Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie

Zvyšovanie bezpečnosti integrácie dronov do leteckého priestoru

Bakalárska práca

AeroGuardian

Systémová príručka

Vedúci bakalárskej práce:

doc. Ing. Peter Papcun, PhD.

Autor:

Štefan Kando

Košice 2024

Obsah

1 Funkcia programu

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. 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 Analýza riešenia

V našej práci bol každý aspekt aplikácie dôkladne vybratý a každému boli zhrnuté pozitíva a negatíva.

2.1 Analýza spôsobov získavania dát

Spôsoby získavania dát sme v nasledovných kategóriach: cena (dostupnosť) pre bežného pilota dronu, akú majú dáta digitalizáciu (ako dobre sa konvertujú na digitálne dáta), aká je presnosť zisťovania polohy a výšky okolitej prevádzky, ako je táto presnosť ovplyvnená počtom účastníkov v okolí, a s akým predstihom / oneskorením vieme prijať dáta o danej prevádzke. pre každý spôsob získavania dát sme navrhli, ako by systém využívajúci tento spôsob získavania dát mohol vyzerať. Výsledok našej analýzy je v tabuľke 2 – 1. Dospeli sme k záveru, že ADS-B je najpraktickejšie a najlepšie pre naše potreby.

ADS-B však aj má svoje nevýhody. Vyhodnotenie výsledkov našej práce ukázalo, že schopnosť prijímať lietadlá v malej výške hyperbolicky klesá so vzdialenosťou.

Spôsob	**Cena**	**Digitalizácia**	**Presnosť**	**Vplyv počtu**	**Predstih**
Vizuálny kontakt	Vysoká	Ľahká	Zlá	Malý	Malý
Zvuk	Stredná	Náročná	Zlá	Veľký	Stredný
Letecké rádio	Nízka	Stredne náročná	Stredná	Stredný	Stredný
ADS-B	Nízka	Ľahká	Veľmi dobrá	Veľmi malý	Veľký

Tabuľka 2 – 1 Porovnanie rôznych spôsobov získavania dát o letovej prevádzke a ich relevancia k aplikácii podľa vybratých kritérií

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.

2.2 Analýza hardvérovej architektúry

Pre prijímanie ADS-B je potrebné použiť SDR (Software Defined Radio) schopné prijímať 1090 MHz. Najlepšie výsledky sme dosiahli s SDR schopným filtrácie a predspracovania ADS-B, konkrétne FlightAware Pro Stick Plus (FAPSP). na prijímanie ADS-B je taktiež potrebná anténa dizajnovaná na 1090 MHz s dobrým výhľadom na oblohu. ADS-B je možné predspracovať, avšak počas realizácie sme prišli na to, že predspracovanie signálu priamo vo FAPSP je dostatočné a ďalšie predspracovanie nie je potrebné.

Medzi základné funkcionality FAPSP patrí prijímanie ADS-B od lietadiel v okolí a schopnosť poskytovať tieto dáta serverom FlightAware, ako aj schopnosť k týmto dátam pristupovať cez internetové API z iných koncových bodov.

Túto funkcionalitu vykonáva softvér vyvinutý FlightAware nazývaný PiAware. PiAware je softvér vyvinutý pre mikropočítače Raspberry Pi (RPi). RPi je ideálne na serverové aplikácie, avšak jeho použitie v teréne je možné, ale nepraktické, vzhľa-

dom na nutnosť mať používateľské rozhranie (napr. dotykový displej), napájací zdroj (power banka) a pripojenie k internetu (prístupový bod z mobilného telefónu).

2.3 Analýza architektúry klient-server

Pre našu aplikáciu sme si vybrali klient-server architektúru, ktorá maximalizuje všestrannosť a použiteľnosť aplikácie. Klient-server architektúra znamená, že pilot dronu nemusí mať žiaden hardvér so sebou a stačí mu akékoľvek koncové zariadenie s pripojením na internet. v takomto prípade môže mať prijímač staticky nainštalovaný niekde v okolí, v ktorom lieta, pričom musí myslieť na limitácie ADS-B a mať prijímač dobre umiestnený. Prípadne môže prijímač úplne vynechať a použiť API, napríklad AeroAPI od FlightAware, avšak stále musí myslieť na to, odkiaľ dáta prichádzajú a aké sú ich limitácie. Klient-server architektúra zároveň nevylučuje prijímanie ADS-B v teréne. Služby našej aplikácie môžu byť hosťované aj lokálne na zariadení, ktoré ADS-B prijíma, a k týmto službám môže byť buď lokálne pristupované, alebo môžu byť posielané na hlavný server, odkiaľ môžu byť dostupné z iných koncových zariadení pripojených na internet.

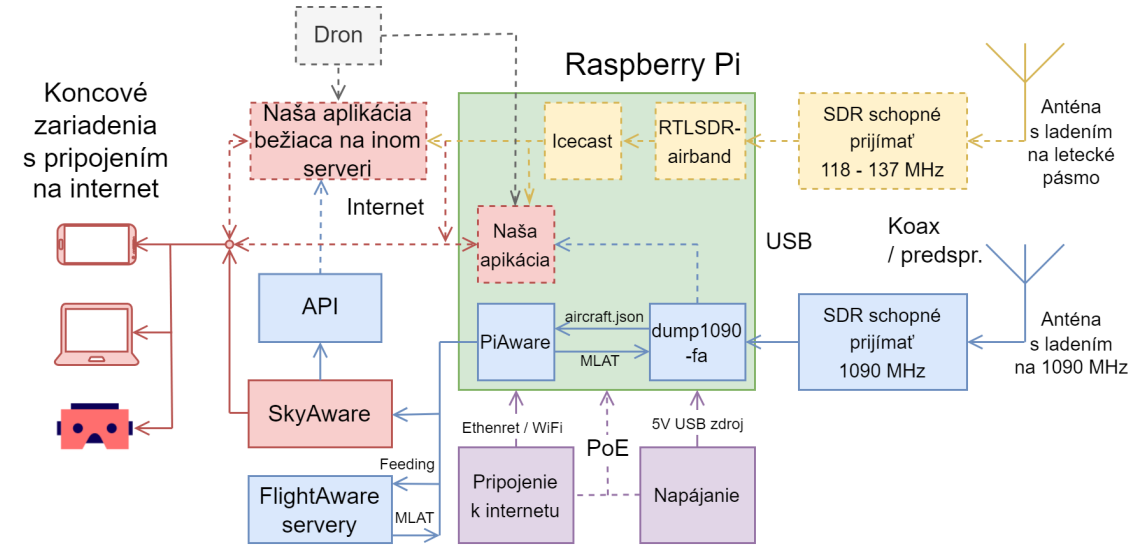
PiAware na vizualizáciu dát poskytuje lokálne hosťovanú stránku SkyAware, ktorá okrem iného obsahuje prípojné body na API.

3 Popis programu

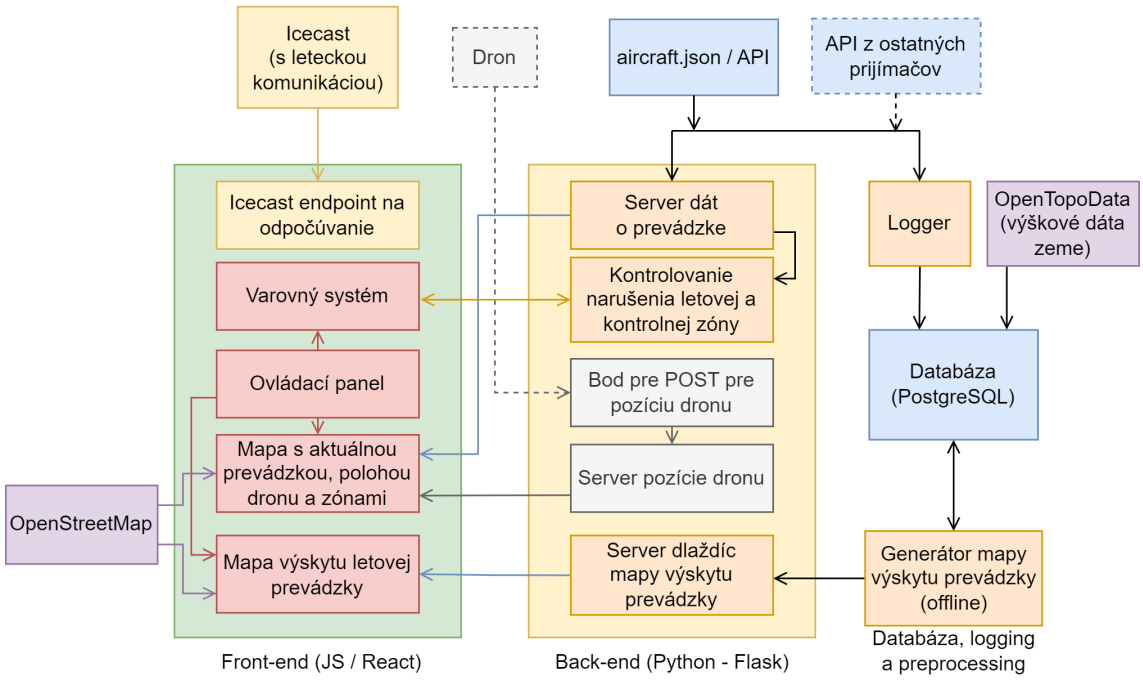
Na obrázku 3 – 1 je zobrazená bloková architektúra aplikácie. na obrázku 3 – 2 je zobrazená softvérová architektúra aplikácie. na obrázku 3 – 3 je bližšie vysvetlená komunikácia medzi komponentami pomocou sekvenčného diagramu.

3.1 Popis riešenia

Na obrázku 3 – 1 je zaznačená bloková architektúra riešenia, na ktorej sú znázornené všetky časti (bloky) aplikácie, od zberu dát až po ich prezentáciu pilotovi dronu

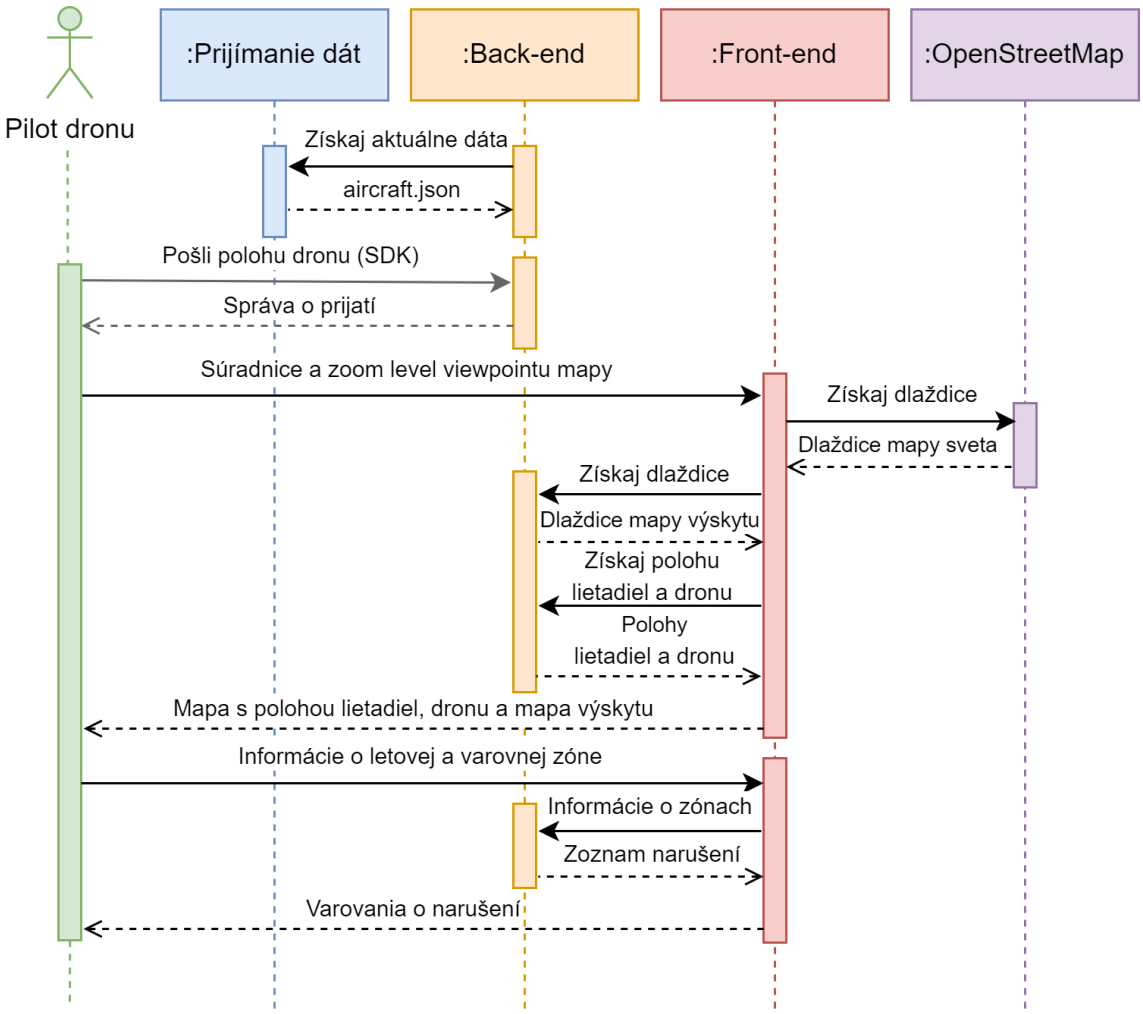


Obr. 3 – 1 Bloková architektúra prijímania rádiových signálov a našej aplikácie



Obr. 3 – 2 Softvérová architektúra aplikácie

cez koncové zariadenia. Časti dôležité pre hlavnú funkcionalitu sú zaznačené plnými čiarami, pričom nekritické časti sú zaznačené čiarkovane. Naša aplikácia je zaznačená čiarkovane, pretože použitie len samotného SkyAware, ktoré je jednoducho



Obr. 3 – 3 Sekvenčný diagram aplikácie

nainštalovateľné, môže výrazne zvýšiť bezpečnosť prevádzky dronu. Taktiež máme možnosť našu aplikáciu nainštalovať buď lokálne na rovnakom zariadení, ktoré ADS-B prijíma, alebo využiť iný server. Žltou farbou je znázornený zber hlasových dát z leteckého rádia, modrou farbou je znázornený zber dát z ADS-B a červenou farbou sú znázornené webové aplikácie.

Na obrázku 3 – 2 je zaznačená softvérová architektúra našej aplikácie (blok „Naša aplikácia“ v predošlom diagrame). Back-end je založený na webovom frameworku Flask v programovacom jazyku Python. Front-end je vytvorený v JavaScript (JS) knižnici React. na ukladanie a spracovanie uložených dát sa používa databáza PostgreSQL, ktorá však nie je potrebná pre beh hlavnej aplikácie.

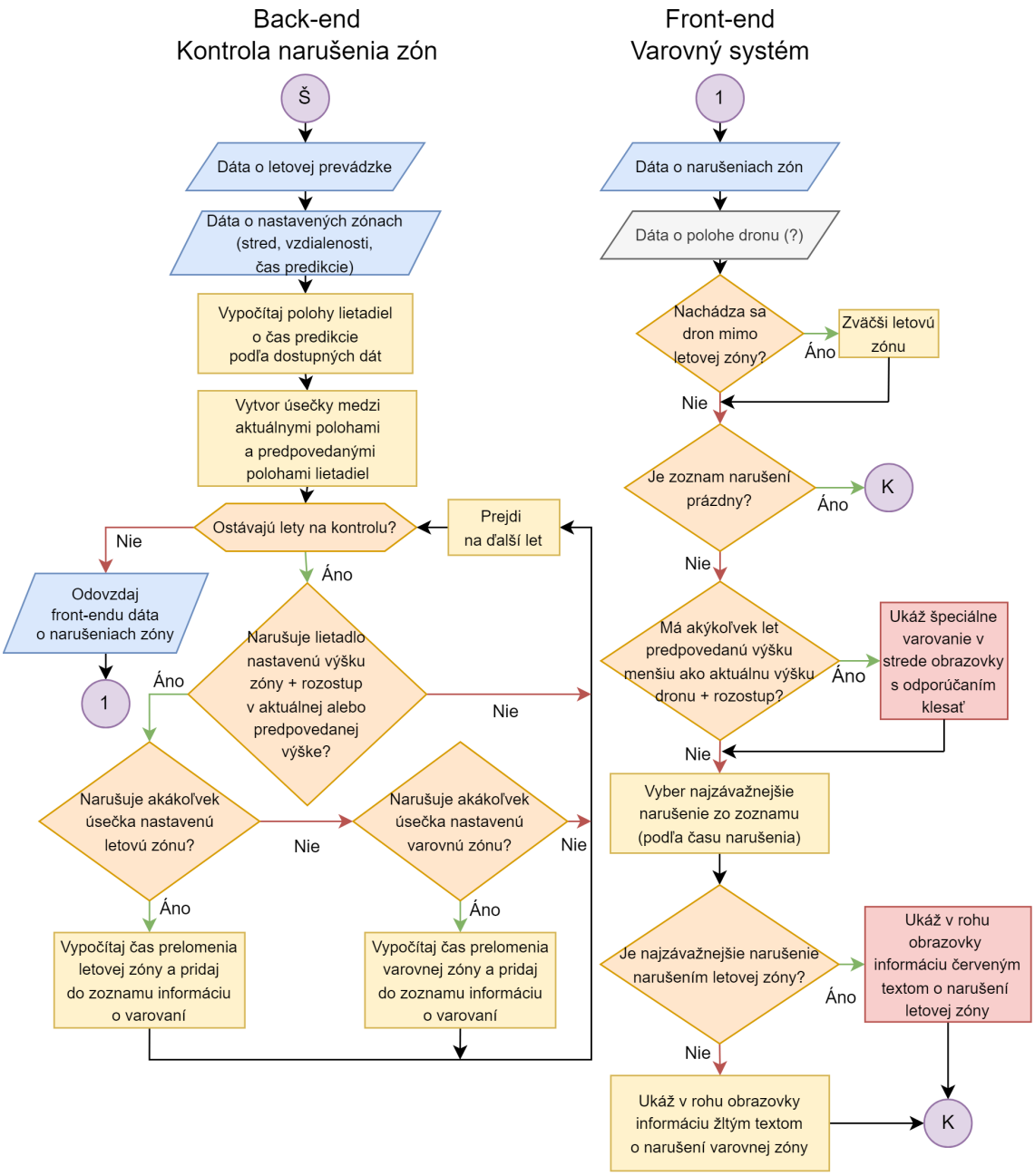
Na obrázku 3 – 3 je možné lepšie vidieť vzájomnú komunikáciu komponentov aplikácie. Jadrom front-endu je knižnica Leaflet, ktorá dokáže zobrazovať mapu užívateľovi vo forme dlaždíc podľa toho, kde sa práve používateľ pozerá a akú má hladinu priblíženia. Dlaždice mapy sveta aplikácia čerpá z open source mapy OpenStreetMap. Súčasťou riešenia sú aj skripty na analýzu nazbieraných dát v databáze, ktoré okrem iného vedia vytvoriť mapu najčastejšieho výskytu letovej prevádzky. Tieto mapy sú taktiež prekonvertované do dlaždíc, ktoré vie Leaflet zobraziť.

3.2 Popis algoritmov

Najdôležitejšou časťou aplikácie je varovný systém, ktorý dokáže pilota dronu varovať pred narušením letovej alebo varovnej zóny lietadlom ešte predtým, než do nej vojde.

Hlavná časť varovného systému je v koncovom bode back-endu, `/check_breach`. Tento koncový bod cez POST prijíma aktuálne nastavenie zón a vracia zoznam zistených narušení. Medzi nastaveniami zón sa nachádza aj čas predikcie, v ktorom si užívateľ želá predpovedať vývin situácie v letovej prevádzke. Funkcionalita `/check_breach` spočíva vo výpočte, kde sa budú lietadlá za určený čas predikcie nachádzať, horizontálne aj vertikálne. Výpočet predpovedanej horizontálnej polohy funkcia robí pomocou dát, ktoré má k dispozícii, ako je aktuálna poloha lietadla, jeho rýchlosť a smer. Potom vypočíta aj predpovedanú výšku lietadla za čas na základe jeho aktuálnej výšky a vertikálnej rýchlosti. Pokiaľ je akákoľvek aktuálna alebo vypočítaná výška menšia ako výšková hranica zóny + bezpečný vertikálny odstup, program skontroluje, či úsečka nízko letiaceho lietadla narušuje kružnicu niektorej zo zón, prv letovej, potom varovnej. Ak áno, tak aj vypočíta, že za ako dlho narušenie zóny nastane. Všetky narušenia ukladá do zoznamu, a tento zoznam posiela naspäť front-endu.

Varovný systém vo front-ende sa skladá z dvoch prvkov: varovného boxu a špeciálneho varovného boxu. Ak front-end cez `/check_breach` dostane zoznam, ktorý



Obr. 3 – 4 Vývojový diagram plánovacej a varovnej funkcionality aplikácie

nie je prázdny, 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. Celkový vývojový diagram je znázorený na obrázku 3 – 4.

3.3 Popis získavania dát

Dáta o letovej prevádzke sa získavajú z ADS-B prijímaného cez SDR, ktoré spracováva služba `dump1090-fa`, ktorá je súčasťou PiAware (viď 2.2). PiAware funguje na RPi na operačnom systéme Raspbian.

Univerzálny spôsob získavania polohy dronu v reálnom čase nebol vyvinutý. Pre potreby demonštrácie s naším dronom (DJI Mini 3 Pro) sme využili úpravu ukážkového kódu DJI Mobile SDK. Podrobnosti implementácie s DJI Mobile SDK sú rozpísané v užívateľskej príručke.

3.4 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.

Špecifické komponenty a výzor užívateľského rozhrania sú uvedené v užívateľskej príručke.

3.5 Popis back-endu

Back-end má nasledovné koncové body:

- `/heatmap` – pre hosťovanie dlaždíc pre mapu výskytu prevádzky (ktorú dlaždicu je potrebné zobraziť ovláda `Leaflet`),
- `/drone_location` – s možnosťou metódy `POST` na zasielanie aktuálnej polohy dronu na server s validáciou dát,

- `/data` – poskytujúci dáta z prijímača vo formáte `aircraft.json`,
- `/drone_data` – poskytujúci aktuálnu polohu dronu,
- `/check_breach` – s možnosťou metódy `POST`, ktorý prijíma informácie o nastavení letovej a varovnej zóny a vracia zoznam s vypočítanými narušeniami zóny.

3.6 Popis vstupných a výstupných súborov

- Hlavný súbor, s ktorým aplikácia pracuje, je `aircraft.json`, generovaný službou `dump1090-fa`.
- Skripty na analýzu dát v adresári `utilities` využívajú databázu PostgreSQL.
- SQL príkazy na vytvorenie databázy sú umiestnené v súbore `utilities/postgre/aeroguardian-init.sql`.
- Parametre na pripojenie k databáze je potrebné upraviť v každom skripte, ktorý ju používa. Hlavná aplikácia databázu nepoužíva.
- PostgreSQL je využívaná najmä na generáciu mapy výskytu letovej prevádzky, ktoré sú potom prevedené na dlaždice zobraziteľné v knižnici `Leaflet`.
- Hlavný skript na generovanie máp (`generate_heatmaps_main.py`) je umiestnený v `utilities/heatmap/`.
- Nastavenia front-endu pre spracovanie mapy výskytu letovej prevádzky sa nachádzajú v `frontend-react/src/components/Heatmap.jsx`.

4 Preklad programu

Python a JS sú programovacie jazyky, ktoré sa neprekladajú do strojového kódu, ale pri svojom behu pracujú s kódom, v ktorom sú napísané. Návody na spustenie

všetkých skriptov, či už na vývoj alebo deployment, sú podrobne opísané v užívateľskej príručke.

4.1 Zoznam zdrojových textov

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).

Všetky adresáre, ktoré využívajú Python, majú priložený súbor `requirements.txt` s požadovanými knižnicami.

Všetky súbory, na ktorých sa pracovalo, majú hlavičku a sú okomentované v slovenčine.

Adresár `backend-python` obsahuje `main.py` s hlavným skriptom pre spúšťanie back-endu a `check_breach.py`, ktorý obsahuje funkcionalitu varovného systému v back-ende.

Adresár `frontend-react` obsahuje kód front-endu vytvorený v JS knižnici React. Hlavný kód používaných komponentov sa nachádza v `src/components/`.

Front-end má zdrojový kód komponentov rozdelený nasledovne:

- `AircraftMarkers.jsx` – komponent na pridávanie získaných dát o lietadlách do mapy,
- `BreachStatusComponent.jsx` – komponent na zobrazenie najkritickejšieho varovania o narušení zóny pre pilota dronu,

- `ControlPanelSliderComponent.jsx` – komponent na nastavovanie číselných parametrov pomocou `Slider`-u a `TextField`-u,
- `Drone.jsx` – komponent na zobrazenie dronu na mape,
- `Heatmap.jsx` – komponent na spracovanie mapy výskytu letovej prevádzky,
- `Home.jsx` – hlavný komponent obsahujúci hlavnú mapu, ovládací panel, koncový bod Icecastu a varovný systém,
- `IcecastPlayer.jsx` – koncový bod Icecastu na odpočúvanie leteckého rádia,
- `MapComponent.jsx` – kontajner pre hlavnú mapu aplikácie, zobrazujúcu zachytené lietadlá a polohu dronu,
- `MapControlPanel.jsx` – kontajner pre ovládací panel mapy hlavnej stránky,
- `ProcessData.jsx` – funkcie na spracovávanie a komunikáciu s back-endom,
- `StaticSettings.jsx` – statické nastavenia upraviteľné v kóde,
- `Utils.jsx` – ostatné pomocné funkcie front-endu,
- `Zone.jsx` – komponent fungovania letovej a varovnej zóny.

Adresár `utilities` obsahuje rôzne skripty pre predspracovanie dát:

- `heatmap` – obsahuje skripty na generovanie máp výskytu prevádzky (hlavný skript je `generate_heatmaps_main.py`),
- `logger` – obsahuje `logger.py` na kontinuálne ukladanie zachytených dát do PostgreSQL, a `json_to_postgre.py` na presun zálohovaných `aircraft.json` súborov do PostgreSQL,
- `OpenTopoData-postgre/fill_grid_height.py` – na plnenie výškových dát do PostgreSQL z OpenTopoData,

- `postgre/aeroguardian-init.sql` – SQL na inicializáciu databázy `aeroguardian` v PostgreSQL.

4.1.1 Back-end: `main.py`

Tento skript je hlavný skript back-endu pre aplikáciu. Využíva Flask framework na vytvorenie webového servera, ktorý komunikuje s front-endom a spracúva dáta o polohe dronov a letovej prevádzke.

- Inicializácia Flask aplikácie – Vytvorí sa inštancia aplikácie Flask.
- Povolenie CORS – Nastavuje sa, aby aplikácia mohla prijímať požiadavky z rôznych zdrojov.
- Nastavenie debug režimu – Umožňuje zapnúť alebo vypnúť ladací režim podľa potreby.

Popis vstupných premenných koncových bodov

- `/heatmap/{included}/res_{resolution}/{height}/{z}/{x}/{y}.png` – Slúži na poskytovanie dlaždíc pre heatmapy. Vstupné parametre sú názov heatmapy (`airport_included` alebo `airport_excluded`), rozlíšenie, výška, a koordináty dlaždice.
- `/drone_location` – Endpoint na prijatie a validáciu dát o aktuálnej polohe dronu. Prijíma JSON s kľúčmi `latitude`, `longitude` a `altitude` (v m nad zemou).
- `/check_breach` – Kontroluje narušenia určenej zóny podľa aktuálnych dát. Prijíma JSON s informáciami o zónach a nastaveniach – kľúče `zoneData` a `settings`, `zoneData` má kľúče `lat` a `lng`, `settings` má kľúče `flightRange`, `warningOverhead`, `duration` a `altitude`. Používa funkciu `check_aircraft_zone_violations` z modulu `check_breach.py`.

Popis hlavných premenných

- `latest_drone_location` – Ukladá najnovšie dáta o polohe dronu.
- `aircraft_data` – Obsahuje aktuálne informácie o letovej prevádzke.
- `debug` – Premenná určujúca, či je aplikácia v debug režime.

Funkcie

- `validate_data` – Validuje prijaté dáta o polohe dronu. Kontroluje, či sú prítomné všetky potrebné polia (zemepisná šírka, dĺžka, nadmorská výška) a či sú správneho dátového typu.
- `serve_tile` – Poskytuje dlaždice pre heatmapy. Vstupné parametre sú názov heatmapy, rozlíšenie, výška a koordináty dlaždice.
- `receive_drone_data` – Spracúva dáta prijaté z dronu. Kontroluje, či sú dáta vo formáte JSON a validuje ich pomocou `validate_data`.
- `data` – Získava dáta o letovej prevádzke z externého zdroja. Používa HTTP požiadavku na získanie dát.
- `drone_data` – Poskytuje údaje o najnovšej polohe dronu.
- `check_breach` – Kontroluje narušenia zóny na základe prijatých dát. Používa funkciu `check_aircraft_zone_violations` z modulu `check_breach.py` na vyhodnotenie potenciálnych narušení zóny.

4.1.2 Back-end: `check_breach.py`

Skript obsahuje funkcie pre kontrolu možného narušenia určených zón lietadlami. Využíva predikciu pohybu lietadiel na základe ich súčasnej polohy a plánovaného kurzu na určenie, či dôjde k narušeniu letových zón.

Popis funkcií

- `check_aircraft_zone_violations` – Hlavná funkcia skriptu, ktorá zisťuje možné narušenia letových zón.
  - Vstupné parametre:
    - `zone_info` – Informácie o zóne, obsahuje zemepisné šírky a dĺžky.
    - `settings` – Nastavenia zóny, obsahujúce údaje o rozsahu letu a varovnej výške.
    - `aircraft_data` – Dáta o letovej prevádzke, získané vo formáte JSON.
  - Výstup: Zoznam narušení zóny, každé narušenie obsahuje kód lietadla, typ narušenia, predpovedanú výšku a čas do narušenia v sekundách.
- `latlon_to_xy` – Pomocná funkcia na konverziu zemepisných súradníc na kartézske súradnice.
  - Vstupné parametre:
    - `lat`, `lon` – Zemepisné súradnice bodu, ktorý sa konvertuje.
    - `center_lat`, `center_lng` – Zemepisné súradnice stredového bodu pre konverziu.
  - Výstup: Kartézske súradnice (`x`, `y`) zadaného bodu.
- `predict_position` – Funkcia na predikciu budúcej polohy lietadla na základe jeho súčasnej polohy, rýchlosti a kurzu.
  - Vstupné parametre:
    - `lat`, `lon` – Súčasné zemepisné súradnice lietadla.
    - `speed` – Rýchlosť lietadla v uzloch.
    - `track` – Smer letu lietadla v stupňoch.
    - `duration` – Časové obdobie, po ktorom sa má poloha predpovedať.

- Výstup: Predpovedané zemepisné súradnice lietadla.

- `check_intersection_and_time` – Funkcia na kontrolu, či a kedy dôjde k narušeniu zóny na základe trajektórie letu.
  - Vstupné parametre:
    - `x0`, `y0` – Počiatočné kartézske súradnice lietadla.
    - `x1`, `y1` – Predpovedané kartézske súradnice lietadla.
    - `cx`, `cy` – Kartézske súradnice stredového bodu zóny.
    - `radius` – Polomer zóny v m.
  - Výstup: Informácia o tom, či a kedy dôjde k narušeniu zóny.

Výstup hlavnej funkcie

Hlavná funkcia `check_aircraft_zone_violations` vracia zoznam narušení zóny, kde každé narušenie obsahuje podrobné informácie o lietadle (kód lietadla), typ narušenia, predpovedanú výšku, na ktorej dôjde k narušeniu, a čas do narušenia v sekundách.

4.1.3 Front-end: `Home.jsx`

Súbor `Home.jsx` obsahuje hlavný komponent aplikácie, ktorý zastrešuje viaceré podkomponenty a funkcionalitu. Tento komponent zahrňuje hlavnú mapu, ovládací panel, koncový bod Icecastu pre streamovanie audio dát a varovný systém pre drony. Komponent tiež spravuje stavy dát lietadiel, dronov a zón narušenia.

Integrácia s ostatnými komponentami

Komponent `Home.jsx` integrálne spolupracuje s viacerými podkomponentami, ktoré zahŕňajú:

- `MapComponent` – Zobrazuje hlavnú mapu a súvisiace geografické údaje.

- `MapControlPanel` – Umožňuje užívateľovi interakciu s mapovými nastaveniami.
- `Heatmap` – Poskytuje vizualizáciu dát o letovej prevádzke.
- `Zone` – Zobrazuje zóny narušení a spravuje ich nastavenia.
- `BreachStatusComponent` – Informuje užívateľa o aktuálnych narušeniach.
- `IcecastPlayer` – Streamuje audio z leteckých komunikácií.

Spracovanie dát

Komponent `Home.jsx` zahŕňa funkcie na načítanie, aktualizáciu a spracovanie dát o lietadlách a dronoch, ako aj funkcie na kontrolu a reakciu na možné narušenia zón. Všetky funkcie na získavanie dát pochádzajú z komponentu `ProcessData.jsx`, viď kapitola 4.1.4.

Dynamické aktualizácie

Komponent `Home.jsx` používa intervaly a hooky Reactu na pravidelné aktualizácie dát a kontrolu narušení, čím zabezpečuje, že užívateľ má vždy k dispozícii najaktuálnejšie informácie.

Return

Komponent `Home.jsx` vracia HTML obsah cez funkciu return v JSX formáte. Return blok obsahuje hlavný kontajner div, ktorý zahrňa všetky podkomponenty a UI sekcie. v rámci tohto kontajnera sa nachádzajú ďalšie podkomponenty ako `MapComponent`, `MapControlPanel`, `Heatmap`, `Zone` a `BreachStatusComponent`, ktoré sú zodpovedné za rôzne funkcionalitné aspekty aplikácie. Každý z týchto podkomponentov je spracovaná ako samostatný React komponent a je integrovaná do hlavného komponenta prostredníctvom ich JSX tagov. Taktiež sa tu nachádza `WarningAlert`, ktorý sa zobrazuje, ak sú splnené určité podmienky varovania.

4.1.4 Front-end: `ProcessData.jsx`

Súbor `ProcessData.jsx` obsahuje kľúčové funkcie na spracovávanie a komunikáciu s back-endom aplikácie. Tento modul zabezpečuje načítavanie a spracovanie dát o lietadlách a dronoch, ako aj kontrolu možných narušení letových zón.

Importované moduly a konfigurácie

Súbor importuje konfiguračné premenné ako `BackendIP` a `VerticalOverhead` z modulu `StaticSettings.jsx`, ktoré sú využívané v rámci požiadaviek na back-end server a pri výpočtoch bezpečných odstupov.

Funkcie na získanie dát

- `getAircraftData` – Asynchrónna funkcia na získanie aktuálnych dát o lietadlách z back-endu. v prípade neúspechu alebo chyby v odpovedi servera táto funkcia vracia prázdne pole.
- `getDroneData` – Asynchrónna funkcia na získanie dát o aktuálnej polohe a nadmorskej výške dronu. Kontroluje formát a obsah odpovede, a v prípade potreby vracia `null`.

Funkcie na kontrolu narušení

- `checkForBreaches` – Asynchrónna funkcia, ktorá odosiela požiadavky na server s údajmi o aktuálnych nastaveniach zóny a prijíma informácie o potenciálnych narušeniach. v prípade chyby alebo neúspechu vracia prázdne pole.
- `checkDroneProximityToBreach` – Funkcia na určenie blízkosti dronu k detekovaným narušeniam zóny na základe vertikálneho odstupu. Vracia varovania pre prípady, keď dron prekračuje bezpečnostné limity.

4.1.5 Front-end: `MapComponent.jsx`

Súbor `MapComponent.jsx` obsahuje React komponent, ktorý slúži ako kontajner pre hlavnú mapu aplikácie. Tento komponent zobrazuje mapu s aktuálnymi pozíciami lietadiel a dronu, čím poskytuje užívateľom vizuálne reprezentáciu letovej prevádzky a ich vzťah k polohe dronu.

Importy a závislosti

Komponent importuje potrebné moduly a funkcie pre jeho funkčnosť:

- Knižnica `Leaflet` pre mapové funkcie a jej CSS štýly pre správne zobrazenie.
- Funkcie `addAircraftMarkers` a `updateDroneMarker` pre pridávanie a aktualizáciu markerov na mape.
- Predvolené nastavenia mapy z `StaticSettings.jsx`.

Inicializácia a správa mapy

Komponent používa React hook `useEffect` na inicializáciu mapy ihneď po pripojení komponentu. v rámci tohto hooku sa vytvára inštancia mapy s predvolenými nastaveniami z `DefaultMapSettings` a pridáva sa vrstva mapy pomocou `L.tileLayer` od OpenStreetMap.

Dynamické pridávanie a aktualizácia markerov

- Marker pre lietadlá je pridávaný a aktualizovaný na základe dát poskytnutých v `aircraftData`, kde každé lietadlo na mape je reprezentované markerom, ktorý je prispôsobený podľa aktuálneho nastavenia zobrazenia.
- Marker pre dron je aktualizovaný vždy, keď sa zmenia dáta o polohe dronu. Táto aktualizácia zahŕňa posunutie markera na novú pozíciu dronu na mape.

Reakcia na zmeny nastavení

Komponent reaguje na zmeny v nastaveniach ako `centerOnDrone`, čo znamená, že ak je táto možnosť zapnutá, mapa sa automaticky posunie tak, aby bola pozícia dronu v strede mapy.

Return

Vracaný kontajner je určený na zobrazenie mapy a má nastavenú štýlovú definíciu s výškou 60% a šírkou 100% z celkovej dostupnej plochy. Tento div element slúži ako plátno pre mapu vytvorenú pomocou knižnice Leaflet. v rámci tohto kontajnera sa dynamicky vkladajú a aktualizujú markery pre lietadlá a dron podľa aktuálnych dát, ktoré sú vkladané cez referenciu mapy uloženú v hlavnom komponente.

4.1.6 Front-end: `AircraftMarkers.jsx`

Súbor `AircraftMarkers.jsx` obsahuje funkcie a komponenty pre pridávanie markerov lietadiel na mapu v aplikácii. Tento komponent je zodpovedný za vizualizáciu polôh lietadiel na mape použitím knižnice Leaflet.

Importované zdroje

Komponent importuje knižnicu Leaflet pre mapové funkcie, štýly Leaflet pre správne zobrazenie mapy a špeciálny plugin `leaflet.rotatedMarker` na umožnenie rotácie markerov na mape podľa smeru letu lietadla.

Funkcie na tvorbu markerov

- `createLabelIcon` – Tvorba štítku s textom pre zobrazenie nadmorskej výšky a rýchlosti lietadla.
- `getArrowIcon` – Vytvorenie ikony šípky, ktorá indikuje smer letu lietadla, s možnosťou farebnej diferenciácie na základe výšky.

- `createAircraftMarker` – Vytvorenie dvoch typov markerov na mape: šípka ukazujúca smer a štítok s informáciami. Marker sa pridáva do mapy cez Leaflet a je spojený s konkrétnymi dátami lietadla.

Dynamická správa markerov

Komponent dynamicky spravuje markery na mape podľa aktuálnych dát získaných z komponentu nadradeného. Markery sú aktualizované alebo odstraňované na základe zmien v dátach lietadiel, čo zabezpečuje ich aktuálnosť:

- `addAircraftMarkers` – Pridáva markery na mapu pre každé lietadlo v dátovom zozname. Funkcia najprv odstráni existujúce markery a potom vytvorí nové na základe aktuálnych dát.
- `updateLabelVisibility` – Aktualizuje viditeľnosť štítkov na markerov podľa používateľských nastavení, čo umožňuje zobraziť alebo skryť detailné informácie.

Farba markeru a jej význam

Farba šípky markeru sa mení v závislosti od výšky lietadla, čo poskytuje rýchlu vizuálnu spätnú väzbu o výške lietadiel vzhľadom na predvolený prahový limit.

4.1.7 Front-end: `Drone.jsx`

Súbor `Drone.jsx` obsahuje funkcie na vizualizáciu dronu na mape vo webovej aplikácii, zvlášť zobrazuje aktuálnu polohu a nadmorskú výšku dronu na mape.

Importované zdroje

Komponent využíva knižnicu Leaflet pre mapové funkcie a jej štýly, čo umožňuje vytvárať a spravovať mapové markery efektívne a štýlovo.

Funkcie na tvorbu a správu markeru

- `getDroneIcon` – Vytvára ikonu dronu pomocou SVG grafiky, ktorá slúži ako marker na mape. Táto ikona je štandardne nastavená na veľkosť 32x32 pixelov a je centrovaná na aktuálnu pozíciu dronu.
- `createDroneLabelIcon` – Tvorí textový štítok, ktorý sa zobrazuje nad markerom dronu a uvádza jeho nadmorskú výšku v m.
- `updateDroneMarker` – Zodpovedá za aktualizáciu pozície markeru dronu na mape a jeho štítku podľa posledných dostupných dát. Táto funkcia zaisťuje, že vizuálna reprezentácia polohy dronu je vždy aktuálna.

Integrácia s Leaflet

Marker dronu a jeho štítok sú integrované do mapy pomocou API knižnice Leaflet. Marker je dynamicky aktualizovaný pri každej zmene polohy alebo nadmorskej výšky dronu, čo zaisťuje vysokú úroveň presnosti a aktualizáciu v reálnom čase.

Dynamická správa markeru

- Marker dronu je vytvorený pri prvotnom spustení a následne aktualizovaný. Ak sú k dispozícii nové dáta, marker je posunutý na novú pozíciu a štítok je aktualizovaný o najnovšiu výšku.
- v prípade, že marker ešte neexistuje, je vytvorený nový marker a štítok s príslušnými údajmi, a oba sú pridané na mapu.

4.1.8 Front-end: `IcecastPlayer.jsx`

Súbor `IcecastPlayer.jsx` obsahuje komponent, ktorý funguje ako koncový bod pre odpočúvanie leteckého rádia prostredníctvom Icecast streamovacieho servera. Tento komponent je zodpovedný za poskytnutie audio prehrávača v aplikácii, ktorý umožňuje užívateľom počúvať živé letecké komunikácie.

Importované konfigurácie

Komponent importuje konfiguračnú premennú `IcecastIP` z modulu `StaticSettings.jsx`, ktorá obsahuje URL adresu Icecast streamu. Táto URL je kľúčová pre načítanie a správne fungovanie audio streamu.

Štruktúra komponentu

`IcecastPlayer` je funkčný komponent v Reacte, ktorý vracia kód pre audio prehrávač:

- **Audio kontajner** – HTML element `div`, ktorý má nastavenú absolútnu pozíciu v pravom hornom rohu aplikácie. Tento kontajner obsahuje audio prehrávač, ktorý umožňuje užívateľom ovládať prehrávanie streamu.
- **HTML audio element** – Základný HTML element pre prehrávanie zvuku, ktorý je nakonfigurovaný tak, aby automaticky prehrával zvuk a mal zapnuté ovládacie prvky pre užívateľa.
- **Source element** – Špecifikuje zdroj audio dát pre prehrávač, kde `src` atribút je nastavený na `IcecastIP` a `type` atribút označuje formát zvukového súboru ako `audio/mpeg`.

4.1.9 Front-end: `MapControlPanel.jsx`

Súbor `MapControlPanel.jsx` obsahuje React komponent, ktorý slúži ako ovládací panel pre hlavnú mapu aplikácie. Tento panel umožňuje užívateľom meniť nastavenia mapy vrátane zobrazenia, výšky a rozlišenia mapy.

Importované zdroje a komponenty

Komponent využíva rôzne UI komponenty z knižnice Material-UI, ako sú `Box`, `Grid`, `Button`, `FormControl`, `InputLabel`, `Select` a `MenuItem` pre štruktúrovanie a štýlovanie svojho rozhrania. Nastavenia tém a štýlov sú importované z `ControlPanelTheme` a `StaticSettings`.

Nastaviteľné parametre cez ovládací panel

`MapControlPanel` poskytuje užívateľom rozhranie pre nastavenie rôznych parametrov mapy, čo zahŕňa:

- **Flight Range** – Umožňuje nastavenie polomeru letovej zóny, ktorý je zobrazený na mape.
- **Warning Overhead** – Nastavuje dodatočný varovný obvod okolo zóny, ktorý je prídavným varovným prvkom k letovej zóne.
- **Altitude** – Umožňuje nastaviť výšku, na ktorej dron operuje, čo môže ovplyvniť zobrazenie a interakcie na mape.
- **Duration** – Definuje dĺžku času, počas ktorého sú relevantné letecké údaje a predpovede zobrazované a aktualizované.
- **Resolution** – Umožňuje užívateľom zmeniť rozlíšenie dát zobrazených na mape, čo môže ovplyvniť detailnosť a výkon zobrazenia.
- **Heatmap Opacity** – Nastavenie priehľadnosti tepelnej mapy letovej prevádzky, čo umožňuje užívateľovi upraviť viditeľnosť mapy podľa potreby.
- **Show Labels** – Prepínač pre zobrazenie alebo skrytie textových štítkov lietadiel a dronu na mape.
- **Exclude Airport** – Možnosť vybrať, či sa v zobrazení mapy zahrnie alebo vylúči oblasť okolo letiska.
- **Center on Drone** a **Update Zone Center** – Tieto možnosti umožňujú užívateľom centrovanie mapy priamo na dron alebo nastavenie aktuálnej polohy dronu ako nového centra zóny.

Tieto možnosti poskytujú užívateľom komplexné ovládanie nad tým, ako sú letecké dáta zobrazované a spravované na mape, čo umožňuje prispôsobenie mapy na mieru potrebám a preferenciám užívateľa.

Funkcie ovládania

- `handleChange` – Spracováva zmeny hodnôt nastavení prostredníctvom posuvníkov `ControlPanelSliderComponent`.
- `handleResolutionChange` – Umožňuje zmenu rozlíšenia mapy.
- `toggleLabels` – Prepína zobrazenie štítkov na mape.
- `toggleExcludeAirport` – Umožňuje vylúčiť alebo zahrnúť letisko na mape.
- `setZoneToDrone` a `centerOnDrone` – Poskytujú možnosti pre centrování mapy na aktuálnu polohu dronu a nastavenie tejto polohy ako centra zóny.

Return

Komponent vracia komplexne štruktúrované užívateľské rozhranie s viacerými prvkami, ktoré umožňujú užívateľovi intuitívne ovládanie zobrazenia a funkcií mapy.

4.1.10 Front-end: `ControlPanelSliderComponent.jsx`

Súbor `ControlPanelSliderComponent.jsx` definuje komponent pre React, ktorý umožňuje užívateľom nastavovať číselné parametre prostredníctvom posuvníka a textového vstupného poľa. Tento komponent je široko využívaný v rámci ovládacieho panela mapy na dynamickú úpravu rôznych nastavení mapy.

Importované zdroje

Komponent využíva UI prvky z knižnice Material-UI, ako sú `Typography`, `TextField`, a `Slider` pre zobrazenie a interakciu. Štýlovanie a tému poskytuje komponent `Box` z Material-UI.

Funkcionálne vlastnosti

Komponent kombinuje `Slider` a `TextField` na poskytovanie dvojitého mechanizmu pre užívateľský vstup, kde užívatelia môžu buď zadať hodnotu priamo do textového poľa alebo posunúť posuvník na požadovanú hodnotu.

- `handleSliderChange` – Spracúva zmeny na posuvníku a aktualizuje pridružený stav.
- `handleInputChange` – Umožňuje užívateľom zadávať hodnoty priamo a spracúva ich vstup na zabezpečenie, že hodnoty sú v rámci definovaných limitov.
- `handleBlur` – Zabezpečuje, že po opustení textového poľa sú hodnoty korigované tak, aby zapadali do povoleného rozsahu.

Nastavenia a parametre

Pre každý posuvník môžu byť nastavené špecifické vlastnosti ako minimálna a maximálna hodnota a krok posuvníka. Tieto nastavenia sú získavané z centrálneho konfiguračného súboru `StaticSettings.jsx`, ktorý udržiava konzistenciu a ľahkú údržbu hodnôt pre rôzne použitia komponentu v aplikácii.

Return

`ControlPanelSliderComponent` vracia kód, ktorý obsahuje skombinované grafické prvky pre zjednodušené a intuitívne ovládanie nastavení. Komponent je navrhnutý tak, aby bol ľahko integrovateľný do väčších formulárov alebo ovládacích panelov.

4.1.11 Front-end: `Zone.jsx`

Súbor `Zone.jsx` obsahuje React komponent, ktorý zodpovedá za správu a zobrazenie letovej a varovnej zóny na mape v aplikácii. Tento komponent je kľúčový pre zobrazovanie dynamických bezpečnostných zón v reálnom čase.

Importované zdroje

Komponent využíva knižnicu Leaflet pre manipuláciu s geografickými objektmi na mape a React hooky `useState` a `useEffect` pre správu stavov a bočných efektov v komponente.

Funkcie a správa zón

Komponent definuje niekoľko kľúčových funkcií:

- `clearZones` – Odstraňuje existujúce kruhy z mapy, čím zabezpečuje, že zóny sú vždy aktuálne a nekumulujú sa pri opakovanom nastavovaní.
- `setZoneCenter` – Nastavuje stred zón na základe užívateľských interakcií na mape, čím umožňuje flexibilné umiestnenie zón podľa potreby.
- `onClick` – Funkcia viazaná na udalosť kliknutia na mape, ktorá volá `setZoneCenter` pre dynamické umiestnenie zón.

Reakcia na zmeny a dynamické aktualizácie

- pri každej zmene referencie mapy alebo dát zóny sú zóny znovu nastavené pomocou efektov z `useEffect`, ktoré reagujú na zmeny stavov alebo props.
- Ďalší `useEffect` sleduje zmeny v nastaveniach, ako sú polomer letovej zóny a prídavného varovného obvodu, a aktualizuje ich na mape.

Vizualizácia zón

Zóny sú vizualizované pomocou kruhových objektov vytvorených v Leaflet:

- `flightZoneCircle` – Reprezentuje letovú zónu s červenou farbou a modrým obrysom.
- `warningZoneCircle` – Reprezentuje varovnú zónu, ktorá je vizuálne rozlíšená oproti letovej zóne žltou farbou a väčším polomerom.

4.1.12 Front-end: `BreachStatusComponent.jsx`

Súbor `BreachStatusComponent.jsx` definuje React komponent, ktorý je zodpovedný za zobrazenie informácií o narušení bezpečnostných zón dronom. Tento komponent vizualizuje najkritckejšie varovania, čím poskytuje dôležité informácie pre pilota dronu.

Funkcionalita komponentu

- Komponent prijíma pole `breaches`, ktoré obsahuje informácie o všetkých narušeniach bezpečnostnej zóny.
- Ak pole `breaches` neobsahuje žiadne narušenia, zobrazí sa správa „No breach detected“ so zelenou farbou, indikujúca, že žiadne narušenia neboli zaznamenané.
- v prípade existujúcich narušení sú tieto zoradené podľa typu a času do narušenia, čím sa zabezpečí, že najkritičkejšie varovanie je vždy zobrazené ako prvé.

Return

Komponent `BreachStatusComponent` vracia kód, ktorý reprezentuje vizuálnu informáciu o stave narušenia zóny. v prípade, že neexistujú žiadne narušenia, komponent zobrazuje správu „No breach detected“ so zelenou farbou textu. Táto správa je umiestnená vľavo hore mimo mapy v komponente `Home`, čo zabezpečuje, že je dobre viditeľná a nezasahuje do vizualizácie mapy. pri detekcii narušení sa zobrazuje najkritičkejšie varovanie s detailmi o type narušenia, čase do narušenia a výške, v ktorej narušenie nastane. Farba textu varovania sa mení podľa vážnosti narušenia: červená pre vážne narušenia letovej zóny a žltá pre menej kritické situácie.

4.1.13 Front-end: `Heatmap.jsx`

Súbor `Heatmap.jsx` definuje React komponent, ktorý je zodpovedný za spracovanie a zobrazenie mapy výskytu letovej prevádzky.

Importované zdroje

Komponent využíva knižnicu Leaflet pre mapové funkcionality a React hook `useEffect` na správu vedľajších efektov spojených s aktualizáciou mapy. Importuje tiež konfiguračné nastavenia z `StaticSettings.jsx`, kde je definovaná IP adresa backendu.

Konfigurácia vrstvy mapy výskytu

V komponente `Heatmap`, konkrétne v časti kódu na riadkoch 45 až 52, je dôležitá konfigurácia vrstvy mapy výskytu letovej prevádzky. V tejto časti sa nastavujú geografické hranice zobrazenia, minimálne a maximálne úrovne zoomu, a URL šablóna pre dlaždice. Zmena týchto parametrov musí byť konzistentná s dostupnými dlaždicami na serveri, aby bolo zabezpečené, že mapa korektne zobrazuje údaje pri rôznych úrovniach priblíženia.

Funkcionalita komponentu

- `Heatmap` komponent dynamicky reaguje na zmeny v nastaveniach ako sú nadmorská výška, rozlíšenie, vylúčenie letísk a nepriehľadnosť mapy. na základe týchto nastavení konfiguruje URL dlaždic (tiles), ktoré získava z back-endu.
- Vrstva mapy výskytu je pridávaná a odoberaná z hlavnej mapy v závislosti od týchto nastavení, čo zabezpečuje aktuálnosť zobrazenia podľa posledných užívateľských volieb.

Spracovanie a zobrazenie dát

Na základe nastavených parametrov komponent vypočíta príslušné indexy nadmorskej výšky a podľa nich prispôsobí zobrazenie heatmapy:

- URL šablóna pre dlaždice je konfigurovaná na základe užívateľských volieb, čo umožňuje flexibilitu v zobrazení rôznych typov údajov.
- Pridaná vrstva mapy výskytu prevádzky je dynamicky aktualizovaná pri zmene nastavení, čo zabezpečuje, že zobrazenie je vždy relevantné a presné.

4.1.14 Front-end: `Utils.jsx`

Súbor `Utils.jsx` obsahuje pomocné funkcie používané naprieč front-endom aplikácie.

Funkcia na výpočet vzdialenosti

Hlavnou funkciou v `Utils.jsx` je `calculateDistance`, ktorá poskytuje spôsob, ako vypočítať vzdialenosť medzi dvoma geografickými bodmi na zemskom povrchu pomocou Haversinovej formuly:

- Funkcia prijíma štyri parametre: `lat1`, `lon1` (zemepisná šírka a dĺžka prvého bodu) a `lat2`, `lon2` (zemepisná šírka a dĺžka druhého bodu).
- Výpočet začína prevodom zemepisných súradníc z stupňov na radiány.
- Aplikuje Haversinovu formulu na určenie veľkosti kruhového oblúka medzi dvoma bodmi na zemskom povrchu.
- Výsledná hodnota je vrátená v metroch, poskytujúc presnú meranú vzdialenosť, ktorá môže byť použitá v rôznych častiach aplikácie na určenie relatívnej polohy objektov.

4.1.15 Front-end: `StaticSettings.jsx`

Súbor `StaticSettings.jsx` definuje súbor konfigurácii, ktoré sú použité naprieč front-endom aplikácie. Tento modul centralizuje všetky statické nastavenia, čo zjednodušuje správu a aktualizáciu konfiguračných údajov.

Konfiguračné položky

V súbore sú definované nasledujúce konfiguračné položky:

- `BackendIP` a `IcecastIP` – Adresy pre backend a Icecast server. Tieto IP adresy musia byť verejné, ak sa aplikácia používa mimo lokálneho prostredia, aby boli dostupné z prehliadača.
- `ControlPanelBounds` – Definuje hranice a krokovanie pre posuvníky v ovládacom paneli, vrátane rozsahu pre letovú zónu, varovnú zónu, výšku, predikčnú dobu a priehľadnosť mapy výskytu prevádzky.

- `VerticalOverhead` – Definuje bezpečný vertikálny odstup v metroch.
- `DefaultMapSettings` – Prednastavené hodnoty pre inicializáciu mapy, vrátane geografického centra a úrovne priblíženia, ktoré sú prednastavené na Košice a základný zoom.

4.1.16 PostgreSQL: `aeroguardian-init.sql`

Súbor `aeroguardian-init.sql` obsahuje SQL skripty potrebné na inicializáciu databázového schématu pre systém AeroGuardian. Skripty definujú základnú štruktúru databázy a tabuľky potrebné na ukladanie a spracovanie získaných dát.

Vytvorenie databázy

Skript začína príkazom na vytvorenie novej databázy s názvom `aeroguardian`, čo je predpokladané úložisko pre všetky dáta a metadáta získané z rôznych zdrojov:

```
CREATE DATABASE aeroguardian;
```

Definícia tabuliek

Po vytvorení databázy skript pokračuje definovaním štruktúry tabuliek. Prvá tabuľka, `sources`, ukladá informácie o zdrojoch dát, vrátane identifikátora zdroja, názvu a webovej adresy prijímača:

```
CREATE TABLE sources (
source_id SERIAL PRIMARY KEY,
receiver_name VARCHAR,
receiver_website VARCHAR
);
```

Druhá tabuľka, `aircraft_pointcloud`, je navrhnutá na ukladanie bodových dát zachytených pre jednotlivé lietadlá. Každý záznam v tejto tabuľke predstavuje jedno meranie a obsahuje záznam o čase zachytenia a unikátnom kóde lietadla:

```
CREATE TABLE aircraft_pointcloud (
source_id INTEGER,
capture_time TIMESTAMP WITHOUT TIME ZONE,
hex_code VARCHAR,
PRIMARY KEY (source_id, capture_time, hex_code),
FOREIGN KEY (source_id) REFERENCES sources(source_id)
);
```

Postupné dopĺňanie stĺpcov

Tabuľka `aircraft_pointcloud` je špecificky navrhnutá tak, aby umožnila postupné dopĺňanie ďalších stĺpcov súvisiacich s dátami lietadiel (ako sú pozícia, výška, rýchlosť atď.), ktoré sú pridávané do databázy podľa dostupnosti a potreby. Inicializácia tabuľky zahrňuje len základné identifikačné a časové stĺpce, čo umožňuje flexibilitu v spracovaní a ukladaní rôznorodých datasetov počas rozvoja a prevádzky systému.

4.1.17 Logger: `json_to_postgre.py`

Súbor `json_to_postgre.py` obsahuje skript na presun dát z uložených JSON súborov `aircraft.json` do PostgreSQL. Tento skript je dôležitý pre archiváciu a analýzu historických dát zachytených z rôznych zdrojov.

Vstupné a výstupné parametre funkcií

- `create_daily_partition(conn, date_str)`:
  - **Vstup:** `conn` (psycopg2 spojenie), `date_str` (reťazec dátumu vo formáte 'YYYY-MM-DD').
  - **Výstup:** Funkcia vytvára partíciu v databáze pre daný deň.
- `fetch_existing_columns(conn)`:
  - **Vstup:** `conn` (psycopg2 spojenie).

- **Výstup:** Vráti množinu názvov stĺpcov existujúcich v tabuľke.

- `add_column_if_not_exists(conn, existing_columns, column_name)`:
  - **Vstup:** `conn` (psycopg2 spojenie), `existing_columns` (množina existujúcich stĺpcov), `column_name` (názov nového stĺpca).
  - **Výstup:** Funkcia pridáva stĺpec do tabuľky, ak ešte neexistuje.

- `process_file(file_path, conn, existing_columns, date_str)`:
  - **Vstup:** `file_path` (cesta k JSON súboru), `conn` (psycopg2 spojenie), `existing_columns` (množina existujúcich stĺpcov), `date_str` (dátum vo formáte 'YYYY-MM-DD').
  - **Výstup:** Dáta sú spracované a vložené do databázy.

- `process_hour(directory_path, conn, existing_columns, date_str)`:
  - **Vstup:** `directory_path` (cesta k adresáru), `conn` (psycopg2 spojenie), `existing_columns` (množina existujúcich stĺpcov), `date_str` (dátum vo formáte 'YYYY-MM-DD').
  - **Výstup:** Spracovávajú sa súbory v adresári za konkrétnu hodinu.

- `process_day(directory_path)`:
  - **Vstup:** `directory_path` (cesta k adresáru s dennými dátami).
  - **Výstup:** Spracovávajú sa všetky hodinové adresáre za deň.

Implementácia a použitie

Tento skript dynamicky spravuje štruktúru databázy podľa potreby a zabezpečuje, že všetky dáta zachytené v súboroch `aircraft.json` sú presne a účinne ukladané do databázy. Pomocou funkcií pre paralelné spracovanie a efektívne ukladanie dát skript poskytuje robustné riešenie pre dlhodobé logovanie letovej prevádzky.

4.1.18 Logger: `logger.py`

Súbor `logger.py` obsahuje skript na priebežné logovanie zachytených dát do PostgreSQL, využívajúci tabuľku `sources`. Je to alternatívou k `json_to_postgre.py`, ktorá miesto presunu dát zo záloh do databázy ukladá dáta do databázy priamo. Návod na použitie loggera sa nachádza v užívateľskej príručke.

Vstupné a výstupné parametre funkcií

- `create_daily_partition(conn, date_str)`: rovnaká funkcionalita ako `json_to_postgre.py`, viď kapitola 4.1.17.
- `fetch_sources(conn)`:
  - **Vstup:** `conn` (psycopg2 spojenie).
  - **Výstup:** Vráti zoznam dvojíc obsahujúcich `source_id`, `receiver_name`, a `receiver_website`.
- `fetch_json_from_source(url)`:
  - **Vstup:** `url` (URL adresa zdroja dát).
  - **Výstup:** Vráti JSON objekt alebo `None` pri chybe.
- `add_column_if_not_exists(conn, existing_columns, column_name)`: rovnaká funkcionalita ako `json_to_postgre.py`, viď kapitola 4.1.17.
- `process_data(data, conn, existing_columns, date_str, source_id)`:
  - **Vstup:** `data` (dáta na spracovanie), `conn` (psycopg2 spojenie), `existing_columns` (množina existujúcich stĺpcov), `date_str` (reťazec dátumu vo formáte 'YYYY-MM-DD'), `source_id` (identifikátor zdroja).
  - **Výstup:** Funkcia spracuje a uloží dáta do databázy.

4.1.19 OpenTopoData-postgre: `fill_grid_height.py`

Súbor `fill_grid_height.py` obsahuje skript na naplnenie databázy s výškovými dátami, ktoré sú využívané pri generovaní mapy výskytu letovej prevádzky. Potrebuje fungujúci server OpenTopoData. Návod na inštaláciu servera OpenTopoData je uvedený v užívateľskej príručke.

Vstupné a výstupné parametre funkcií

- `create_connection(connection_parameters)`:
  - **Vstup:** `connection_parameters` (slovník s parametrami pripojenia k databáze).
  - **Výstup:** Vráti psycopg2 spojenie s databázou.
- `get_elevation(lat, lon)`:
  - **Vstup:** `lat` (zemepisná šírka), `lon` (zemepisná dĺžka).
  - **Výstup:** Vráti nadmorskú výšku získanú z OpenTopoData, alebo 0 pri chybe.
- `insert_elevation_data(connection_parameters, records, airport, resolution)`:
  - **Vstup:** `connection_parameters` (parametre pripojenia), `records` (záznamy na vloženie), `airport` (kód letiska), `resolution` (rozlíšenie).
  - **Výstup:** Funkcia vkladá nadmorské výšky do databázy.
- `process_patch(lat_start, lat_end, lon_start, lon_end, resolution)`:
  - **Vstup:** `lat_start`, `lat_end` (začiatočná a koncová zemepisná šírka), `lon_start`, `lon_end` (začiatočná a koncová zemepisná dĺžka), `resolution` (rozlíšenie).
  - **Výstup:** Vráti zoznam záznamov obsahujúcich zemepisné súradnice a nadmorskú výšku pre vloženie do databázy.

- `fill_grid_ground_height(connection_parameters, lat_bounds, lon_bounds, resolution, airport, patch_size, max_workers)`:
  - **Vstup:** `connection_parameters`, `lat_bounds` (hranice zemepisnej šírky), `lon_bounds` (hranice zemepisnej dĺžky), `resolution` (rozlíšenie), `airport` (kód letiska), `patch_size` (veľkosť časti pre paralelné spracovanie), `max_workers` (maximálny počet procesov).
  - **Výstup:** Funkcia napĺňa databázu nadmorskými výškami na základe rozsahu zemepisných súradníc.

4.1.20 Heatmap: `generate_heatmaps_main.py`

Súbor `generate_heatmaps_main.py` slúži ako hlavný skript pre automatizované generovanie máp výskytu letovej prevádzky a následné tvorbu vizuálnych dlaždíc pre mapové zobrazenie. Tento skript koordinuje celý proces od získania predspracovaných dát až po ich vizualizáciu.

Nastavenie vstupných parametrov

V tomto skripte sa nastavujú kľúčové parametre, ktoré ovplyvňujú generovanie máp výskytu a dlaždíc. Tieto parametre určujú rôzne aspekty spracovania a vizualizácie dát:

- `resolutions`: Pole rozlíšení, pre ktoré sa budú generovať mapy. Umožňuje generovanie máp v rôznych mierkach detailov. Pod rozlíšením sa rozumie, na koľko častí sa rozdelí jeden stupeň zemepisnej šírky alebo dĺžky.
- `max_capture_time`: Určuje najnovší čas zachytenia dát pre generovanie máp.
- `airport`: Kód letiska, pre ktoré sa mapy generujú, využívaný pre odlíšenie v databáze.
- `grid_height_resolution`: Rozlíšenie výškového modelu mriežky, nadväzuje na tabuľku vygenerovanú z `fill_grid_height.py`.

- `max_height`: Maximálna výška v metroch, do ktorej sú dáta zohľadnené.
- `max_drawing_height`: Určuje maximálnu výšku v metroch, do ktorej sa budú dáta vizuálne zobrazovať.
- `vertical_step`: Vertikálny krok vo výpočtoch, určuje krokovanie medzi jednotlivými výškovými hladinami v metroch.
- `bounds`: Geografické hranice záujmovej oblasti, definované ako minimálna a maximálna zemepisná šírka a dĺžka, určujú oblasť, pre ktorú sú mapy generované.
- `zoom_levels`: Definuje úrovne priblíženia dlaždíc, ktoré sú potrebné pre ich generovanie.

Využívané funkcie

Skript volá rôzne funkcie z externých súborov, pričom každá má špecifické parametre:

- `run_parallel_preparation(...)` vykonáva predspracovanie dát v databáze.
- `generate_heatmaps(...)` a `generate_tiles_main(...)` sa starajú o vizuálne spracovanie a prezentáciu dát.
- `flip_images(...)` obracia dlaždice pre vizuálne korektné zobrazenie na mape.

Procesy

- Skript sa postupne zaoberá:
  - Paralelným predspracovaním dát.
  - Generovaním map výskytu podľa nastavených kritérií.
  - Tvorbu dlaždíc pre každé rozlíšenie.
  - Obracaním obrázkov dlaždíc pre korektné zobrazenie.
- Tieto kroky sú efektivizované pomocou viacvláknového spracovania.

4.1.21 Heatmap: `preparation_sql.py`

Tento skript obsahuje funkciu `run_preparation_sql`, ktorá zabezpečuje prípravu databázy pred generovaním máp výskytu letovej prevádzky. Vstupné parametre funkcie sú:

- `connection_parameters` – Parametre pripojenia k databáze.
- `airport` – Identifikátor letiska, pre ktoré sa dáta pripravujú.
- `resolution` – Rozlíšenie dátových bodov.
- `grid_height_resolution` – Rozlíšenie výškového modelu mriežky, nadväzuje na tabuľku vygenerovanú z `fill_grid_height.py`.
- `max_height` – Maximálna nadmorská výška, ktorá sa zvažuje pri vytváraní máp výskytu prevádzky.
- `max_drawing_height` – Maximálna výška pre kreslenie na mapách výskytu prevádzky.
- `vertical_step` – Vertikálny krok výšky, ktorý určuje intervaly pre agregáciu dát.
- `bounds` – Geografické hranice oblasti záujmu.
- `max_capture_time` – Najneskorší čas zachytenia údajov pre spracovanie.

Funkcia začína odstránením existujúcich tabuliek, čo zabezpečuje, že všetky predchádzajúce dáta sú vymazané a nebude dochádzať k ich akumulácii. Nasleduje vytvorenie nových tabuliek a vkladanie upravených dát do týchto tabuliek.

Procesy sú automatizované a paralelizované pomocou modulu `multiprocessing`, čo umožňuje efektívne spracovanie veľkého množstva dát. v rámci tohto kroku sa vykonávajú SQL skripty, ktoré zahŕňajú:

- Vytvorenie nových tabuliek pre ukladanie dát s príslušnými štruktúrami.

- Vkladanie dát z predspracovaných zdrojov, ktoré boli agregované podľa zadaných parametrov.

Výstupom funkcie je databáza pripravená na generovanie máp výskytu letovej prevádzky, čo zahŕňa pripravené dáta pre následné vizualizačné spracovanie. Databáza obsahuje aktualizované informácie, ktoré sú pripravené na efektívne vykreslenie výsledkov v rámci nástrojov na generovanie máp výskytu prevádzky.

4.1.22 Heatmap: `generate_heatmaps.py`

Tento skript je zodpovedný za generovanie máp výskytu letovej prevádzky pomocou dát získaných z PostgreSQL databázy a ich vizualizáciu cez knižnicu `matplotlib`. Mapy sú generované vo veľkom rozlíšení vo formáte PNG, pričom sú k dispozícii dve verzie pre každé rozlíšenie: jedna zahrňujúca letiskové oblasti do výpočtu gradientu a druhá ich nezahrnujúca. To umožňuje analýzu vplyvu letiskových operácií na vzdušný priestor.

Nastavenie vstupných parametrov

V skripte sa nastavujú geografické hranice letiska a vzletovo-pristávacej dráhy pomocou objektov `ShapelyPolygon`. Tieto polygóny slúžia na definíciu oblastí, ktoré budú alebo nebudú zahrnuté pri generovaní gradientov výskytu. Vstupné parametre pre generovanie zahŕňajú:

- `resolutions` – Pole rozlíšení, pre ktoré sa generujú mapy, umožňujúce detailné zobrazenie na rôznych úrovniach detailu.
- `max_alt` – Maximálna výška, pre ktorú sa zobrazujú dáta.
- `bounds` – Geografické hranice oblasti záujmu.
- `exclude_airport` – Boolovská hodnota určujúca, či sa má letisko započítať do gradientu.

Vykreslenie mapy výskytu

Proces vykreslenia zahŕňa aplikáciu gradientu na maticu, kde každá hodnota gradientu začína od 0.25, zabezpečujúc, že každý bod s výskytom má minimálnu viditeľnosť. Nápisy na mapách jasne označujú letiská, čo zlepšuje orientáciu na mapách. Mapy sú špecificky prispôsobené pre analýzu vzdušného priestoru okolo určených letísk.

Gradient a vizualizácia

Gradient pre každú bunku mriežky je vypočítaný ako lineárna interpolácia medzi minimálnou a maximálnou hodnotou počtu preletov, čo umožňuje detailné znázornenie distribúcie letovej prevádzky. Tento gradient poskytuje vizuálny prehľad o frekvencii preletov v rôznych oblastiach a rôznych výškach.

Celý proces generovania máp je paralelizovaný s použitím modulu `multiprocessing`, čo značne zvyšuje efektivitu generovania veľkého množstva dát. Mapy, ktoré sú vytvorené, sú užitočné pre plánovacie, monitorovacie a regulačné účely v letectve.

Výstup skriptu

Výstupom skriptu `generate_heatmaps.py` sú vygenerované mapy výskytu letovej prevádzky, ktoré sú uložené vo formáte PNG v adresárovom systéme. Tieto mapy zobrazujú akumulovaný výskyt letovej prevádzky v rôznych výškach nad zemou, kde každá mapa je generovaná pre konkrétne nastavené rozlíšenie a výškový krok.

Finálna adresárová štruktúra pre `generate_heatmaps.py` zahŕňa koreňový adresár `heatmap` rozdelený na dve hlavné sekcie: `airport_included` a `airport_excluded`, reprezentujúce mapy s letiskom započítaným, resp. nezapočítaným do gradientu. Každá sekcia obsahuje podadresáre pre rôzne rozlíšenia (`res_100`, `res_250` atď.), v ktorých sa nachádzajú PNG súbory pre rôzne výšky (500 m, 1 000 m atď.).

4.1.23 Heatmap: `generate_tiles.py`

Modul `generate_tiles.py` slúži na automatické generovanie dlaždíc z vygenerovaných máp výskytu letovej prevádzky. Tieto dlaždice sú určené pre použitie vo webových mapovacích aplikáciách ako Leaflet, umožňujúce efektívne zobrazenie veľkých množstiev dát na rôznych úrovniach zoomu.

Potrebné knižnice a nastavenie kódu

Proces generovania dlaždíc závisí na systémovej knižnici `GDAL`, ktorá poskytuje nástroje na manipuláciu a transformáciu obrazových dát, vrátane georeferencovania a konverzie formátov. pre prácu s dlaždicami je nutný externý skript `gdal2tiles.py`, ktorý je súčasťou GDAL a bežne je dostupný aj ako súčasť distribúcie `anaconda3`. Cesta k tomuto skriptu musí byť správne nastavená v kóde, ako je uvedené
na riadku 42.

Georeferencovanie obrázkov

Na začiatku procesu každý obrázok podstúpi georeferencovanie. Tento krok zahŕňa priradenie geografických koordinátov k obrazovým dátam, čo umožňuje ich správne zobrazenie na zemepisnej mape. Skript vykonáva georeferencovanie pomocou nástroja GDAL, konkrétne s príkazom `gdal_translate`.

Generovanie dlaždíc

Po georeferencovaní nasleduje generovanie dlaždíc pomocou skriptu `gdal2tiles.py`. Tento nástroj rozdelí obrázok na menšie sekcie (dlaždice), ktoré sú efektívnejšie pri načítaní v mapovacích aplikáciách. Parametre ako úroveň zoomu a rozlíšenie sú definované vstupnými parametrami skriptu, čo zaisťuje flexibilitu pre rôzne typy použitia.

Vstupné a výstupné premenné funkcií

Skript na generáciu dlaždíc definuje nasledujúce vstupné a výstupné premenné

pre hlavné funkcie a pomocné rutiny:

- **Funkcia `generate_tiles_main`**:
  - Vstupné premenné:
    - `resolution` – rozlíšenie dlaždic, určuje detaľnosť.
    - `bounds` – geografické hranice záujmovej oblasti.
    - `zoom_levels` – úrovne priblíženia pre dlaždice.
  - Výstup:
    - Štruktúrovaný priečinok s dlaždicami pripravenými na použitie v mapovacích aplikáciách.
- **Funkcia `georeference_image`**:
  - Vstupné premenné:
    - `input_png` – cesta k vstupnému PNG obrázku.
    - `output_tif` – cesta k výstupnému TIFF súboru.
    - `bounds` – geografické hranice pre georeferencovanie.
  - Výstup:
    - Georeferencovaný TIFF súbor.
- **Funkcia `generate_tiles`**:
  - Vstupné premenné:
    - `georef_tif` – cesta k georeferencovanému TIFF súboru.
    - `tiles_dir` – cesta k adresáru, kde sa majú uložiť dlaždice.
    - `zoom_levels` – špecifikované úrovne zoomu pre generovanie.
  - Výstup:
    - Adresár s dlaždicami pre použitie v mapovacích aplikáciách.

Paralelné spracovanie

Na optimalizáciu času spracovania, modul využíva paralelné spracovanie s pomocou `ThreadPoolExecutor` z Python standardnej knižnice. Tento prístup umožňuje súčasné spracovanie viacerých obrázkov, čo je výhodné najmä pri práci s veľkým počtom dát.

4.1.24 Heatmap: `flip_images.py`

Skript `flip_images.py` je určený na obracanie vygenerovaných dlaždíc pre aplikácie mapovania, zabezpečujúci, aby boli dlaždice správne otočené podľa geografickej orientácie. Skript využíva knižnicu Pillow na manipuláciu s obrázkami a paralelné spracovanie pre rýchlejšie vykonávanie operácií.

Funkcie

- `flip_image`: Otočí obrázok zhora nadol a uloží zmeny. Vstupnými premennými sú cesty k pôvodnému a výslednému obrázku.
- `collect_images_to_process`: Identifikuje všetky obrázky, ktoré vyžadujú spracovanie v špecifikovanom adresári a vráti zoznam s cestami k obrázkom.
- `flip_images`: Vykonáva hromadné obracanie obrázkov na základe zoznamu obrázkov získaných funkciou na ich zbieranie. Táto funkcia nemá priamy výstup, ale modifikuje obrázky priamo v zadanom adresári.

5 Náväznost na iné programové produkty

Back-end používa Python (najstaršia testovaná verzia 3.9) s frameworkom `Flask` (ku každému Python skriptu je priložený súbor `requirements.txt` s požadovanými knižnicami). Front-end na spustenie a preloženie používa platformu `Node.js`. Pri vývoji beží aplikácia priamo na `Node.js`, pri jej nasadení sa odporúča webový server

`Nginx`. Aplikácia funguje so spracovávaním ADS-B s SDR cez `dump1090-fa`, ktorý je dizajnovaný pre OS Raspbian.

Spracovanie dát používa PostgreSQL. Skript na generáciu dlaždíc mapy výskytu prevádzky používa systémovú knižnicu `GDAL` a externý skript `gdal2tiles.py` (ktorý je zahrnutý napríklad v distribúcii Pythonu `anaconda3`).

6 Zhodnotenie riešenia

Celkovo sa podarilo dosiahnuť stanovené ciele. Bola vytvorená rozšíriteľná sieť prijímačov dát o letovej prevádzke cez ADS-B, ktoré výrazne zlepšujú prehľad pilota dronu o okolitej prevádzke. Bol vytvorený aj funkčný plánovací a varovný systém, ktorý dokáže predpovedať budúci vývin situácie v letovej prevádzke a varovať pilota dronu včas pred budúcimi hrozbami. 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 použitej literatúry

FlightAware.com, 2024. *Dokumentácia k softvéru PiAware*. Dostupné na internete: `https://www.flightaware.com/adsb/piaware/` (citované 18.5.2024).

Richardson, M. and Wallace, S., 2012. *Getting started with raspberry PI*. O'Reilly Media, Inc.

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).

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

7 Zoznam príloh

Zdrojový kód aplikácie je dostupný na CD médiu záverečnej práce, alebo je dostupný na `https://github.com/IKS-TUKE-studenti-2023-2024/BP-Kando/tree/main/aeroguardian-BP-release`).

Zoznam obrázkov

Zoznam tabuliek