Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie

Zvyšovanie bezpečnosti integrácie dronov do leteckého priestoru

Bakalárska práca

AeroGuardian

Užívateľská príručka

Vedúci bakalárskej práce:

doc. Ing. Peter Papcun, PhD.

Autor:

Štefan Kando

Košice 2024

Obsah

1 Funkcia aplikácie

Aplikácia je určená pre pilotov dronov využívajúcich drony na súkromné a rekreačné účely. Slúži ako platforma na spracovanie a vizualizáciu dát prijatých z okolitej letovej prevádzky. Zároveň umožňuje pilotom dronov si svoje lety plánovať (s bodom vzletu, vzdialenosťou od bodu vzletu a výškou) a zistiť bezpečnosť svojich plánovaných letov vzhľadom na okolitú prevádzku. Tá istá aplikácia ho dokáže v reálnom čase varovať pred náhle vzniknutými hrozbami. Aplikácia potrebuje k fungovaniu zdroj dát o letovej prevádzke, bola vyvíjaná predovšetkým na získavanie dát z ADS-B prijímača (obrázok 3 – 2). Aplikácia má taktiež koncový bod pre Icecast, ktorý je používaný na odpočúvanie leteckého rádia, avšak keďže sa nám letecké rádio nepodarilo softvérovo spracovať, prijímanie leteckého pásma nie je súčasťou tejto príručky.

2 Aplikačný výstup

Základné komponenty aplikácie sú rozdelené do troch adresárov:

- `backend-python` obsahuje skripty pre fungovanie back-endu vytvorenom v Pythone vo frameworku `Flask`,
- `frontend-react` obsahuje front-end vytvorený v JavaScript knižnici `React`,
- `utilities` obsahuje rôzne skripty v Pythone a SQL skripty pre PostgreSQL pre spracovanie dát a predovšetkým vytvorenie mapy výskytu letovej prevádzky z nazbieraných dát (nie je nutné pre fungovanie hlavnej aplikácie).

K aplikácii je priložená používateľská a systémová príručka.

3 Inštalácia aplikácie

Hlavná aplikácia sa delí na front-end, back-end a prijímanie dát cez ADS-B. V tejto príručke bude uvedený stručný návod na inštaláciu a prácu so všetkými časťami

tejto aplikácie, ako aj s niektorými ďalšími (nepovinnými), predovšetkým na predspracovanie a prehĺbenie funkcionality aplikácie.

3.1 Požiadavky na technické prostriedky

- Back-end a front-end sú navrhnuté tak, aby fungovali na Raspberry Pi (RPi), konkrétne na modeli Raspberry Pi 5 s 8GB RAM.
- Aplikácia samotná je optimalizovaná tak, aby nepotrebovala vysoký výkon. Niektoré skripty na predspracovanie dát, ako je vytvorenie mapy výskytu letovej prevádzky, využívajú multiprocessing pre rýchlejšie spracovanie. Generovanie dát bolo vykonané na počítači s procesorom Intel i5 13600K a 32GB RAM. Predvolené hodnoty sú prispôsobené pre tento typ hardvéru. Všetky parametre generovania sú upraviteľné.
- Pre prijímanie ADS-B signálu je možné použiť ľubovoľné SDR (Software Defined Radio) schopné prijímať frekvenciu 1090 MHz. Odporúča sa použitie SDR, ako je FlightAware Pro Stick Plus (FAPSP), ktoré podporuje filtráciu a predspracovanie ADS-B signálu.
- Je potrebná anténa navrhnutá na 1090 MHz s dobrým výhľadom na oblohu.
- SDR musí byť pripojený k RPi.
- Bloková architektúra systému a hardvéru je znázornená na obrázku 3 – 2.

3.2 Požiadavky na programové prostriedky

- Aplikácia bola testovaná na operačnom systéme Raspbian (Linux) a Windows 11.
- Back-end aplikácie používa Python, najstaršia testovaná verzia je 3.9, s frameworkom `Flask`. Ku každému Python skriptu je priložený súbor `requirements.txt` obsahujúci požadované knižnice.

- Front-end používa na spustenie a preloženie platformu `Node.js`. Pri vývoji beží aplikácia priamo na `Node.js`, pri nasadení sa odporúča použiť webový server `Nginx`.
- Aplikácia spracúva ADS-B signály s využitím SDR prostredníctvom `dump1090-fa`, ktorý je špeciálne navrhnutý pre operačný systém Raspbian.
- Na spracovanie dát je využívaný systém PostgreSQL.
- Skript na generáciu dlaždíc mapy výskytu letovej prevádzky využíva systémovú knižnicu `GDAL` a externý skript `gdal2tiles.py`, ktorý je zahrnutý napríklad v distribúcii Pythonu `anaconda3`.

3.3 Vlastná inštalácia

Pri vlastnej inštalácii hlavnej aplikácie máme na výber, či chceme aplikáciu spustiť pre potreby vývoja, alebo pre potreby hosťovania. Ukážeme si taktiež, ako nainštalovať služby potrebné pre prijímanie ADS-B a ako inštalovať služby a spúšťať skripty pre spracovanie prijatých dát.

3.3.1 Inštalácia dump1090-fa / PiAware / SkyAware (RPi)

Tieto služby sú určené na prijímanie ADS-B a k fungovaniu potrebujú SDR s anténou schopné prijímať 1090 MHz.

Návod na nainštalovanie `dump1090-fa`, spolu s `PiAware` a `SkyAware` na operačný systém Raspbian je dostupný na `https://www.flightaware.com/adsb/piaware/install`. Služba `dump1090-fa` vytvára súbor `aircraft.json`, ktorý je priamo spracovateľný na tom istom zariadení back-endom. Služba `SkyAware` tento súbor hosťuje na `/data/aircraft.json`, na ktorý je taktiež back-end pripojiteľný.

3.3.2 Spustenie back-endu a front-endu (príkazový riadok)

Back-end v adresári `backend-python` sa spúšťa nasledovne:

1. Vytvoríme si virtuálne prostredie pomocou príkazu `python -m venv venv`
2. Aktivujeme virtuálne prostredie:
   - Windows: `call venv/Scripts/activate`
   - Raspbian (Linux): `source venv/bin/activate`
3. Nainštalujeme knižnice zo súboru `requirements.txt` pomocou príkazu `pip install -r requirements.txt`
4. Spustíme aplikáciu pomocou príkazu `python main.py`

Pokiaľ máme nainštalovaný `Node.js`, potrebné knižnice front-endu nainštalujeme v adresári `frontend-react` pomocou príkazu `npm install` v adresári . Pre potreby vývoja a debugu použijeme príkaz `npm run dev`.

Pre deployment / hosťovanie pozri 3.3.5.

3.3.3 Úprava programových premenných

Skript back-endu obsahuje flag `debug` v riadku 35, ktorým sa prepína medzi hosťovaním pre vývoj a hosťovaním pre deployment. V riadku 56 sa dá upraviť `base_folder` na priečinok s dlaždicami na hosťovanie mapy výskytu letovej prevádzky. V riadku 94/95 sa dá nastaviť zdroj dát úpravou premennej `url` (dá sa nastaviť na IP Sky-Aware, v prípade lokálneho `dump1090-fa` sa dá nastaviť aj na lokálne vytváraný súbor `aircraft.json`). V skripte `check_breach.py` sa dá v riadku 26 upraviť premenná `vertical_overhead` na nastavenie bezpečného vertikálneho odstupu.

Nastavenia front-endu sa nachádzajú v `src/components/StaticSettings.jsx`. V riadku 25 sa dá úpravou premennej `BackendIP` nastaviť IP back-endu. V riadku 31 sa úpravou `IcecastIP` nastavuje IP pre koncový bod Icecastu. Nastavenia ovládacieho panela sa nastavujú v riadku 34 v premennej `ControlPanelBounds`. Bezpečný

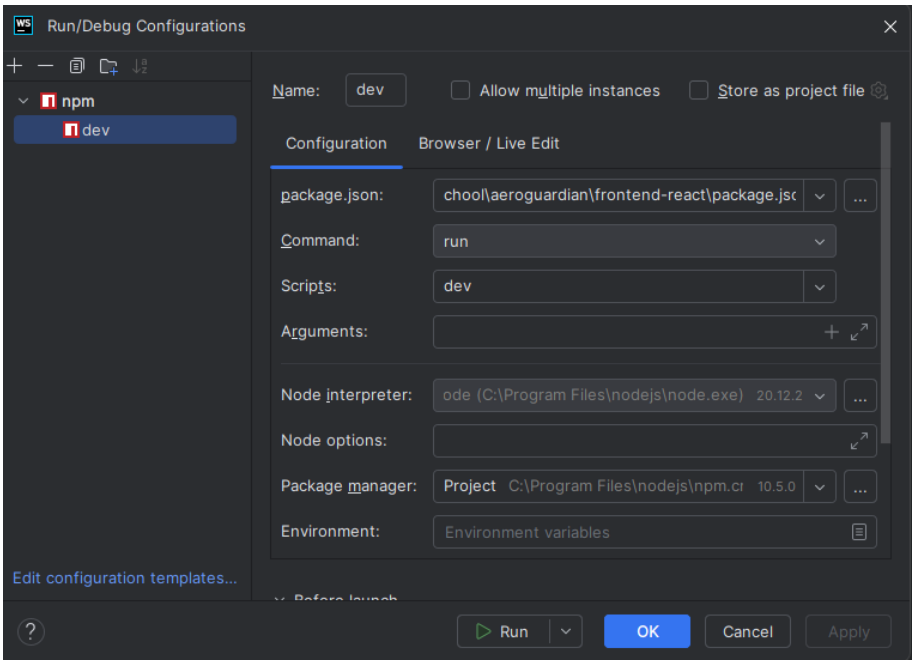
vertikálny odstup sa dá prenastaviť v riadku 43 v premennej `VerticalOverhead`. V riadku 46 v premennej `DefaultMapSettings` sa nastavujú pôvodné nastavenia pre stred a priblíženie mapy.

3.3.4 Odporúčané postupy pre vývoj aplikácie

Pri vývoji aplikácie je odporúčané používať vývojové prostredia od JetBrains, pre Python PyCharm a pre React WebStorm.

PyCharm pri otvorení adresára `backend-python` automaticky vytvorí virtuálne prostredie, kde nainštaluje knižnice uvedené v `requirements.txt`. Potom už len stačí si otvoriť `main.py` v projektovej štruktúre a spustiť projekt cez `Shift + F10`.

WebStorm nám pri otvorení adresára `frontend-react` ponúkne možnosť nainštalovať používané knižnice zo súboru `package.json` pomocou príkazu `npm install`. Pre spustenie si musíme nastaviť konfiguráciu `npm`, príkaz `run` a skript `dev` (obrázok 3 – 1).



Obr. 3 – 1: Konfigurácia prostredia WebStorm pre spustenie aplikácie

3.3.5 Hosťovanie (Raspbian / Linux)

Pre hosťovanie na Raspbiane použijeme Linuxovú službu na spúšťanie back-endu cez jeho virtuálne prostredie. Pre vytvorenie virtuálneho prostredia pozri 3.3.2. Pre hosťovanie back-endu sa taktiež odporúča vypnúť flag `debug` v riadku 35 v `main.py`. Na front-end použijeme webový server `nginx`.

1. Vytvorenie systémovej služby pre back-end:

   - Vytvoríme nový súbor služby pomocou príkazu
     `sudo nano /etc/systemd/system/backend.service`
   - Do súboru vložíme nasledovný obsah
     (upravte cesty k zložke, `venv` a `main.py`):

```
[Unit]
Description=Back-end Service
After=network.target

[Service]
User=pi
WorkingDirectory=/cesta/k/vašej/zložke/backend-python
ExecStart=/cesta/k/backend-python/venv/bin/python
  (pokračovanie) /cesta/k/backend-python/main.py
Restart=always

[Install]
WantedBy=multi-user.target
```

   - Uložíme a zatvoríme súbor (`Ctrl + X`, `Y`, `Enter`)
   - Načítame novú službu pomocou príkazu `sudo systemctl daemon-reload`
   - Spustíme službu príkazom `sudo systemctl start backend.service`

- Nastavíme automatické spúšťanie služby príkazom
  `sudo systemctl enable backend.service`

2. Pred hosťovaním Reactu si ho musíme dať do `build` podoby:
   - Prejdeme do adresára s front-endom pomocou príkazu
     `cd /cesta/k/frontend-react`
   - Spustíme buildovanie projektu príkazom `npm run build`
3. Inštalácia a konfigurácia `nginx` pre front-end:
   - Nainštalujeme `nginx` pomocou príkazu `sudo apt-get install nginx`
   - Upravíme konfiguráciu servera pomocou príkazu
     `sudo nano /etc/nginx/sites-available/default`
   - Do súboru vložíme nasledovný obsah (upravte cestu k frontend-react a ostatné nastavenia podľa potreby):

```
server {
listen 80;
server_name your_domain_or_IP;

location / {
root /cesta/k/frontend-react/build;
index index.html index.htm;
try_files $uri $uri/ /index.html;
}



location /api/ {
proxy_pass http://localhost:5000/;
proxy_set_header Host $host;
```

```
(pokračovanie)
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Forwarded-Proto $scheme;
}}
```

- Uložíme a zatvoríme súbor (`Ctrl + X`, `Y`, `Enter`)
- Reštartujeme `nginx` pomocou príkazu `sudo systemctl restart nginx`

3.3.6 Získavanie polohy dronu v reálnom čase

Univerzálny spôsob získavania polohy dronu v reálnom čase nebol vyvinutý. Na prijímanie polohy dronu má back-end koncový bod `/drone_location`, kde sa poloha dronu posiela metódou `POST` vo formáte `JSON`, ktorý má kľúče `latitude`, `longitude` a `altitude` (v metroch). Pre potreby demonštrácie s naším dronom (DJI Mini 3 Pro) sme využili úpravu ukážkového kódu DJI Mobile SDK, dostupnú na GitHube: `https://github.com/dji-sdk/Mobile-SDK-Android`. Metódu POST na náš back-end server sme vykonali úpravou mapového komponentu, ktorý sa nachádzal v súbore `SampleCode-V5/android-sdk-v5-uxsdk/src/main/java/dji/v5/ux/map/MapWidget.java:657`, funkcia `updateAircraftLocation()`. Polohu dronu sme získali z objektu `LocationCoordinate3D`.

3.3.7 Ukladanie dát do PostgreSQL (nepovinné)

Ak máme nainštalovanú databázu PostgreSQL, musíme si ju inicializovať skriptom v `utilities/postgre/aeroguardian-init.sql`. V inicializovanej databáze musíme upraviť tabuľku `sources`, do ktorej musíme pridať vstup o našom prijímači / našich prijímačoch. Do kolónky `source_id` pridáme akýkoľvek unikátny číselný identifikátor, do `receiver_name` pridáme vymyslené meno nášho prijímača a do `receiver_website` pridáme adresu SkyAware. Táto adresa musí byť dostupná zo zariadenia, na ktorom spustíme logger.

Skript na logging zo SkyAware prijímačov v tabuľke `sources` do tabuľky `aircraft_pointcloud` sa nachádza v `utilities/logger/logger.py`. Knižnice potrebné pre logger sú uložené v súbore `utilities/requirements.txt`. Logger sa odporúča spustiť ako službu, analogicky k spúšťaniu back-endu, viď 3.3.5 (použite iný názov pre službu).

3.3.8 Inštalácia OpenTopoData (nepovinné)

OpenTopoData je server, ktorý dokáže poskytovať skriptom v predspracovaniu výškové dáta zeme. Pre Európu odporúčame použiť dataset OpenTopoData – `https://www.opentopodata.org/datasets/eudem/`.

OpenTopoData síce má verejne hosťovaný server, ale ten má obmedzenia, čo ho robia nepoužiteľným pre naše potreby. Dá sa však hosťovať lokálne. Repozitár aj s návodom je dostupný na `https://github.com/ajnisbet/opentopodata`.

OpenTopoData je využívaný v predspracovaní dát, viď kapitola 5.

3.3.9 Generovanie máp výskytu prevádzky (nepovinné)

Skripty na spracovávanie dát, vrátane generovania máp výskytu prevádzky, sa nachádzajú v adresári `utilities`.

Pred generovaním máp je potrebné mať inicializovanú PostgreSQL databázu (viď kapitola 5), plnú dát o lietadlách v oblasti, v ktorej chceme generovať (`logger`), a naplnenú výškovými dátami z OpenTopoData (`OpenTopoData-postgre/fill_grid_height`).

Generovanie máp výskytu prevádzky sa pre vysokú výpočtovú náročnosť neodporúča vykonávať na RPi.

Hlavný skript na generovanie máp (`generate_heatmaps_main.py`) sa nachádza v adresári `heatmap`.

Pred spustením je potrebné v `generate_tiles.py` upraviť referenciu na skript `gdal2tiles.py` v riadku 42 (viď 3.2).

V skripte `generate_heatmaps.py` je možné v riadkoch 31 až 44 definovať polygóny letiska a pristávacej dráhy na vynechanie z gradientu. Nápisy letísk na mape

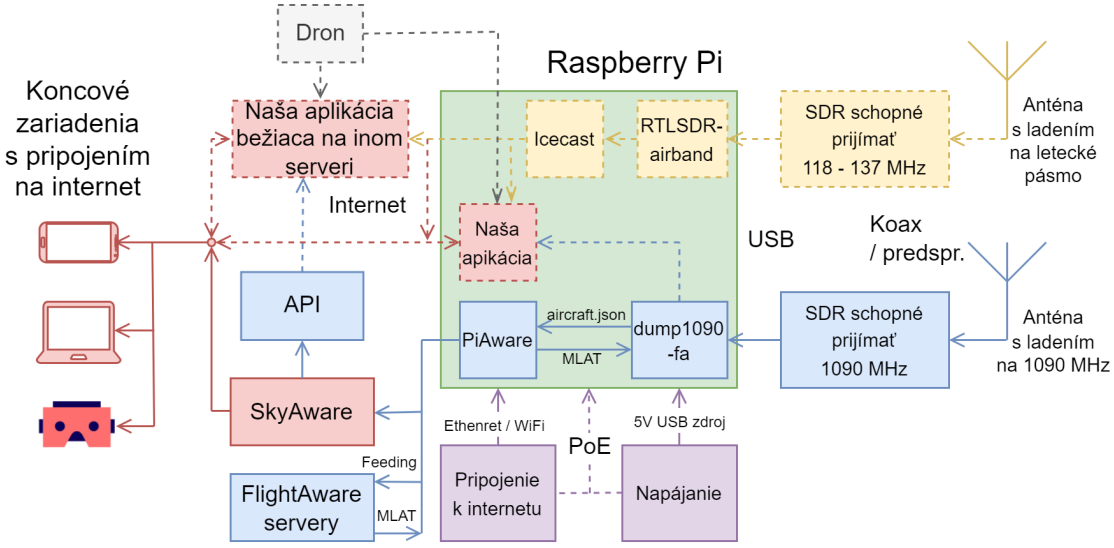
sa dajú upraviť v riadkoch 119 – 120.

Pred spustením hlavného skriptu si pozorne prečítajte a skontrolujte všetky nastavenia, ktoré sa v ňom nachádzajú. Nastavenia sú okomentované v slovenčine.

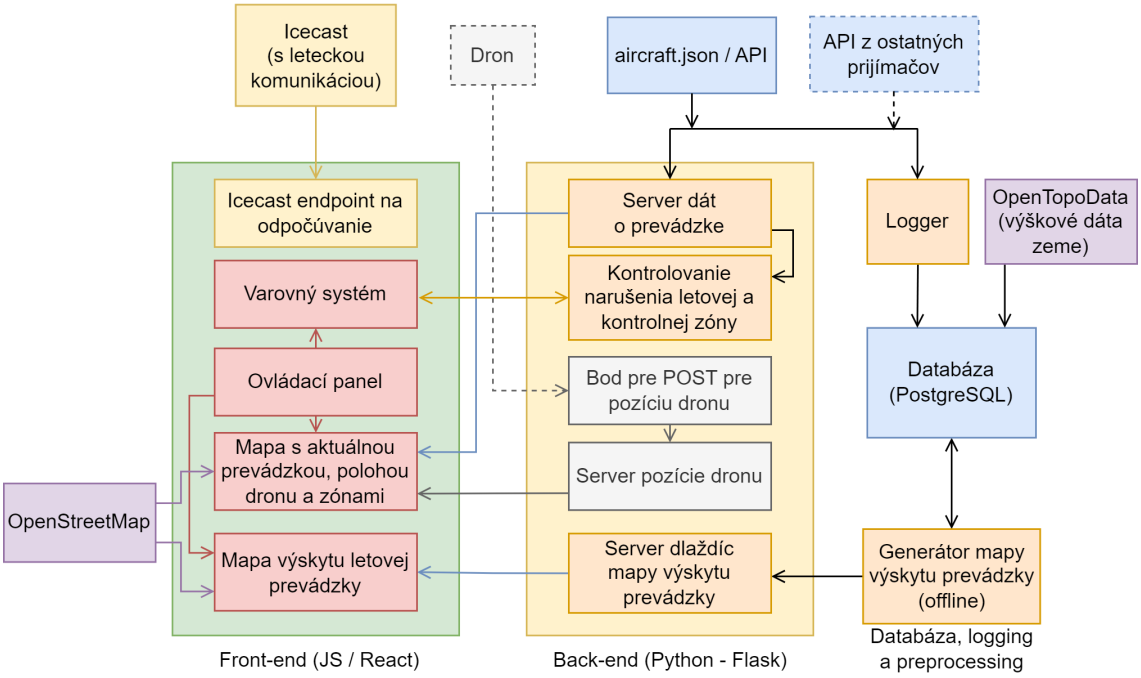
3.4 Popis štruktúry aplikácie

Na obrázku 3 – 2 je zobrazená bloková architektúra aplikácie. Na obrázku 3 – 3 je zobrazená softvérová architektúra aplikácie.

Front-end čerpá dáta z back-endu, ktorý zasa čerpá dáta z ADS-B (PiAware). Aby aplikácia fungovala správne, je potrebné, aby boli k dispozícii a správne nastavené tieto tri časti aplikácie. Podrobnejšie fungovanie aplikácie je vysvetlené v systémovej príručke.



Obr. 3 – 2: Bloková architektúra prijímania rádiových signálov a našej aplikácie



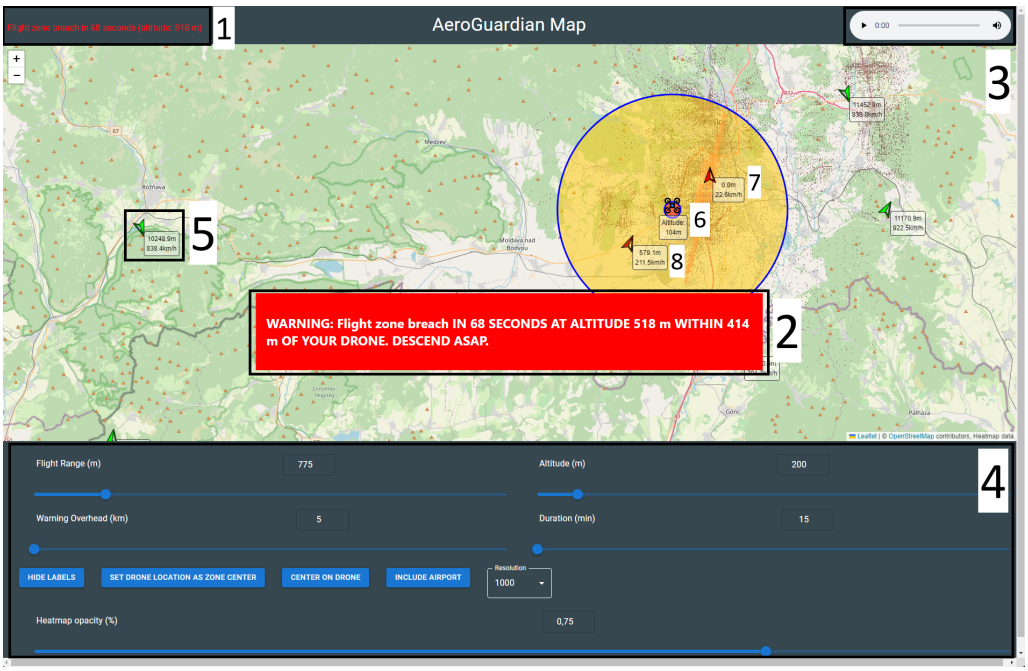
Obr. 3 – 3: Softvérová architektúra aplikácie

4 Použitie aplikácie

Vzhľad front-endu je znázornený na obrázku 4 – 1.

Front-end má tieto komponenty:

1. Hlavný varovný box – momentálne ukazuje varovanie o narušení letovej zóny za 68 sekúnd vo výške 518 metrov,
2. špeciálny varovný box – rozdiel výšky dronu a narušenia letovej zóny je menej ako 500 metrov (čo je v kóde nastavený bezpečný vertikálny odstup), tým pádom sa ukazuje špeciálne varovanie, ktoré varuje pilota, aby klesol,
3. koncový bod Icecastu na odpočúvanie leteckého rádia,
4. ovládací panel, obsahuje nastavenia mapy, aktuálnej letovej zóny, mapy výskytu letovej prevádzky a podobne,



Obr. 4 – 1: Vzhľad aplikácie AeroGuardian

5. príklad lietadla (v reálnom čase zachyteného), ktoré reprezentuje šípka, ktorá má farebný gradient podľa toho, ako je jeho výška nebezpečná oproti výšky nastavenej zóny; lietadlo má aj štítok, ktorý zobrazuje jeho výšku v metroch a rýchlosť v km/h,

6. ikonka dronu s jeho výškou a nastavená letová zóna, v okolí letiska je taktiež vykreslená mapa výskytu lietadiel pre aktuálnu výšku a určené nastavenia,

7. lietadlo, ktoré je na zemi vo varovnej zóne; varovanie o ňom sa nezobrazuje, pretože je aktuálna väčšia hrozba,

8. lietadlo smerujúce smerom k letovej zóne dronu letiace v malej výške.

4.1 Popis užívateľského rozhrania

Výstup zozbieraných informácií prebieha cez mapu, ktorá funguje v knižnici `Leaflet` s mapou OpenStreetMap. Na mapu sa dá klikať, čím sa dá určiť stred letovej zóny. Polomer letovej zóny, prídavný polomer varovnej zóny a výška zóny sa nastavuje

v ovládacom paneli. V ovládacom paneli sa taktiež nastavuje dĺžka predikcie v minútach, v ktorej back-end počíta budúce narušenia zóny.

Varovanie pilota dronu pred budúcimi hrozbami sa objavujú v komponentoch 1 (varovný box) a 2 (špeciálny varovný box) (obrázok 4 – 1). To najzávažnejšie narušenie zóny (berie sa do úvahy, ktorá zóna bola narušená (letová/varovná) a za aký čas bude narušená) sa zobrazí vo varovnom boxe mimo mapy. Varovanie pred narušením letovej zóny sa zobrazí červenou farbou, pred narušením varovnej zóny sa zobrazí žltou farbou. Ak je akékoľvek varovanie pod výškou dronu + bezpečný vertikálny odstup, v strede obrazovky sa zobrazí špeciálny varovný box, ktorý varuje pilota dronu, aby čo najskôr klesol na bezpečnú výšku.

5 Popis vstupných, výstupných a pracovných súborov

Hlavný súbor, s ktorým aplikácia pracuje, je `aircraft.json`. Tento súbor je generovaný službou `dump1090-fa` (viď 3.3.1). Okrem toho skripty na analýzu dát, ktoré sa nachádzajú v adresári `utilities`, využívajú databázu PostgreSQL. SQL na vytvorenie databázy sa nachádza v `utilities/postgre/aeroguardian-init.sql`. Parametre na pripojenie na databázu je potrebné upraviť v každom skripte, ktorý ju používa (hlavná aplikácia ju nepoužíva).

PostgreSQL sa využíva najmä na generáciu mapy výskytu letovej prevádzky, ktorá sa ďalej menia na dlaždice, ktoré sú zobraziteľné v knižnici `Leaflet`. Návod na generáciu máp výskytu prevádzky sa nachádza v kapitole 3.3.9.

Nastavenia front-endu pre spracovanie mapy výskytu prevádzky sa nachádzajú v `frontend-react/src/components/Heatmap.jsx`.

6 Obmedzenia aplikácie

Keďže aplikácia je závislá na zdroju dát, ktorý je pre nás ADS-B, väčšina obmedzení aplikácie sú kvôli obmedzeniam ADS-B. Vyhodnotenie výsledkov našej práce ukázalo, že schopnosť prijímať lietadlá v malej výške hyperbolicky klesá so vzdialenosťou. Priemerný čas varovania pred narušením zóny sú dve minúty. ADS-B je taktiež obmedzované okolitým reliéfom a zle funguje medzi kopcami. Pilot dronu musí na tieto obmedzenia myslieť a vždy lietať zodpovedne a bezpečne.

7 Chybové hlásenia

Pre zistenie chýb vo front-ende odporúčame otvoriť konzolu v prehliadači pri prezeraní front-endu. Back-end vypisuje chyby priamo do konzole, v ktorej je spúšťaný (v prípade Linuxovej služby sa dá skontrolovať stav služby, kde bude jeho konzola, pomocou `systemctl status`).

8 Príklad použitia

Vďaka klient-server architektúre, ak máme server dostupný cez internet a zdroj dát sa nachádza v blízkosti oblasti, v ktorej chceme lietať, môžeme používať aplikáciu s akýmkoľvek koncovým zariadením s pripojením na internet.

Aplikácia sa má používať najmä pri predletovej kontrole na skontrolovanie letovej prevádzky v okolí. Pokiaľ pilot dronu zistí, že je v okolí prevádzka, odporúča sa, aby počkal, kým prevádzka odletí. Nadväzujúc na to je ideálne, aby mal pilot dronu aplikáciu na inom zariadení, ako používa na lietanie, aby ho mohol sledovať a kontrolovať pre naskytnuté hrozby.

9 Popis demo-verzie

Aplikácia je vo svojej aktuálnej podobe funkčná a výrazne zvyšuje bezpečnosť integrácie dronov a prehľad pilota dronu o okolitej prevádzke. Ďalší vývoj by mohol smerovať na implementáciu iných spôsobov získavania dát o letovej prevádzke, predovšetkým leteckého rádia, ktoré by bolo možné spracovať algoritmami na potlačenie šumu a špeciálne trénovanými STT algoritmami. Mohol by sa aj vyvinúť univerzálny systém na prijímanie polohy dronu v reálnom čase. Okrem iného by ďalší vývoj mohol ešte viac prehĺbiť klient-server architektúru a umožniť prístup pilotom dronov bez toho, aby si museli inštalovať celý systém prijímania a hosťovania.

Zoznam obrázkov

Zoznam použitej literatúry

FlightAware, 2024. *PiAware Installation Instructions*. Dostupné na internete: `https://www.flightaware.com/adsb/piaware/install` (citované 23.5.2024).

Python Software Foundation, 2024. *Python Language Reference, version 3.9*. Dostupné na internete: `https://www.python.org` (citované 23.5.2024).

Node.js, 2024. *Node.js*. Dostupné na internete: `https://nodejs.org` (citované 23.5.2024).

GDAL, 2024. *GDAL: Geospatial Data Abstraction Library - Downloads*. Dostupné na internete: `https://gdal.org/download.html` (citované 23.5.2024).

Anaconda, 2024. *Anaconda Software Distribution, version 3*. Dostupné na internete: `https://www.anaconda.com/products/distribution` (citované 23.5.2024).

OpenStreetMap contributors, 2023. *OpenStreetMap*. Dostupné na internete: `https://www.openstreetmap.org` (citované 19.5.2024).

Shenzhen DJI Sciences and Technologies Ltd., 2024. *Mobile SDK for Android*. Dostupné na internete: `https://github.com/dji-sdk/Mobile-SDK-Android` (citované 23.5.2024).

Nisbet, A., 2020. *OpenTopoData dokumentácia, použitý dataset EU-DEM*. Dostupné na internete: `https://www.opentopodata.org/datasets/eudem/` (citované 19.5.2024).

Nisbet, A., 2020. *OpenTopoData*. Dostupné na internete: `https://github.com/ajnisbet/opentopodata` (citované 19.5.2024).