77&url=12.12.12.12%3A1212
↪  &friendly-name=new%20name" -H  "accept: */*" -H  "user-tok
↪  en: 123"
```

5.8 Endpointy pro složené detekce: `/compound/*`

Jednotlivé složené detekce jsou definovány v konfiguračním souboru. V tomto souboru je pro každou takovou detekci specifikováno:

- `name`: název složené detekce
- `type`: typ složené detekce, který udává, zda je detekce prováděna offline nebo online
- `source_count`: udává, kolik zdrojů (tabulek) složená detekce využívá

- `filters`: atributy, podle kterých jsou výsledky detekce filtrovány - tyto hodnoty musí poté být zadány v těle požadavku spouštějícího tuto detekci
- `defaults`: implicitní hodnoty odpovídajících filtrů
- `columns`: metadata výsledku, tj. názvy a datové typy atributů, které jsou součástí výsledků složené detekce
- `sql`: definice SQL dotazu, který provádí samotnou složenou detekci

5.8.1 Informace o všech dostupných složených detekcích: `/compound/get-all`

Tento endpoint přečte informace z konfiguračního souboru a vrátí definice všech dostupných složených detekcí.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/compound/get-all"
↪  -H  "accept: */*" -H  "user-token: 123"
```

5.8.2 Informace o konkrétní složené detekci: `/compound/details`

Pro zadanou složenou detekci (její unikátní název je potřeba specifikovat jako parametr požadavku), zpřístupní celou definici této detekce.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/compound/details?ev
↪  ent-name=accidents_online" -H  "accept: */*" -H  "user-tok
↪  en: 123"
```

Příklad výsledku tohoto endpointu: Odpověď na úspěšný požadavek: 200


```
{
  "status": "success",
  "data": {
    "source_count": 1,
    "filters": [
      "rest_time",
      "car_count_threshold",
      "speed_px_min"
    ],
    "defaults": [
      4, 3, 5
```

```
    ],
    "columns": [
      {
        "colname": "frame_ts",
        "coltype": "numeric"
      }
    ],
    "sql": "SELECT public.accident_detection_online (<rest_tim
    ↪  e>, <car_count_threshold>, <speed_px_min>, '<s1>') AS
    ↪  frame_ts"
  }
}
```

5.8.3 Spuštění složené detekce: `/compound/query`

Je nutné specifikovat unikátní jméno složené detekce, a parametry `offset` a`row-count` (opět mají stejný význam jako tytéž atributy u endpointů `/detection/*`). V těle požadavku jsou specifikovány vstupní data a hodnoty jednotlivých filtrů. Na základě těchto vstupů budou jejich hodnoty vloženy do SQL dotazu z konfiguračního souboru a dotaz bude spuštěn.

Příklad cURL požadavku Zde je spuštěna složená detekce s názvem `accidents_time`, pracuje s jednou zdrojovou tabulkou (je zadána pomocí svého datasetu (`dataset`) a identifikátoru detekce (`config-id`). Dále následují hodnoty čtyř filtrů, které jsou uvedeny v konfiguračním souboru - pokud hodnota některého z filtrů není zadána, pak se použijí implicitní hodnoty z konfiguračního souboru.

```
curl -X POST "http://localhost/vian_vianapi/compound/query?eve
↪  nt-name=accidents_time&offset=0&row-count=10" -H  "accept:
↪  */*" -H  "user-token: 123" -H  "Content-Type: application/
↪  json" -d "{\"sources\":[{\"dataset\":\"test\",\"config-i
↪  d\":\"0fc639c8-9f63-4112-bc0a-bbb767a6e93d\"}],\"begin_fra
↪  me_time\":\"2019-10-07 14:01:00\",\"end_frame_time\":\"201
↪  9-10-07 14:05:00\",\"rest_time\":4,\"car_count_threshol
↪  d\":2,\"speed_px_min\":5}"
```

5.9 Přehrávání video sekvencí: `/replay/*`

Tyto endpointy využívají box pro přehrávání videa, které se využívají pro přehrávání sekvencí pocházejících z detekcí na ViAn Serveru. Je zde možná spustit přehrávání a ukončit přehrávání.

5.9.1 Spuštění přehrávání: `/replay/start`

Pro zadanou detekci (specifikovanou pomocí `config-id`), video sekvence je nalezena a přehrána. Endpoint vrací URL, na kterém je video sekvence dostupná a je možné ji přehrát pomocí programu ffplay. Dále je výstupem identifikátor přehrávací úlohy, který lze poté využít pro její zastavení. Nepovinně je možná zadat časový údaj pro posunutí ve videu.

Příklad cURL požadavku

```
curl -X POST "http://localhost/vian_vianapi/replay/start?detec
↪  tion=0fc639c8-9f63-4112-bc0a-bbb767a6e93d" -H  "accept: */
↪  *" -H  "user-token: 123"
```

5.9.2 Zastavení přehrávání video sekvence: `/replay/stop`

Po zadání identifikátoru úlohy přehrávání je toto přehrávání zastaveno. Nejsou zde žádné návratové hodnoty.

Příklad cURL požadavku

```
curl -X POST "http://localhost/vian_vianapi/replay/stop?replay
↪  -id=replay_afe1244f" -H  "accept: */*" -H  "user-token: 12
↪  3"
```

5.10 Podobnostní vyhledávání: `/similar/*`

Tato skupina endpointů slouží k využití funkcionality pro podobnostní vyhledávání na Vian Serveru.

5.10.1 Spuštění úlohy podobnostního vyhledávání: `/similar/run`

Tento endpoint slouží ke spuštění programu, který spouští úlohu podobnostního vyhledávání s použitím zadaných parametrů. Vzhledem k tomu, že samotné vyhledávání trvá při prohledávání většího objemu dat poměrně dlouho, proto je tímto endpointem vrácen identifikátor úlohy, tzv. `task-id`. Tento `task-id` je poté dále využit pro zjištění stavu úlohy a pro přístup k získaným výsledkům úlohy. Zde je seznam parametrů požadavku:

- `dataset`: název datasetu, kde se nachází data, ve kterých chceme provést podobnostní vyhledávání.
- `config_id`: identifikátor detekce, jejíž data chceme pro podobnostní vyhledávání využít
- `nn`: kolik nejbližších sousedů chceme pomocí podbnostního vyhledávání nalézt

- `low-time`: dolní mez časového intervalu, ve kterém chceme data prohledávat
- `high-time`: horní mez časového intervalu, ve kterém chceme data prohledávat
- `query-id`: identifikátor záznamu v tabulce, který představuje dotaz - hledáme nejpodobnější objekt k tomu, který je popsán vektorem rysů v tomto záznamu

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/similar/run?dataset
↪  =test&config-id=0fc639c8-9f63-4112-bc0a-bbb767a6e93d&nn=5&
↪  low-time=2019-10-07%2014%3A00%3A10&high-time=2019-10-07%20
↪  14%3A01%3A10&query-id=100" -H  "accept: */*" -H  "user-tok
↪  en: 132"
```

5.10.2 Zjištění stavu běžící úlohy podobnostního vyhledávání: `/similar/status`

Tento endpoint slouží ke zjištění stavu dané úlohy podobnostního vyhledávání. Je potřeba specifikovat její `task-id`. Pokud je vrácena hodnota "true", pak je již úloha dokončena a je možné načíst její výsledky.

Příklad cURL požadavků

```
curl -X GET "http://localhost/vian_vianapi/similar/status?task
↪  -id=task_5ff2dbe7b92c1" -H  "accept: */*" -H  "user-token:
↪  132"
```

5.10.3 Výsledky podobnostního vyhledávání: `/similar/results`

Tento endpoint slouží ke zpřístupnění výsledků podobnostního vyhledávání, pokud je již úloha dokončena. V požadavku je potřeba specifikovat identifikátor úlohy, tj. `task-id`. Výsledkem je seznam identifikátorů záznamů v tabulce, jejichž vektory rysů byly vyhodnoceny jako nejpodobnější.

Příklad cURL požadavku

```
curl -X GET "http://localhost/vian_vianapi/similar/results?tas
↪  k-id=task_5ff2dbe7b92c1" -H  "accept: */*" -H  "user-token
↪  : 132"
```

6 Společný základ instalačního manuálu API na ViAn Serveru a Snímacím uzlu

6.1 Požadavky na software

- PHP 7.2
- Apache
- Composer
- Docker
- GIT
- ffmpeg
- php-curl
- php-pdo
- python3

6.1.1 Požadavky na software pro API pouze na ViAn Serveru:

- PostgreSQL 10/11
- php-pdo-postgres

6.2 Instalační manuál společný pro ViAn Server i Snímací uzel

Zde se nachází pouze společná část instalace pro API na straně ViAn Serveru a na straně Snímacího uzlu. Konkrétní dílčí instalační manuály pro jednotlivá API jsou v následujících kapitolách.

Instalace GIT, Composer, Docker, ffmpeg

```
sudo apt install -yqq git
sudo apt install -yqq composer
sudo apt install -yqq docker.io
sudo apt install -yqq ffmpeg
sudo apt install -yqq python3
sudo apt install -yqq python3-pip
```

Instalace Apache, PHP a potřebných PHP modulů:

```
sudo apt install -yqq apache2
sudo apt install -yqq php
sudo apt install -yqq php-curl
sudo apt install -yqq php-pdo
sudo apt install -yqq php-pdo-pgsql
```

Poznámka: Balíček ***apache2*** *může být pojmenován rovněž* ***apache*** *nebo* ***httpd*** *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*

Konfigurace Apache

```
sudo usermod -aG docker www-data
sudo a2enmod ssl
sudo a2enmod rewrite
sudo sed -i '/DocumentRoot*/a \\t<Directory /var/www/html >\n\
↪  t\tAllowOverride All\n\t</Directory>'
↪  /etc/apache2/sites-available/000-default.conf
sudo sed -i '/DocumentRoot*/a \\t\t<Directory /var/www/html >\
↪  n\t\t\tAllowOverride All\n\t\t</Directory>'
↪  /etc/apache2/sites-available/default-ssl.conf
sudo a2ensite default-ssl
sudo systemctl restart apache2
```

Poznámky:

- *Skupina* ***www-data*** *může být pojmenována* ***apache*** *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*
- *Služba* ***apache2*** *může být pojmenována* ***httpd*** *nebo* ***apache*** *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*
- *Adresář* ***/etc/apache2/...*** *může být pojmenován* ***/etc/httpd/...*** *nebo* ***/etc/apache/...*** *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*

Nastavení členství aktuálního uživatele do skupiny, kterou využívá server Apache *Upozornění*: Po provedení následujícího příkazu je potřeba operační systém restartovat.

```
sudo usermod -aG www-data $USER
```

Poznámka: Skupina ***www-data*** *může být pojmenována* ***apache*** *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*

6.2.1 Instalační manuál - pokračování pouze pro ViAn Server

Instalace PostgreSQL 10

```
sudo apt install -yqq postresql-10
```

Vytvoření databáze a uživatele v PgSQL

```
echo "localhost:5432:vian:vian:vian2019fitbut" > ~/.pgpass
chmod 0600 ~/.pgpass
sudo su - postgres -c "psql -c \"CREATE USER vian LOGIN PASSWO
↪  RD 'vian2019fitbut'\";"
sudo su - postgres -c "psql -c \"CREATE DATABASE vian OWNER vi
↪  an\";"
```

Instalace rozšíření UUID do PgSQL

```
sudo su - postgres -c "psql vian -c \"CREATE EXTENSION IF NOT
↪  EXISTS \\\"uuid-ossp\\\"\";"
```

Stažení databázových skriptů

```
cd /tmp
git clone
↪  https://git.fit.vutbr.cz/vian/vianserver_database.git
```

Vytvoření databázové infrastruktury pro ViAn Server

```
cd vianserver_database
psql vian vian -h localhost -f vian_schema.pg.sql
psql vian vian -h localhost -f dbfunctions.sql
psql vian vian -h localhost -f dataset_test.pg.sql
```

7 Instalační manuál ViAn SensingAPI

7.1 Závislosti

Závislosti jsou spravovány skrze Composer. Aktuální závislosti lze nalézt v souboru `protected/composer.json` v repozitáři API.

7.2 Instalace ViAn SensingAPI

Instalace všech požadovaných součástí ViAnServeru

Před pokračováním v této části je třeba mít nainstalované a nastavené požadované součásti ViAnServeru popsané ve Společném základu instalačního manuálu z kapitoly 6.

Stažení zdrojů a nastavení vlastnictví

```
cd /var/www/html
sudo git clone
↪  https://git.fit.vutbr.cz/vian/vianserver_sensingapi.git
↪  vian_sensingapi
sudo chown -R $USER:$USER vian_sensingapi
```

Správa PHP závislostí skrze `Composer`

```
cd vian_sensingapi/protected
composer update
```

Konfigurace databáze v `Nette Frameworku`

```
cp app/config/config.local.neon.example
↪  app/config/config.local.neon
sed -i "s/user: ''/user: 'vian'/" app/config/config.local.neon
sed -i "s/password: ''/password: 'vian2019fitbut'/"
↪  app/config/config.local.neon
```

Volitelně lze také přizpůsobit umístění (cílový adresář) pro nahrávané video soubory, a to v souboru `app/config/config.local.neon` (sekce `constants` → položka `VIDEO_STORAGE` - výchozí hodnota je `/mnt/vian-video`).

Vytvoření složky pro nahrávání a úprava práv

```
mkdir -p <cesta z˜hodnoty konstanty VIDEO_STORAGE
 ↪  z˜předchozího kroku>
chown -R $USER:www-data <cesta z˜hodnoty konstanty
 ↪  VIDEO_STORAGE z˜předchozího kroku>
chmod -R g+w <cesta z˜hodnoty konstanty VIDEO_STORAGE
 ↪  z˜předchozího kroku>
```

Nastavení vlastnictví a práv pro zápis do systémových složek

```
chown -R $USER:www-data log
chown -R $USER:www-data temp
chmod -R g+w log
chmod -R g+w temp
```

Poznámka: Adresář `www-data` *může být pojmenován* `apache` *v závislosti na používané distribuci OS Linux a verzi serveru Apache.*

8 Instalační manuál VianAPI

Souhrnné informace o VianAPI naleznete v kapitole 5.

8.1 Závislosti

Všechny závislosti jsou spravovány pomocí nástroje Composer. Aktuální závislosti je možné najít zde: `protected/composer.json`.

8.2 Instalace ViAnAPI

Nejdříve je potřeba nainstalovat a nastavit všechny součásti Vian Serveru popsané ve Společném základu instalačního manuálu z kapitoly 6.

Stažení zdrojového kódu VianAPI a nastavení vlastnictví

```
cd /var/www/html
sudo git clone
↪  https://git.fit.vutbr.cz/vian/vianserver_vianapi.git
↪  vian_vianapi
sudo chown -R $USER:$USER vian_vianapi
```

Správa PHP závislostí pomocí Composeru

```
cd vian_vianapi/protected
composer update
```

Konfigurace databáze v Nette frameworku

```
cp app/config/config.local.neon.example
↪  app/config/config.local.neon
sed -i "s/user: ''/user: 'vian'/" app/config/config.local.neon
sed -i "s/password: ''/password: 'vian2019fitbut'/"
↪  app/config/config.local.neon
```

Volitelně lze přizpůsobit umístění, tj. cílový adresář pro nahrávání video souborů, a to v souboru `app/config/config.local.neon` (sekce `constants` -> item `VIDEO_STORAGE` - výchozí složka je `/mnt/vian-video`).

Vytvoření složky pro nahrávaná videa a nastavení práv

```
mkdir -p <path from VIDEO_STORAGE value of previous step>
chown -R $USER:www-data <path from VIDEO_STORAGE value of
↪  previous step>
chmod -R g+w <path from VIDEO_STORAGE value of previous step>
```

Nastavení vlastnictví a práv pro zápis do systémových adresářů

```
chown -R $USER:www-data log
chown -R $USER:www-data temp
chmod -R g+w log
chmod -R g+w temp
```

Pozn.: Skupina `www-data` *může být pojmenována* `apache` *v závislosti na operačním systému a verzi Apache serveru.*

9 Snímací uzel

Snímací uzel je velmi zjednodušeně řečeno výkonný výpočetní počítač / uzel, na kterém se nachází sada snímacích modulů (jako je například detektor aut a další), které mohou být na tomto uzlu spuštěny pro zpracování videa a takto zpracovaná data z videí (příslušné detekované objekty spolu s vlastními daty detekce) jsou odesílána na ViAn Server.

9.1 Součásti snímacího uzlu

Snímací uzel se skládá z následujících součástí

- `SeNoAPI` - Rozhraní REST API pro vzdálenou správu modulů (spouštění a ukončování).
- `VideoBoxes` - Sada nástrojů pro prác s videem, z nichž pro snímací uzel potřebný je například nástroj pro přidání značky videa, pro simulaci streamu z video souboru.
- `SensingMQ2ViAn Sender` - vyčítač fronty RabbitMQ, který data vyčtená z této fronty odesílá do ViAn Serveru (zjednodušení přenosu dat z modulu do ViAn Serveru pro vývojáře snímacích modulů jen na odeslání zprávy do fronty RabbitMQ místo řešení komunikace s REST API každým modulem samostatně - i s ohledem na možné změny v API)
- samotné snímací (výpočetní) moduly
- spouštěcí skript - Spouštěcí skript je spouštěn skrze `SeNoAPI` a jeho úlohou je zajistit vše potřebné pro běh modulu (nastartování příslušných nástrojů z `VideoBoxů`; zajistit nahrávání streamu, pokud je požadováno; spustit `SensingMQ2ViAn Sender` a nakonec rovněž spuštění vybraného snímacího modulu) nebo naopak ukončení modulu při požadavku na jeho ukončení a rovněž zajištění ukončení všech nástrojů, které byly tímto skriptem spuštěny, po ukončení modulu. *Poznámka: Tento skript může být rovněž použit pro spuštění modulu samostatně.*

9.2 Specifikace zpráv zasílaných moduly do MQ

```
{
  "vian_token": "some secret!!!",    # Token, kterým se snímací
  ↪  modul autentizuje (resp. je jím určeno, ke které
  ↪  instanci data patří).
  "vian_server_url": "https://vian.demo/sensingapi/", # URL
  ↪  ViAn Serveru
  "vian_project_id": "foo-bar",      # Dataset, kam budou data
  ↪  ukládána.
```

```
"header": { # Páry "klíč-hodnota", kde klíč položky značí
↪  jméno ukládané vlastnosti a~hodnota položky udává její
↪  datový typ
  # Povinné
  "frame_ts" : "float",           # Časová značka
  ↪  z~VideoBoxu Replicator

  # Pokud se používá položka se stejným významem, měla by
  ↪  být rovněž stejně pojmenována
  "obj_track_id" : "int",         # Identifikátor
  ↪  sledovaného objektu
  "obj_track_status" : "string",  # Stav sledování (hodnoty
  ↪  "New" - nový a~"Detected" - již dříve detekovaný
  "obj_class" : "string",         # Třída detekovaného
  ↪  objektu
  "obj_class_score" : "float",    # Skóre detekované třídy
  "obj_bbox" : "int[4]",          # Ohraničující obdélník
  ↪  detekovaného objektu (pořadí hodnot v~poli: zleva,
  ↪  zvrchu, zprava, zespod)
  "obj_bbox_score" : "float",     # Skóre detekovaného
  ↪  ohraničujícího obdélníku
  "obj_position" : "int[2]",      # Pozice objektu uvnitř
  ↪  snímku (typicky střed nebo hlavní bod)
  "obj_is_moving" : "bool",       # Příznak udávající, že se
  ↪  jedná o~pohybující se ojekt
  "obj_direction" : "float",      # Určení směru pohybu (dle
  ↪  standardní sémantiky úhlu)
  "obj_speed" : "float",          # Reální rycholst (v km/h)
  "obj_speed_px" : "float",       # Rychlost v~pixelech na
  ↪  snímek
  "obj_height" : "float",         # Reálná výška objektu (v
  ↪  metrech)
  "obj_height_px" : "float",      # Výška v~pixelech na
  ↪  frame
  "obj_feature_vector" : "float[128]", # Charakteristika
  ↪  objektu (tzv. feature vector)

  # Modulově-specifické položky
  "veh_class" : "string",
  "veh_class_prob" : "float",
  "veh_model" : "string",
  "veh_model_prob" : "float",
  "veh_color_class" : "string",
  "veh_color_class_prob" : "float"
```

```
  },
  "data": { # Páry "klíč-hodnota", kde klíč položky značí
  ↪  jméno ukládané vlastnosti a~hodnota udává její
  ↪  detekovanou hodnotu.
    "frame_ts": 12508125.12,
    "obj_class" : "car",
    "obj_class_score": 0.999,
    "obj_track_id": 124,
    "obj_bbox" : [20, 50, 80, 60],
    "obj_bbox_score" : 0.98989,
    # Zbývající klíče z~hlavičky s hodnotou dané vlastnosti /
    ↪  položky
    # ...
  }
}
```

Každá zpráva odeslaná stejnou instancí snímacího modulu by měla obsahovat všechny položky první úrovně ze specifikace zprávy výše včetně hlavičky (ta by měla být stejná po celou dobu běhu modulu).

Po dokončení zpracování videa (těsně před ukončením) je třeba zaslat zprávu se stejnou hlavičkou (jako u předchozích zpráv), abčak s prázdným objektem v datové položce.

```
{
  "vian_token": "some secret!!!",
  "vian_server_url": "https://vian.demo/sensingapi/",
  "vian_project_id": "foo-bar",
  "header": {
    "frame_ts" : "float",
    # Další definice vlastností / položek
  },
  "data": {}  # Prázdná data = Prázdný objekt v~datové položce
}
```

9.3 Požadavky na snímací moduly

Snímací modul odesílá výstupní data formou zpráv (dle výše definované zprávy) do fronty zpráv. Položky datové zprávy se během běhu instance nemění - není přípustné zaslat zprávy s různými hlavičkami v rámci jedné instance modulu. Je však možné položky s prázdnými daty v části `data` nezasílat, v položce `header` však musí být nadále uvedeny.

9.3.1 Vstupní parametry

Pro realizaci snímacího modulu jsou povinné následující vstupní parametry:

- `--vian <URL adresa ViAn SensingAPI>` - URL adresa ViAn SensingAPI, kam jsou zasílána data.
- `--dataset <jméno>` - Jméno datasetu, do kterého se data ukládají.
- `--access-token <token>` - Přístupový token pro daný konkrétní modul, resp. jeho instanci.
- `--stream <URL adresa replikátoru>` - URL adresa streamu (spuštěného přes replikátor), který bude zpracováván
- `--rmq-uri <adresa fronty>` - adresa fronty, do níž budou vkládány zprávy s daty *(pro zjednodušení a odstínění od komunikace skrze API v každém modulu zvlášť).*

9.3.2 Požadavky na formu a umístění

Snímací model musí být umístěn na Snímacím uzlu v jedné z následujících forem:

- `Docker kontejner` - Modul je zabalený do Docker kontejneru (preferovaná varianta), který je pojmenován dle modulu. Zároveň vstupní bod reaguje na výše uvedené požadované vstupní parametry.
- `Binární soubor nebo skript v souborovém systému` - Modul může být napsán v kterémkoliv skriptovacím jazyku (běžícím na OS Linux) nebo v kterémkoliv jazyku přeložitelném do binární podoby (na OS Linux). Umístění modulu může být libovolné, je však potřeba vytvořit `symlink` na spustitelnou podobu modulu. Symlink se musí nacházet v adresáři `/usr/bin/vian_modules/` *(pokud se adresář v systému nenachází je potřeba jej samozřejmě vytvořit).*

10 SensingMQ2ViAn Sender

Tento nástroj, napsaný v `C++`, zjednodušuje přenos dat mezi `snímacím modulem` a `ViAn serverem`. Hlavní výhodou je, že odstiňuje programátora konkrétního modulu od znalosti ohledně komunikace mezi modulem samotným a `ViAn serverem`. Programátor nadefinuje jména a datové typy pro výstup svého modulu, který má být uložen v úložišti. Tato data pomocí zprávy (definice zprávy je níže) vloží do fronty zpráv, konkrétně do `RabbitMQ`. Tento nástroj zabezpečuje zpracování zpráv vložených do fronty zpráv a následně jejich odesílání na `ViAn server`, konkrétně `ViAn SensingAPI`. Samotné odesílání dat probíhá nezávisle na pozadí a nezdržuje tak získávání dalších zpráv z fronty.

10.1 Princip fungování

Když SensingMQ2ViAn Sender přijme první data po svém spuštění, odesílá hlavičku skrze endpoint `/data/header` (kapitola 4.1.2) na ViAn Server. Následně již hlavičku ignoruje a zasílá pouze data (až do konce zpracování) skrze endpoint `/data/store` (kapitola 4.1.3) na ViAn Server.

Jakmile modul končí se zpracováním, odešle zprávu s prázdnými daty (prázdný objekt v datové položce zprávy). `SensingMQ2ViAn Sender` odešle všechna předchozí data z fronty na ViAn Server skrze ViAn SensingAPI a jakmile se dostane k této zprávě s prázdnými daty, bere ji jako signál toho, že data skončila a rovněž se ukončí.

`SensingMQ2ViAn Sender` může být spuštěn ve speciálním démon-režimu, který způsobí to, že se neukončí po obdržení prázdných dat. Po obdržení prázdných dat se příště znovu odešle hlavička jako by se `SensingMQ2ViAn Sender` znovu spustil od začátku (to je užitečné / vhodné například pro vývojové účely vývojářů snímacích modulů). *Poznámka: Tento mód způsobí, že se program sám neukončí, je třeba jej ukončit zasláním signálu z terminálu.*

10.2 Závislosti

- Pro binární podobu umístěnou fyzicky na počítači:
  - `C++` (se standardem C++-14)
  - `CMake` (minimální verze 3.10)
  - `pkg-config`
  - `RabbitMQ` [vyvíjeno ve verzi 3.7]
  - `libev-devel` (`libev-dev` / .. - závisí na distribuci OS Linux) [vyvíjeno s verzí 4.25]
  - `boost-devel` (`libboost-dev` / .. - závisí na distribuci OS Linux) [vyvíjeno s verzí 1.68]

  - `lib64curl-devel` (`libcurl4-openssl-dev` / .. - závisí na distribuci OS Linux) [vyvíjeno s verzí 7.66]
  - následující GIThub repositáře:
    * `copernica/amqp` (`cmake .. -DAMQP-CPP_LINUX_TCP=ON -DAMQP-CPP_BUILD_SHARED=ON`) [vyvíjeno s verzí 4.1]
    * `nlohmann/json` [vyvíjeno s vezí 0.8]
    * `jpbarrette/curlpp` [vyvíjeno s verzí 3.7]
- Pro vytvoření docker kontejneru:
  - `docker.io` (`docker` - závisí na distribuci OS Linux)

10.3 Překlad zdrojových kódů

`SensingMQ2ViAn Sender` využívá pro binární sestavení programu `CMake`. Pro přizpůsobení ladících výpisů slouží argument `VIAN_DEBUG`, jehož hlavní úrovně jsou následující:

- `0` - vůbec nezobrazuje ladící výpisy (`výchozí`)
- `1` - zobrazuje ladící výpisy požadavků
- `2` - zobrazuje ladící výpisy odpovědí
- `4` - zobrazuje běhové ladící výpisy

Tyto jednotlivé základní úrovně lze mezi sebou libovolně kombinovat operací `OR` (binární součet) - tímto způsobem lze využít ladící výpisy od úrovně `0` po úroveň `7`.

Pro přeložení `SensingMQ2ViAn Sender` je zapotřebí spustit následující množinu příkazů (pokud se nemá vytvořit fyzická binárka, ale docker kontejner, pak je tento krok zbytečný a lze jej přeskočit):

```
mkdir build
cd build
cmake [-DVIAN_DEBUG=<level>] ..
make
make install
```

10.4 Spuštění

`SensingMQ2ViAn Sender` má následující argumenty pro spuštění:

```
  -h (--help)         Výpíše nápovědu.
  -u (--vian) arg     URL ViAn Serveru, na který budou data
↪  zasílána a~kde budou zpracovávána / ukládána.
  -d (--dataset) arg  Dataset, do kterého budou data uložena.
```

```
  -t (--token) arg    Přístupový token k~autentizaci požadavku
↪  na API (rovněž k~určení, ke které instanci data patří).
  [--daemon-mode]     Démon-režim spuštění - (neukončuje se
↪  sám, je třeba jej zabít - užitečné pro vývojové účely
↪  vývojářů snímacích modulů)
```

Parametry `--vian`, `--dataset` a `--token` **jsou povinné!**

10.5 Docker kontejner

Repozitář obsahuje také předpřipravený skript pro zdárné vytvoření `Docker` kontejneru, který zajistí vše potřebné. Vytvořit docker kontejner je možné následujícím příkazem:

```
docker build -t sensingmq2vian_sender [--build-arg
↪  VIAN_DEBUG=<level>] .
```

Při spuštění kontejneru je třeba nezapomenout na všechny povinné parametry spuštění `SensingMQ2ViAn Senderu`:

```
docker run sensingmq2vian_sender --vian <ViAn URL> --dataset
↪  <jméno> --token <token>
```

11 Snímací uzel - SeNoAPI

SeNoAPI je rozhraní REST API poskytované snímacím uzlem, které umožňuje spustit nebo zastavit zpracování snímacího modulu. Součástí tohoto API je také přijmutí notifikace o nepoužívaném replikátoru pro účely jeho ukončení. SeNoAPI nabízí pouze velmi omezené množství endpointů:

- `/run` - Spustí zpracování vybraným snímacím modulem.
- `/stop` - Ukončí zpracování snímacího modulu (a umožní zaslat ještě neodeslaná data).
- `/kill` - Vynutí okamžité ukončení snímacího modulu (do té doby neodeslaná data nebudou uložena).
- `/notify/replicator/unused` - Slouží k ukončení nepoužívaného replikátoru (notifikaci zasílá ViAn Server).

11.1 Endpoint pro spuštění modulu: `/run`

Endpoint zajistí spuštění vybraného snímacího modulu skrze spouštěcí skript (který bude běžet na pozadí i po odeslání odpovědi na požadavek).

Metoda požadavku: `GET`

Parametry požadavku:

- `vian`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: URL ViAn serveru, kde budou data uložena (tj. kam se mají data zasílat).
- `access-token`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: Přístupový token pro kontrolu přístupu a také pro rozlišení instance modulu, ke které data patří.
- `dataset`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)

  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: Dataset, kam mají být data ukládána.

- `camera`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: URL streamu, který má být zpracováván snímacím modulem.

- `fetch-stream`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: volitelný
  - datový typ: boolean
  - výchozí hodnota: true
  - pois: Určuje, zda se má stream z kamery nahrávat na straně ViAn Serveru či nikoliv.

- `module`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: Výběr snímacího modulu, kterým bude předávaný stream zpracováván. video.

Příklad požadavku v cURL:

```
curl -X GET "https://localhost/sensingnode_senoapi/run" -H "ac
↪  cept: application/json" -H "vian: https://address.to.vian/
↪  vian_sensingapi" -H "access-token: ***" -H "dataset: test"
↪  -H "camera: rtmp://some.replicated.camera/stream" -H "modu
↪  le: traffic_analyser" -H  "Content-Type: application/json"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
```

```
  "data": {
    "smid": "12345678-90ab-cdef-0123-456789abcdef"
  }
}
```

Položka `smid` v datové části odpovědi je identifikátor spuštěné instance snímacího modulu - tento identifikátor je potřebný pro zastavení modulu.

11.2 Endpoint pro ukončení modulu: `/stop`

Endpoint zajistí ukončení snímacího modulu skrze zaslání specifického systémového signálu (USR1) spouštěcímu skriptu běžícímu na pozadí. Toto ukončení rovněž ukončí další dříve spuštěné součásti, avšak v případě `SensingMQ2ViAn Sender` umožní ještě před jeho ukončením zaslat dosud nezaslaná data, která se po ukončení modulu ve frontě zpráv nacházela.

Metoda požadavku: `GET`

Požadované parametry:

- `access-token`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: Přístupový token pro kontrolu přístupu a také pro rozlišení instance modulu, ke které data patří.
- `smid`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)
  - popis: ID zpracování (instance snímacího modulu) k ukončení.
- `vian`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: volitelný
  - datový typ: string (řetězec)
  - popis: Určení URL ViAn SensingAPI pro případ nekonzistentního stavu modulu.

- `dataset`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: volitelný
  - datový typ: string (řetězec)
  - popis: Určení datasetu pro případ nekonzistentního stavu modulu.

cURL request:

```
curl -X GET "https://localhost/sensingnode_senoapi/stop" -H "a
↪  ccept: application/json" -H "access-token: ***" -H "smid:
↪  some-processing-identifier" -H "Content-Type: application/
↪  json"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

11.3 Endpoint pro vynucené ukončení modulu: `/kill`

Endpoint zajistí vynucené ukončení snímacího modulu skrze zaslání specifického systémového signálu (USR2) spouštěcímu skriptu běžícímu na pozadí. Toto ukončení rovněž vynutí ukončení dalších dříve spuštěných součástí, avšak neumožňuje odeslat data nacházející se ve frontě zpráv, která ještě nestihla být odeslána.

Metoda požadavku: `GET`

RParametry požadavku:
Parametry požadavku jsou stejné jako v případě endpointu `/stop`.

Příklad požadavku v cURL:

```
curl -X GET "https://localhost/sensingnode_senoapi/kill" -H "a
↪  ccept: application/json" -H "access-token: ***" -H "smid:
↪  some-processing-identifier" -H "Content-Type: application/
↪  json"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

11.4 Endpoint pro informování o nepoužívaném replikátoru: `/notify/replicator/unused`

Tento endpoint slouží pro informování ze strany ViAn Serveru o již nepoužívaném replikátoru (zjištěném na základě registrací replikátoru). Pokud ViAn Server uvolní registraci replikátoru posledním uživatelem této služby, tak následně skrze tento endpoit dá vědět snímacímu uzlu, že může být příslušný replikátor ukončen.

Metoda požadavku: `GET`

Parametry požadavku:

- `replicator`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinné
  - datový typ: string (řetězec)
  - popis: URL již nepoužívaného replikátoru k ukončení.
- `owner-token`:
  - umístění parametru: předáván jako součást hlavičky požadavku (requestu)
  - povinnost: povinný
  - datový typ: string (řetězec)

– popis: token, kterým ViAn Server prokáže snímacímu uzlu legitimitu požadavku *(token je generován `ViAn Serverem`) při registraci replikátoru).*

Příklad požadavku v cURL:

```
curl -X GET "https://localhost/sensingnode_senoapi/notify/repl
↪  icator/unused" -H "accept: application/json" -H "replicato
↪  r: rtmp://some.replicated.camera/stream" -H "owner-token:
↪  ***" -H "Content-Type: application/json"
```

Odpověď na požadavek:
V případě úspěšného požadavku je zaslán HTTP stavový kód `200` s následující odpovědí:

```
{
  "status": "success",
  "data": []
}
```

11.5 Instalační manuál

11.5.1 Závislosti

Závislosti jsou spravovány skrze Composer. Aktuální závislosti lze nalézt v souboru `protected/composer.json` v repozitáři API.

11.5.2 Instalace SeNoAPI na snímací uzel

Instalace všech požadovaných součástí Snímacího uzlu

Před pokračováním v této části je třeba mít nainstalované a nastavené požadované součásti ViAnServeru popsané ve Společném základu instalačního manuálu z kapitoly 6.

Stažení zdrojů a nastavení vlastnictví

```
cd /var/www/html
sudo git clone
↪  https://git.fit.vutbr.cz/vian/sensingnode_senoapi.git
sudo chown -R $USER:$USER sensingnode_senoapi
```

Správa PHP závislostí skrze `Composer`

```
cd sensingnode_senoapi/protected
composer update
```

Vytvoření složky pro ukázková videa a úprava práv

```
# vytvoření adresáře
mkdir /mnt/vian-example-video

# NEBO
# vytvoření symlink na existující adresář
ln -s <existující cesta> /mnt/vian-example-video

# pro použití příkazu mount postupujte dle manuálových stránek
↪  příkazu
```

Nastavení vlastnictví a práv pro zápis do systémových složek

```
chown -R $USER:www-data log
chown -R $USER:www-data temp
chmod -R g+w log
chmod -R g+w temp
```

Poznámka: skupina `www-data` *může být pojmenována* `apache` *v závislosti na používané distribuci OS Linux nebo verze Apache.*

12 ViAn GUI

ViAn GUI je demonstrační operativní aplikace s grafickým uživatelským rozhraním. Jedná se o multiplatformní aplikaci založenou na technologiích python, fbs a PyQt, která je kompatibilní s operačními systémy Windows (Windows 7, Windows 10 apod.) a Linux (Ubuntu apod.). Tato aplikace komunikuje se zbytkem systému výhradně pomocí API (ViAn API a SensingNode API), což umožňuje její snadné spuštění i na externích počítačích, které jinak nejsou součástí detekčního systému.

Tato sekce se zabývá popisem základních pojmů a technologií, které se v rámci ViAn GUI využívají. Následně jsou zde také představeny konkrétní dostupné funkce z jednotlivých částí aplikace a to včetně krátkého popisu jejich obsluhy a možného nastavení.

12.1 Základní pojmy

V rámci popisu a práce s ViAn GUI je možné se setkat s následujícími pojmy a jejich důsledky, které jsou zde v krátkosti představeny.

Distribuovaný archiv

ViAn GUI je možné distribuovat v podobě zkompilovaného (přeloženého) programu pro konkrétní operační systém. V takovémto případě se jedná o před-připravený adresář, který obsahuje spustitelný soubor, konfigurační soubor a dodatečné knihovny potřebné pro samostatný běh programu bez závislostí na dalších částech operačního systému. Obsah tohoto adresáře je poté ještě typicky zabalen pomocí některého z komprimačních programů (např. zip) a distribuován jako jeden samostatný soubor (archiv).

Pro běžné použití ViAn GUI stačí tento dodaný archiv pouze rozbalit a v rozbaleném adresáři nalézt spustitelný soubor.

Spustitelný soubor

Přesný název spustitelného souboru se odvíjí od cílového operačního systému: `viangui.exe` pro Windows a `viangui` pro Linux. Tento soubor je přitom možné spustit buď bez parametrů nebo s parametry.

Spuštění bez parametrů

Pro spuštění bez parametrů stačí v grafickém uživatelském rozhraní pouze poklepat na spustitelný soubor. V takovémto případě se spustí hlavní okno programu, ze kterého je již možné se graficky navigovat do libovolných částí operativní aplikace.

Spuštění s parametry

Pro spuštění s parametry je nutné využít příkazovou řádku nebo terminál, kde se dodatečné parametry zadají za název spustitelného souboru. Tato forma spuštění je určena především pro snímací moduly a umožňuje jim přeskočit hlavní okno programu a zobrazit již přímo prohlížeč událostí pro konkrétní detekci s před-připraveným stavem uživatelského rozhraní.

Popis všech dostupných parametrů aktuální verze operativní aplikace lze zobrazit pomocí příkazu:

```
viangui --help
```

Konfigurační soubor

Součástí distribuovaného adresáře je také konfigurační soubor `config.yaml` ve formátu YAML, který umožňuje upravit předvolby aplikace před jejím spuštěním. Význam jednotlivých předvoleb je představován průběžně v následujícím textu a to vždy u popisu konkrétních části operativní aplikace, kde jsou v danou chvíli předvolby relevantní.

Uživatelský token

Pro zabezpečení přístupu k API se u jednotlivých požadavků zadává uživatelský token, který identifikuje a autentizuje daného uživatele. Uživatelský token se zadává v konfiguračním souboru v předvolbě `user_token`. V testovacím módu je možné v konfiguraci ponechat výchozí uživatelský token, který je součástí distribuovaného archivu. V jiných případech ale může být nutné token upravit na vlastní hodnotu.

12.2 Použité technologie

Následující část v krátkosti shrnuje technologie použité při vývoji ViAn GUI.

Python

Hlavní část operativní aplikace je napsána v jazyce Python, který je dostupný pro velkou řadu operačních systémů a zároveň umožňuje rychlý vývoj aplikací. Pro kompilaci zdrojových kódů do spustitelného souboru je využit nástroj `fbs` (`https://build-system.fman.io/`), který umožňuje snadný multiplatformní vývoj aplikací. Z důvodu omezení současné verze tohoto nástroje se pro vývoj ViAn GUI používá Python ve verzi 3.7.1. Pro novější verze Pythonu není zatím kompilace přes `fbs` úspěšná. Komunikace operativní aplikace s API je zajištěna pomocí knihovny `requests` (`https://pypi.org/project/requests/`).

Qt

Pro vytvoření multiplatformního uživatelského rozhraní byl využit framework Qt (`https://www.qt.io/`). Tento framework je napsaný v jazyce C a pro jeho připojení do Pythonu je využita knihovna `PyQt5` (`https://pypi.org/project/PyQt5/`). Přes tuto knihovnu je vykreslováno veškeré uživatelské rozhraní a mimo jiné také zajištěn vícevláknový běh aplikace.

Ffmpeg

Pro zpracování videa se využívá nástroj `ffmpeg` (`https://ffmpeg.org/`), který je napsán v jazyce C a do Pythonu je připojen přes knihovnu `av` (`https://pypi.org/project/av/`). Tato knihovna zajišťuje příjem a předzpracování videa z ViAn serveru, které je následně dodáváno do operativní aplikace v podobě série jednotlivých snímků videa s dodatečnými meta-informacemi (např. časovou značkou).

YAML

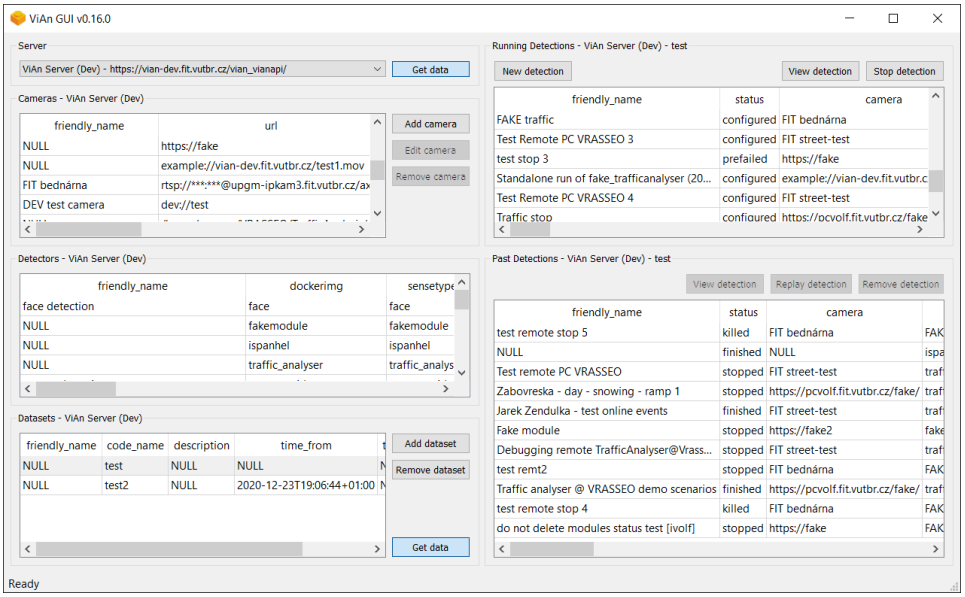
Pro zápis konfiguračního souboru byl zvolen formát YAML (`https://yaml.org/`), jenž umožňuje využívat pokročilejší konstrukce při zápisu konfigurací do textových souborů. Pro zápis a čtení konfiguračního souboru je využita knihovna `PyYAML` (`https://pypi.org/project/PyYAML/`) a pro ověření správnosti načtené konfigurace je využita knihovna `schema` (`https://pypi.org/project/schema/`).

12.3 Hlavní okno programu

Při spuštění ViAn GUI bez parametrů se otevře hlavní okno programu (viz Obrázek 5), které umožňuje spravovat značnou část informací uložených na ViAn serveru a také navigovat uživatele na další pod-okna aplikace pracující s detekovanými událostmi.

Výběr serveru

Jako první je nutné zvolit adresu ViAn serveru, ze kterého má ViAn GUI načítat data. Seznam dostupných ViAn serverů se zadává v konfiguračním souboru v části `servers`. Primární adresou je typicky hlavní ViAn server běžící v datovém centru. Je možné ale využívat i lokální verzi ViAn serveru, pokud se pracuje se stanicí v terénu, která se o správu všech informací stará sama bez připojení na hlavní server. V konfiguračním souboru v části `autoload.server` je také možné zvolit server, ke kterému se bude ViAn GUI automaticky připojovat při spuštění (ve výchozím stavu je to hlavní server).

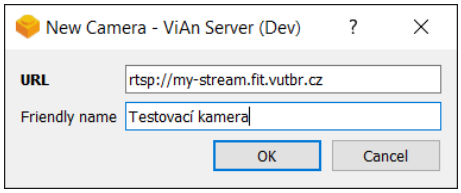


Obrázek 5: ViAn GUI – Hlavní okno

Správa kamer

Hlavní okno umožňuje správu (přidání, úpravu a mazání) kamer, které jsou uloženy na ViAn serveru a které je možné použít při spouštění snímacích modulů. Ukázka přidání nové kamery je vidět na Obrázku 6.

Poznámka: Kamera se ukládá na ViAn server a používá se ve snímacích uzlech, zadaná adresa musí být proto dostupná i na těchto zařízeních.



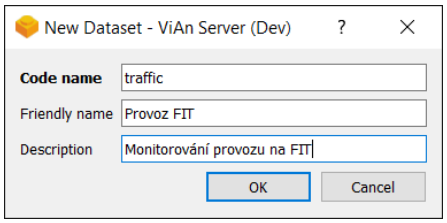
Obrázek 6: ViAn GUI – Přidání kamery

Správa a výběr datasetů

Na jednom ViAn serveru je možné pracovat s větším množstvím navzájem nesouvisejících detekcí. Uživatelé proto mohou využít různé datasety a zobrazit si tak pouze vybrané detekce, které patří do daného datasetu. ViAn GUI umožňuje správu (přidání a mazání) datasetů (viz Obrázek 7).

Načtení detekcí z datasetu je možné provést buď dvojklikem na daný řádek v tabulce datasetů anebo výběrem řádku a kliknutím na tlačítko *Get*

data. V konfiguračním souboru v části `autoload.dataset` je možné zadat `code_name` datasetu, který se má automaticky načíst při spuštění.

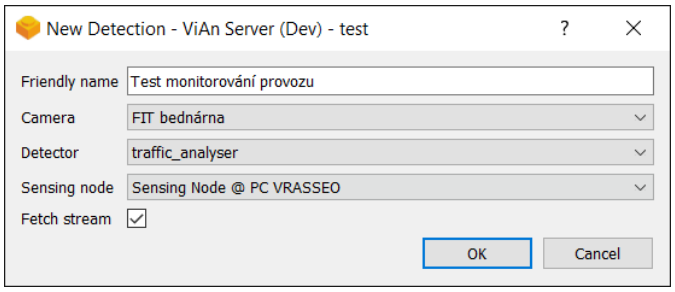


Obrázek 7: ViAn GUI – Přidání datasetu

Správa a zobrazení detekcí

Po výběru datasetu se na pravé straně hlavního okno zobrazí seznam probíhajících a dokončených detekcí. Tyto seznamy se aktualizují automaticky každých 60 sekund anebo je možné provést jejich okamžitou aktualizaci opakovaným výběrem datasetu nebo stiskem klávesy F5. Nastavení automatické aktualizace je možné změnit v konfiguračním souboru v části `detection_status_auto_refresh`.

ViAn GUI umožňuje spouštění nových detekcí přes tlačítko *New detection* (viz Obrázek 8). Při spuštění nové detekce je nutné zadat název této detekce a vybrat kameru, snímací modul a také snímací uzel, na kterém bude snímací modul spuštěn. Poslední položkou je možnost zapnutí nahrávání video streamu z kamery. Toto nahrávání je nutné zapnout v případech, kdy se mají dát detekované události zpětně zobrazit na videu. Po vytvoření nové detekce se automatická aktualizace seznamů přepne do zrychleného módu, kdy se aktualizace provádí každých 10 sekund, a tento mód je aktivní do doby, dokud je nová detekce ve stavu *launching*. Toto nastavení lze opět ovlivnit v konfiguračním souboru. Operativní aplikace dále umožňuje zastavit běžící detekce a smazat dokončené detekce.



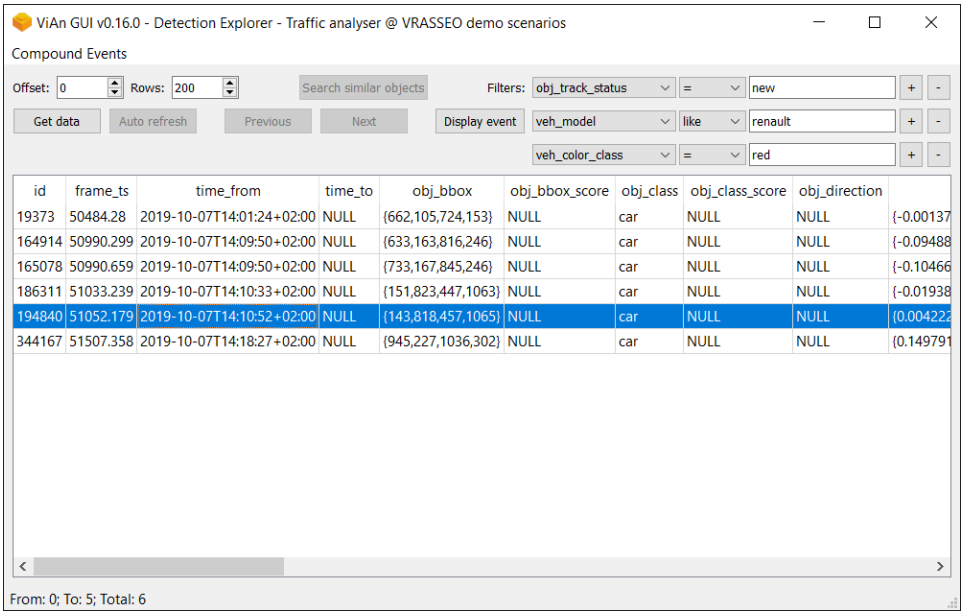
Obrázek 8: ViAn GUI – Spuštění nové detekce

Po výběru konkrétní detekce je možné přejít do okna prohlížeče událostí, toto je možné provést buď dvojklikem na daný řádek v tabulce detekcí anebo

přes tlačítko *View detection*. U dokončených detekcí je navíc možné přejít do okna přehrávání záznamu přes tlačítko *Replay detection*, pokud bylo u dané detekce zapnuto nahrávání video streamu.

12.4 Prohlížeč událostí

V okně prohlížeče událostí je možné pracovat s událostmi, které byly zachyceny snímacím modulem v rámci zvolené detekce. Ukázka prohlížeče událostí je vidět na Obrázku 9. V této ukázce jsou vyfiltrovány události zachycující nové detekce aut značky Renault červené barvy, které byly nasnímány v rámci jednoho z demo scénářů.



Obrázek 9: ViAn GUI – Prohlížeč událostí

Běhové módy

Okno prohlížeče událostí se spouští ve dvou odlišných módech a to v závislosti na tom, zda jde o běžící nebo dokončenou detekci. Pro každý mód jsou v tomto okně přednastaveny jiné výchozí hodnoty maximálního počtu zobrazovaných událostí a také jiný výchozí způsob jejich řazení. Pro běžící detekce lze navíc přes tlačítko *Auto refresh* zapnout automatickou aktualizaci tabulky zobrazených událostí, tato tabulka se poté aktualizuje každých 5 sekund a nové události jsou v ní vyznačeny žlutou barvou. Ruční aktualizaci tabulky lze provést pomocí tlačítka *Get data* nebo stiskem klávesy F5. Nastavení běhových módů jde upravit v konfiguračním souboru v sekci `detection_explorer`.

Navigace

Hodnota *rows* udává maximální počet událostí, které se mohou v tabulce současně zobrazit. Hodnota *offset* poté udává počet událostí, které byly při zobrazování výsledku přeskočeny. Pro posun ve výsledcích lze ale využít i tlačítka *Previous* a *Next* a hodnotu *offset* tak není nutné při malých posunech měnit ručně. Výsledky dotazu lze také řadit a to podle libovolného sloupce tabulky. Změnu řazení lze provést kliknutím na hlavičku příslušného sloupce. Celkový počet událostí odpovídajících zadanému dotazu lze nalézt na stavovém řádku ve spodní části okna.

Filtrování

Prohlížeč událostí umožňuje využít libovolné množství filtrů, které redukují počet relevantních událostí k zobrazení. Filtry lze přidávat tlačítkem + a odebírat tlačítkem −. Filtr se vždy vztahuje ke konkrétnímu sloupci tabulky a využívá jeden z následujících operátorů:

- = – porovnání na přesnou shodu,
- > – větší než zadaná hodnota,
- < – menší než zadaná hodnota,
- >= – větší nebo rovno zadané hodnotě,
- <= – menší nebo rovno zadané hodnotě,
- *between* – je v rozmezí zadaných hodnot,
- *like* – zadaná hodnota je podřetězcem,
- *in* – porovnání na přesnou shodu s některou z hodnot (jednotlivé hodnoty jsou od sebe odděleny pomocí středníku).

Obrázek 10: ViAn GUI – Zobrazení události

Zobrazení události

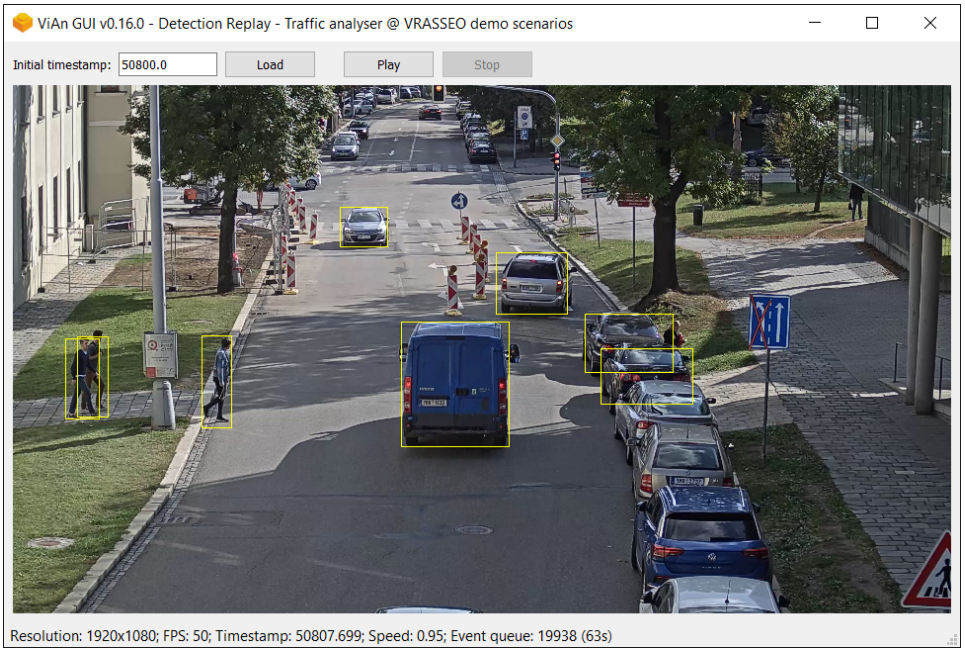
Pokud bylo u vybrané detekce zapnuto nahrávání video streamu, je možné zobrazit vybranou událost na videu a to buď pomocí dvojkliku na příslušný řádek v tabulce anebo stisknutím tlačítka *Display event*. Ukázku zobrazení události je možné vidět na Obrázku 10. Vybraná událost je na snímku vyznačena zeleným obdélníkem. Žluté obdélníky poté vyznačují ostatní události se stejnou časovou značkou.

Ostatní

U výsledků z vybraných snímacích modulů je dále možné využít další přidané funkce. Stiskem *Compound Events* je možné se dostat do okna pro vyhledávání složených událostí. Stiskem *Search similar objects* je možné vyhledávat související objekty na základě podobnostních rysů.

12.5 Přehrávání záznamů

Při přehrávání záznamu si operativní aplikace vyžádá od ViAn serveru nahraný video stream i seznam všech detekovaných událostí. Tyto události jsou poté při přehrávání videa vyznačovány na snímcích dle příslušných časových značek pomocí žlutých obdélníků. Přehrávání videa je možné spouštět a zastavovat stiskem mezerníku anebo pomocí tlačítek *Play* a *Stop*. Záznam je také možné posunout na konkrétní časovou značku.



Obrázek 11: ViAn GUI – Přehrávání záznamů

13 Kompilační manuál ViAn GUI

Z uživatelského pohledu je možné ViAn GUI distribuovat v podobě zkompilovaného programu pro konkrétní operační systém (Windows, Ubuntu apod.). Uživatel pouze spustí dodaný spustitelný soubor a program běží samostatně bez dalších závislostí v systému. Tato kapitola popisuje postup, jak ViAn GUI zkompilovat z GIT repozitáře projektu a zdrojových kódů programu. Kompilaci je přitom možné provést buď přes Python nebo Docker.

13.1 Závislosti

- podporovaná platforma: Windows nebo Linux (Ubuntu apod.)
- GIT (repozitář: `https://git.fit.vutbr.cz/vian/viangui`)
- Python (využívána je verze 3.7.1) a/nebo
- Docker a Docker Compose (umožňuje kompilaci pouze pro Linux)

13.2 Příprava prostředí pro překlad přes Python

Před kompilací samotného programu je nutné připravit izolované prostředí Pythonu. Postup vytvoření izolovaného prostředí se může lišit v závislosti na použitém operačním systému. Následně je nutné do tohoto prostředí ještě doinstalovat potřebné závislosti.

13.2.1 Vytvoření izolovaného prostředí na Windows

Je nutné použít Python ve verzi 3.7.1 a pomocí něj vygenerovat izolované prostředí nad repozitářem:

```
py -3.7 -m venv .venv
.venv\Scripts\activate
```

13.2.2 Vytvoření izolované prostředí na Linuxu

Na distribucích Linuxu pro přípravu izolovaného prostředí využijeme nástroj *penv*. Instalaci *pyenv* je možné provést například podle návodu na `https://realpython.com/intro-to-pyenv/`. Pomocí tohoto nástroje se poté nechá připravit celý Python 3.7.1 včetně izolovaného prostředí:

```
env PYTHON_CONFIGURE_OPTS="--enable-shared" pyenv install 3.7.1
pyenv virtualenv 3.7.1 viangui
pyenv local viangui
```

13.2.3 Instalace závislostí do izolovaného prostředí

Instalace závislostí probíhá na obou typech operačních systémů stejně:

```
pip install PyQt5
pip install fbs
pip install requests
pip install av
pip install PyYAML
pip install schema
```

Další informace ohledně řešení případných potíží při instalaci konkrétních verzí jednotlivých závislostí lze nalézt v `README.md`.

13.3 Překlad přes Python

Spuštění ViAn GUI je možné provést pomocí příkazu:

```
fbs run
```

Kompilaci ViAn GUI je možné provést přes:

```
fbs freeze
```

13.4 Překlad přes Docker

Při využití Dockeru je kompilace přes Python zapouzdřena v kontejneru. Tento kontejner nad cílovým operačním systémem vše nainstaluje, zkompiluje a vygeneruje výsledný komprimovaný archiv se soubory k distribuci.

V projektu je nachystána kompilace pro operační systém Ubuntu 18.04, kterou je možné provést pomocí následujících příkazů nad repozitářem:

```
cd docker-deploy
docker-compose up --build
```