VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZHRANÍ PRO ZOBRAZOVÁNÍ RADAROVÝCH DAT A PROPOJENÍ SE SIMULÁTOREM ŘÍZENÍ LETECKÉHO PROVOZU

RADAR INTERFACE AND ITS INTERCONNECTION TO AIR TRAFFIC CONTROL SYSTEM SIMULATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **TEREZA BUCHNÍČKOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

145574

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Studentka: **Buchníčková Tereza**
Program: Informační technologie
Specializace: Informační technologie
Název: **Rozhraní pro zobrazování radarových dat a propojení se simulátorem řízení leteckého provozu**
Kategorie: Uživatelská rozhraní
Akademický rok: 2022/23

Zadání:

1. Seznamte se s existujícími možnostmi systémů pro trénink pracovníků řízení letového provozu a způsoby integrace rozpoznávání a syntézy řeči v této oblasti.
2. Prostudujte dostupné nástroje pro integraci radarových dat, případně další kontextové informace.
3. Na základě získaných znalostí rozšiřte stávající systém pro rozpoznávání řeči pilotů a operátorů řízení leteckého provozu, aby jej bylo možné využít pro výcvik začínajících operátorů.
4. Vytvořte testovací sadu pro vyhodnocení přínosu systému a vyhodnoťte vytvořený systém pomocí standardních uživatelských metrik.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Cílem této práce je vytvořit aplikaci pro výuku začínajících pracovníků řízení letového provozu. Systém je implementován jako webová aplikace v jazyce JavaScript s použitím knihoven JQuery a Leaflet. Serverová část je napsána v jazyce Python a využívá knihovnu BlueSky pro simulaci letového provozu. Práce obsahuje teoretickou část, návrh řešení a popis implementace. Výsledkem je aplikace, která nabízí zobrazení aktuálního letového provozu nebo v ní uživatel v roli řídícího letového provozu může na simulovaném letovém provozu trénovat komunikaci s pilotem. Aplikace umožňuje nahrávání hlasové komunikace a pomocí propojení se systémem rozpoznání řeči tuto komunikaci převádí do textu, který se zobrazuje na obrazovce. Kromě podpory výuky pracovníků letového provozu tato aplikace také slouží jako demonstrace výsledků výzkumných skupin KnoT a Speech z Fakulty informačních technologií VUT v Brně.

Abstract

This thesis aims to create an application for training new air traffic control officers. The system is implemented as a JavaScript web application using the JQuery and Leaflet libraries. The server part is written in Python using the BlueSky library for air traffic simulation. The thesis presents a theoretical background and discussed the design and implementation of the system. The result is an application that offers to display current air traffic, or where the user, in the role of air traffic control officer, can practice communication with a pilot on simulated air traffic. The application allows the recording of voice communication and, in cooperation with an automatic speech recognition system, converts this communication into text displayed on the screen. In addition to the support of the training of air traffic operators, this application also serves as a demonstration of the results of the research groups KnoT and Speech from the Faculty of Information Technology, Brno University of Technology.

Klíčová slova

řízení letového provozu, simulace, rozpoznání řeči, webová aplikace, řídící letového provozu, Python, JavaScript, RabbitMQ, BlueSky, WebRTC

Keywords

air traffic control, simulation, speech recognition, web application, air traffic control officer, Python, JavaScript, RabbitMQ, BlueSky, WebRTC

Citace

BUCHNÍČKOVÁ, Tereza. *Rozhraní pro zobrazování radarových dat a propojení se simulátorem řízení leteckého provozu*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Rozhraní pro zobrazování radarových dat a propojení se simulátorem řízení leteckého provozu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana docenta RNDr. Pavla Smrže, Ph.D. Další informace mi poskytl Ing. Karel Ondřej. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Tereza Buchníčková
7. května 2023

Poděkování

Chtěla bych poděkovat vedoucímu mé bakalářské práce doc. RNDr. Pavlu Smržovi za jeho odborné vedení, ochotu a cenné rady při vypracování této práce. Dále bych chtěla poděkovat Ing. Karlu Ondřejovi za poskytnutí důležitých informací.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Spolehlivé řízení letového provozu je velmi důležité pro zajištění plynulosti a bezpečnosti v oblasti civilního letectví. Řídící letového provozu potřebují pro svou práci nejen velké množství znalostí, ale také schopnost rychle reagovat a správně a srozumitelně komunikovat s pilotem. Nástroje pro trénink řídících letového provozu jsou nezbytné pro jejich správný výcvik.

Účelem této práce je vytvoření systému, který by bylo možné použít pro trénink začínajících řídících letového provozu. Dalším využitím je demonstrace výsledků výzkumných skupin KnoT a Speech z Fakulty informačních technologií VUT v Brně. Cílem je návrh a implementace aplikace, která bude zobrazovat aktuální radarová data nebo simulovaný letový provoz. Aplikace bude propojena se systémem pro rozpoznávání řeči pilotů a řídících letového provozu a rozšíří možnosti tohoto systému. Díky tomuto propojení by mělo být možné v aplikaci zobrazovat přepis komunikace mezi pilotem a řídícím letového provozu a na základě této komunikace ovlivňovat simulovaný provoz. Spojením simulátoru se systémem rozpoznání řeči vznikne komplexní nástroj pro výcvik, který umožní budoucím řídícím letového provozu, aby si prakticky vyzkoušeli a nacvičili všechny aspekty své práce.

Využití simulace umožňuje vytvoření virtuálního prostředí, ve kterém si řídící letového provozu může vyzkoušet různé scénáře a to, jak by v těchto případech reagoval. Simulace mu umožňuje připravit se na reálné situace při provozu a získat praktické dovednosti bez jakéhokoliv rizika. Simulace hraje klíčovou roli také v testování nových systémů. V aplikaci bude možné zadávat scénáře různé obtížnosti a díky tomu ji bude možné využít v různých fázích výcviku.

Systém bude dále poskytovat možnost vyzkoušet si komunikaci s pilotem. Správná komunikace mezi pilotem a řídícím letového provozu je významná pro zajištění bezpečnosti. Systém pro rozpoznání řeči průběžně převádí komunikaci na text a dále z ní vytváří přehledný zápis. Přepis komunikace může při výcviku pomoci například tak, že si účastníci mohou procházet historii komunikace a příkazů, které použili a mohou je zpětně analyzovat. Díky tomuto systému si také řídící letového provozu může všimnout chyby v komunikaci, především chyby při opakování příkazu pilotem, a reagovat na ni. Přehlédnutí takové chyby v reálném provozu by mohlo mít fatální následky. Systém rozpoznání řeči také pomáhá snížit pracovní zátěž řídících letového provozu.

Text práce je rozdělený do deseti kapitol. Kapitola 2 shrnuje informace týkající se letového provozu. Zejména se zaměřuje na to, jak probíhá řízení letového provozu a rádiová komunikace. Dále popisuje využití simulace v této oblasti a simulační knihovnu BlueSky. V kapitole 3 se nachází teorie z oblasti vizualizace geografických dat na mapě a popis několika knihoven, které je možné pro vizualizaci použít. Kapitola 4 se věnuje vývoji webových

aplikací a jejich rozdělení a popisuje technologie a jazyky, které jsou používané při tvorbě webových stránek. Dále jsou zde také popsány možné způsoby komunikace mezi serverem a webovou aplikací. Kapitola 5 popisuje programovací jazyk Python, systémy pro zasílání zpráv, a dále použití systémů rozpoznávání řeči v letectví. Kapitola 6 se zabývá volně dostupnými nástroji pro sledování letového provozu a dále popisuje různé systémy pro výuku řídících letového provozu. Kapitola 7 se věnuje návrhu aplikace, formátům dat zasílaných webové aplikaci a použitým technologiím. Implementace aplikace je rozebrána v kapitole 8. Kapitola 9 popisuje postupy využité při testování a vyhodnocení jeho výsledků. Poslední kapitola 10 obsahuje závěrečné shrnutí celé práce.

Kapitola 2

Letový provoz

Tato kapitola shrnuje vybrané teoretické informace z oblasti letového provozu a řízení letového provozu, které jsou důležité pro správné pochopení všech částí aplikace. Zaměřuje se na to, jakým způsobem řízení probíhá, kdo ho zajišťuje a jak mezi sebou obě strany komunikují. Dále jsou v kapitole popsány některé základní údaje o letadlech, které jsou důležité při komunikaci. Závěrem kapitola obsahuje informace o standardním formátu pro zasílání a výměnu informací služeb řízení letového provozu.

2.1 Řízení letového provozu

Řízení letového provozu (ATC[1]) je služba, která je poskytována pilotům letadel. Jejím hlavním účelem je předcházet a zabraňovat kolizím mezi letadly. Dále zajišťuje organizovaný a plynulý provoz a v případě potřeby poskytuje podporu bezpečnostním organizacím [15].

Před odletem letecká společnost nahlásí ATC letový plán. Předání informací o plánu letu je velmi důležité. Každý pracovník kontroly letového provozu, který je odpovědný za daný let, musí mít podrobnou představu o celé trase letu. Na letišti jsou piloti letadla v kontaktu s místními řídícími letového provozu (ATCO[2]) v řídící věži na letišti. ATC poté uděluje letadlu povolení ke startu. Jakmile je letadlo ve vzduchu, pilot komunikuje s dalším řídícím letového provozu, který pomocí radarové obrazovky sleduje postup letadla. ATC zajišťuje, že se piloti drží plánu letu a jsou celou dobu v kontaktu pro případ, že by z nějakého důvodu došlo k vybočení z trasy [49].

Řízení letového provozu je velmi komplexní, a proto je zajišťováno různými specializovanými středisky, která mezi sebou spolupracují. Obvykle se jedná o následující tři střediska [57]:

- **Letištní služba řízení:** Na letišti je provoz řízen týmem pracovníků z řídící věže. Pracovníci ve věži zajišťují bezpečný provoz na přistávací dráze, pojezdových drahách a v řízeném okrsku. Řízený okrsek je vzdušný prostor v okolí letiště.
- **Přibližovací služba řízení:** Přibližovací služba řídí provoz v oblasti vzdušného prostoru v širším okolí letiště. Tato oblast se nazývá koncová řízená oblast a je v ní velký provoz přilétajících a odlétajících letadel. Přibližovací služba zajišťuje, aby byly mezi letadly během letu dostatečné rozestupy, aby bylo zabráněno srážkám, a letadla se bezpečně přiblížila k letišti.

[1] ATC – air traffic control
[2] ATCO – air traffic control officer

- **Oblastní služba řízení:** Oblastní služba řízení spravuje provoz v dané části vzdušného prostoru, tzv. řízené oblastí. Tyto oblasti jsou většinou rozsáhlé a mohou zahrnovat i území celých států.

Komunikace v letovém provozu

Pro správnou organizaci letového provozu je podstatná komunikace pilota s ATC. Jak má tato komunikace probíhat, je přesně definováno a obě strany musí tyto postupy dodržovat. Je velmi důležité, aby se obě strany po celou dobu snažily vyjadřovat co nejpřesněji a nejsrozumitelněji, aby nedocházelo k nedorozuměním. Komunikace musí obsahovat všechny důležité informace, ale zároveň by měla být také stručná, protože ATCO monitoruje několik letadel najednou. V případě příliš dlouhých sdělení by ATCO nestíhal předávat pokyny ostatním pilotům [14].

Jak již bylo zmíněno v předešlé části, letecký prostor je rozdělen do oblastí. V každé oblasti, ve které se letadlo právě nachází, je monitorováno jedním nebo více řídícími letového provozu, kteří jsou zodpovědni za danou oblast. Po dobu, co se letadlo nachází v této oblasti, ho kontrolují a dávají mu pokyny. Když letadlo oblast opustí, přesune se tato zodpovědnost na ATCO nové oblasti [17].

Při navázání prvního kontaktu se stanicí nebo s novým ATCO je nutné nejprve uvést název stanice, se kterou se snaží pilot komunikovat. Dále je potřeba udat volací znak, který identifikuje dané letadlo. Správné udání volacího znaku je velmi důležité, aby se tím potvrdilo, že příkazy dorazily správnému letadlu. Jinak by mohlo dojít k tomu, že by pilot provedl příkazy určené pro někoho jiného.

Pokud se letadlo nachází na letišti, je nutné sdělit jeho polohu. Dále se uvede typ zprávy nebo požadavek. Pokud pilot volá pozemní stanici, měl by zahájit komunikaci udáním názvu místa a typu stanice.

Při rádiové komunikaci se používá hlásková abeceda, zvláště v případě špatných komunikačních podmínek. Používá se také, pokud na stejné frekvenci komunikují letadla s podobnými volacími znaky, a dále při prvním kontaktu s řídícím letového provozu. Většina čísel se vyslovuje po číslicích. Časové jednotky se udávají jako koordinovaný světový čas (UTC[3]), který se označuje jako „zulu time“ [14].

Výcvik řídících letového provozu

Výcvik řídících letového provozu je rozdělen do několika fází. Během výcviku musí uchazeč prokázat teoretické znalosti a praktické dovednosti. První částí je vstupní výcvik, který se dále dělí na základní výcvik (basic training) a výcvik pro udělení kvalifikace (rating training).

Cílem základního výcviku je poskytnout uchazečům znalosti a praktické dovednosti v oblasti základních provozních postupů. Kvalifikační výcvik poskytuje uchazečům znalosti a praktické dovedností týkající se jejich budoucího zaměření.

Po úspěšném absolvování vstupního výcviku následuje místní výcvik (operational training). Místní výcvik probíhá na stanovišti letových provozních služeb a je rozdělen do tří fází [45]:

- **Přechodová fáze (transitional training):** Během této fáze probíhá seznámení se stanovištěm, teoretická příprava a výcvik na simulátorech.

[3]UTC – coordinated universal tme

- **Předprovozní výcvik (pre on the job training):** Tato fáze se soustředí na přípravu uchazečů na úspěšné absolvování provozního výcviku. Fáze zahrnuje rozsáhlé simulace komplexních provozních situací a také nouzových situací až do maximální provozní zátěže daného pracoviště.
- **Provozní výcvik (on the job training):** V této fázi se provádí praktický výcvik v reálných podmínkách, kdy jsou znalosti postupů a dovednosti získané v předchozí fázi dále rozvíjeny a upevňovány.

2.2 Důležité pojmy

Na radarové obrazovce je třeba zobrazovat některé základní údaje o letadlech, jako je jednoznačná identifikace letadla, rychlost nebo výška. Dále je nutné znát zeměpisné souřadnice, aby bylo možné zobrazit polohu letadla. V této podkapitole jsou rozepsány některé základní pojmy týkající se letadel a řízení letového provozu.

Volací znak

Volací znak (v angličtině callsign) je sekvence alfanumerických znaků, která se používá pro identifikaci letadla při rádiové komunikaci. Může se jednat o registrační číslo letadla, nebo o kód určený danou leteckou společností, za kterým následuje číslo letu.

Soukromá a komerční letadla obvykle používají jako volací znak registrační číslo, které je každému letadlu přiděleno při registraci u vládní agentury FAA[4]. Toto číslo se nachází na trupu letadla. Každý stát má specifické počáteční znaky registračního čísla. Pro Českou republiku jsou to například písmena OK.

Letecké společnosti mají většinou specifické volací znaky. Pro rozlišení letadel v rámci letecké společnosti se k názvu přidává číslo letu. Příklady volacích znaků leteckých společností [30]:

- Delta Air Lines – DELTA
- British Airways – SPEEDBIRD
- Czech Airlines – CSA

SQUAWK kód

SQUAWK kód je čtyřmístné číslo, které přiděluje ATC všem letadlům ve vzdušném prostoru a používá se pro komunikační účely. Piloti musí zadávat SQUAWK kódy do svého komunikačního zařízení – transpondéru, aby mohli komunikovat s řídícími letového provozu. [42].

Výška letadla

Výška letadla se obvykle zjišťuje pomocí výškoměru, který měří barometrický tlak v okolí letadla. Výšku letadla není možné počítat vždy stejným způsobem. Pokud je letadlo na letišti, udává se výška na základě korigovaného tlaku (QNH) a jedná se o nadmořskou výšku letadla. Když je letadlo v převodní výšce (TA[5]), pilot nastaví výškoměr na standardní

[4] FAA – Federal Aviation Administration
[5] TA – transition altitude

tlak 1013 hPa. Převodní výška se na různých místech liší. V Evropě je to například 5000 stop. Výška letadla se nad touto hranicí měří na základě standardního tlaku a uvádí se jako letová hladina (FL[6]). Letová hladina se udává ve stovkách stop (FL 200 je tedy 20000 stop). Od daného okamžiku mají všechna letadla v okolí totožně nastavený výškoměr a díky tomu mění svou výšku stejně. Tímto způsobem jsou dány neměnné výškové rozestupy mezi letadly.

Nad převodní výškou se také nachází převodní hladina (TL[7]). V případě klesání letadla musí pilot při průletu touto hladinou nastavit výškoměr na skutečný tlak na letišti. Informaci o převodní hladině získá pilot od řídícího letového provozu [36].

Traťové body

Pro každý let je nutné vytvořit plán letu. Traťové body (waypoints) jsou body, kterými je tvořena dráha letu. Jsou definovány geografickými souřadnicemi a názvem, který je nejčastěji tvořený pěti velkými písmeny.

Traťové body nejsou rozmístěny náhodně. Například nad Evropou může letadlo proletět přes traťový bod každou minutu, ale na místech, kde není letecká doprava příliš hustá, se nenachází tolik traťových bodů. Většina traťových bodů, přes které letadlo prolétá, se nachází na začátku a na konci letu, kdy letadlo musí častěji měnit směr kvůli vzletu a přistání [56].

2.3 Výměna informací v oblasti letectví

Výměna informací a dat v oblasti letectví je velmi důležitá, zvláště pro zajištění bezpečnosti. Informace si mezi sebou vyměňují letadla, kontrolní věže, letová informační centra a další subjekty. Pro komunikaci se využívá protokol ASTERIX, který zajišťuje rychlou a efektivní výměnu dat.

Komunikační protokol ASTERIX

ASTERIX je protokol založený na binárním formátu dat, který byl navržen pro komunikační zařízení s omezenou šířkou pásma. Proto byl vytvořen tak, aby bylo možné přenášet maximum informací při využití co nejmenšího objemu dat [12]. Zkratka ASTERIX označuje „All–Purpose Structured Eurocontrol Surveillance Information Exchange“. Tento formát byl vyvinut organizací Eurocontrol[8]. ASTERIX se používá jako standard pro přenos a výměnu informací letových provozních služeb (ATS) [53].

ATS je obecný název služeb, které pomáhají letadlům v reálném čase, a zajišťují jejich bezpečný provoz. Tyto služby zabezpečují organizování letového provozu a zabraňují vzniku nehod mezi letadly. Jejich úkolem je také informovat příslušné organizace v případě nehody a nutnosti pátracích a záchranných operací a dále těmto organizacím případně asistují [50].

ASTERIX obsahuje několik typů zpráv (tzv. kategorií), z nichž každá se zabývá konkrétním druhem informací a má specifickou strukturu. Mezi tyto informace patří kupříkladu data o letadlech z přehledových senzorů, jako například radarů. Dále také předané informace, jako jsou trasy letadel a zprávy o stavu systému. Každé zařízení využívající formát ASTERIX má přiřazeno jedinečný identifikátor, který se skládá ze dvou osmibitových hodnot, System Area Code (SAC) a System Identification Code (SIC) [53].

[6] FL – flight level
[7] TL – transition level
[8] https://www.eurocontrol.int/

2.4 Simulace letového provozu

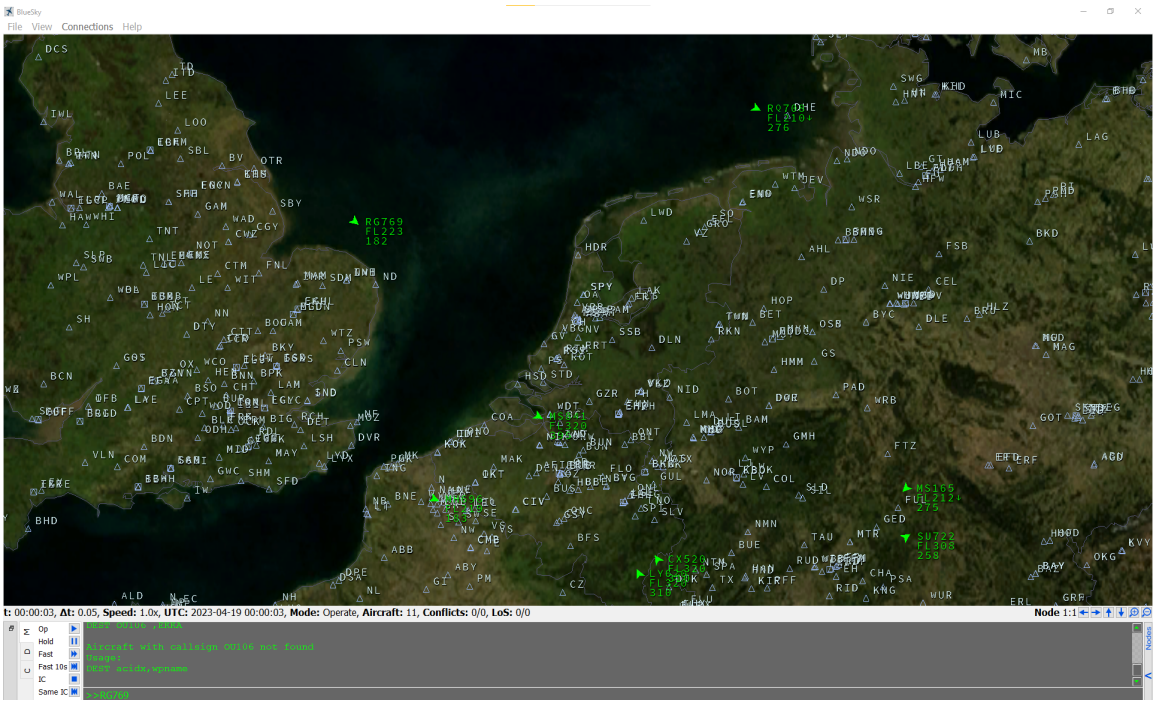
Simulace letového provozu je důležitá především při vývoji leteckých systémů. Simulace se používá k vytvoření virtuálního prostředí, ve kterém je možné zkoušet různé scénáře a neočekávané situace. Simulace letového provozu je také významná pro výuku nových řídících letového provozu a pilotů, kteří si tímto způsobem mohou vyzkoušet, jak by reagovali v různých situacích.

BlueSky

BlueSky[9] je open-source simulační nástroj, který je určen pro výzkum řízení letového provozu a toků letového provozu. Nástroj je možné dále rozšířit za pomocí samostatných modulů (pluginů). Simulace je řízena pomocí uživatelských vstupů z konzole nebo přehráváním souborů obsahujících scénáře.

BlueSky umožňuje uživatelům přizpůsobit simulaci jejich specifickým potřebám a požadavkům pomocí úpravy mnoha parametrů. Samotnou simulaci je možné ovládat pomocí sady příkazů. Pomocí těchto příkazů je možné například přidat nové letadlo do simulace, změnit rychlost letadla, změnit směr letadla nebo přidat traťový bod do plánu letu.

Simulátor BlueSky je uživatelsky přívětivý a poskytuje intuitivní uživatelské rozhraní, díky kterému lze vizualizovat simulaci letového provozu a zadávat příkazy, které simulaci ovlivňují [24]. Ukázku tohoto uživatelského rozhraní je možné vidět na obrázku 2.1



Obrázek 2.1: Ukázka uživatelského rozhraní BlueSky simulátoru

[9] http://homepage.tudelft.nl/7p97s/bluesky/

Kapitola 3

Vizualizace dat na mapě

Tato kapitola se nejprve zabývá obecnými informacemi o geografických údajích. Dále popisuje formáty pro jejich reprezentaci a zaměřuje se na formát GeoJSON, který byl použit v této práci pro zobrazení dat na mapě. Následně se kapitola také věnuje některým vybraným knihovnám pro vizualizaci geografických dat a popisuje jejich použití.

3.1 Geografické údaje

Geografické údaje jsou informace popisující objekty nebo jevy na povrchu Země nebo v blízkosti jejího povrchu. Obvykle obsahují informaci o poloze, která se většinou udává jako zeměpisné souřadnice, další atributy (charakteristiky daného objektu nebo jevu) a časové údaje, které definují dobu platnosti polohy. Poloha objektu může být dynamická nebo statická. Dynamická se na rozdíl od statické v čase mění [51].

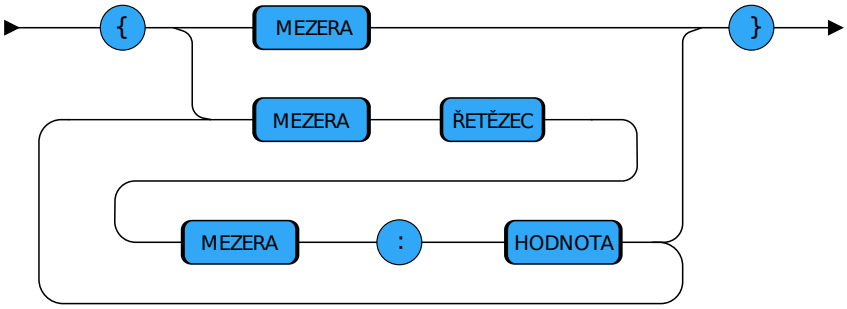
3.2 Formát pro posílání geografických dat

Pro zpracování geografických dat je možné použít mnoho různých formátů mezi které patří například Shapefile, KML/KMZ, GML nebo WKT a mnoho dalších. Knihovna pro vytváření map používaná v této práci, podporuje formát GeoJSON. Základem GeoJSONu je formát JSON (JavaScript Object Notation) [19].

JSON je formát určený pro přenos dat. Tento formát je pro člověka velmi dobře čitelný a dá se jednoduše zapsat. Pro stroje je zase snadné ho analyzovat a generovat. Struktura JSONu se skládá z dvojic, které obsahují název pole a jeho hodnotu. Hodnota může být různého typu. Může se jednat o řetězec, číslo, pravdivostní hodnotu (true, false), objekt nebo pole. Tyto struktury mohou být také vnořené [35]. Struktura formátu JSON je přehledně zobrazena na obrázku 3.1.

Formát GeoJSON je ve své podstatě stejný jako JSON, ale musí dodržovat určitou specifickou strukturu. Koncepty použité v tomto formátu nejsou úplně nové, ale jsou převzaty z již existujících geografických informačních systémů a byly upraveny do podoby, která je efektivnější pro použití ve webových aplikacích. Jednotlivé objekty v tomuto formát obsahují atributy `type`, `geometry` a `properties`.

Atribut `type` označuje, jestli se jedná o jeden prvek (Feature) nebo seznam více prvků (FeatureCollection). Pomocí GeoJSONu je možné popsat jakékoliv geografické objekty a nemusí se ani jednat o fyzické struktury.



Obrázek 3.1: Schéma zobrazující části formátu JSON

Atribut `geometry` udává, o jaký geometrický útvar se jedná (type). Dále udává souřadnice jednotlivých bodů daného objektu, které se skládají ze zeměpisné šířky a délky (coordinates). GeoJSON podporuje následující geometrické útvary:

- **Point:** bod v prostoru, který má jen jednu dvojici souřadnic
- **LineString:** čára v prostoru, jejíž souřadnice se skládají z souřadnic krajových bodů zapsaných v poli
- **Polygon:** mnohoúhelník
- **MultiPoint:** skupina bodů
- **MultiLineString:** skupina čar v prostoru
- **MultiPolygon:** skupina mnohoúhelníků
- **GeometryCollection:** skupina různých geometrických útvarů

Atribut `properties` může obsahovat jakékoliv doplňující informace, týkající se daného geografického objektu (například název, adresa, popis) [19].

Následující úryvek kódu ukazuje příklad zápisu bodu na mapě ve formátu GeoJSON [6].

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [14.42,50.087]
      },
      "properties": {
        "name": "Prague"
      }
    }
  ]
}
```

3.3 Knihovny pro vizualizaci geografických dat

Pro snadné zobrazování geografických dat existuje mnoho různých knihoven. Tyto knihovny podporují vytváření nejrůznějších druhů map a vizualizací. V této kapitole jsou popsány knihovny v jazyce Python a dále knihovna Leaflet[1] v jazyce JavaScript, která byla použita v této práci.

Folium

Folium[2] je knihovna v Pythonu, která umožňuje rychlé vytvoření jednoduchých vizualizací na mapě. Pro tvorbu interaktivních map využívá Folium knihovnu Leaflet. Folium umožňuje jednoduché vytvoření interaktivní mapy světa, kterou je možné posouvat a měnit přiblížení. Folium podporuje využití různých mapových dlaždic (map tiles), jako jsou například OpenStreetMap. Dále je možné za pomocí této knihovny přidávat na mapu různé vrstvy, značky a mnoho dalšího [5].

GeoPandas

GeoPandas[3] je open-source projekt, který usnadňuje práci s geografickými daty v Pythonu. GeoPandas rozšiřuje datové typy z knihovny Pandas a geometrické operace jsou prováděny za pomocí knihovny Shapely. GeoPandas dále využívá knihovnu Fiona pro načítání dat a Matplotlib pro vytváření vizualizací [18].

GeoPandas poskytuje rozhraní pro rychlé a snadné vykreslení statických map. GeoPandas je možné použít pro přidávání podkladových map, zpracování a analýzu geografických dat nebo například pro manipulaci s geografickými daty [1].

Holoviz

Holoviz[4] je knihovna v jazyce Python, která umožňuje jednoduché a přesné vytváření vizualizací. Holoviz poskytuje mnoho nástrojů pro vizualizaci dat. Mezi tyto nástroje patří například Panel pro vytváření aplikací a grafů. HoloViews pro vytváření vizualizací z dat a Datashader, který je vhodný pro práci s velkými datovými sadami [26].

Pro zobrazení geografických dat se používá knihovna Geoviews. Tato knihovna usnadňuje výzkum a vizualizaci geografických, meteorologických a oceánografických dat. Pomocí této knihovny je možné snadno vytvářet interaktivní mapy nebo grafy [1].

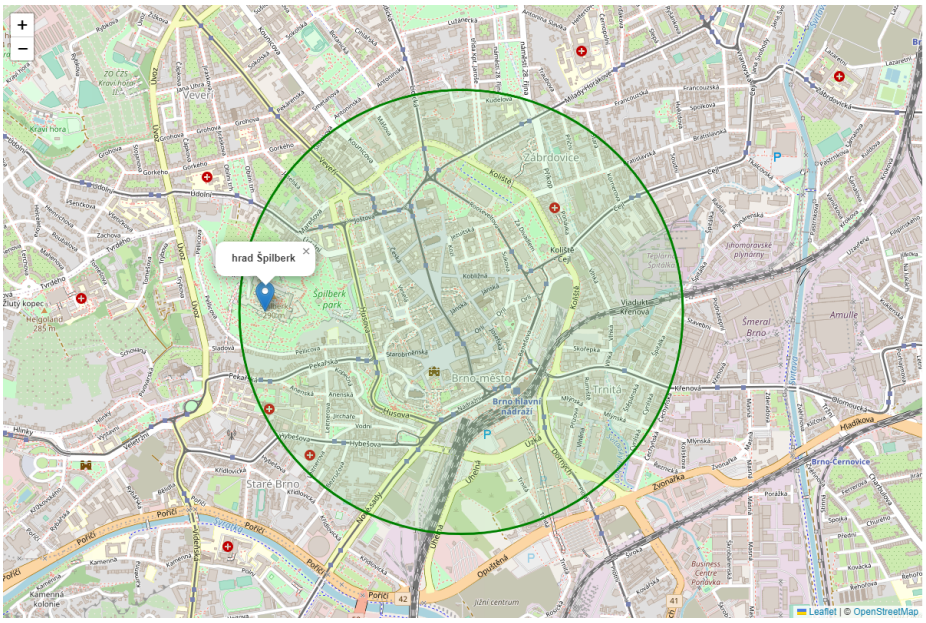
Leaflet

Leaflet je open-source knihovna napsaná v jazyce JavaScript určená pro vytváření interaktivních map ve webových prohlížečích. Knihovna nabízí mnoho funkcí a rozšíření. Leaflet byl vytvořen se snahou o jednoduchost a použitelnost a je obsáhle a podrobně zdokumentován. Tato knihovna je vhodná pro webové i mobilní aplikace [2].

Mezi její hlavní výhody patří možnost snadno vytvořit a zobrazit dynamickou mapu na základě různých mapových podkladů. Poskytuje možnosti, jak označit určité místo na mapě libovolnou ikonou a přidat k němu popisek, přidat na mapu několik nezávislých vrstev, barevně zvýraznit určitou oblast, a mnoho dalších funkcí. Leaflet také podporuje vykreslení

[1] https://leafletjs.com/
[2] https://python-visualization.github.io/folium/
[3] https://geopandas.org/
[4] https://holoviz.org/

geografických dat, případně i celé mapy ve formátu GeoJSON. Vzhled mapy i veškerých zobrazených prvků lze přizpůsobit pomocí kaskádových stylů. Knihovna také podporuje jednoduchý způsob implementace reakcí na akce uživatele, jako je výběr objetu myší nebo změna přiblížení mapy.



Obrázek 3.2: Ukázka zobrazení mapy pomocí knihovny Leaflet

Kapitola 4

Webové technologie

Tato kapitola nejprve popisuje základní informace ohledně tvorby webových stránek a aplikací. Dále popisuje rozdíl mezi statickou a dynamickou stránkou. Poté se zabývá rozdělením vývoje na backendovou a frontendovou část. Kapitola také popisuje jednotlivé jazyky a knihovny určené pro vývoj webových aplikací a zejména se zaměřuje na jazyky použité v této práci. Kapitola se také věnuje tomu, jak probíhá komunikace s webovou aplikaci a jaké protokoly se pro komunikaci využívají.

4.1 Vývoj webových aplikací

Termín vývoj webu představuje proces navržení a vytvoření webové stránky nebo aplikace. Vývoj webové stránky se skládá z mnoha kroků, jako je návrh vzhledu, vytváření obsahu, programy a skripty na klientské a serverové straně a dále například řešení zabezpečení stránky. Důležitá je také správa, údržba a průběžné aktualizace. Webové stránky mohou být velmi jednoduché, ale může se také jednat o rozsáhlé aplikace s mnoha dynamickými prvky [46].

Webové aplikace se od webové stránky liší tím, že uživatel může vytvářet a měnit data, která se v aplikaci zobrazují. Naopak statické webové stránky poskytují uživateli pouze výstup a většinou se na nich vyskytuje obsah jen pro čtení [33].

Statická a dynamická webová stránka

Webové stránky se dále dělí na statické a dynamické. Statický web se skládá ze stránek, které jsou vytvořeny pomocí HTML, CSS a JavaScriptu. Jeden z charakteristických znaků statické stránky je, že každý uživatel vidí ten stejný obsah.

Dynamické webové stránky umožňují zobrazení dynamicky se měnícího obsahu na základě požadavků uživatele a jeho interakcí. Díky tomu je možné přizpůsobit webovou stránku a její obsah konkrétnímu uživateli [55].

Frontend a Backend

Vývoj webové stránky nebo aplikace se dělí na dvě hlavní oblasti frontend a backend. Frontend je klientská část webu, se kterou uživatel komunikuje a interaguje. Frontend by měl být navržen tak, aby byla stránka přehledná a pro uživatele bylo jednoduché s ní pracovat. Mezi často používané prvky na frontendu patří tabulky, obrázky, barvy, text a mnoho dalších vizuálních prvků. Pro vývoj frontendu webových stránek se používají různé jazyky,

jako například HTML, CSS a JavaScript. V dnešní době také existuje mnoho frameworků, mezi které patří kupříkladu jQuery, React a AngularJS, které vývoj ulehčují.

Backend se zabývá správou dat webové stránky nebo aplikace. Zejména řeší posílání, organizaci a ukládání dat. Backendová část je uživateli skryta. Jedná se o serverovou část webu. Backend je s frontendem propojený a poskytuje mu funkcí v oblasti správy dat. Mezi jazyky používané při vytváření backendu patří například PHP, Python, C++, Ruby a Java. Dále existuje také množství frameworků, které ulehčují práci s backendem jako Laravel, Spring Django nebo Express [25].

4.2 Jazyky pro vývoj webových aplikací

Při vývoji webových aplikací je možné použít různé jazyky a technologie. Mezi nejznámější jazyky pro vývoj webových aplikací patří JavaScript, který se používá pro vytváření interaktivních prvků. Pro tvorbu grafického uživatelského rozhraní se nejčastěji používají jazyky HTML a CSS.

HTML

HTML[1] je základem všech webových stránek. Kód v souboru HTML je tvořen značkami (tag). Značky se dělí na párové a nepárové. Nepárové značky nemají ukončovací část. Značky také často obsahují povinné a nepovinné parametry, které udávají dodatečné informace k danému elementu. Tyto značky určují, jak má element vypadat, jaké se má použít formátování a o jaký element se jedná [58].

Kód HTML uchovává a popisuje obsah webové stránky. Webová stránka se skládá ze tří komponent: textového obsahu, odkazů na jiné dokumenty a HTML značek. Odkazy slouží pro vložení obrázků nebo například videozáznamů na stránku. Dále se používají pro odkazování na jiné HTML stránky, šablony stylů a soubory s kódem v jazyce JavaScript. Kromě těchto komponent obsahuje HTML kód také informace o dané webové stránce. Mezi tyto informace patří například jazyk obsahu stránky a znaková sada textu [7].

CSS

CSS[2](v češtině skádové styly) je jazyk určený pro popis vzhledu elementů na stránce napsané v jazycích HTML, XHTML nebo XML. Důvodem vzniku tohoto jazyka byla snaha oddělit od sebe vzhled stránky a jeho obsahovou část. Jazyk CSS byl vytvořen organizací W3C[3] [11]. Tento jazyk umožňuje formátovat text, nastavit rozvržení jednotlivých elementů na stránce a poskytuje také podporu pro formátování tisku. Jazyk CSS se skládá z pravidel, která mají vždy dvě části: selektor a deklarační blok. Selektor udává, pro jaký element je styl definován, deklarační blok specifikuje, co by se mělo s elementem dít. Tento blok může obsahovat několik vlastností a každá vlastnost má název a hodnotu [7]. Struktura CSS pravidla je znázorněno na obrázku 4.1.

[1] HTML – Hypertext Markup Language
[2] CSS – Cascading Style Sheets
[3] W3C – World Wide Web Consortium

Obrázek 4.1: Schéma pravidel stylů jazyka CSS

JavaScript

JavaScript je hlavním programovacím jazykem v rámci webu. JavaScript vznikl v roce 1995, jeho autorem je Brendan Eich z tehdejší společnosti Netscape. Jedná se o velmi silný, rychlý a jednoduchý jazyk, který je součásti webových stránek, a zajišťuje jejich dynamičnost. JavaScript přináší mnoho možností pro vytváření webové stránky. Mezi funkce JavaScriptu patří ovládání interaktivních prvků na stránce, jako jsou tlačítka, rozbalovací menu nebo zatrhávací políčka. Dále se používá pro ověření dat zadaných uživatelem. Díky tomu není potřeba posílat data pro ověření zpět na server a to urychluje celý proces. Syntax tohoto jazyka vychází z jazyka C. Jde o dynamicky typovaný jazyk, což znamená, že proměnná může nabývat hodnot různých datových typů. Kód JavaScriptu je možné vložit rovnou do dokumentu HTML [27] [59].

4.3 Knihovny pro vývoj webových aplikací

V dnešní době existuje mnoho knihoven, které usnadňují vývoj webových stránek a aplikací. V této kapitole jsou popsány knihovny, které se využívají v této práci – jQuery a Bootstrap. Při vývoji složitějších webových aplikací se v dnešní době nejčastěji používají knihovny jako je React, Angular a Vue.js.

Bootstrap

Jedná se o nástroj pro jednoduchý a rychlý vývoj frontendu webových aplikací, který poskytuje všestranné funkce. Jeho používání je velice intuitivní a s jeho pomocí lze vytvořit moderní uživatelské rozhraní. Pro použití tohoto frameworku jsou nutné základní znalosti HTML a CSS. Pomocí Bootstrapu je možné vytvořit komplexní uživatelské rozhraní webové stránky založené na standardním HTML. Kromě toho Bootstrap poskytuje další komponenty, jako jsou například carousel[4], tlačítka, popupy a další. Tento nástroj je také kompatibilní se všemi moderními prohlížeči a podporuje responzivní design [47]. Responzivní design má dynamické rozvržení a velikost komponentů se mění v závislosti na velikosti obrazovky zařízení.

[4]Carousel je prvek webové stránky, který postupně na jednom místě zobrazuje obsah v podobě slideshow. Nejčastěji se jedná o obrázky.

Bootstrap byl vytvořen Markem Ottem a Jacobem Thortonem ve společnosti Twitter. V současné době se jedná o open-source projekt na GitHubu. Je to často používaný nástroj pro tvorbu webových stránek i aplikací, zejména díky tomu, že šetří při vývoji čas. Dále má Boostrap také velmi dobrou dokumentaci všech svých komponent [38].

jQuery

Jedná se o malou open-source knihovnu, která je určená pro jazyk JavaScript. Díky této knihovně je možné výrazně zjednodušit některé úkoly a operace, které by byly za použití čistého JavaScriptu velmi obtížné. Tato knihovna umožňuje jednoduchou interakci mezi DOM[5] (objektovým modelem dokumentu) a JavaScriptem. Mezi její funkce patří zjednodušení výběru elementů, procházení HTML dokumentu, zpracování událostí prohlížeče a reakce na ně, vytváření animací a efektů a interakce použitím technologie AJAX[6]. Dále umožňuje celkové zjednodušení programování skriptů v jazyce JavaScript tak, aby fungovaly ve všech moderních prohlížečích a vývojář nemusel řešit problémy s rozdílností prohlížečů. JQuery je vytvořená tak, aby bylo možné řetězení metod, což umožňuje zavolat libovolné množství metod najednou na vybraný element. Díky tomu není nutné opakovaně procházet DOM, což výrazně zvyšuje efektivitu [34].

4.4 Komunikace s webovou stránkou

Komunikace s webovou stránkou může probíhat různými způsoby. Jedná se o komunikaci mezi klientem a serverem. Základním protokolem pro webovou komunikaci je protokol HTTP. Další technologie jsou například protokol WebSocket nebo WebRTC. Tyto technologie umožňují komunikaci v reálném čase.

Protokol HTTP

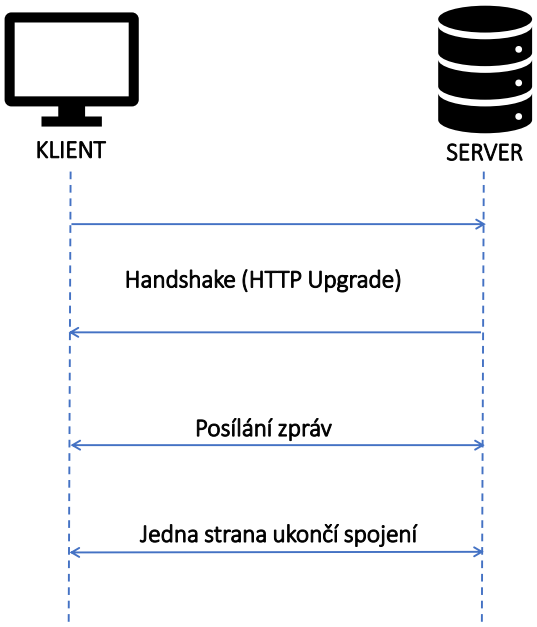
Jedná se o protokol určený k přenosu hypertextových dokumentů ve formátu HTML. Tento protokol je základem veškeré výměny dat na webu. Jedná se o protokol typu klient-server, což znamená, že komunikaci začíná klient (příjemce dat), kterým je obvykle webový prohlížeč. Komunikace funguje tak, že si klient a server mezi sebou vyměňují zprávy. Zprávy odeslané webovým prohlížečem, se nazývají požadavky (requests) a server následně posílá odpovědi (responses).

Protokol WebSocket

Jedná se o rozhraní, které poskytuje oboustranný komunikační kanál přes jedno spojení. WebSocket pomáhá ke snížení síťového provozu a latence. Rozhraní WebSocket umožňuje komunikaci v reálném čase. Aby bylo možné navázat spojení pomocí protokolu WebSocket, musí nejprve sever a klient provést tzv. „handshake“. Klient pošle HTTP request na server s žádostí o navázání WebSocket spojení, na který server odpovídá také pomocí protokolu HTTP. Po navázání spojení si obě strany mohou posílat zprávy v obousměrném (duplexním) režimu přes komunikační kanál. To znamená, že obě strany mohou posílat zprávy zároveň. Tento kanál zůstává aktivní do té doby, dokud ho některá ze stran neuzavře. Protokol

[5] DOM – Document Object Model
[6] AJAX – Asynchronous JavaScript and XML

WebSocket je podporován ve všech rozšířených webových prohlížečích [39]. Na následujícím obrázku 4.2 je znázorněn průběh komunikace prostřednictvím tohoto protokolu.



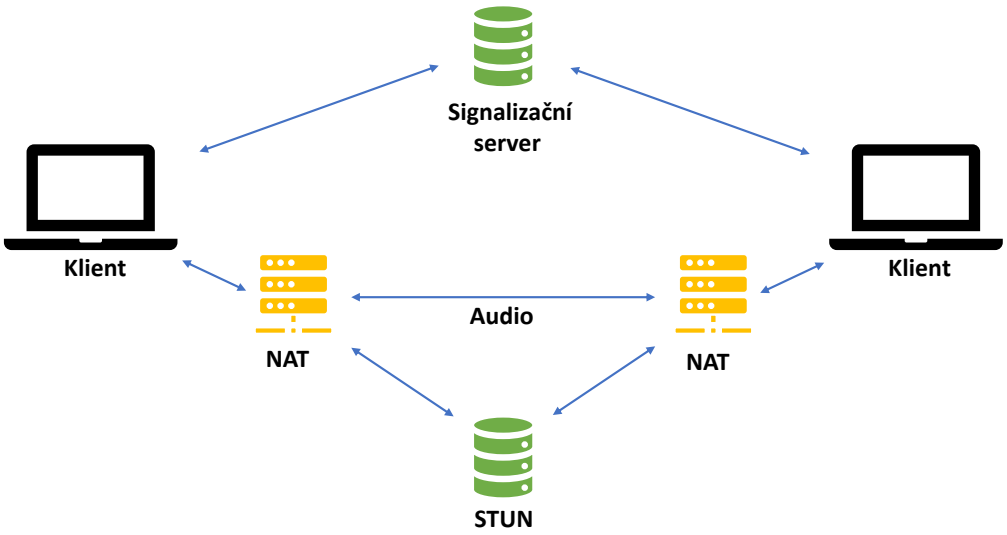
Obrázek 4.2: Diagram popisující komunikaci pomocí WebSocket

WebRTC

WebRTC[7] je technologie, která zajišťuje komunikaci v reálném čase. Jedná se o peer-to-peer[8] komunikaci, která je možná bez využití speciálních rozhraní, instalace modulů nebo dalšího software. Jedná se o technologii, která umožňuje webovým aplikacím a webovým stránkám zachycovat a streamovat audio, video nebo si vyměňovat jakákoli data. WebRTC se používá převážně pro videochaty a schůzky na platformách pro videohovory [48].

Pro zahájení spojení je nutný signalizační server (signaling server). Ten slouží k výměně informací před zahájením spojení. Klient musí nejprve vytvořit nabídku spojení (offer), která obsahuje SDP[9] objekt, což je protokol popisující multimediální komunikaci, který klient odešle na signalizační server. Druhý klient odpoví vytvořením odpovědi (answer). Tento proces je nutný pro výměnu informací o spojení. Mezi tyto informace patří například druh odesílaného média, jeho formátu používaném přenosovém protokolu, IP adrese a portu koncového bodu a dalších. Tato výměna je řešena pomocí Interactive Connectivity Establishment (ICE) protokolu, který umožňuje dvěma zařízením navázat peer-to-peer spojení, i když se IP adresy zařízení mění díky použití překladu síťových adres (NAT[10]) [41]. Tento proces je znázorněn na obrázku 4.3.

[7] WebRTC – Web Real-Time Communication
[8] Termín peer-to-peer označuje přímou komunikaci mezi klienty
[9] SDP – Session Description Protocol
[10] NAT – Network address translation



Obrázek 4.3: Ukázka všech částí komunikace za pomocí WebRTC

AIORTC

AIORTC je knihovna v jazyce Python, která implementuje Web Real-Time Communication (WebRTC) a Object Real-Time Communication (ORTC). Tato knihovna se používá k vývoji aplikací využívajících uvedené technologie. AIORTC umožňuje pohodlnější práci s WebRTC, díky tomu, že je jeho rozhraní poměrně jednoduché a čitelné [37].

Kapitola 5

Technologie na straně serveru

Tato kapitola se zabývá serverovou částí aplikace. Nejprve je zde popsán jazyk Python, který je využit pro implementaci této části. Kapitola dále shrnuje základní informace o zasílání dat pomocí Message brokeru a o jeho implementaci pomocí RabbitMQ. Je zde popsáno, jak tato komunikace probíhá a z jakých částí se skládá. Poté je zde také zmíněno rozšíření Web STOMP, které je v této práci použito pro posílání dat ze serveru do webového prohlížeče. Poslední část kapitoly se věnuje sytému rozpoznávání řeči.

5.1 Programovací jazyk Python

Python je univerzální, vysokoúrovňový objektově orientovaný jazyk. Vysokoúrovňovým jazykem je myšleno, že programovací jazyk používá vysokou míru abstrakce. Python bývá někdy označován také jako skriptovací jazyk.

Python má velmi jednoduchý a čitelný syntax. Zaměřuje se na to, aby byl kód znovu použitelný a udržitelný. Kód v tomto programovacím jazyce je často výrazně kratší, než kdyby byl napsán v jiném jazyce, jako je C, C++ nebo Java. Díky tomu je psaní kódu výrazně rychlejší a program se dá snadno testovat a ladit. Výraznou výhodou Pythonu je jeho jednoduchost z hlediska učení.

Většina programů v Pythonu je spustitelná na počítačích s různými operačními systémy. Python obsahuje velké množství funkcí a standardních knihoven, které umožňují snadné řešení úloh z různých oblastí. Python může být také rozšířen o vlastní knihovny a je možné dohledat velké množství veřejně dostupných knihoven. Nevýhodou tohoto jazyka je, že jeho rychlost provedení programu je o něco horší v porovnání s nízkoúrovňovými jazyky.

V dnešní době se Python využívá v mnoha oblastech a je možné ho použít například pro vytváření grafického uživatelského rozhraní, analýzu dat, vytváření skriptů, vývoj webových aplikací, databázové programování a v mnoha dalších oblastech. [40]

5.2 Systémy pro zasílání zpráv

Sytém pro zasílání zpráv (message broker) umožňuje vzájemnou komunikaci a výměnu informací mezi aplikacemi. Pro zasílání zpráv jsou využívány různé protokoly. Díky tomu mezi sebou mohou komunikovat aplikace, které jsou napsány v různých jazycích a běží na odlišných strojích. Message brokery mohou ověřovat, ukládat, směrovat a doručovat zprávy příslušným příjemcům. Jejich velkou výhodou je, že umožňují odesílateli posílání dat bez

nutnosti vědět, jestli data získá jeden nebo více příjemců a jestli je příjemce aktivní. Díky tomu je možné oddělení procesů a služeb v rámci systémů.

Aby bylo možné zajistit spolehlivou komunikaci a ukládání zpráv, bývá často součástí tohoto softwaru fronta zpráv (message queue), která uchovává zprávy do té doby, než je může cílová aplikace přijmout. Zprávy jsou zde uloženy ve stejném pořadí, v jakém byly odeslány.

Message brokery umožňují využití asynchronní komunikace. Tento typ komunikace zajišťuje, že zpráva bude vždy doručena a dorazí ve správném pořadí vzhledem k ostatním zprávám, a to i v případě občasných problémů se síťovou komunikací a dobou odezvy, které jsou typické pro veřejné sítě [28].

RabbitMQ

RabbitMQ[1] je jedna z nejrozšířenějších implementací message brokeru. RabbitMQ fronta přebírá zprávy od odesílatele a posílá je příjemci. Dále zajišťuje dočasné úložiště dat a zabraňuje jejich ztrátě. RabbitMQ implementuje komunikační protokol AMQP (Advanced Message Queuing Protocol). Dále umožňuje mnohá rozšíření, jako například podporu pro protokoly MQ Telemetry Transport (MQTT), Streaming Text Oriented Messaging Protocol (STOMP) a další. Jedná se o spolehlivý a vysoce škálovatelný software. RabbitMQ podporuje mnoho programovacích jazyků a umožňuje použití s různými operačními systémy. [29]

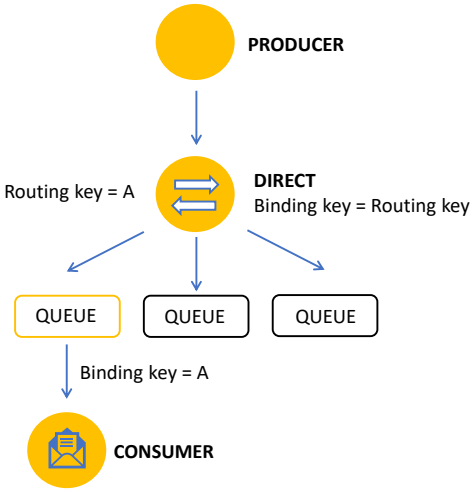
Základní pojmy: [31]

- **Producer (odesílatel):** Aplikace, která odesílá zprávy do fronty na základě jména fronty.
- **Queue (fronta):** Datová struktura, ve které se uchovávají zprávy a posílají se přes ni od odesílatele k příjemci. Zprávy jsou zde uchovány do té doby, než je může příjemce zpracovat. Fronta je typu FIFO (First in First out), ale RabbitMQ nabízí také další funkce fronty, jako jsou priority a opětovné řazení do fronty.
- **Consumer (příjemce):** Consumer se přihlašuje k odběru zpráv z fronty.
- **Exchange:** Exchange přebírá zprávu od odesilatele a předává ji všem odpovídajícím frontám. Exchange je zodpovědná za správné směrovaní.
- **Binding:** Vazba mezi frontou a exchange, která je určená pomocí klíče (binding key).
- **Routing key:** Klíč, podle kterého se exchange rozhoduje, do jaké fronty zprávu směrovat.

Exchange zajišťuje posílání zpráv správným frontám na základě routing key, binding key a dalších parametrů. To, jak se bude fronta vybírat,je určeno pomocí typu exchange. Mezi tyto typy patří: [32]

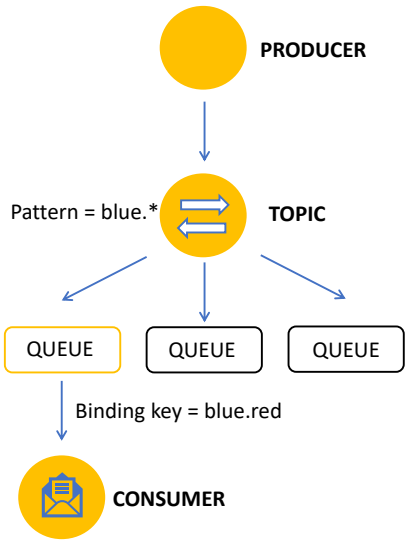
- **Direct exchange:** Tento typ exchange posílá zprávy do front na základě routing key dané zprávy. Routing key je atribut zprávy, který do její hlavičky přidává odesílatel. Zpráva je odeslána do fronty, která má přesně stejný binding key, jako je routing key zprávy. Příklad posílání zpráv přes direct exchange je na obrázku 5.1

[1] https://www.rabbitmq.com/



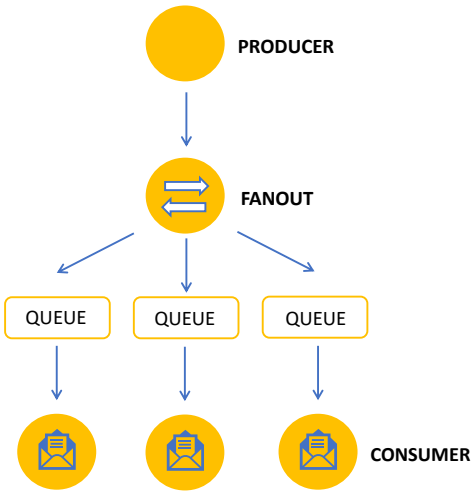
Obrázek 5.1: Schéma posílání zpráv přes direct exchange

- **Default exchange:** Jedná se o specifický typ direct exchange, hlavním rozdílem je to, že exchange porovnává routing key s názvem fronty.
- **Topic exchange:** Směřuje zprávy na základě částečné nebo celkové shody mezi routing key a routing pattern. Routing pattern vytváří příjemce. Routing key se skládá ze seznamu slov, která jsou oddělená tečkou. Routing pattern může obsahovat znak `*`, který nahrazuje slovo v routing key. Dále může obsahovat také znak `#`, který nahrazuje 0 – N slov. Konkrétní příklad můžete vidět v obrázku 5.2



Obrázek 5.2: Schéma posílání zpráv přes topic exchange

- **Fanout exchange:** Tento typ exchange posílá zprávu do všech připojených front bez ohledu na routing key. Názorný příklad je vykreslen na obrázku 5.3

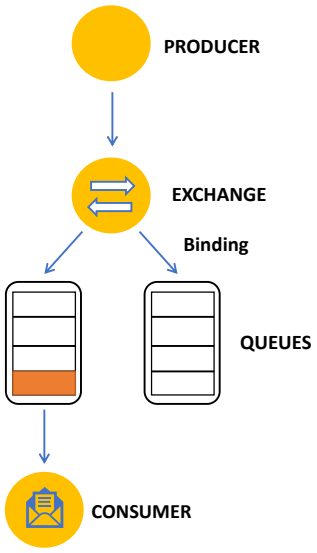


Obrázek 5.3: Schéma posílání zpráv přes fanout exchange

- **Headers exchange:** Headers exchange směřuje zprávy na základě argumentů obsahujících hlavičky a volitelné hodnoty. Exchange předává zprávu, pokud se hodnoty hlaviček shodují s hlavičkami uvedenými příjemcem při připojení k frontě. Binding může obsahovat speciální argument „x-match“, který udává, jestli se musí shodovat všechny hlavičky nebo jen jedna. Může nabývat hodnot „any“ nebo „all“. V případě „all“ se musí všechny hlavičky rovnat a v případě „any“ je potřeba, aby se shodovala alespoň jedna hlavička.

Průběh komunikace [31]:

1. Odesílatel nejprve pošle zprávu do exchange. Při vytváření exchange musí být specifikován typ.
2. Po přijetí zprávy je exchange zodpovědná za správné směrování zpráv do front. Výběr správných front závisí na různých atributech podle toho, o jaký typ exchange se jedná.
3. Exchange odešle zprávu příslušným frontám.
4. Zpráva zůstává ve frontě do té doby, než ji zpracuje příjemce.
5. Příjemce zpracuje zprávu.



Obrázek 5.4: Schéma průběhu posílání zpráv přes RabbitMQ

RabbitMQ Web STOMP Plugin

STOMP (Streaming Text Oriented Messaging Protocol) je jednoduchý textový protokol pro zasílání zpráv. STOMP nabízí širokou interoperabilitu v zasílání zpráv napříč různými jazyky, platformami a brokery. To znamená, že klienti protokolu STOMP mohou komunikovat s jakýmkoliv STOMP message brokerem [52]. Web STOMP rozšíření pro RabbitMQ umožňuje použití STOMP protokolu při komunikaci s webovou aplikací. RabbitMQ Web STOMP plugin poskytuje propojení mezi implementací STOMP protokolu v pluginu RabbitMQ STOMP a klienty protokolu WebSocket [44].

5.3 Systémy pro rozpoznání řeči v letectví

Komunikace mezi ATCO a piloty probíhá převážně prostřednictvím rádiové komunikace. Řídící letového provozu vyslovuje verbální příkazy pro pilota letadla. Příkazy, které ovlivňují pohyb letadla, např. příkazy pro změnu výšky, rychlosti nebo směru, musí pilot zopakovat. Jedením ze zásadních úkolů ATCO je vyhodnocovat opakování příkazu (readback) a reagovat na případné chyby. Pokud si řídící letového provozu chyb při opakování příkazu nevšimne, může to mít dramatický dopad na bezpečnost leteckého provozu. Automatický přepis komunikace řízení letového provozu přispěje ke zvýšení bezpečnosti letového provozu, zlepšení výcviku řídících letového provozu a snížení jejich pracovní zátěže [22].

Projekt HAAWAII

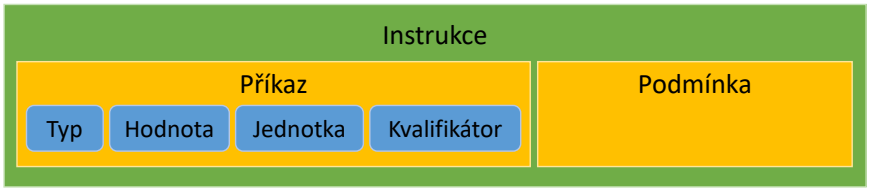
Cílem projektu HAAWAII je vývoj spolehlivého řešení pro automatický přepis hlasové komunikace mezi ATCO a pilotem. Projekt využívá strojového učení a rozsáhlého souboru řečových dat k vytvoření nových modelů pro rozpoznávání řeči. Tento soubor obsahuje nahrávky reálné komunikace mezi ATCO a pilotem [21].

Kromě přepisu komunikace do textové podoby tento projekt také provádí extrakci příkazů, což může být interpretováno jako překlad sekvencí slov do tvaru srozumitelného pro počítač. Tato extrakce je základem pro implementaci detekce chyb při opakování příkazů pilotem [22].

Ontologie extrakce příkazů

V kontextu počítačových a informačních věd definuje ontologie soubor reprezentačních klíčových pojmů v určité oblasti. Tyto klíčová slova jsou typicky třídy (nebo množiny), atributy (nebo vlastnosti) a vztahy (nebo vztahy mezi členy tříd). Definice klíčových slov popisují sémantický význam jednotlivých pojmů, jejich vlastnosti a vztahy mezi nimi [20].

Hlavními prvky ontologie extrakce příkazů jsou volací znak a instrukce. Obrázek 5.5 ukazuje, že instrukce se vždy skládá z povinného příkazu a jedné nebo více volitelných podmínek. Každý příkaz obsahuje typ, jednu nebo více hodnot a odpovídající jednotky (např. `FL`, `ft` nebo `None`). Poté následuje volitelný kvalifikátor (např. `LEFT`, `RIGHT`, `OR_LESS`, `BELOW`) [23].



Obrázek 5.5: Struktura extrahované instrukce řídícího letového provozu

Kapitola 6

Analýza současných řešení v oblasti letového provozu

Tato kapitola analyzuje existující řešení v oblasti letového provozu. Nejprve jsou zde rozebrána některá řešení pro sledování letového provozu a dále jsou zde popsané systémy a programy pro trénink pracovníků řízení letového provozu.

6.1 Systémy pro sledovaní letového provozu

V dnešní době existuje několik webových stránek, na kterých mohou uživatelé sledovat aktuální letový provoz (tzv. flight trackers). Tyto stránky často využívají lidé, kteří se zajímají o letadla, nebo cestující, kteří chtějí zjistit aktuální informace o jejich letadle. Mezi nejznámější stránky tohoto typu patří Flightradar24 nebo FlightAwarer. Tyto stránky poskytují aktuální informace o poloze letadel, ale také mnoho dalších informací, jako je například rychlost letadla, cílové letiště nebo výška. Tato data často získávají z mnoha zdrojů.

Flightradar24

Flightradar24 je veřejně dostupná stránka pro zobrazení polohy letadel. Tato stránka zobrazuje aktuální letový provoz z celého světa. Data o letadlech jsou získána ze zdrojů, jako jsou radary, ADS-B[1] nebo MLAT[2]. Tato data jsou pak propojena s letovým řádem od jednotlivých leteckých společností a letišť.

Hlavní technologií, ze které získává Radar24, data je ADS-B. Letadlo získává data o své poloe ze satelitu. Transpondér ADS-B v letadle vysílá signál obsahující informace o poloze letadla. Tento signál zachycuje přijímač a posílá data do Flightradar24 [16].

OpenSky Network

OpenSky Network je nezisková asociace se sídlem ve Švýcarsku, která vznikla v roce 2012. Cílem této asociace je zlepšit bezpečnost, spolehlivost a efektivitu využívání vzdušného prostoru. OpenSky Network poskytuje veřejnosti přístup k reálným datům z letového provozu. Všechna shromážděná data jsou archivována do databáze a slouží výzkumníkům z různých

[1] ADS-B – Automatic dependent surveillance-broadcast
[2] MLAT – multilaterace, zjištění polohy letadla podle rozdílu času příchodu signálů z několika bodů

oblastí k analýze, ve snaze zlepšit fungování řízení letového provozu. Hlavní technologií pro získání dat je stejně jako u FlightRadar24 ADS-B [54].

OpenSky Network poskytuje rozhraní API, které umožňuje získávat aktuální informace o vzdušném prostoru pro výzkumné a nekomerční účely. Data jsou poskytována ve formě stavových vektorů (State Vectors). Stav označuje souhrn všech informací o konkrétním letadle v určitém časovém okamžiku, jako je pozice, rychlost a identifikace. OpenSky Network API je použito pro získávání dat v tomto projektu.

6.2 Systémy pro výcvik řídících letového provozu

Správný výcvik řídících letového provozu je nezbytný pro zajištění bezpečnosti letového provozu. Simulátory jsou velmi užitečným nástrojem pro řídící letového provozu a mohou si díky nim nacvičit chování v různých situacích. V dnešní době existuje několik veřejně dostupných simulátorů jako je například:

- **ATC-SIM:** Online simulátor, který je veřejně dostupný a zdarma. ATC-SIM umožňuje vyzkoušet si řídit letový provoz na různých letištích po celém světě. Uživatel si také může nastavit frekvenci změn směru větru a obtížnost simulace.

  Nejedná se o tak realistický simulátor jako některé jiné, které budou dále zmíněny, ale jeho velkou výhodou je to, že je veřejně dostupný, a pro použití není nutné instalovat žádný software [4].

- **ATC Simulator – CS Soft:** Jedná se o simulátor letového provozu, který byl vyvinut společností CS Soft a umožňuje uživatelům simulovat práci řídícího letového provozu. Jedná se o placený software. Tento simulátor přesně kopíruje fungování pracoviště ATC. Letecký provoz je simulován s velkou přesností. Uživatel může dynamicky měnit hustotu provozu, meteorologická data a mnoho dalšího. Simulátor také poskytuje možnost nahrávání a umožňuje cvičení s různou obtížností [9].

Tyto veřejně dostupné simulátory poskytují pouze omezené množství funkcí a nejsou dostatečně realistické. Dále existuje mnoho profesionálních simulátorů, které jsou vybaveny pokročilými technologiemi a využívají pokročilé fyzikální modely letadel a okolního prostředí. Pracovníci řízení letového provozu na nich mohou trénovat mnoho různých situací, včetně nouzových scénářů, změn počasí a simulovaných poruch letadel.

- **ANSART simulátor:** Tento simulátor byl vyvinut společností ANSART. Jedná se o velmi komplexní a flexibilní nástroj, který nabízí uživatelům široké spektrum funkcí a možností pro výcvik a školení v oblasti letového provozu. ANSART simulátor není veřejně dostupný.

  Tento nástroj není určen jen pro výcvikové účely, ale lze jej využít i pro výzkumné činnosti, například pro revizi nových metod a postupů řízení letového provozu. Je vhodný pro základní i pokročilý výcvik ve všech oblastech řízení letového provozu. Simulátor umožňuje také 3D vizualizaci [3].

- **ESCAPE:** ESCAPE je vyspělý simulátor organizace Eurocontrol, který simuluje provoz na různých evropských letištích. Simulátor slouží ke školení pracovníků, k výzkumu a vývoji, vyhodnocení nových operačních konceptů a nástrojů a předprovozní validaci. Při školení poskytuje řídícím letového provozu možnost trénovat a vylepšovat své dovednosti v oblasti letového provozu. Jedná se o velmi realistický simulátor.

ESCAPE poskytuje také vlastní analytické nástroje, které umožňují rychle zpracovat data ze cvičení a vytvářet zprávy typu dashboard (grafy), které lze poté použít pro procházení a rozbor událostí, jež nastaly během simulace. Jedná se o uzavřený systém, který je dostupný jen pro profesionální výcvik [13].

Mezi další simulátory letového provozu, patří například Adacel MaxSim, ATCoach nebo Micro Nav BEST. Žádné z těchto řešení není však veřejně dostupné a používají se pro profesionální výcvik řídících letového provozu.

Kapitola 7

Návrh řešení systému

Tato kapitola se zabývá návrhem řešení systému. Nejprve je zde popsáno, jak probíhal výběr způsobu implementace, přesněji rozhodnutí mezi desktopovou a webovou aplikací. Poté je v této kapitole stručný popis funkcí aplikace. Samotný návrh je rozdělen do dvou částí – návrh webové aplikace a návrh serverové části.

7.1 Výběr mezi webovou a desktopovou aplikací

Při návrhu bylo třeba porovnat různé způsoby implementace aplikace a rozhodnout se, jaké nástroje budou při vývoji použity. Jednou z klíčových částí bylo rozhodnutí, zda se bude jednat o webovou nebo desktopovou aplikaci. Obě tyto možnosti mají své výhody i nevýhody.

Desktopové aplikace

Desktopová aplikace je program, který je nutné nainstalovat na pevný disk počítače, poté může fungovat bez nutnosti připojení k internetu. Výhodou desktopových aplikací je jejich dostupnost. Aplikaci si uživatel sice musí nejprve stáhnout a nainstalovat do počítače, nejčastěji pomocí internetu, ale při dalším použití už není internet potřeba. Nevýhodou desktopové aplikace je závislost na konkrétním zařízení. Další nevýhodou je potřeba ručně instalovat aktualizace a nové verze softwaru.

Webové aplikace

Webové aplikace jsou dostupné prostřednictvím webových prohlížečů. Namísto uložení aplikačních souborů do vlastního počítače, jsou tyto soubory umístěny na vzdálený server. Webový prohlížeč načítá obsah stránky a veškerý spustitelný kód. Na rozdíl od jednoduché statické webové stránky jsou webové aplikace interaktivní a poskytují mnoho funkcí.

Mezi výhody webových aplikací patří jejich jednoduché spuštění bez potřeby instalace. Pro použití aplikace stačí zadat správnou adresu (URL), je tedy nutné připojení k internetu. Aplikace nezabírá místo na pevném disku počítače, aktualizace jsou automatické a při každém spuštění vidí uživatel vždy poslední verzi. Webové aplikace jsou nezávislé na operačním systému počítače. Dají se spustit z jakéhokoliv zařízení s webovým prohlížečem [10].

Výběr typu aplikace

Tyto informace byly zhodnoceny a nejdříve byla vybrána desktopová aplikace s vytvářením mapy pomocí knihovny Folium. Ta ale vytváří mapu ve formě HTML stránky (za pomocí knihovny Leaflet), zobrazené v grafickém prostředí jazyka Python. Díky tomu by bylo problematické vyřešit ovládání a aktualizaci informací na mapě. Z tohoto důvodu byla nakonec vybrána implementace aplikace jako webové stránky se zobrazováním mapy pomocí knihovny Leaflet.

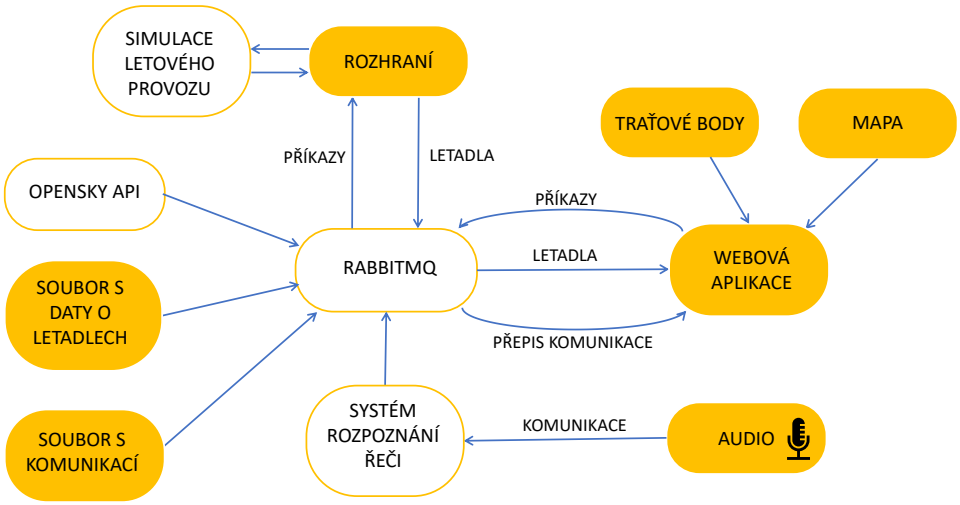
7.2 Popis aplikace

Aplikace pracuje ve dvou režimech:

- V prvním režimu je aplikaci možné využít jako flight tracker zobrazující reálná data a zkoumat aktuální stav letového provozu.
- Ve druhém režimu se zobrazuje simulovaný letecký provoz, který je možné ovládat pomocí příkazů.

Prostřednictvím aplikace spolu mohou komunikovat dva uživatelé – jeden v roli řídícího letového provozu a druhý v roli pilota letadla. Komunikace se přehledně vypisuje na obrazovce a je možné si zobrazit i její historii. Přepis komunikace na text a extrakci příkazů zajišťuje systém pro rozpoznání řeči. Simulaci letového provozu je možné ovládat buď automaticky, na základě extrahovaných příkazů v přepisu komunikace, nebo manuálním zadáním příkazů do aplikace.

Celá aplikace se skládá ze serverové části a uživatelského rozhraní. Serverová část zajišťuje načítání informací o letadlech, simulaci letového provozu, zpracování komunikace a propojení se systémem rozpoznání řeči. Uživatelské rozhraní je vytvořeno ve formě webové aplikace. Na obrázku 7.1 je blokové schéma, které ukazuje vzájemné vazby jednotlivých částí systému.



Obrázek 7.1: Blokové schéma systému zobrazující propojení jednotlivých částí

7.3 Návrh webové aplikace

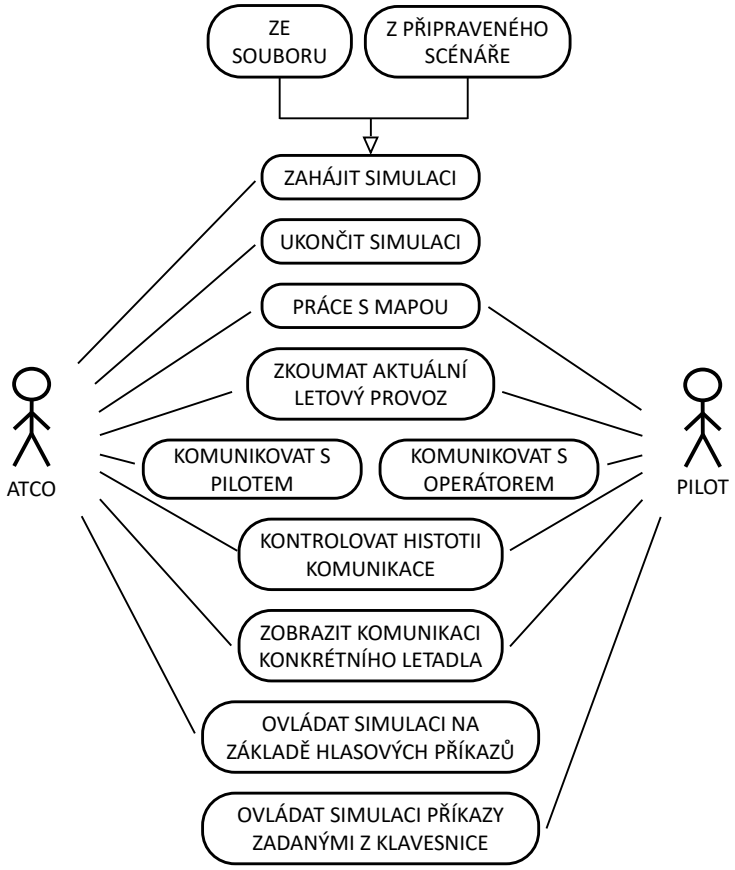
V této kapitole jsou popsány důležité části návrhu webové aplikace, jako je popis cílové skupiny, případů užití a návrh uživatelského rozhraní. Uživatelské rozhraní bylo navrženo tak, aby se podobalo radarovým obrazovkám, které používají řídící letového provozu při své práci.

Cílová skupina

Cílovou skupinou aplikace jsou začínající řídící letového provozu, kteří nemají žádný volně dostupný nástroj pro výcvik. Pomocí této aplikace se mohou učit jak komunikovat s pilotem letadla. Další cílovou skupinou jsou pracovníci z výzkumných skupin KnoT a Speech na Fakultě informačních technologií VUT v Brně, kteří mohou tento systém využít pro demonstraci výsledků své práce. Tito pracovníci potřebují nástroj, díky kterému by bylo možné zobrazit přepis komunikace mezi pilotem a ATCO ze systému pro rozpoznání řeči.

Případy užití

Primárním uživatelem aplikace je ATCO, ale může ji využívat také uživatel v roli pilota, který jejím prostřednictvím komunikuje s ATCO.



Obrázek 7.2: Diagram případů užití aplikace

Po startu aplikace se zobrazí mapa světa s vyznačenou polohou letadel a směru jejich letu. Mapa je dynamická, takže s ní uživatel může pohybovat, přibližovat ji nebo oddalovat. Uživatel může vybrat konkrétní letadlo, které se zvýrazní, a zobrazí se tabulka se základními informacemi o tomto letadle a historie jeho komunikace. Uživatel si může také zobrazit historii veškeré komunikace.

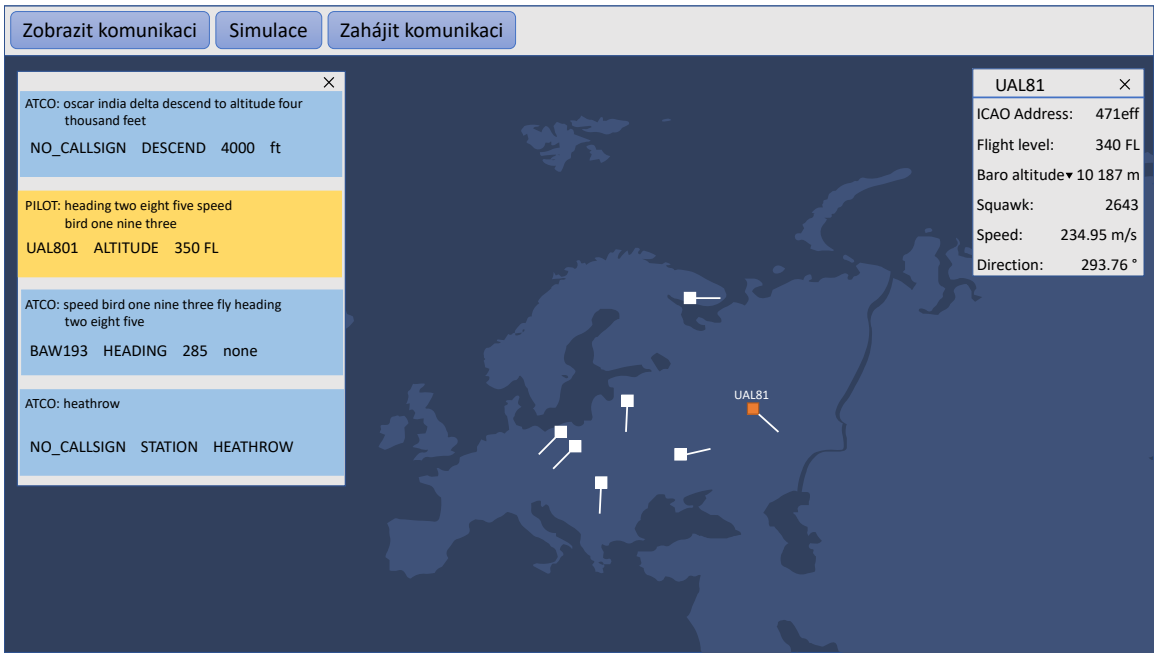
Uživatel může zkoumat aktuální letový provoz nebo může zahájit simulaci. Simulace se zahajuje vložením dat ze souboru, nebo spuštěním jednoho z připravených scénářů. Následně může uživatel s letadly komunikovat a dávat jim příkazy pomocí hlasové komunikace. Hlasové příkazy mohou být automaticky předávány simulaci, která je provádí, nebo tyto příkazy manuálně zadává uživatel v roli pilota. Simulaci lze ukončit stiskem tlačítka „Ukončit simulaci“. Všechny případy užití jsou zobrazeny na obrázku 7.2.

Návrh uživatelského rozhraní

- **Mapa:** Mapa světa musí být jednoduchá a nesmí obsahovat rušivé elementy, jako například názvy měst. Měla by být v tmavých barvách a zobrazovat jen obrysy států, jako je to běžně u radarových obrazovek řídících letového provozu.
- **Zobrazení letadel:** Letadla jsou na mapě zobrazena jako jednoduché čtvercové ikony, ke kterým je připojena krátká rovná čára znázorňující směr letu. Standardně budou mít všechna letadla stejnou barvu, pouze letadlo, o kterém probíhá aktuální komunikace, a letadla o kterých v poslední zprávě nepřišly žádné informace budou barevně odlišena. Bude také možné definovat oblasti určené zeměpisnými souřadnicemi a letovou hladinou. Letadla nacházející se v takových oblastech mohou mít také odlišnou barvu.
- **Výpis komunikace:** Komunikace mezi pilotem a ATCO bude přehledně vypsána do bočního panelu, který se zobrazí po kliknutí na tlačítko. Každá zpráva bude vypsána do samostatného bloku, který musí obsahovat všechny důležité informace. Vzhled výpisu komunikace je inspirován ukázkou z projektu Haawaii[1]. Hlavička bočního panelu bude označovat, zda se jedná o výpis celkové komunikace nebo o výpis komunikace konkrétního letadla. Výpis komunikace konkrétního letadla se zobrazí po kliknutí myši na ikonu letadla na mapě.
- **Výpis informací o letadle:** Po kliknutí na konkrétní letadlo se zobrazí základní informace, jako je volací znak, adresa letadla, výška, squawk kód, rychlost a směr. Tyto informace budou zobrazeny v samostatné tabulce v pravé části obrazovky.
- **Simulace:** Ovládání simulace bude možné pomocí menu v horní části stránky. Menu bude obsahovat následující položky:
  - Zahájení simulace na základě vložení dat ze souboru, nebo vybráním jednoho z předem připravených scénářů
  - Ruční zadávání jednotlivých příkazů pro simulaci
  - Ukončení simulace
  - Nastavení automatického provádění příkazů od ATCO

[1] https://www.haawaii.de/wp/

Pokud bude nastaveno automatického provádění příkazů, budou se příkazy obsažené v komunikaci ze strany ATCO automaticky vyhodnocovat a posílat do simulace. Seznam příkazů ATCO, pro které je automatické provádění podporováno, se nachází v příloze C. Uživatel bude mít také možnost zadat příkaz pro simulaci ručně nebo načíst jeden, či více příkazů ze souboru. Příkazy se budou zadávat v textovém formátu definovaném knihovnou BlueSky.[2]



Obrázek 7.3: Návrh hlavní stránky aplikace s mapu světa, na které se zobrazují letadla. V levé části je výpis komunikace mezi pilotem a ATCO, v pravé částí se zobrazují informace o vybraném letadle.

7.4 Návrh serverové části

Serverovou část tvoří několik programů:

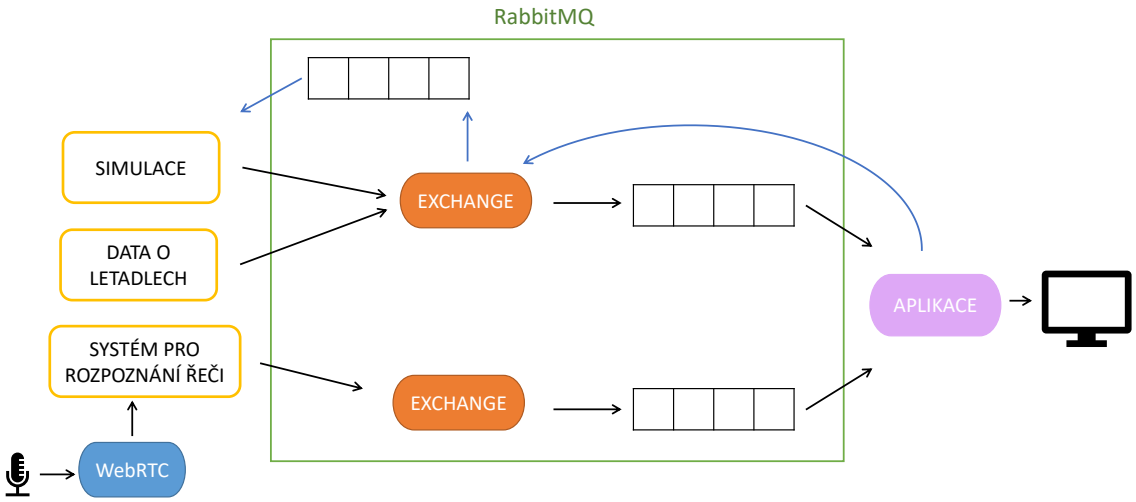
- Server, který zajišťuje spuštění webové aplikace, přenos a zpracování hlasových dat a jejich předání do systému rozpoznávání řeči.
- Program pro simulaci letového provozu postavený na knihovně BlueSky.
- Program pro získávání reálných dat o letadlech prostřednictvím OpenSky API.

Pro komunikaci webové aplikace s ostatními částmi systému je použit nástroj RabbitMQ. Data jsou posílána prostřednictvím tří front zpráv:

- První fronta slouží pro posílaní aktuálních údajů o reálných letadlech, nebo informací o letadlech ze simulace. Reálná data o letadlech jsou získávána přes OpenSky API, které poskytuje aktuální informace o letové provozu po celém světě. Simulace letového provozu vzniká pomocí knihovny BlueSky, která průběžně posílá do aplikace výstup z jednotlivých kroků simulace.

[2] https://github.com/TUDelft-CNS-ATM/bluesky/wiki/Command-Reference

- Druhá fronta umožňuje ovládání simulace pomocí příkazů z aplikace. Tyto příkazy mohou být vytvářeny automaticky na základě hlasových zpráv, nebo je uživatel zadává ručně.
- Poslední fronta složí k přenosu komunikace mezi piloty letadel a ATCO. Webová aplikace umožňuje nahrávat hlasovou komunikaci, která se posílá do systému pro rozpoznání řeči. Ten převádí komunikaci na strukturovaná data formátu JSON, která se pak posílají do aplikace pomocí této RabbitMQ fronty.



Obrázek 7.4: Návrh komunikace mezi serverovou částí a webovou aplikací pomocí nástroje RabbitMQ.

7.5 Návrh formátu pro posílání dat o letadlech

Aplikace přijímá informace o letadlech ve formátu JSON. Nejdůležitější informací pro vykreslení na mapu je poloha letadla určená zeměpisnými souřadnicemi. Dále je také důležitá jednoznačná identifikace letadla, která je zajištěna pomocí volacího znaku a adresy ICAO24. Formát dále obsahuje informace výšce letadla, směru letu a rychlosti. Podrobné schéma formátu se nachází v příloze B.

7.6 Použité technologie

V této kapitole se nachází souhrn použitých technologií a knihoven, který je rozdělen na část klientskou a serverovou.

- **Klientská část:** Parcel, Bootstrap, jQuery, webstomp-client, Leaflet
- **Serverová část:** Pika, BlueSky, OpenSkyAPI, aiohttp, aiortc

Kapitola 8

Implementace systému

V této kapitole je popsána implementace aplikace. Kapitola se nejdříve věnuje inicializaci prostředí a konfiguraci, poté je zde popsána implementace webové aplikace. Následuje popis implementace serverové strany a způsob zasílání dat ze serveru do webové aplikace. Dále je rozebrána implementace simulace, zachycení a posílání hlasových zpráv a nástroj na konverzi protokolu ASTERIX.

8.1 Inicializace prostředí, vytvoření projektu

Pro správu knihoven v jazyce Python je použit nástroj pro správu balíčků a vývojového prostředí Conda. Conda umožňuje rychlou instalaci a aktualizaci balíčků, včetně všech závislostí. Dále nabízí snadné vytvoření různých vývojových prostředí a přepínání mezi nimi [8]. Pro účely tohoto projektu byl použit nástroj Conda, který je součástí softwaru Miniconda[1]. Definice prostředí se nachází v souboru `python_env.yml`.

Pro sestavení webové aplikace, včetně všech externích knihoven, byl použit nástroj Parcel. Jedná se o správce modulů v jazyce JavaScript určeného pro tvorbu webových aplikací. Parcel zjednodušuje instalaci knihoven, sestavení a vývoj aplikace. Při změně kódu automaticky aktualizuje webovou stránku. Jedná se o velmi rychlý a jednoduše použitelný nástroj [43]. Parcel zrychluje práci na vývoji webových stránek a umožňuje vývojářům soustředit se na obsah a funkcionalitu stránky.

8.2 Konfigurace aplikace

Konfigurace a nastavení aplikace je uloženo v souboru `config.yml` ve složce `config`. Tento soubor je využíván serverovou částí i webovou aplikací. Konfigurace obsahuje zejména nastavení pro připojení k RabbitMQ, k systému rozpoznávání řeči a dále nastavení mapy a oblasti, kterou zobrazuje. Popis jednotlivých nastavení je uveden v komentářích v tomto souboru. Složka `config` dále obsahuje soubory `waypoints.json` s definicí traťových bodů ve formátu GeoJSON a `areas.yml` s definicí oblastí pro zobrazení letadel odlišnou barvou.

[1] https://docs.conda.io/en/latest/miniconda.html

8.3 Implementace webové aplikace

Pro vytvoření jednotného a moderního vzhledu aplikace je použita knihovna Bootstrap 5.2.2. Pro zjednodušení interakcí mezi kódem v jazyce JavaScript a webovou stránkou v HTML byla použita knihovna jQuery.

Základem rozhraní webové stránky je mapa světa, na které jsou zobrazeny traťové body a aktuální poloha letadel získaná z reálných dat nebo ze simulace letového provozu. Na mapě se dále zobrazují informace o vybraném letadle a výpis komunikace mezi řídícím letového provozu a pilotem.

Strukturu zdrojových souborů webové aplikace zobrazuje uvedený adresářový strom. Obsah jednotlivých souborů je popsán v následujících odstavcích, soubory `rabbit_comm.js` a `audio.js` jsou popsány v samostatných kapitolách.

```
/
├── /config
│   ├── /areas.yml
│   ├── /config.yml
│   └── /waypoints.json
└── /webapp
    ├── /src
    │   ├── /json
    │   │   └── /map_medium.json
    │   ├── /scripts
    │   │   ├── aircrafts.js
    │   │   ├── audio.js
    │   │   ├── comm_parser.js
    │   │   ├── commands.js
    │   │   ├── main.js
    │   │   ├── map.js
    │   │   ├── rabbit_comm.js
    │   │   └── waypoints.js
    │   └── /styles
    └── index.html
```

- **index.html** – obsahuje definici webové stránky v jazyce HTML a všech objektů, které se na této stránce vyskytují.
- **main.js** – hlavní program, který provede načtení konfigurace, mapy, oblastí a traťových bodů. Dále provádí inicializaci jednotlivých modulů a také obsahuje obsluhu ovládacích prvků aplikace.
- **map.js** – tento modul obsahuje funkce pro práci s mapou. Mapa se vytváří za pomocí knihovny Leaflet. Měla by obsahovat jen obrysy států, proto byla použita vektorová data z veřejné mapové datové sady Natural Earth[2]. Tato data byla následně převedena do formátu GeoJSON pomocí nástroje Mapshaper[3]. Tento modul dále obsahuje funkce pro nastavení zobrazeného výřezu mapy a nastavení vzhledu mapy.

[2] https://www.naturalearthdata.com/downloads/50m-cultural-vectors
[3] https://mapshaper.org

- **waypoints.js** – v tomto modulu je řešeno zobrazení traťových bodů. Vytvoří se nová vrstva na mapě, která obsahuje traťové body. Modul dále obsahuje logiku, pro zobrazení traťových bodů pouze při určitém přiblížení mapy.

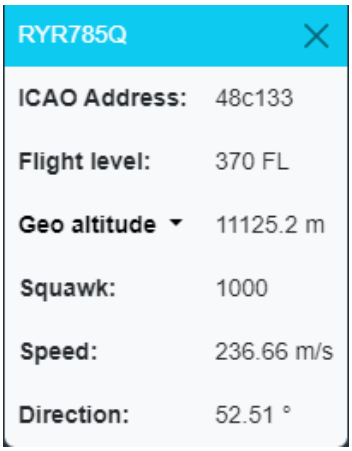
  Seznam traťových bodů byl získán z veřejně dostupného dokumentu na stránkách společnosti Eurocontrol[4] ve formátu xlsx. Z tohoto dokumentu byly vybrány traťové body ležící na území České republiky a ty byly exportovány do formátu csv. Následně proběhl převod do formátu GeoJSON, který je implementován v programu `waypoints_geojson.py`. Traťové body jsou na mapě vyznačeny bílými trojúhelníky.

- **aircrafts.js** – v tomto modulu je implementováno zobrazení aktuální polohy letadel. Při inicializaci se vytvoří samostatné vrstvy na mapě pro zobrazení letadel a směrů letu.

  Funkce `addAircrafts` zpracovává přijatá data ve formátu JSON a dále tato data porovnává s aktuálním stavem na mapě. U letadel, která jsou již zobrazena na mapě, se aktualizují jejich údaje a poloha. Nově přidaná letadla jsou zpracována, převedena do formátu GeoJSON a vložena do mapy. V případě, že přijatá data neobsahují údaje o některém letadle, které je zobrazené na mapě, změní se jeho barva na šedou. Pokud není takové letadlo obsaženo ani v následujícím bloku přijatých dat, je letadlo smazáno.

  Funkce `directionVector` vypočítává vektor směru letadla, který se na mapě zobrazuje jako krátká úsečka. Kód pro výpočet souřadnice vektoru byl převzat ze webové stránky Movable Type Scripts[5].

  Funkce `onClick` obsluhuje označení letadla uživatelem pomocí myši. Letadlo se po označení zvýrazní a na pravé straně obrazovky se zobrazí základní informace o vybraném letadle. Pokud jsou informace zobrazené, tak se po přijetí nového bloku dat automaticky aktualizují.



Obrázek 8.1: Tabulka se základními informace o vybraném letadle

  Funkce `setColorForArea` umožňuje nastavení barvy letadla podle oblasti, ve které se letadlo nachází. Oblasti jsou definovány zeměpisnými souřadnicemi a výškou letadla.

[4] https://www.eurocontrol.int/publication/free-route-airspace-fra-points-list-ecac-area
[5] https://movable-type.co.uk/scripts/latlong.html

Údaje o oblasti, včetně požadované barvy, je možné nastavit v konfiguraci v souboru `areas.yml`.

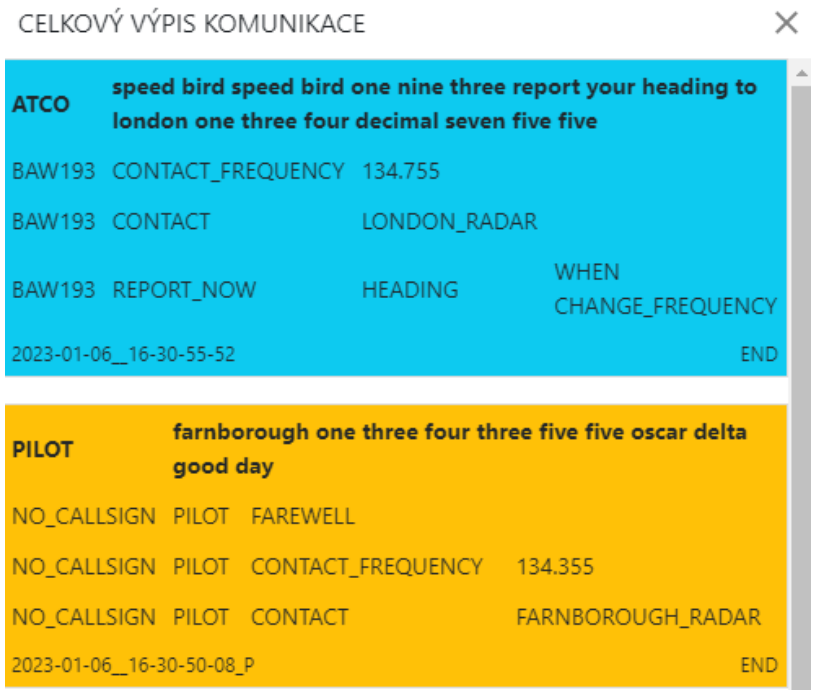
Dále tento modul obsahuje funkce pro zvýraznění letadla, o kterém probíhá aktuální komunikace, výpis informací o letadle a další pomocné funkce.

- **comm_parser.js** – v tomto modulu se zpracovává příchozí komunikace řídícího letového provozu a pilota, aby ji bylo možné přehledně vypsat na obrazovku.

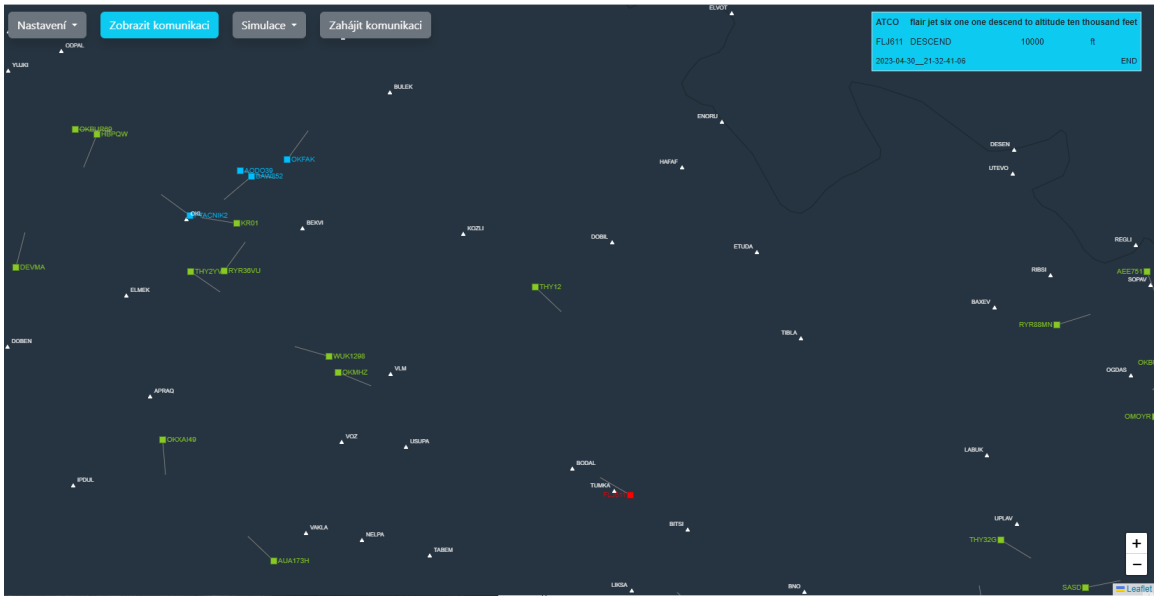
  Komunikace se posílá průběžně a je rozdělena do konkrétních zpráv od pilota, nebo řídícího letového provozu. Samotná zpráva se posílá okamžitě, i když ještě daná osoba nedomluvila, takže text zprávy může přijít několikrát neúplný, dokud nedojde konečný text. To je možné rozeznat tak, že konečný text obsahuje položku `endpoint` s hodnotou `true`. Při příchodu nové zprávy je tedy nutné nejprve ověřit, jestli se jedná o novou zprávu, nebo aktualizaci stávající zprávy.

  Aktuální zpráva se zobrazí v okně na pravé straně obrazovky. Historii zpráv je možné sledovat v bočním panelu na levé straně. V případě, že uživatel klikne na nějaké letadlo, zobrazí se v tomto panelu pouze komunikaci s daným letadlem, pokud již nějaká proběhla.

- **commands.js** – tento modul zajišťuje automatické provádění příkazů z hlasové komunikace. Pokud je součástí komunikace mezi ATCO a pilotem nějaký příkaz, vyvolá se funkce z tohoto modulu, která příkaz převede do formátu srozumitelného pro simulační knihovnu BlueSky a odešle ho simulačnímu programu prostřednictvím RabbitMQ. Jsou podporovány pokyny pro změnu směru letu, výšky letadla a rychlosti. Seznam podporovaných příkazů se nachází v příloze C.



Obrázek 8.2: Výpis komunikace mezi pilotem a ATCO v aplikaci. Modrou barvou je označena komunikace ATCO a žlutou barvou je označena komunikace pilota.



Obrázek 8.3: Hlavní stránka aplikace s ukázkou výpisu aktuální komunikace

8.4 Implementace webového serveru

Webový server je implementován v jazyku Python v modulu `radar_server.py`. Tento program plní několik funkcí:

- Server pro komunikaci s webovou aplikací implementovaný pomocí knihovny `aiohttp`[6] podporující protokoly HTTP a HTTPS. Protokol HTTPS je vyžadován pro připojení pomocí WebRTC.
- Signalizační server po protokol WebRTC
- Klient protokolu WebRTC pro příjem záznamu zvuku a jeho předání do systému rozpoznávání řeči.

Jako inspirace pro tento server byl využit program `server.py`[7], který je součástí knihovny aiortc.

8.5 Získávání reálných dat z letového provozu

Reálná data o letadlech se získávají prostřednictvím OpenSky Network API[8], které poskytuje aktuální informace o letovém provozu ve vybrané oblasti. Pro získání neomezeného přístupu k datům je nutná registrace na stránce Opensky Network[9] a následná autentizace při použití API. Data z OpenSky Network API jsou zpracována, převedena do formátu požadovaného webovou aplikací (viz kapitola 7.5) a poslána aplikaci prostřednictvím RabbitMQ.

[6] https://github.com/aio-libs/aiohttp
[7] https://github.com/aiortc/aiortc/blob/main/examples/server/server.py
[8] https://openskynetwork.github.io/opensky-api
[9] https://opensky-network.org

8.6 Zasílání zpráv mezi serverovou částí a webovou aplikací

Jak již bylo zmíněno v kapitole 7.4, komunikace s aplikací probíhá pomocí systému pro zasílaní zpráv RabbitMQ. Webová aplikace s k RabbitMQ připojuje prostřednictvím pluginu RabbitMQ Web STOMP.

Pro připojení serverové strany je použita knihovna Pika[10], která nabízí rozhraní mezi RabbitMQ a jazykem Python. Tuto knihovnu využívají programy `opensky_publisher.py` pro zasílání dat o reálných letadlech, `sim_publisher.py` a `sim_cunsumer.py` pro data simulace a testovací program `comm_publisher.py` pro testování komunikace mezi pilotem a ATCO. Připojení vždy začíná autentizací, dále je zvolen typ exchange a routing key, jejichž význam byl popsán v teoretické části. Pro všechny typy zpráv jsou podporovány exchange typu topic a direct. Pro každý typ zprávy je definován specifický routing key.

Připojení ze strany webové aplikace je implementováno v modulu `rabbit_comm.js`. Pro připojení se používá knihovna webstomp-client[11]. Webová aplikace čte z front zpráv informace o letadlech a komunikaci mezi ATCO a pilotem. Do RabbitMQ naopak zapisuje příkazy pro simulaci letového provozu, které jsou zadané uživatelem, nebo získané z komunikace. Připojení k RabbirMQ probíhá prostřednictvím protokolu WebSocket. Je podporováno připojení pomocí nezabezpečeného (`ws`) i zabezpečeného (`wss`) protokolu WebSocket.

Veškerá nastavení pro připojení k RabbitMQ na straně webové aplikace i serveru je možné konfigurovat v souboru `config.yml` (viz kapitola 8.2).

8.7 Simulace letového provozu

Simulace letového provozu je implementování pomocí knihovny BlueSky[12]. Simulace pracuje ve dvou vláknech. První vlákno provádí simulaci a dále bude nazýváno jako simulační vlákno. Druhé vlákno zpracovává příkazy, které přichází z prohlížeče přes RabbitMQ, a přidává je do fronty pro simulaci. Dále bude nazýváno jako vlákno zpracování příkazů. Strukturu zdrojových souborů simulace zobrazuje následující adresářový strom:

```
/
├── /config
└── /simulation
    ├── sim_consumer.py
    ├── sim_flights.py
    ├── sim_publisher.py
    └── traffic_simulation.py
```

- **sim_publisher.py** – tento modul obsahuje třídu `simPublisher`, která posílá aktuální informace o všech letadlech v daném kroku simulace do RabbitMQ. Bližší informace o připojení k RabbitMQ jsou uvedeny v kapitole 8.6.
- **sim_consumer.py** – tento modul obsahuje třídu `simConsumer`, která čte příkazy pro simulaci z fronty RabbitMQ. Po přijetí nové zprávy je vyvolána metoda `callback`, která dekóduje přijatý příkaz a ten pak předá simulaci.
- **sim_flights.py** – hlavní program simulace. Nejprve provede zpracování parametrů z příkazové řádky, poté nastaví interval a počet kroků simulace. Následně vytvoří

[10] https://pypi.org/project/pika/
[11] https://www.npmjs.com/package/webstomp-client
[12] https://github.com/TUDelft-CNS-ATM/bluesky

objekty pro simulaci a komunikaci přes RabbitMQ. Pro příjem příkazů z RabbitMQ je vytvořeno samostatné vlákno. Nakonec je zahájena simulaci.

- **traffic_simulation.py** – v tomto modulu probíhá vlastní simulace prostřednictvím knihovny BlueSky. BlueSky standardně předpokládá propojení s grafickým uživatelským rozhraním jazyka Python. To se v této práci nevyužívá, a proto třída `ConsoleOutput` přesměrovává výstup simulace na konzolu. Třída `TrafficSimulation` poskytuje metody pro inicializaci simulace, načtení příkazů ze souboru, zpracování příkazu a provedení jednoho kroku simulace. Krok simulace je řešený v metodě `simStep`. Jedná se o přepracovanou metodu `update`[13] rozšířenou o synchronizaci vláken pomocí objektu `Condition`[14] a přerušení simulace v případě, že nejsou vytvořena žádná letadla.

  Simulace probíhá následujícím způsobem:

  1. Simulační vlákno je uspáno na dobu intervalu mezi dvěma kroky simulace (standardně 10s). Po probuzení vykoná nové příkazy ve frontě příkazů a provede jeden krok simulace. Poté přečte aktualizovaný seznam informací o letadlech a pomocí metody `getSimData` jej převede do formátu JSON pro webovou aplikaci a ten odešle prostřednictvím RabbitMQ. Pokud neexistují žádná letadla, uspí se poté toto vlákno až do obdržení nového příkazu, jinak se uspí do doby provedení dalšího kroku.
  2. Vlákno zpracování příkazů čeká na obdržení příkazu. Pokud přijde nový příkaz, vyvolá se metoda `addCommand`. Tato metoda jej zpracuje a přidá do fronty příkazů ke zpracování. Poté pomocí metody `notify` objektu `Condition` informuje simulační vlákno o novém příkazu, čímž dojde k okamžitému probuzení simulačního vlákna a vyvolání dalšího kroku simulace.

  Použití zamykání pomocí objektu `Condition` zajišťuje, že nemůže dojít ke kolizi obou vláken při přidávání a zpracování příkazů.

8.8 Hlasová komunikace

Aplikace umožňuje uživateli zadávat hlasové příkazy simulovaným letadlům a reagovat na tyto příkazy jako pilot. Obsah by měl odpovídat pokynům, které dává řídící letového provozu pilotům. Zpráva uživatele se z aplikace posílá do Systému pro rozpoznání řeči, který ji převede na text. Text se poté zobrazuje uživateli v aplikaci.

Na straně webové aplikace je zachytávání mluveného slova implementováno v modulu `audio.js` a využívá protokol WebRTC. Kód použitý v tomto modulu je inspirovaný souborem `client.js`[15], který je součástí knihovny aiortc.

Po kliknutí na tlačítko pro nahrávání se zavolá funkce `start`, která zahájí záznam z mikrofonu. Nejprve se vytvoří nové WebRTC spojení pomocí funkce `createPeerConnection`. Při prvním použití funkce se objeví výzva uživateli, aby možnost záznamu z mikrofonu povolil. Následně je vytvořena nabídka (offer), která je zaslána pomocí HTTP protokolu na webový server. Server po přijetí nabídky vytvoří druhého klienta protokolu WebRTC a zašle webové aplikaci informace k navázání spojení. Poté už probíhá přenos záznamu zvuku

[13] https://github.com/TUDelft-CNS-ATM/bluesky/blob/master/bluesky/simulation/simulation.py
[14] https://docs.python.org/3/library/threading.html#condition-objects
[15] https://github.com/aiortc/aiortc/blob/main/examples/server/client.js

až do chvíle, kdy uživatel ukončí komunikaci a spojení se uzavře. Na straně webového serveru je použita knihovna `aiortc`[16], která zajišťuje navázání komunikace přes protokol WebRTC a vytvoření druhého klienta, který přijímá záznam zvuku webové aplikace a dále jej předává do systému rozpoznávání řeči. Pro tuto funkci byla použita upravená třída `MediaRecorder`[17], která původně sloužila k ukládání záznamu zvuku do souboru. Modifikovaná verze místo toho předává tento signál přes TCP spojení na adresu a port systému rozpoznávání řeči.

8.9 Konvertor protokolu ASTERIX

Standardně se jako komunikační protokol pro přenos informací mezi letadly a pozemními službami na letištích používá protokol ASTERIX. V této práci je implementován konvertor tohoto formátu v souboru `asterix_parser.py`, který za pomocí knihovny `asterix`[18] pro jazyk Python převádí formát ASTERIX na formát JSON (viz příloha B). Konvertor podporuje pouze protokol ASTERIX kategorie 62, která svým obsahem nejlépe odpovídá formátu podporovanému webovou aplikací. Tento konvertor se v aplikaci nevyužívá, ale byl zde ponechán pro případné využití v navazujících projektech.

[16] https://github.com/aiortc/aiortc
[17] https://github.com/aiortc/aiortc/blob/main/src/aiortc/contrib/media.py
[18] https://github.com/CroatiaControlLtd/asterix

Kapitola 9

Testování a vyhodnocení systému

Tato kapitola se věnuje testování vytvořeného systému. Během vývoje byly průběžně testovány aktuálně implementované funkce. Kapitola popisuje testování funkcí aplikace a věnuje se zejména popisu částí, které bylo nutné komplexněji otestovat. Dále je zde popsán průběh a výsledky testování uživatelského rozhraní.

9.1 Testování funkcí aplikace

Veškeré funkce aplikace byly testovány průběžně během vývoje. Pro testování a ladění frontendu byly využity pokročile vývojářské nástroje prohlížeče Chrome. Ladění backendu probíhalo pomocí logovacích výpisů do konzole. Funkčnost frontendové části byla otestována v nejpoužívanějších prohlížečích Chrome a Edge. Funkčnost serverové části byla ověřena ve WSL (Windows Subsystem for Linux) se systémem Ubuntu 20.04 a dále na školním serveru athena1. Pro otestování zobrazení letadel byla použita testovací sada získaná přes OpenSkyAPI (cca 1 hodina letového provozu nad Českou republikou) a simulační scénáře pro knihovnu BlueSky. Pro testování správného zobrazení komunikace mezi pilotem a ATCO byla použita testovací sada vytvořená systémem pro rozpoznání řeči.

9.2 Testování uživatelského rozhraní

Následně bylo provedeno testování uživateli, zaměřené na pochopení všech funkcí uživatelského rozhraní. Účelem bylo také ověřit, je-li uživatelské rozhraní intuitivní a aplikace snadno ovladatelná. Aplikaci testovali čtyři uživatelé. Těm byla nejprve objasněna funkce aplikace a cíl testování. Pote obdrželi následující seznam úkolů k otestování:

1. Spusťte simulační scénář s názvem „Základní test“.
2. Najděte na mapě letadlo FLJ611 a zobrazte o něm dostupné informace.
3. Přepněte se do role ATCO a řekněte pilotovi následující příkaz: „flair jet six one one descend to altitude ten thousand feet“.
4. Znovu najděte letadlo na mapě a zkontrolujte si, že provedlo vaše pokyny (letadlo postupně klesá až na letovou hladinu 100).
5. Vypněte automatické provádění příkazů.

6. Řekněte následující příkaz: „easy one five quebec kilo turn left right heading one eight zero degrees“.
7. Přepněte se do role pilota a zadejte letadlu ručně tento příkaz: „HDG EZY15QK 180“.
8. Zkontrolujte historii komunikace.
9. Zastavte simulaci.

Všichni uživatelé úspěšně splnili zadané úkoly. Při testování bylo zjištěno několik chyb v chování aplikace, které byly následně odstraněny. Jelikož se jedná o specializovanou aplikaci z oblasti řízení letového provozu, bylo pro některé uživatele zpočátku obtížné pochopit význam a smysl jednotlivých funkcí. Dva uživatelé měli problém zjistit, jak zobrazit informace o letadle, ale po krátké době našli správný postup. Tyto problémy souvisely s prvním použitím nové aplikace a při častějším používání se nevyskytnou. Uživatelské rozhraní bylo hodnoceno převážně kladně. Na základě podnětů uživatelů byly upraveny texty a funkce tlačítek v sekci nastavení.

Kapitola 10

Závěr

Cílem této práce byl návrh a implementace aplikace pro zobrazení radarových dat, kterou by bylo možné využít pro výcvik začínajících řídících letového provozu. Aplikace měla být propojená se systémem pro rozpoznání řeči a měla v ní být implementována simulace letového provozu. Před vytvořením návrhu a samotné implementace bylo nutné prostudovat teorii z oblasti letectví a řízení letového provozu. V rámci teoretické části práce bylo poté potřeba nastudovat principy tvorby webových stránek, vizualizace dat na mapě, simulace letového provozu, posílání zpráv a hlasové komunikace. Kromě toho bylo zapotřebí prozkoumat podobná řešení a technologie v této oblasti.

Výsledná aplikace pracuje ve dvou režimech. V prvním režimu zobrazuje aplikace na mapě světa reálná aktuální data o letadlech. V tomto režimu může uživatel zkoumat aktuální letový provoz a zobrazovat si informace o konkrétních letadlech. Ve druhém režimu se na mapě zobrazuje simulovaný letecký provoz. Dva uživatelé mezi sebou mohou navzájem prostřednictvím aplikace komunikovat, jeden v roli ATCO a druhý v roli pilota. Simulace automaticky reaguje na příkazy od ATCO. Pilot má také možnost změnit simulaci sám a to tak, že zadá příkazy ručně.

Jednotlivé funkce aplikace byly testovány v průběhu vývoje a celkové výsledky mé práce byly následně otestovány čtyřmi nezávislými uživateli. Zjištěné závady byly odstraněny a uživatelské rozhraní bylo upraveno na základě podnětů z testování.

Po konzultaci s odborníky v oblasti školení řídících letového provozu by bylo možné aplikaci upravit přesně na míru tomuto procesu. Aplikace by mohla být rozdělena do různých úrovní a uživatel by například začal s učením příkazů pro rádiovou komunikaci a dále by si zkoušel komunikaci již na simulaci, která by postupně zvyšovala obtížnost řešených situací.

Dále by mohl být systém rozšířen o podporu pro tzv. callsign boosting, který zlepšuje přepis komunikace do textu tím, že přiřazuje větší pravděpodobnost volacím znakům letadel na radaru. Aplikace by musela být doplněna o zasílání volacích znaků letadel aktuálně zobrazených na obrazovce do systému rozpoznávaní řeči.

Aplikace by mohla být také rozšířena o syntézu řeči, aby bylo možné simulovat odpovědi pilota.

Literatura

[1] Abdishakur. *Best Libraries for Geospatial Data Visualisation in Python* [online], 29. července 2020 [cit. 2022-11-2]. Dostupné z: https://towardsdatascience.com/best-libraries-for-geospatial-data-visualisation-in-python-d23834173b35.

[2] Agafonkin, V. *Leaflet* [online]. Leaflet [cit. 2022-11-15]. Dostupné z: https://leafletjs.com/.

[3] ANSART. *Air Traffic Control Simulator* [online]. [cit. 2023-2-20]. Dostupné z: https://ansart.nl/air-traffic-control-simulator/.

[4] ATC-SIM. *ATC-SIM - Air Traffic Control Simulation* [online]. [cit. 2023-2-20]. Dostupné z: https://atc-sim.com/.

[5] Breuss, M. *Python Folium: Create Web Maps From Your Data* [online]. Real Python, 11. ledna 2023 [cit. 2023-01-15]. Dostupné z: https://realpython.com/python-folium-web-maps-from-data/.

[6] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. et al. *The GeoJSON Format* [online]. RFC Editor, srpen 2016. ISSN 2070-1721. RFC 7946. Dostupné z: https://www.rfc-editor.org/rfc/rfc7946.txt.

[7] Castro, E. a Hyslop, B. *HTML5 a CSS3: názorný průvodce tvorbou WWW stránek*. 1. vyd. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[8] Conda. *Conda* [online]. [cit. 2023-4-8]. Dostupné z: https://docs.conda.io/en/latest/#.

[9] CS-Soft. *ATC Simulator* [online]. [cit. 2023-2-20]. Dostupné z: https://www.cs-soft.cz/en/atc-simulator.

[10] DigitalSkyNet. *Desktop App vs. Web App: Comparative Analysis* [online]. [cit. 2022-11-20]. Dostupné z: https://digitalskynet.com/blog/Desktop-App-vs-Web-App-Comparative-Analysis.

[11] Domes, M. *333 tipů a triků pro CSS: sbírka nejužitečnějších návodů pro váš web*. 1. vyd. Brno: Computer Press, 2009. ISBN 978-80-251-2360-7.

[12] Eurocontrol. *ASTERIX: All-purpose structured EUROCONTROL surveillance information exchange* [online]. EUROCONTROL [cit. 2022-11-02]. Dostupné z: https://www.eurocontrol.int/asterix.

[13] Eurocontrol. *ESCAPE: EUROCONTROL simulation capabilities and platform for experimentation* [online]. [cit. 2023-2-20]. Dostupné z: https://www.eurocontrol.int/simulator/escape.

[14] Federal Aviation Administration. *AIM - Chapter 4. Air Traffic Control* [online]. 2021 [cit. 2023-02-12]. Dostupné z: https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap4_section_2.html.

[15] Federal Aviation Administration. *Order JO 7110.65AA - Air Traffic Control* [online]. Order. Duben 2023 [cit. 2023-4-23]. 37 s. Dostupné z: https://www.faa.gov/regulations_policies/orders_notices/index.cfm/go/document.information/documentID/1029467.

[16] Flightradar24. *How flight tracking works* [online]. [cit. 2022-12-14]. Dostupné z: https://www.flightradar24.com/how-it-works.

[17] Freudenrich, C. *How Air Traffic Control Works* [online]. HowStuffWorks, 12. května 2021 [cit. 2023-02-12]. Dostupné z: https://science.howstuffworks.com/transport/flight/modern/air-traffic-control.htm.

[18] GeoPandas. *GeoPandas 0.12.2* [online]. [cit. 2022-12-16]. Dostupné z: https://geopandas.org/en/stable/.

[19] González, M. *Geospatial Data Representation: The GeoJSON Format* [online]. datascience.aero, 7. března 2022 [cit. 2022-12-18]. Dostupné z: https://datascience.aero/geospatial-data-representation-the-geojson-format/.

[20] Gruber, T. *Ontology*. Boston, MA: Springer US, 2009. 1963–1965 s. ISBN 978-0-387-39940-9. Dostupné z: https://doi.org/10.1007/978-0-387-39940-9_1318.

[21] HAAWAII. *HAAWAII: Highly Automated Air Traffic Controller Workstations With Artificial Intelligence Integration*. [cit. 2023-4-4]. Dostupné z: https://www.haawaii.de/wp/.

[22] Helmke, H., Ondřej, K., Shetty, S., Arilíusson, H., Simiganosch, T. S. et al. Readback Error Detection by Automatic Speech Recognition and Understanding - Results of HAAWAII project for Isavia's Enroute Airspace. In: *12th SESAR Innovation Days*. December 2022. Dostupné z: https://elib.dlr.de/190668/.

[23] Helmke, H., Slotty, M., Poiger, M., Herrer, D. F., Ohneiser, O. et al. Ontology for Transcription of ATC Speech Commands of SESAR 2020 Solution PJ.16-04. In: *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. 2018, s. 1–10. DOI: 10.1109/DASC.2018.8569238. Dostupné z: https://ieeexplore.ieee.org/document/8569238.

[24] Hoekstra, J. M. *BlueSky - The Open Air Traffic Simulator* [online]. Srpen 2022 [cit. 2023-2-3]. Dostupné z: https://github.com/TUDelft-CNS-ATM/bluesky/wiki.

[25] Holcombe, J. *Backend vs Frontend: How Are They Different?* [online]. Kinsta, 23. prosince 2022 [cit. 2022-12-14]. Dostupné z: https://kinsta.com/blog/backend-vs-frontend/.

[26] Holoviz contributors. *High-level tools to simplify visualization in Python* [online]. Holoviz, 22. března 2023 [cit. 2022-11-15]. Dostupné z: https://holoviz.org/.

[27] Holzner, S. *JavaScript profesionálně: kompletní referenční příručka.* Praha: Mobil Media, 2003. ISBN 80-86593-40-1.

[28] IBM. *What are message brokers?* [online]. [cit. 2022-11-2]. Dostupné z: https://www.ibm.com/cz-en/cloud/learn/message-brokers.

[29] Jadon, A. *Understanding RabbitMQ Queue & Messaging Simplified 101* [online]. HEVO, 22. dubna 2022 [cit. 2022-11-2]. Dostupné z: https://hevodata.com/learn/rabbitmq-queue/.

[30] James, R. *Why Do Aircraft & Pilots Have Call Signs?* [online]. Pilot Teacher [cit. 2023-4-20]. Dostupné z: https://skybrary.aero/articles/aircraft-call-sign.

[31] Johansson, L. *Part 1: RabbitMQ for beginners - What is RabbitMQ?* [online]. CloudAMQP, 23. září 2019 [cit. 2022-11-3]. Dostupné z: https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html.

[32] Johansson, L. *Part 4: RabbitMQ Exchanges, routing keys and bindings* [online]. CloudAMQP, 24. září 2019 [cit. 2022-11-3]. Dostupné z: https://www.cloudamqp.com/blog/part4-rabbitmq-for-beginners-exchanges-routing-keys-bindings.html.

[33] Johnston, J. *A beginners guide to web application development (2022)* [online]. Budibase, 24. ledna 2020 [cit. 2022-12-10]. Dostupné z: https://budibase.com/blog/web-application-development/.

[34] jQuery Community Experts. *JQuery - kuchařka programátora: programátorské techniky a webové technologie.* 1. vydání. Brno: Computer Press, 2010. ISBN 978-80-251-3152-7.

[35] JSON. *Introducing JSON* [online]. [cit. 2022-12-16]. Dostupné z: https://www.json.org/json-en.html.

[36] Kmitocty. *Letecká radiová komunikace* [online]. [cit. 2022-12-3]. Dostupné z: https://kmitocty.cz/?p=143.

[37] Lainé, J. *Aiortc* [online]. 2023 [cit. 2023-3-14]. Dostupné z: https://aiortc.readthedocs.io/en/latest/.

[38] Lett, J. *Bootstrap 4: quick start.* Michigan: Bootstrap Creative, 2019. ISBN 978-1-7322058-1-9.

[39] Lubbers, P., Albers, B. a Salim, F. *HTML5: programujeme moderní webové aplikace.* Vyd. 1. Brno: Computer Press, 2011. ISBN 978-80-251-3539-6.

[40] Lutz, M. *Learning Python.* 5th ed. Sebastopol: O´Reilly, 2013. ISBN 978-1-449-35573-9.

[41] Mozilla Contributors. *WebRTC connectivity* [online]. Mozilla Developer Network, 15. března 2023 [cit. 2023-01-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Connectivity.

[42] Pande, P. a Hardiman, J. *A Look At The World Of Squawk Codes And Their Meanings* [online]. Simple Flying, duben 2022 [cit. 2022-10-16]. Dostupné z: https://simpleflying.com/squawk-codes/.

[43] Parcel. *Parcel* [online]. [cit. 2023-2-15]. Dostupné z: https://parceljs.org/.

[44] RabbitMQ. *RabbitMQ web STOMP plugin* [online]. [cit. 2022-12-21]. Dostupné z: https://www.rabbitmq.com/web-stomp.html.

[45] Řízení letového provozu ČR. *Výcvik řlp-žáků* [online]. [cit. 2023-20-04]. Dostupné z: https://www.rlp.cz/articlesb/A2_2_4?CatCode=B2.

[46] Rouse, M. *Web Development* [online]. techopedia, 31. srpna 2020 [cit. 2022-12-8]. Dostupné z: https://www.techopedia.com/definition/23889/web-development.

[47] Schäferhoff, N. *Bootstrap Tutorial: How to Setup and Use Bootstrap (Step-by-Step)* [online]. WebsiteSetup Logo, 8. prosince 2020 [cit. 2022-12-18]. Dostupné z: https://websitesetup.org/bootstrap-tutorial-for-beginners/.

[48] Shacklett, M. E. *WebRTC (Web Real-Time Communications)* [online]. TechTarget network, říjen 2021 [cit. 2023-01-24]. Dostupné z: https://www.techtarget.com/searchunifiedcommunications/definition/WebRTC-Web-Real-Time-Communications.

[49] Singh, S. *How Does Air Traffic Control Work?* [online]. Simple Flying, 21. října 2022 [cit. 2022-12-4]. Dostupné z: https://simpleflying.com/how-does-air-traffic-control-work/.

[50] Skybrary. *Air Traffic Service (ATS)* [online]. [cit. 2022-25-11]. Dostupné z: https://skybrary.aero/articles/air-traffic-service-ats.

[51] Stock, K. a Guesgen, H. *Automating Open Source Intelligence*. Boston: Syngress, 2016. 171-204 s. ISBN 978-0-12-802916-9. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780128029169000105.

[52] Stomp. *Stomp* [online]. 2012 [cit. 2023-3-14]. Dostupné z: http://stomp.github.io/.

[53] Techmap. *ASTERIX (ATC Standard)* [online], 20. května 2021 [cit. 2022-11-25]. Dostupné z: https://techmap.io/technology/ASTERIX_(ATC_Standard),Skill/60f6f2a1ba93cb76f893e15d.

[54] The OpenSky Network. *The OpenSky Network* [online]. [cit. 2022-12-14]. Dostupné z: https://opensky-network.org/about/about-us.

[55] Tomasis, R. *Static vs Dynamic Websites: The Differences, Advantages and Which to Use* [online]. Wix, listopad 2021 [cit. 2022-12-14]. Dostupné z: https://www.wix.com/blog/2021/11/static-vs-dynamic-website/.

[56] Vanhoenacker, M. A Pilot Explains Waypoints, the Hidden Geography of the Sky. *Condé Nast Traveler* [online]. Červen 2015. Dostupné z: https://www.cntraveler.com/stories/2015-06-02/a-pilot-explains-waypoints-the-hidden-geography-of-the-sky.

[57] Výletiště. *Zastávka: Řízení letového provozu* [online]. [cit. 2022-12-16]. Dostupné z: https://www.vyletiste.cz/zastavka/11/rizeni-letoveho-provozu.

[58] Wempen, F. *HTML a CSS: krok za krokem*. 1. vyd. Brno: Computer Press, 2007. ISBN 978-80-251-1505-3.

[59] Žára, O. *JavaScript: programátorské techniky a webové technologie*. 2. vydání. Brno: Computer Press, 2021. ISBN 978-80-251-5026-9.

Příloha A

Obsah přiloženého paměťového média

Paměťové médium obsahuje následující složky a soubory:

- `app/` – zdrojové soubory aplikace a návod na spuštění v souboru `README.md`
- `doc/` – zdrojové soubory technické zprávy
- `xbuchn00-Radar.pdf` – technická zpráva ve formátu PDF
- `xbuchn00-Plakat.pdf` – plakát k práci ve formátu PDF

Příloha B

Formát pro zasílání informací o letadlech

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Aircraft data",
  "description": "Format for sending aircraft data",
  "type": "object",
  "properties": {
    "lon": {
      "description":
        "Geographic coordinate; the east-west position in degrees",
      "type": "number",
      "minimum": -180,
      "maximum": 180
    },
    "lat": {
      "description":
        "Geographic coordinate; the north-south position in degrees",
      "type": "number",
      "minimum": -90,
      "maximum": 90
    },
    "adr": {
      "description":
        "ICAO24 address of the transmitter in hex string representation",
      "type": "string"
    }
    "callsign": {
      "description": "Unique identifier for a transmitter station",
      "type": "string"
    },
    "geo_altitude": {
      "description": "Geografic altitude in meters",
      "type": "number"
```

```
    },
    "baro_altitude": {
      "description": "Barometric altitude in meters",
      "type": "number"
    },
    "FL": {
      "description":
        "Altitude at standard air pressure in hundreds of feet",
      "type": "number"
    },
    "time": {
      "description": "Seconds since the last position report",
      "type": "string"
    },
    "squawk": {
      "description": "Transponder code",
      "type": "string"
    },
    "direction": {
      "description": "Direction in decimal degrees, where 0 is north",
      "type": "number"
    },
    "velocity": {
      "description": "Speed in m/s",
      "type": "number"
    },
  },
  "required": ["lon", "lat"]
}
```

Příloha C

Příkazy pro simulaci

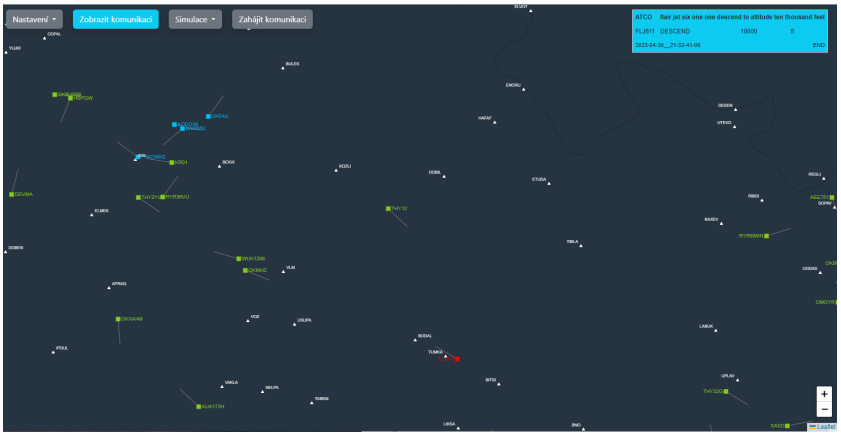
Příkaz	Popis	Jednotka
DIRECT_TO	přidání jednoho nebo více traťových bodů do letového plánu	
HEADING	změna směru letu	stupně
TURN_BY	změna směru letu o daný počet jednotek	stupně
INCREASE	zvýšení rychlosti letadla na určitou rychlost	kt, MA
INCREASE_BY	zvýšení rychlosti letadla o daný počet jednotek	kt
REDUCE	zpomalení letadla na určitou rychlost	kt, MA
REDUCE_BY	zpomalení letadla o daný počet jednotek	kt
SPEED	změna rychlosti letadla	kt, MA
CLIMB	stoupání letadla	FL, ft
DESCEND	klesání letadla	FL, ft

Příloha D

Plakát

Systém pro výcvik řídících letového provozu

Bakalářská práce
autor: Tereza Buchníčková
vedoucí: doc. RNDr. Pavel Smrž, Ph.D.
rok: 2023



- komunikace v roli řídícího letového provozu
- automatický přepis komunikace na text
- simulace letového provozu
- zobrazení reálného letového provozu