**Technická univerzita v Košiciach**
**Fakulta elektrotechniky a informatiky**

# Využitie techník dátového inžinierstva pre pozorovaciu stanicu AMON-ES

**Diplomová práca**

**2023** **Bc. Erik Kandalík**

**Technická univerzita v Košiciach**
**Fakulta elektrotechniky a informatiky**

# Využitie techník dátového inžinierstva pre pozorovaciu stanicu AMON-ES

**Diplomová práca**

| | |
|---|---|
| Študijný program: | Informatika |
| Študijný odbor: | 9.2.1. Informatika |
| Školiace pracovisko: | Katedra počítačov a informatiky (KPI) |
| Školiteľ: | RNDr. Šimon Mackovjak, PhD. |
| Konzultant: | Ing. Michal Solanik |

**Košice 2023** **Bc. Erik Kandalík**

## Abstrakt v SJ

Táto práca sa zaoberá problematikou merania, ukladania a spracovania dát o optickom fenoméne menom airglow. V prvých kapitolách diplomovej práce sa čitateľ bližšie zoznámi s fyzikálnou podstatou tohto fenoménu spolu s priblížením jeho vzniku a motiváciou štúdia tohto javu. Následne sa v práci venujeme návrhu a implementácii s využitím cloudovej platformy AWS. Sústredíme sa na dva prístupy, pričom v prvom budeme využívať serverless služby a v druhom služby založené na klasických hardvérových službách. Popíšeme výhody a nevýhody navrhnutých architektúr a výsledky pri ich implementácii. V závere práce opíšeme dosiahnuté výsledky pri návrhu a implementácii navrhnutej cloudovej architektúry.

## Kľúčové slová v SJ

Vesmír, Airglow, AWS, Python, Serverless, Cloudová architektúra

## Abstrakt v AJ

This thesis deals with the issue of measuring, storing, and processing data about the optical phenomenon called airglow. In the first chapters of the thesis, the reader will become acquainted with the physical nature of this phenomenon, along with an approximation of its origin and motivation for studying this phenomenon. Subsequently, we focus on the design and implementation using the AWS cloud platform. We concentrate on two approaches, where in the first, we will be using serverless services, and in the second, services based on classical hardware services. We describe the advantages and disadvantages of the proposed architectures and the results of their implementation. In the conclusion of the thesis, we describe the achieved results in the design and implementation of the proposed cloud architecture.

## Kľúčové slová v AJ

Space, Airglow, AWS, Python, Serverless, Cloud architecture

## Bibliografická citácia

KANDALÍK, Erik. *Využitie techník dátového inžinierstva pre pozorovaciu stanicu AMON-ES*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2023. 58s. Vedúci práce: RNDr. Šimon Mackovjak, PhD.

68885

**TECHNICKÁ UNIVERZITA V KOŠICIACH**

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

# ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

**Využitie techník dátového inžinierstva pre pozorovaciu stanicu AMON-ES**

Data Engineering Techniques for AMON-ES

Študent: **Bc. Erik Kandalík**
Školiteľ: **RNDr. Šimon Mackovjak, PhD.**
Školiace pracovisko:
Konzultant práce: **Ing. Michal Solanik**
Pracovisko konzultanta: **Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:
1. Oboznámiť sa s fungovaním pozorovacej stanice AMON-ES (Airglow Monitor - Extended Station).
2. Navrhnúť vhodnú technológiu na uskladanie, triedenie a prezentovanie nameraných dát.
3. Implementovať navrhnutú technológiu.
4. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 21.04.2023
Dátum zadania diplomovej práce: 31.10.2022

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

## Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 13.5.2023 .....

*Vlastnoručný podpis*

## Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce RNDr. Šimonovi Mackovjakovi, PhD. a môjmu konzultantovi práce Ing. Michalovi Solanikovi, za ich čas a odborné vedenie počas riešenia mojej záverečnej práce.

Rovnako by som sa rád poďakoval svojim rodičom, svojej priateľke Kataríne Golenyovej a priateľom za ich podporu a povzbudzovanie počas celého môjho štúdia.

# Obsah

# Zoznam obrázkov

# Zoznam tabuliek

# Úvod

Zemská atmosféra je turbulentný a dynamický systém, na ktorý vplýva široké spektrum vesmírnych a pozemských procesov. Vrchné časti zemskej atmosféry sú ovplyvňované svetelným žiarením, nabitými časticami prichádzajúcimi zo Slnka a z hlbokého vesmíru, ale aj pozemským počasím, ako sú búrky či blesky. Porozumenie týmto procesom a tomu, ako ovplyvňujú zemskú atmosféru má priamy vplyv na náš každodenný život. Napríklad rádiová alebo satelitná komunikácia je ovplyvnená nabitými časticami v horných častiach zemskej atmosféry, nazývanej ionosféra. Pri skúmaní zemskej atmosféry sa využívajú rôzne priame či nepriame prístupy. Jedným zo spôsobov, ako skúmať vyššie časti zemskej atmosféry je pozorovaním optického fenoménu nazývaného airglow, vďaka ktorému je možné pozorovať, merať a študovať dynamiku hornej časti zemskej atmosféry a toho aký vplyv naň majú rôzne externé procesy. Airglow pripomína polárnu žiaru, prejavuje sa ako slabé svetelné žiarenie, no na rozdiel od polárnej žiary je prítomný vo všetkých geografických šírkach. Vznik tohoto fenoménu je ovplyvnený sériou rôznych atmosférických procesov, no jeho primárnym zdrojom vzniku je ultrafialové žiarenie pochádzajúce zo Slnka. Nanešťastie, airglow nie je zo zemského povrchu pozorovateľný voľným okom, práve preto Slovenská Akadémia Vied na Slovensku v súčasnosti prevádzkuje pozorovaciu stanicu, nazývanú AMON-ES, ktorej úlohou je práve pozorovanie, meranie a štúdium toho fenoménu, v korelácií s inými kozmickými a atmosférickými procesmi. Dlhodobým cieľom tohto projektu je automatizácia procesu zbierania, ukladania a spracovania týchto dát a zároveň je to aj hlavným cieľom tejto diplomovej práce. V nasledujúcich kapitolách tejto práce sa pokúsime čitateľovi predstaviť základné informácie o jave airglow, bližšie popísať proces jeho vzniku a ukázať, ako je možné vďaka jeho štúdiu nepriamo pozorovať atmosféru v špecifickej nadmorskej výške. Popíšeme a ukážeme, ako vyzerá pozorovacia stanica AMON-ES a z akých komponentov sa skladá. V súčasnosti je využívanie cloudových služieb stále populárnejšie a dostupnejšie. Využívanie cloudových služieb so sebou dokáže priniesť mnoho výhod, ako napríklad zníženie nákladov pri zaobstarávaní vlastného hardvéru

či možnosť rýchlejšie a flexibilnejšie reagovať na nové požiadavky. Práve preto je jednou z požiadaviek pri navrhnúť softvérové riešenie pre pozorovaciu stanicu AMON-ES využitie cloudovej platformy AWS. V tejto práci sa preto ďalej sústredíme na návrh architektúry a implementáciu tohoto softvérového riešenia práve na tejto cloudovej platforme. Pri tomto návrhu sa budeme nie len snažiť vytvoriť robustnú architektúru aplikácie, ale aj dbať na to, aby ďalší študenti či zamestnanci Slovenskej Akadémie Vied vedeli túto aplikáciu jednoducho rozvíjať a upravovať, keďže tak ako väčšina softvérových riešení, aj aplikácia AMON-ES je kontinuálnym projektom, pri ktorom veríme, že sa bude stále zlepšovať.

# 1 Fyzikálna podstata javu airglow

Táto diplomová práca sa zaoberá javom airglow a systémom na zbieranie, monitorovanie a automatizáciu procesu zberu dát o tomto jave na Slovensku. V tejto kapitole predstavíme základné informácie o tomto jave a vysvetlíme jeho fyzikálnu podstatu. Následne popíšeme jednotlivé typy tohto javu, ktoré je možné pozorovať na nočnej oblohe a zhrnieme prínos jeho štúdia ako aj aktuálne výskumné projekty, ktoré s airglow súvisia.

## 1.1 Airglow

Airglow je optický fenomén vznikajúci vo vrchných častiach atmosféry, ktorý je možné pozorovať, ako jemnú žiaru špecifickej farby vyskytujúcu sa na celej nočnej oblohe. Airglow nie je zo zemského povrchu pozorovateľný voľným okom, preto sa pri jeho štúdiu využíva rad vysokocitlivých optických zariadení, ktoré sú schopné ho zachytiť. Na obrázku 1.1 je možné vidieť airglow zachytený z Medzinárodnej vesmírnej stanice, na ktorom vidno zelenú a červenú žiaru obklopujúcu horizont Zeme. [1]

Airglow svojím vzhľadom môže na prvý pohľad pripomínať polárnu žiaru, no sú to iné javy vznikajúce z iných príčin. Airglow vzniká primárne v dôsledku rekombinácie atómov a molekúl atmosféry počas noci, ktoré sú počas dňa ožarované ultrafialovým svetlom zo Slnka. Na rozdiel od polárnej žiary, ktorú je možné pozorovať v blízkosti magnetických polí Zeme, je airglow prítomný na celej nočnej oblohe vo všetkých geografických šírkach, čo spôsobuje, že na nočnej oblohe je stále prítomné svetlo. [1]

Airglow je možné deliť na základe viacerých faktorov, medzi ktoré patrí napríklad čas výskytu počas noci alebo zdroja emitujúceho svetla. Zaujímavosťou je, že jednotlivé farby žiary airglow na nočnej oblohe zodpovedajú špecifickým vlnovým dĺžkam alebo farbám svetla, ktoré emitujú, vďaka čomu je možné vďaka pozorovaniu tohto javu sledovať špecifickú vrstvu atmosféry a deje, ktoré sa v nej odohrávajú. [1]

Obr. 1.1: Obrázok airglow z ISS, kde je možné na ľavej strane vidieť polárnu žiaru a na pravej strane Airglow [1]

## 1.2 Atmosféra Zeme

Na lepšie pochopenie javu airglow je podstatné spomenúť základné delenie jednotlivých vrstiev Zemskej atmosféry spolu s ich stručnou charakteristikou. Toto teoretické povedomie pomôže lepšie pochopiť fyzikálnu podstatu javu airglow, rovnako ako aj jednotlivé variácie tohoto javu. V neposlednom rade je uvedenie vlastností vrstiev atmosféry, napríklad v kontexte prenosu rádiových vĺn alebo v kontexte fungovania navigačných technológií, prínosné ako úvod do motivácie, prečo je podstatné tento jav pozorovať a študovať.

Zemskú atmosféru Zeme je možné deliť na základe rôznych charakteristík, pričom asi najznámejšie je rozdelenie na základe charakteristickej nadmorskej výšky, ktoré vyzerá nasledovne [1]:

- **Troposféra** - Najhustejšia vrstva Zemskej atmosféry, obsahujúca väčšinu vlhkosti atmosféry ako aj väčšinu vzduchu potrebného pre fotosyntézu rastlín a dýchanie živočíchov. Teplota v tejto vrstve primárne závisí od prenosu tepla z povrchu zeme, a preto sa zvyčajne teplota znižuje so zvyšujúcou sa výškou. V tejto časti atmosféry takisto prebieha väčšina dejov spojených s počasím. Rozprestiera sa vo výškovom intervale 0km - 12km.

[1] Zdroj obrázku: https://www.nasa.gov/image-feature/an-aurora-dimly-intersects-with-earths- airglow

- **Stratosféra** Ozónová vrstva atmosféry je jednou z kľúčových vrstiev, ktorá chráni život na Zemi pred škodlivými účinkami ultrafialového žiarenia. Jej hlavnou zložkou je ozón, ktorý sa vyskytuje v najvyšších koncentráciách spomedzi všetkých atmosférických vrstiev práve v stratosfére. Táto vrstva sa nachádza vo výškovom intervale medzi 12 km a 50 km nad povrchom Zeme.

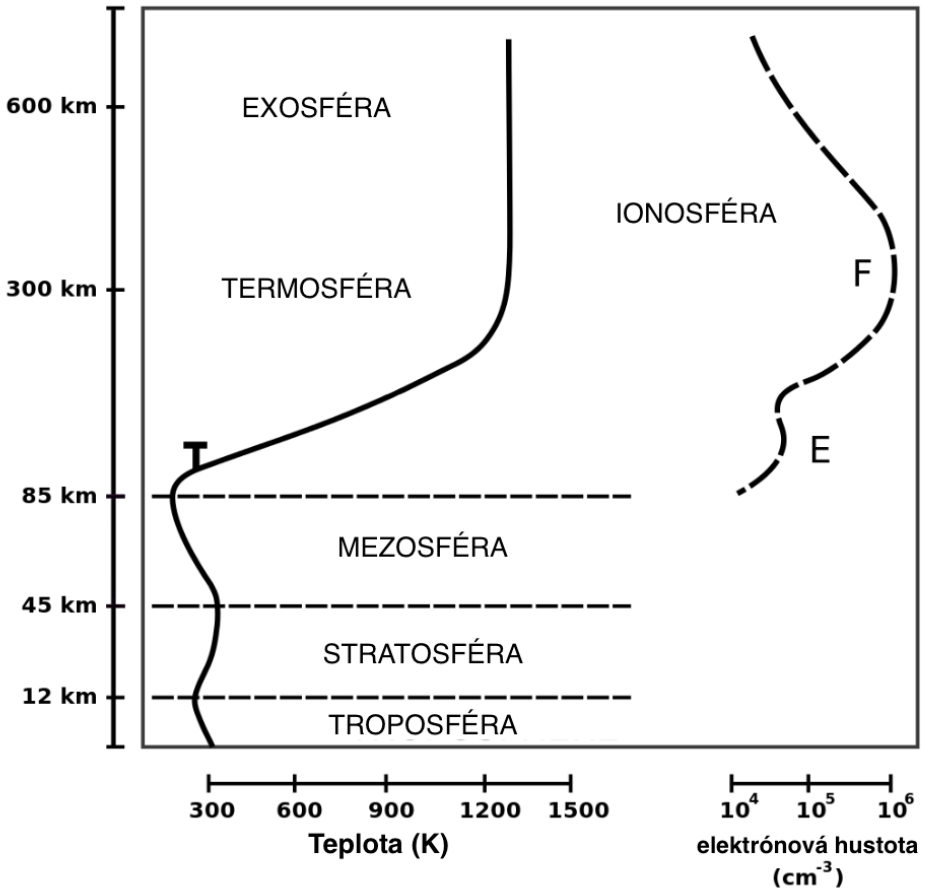
- **Mezosféra** Mezosféra je vrstvou Zemskej atmosféry, v ktorej s nadobúdajúcou výškou výrazne klesá teplota. Teplota na vrchole mezosféry nadobúda hodnoty približne -80 stupňov Celzia. Rozprestiera sa vo výškovom intervale 50km - 80km.

- **Termosféra** Termosféra sa rozprestiera vo výškovom intervale 80km - 700 km. Spodnú časť tejto vrstvy tvorí Ionosféra, ktorú si priblížime bližšie v nasledujúcej sekcii. Hustota tejto časti atmosféry je veľmi riedka a je charakteristická postupným nárastom teploty so zvyšujúcou sa nadmorskou výškou.

- **Exosféra** je najvrchnejšou časťou Zemskej atmosféry s veľmi malou hustotou častíc, ktoré môžu uniknúť do okolitého vesmíru. Rozprestiera sa vo výškovom intervale 700km - 10000km.

Na obrázku 1.2 je možné vidieť charakteristiku teploty s nadmorskou výškou. Ako môžeme vidieť, teplota v atmosfére sa mení v závislosti od výšky. V troposfére, ktorá je najnižšou vrstvou atmosféry a kde sa nachádza väčšina života našej planéty, teplota klesá s nadobúdajúcou výškou. Toto klesanie teploty je spôsobené tlakom, ktorý klesá s výškou, a preto aj teplota klesá. V stratosfére, ktorá sa nachádza nad troposférou, teplota opäť klesá s nadobúdajúcou výškou, ale potom sa začína zvyšovať. Toto zvyšovanie teploty je spôsobené prítomnosťou ozónu v tejto vrstve, ktorý absorbuje ultrafialové lúče slnka a tým zvyšuje teplotu. V mezosfére, ktorá sa nachádza nad stratosférou, teplota opäť klesá s nadobúdajúcou výškou. Tento pokles teploty je spôsobený nedostatkom atmosférického tlaku a chýbajúcim ozónom v tejto vrstve. V termosfére, ktorá sa nachádza nad mezosférou, teplota opäť stúpa s nadobúdajúcou výškou. Toto zvyšovanie teploty je spôsobené vysokou energiou, ktorá prichádza zo slnečného žiarenia. Nakoniec v exosfére, ktorá sa nachádza najvyššie v atmosfére, teplota výrazne klesá s nadobúdajúcou výškou.

Ďalším faktorom, ktorý definuje zemskú atmosféru, je prítomnosť a množstvo elektrického náboja. Časť zemskej atmosféry, v ktorej je prítomnosť elektrického



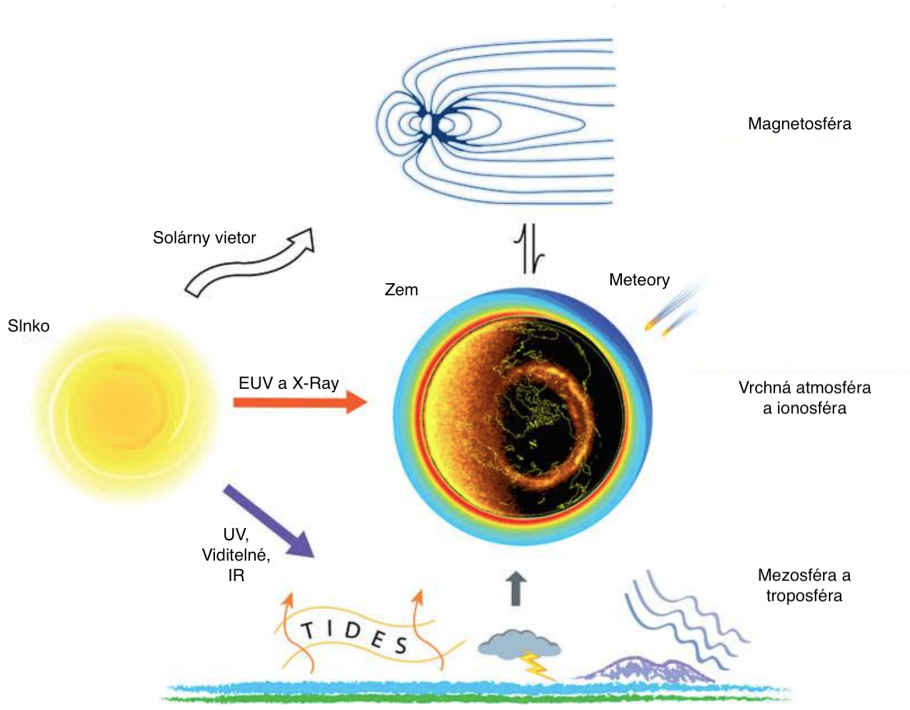

Obr. 1.2: Vľavo: Teplotná charakteristika neutrálnej atmosféry vzhľadom na nadmorskú výšku.
Vpravo: Charakteristika množstva voľných elektrónov vzhľadom na nadmorskú výšku v Ionosfére [1] (texty obrázku boli preložené)

náboja najvyššia, sa nazýva ionosféra. Ionosféra tvorí rozhranie medzi nižšími a hustejšími časťami Zemskej atmosféry a vesmírom, pričom tvorí ochrannú vrstvu pred ultrafialovým žiarením pochádzajúcim zo Slnka, ktoré na rozdiel od nabitých častíc ako elektróny alebo ióny, nie je ovplyvňované magnetosférou.

Na obrázku 1.2 je možné vidieť výškové zobrazenie atmosféry v spomenutých častiach. Ako je vidieť, ionosféra sa nachádza vo výškovom pásme okolo 85 km a prechádza cez termosféru a exosféru. Na obrázku je rovnako možné vidieť aj charakteristiku elektrického náboja v ionosfére voči nadmorskej výške a označenia častí ionosféry na E a F. E-vrstva sa nachádza v najnižšej časti ionosféry a obsahuje relatívne malé množstvo ionizovaných častíc, zatiaľ čo F-vrstva sa nachádza v najvyššej časti ionosféry a obsahuje najväčšie množstvo ionizovaných častíc.

## 1.3 Vznik javu airglow

Airglow vzniká emitovaním svetla excitovaných atómov a molekúl. Zdroj excitácie týchto molekúl je zapríčinený absorpciou svetelného žiarenia zo Slnka či už priamo alebo nepriamo. Pôvod tohto javu, ako aj jeho intenzita úzko súvisí s kompozíciou a koncentráciou neutrálnych atómov v atmosfére, ktoré závisia od aktuálneho stavu zemskej atmosféry, a teda aj od procesov, ktoré tento stav priamo či nepriamo ovplyvňujú [2]. Zemskú atmosféru môžu ovplyvňovať rôzne javy, ktoré prichádzajú z vesmíru a ovplyvňujú hornú atmosféru (slnečné žiarenie, slnečný vietor a účinky v magnetosfére) alebo prichádzajúce zo zemského povrchu (účinky ako blesky, slapové javy, planetárne vlny a atmosférické gravitačné vlny v mezosfére a troposfére). Na obrázku 1.3 je možné vidieť zhrnutie rôznych procesov, ktoré ovplyvňujú stav zemskej atmosféry.[3]




Obr. 1.3: Ilustrácia fyzikálnych javov, ktoré vplývajú na stav atmosféry [4]

Jeden z týchto procesov je absorpcia slnečného EUV (z angl. Extreme ultraviolet) žiarenia vrchnými vrstvami zemskej atmosféry, ktorý sa nazýva fotodisociácia[2]. Tento proces spôsobuje štiepenie kovalentných väzieb molekúl fotónom a rozdelenie molekúl kyslíka na dva samostatné atómy kyslíka. Vyjadrenie tohto procesu je možné vidieť v rovnici 1.1 [3], kde $hv$ predstavuje fotón so špecifickou energiou potrebnou pre rozpad kovalentných väzieb kyslíka.

$$O_2 + hv \rightarrow O + O + energia \tag{1.1}$$

Tento proces je kľúčový, keďže prítomnosť atómov kyslíka v atmosfére je hlavným prispievateľom procesu fotoionizácie, teda tvorby elektricky nabitých častíc v ionosfére [1]:

$$O + hv \rightarrow O^+ + e^- \tag{1.2}$$

V rovnici 1.2[2] $O^+$ predstavuje ión, teda elektrický nabitý atóm kyslíka, $e^-$ predstavuje voľný elektrón a $hv$ predstavuje fotón so špecifickou energiou potrebnou pre ionizáciu. Koncentrácia atómov kyslíka v horných častiach atmosféry ovplyvňuje ionosféru dvoma spôsobmi. Po prvé, koncentrácia atómov kyslíka ovplyvňuje množstvo elektricky neutrálneho plynu dostupného pre fotoionizáciu. Po druhé, koncentrácia atómov kyslíka určuje hĺbku, do ktorej môže slnečné EUV žiarenie prenikať [2].

Ďalším procesom, ktorý ovplyvňuje vrchnú atmosféru, je jej ohrievanie slnečným žiarením, v dôsledku čoho dochádza k jej expanzii. Oba tieto procesy - pohltenie EUV žiarenia a ohrievanie vrchnej atmosféry - vytvárajú rôzne variácie v horných častiach zemskej atmosféry a závisia od faktorov, ako je pozícia Zeme voči Slnku počas roka, ale aj počas dňa. [3]

Medzi ďalšie procesy ovplyvňujúce vrchnú časť zemskej atmosféry patria:

- slnečný vietor - pri tomto procese energeticky nabité častice prichádzajúce zo Slnka interagujú s magnetickým poľom Zeme. Tieto častice vytvárajú silné elektrické prúdy a interagujú s vrchnou časťou atmosféry primárne vo vysokých zemepisných šírkach.
- kozmické žiarenie,
- meteory,

pričom kozmické žiarenie a meteory majú len malý vplyv. [3]

Mimo procesy, ovplyvňujúce vrchnú atmosféry zhora - z vesmíru - existujú aj také, ktoré ju ovplyvňujú zdola. Medzi tieto procesy patrí napríklad [3]:

- elektrické výboje - blesky,
- slapové javy (príliv a odliv),
- planetárne vlny,

- planetárne gravitačné vlny.

Ako je vidieť, vrchná časť atmosféry je dynamický systém, ktorý je ovplyvňovaný nielen kozmickými faktormi, ale aj počasím spodnej atmosféry Zeme. Všetky tieto procesy priamo či nepriamo ovplyvňujú stav atmosféry a môžu mať vplyv na vznik airglowu, keďže pôvod tohto javu, ako aj jeho intenzita úzko súvisí s kompozíciou a koncentráciou neutrálnych atómov v atmosfére.

V závislosti na atóme či molekule, ktorá prispieva k tvorbe airglow patrí reprezentujúca špecifická vlnová dĺžka svetla tohto žiarenia ako aj charakteristická nadmorská výška, v ktorej vzniká. Vďaka tomu vznikajú akési vrstvy javu airglow. Ukážku týchto vrstiev je možné vidieť na obrázku 1.4. Každá vrstva je charakteristická svojim pôvodom - teda atómom či molekulou, z ktorej emitované žiarenia pochádza - nadmorskou výškou, v ktorej žiarenia vzniklo ako aj špecifickou vlnovou dĺžkou emitovaného svetla.[3]

Ako príklad si uvedieme žiarenie airglow s vlnovou dĺžkou 630.0 nm, nazývané aj červený airglow. Toto žiarenie pochádza z časti atmosféry v rozmedzí 250-300 km. Táto výšková oblasť je charakteristická vysokou intenzitou nabitých častí v ionosfére 1.2, čo ovplyvňuje intenzitu tohto žiarenia. Najvýraznejším procesom prispievajúcim k tvorbe tohto žiarenia je disociatívna rekombinácia častíc [3]:

$$O^+ + O_2 \rightarrow O_2^+ + O,$$

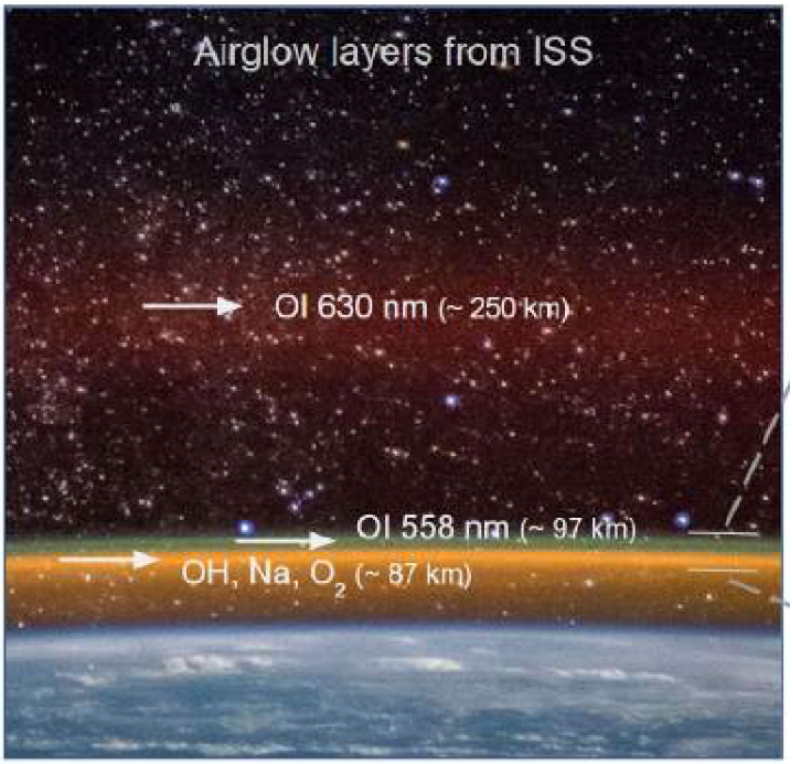
$$O_2^+ + e^- \rightarrow O^*(^1D) + O,$$

$$O^*(^1D) \rightarrow O(^3P_2) + hv(630nm)$$

Obdobným spôsobom sa dá pokračovať a opísať proces vzniku zvyšných airglow vrstiev.

## 1.4 Prínos a využitie štúdia airglow

Ako bolo opísané v predchádzajúcich častiach kapitoly, airglow je proces, ktorý vzniká v horných častiach zemskej atmosféry. Vrchná časť zemskej atmosféry je ovplyvnená širokou škálou procesov, ktoré ovplyvňujú jej aktuálny stav a zdroj týchto procesov môže pochádzať ako z vesmíru, tak aj zo zemského povrchu. Svetelné žiarenie javu airglow je charakteristické špecifickými vlnovými dĺžkami svetla, ktoré sú charakteristické pre nadmorskú výšku, v ktorej toto svetlo vzniká, vďaka čomu je možné pozorovanie stavu ionosféry v konkrétnej nadmorskej výške. Airglow si je možné predstaviť ako množinu procesov vo výškach 80 - 300 km.




Obr. 1.4: Zobrazenie vrstiev javu airglow. Každá vrstva je špecifická svojim chemickým pôvodom, nadmorskou výškou, v ktorej vzniká, ako aj vlnovou dĺžkou emitovaného žiarenia.[3]

Vďaka tomuto zobrazeniu je tak možné študovať doposiaľ stále nedostatočne pochopené prostredie rozhrania medzi vesmírom a Zemou. Toto štúdium prispieva k lepšiemu pochopeniu dynamiky procesov, ktoré ovplyvňujú nielen termosféru a ionosféru, ale aj celú atmosféru, ktorá bezprostredne ovplyvňuje ľudskú aktivitu a celkovo život na Zemi. [2, 5]

# 2 Počiatočný stav projektu

Táto diplomová práca je pokračovaním v snahe vylepšiť proces automatizácie merania, zberu a spracovania dát o jave airglow na Slovensku. V tejto kapitole sa zameriame na súčasný stav projektu. Stručne opíšeme fungovanie pozorovacej stanice určenej na zber airglow dát a súčasný stav aplikácie na vizualizáciu a prístup k týmto dátam. Popíšeme existujúce časti aplikácie spolu s technológiami, ktoré boli pri ich vývoji využité.

## 2.1 Pozorovacia stanica AMON-ES

Oddelenie kozmickej fyziky, Ústavu experimentálnej fyziky, Slovenskej akadémie vied v. v. i. prevádzkuje meraciu stanicu na pozorovanie airglowu na Astronomickom observatóriu v Kolonickom sedle nazývanú AMON-ES (Airglow MONitor - Extended Station). Hlavnými úlohami AMON-ES sú:

- získanie širšieho porozumenia javov, ktoré ovplyvňujú vznik airglowu vo vyšších častiach atmosféry. [6],
- odfiltrovanie všetkých zdrojov kontaminácie nameraných airglow dát (poveternostné podmienky, astronomické objekty, chyby spôsobené technickým vybavením) [6],
- porovnanie pozorovacích dát s dátami z ionosferických meraní s využitím signálov z GNSS (z angl. Global Navigation Satellite System) [3]

Na obrázku 2.1 je možné vidieť celú pozorovaciu stanicu AMON-ES so všetkými jej komponentami. Medzi tieto komponenty patria [6]:

- prevádzková platforma
- AAC z angl. AMON All-Sky Camera
- Zariadenie AMON


Obr. 2.1: Zobrazenie pozorovacej stanice AMON-ES

- Prijímač GNSS (z angl. Global Navigation Satellite System)
- Zaradenie na monitorovanie počasia (z angl. Weather Monitoring system)
- Kamera na sledovanie TLE (z angl. Transient Luminous Events) [7]

Platforma zariadenia AMON sa skladá z viacerých častí. Medzi najdôležitejšie časti patrí nosná konštrukcia, ktorá siaha do výšky troch metrov a je zložená z pevného kovového rámu, ktorý je povrchovo upravený proti korózii. Tento rám


Obr. 2.2: Zobrazenie GNSS prijímača pozorovacej stanice AMON-ES [6]

je upevnený v betónovej konštrukcii, aby bola zabezpečená dostatočná stabilita zariadenia. [6].

Medzi ďalšie časti prevádzkovej platformy patria elektrické rozvody, internetové rozvody a zariadenia potrebné pre internetové pripojenie, server určený na spracovanie dát z meraní. Všetky tieto zariadenia sú umiestnené v kovovej skrine a izolované epoxidom pre ochranu proti vode a vlhkosti.[6]. Na obrázku 2.3 je možné vidieť jednotlivé časti zariadenie pred inštaláciu skrinky na kovovú platformu.

Jednou z najdôležitejších častí stanice AMON-ES je celooblohová kamera, ktorá je určená na zachytávanie špecifických vlnových dĺžok svetla, zodpovedajúcim jednotlivým typom airglow emisie, respektíve zodpovedajúcim vrstvám, v ktorých žiarenie airglow vzniká. Táto kamera je vyrobená spoločnosťou Moravian Instruments, má rozlíšenie 2056 × 2062 pixlov, je vybavená Sigma fish-eye objektívom s ohniskovou vzdialenosťou 4,5 mm, uhlom záberu 180 stupňov, clonou f/2.8 a špeciálnymi filtrami, ktoré prepúšťajú len špecifické vlnové dĺžky svetla. Tieto filtre sú umiestnené nad optickým snímačom kamery a je možné ich automatizovane meniť. [3]. Na obrázku 2.4 je možné v ľavom hornom rohu vidieť vyššie spomínané filtre. Na obrázku 2.5 je možné vidieť graf zobrazujúci jednotlivé filtre a im prislúchajúce vlnové dĺžky svetla.

Zariadenie AAC automaticky monitoruje nočnú oblohu v pravidelných 5 mi-


Obr. 2.3: Zobrazenie základných prevádzkových komponentov systémov AMON-ES. [3]

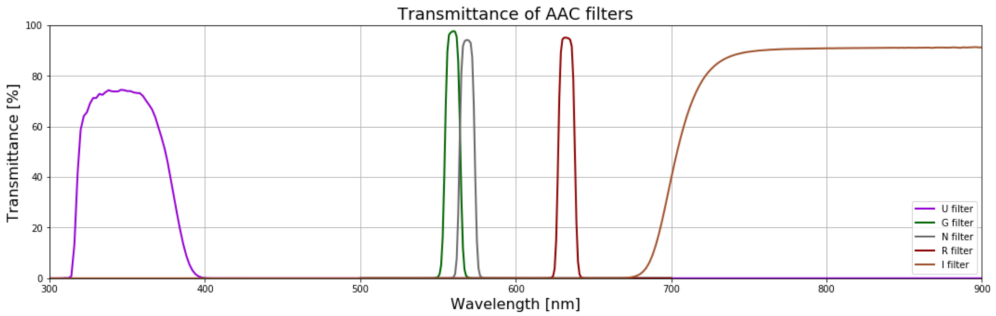
nútových intervaloch intervaloch. Pre každé meranie sú zachytené snímky oblohy skrze všetky prítomné svetelné filtre. Získané dáta sa ukladajú na lokálny server a nasledujúce ráno po pozorovaní sa odosielajú automaticky na vzdialený disk, ktorý sa nachádza na pracovisku Ústavu experimentálnej fyziky SAV, v. v. i. v Košiciach. Všetky časti monitorovacieho procesu je možné manuálne obsluhovať aj na diaľku, vďaka prevádzkovému serveru, ktorý je prítomný v zariadení, čo umožňuje meniť parametre merania bez potreby fyzického prístupu k zariadeniu.

Pozorovacia stanica AMON-ES ďalej zahŕňa monitorovací systém počasia, ktorý pozostáva z kamery Raspberry-Pi s rozlíšením 2592 × 1944 pixlov, infračerveného snímača na detekciu oblakov a zo samotného mikropočítača Raspbery-Pi. Zábery oblohy sa automaticky získavajú každých 10 minút a ukladajú sa na hlavný počítač. Nízka citlivosť kamery v noci sa kompenzuje pomocou infračerveného snímača, ktorý meria teplotu oblohy a pomáha určovať kvalitu pozorovacích podmienok. [3]

Ďalšou súčasťou pozorovacej stanice je zariadenie GNSS prijímač. Toto zariadenie prijíma signál z GPS a Galileo satelitov a dokáže kontinuálne zbierať dáta o ionosférickej scintilácií, ktoré sa v korelácii s dátami zo zariadenia AMON a AAC využívajú pri štúdiu javu airglow.

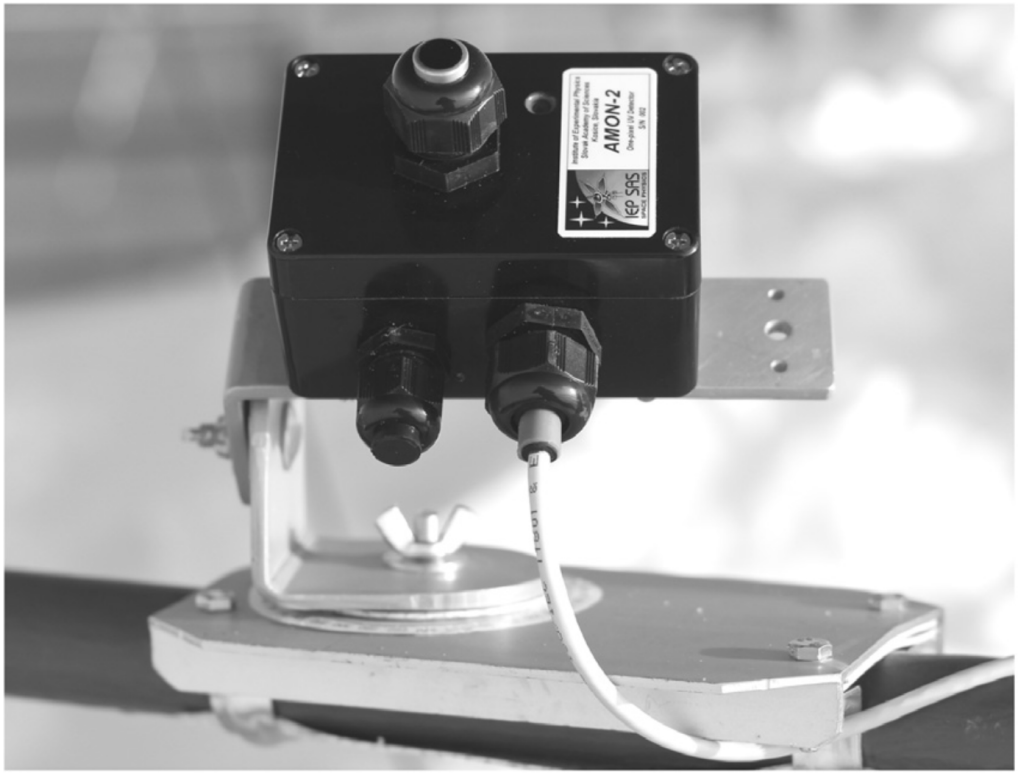
Ďalšou časťou pozorovacej stanice AMON-ES je zariadenie AMON. AMON je jedno-pixlový detektor určený na monitorovanie svetelného pozadia nočnej oblohy v blízkosti ultrafialového spektrálneho pásma, ktorý bol vyvinutý priamo na ÚEF SAV. Toto zariadenie je konštruované tak, aby zvládalo dlhodobú, konti-


Obr. 2.4: Zobrazenie celooblohovej kamery AAC pred inštaláciou. [3]




Obr. 2.5: Zobrazenie filtrov AAC kamery a zodpovedajúcim vlnovým dĺžkam. [3]

nuálnu prevádzku a aby odolával vonkajším poveternostným podmienkam. Meranie je vykonávané pomocou vysoko citlivého fotosenzora s nízkou spotrebou energie. Mimo to je zariadenie AMON doplnené o ďalšie senzory ako barometer, 3D akcelerometer, gyroskop, magnetometer či GPS prijímač, ktoré umožňujú zariadeniu dlhodobú kontinuálnu prevádzku. Celkovo zariadenie AMON váži 575 gramov a je úplne vodotesné [5]. Zariadenie AMON je možné vidieť detailne na obrázku 2.6.




Obr. 2.6: Ukážka zariadenia AMON [5]

## 2.2 Východiskový stav softvérového riešenia pozorovacej stanice AMON-ES

Mimo iné je cieľom práce doplnenie chýbajúcich funkcií aplikácie, ktoré doteraz neboli implementované ako aj jej nasadenie do produkčnej prevádzky. V tejto časti sa budeme bližšie zaoberať počiatočným stavom aplikácie AMON-ES, jej architektúrou a využitými technológiami.

### 2.2.1 Využité technológie

Existujúca aplikácia AMON-ES je rozdelená na serverovú a grafickú časť, pričom obe boli vytvorené s pomocou programovacieho jazyka Python s využitím viacerých technológií, ktoré si bližšie rozoberieme.

Knižnica **Flask** bola základnou využitou technológiou serverovej časti aplikácie. Flask je Python knižnica pre tvorbu webových aplikácií a aplikačných rozhraní. Je jednoduchá na použitie a má malú veľkosť, čo je výhodou pri tvorbe malých a stredne veľkých aplikácií. V porovnaní s alternatívami, ako je Django alebo Tornado, je Flask jednoduchší na použitie, no môže poskytovať menej funkcií, čo môže byť problémom pri tvorbe veľkých aplikácií. V kontexte aplikácie

AMON-ES, Flask slúži na implementáciu REST rozhrania určeného pre prístup k dátam AMON-ES. [8]. V rámci grafickej časti aplikácie je knižnica Flask využitá pre poskytovanie Dash aplikácie.

Ako nadstavba pri tvorbe koncových bodov serverovej časti aplikácie bola využitá technológia **Flask RestPlus**, čo je knižnica pre tvorbu RESTful API v jazyku Python. Táto knižnica je určená pre Flask a umožňuje jednoducho vytvoriť a spravovať RESTful API. Používa sa na automatizáciu procesu tvorby REST API, vrátane automatickej dokumentácie pomocou nástroja OpenAPI 3, validácie či prípadnej autentifikácie. Táto knižnica je veľmi flexibilná a umožňuje jednoducho prispôsobiť API pre potreby konkrétnej aplikácie. [9] [10] [11] V súčasnosti je táto knižnica bez aktívneho vývoja pričom jej priamou alternatívou je knižnica Flask RestX.

Dokumentácia realizovaná pomocou nástroja Flask RestPlus využívala **OpenAPI 3**, čo je otvorený štandard pre dokumentovanie API, ktorý umožňuje jednoduché a prehľadné definovanie REST API. Pomocou OpenAPI 3 je možné zadefinovať vstupné a výstupné dáta, kontrolovať integritu API a automatizovať procesy, ako napríklad generovanie grafickej verzie dokumentácie a testovania. Štandard OpenAPI 3 je platformovo nezávislý a podporovaný vo viacerých softvérových rámcoch. Výhodou použitia OpenAPI 3 je jednoduchá dokumentácia REST rozhrania a ľahká integrácia s nástrojmi, ako napríklad Swagger UI, ktorý na základe OpenAPI 3 schémy dokáže vytvoriť webovú stránku zobrazujúcu jednotlivé koncové body aplikácie spolu s ich vstupnými parametrami a možnosťou ich otestovania. [12]

Dáta, ktoré serverová časť aplikácie sprístupňuje boli uložené v **SQLite 3**, čo je jednoduchá súborová relačná databáza, ktorá má nízke požiadavky na systémové zdroje a umožňuje vývojárom vytvoriť aplikácie s databázou bez nutnosti špeciálneho servera. [13] Medzi hlavné výhody tejto technológie patria nízke nároky na systémové zdroje, jednoduchosť implementácie a nízka cena. Tieto vlastnosti umožňujú rýchle prototypovanie aplikácií. Nevýhodou je však nižší výkon oproti plnohodnotným databázam a obmedzenie počtu používateľov. [13]

Základom grafickej časti aplikácie AMON-ES je technológia **Dash**, čo je knižnica pre tvorbu interaktívnych webových aplikácií v jazyku Python. Táto knižnica je postavená na technológiách ako Flask, Plotly, ktoré umožňujú jednoducho vytvoriť jednoduché grafické rozhrania pre vizualizáciu dát. Dash poskytuje aj už pripravené komponenty, ako sú grafy, tabuľky a mapy, čo uľahčuje prácu s dátami a umožňuje ich ľahko zobraziť na webových stránkach. Tiež umožňuje jednoducho pridávať interaktívne prvky, ako sú tlačidlá a vstupné pole, čo uľahčuje

interakciu s dátami. [14] [15].

### 2.2.2 Serverová časť aplikácie

Ako už bolo vyššie spomenuté serverová časť aplikácie AMON-ES bola vytvorená s pomocou programovacieho jazyka Python a s využitím technológií ako Flask a Flask RestPlus. Základnou úlohou serverovej časti aplikácie bolo sprístupňovanie dát s využitím REST rozhrania grafickej časti aplikácie ako aj nahrávanie nových metadát o obrázkoch z AAC. Štruktúra serverovej časti aplikácie vyzerá nasledovne:

- **sontroller** - v tejto časti sa nachádzajú definície koncových bodov aplikačného rozhrania spolu s funkciami na serializáciu vstupu a výstupu
- **satabase** - v tomto priečinku je zoskupený kód určený na komunikáciu s databázou aplikácie spolu s definíciou SQL dopytov
- **service** - v tomto priečinku sa nachádzajú zdrojové kódy určené na manipuláciu a transformáciu databázových dát a ich následné sprostredkovanie koncovým bodom aplikácie
- **setup** - obsahuje kód aplikácie určený na nastavenie aplikácie, vrátane nastavení premenných a použitých knižníc.

V rámci aplikácie sú definované tri koncové body, dve určené na prístup ku AMON a AAC dátam, jeden určený na zápis nových metadát o obrázkoch z AAC.

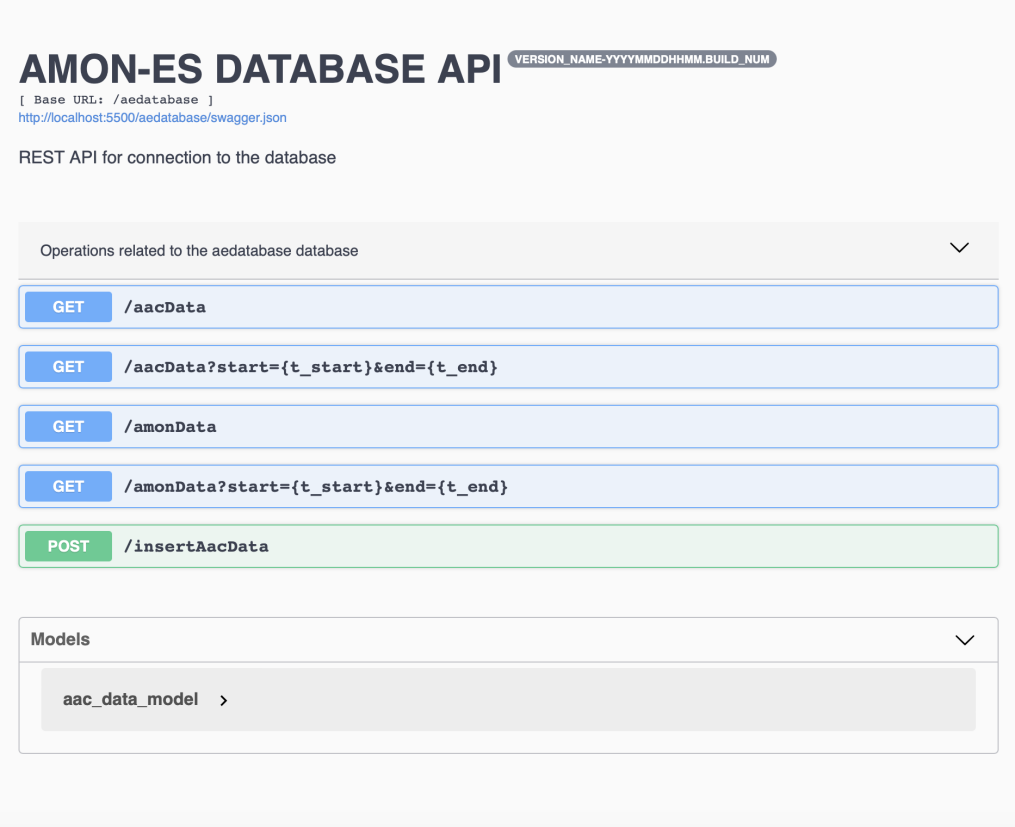
- **Prístup k AMON dátam** - tento koncový bod má dva vstupné parametre: počiatočný a koncový dátum, ktoré určujú časové okno vrátených AMON dát
- **Prístup k AAC dátam** - rovnako ako pri AMON dátach, aj tento koncový bod prijíma počiatočný a koncový dátum, ktoré ohraničujú vrátené údaje o AAC dátach
- **Zapisovanie AAC dát** - tento prístupový bod slúži na zápis nových AAC dát do databázy.

Pričom pri oboch koncových bodoch aplikácie určených na získavanie dát sú vstupné parametre ohraničujúce časové okno nepovinné.

Na obrázku 2.7 je možné vidieť ukážku webovej stránky zobrazujúcej OpenApi 3 dokumentáciu dostupnú pre serverovú časť aplikácie. Na obrázku je

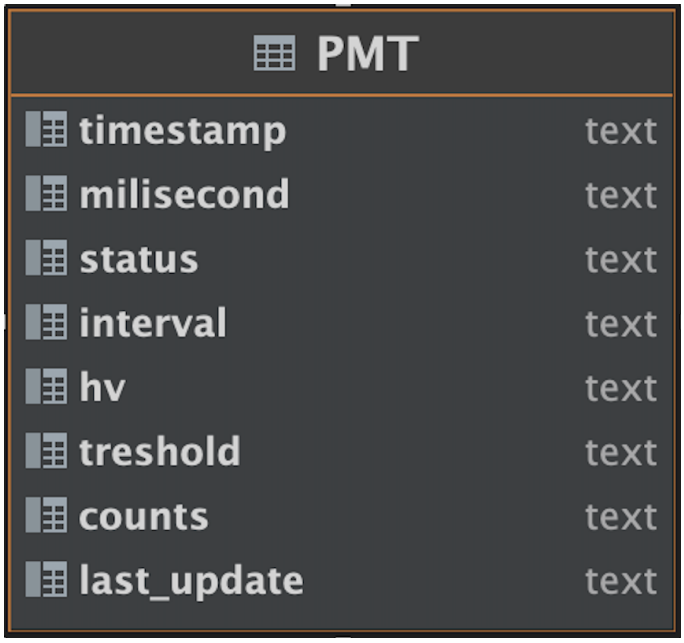
možné vidieť duplicitne zobrazené koncové body pre prístup k jednotlivým dátam, čo je s najväčšou pravdepodobnosťou spôsobené neúplnou alebo nesprávnou konfiguráciou nástroja Flask RestPlus.

Ukladanie AMON dát a AAC metadát aplikácie je realizované pomocou súborovej databázy SQLite 3. Tieto dáta sú uložené v dvoch tabuľkách pričom ich schému je možné vidieť na obrázkoch 2.8 a 2.9. Tabuľka PMT reprezentuje AMON dáta a tabuľka AACI reprezentuje AAC metadáta. Ako je možné vidieť tabuľka pre metadáta obsahuje len záznam o názve obrázku spolu s triedou (označenie class), ktorá reprezentuje výsledok klasifikácie obrázku. Táto klasifikácia sa získava s využitím samostatného algoritmu, ktorý v súčasnosti nie je súčasťou aplikácie. Rovnako je možné vidieť, že aplikácia neposkytuje koncový bod pre pridávanie AMON dát a pre spustenie samotnej aplikácie je potrebné manuálne tieto dáta pridať do súborovej databázy SQLite 3.

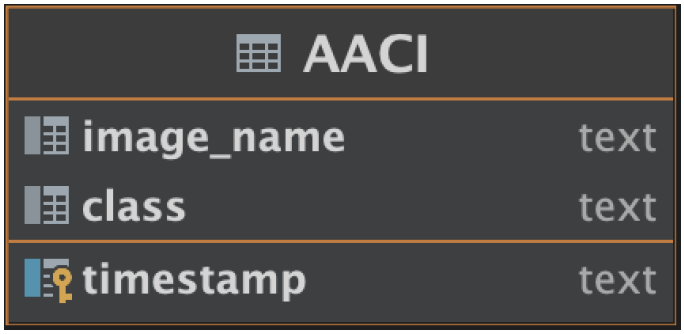



Obr. 2.7: Zobrazenie koncových bodov aplikácie v Swagger UI

Samotné AAC obrázky, ktoré sú zobrazené v grafickej časti aplikácie sú získavané ako statické assety a je potrebné ich mať v súčasnosti uložené v priečinku spolu s kódom serverovej časti aplikácie. Vzhľadom na ich počet a veľkosť rovnako nie sú súčasťou git repozitára serverovej časti aplikácie, a teda je potrebné ich manuálne pridať do projektu pri spúšťaní aplikácie.




Obr. 2.8: Zobrazenie databázovej schémy PMT (Východiskový názov SQL tabuľky) určenej na ukladanie AMON dát




Obr. 2.9: Zobrazenie databázovej schémy AACI (Východiskový názov SQL tabuľky) určenej na ukladanie AAC metadát

V súčasnej dobe nie je serverová časť aplikácie AMON-ES nasadená na žiadnom produkčnom prostredí a je možné spustiť ju len lokálne.

### 2.2.3 Grafická časť aplikácie

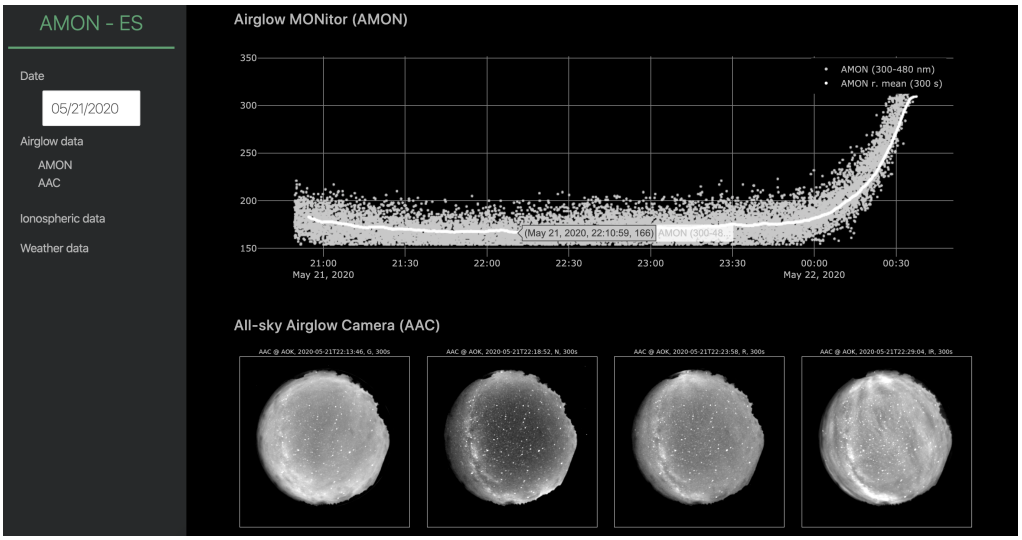
Grafická časť aplikácie slúži na vizuálne sprístupnenie a prezentáciu dát AMON a AAC pomocou interaktívneho webového rozhrania. Táto časť je implementovaná pomocou už spomenutej knižnice Dash, pričom štruktúra grafickej časti aplikácie je podobná tej serverovej a pozostáva z týchto základných častí:

- **controller** - vzhľadom na to, že knižnica Dash interne využíva knižnicu Flask, rovnako ako pri serverovej časti aplikácie, táto časť slúži na definíciu koncových bodov aplikácie. V kontexte grafickej časti sa tu nachádza

len jeden koncový bod určujúci URL, na ktorom bude webová aplikácia prístupná. Okrem definície koncových bodov tu sú definované aj pomocné funkcie určené na prístup k dátam zo serverovej časti aplikácie a na manipuláciu a transformáciu týchto dát,

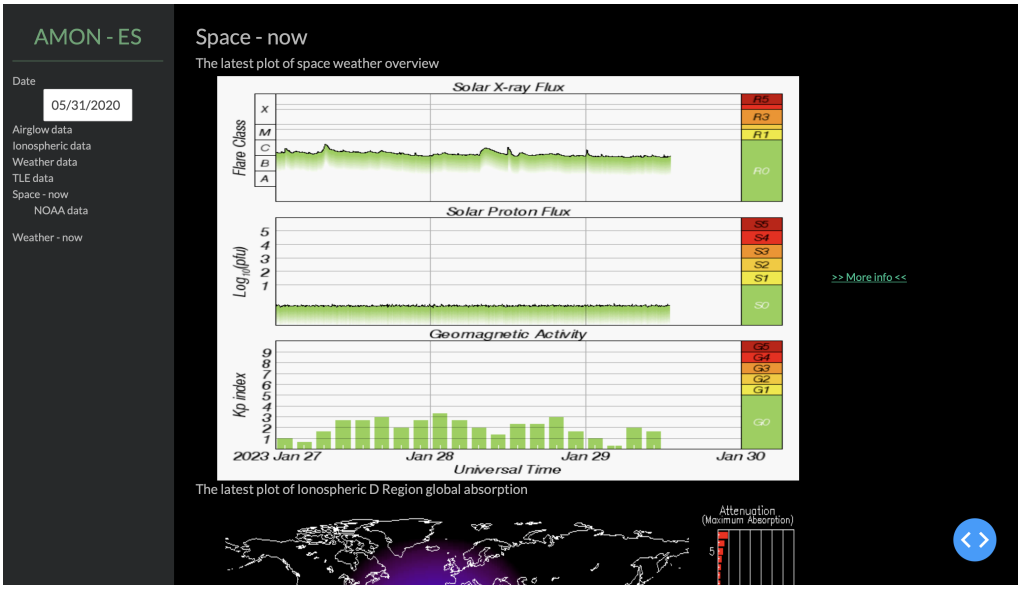
- **setup** - V tejto časti sa nachádzajú konfiguračné časti aplikácie, definície premenných a inicializácia samotnej aplikácie,
- **service** - Tu sa nachádza inicializácia knižnice Dash. Táto časť obsahuje kód definujúci rozloženie webovej stránky, nastavenie kaskádových štýlov a nastavenie premenných, využívaných vo webovej aplikácii.

Na obrázku 2.10 je možné vidieť ukážku grafickej časti aplikácie zobrazujúcu vizualizáciu AMON a AAC dát v podobe grafu a obrázkov pre konkrétny deň. Graf zobrazujúci AMON dáta vykresľuje počet zachytených záznamov (v databázovej tabuľke PMT zobrazené ako counts) vzhľadom na čas. Samotný graf je plne interaktívny a poskytuje možnosť priblíženia dát na základe zvoleného časového rozsahu. AAC obrázky pod grafom sa interaktívne menia vzhľadom na zvolený interval na grafe.




Obr. 2.10: Zobrazenie webovej aplikácie zobrazujúcej AMON-ES a AAC dáta

V rámci aplikácie boli pridané aj vizualizácie tretích strán, ktoré spolu s dátami z pozorovacej stanice v Kolonici poskytujú presnejší a komplexnejší prehľad o aktuálnom stave zemskej atmosféry. Príkladom takto pridanej služby je Space Now, ktorá ponúka zobrazenie aktuálneho stavu solárnej a geomagnetickej aktivity. Na obrázku 2.11 môžete vidieť ukážku stránky, ktorá zobrazuje tieto dáta.




Obr. 2.11: Zobrazenie webovej aplikácie zobrazujúcej Space-now dáta

Podobne, ako v prípade serverovej časti aplikácie AMON-ES aj grafickú časť aplikácie je momentálne možné spustiť len v lokálnom prostredí a zatiaľ nie je nasadená na žiadnu produkčnú platformu. To znamená, že nie je možné ju používať na vzdialených zariadeniach a nie je k dispozícii pre širšie publikum.

# 3 Serverless architektúra

Jedným z hlavných cieľov tejto diplomovej práce je implementovať riešenie pre zber a spracovanie nameraných dát na cloudovú platformu AWS Amazon Web Services. Táto služba ponúka rad služieb, spoločne nazývaných serverless, ktoré sú špecifické tým, že pri ich využívaní neplatíme za hardvér, ale len za spotrebované zdroje. V tejto kapitole sa budeme venovať návrhu riešenia pre aplikáciu AMON-ES, ktorá bude využívať serverless služby. Popíšeme základné služby, ktoré cloudová platforma AWS ponúka spolu s možnosťami ich prepojenia. Tieto služby zahŕňajú napríklad cloudové úložiská, služby na zdieľanie a spracovanie dát, ale aj služby využívané pri strojovom učení. V rámci kapitoly sa sústredíme najmä na tie, ktoré sú v kontexte navrhnutej architektúry aplikácie najrelevantnejšie.

V ďalšej časti predstavíme konkrétny návrh architektúry pre ukladanie, spracovanie a prezentáciu dát získaných z pozorovacej stanice AMON-ES. Ukážeme, ako tieto dáta budú prenášané, ukladané a spracovávané na platforme AWS, a tiež zdôrazníme pozitíva a negatíva tejto architektúry.

## 3.1 Cloudová platforma AWS

AWS je cloudová platforma ponúkajúca širokú škálu globálne dostupných, jednoducho škálovateľných produktov, ktoré sú dostupné v priebehu sekúnd s platobným modelom, v ktorom užívatelia platia len za využívanie daných produktov a nie za hardvérové prostriedky. Medzi tieto produkty patria napríklad služby zamerané na výpočtové, sieťové a mobilné aplikácie, databázové systémy, analytiku, strojové učenie, IoT a mnoho ďalších [16].

Využívanie cloudových produktov v softvérových riešeniach v dnešnej dobe ponúka možnosti ako znížiť vstupné náklady pri nasadení prvého riešenia s možnosťou jednoducho toto riešenie škálovať. Pri nasadzovaní nových produktov na cloudovú platformu teda nie je potrebné vopred kupovať alebo prenajať vlastný hardvér určený na beh aplikácie, ale je možné využívať služby cloudovej plat-

formy, ktoré je možné jednoducho a automaticky meniť na základe potreby. Využívanie cloudových technológii teda prináša rad výhod, medzi ktoré partia najmä [16]:

- **Platba za spotrebované zdroje** - V klasickom prístupe, v ktorom je potrebné pre chod softvérového riešenia kúpiť hardvér určený na jeho chod, je značnou nevýhodou fakt, že je niekedy náročné odhadnúť počet používateľov daného softvérového riešenia. Výsledkom teda môže byť systém, ktorý má príliš veľa nevyužitých hardvérových zdrojov alebo naopak ich má príliš málo. Pri využívaní cloudových technológií je možné tento problém eliminovať, keďže je potrebné zaplatiť len za zdroje, ktoré boli spotrebované, prípadne začať so slabším hardvérom, ktorý je možné následne jednoducho škálovať.
- **Zníženie nákladov na prevádzku** - Využívaním cloudových služieb je možné rovnako redukovať náklady za správu a prevádzku vlastných dátových centier.
- **Globálny dosah** - Vďaka tomu, že AWS prevádzkuje dátové centrá v celom svete je nasadenie aplikácie na inom kontinente jednoduché.

## 3.2 Serverless AWS služby

V tejto časti stručne opíšeme služby dostupné na cloudovej platforme AWS, pričom sa sústredíme najmä na tie, ktoré majú v kontexte práce najväčšie možné využitie. Na začiatok predstavíme pravdepodobne najznámejšie a najviac využívané služby, ktoré AWS ponúka. Následne spomenieme služby, ktoré sú menej univerzálne, no v kontexte tejto práce môžu byť veľmi prínosné.

### 3.2.1 AWS Lambda funkcie

Pravdepodobne najdôležitejšou službou, a to aj mimo kontext serverless, je služba AWS Lambda. AWS Lambda je služba umožňujúca vykonávať špecifický kód rôznych programovacích jazykov v cloudovom prostredí bez potreby správy virtuálneho, či fyzického serveru [16]. Pre svoj chod teda nepotrebuje administráciu žiadnych serverov. Lambda sa často využíva na integráciu viacerých služieb dostupných na platforme AWS, keďže je schopná reagovať na rôzne udalosti, ktoré môžu v týchto službách nastať. Príkladom môže byť spustenie špecifického kódu v službe Lambda pri expirácii alebo nahraní nového objektu do služby AWS S3.

Obdobný spôsobom je možné integrovať takmer všetky AWS služby, vďaka čomu je lambda využívaná takmer vo všetkých cloudových aplikáciach [17].

Úzko súvisiacou službou, využitou pri návrhu serverless architektúry aplikácie AMON-ES, je služba AWS Step Funkcie (z angl. AWS Step Functions). Táto služba umožňuje zreťaziť jednotlivé lambda funkcie za sebou s využitím vyššie spomínaných udalostí, vďaka čomu je možné vytvoriť stavové automaty pozostávajúce z lambda funkcií vykonávajúcich špecifický kód, bez nutnosti administrácie virtuálnych či fyzických serverov [18].

Jednou z veľkých výhod tejto služby je jej cena, ktorá sa odvíja len od spotrebovaných zdrojov počas vykonávania kódu. Medzi faktory patrí dĺžka trvania vykonávaného kódu, veľkosť využitej operačnej pamäte počas behu programu a veľkosť prenesených dát. V rámci ceny je potrebné spomenúť, že v čase písania tejto práce cloudová platforma AWS ponúka obmedzený počet lambda funkcií, ktoré je možné za mesiac spustiť zadarmo, čo zvyšuje atraktivitu ich použitia v cloudových aplikáciach [17].

### 3.2.2 S3

S3 (z angl. Simple Storage Service) je objektové úložisko s mierou perzistencie, ktorá je udávaná cloudovou službou AWS, 99,999999999%. Táto služba ponúka ďalej množstvo doplňujúcich služieb na správu uložených dát, ako napríklad automatické vymazávanie po stanovenom čase, automatická záloha a replikovanie dát, či nastavenie práv pre prístupnosť uložených dokumentov. Služba S3 je dostupná vo viacerých variantách, ktoré sa medzi sebou líšia primárne v dostupnosti uložených dát a cenou. Základné typy tried služby S3 sú nasledovné [19]:

- **S3 Štandard** - Ponúka vysokú mieru perzistencie a dostupnosti dát, rovnako ako nízku prístupovú latenciu a vysokú priepustnosť. Je primárne určená na aplikácie v ktorých sú uložené dáta často dotazované. Táto trieda je vhodná na väčšinu cloudových aplikácií, pričom jednou z nevýhod môže byť vyššia cena v porovnaní s niektorými inými triedami, určenými na špecifické použitie.
- **S3 Intelligent-Tiering** - Táto trieda ponúka možnosť automatického optimalizovania nákladov, tým že uložené dáta kategorizuje a v závislosti na frekvencií prístupu k nim. Dáta, ktoré sú málo dopytované následne presúva do S3 triedy s nižšími nákladmi, no pomalším časom prístupu.
- **S3 Glacier Instant Retrieval** - Táto trieda je určená na archiváciu uložených dát, ktorých frekvencia prístupu je pomerne nízka, no pri ich dopyte

je rovnako potrebná aj nízka latencia. Cena za uložené dáta je v porovnaní s triedou S3 Štandard o 60% nižšia, no pri tejto triede sa platí aj za dáta, ktoré sú dopytované. Príkladom vyžitia môžu byť napríklad medicínske, či zdravotné údaje.

- **S3 Glacier Flexible Retrieval** - Táto archívna trieda je podobná ako predchádzajúca S3 Glacier Instant Retrieval, no jej cena môže ďalej klesnúť o 10% za uložené dáta. Hlavným rozdielom je doba prístupu k údajom, ktorá sa pohybuje v rozmedzí niekoľkých minút až niekoľkých hodín.
- **S3 Glacier Deep Archive** - Najlacnejšou so všetkých S3 tried, pričom je určená na archiváciu dát, pri ktorých sa predpokladá minimálny dopyt (1x - 2x ročne). Najväčším rozdielom je doba prístupu ktorá je stanovená do 12 hodín.

Tabuľka 3.1: Porovnanie parametrov S3 úložiska

| | S3 Standard | S3 Intelligent-Tiering | S3 Standard-IA | S3 One Zone-IA |
|---|---|---|---|---|
| Designed for durability | 99.999999999 | 99.999999999 | 99.999999999 | 99.999999999 |
| Designed for availability | 99.99% | 99.9% | 99.9% | 99.5% |
| Minimum object capacity | N/A | N/A | 128 KB | 128 KB |
| Minimum storage duration | N/A | N/A | 30 days | 30 days |
| Retrieval charge | N/A | N/A | per GB retrieved | per GB retrieved |
| First byte latency | milliseconds | milliseconds | milliseconds | milliseconds |

### 3.2.3 Amazon DynamoDB

Amazon DynamoDB je plne spravovaná NoSQL databáza v cloudovom prostredí, ktorá sľubuje rýchle a spoľahlivé možnosti škálovania, vďaka čomu je ideálnou voľbou pre aplikácie s vysokou mierou škálovateľnosti, prípadne s častými zmenami požiadaviek na kapacitu. Služba DynamoDB sľubuje škálovanie, ktoré umožňuje ukladanie a získavanie ľubovoľného množstva dát s možnosťou zvyšovania

alebo znižovania kapacity bez výpadkov. To znamená, že je možné prispôsobovať kapacitu databázy podľa aktuálnych potrieb a zároveň zabezpečiť spoľahlivý chod aplikácie bez prerušenia pre používateľov. [20]

Jednou z hlavných výhod DynamoDB je to, že umožňuje odloženie administratívnych bremien súvisiacich s prevádzkou a škálovaním distribuovanej databázy. Administratívne úlohy, ako je správa hardvéru, konfigurácia, aktualizácia softvéru a škálovanie, sú prebrané na starosť poskytovateľa, teda cludovej platformy AWS. Medzi ďalšie výhody patrí jednoduché šifrovanie uložených dát, automatické zálohovanie, či automatické mazanie starých a nepotrebných záznamov. [20]

Cena za DynamoDB sa odvíja za zápis a za čítanie jednotlyvých dát, pričom táto služba ponúka dva modely pre správu kapacity čítania a zápisu. Podobne ako v predchádajúcich službách, tak aj DynamoDB poskytoje obmedzenú kapacitu zápisu a čítania zadarmo v rámci Free tier. [21]

### 3.2.4 AWS API Gateway

AWS API Gateway je výkonná služba, ktorá umožňuje vývojárom a administrátorom vytvárať, publikovať, spravovať a monitorovať aplikačné rozhrania. Táto služba je kľúčovou súčasťou architektúry aplikácií, ktorá poskytuje jednoduchý a flexibilný spôsob, ako komunikovať s inými službami v cloudovom prostredí a poskytovať koncovým používateľom prístup k dátam a požadovaným funkciám.

API Gateway podporuje vytváranie RESTful aj Websocket rozhraní. Táto služba je výborným nástrojom pre integráciu s inými AWS službami, ako napríklad úložiskom S3 alebo funkciami Lambda. Integrácia týchto služieb umožňuje vytvárať komplexné aplikácie s vysokými požiadavkami na dostupnosť a výkon.

Flexibilita API Gateway umožnuje využitie tejto služby ako vstupnej brány pre prístup k inak súkromným zdrojom, ako napríklad už spomínaných S3 úložísk, či funkcii Lambda. Jednotlivé koncové body rozhrania API Gateway môžu byť zabezpečené integráciou externých alebo interných autentifikačných služieb, alebo použitím vlastného autentifikačného mechanizmu pomocou funkcií Lambda.

Automatické škálovanie je ďalšou výhodou API Gateway. Táto služba nevyžaduje alokáciu hardvérových zdrojov, čo umožňuje jednoduché a rýchle škálovanie v prípade rýchleho rastu počtu používateľov aplikácie. To znamená, že aplikácia je schopná prispôsobiť sa zmene požiadaviek a rastu počtu používateľov, čím sa zabezpečí neprerušený chod aplikácie a zákazníci budú mať stále k dispozícii optimálne výkonné aplikačné rozhranie. Toto prispieva k zlepšeniu celkovej používateľskej skúsenosti a znižuje pravdepodobnosť výpadkov aplikácie.

Okrem toho, automatické škálovanie umožňuje úspory nákladov na hardvér, pretože cena sa odvíja len od spotrebovaných zdrojov.

Tabuľka 3.2: Porovnanie ceny vzhľadom na počet požiadaviek, pre lokáciu Frankfurt

| Počet REST API dopytov (v miliónoch) | Cena (za milión dopytov) |
|---|---|
| prvých 333 | 3.70$ |
| nasledujúcich 667 | 3.19$ |
| nasledujúcich 19 000 | 2.71$ |
| nad 20 000 | 1.72$ |

V tabuľke 3.2 je možné vidieť ceny vzhľadom na počet požiadaviek, pre lokáciu Frankfurt.

### 3.2.5 AWS EventBridge

AWS EventBridge služba slúži na spojenie viacerých AWS služieb pomocou udalostí. Táto služba umožňuje prenášať udalosti (z angl. event) medzi aplikáciami a službami, čím umožňuje automatizovať procesy a reakcie na udalosti v reálnom čase. Výhodou je, že na udalosti môžu reagovať aj interne poskytované AWS služby aj aplikácie tretích strán. Napríklad vlastná aplikácia môže reagovať na udalosť vytvorenú inou AWS službou, ako je napríklad AWS Lambda funkcia alebo S3 úložisko. Táto služba teda umožňuje jednoduchý model prenosu informácií naprieč službami a aplikáciami v AWS infraštruktúre bez nutnosti riešiť správu hardvérových zdrojov alebo škálovanie služby. [22] V tabuľke 3.3 je možné vidieť ukážku porovnania cien v závislosti na type služby.

Tabuľka 3.3: Porovnanie ceny vzhľadom na počet a typ udalostí, pre lokáciu Frankfurt

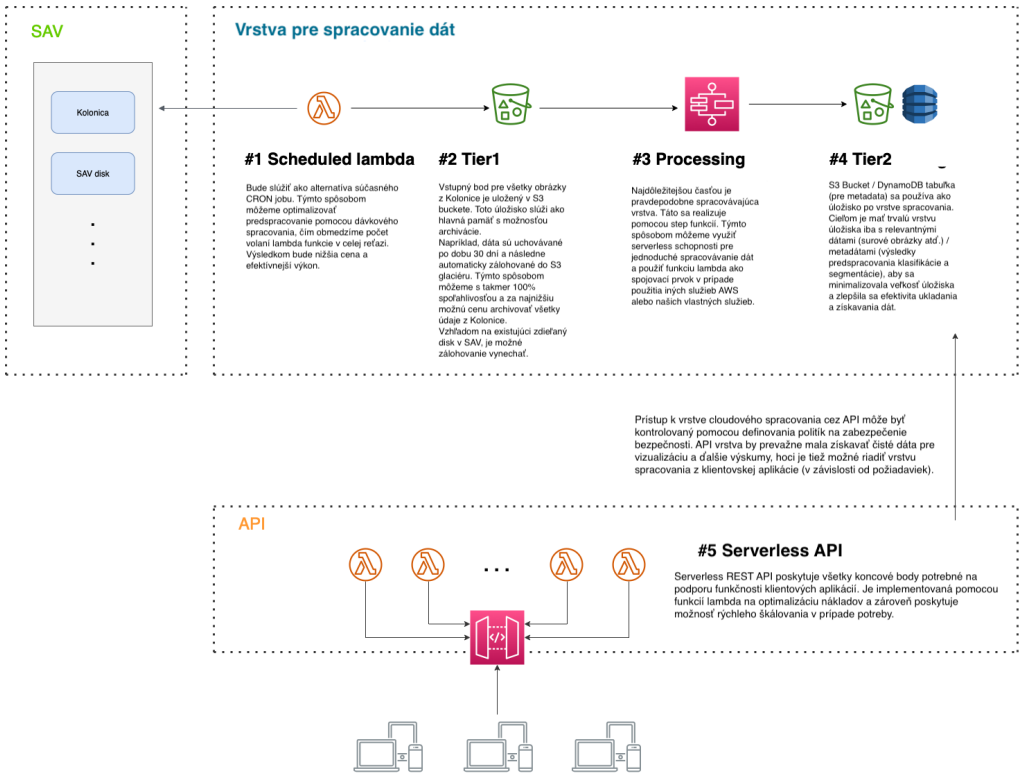
| Typ EventBridge udalosti | Cena |
|---|---|
| Udalosti zo štandardných AWS služieb | zadarmo |
| Vlastné udalosti | 1$ za milión odoslaných udalostí |
| udalosti aplikácií tretích strán | 1$ za milión odoslaných udalostí |

## 3.3 Návrh serverless architektúry

V tejto časti bude opísaný prístup implementácie cloudovej infraštruktúry pre spracovanie **AMON-ES** dát. Predstavíme prvý návrh architektúry spolu s výhodami a problémami. V závere kapitoly zhrnieme nadobudnuté poznatky a predstavíme možné spôsoby riešenia nedostatkov navrhnutej serverless architektúry.

Prvý návrh architektúry pozostával primárne zo serverless služieb, ktoré platforma AWS ponúka. Pod pojmom serverless myslíme, služby pri ktorých nie je nutné platiť za hardware, na ktorom je náš kód vykonávaný, ale len za čas a zdroje spotrebované pri jeho vykonávaní. Príkladom takýchto služieb sú napríklad lambda funkcie alebo služba S3.

Pravdepodobne najväčšou výhodou serverless služieb je jednoduché škálovanie. Vďaka jednoduchému škálovaniu je možné vytvoriť infraštruktúru, ktorá pri desiatkach používateľov nestojí takmer nič no zároveň je schopná obslúžiť desiatky tisíce používateľov.




Obr. 3.1: Diagram prvého návrhu architektúry

Na obrázku 3.1 je zobrazený diagram prvého návrhu cloudovej architektúry pre pre spracovanie a prístup AMON-ES dátam. Ako je možné vidieť architektúra pozostáva z troch základných častí

- Existujúcej infraštruktúry na pôde Slovenskej akadémie vied

- Časti určenej na spracovanie nových dát
- Časti určenej na získavanie spracovaných dát

Začneme s popisom časti určenej na spracovanie a ukladanie nových AMON-ES dát. Táto časť sa skladá zo základných zreťazených služieb, ktorých hlavnou úlohou je získavanie a ukladanie nových dát, ako aj spracovanie a analýzu AAC obrázkov a ich následne uloženie.

Prvou časťou v tomto procese je lambda funkcia, určená na získanie nových AMON dát z databázy Slovenskej akadémie vied a ich následné uloženie do DynamoDB databázy, rovnako ako aj získanie nových AAC obrázkov a ich uloženie do dočasného S3 bucketu, určeného pre ďalšie spracovanie. Táto lambda funkcia je spúšťaná v pravidelných intervaloch s granularitou 24 hodín, keďže nové dáta sú do databázy Slovenskej akadémie vied ukladané taktiež s granularitou 24 hodín. Automatické spúšťanie je v tomto návrhu realizované pomocou AWS služby CloudWatch, ktorá v pravidelných intervaloch, každých 24 hodín, vytvorí udalosť (z angl. event). Na túto udalosť je naviazaná už spomínaná lambda funkcia, ktorá sa po jej odchytení spustí a vykoná požadovaný kód. Využite služby CloudWatch je nutné, keďže samotné lambda funkcie neponúkajú možnosť automatického spúšťania v pravidelných intervaloch.

S3 bucket, do ktorého sú ukladané nové AAC obrázky, slúži ako buffer, na ktorý sú naviazané ďalšie služby. Dáta v tomto S3 buckete sú automaticky vymazané po uplynutí 30 dní, vďaka čomu sa šetrí pamäť a teda aj náklady, ktoré sú s ňou spojené.

Ďalšou časťou je samotné spracovanie nových AAC dát. Nápadom pri tejto časti bolo využitie lambda funkcií, zreťazených za sebou, s využitím stavového automatu. Každá lambda funkcia by v tomto prípade predstavovala jednu časť analýzy AMON-ES dát (ako napríklad klasifikáciu alebo segmentáciu obrázkov) a následne uloženie vhodných dát pre ďalšie spracovanie, ako aj metadát (dodatočných informácií, získaných z analýzy). Tento proces narazil na zopár limitácií, ktoré sú bližšie opísané v nasledujúcich sekciách tejto kapitoly.

Po spracovaní AAC obrázkov sú následne tieto dáta uložené do finálneho S3 bucketu, spolu s metadátami o analýze týchto obrázkov, ktoré sú uložené v DynamoDB databáze.

**Prístup k spracovaným dátam** je v tomto cloudovom návrhu realizovaný pomocou AWS služby API Gateway, vo forme REST rozhrania a jednotlivými lambda funkciami, ktoré realizujú funkcie na získavanie AMON a AAC dát, uložených v DynamoDB databáze a S3 buckete.

## 3.4 Využitie IAC nástrojov pri tvorbe cloudovej infraštruktúry

V priebehu implementácie serverless cloudovej infraštruktúry zohrávali kľúčovú úlohu nástroje pre infraštruktúru ako kód (IAC). IAC nástroje predstavujú programové nástroje, ktoré umožňujú vytvárať, konfigurovať a orchestráciu cloudových služieb a infraštruktúry. Ich využitie je založené na paradigme deklaratívneho programovania, ktorá popisuje, akú konfiguráciu a služby chceme mať v našej cloudovej infraštruktúre, bez nutnosti manuálnej interakcie s grafickým rozhraním cloudovej platformy.

Jednou z hlavných výhod IAC nástrojov je možnosť rýchleho a efektívneho dopĺňania alebo úpravy cloudových služieb bez potreby ručnej interakcie. V dôsledku toho sa vývojári môžu zamerať na samotnú funkcionalitu a návrh cloudových služieb, namiesto manuálnej správy a konfigurácie infraštruktúry.

V súčasnosti existuje viacero alternatív pri výbere IAC nástrojov, pričom medzi najpopulárnejšie, a v kontexte tejto diplomovej práce najrelevantnejšie, patria AWS CloudFormation, Terraform a Pulumi.

### 3.4.1 AWS CloudFormation

AWS CloudFormation je IaC (z angl. Infrastructure as a code) nástroj poskytovaná cloudovým poskytovateľom AWS, ktorá slúži na vytváranie, úpravu a modelovanie rôznych cloudových služieb. Jednou z výhod tohto nástroja je možnosť definovania infraštruktúry ako kódu v takzvaných CloudFormation šablónach vo formáte YAML alebo JSON, ktoré je následne možné aplikovať a vytvoriť definované služby priamo v cloudovom prostredí. V tomto procese sa šablóna nahrá na cloudovú platformu AWS, konkrétne do služby AWS CloudFormation, kde sa automaticky spustí vytváranie služieb definovaných v tejto šablóne. Je dôležité zdôrazniť, že CloudFormation je určený iba pre použitie v AWS cloude a neexistuje žiadna podpora pre iné cloudové platformy. Táto služba tiež poskytuje špecifické nástroje na validáciu a kontrolu správnosti vytvorených šablón, ktoré sa líšia v závislosti od služby, ktorú používajú. Vďaka tomu, že CloudFormation je poskytovaný priamo od AWS, je výhodou fakt, že stav reprezentujúci vytvorené, či spravované služby nie je nutné manuálne spravovať ako to je v prípade služieb ako Pulumi alebo Terraform. Tento stav sa ukladá vo forme tagov, teda metadát, uložených pri každej službe vytvorenej pomocou nástroja CloudFormation. Vďaka týmto tagom, vie následne služba CloudFormation detegovať,

ktoré služby v cloudovom prostredí sú spravované manuálne, a ktoré s využitím CloudFormation šablón. [23]

### 3.4.2 Terraform

Služba Terraform, poskytovaná od spoločnosti HashiCorp, je rovnako ako AWS CloudFormation určená na vytváranie, úpravu a replikáciu cloudových služieb. Hlavným rozdielom je, že Terraform nie je určený iba pre cloudovú platformu AWS, ale aj pre ostatné cloudové platformy, ako napríklad Azure alebo Google Cloud. Fakt, že Terraform je možné využívať na viacerých cloudových platformách, musí byť zohľadnený aj vo formáte určenom na opis cloudových služieb, pretože každý cloudový poskytovateľ môže vyžadovať rôzne formáty. Pre opis cloudových služieb je preto potrebné využívať doménovo špecifický jazyk nazývaný HCL (z angl. HashiCorp language), ktorý je následne prekladaný na špecifický formát vyžadovaný cloudovým poskytovateľom. Napríklad, v prípade poskytovateľa AWS sa jazyk HCL prekladá na CloudFormation. [24]

Jednou z nevýhod v prípade nástroja Terraform je nutnosť správy stavu reprezentujúceho spravované služby Terraformom, manuálne. Inak povedané, počas vytvárania cloudových služieb Terraform vygeneruje súbor, ktorý obsahuje informácie o tom, aké služby boli práve vytvorené. Tento súbor je následne potrebné v prípade, ak je potrebné tieto služby upraviť, či vymazať. Strata tohto súboru alebo jeho poškodenie môže viesť k tomu, že Terraform nebude vedieť tieto služby rozoznať a následne bude potrebné tieto služby spravovať manuálne. [25]

Výhodou služby Terraform je však to, že sa od služby AWS CloudFormation líši v tom, že je univerzálnejší a môže byť používaný na viacerých cloudových platformách, nielen na AWS. Využitie doménovo špecifického jazyka HCL je vďaka doplnkovým nástrojom a integráciám do rôznych IDE, ktoré spoločnosť HashiCorp vyvíja, jednoduché a dobre použiteľné. Subjektívne sa s jazykom HCL pracuje lepšie ako v prípade CloudFormation šablón, a preto bol pre potreby tejto práce vybraný nástroj Terraform.

### 3.4.3 Pulumi

Ďalším nástrojom, ktorý získava v súčasnosti stále väčšiu popularitu, je Pulumi. Podobne ako Terraform, aj Pulumi je možné použiť pre široké spektrum cloudových poskytovateľov. Významným rozdielom oproti už spomínaným nástrojom je skutočnosť, že pri využívaní Pulumi je možné definovať cloudovú infraštruktúru pomocou programovacieho jazyka, ktorý je všeobecne použiteľný a nie po-

mocou doménovo špecifického jazyka. Pulumi momentálne podporuje viaceré populárne programovacie jazyky, ako napríklad Python, TypeScript, Java a Go. [26] [27]

Výhody využitia programovacieho jazyka všeobecného použitia je mnoho. Programátor nemusí zdokonaľovať svoje znalosti v doménovo špecifickom jazyku, čo by mohlo byť časovo náročné a nákladné. Pri navrhovaní architektúry softvérového systému môže programátor využiť prostriedky, na ktoré je zvyknutý pri písaní kódu, ako napríklad integrované vývojové prostredie (IDE) a nástroje na vyhľadávanie dokumentácie. Moderné IDE navyše umožňujú programátorom získať informácie o očakávaných parametroch pre jednotlivé služby, čím zvyšujú efektivitu práce a minimalizujú pravdepodobnosť chýb. [26]

### 3.4.4 Zhrnutie IaC nástrojov

V tejto časti sme stručne opísali tri pravdepodobne najpoužívanejšie IaC nástroje, ktoré je možné využiť pri vytváraní rôznych služieb na cloudovej platforme AWS. Pri výbere konkrétnej služby sme sa zohľadňovali najmä nástroj Terraform a Pulumi, a to z dôvodu, že nie sú obmedzené len ne jednu cloudovú platformu.

Vďaka tomu, že nástroj Pulumi umožňuje definovať cloudové služby s využitím jazyka všeobecného použitia, zdal sa byť lepšou voľbou. Tým by bolo možné napísať jednotlivé služby priamo v jazyku Python, ktorý je bežne využívaný pri projektoch Slovenskej akadémie vied. Preto by tento nástroj pravdepodobne bol pre zamestnancov SAV jednoduchší na úpravu v prípade potreby. Avšak v čase písania tejto práce mal nástroj Pulumi menej kvalitnú a konzistentnú dokumentáciu v porovnaní s nástrojom Terraform. Pri testovaní tohto nástroja sme sa stretli s viacerými problémami, ktoré boli často nesprávne alebo nedostatočne zdokumentované. Preto sme sa pre potreby tejto práce rozhodli použiť nástroj Terraform, ktorého dokumentácia v čase písania tejto práce bola v porovnaní s nástrojom Pulumi lepšia.

## 3.5 Implementácia serverless architektúry

V úvodnej fáze implementácie serverless architektúry bol dôraz kladený na časť spracovania AAC obrázkov s využitím lambda funkcie spustenej v pravidelných intervaloch a ich ukladanie do cloudového úložiska S3, pretože časť spracovania nových AAC obrázkov bola chýbajúcim prvkom v počiatočnej verzii aplikácie AMON-ES. Pre tento účel bol vytvorený nový projekový repozitár, ktorý pozostával z týchto základných častí:

- **infrastructure** - táto časť zoskupuje Terraform súbory určené na tvorbu cloudovej infraštruktúry,
- **src** - časť zrdužujúca zrdojové programoé súbory lambda funkcí, určených na spracovanie AAC obrázkov,
- **scripts** - táto časť bola určená na zdrojové súbory potrebné pre prípravu zdrojových kódov lambda funkcií pred ich nasadením na cloudovú platformu AWS.

V infraštruktúrnej časti je možné nájsť Terraform súbory, ktoré implementujú tieto časti serverless architektúry určené na spracovanie obrázkov:

- **S3 úložisko Tier 1** - toto úložisko slúžilo na ukladanie obrázkov zo serverov SAV a bolo doplnené o automatické mazanie dát starších ako 30 dní
- **S3 úložisko Tier 2** - slúžilo na ukladanie odfiltrovaných AAC po analýze obrazu,
- **lambda funkcia na sťahovanie obrázkov** - táto funkcia simulovala proces pravidelného sťahovania dát zo serverov SAV.
- **Lambda funkcia na spracovanie obrazu** - funkcia vykonávajúca analýzu AAC obrázkov a ich nahrávanie do úložiska **Tier 2** a do DynomoDB databázy.

Okrem infraštruktúry pre jednotlivé Lambda funkcie, repozitár obsahoval aj zdrojové kódy napísané v jazyku Python, ktoré implementovali nasledujúce funkcie:

- Prvá lambda funkcia slúžila na simuláciu sťahovania obrázkov zo serverov SAV. Na tento účel bolo vytvorené S3 úložisko s testovacími dátami, ktoré predstavovalo alternatívu k serverom SAV. Následne sa po zachytení udalosti zo služby EventBridge pravidelne spustila táto funkcia, simulovala sťahovanie obrázkov a uložila ich na S3 úložisko označené ako **Tier 1**.
- Po dokončení nahrávania obrázkov sa spustila funkcia určená na analýzu obrázkov AAC s využitím existujúceho algoritmu na analýzu obrazu. (2) V závislosti na výsledku tejto analýzy boli vyfiltrované obrázky, vhodné na ďalšie štúdium, uložené na S3 úložisko označené ako **Tier 2**.

Na záver, samotný proces nasadzovania opísanej infraštruktúry si vyžadoval niekoľko ďalších krokov. Keďže nástroj Terraform je primárne určený na vytváranie infraštruktúry a nie na nahrávanie zdrojových kódov, bolo potrebné automatizovať proces nahrávania samotných zdrojových kódov lambda funkcií na cloudovú platformu AWS. Najjednoduchším spôsobom bolo využiť možnosť priameho nahrávania zdrojového kódu zabaleného v komprimovanom formáte. Pre tento účel bol vytvorený bash skript, ktorý pred aplikovaním infraštruktúrnych zmien zabalil, komprimoval zdrojové kódy a uložil a pripravil ich na nahrávanie v výstupnom priečinku **build**.

Okrem komprimácie samotných zdrojových kódov bolo potrebné do výsledného balíka pridať aj externé závislosti jednotlivých lambda funkcií. Toto je podmienené tým, že lambda funkcie sa spúšťajú v prostrediach, kde nie je možné definovať krok inštalácie externých závislostí, ako sú napríklad knižnice. Táto limitácia je detailnejšie opísaná v nasledujúcej sekcii.

## 3.6 Limitácie serverless architektúry

Už pri úvodnej fáze vývoja tejto serverless architektúry nastalo niekoľko problémov, ktoré nakoniec viedli k prehodnoteniu opísaného riešenia a analýze a hľadaniu alternatívnych, lepšie udržateľných riešení.

Prvým problémom, ktorý sa objavil, bolo obmedzenie veľkosti externých závislostí lambda funkcií pre analýzu obrazu. Vo vstupnej implementácii boli použité existujúce algoritmy, ktorých veľkosť presahovala maximálnu veľkosť lambda funkcie, ktorá predstavuje 250MB v nekomprimovanom formáte. Tento problém bol možné vyriešiť optimalizáciou veľkosti externých závislostí lambda funkcií. Avšak, tento prístup nie je vždy aplikovateľný a berúc do úvahy fakt, že spracovanie a analýza AAC obrázkov bude s najväčšou pravdepodobnosťou v budúcnosti upravovaná a vylepšovaná. Vzhľadom na to, že moderné prístupy využívajúce strojové učenie pri analýze obrazu môžu byť značne komplexné, bolo toto obmedzenie veľkosti veľmi limitujúcim faktorom. Ďalšou možnosťou, bolo využitie vlastných Docker obrazov, ktoré by boli spúšťané lambda funkciami a vďaka ktorým by bolo možné vyhnúť sa pamäťovým obmedzeniam. Nevýhodami tohto prístupu je to že, by si však vyžadoval dodatočné AWS služby pre nahrávanie Docker obrazov, komplexnejší spôsob automatizácie tvorby týchto Docker obrazov, ako aj náročnejší vývoj či komplikovanejšie riešenie prípadných chýb.

Ďalším problémom pri práci s lambda funkciami bolo časový faktor pri vývoji ako aj náročné testovanie. Pre otestovanie novo pridanej funkcionality do systému

bolo potrebné jenodltivé zdrojové kódy spolu so závislosťami komprimovať a pripraviť na nahrávanie, pomocou Terraform-u aplikovať nové zmeny a počkať na aktualizáciu infraštruktúry, na webovej platforme AWS vytvoriť testovaiu udalosť pre vytvorenú lambda funkciu s štruktúrou zodpovedajúcou napríklad štruktúre udalosti zo služby EventBridge a následne otestovanie samotnej lambda funkcie. Pri vývoji aplikácie je časový faktor významne ovplyvnený trvaním celého procesu, ktorý môže trvať aj niekoľko minút, ak sa vyskytne nejaký problém, ktorý je potrebné riešiť.

Kombinácia týchto problémov a faktu, že pri prípadnej potrebe úpravy tejto aplikácie by bolo potrebná pomerne vysoká znalosť týchto cloudových služieb, sme sa rozhodli v rámci AMON-ES aplikácie prehodnotiť cieľovú architektúru a v serverless prístupe nepokračovať.

## 3.7 Zhrnutie

Cloudová platforma AWS poskytuje široké spektrum služieb, z ktorých v tejto kapitole zdôrazňujeme tie najdôležitejšie pre kontext tejto práce. Základné služby, ako napríklad S3, Lambda funkcie a EventBridge, sú navrhnuté s cieľom umožniť jednoduchú implementáciu, vysokú škálovateľnosť a flexibilný potenciál využitia. Tieto služby sa využívajú v širokom spektre cloudových aplikácií.

Následne sme v kapitole popísali prvý prístup pri tvorbe cloudovej architektúry na platforme AWS. Pri jej implementácii sme sa snažili infraštruktúru vytvárať automaticky pomocou nástroja Terraform, vďaka ktorému bolo možné jednoducho infraštruktúru definovať pomocou doménovo špecifického jazyka, ktorý tento nástroj ponúka, rovnako ako jednoducho vytvorenú infraštruktúru z cloudovej platformy vymazať. Snažili sme sa využívať AWS služby, ktoré sú jednoducho škálové ako S3 alebo lambda funkcie. Pri samotnej implementácii časti architektúry určenej na analýzu AAC obrázkov sme narazili na zopár problémov, ktoré so sebou tieto služby prinášajú, ako napríklad lokálne testovanie lambda funkcií, ktoré závisia na iných AWS službách alebo pamäťové limity lambda funkcií.

V súvislosti s uvedenými problémami sa ukázalo, že použitie vlastných Docker obrazov je nevyhnutným krokom pri implementácii funkcií určených na spracovanie AAC obrázkov. Kombinácia vlastných Docker obrazov so serverless prístupom sa však ukázala ako zvláštna, keďže vlastné Docker obrazy môžeme spustiť na iných službách AWS, ako napríklad Fargate alebo EC2 s automatickým škálovaním, a stále nevyriešila problémy s vývojom a lokálnym testovaním počas vývoja. Vzhľadom na to, že časový faktor a udržateľnosť výsledného riešenia s

čo najmenším rizikom boli kľúčovými faktormi, bol ďalším krokom analýza alternatívnych prístupov s využitím vlastných Docker obrazov, ktoré by zlepšili aj už spomínané problémy s lokálnym testovaním a rýchlosťou vývoja. Viac o tomto návrhu je opísané v kapitole 4.

# 4 Serverfull architektúra

V tejto kapitole sa sústredíme na návrh a implementáciu novej architektúry aplikácie AMON-ES, ktorej základom budú Docker kontajnery. Cieľom bude vytvoriť robustnú, no jednoduchú architektúru, ktorá umožní beh aplikácie v cloudovom prostredí, no zároveň bude umožňovať jednoduchú úpravu aplikácie, či jej lokálne spustenie a testovanie. Následne opíšeme upravenú serverovú a grafickú časť aplikácie, predstavíme proces vytvárania cloudovej infraštruktúry a v závere kapitoly si ukážeme príklad možnej úpravy tejto architektúry.

## 4.1 Využité AWS služby

V rámci nového návrhu architektúry boli využité viaceré AWS služby, ktoré neboli bližšie opísané v predchádzajúcej kapitole. V tejto časti kapitole si teda bližšie predstavíme AWS služby, ktoré budú neskôr využité v rámci návrhu novej architektúry aplikácie AMON-ES. Je dôležité poznamenať, že v porovnaní s predchádzajúcou kapitolou, ktorá sa zaoberala serverless službami, sa tentoraz budeme sústrediť na služby, ktoré využívajú klasické hardvérové zdroje. Tieto služby sa vyznačujú tým, že využívajú virtuálne alebo fyzické servery, ktoré sú poskytované a spravované poskytovateľom cloudu.

Medzi najdôležitejšie služby, ktoré boli zahrnuté do návrhu architektúry, patrí napríklad Amazon EC2 (z angl. Elastic Compute Cloud), ktorý poskytuje výpočtové kapacity v cloude a umožňuje vytvárať a spravovať virtuálne servery alebo služba RDS (z angl. Relational Database Service), ktorá slúži na správu relačných databáz.

### 4.1.1 EC2

Začneme s jednou z najzákladnejších služieb, ktorou je služba EC2. Táto služba ponúka jednoducho škálovateľné virtuálne stroje s nastaviteľnými parametrami, ako sú výkon procesora, operačná pamäť, veľkosť a typ pevného disku a podobne.

Táto služba je navrhnutá tak, aby bolo jednoduché zväčšovať alebo zmenšovať jej výkon podľa potreby [16].

AWS ponúka viacero typov EC2 inštancií, ktoré sa líšia ako v dostupnosti tak aj v cene. Základné typy EC2 inštancií sú nasledovné [16]:

- **On-Demand inštancie** - Tieto inštancie sú charakteristické tým, že ich cena závisí len od výkonu inštancie a času, ktorý je táto inštancia spustená, pričom na ich využívanie nie je potrebná žiadna platba vopred. Ich výkon je možné flexibilne zvyšovať alebo znižovať v závislosti od potreby.

- **Spot inštancie** - Tento typ EC2 inštancií je ponúkaný s redukovanou cenou na úkor zníženej dostupnosti. Výška zľavy, ktorú môže tento typ inštancie dosiahnuť v porovnaní s On-Demand inštanciami, je až 90%. Nevýhodou však je, že tento typ inštancie je dostupný len za predpokladu, že je v danej chvíli dostatočne veľa voľných výpočtových zdrojov. Tieto inštancie sa hodia napríklad na typy výpočtových úloh, ktorých vykonanie nemusí nastať okamžite.

- **Rezervované inštancie** - Tieto inštancie sú charakteristické tým, že ponúkajú zľavy za ich využívanie za predpokladu, že je výpočtová kapacita rezervovaná vopred. Sú vhodné najmä na typy výpočtových aplikácií, ktorých doba prevádzky je vopred známa.

- **Saving Plans inštancie** - Inštancie, ktoré ponúkajú zľavy pri viazanosti využívania dohodnutej kapacity v časovom horizonte 1 až 3 roky.

- **Dedikované inštancie** - Fyzické servery dedikované pre využívanie zákazníkom, ktorý si ich zaobstaral. Príkladom využitia môžu byť napríklad aplikácie s vysokými nárokmi na bezpečnosť, v ktorých využívanie virtuálnych serverov nepripadá do úvahy.

EC2 inštancie sú často základom iných AWS služieb, ako napríklad služieb súvisiacich s ukladaním dát alebo sieťového pripojenia. EC2 inštancie môžu byť využívané pre beh aplikácií, ktoré vyžadujú prístup k výpočtovým zdrojom, ale aj pre web servere, databázové servery alebo pre iné typy aplikácií. EC2 inštancie sú veľmi flexibilné a môžu byť nakonfigurované a prispôsobené potrebám konkrétneho využitia. [16]

### 4.1.2 Autoscaling group

AWS Autoscaling Group je služba poskytovaná Amazon Web Services, ktorá umožňuje automaticky prispôsobovať počet EC2 inštancií v cloude v závislosti od aktuálneho zaťaženia aplikácie. Cieľom je zabezpečiť dostupnosť aplikácie a optimalizovať využitie zdrojov. Pri vytvorení služby Autoscaling Group sa definuje minimálny a maximálny počet inštancií, ktoré majú byť k dispozícii. Na základe nastavení a sledovania metrík zaťaženia aplikácie sa Autoscaling Group automaticky pridáva alebo odoberá EC2 inštancie. V prípade zvýšeného zaťaženia sa pridajú nové EC2 inštancie, ktoré pomôžu zvládnuť väčšie množstvo požiadaviek. Ak sa zaťaženie zníži, Autoscaling Group odoberie nepotrebné EC2 inštancie, aby sa ušetrili zdroje a znížili náklady. [28]

AWS Autoscaling Group umožňuje optimalizovať využitie zdrojov a zlepšiť dostupnosť aplikácie. Výhodou je tiež, že sa sám stará o správu inštancií a umožňuje rýchlu reakciu na zmeny zaťaženia aplikácie.

### 4.1.3 AWS Load Balancer

AWS Load Balancer je služba poskytovaná Amazon Web Services, ktorá slúži na distribúciu záťaže medzi viacerými cieľovými EC2 inštanciami v cloude. Hlavným cieľom Load Balanceru je zlepšenie výkonu a dostupnosti aplikácií tým, že zabezpečuje rovnomerné rozloženie záťaže medzi viacerými inštanciiami a zabezpečenie ich vysokej dostupnosti. Príkladom môže byť aplikácia, ktorá je spustená na viacerých EC2 inštanciách, kde AWS Load Balancer rovnomerne distribuuje komunikáciu medzi týmito inštanciami. [29]

Existujú tri hlavné typy Load Balancerov v AWS:

- Classic Load Balancer,
- Application Load Balancer
- Network Load Balancer.

Každý typ má svoje vlastné použitie a výhody. Classic Load Balancer je starší typ a funguje na základe IP adresy, portov. Application Load Balancer a Network Load Balancer sú novšie verzie, ktoré poskytujú pokročilejšie funkcie, ako napríklad možnosť rovnomerného rozdeľovania záťaže na úrovni aplikácií a možnosť použitia statických IP adries. [30]

V kontexte tejto práce je najrelevantnejší práve Aplication load balancer. V kombinácií týmto load balancerom sa často využíva služba **Target Group**, ktorá

špecifikuje množinu EC2 inštancií alebo iných cloudových zdrojov, na ktoré load balancer distribuuje komunikáciu. Target Group je možné nakonfigurovať na pravidelnú kontrolu funkčnosti jednotlivých EC2 inštancií, vďaka čomu je možné automaticky monitorovať ich stav a odstrániť ich v prípade, že nie sú funkčné. [31]

### 4.1.4 Amazon RDS

Amazon Relational Database Service (RDS) je cloudová služba od spoločnosti AWS, ktorá umožňuje jednoduché a efektívne vytváranie, prevádzku a škálovanie relačných databáz. Amazon RDS ponúka viac možností databázových enginov vrátane MySQL, PostgreSQL, Oracle, SQL Server a MariaDB. Tieto enginy sú konfigurované a spravované AWS, čo znamená, že zákazníci nemusia zvládať operácie, ako je inštalácia a aktualizácia softvéru, zálohovanie alebo monitorovanie vlastného hardvéru. Toto riešenie poskytuje zákazníkom možnosť plne sa sústrediť na vývoj aplikácií a využívať k dispozícii databázové kapacity podľa potreby. [32]

Jednou z najdôležitejších vlastností Amazon RDS je jeho schopnosť škálovania. Zákazníci môžu jednoducho zvyšovať alebo znižovať veľkosť ich databázových inštancií v závislosti na nárokoch aplikácie a počtu používateľov. Taktiež, existuje možnosť použiť tzv. read replicas pre zvyšovanie kapacity pre čítanie databázy. Read replicas sú replikované databázové inštancie, ktoré sú asynchrónne synchronizované so zdrojovou inštanciou. Týmto spôsobom sa zvyšuje kapacita databázy pre čítanie a zároveň sa zníži záťaž na zdrojovú inštanciu. [32]

Okrem toho, Amazon RDS umožňuje zákazníkom nastavenie vysoko dostupných (z angls. highly available) databázových inštancií, ktoré sú odolné voči výpadkom. Táto vlastnosť zabezpečuje, že aplikácia bude dostupná aj v prípade výpadku jedného databázového uzla. [32]

### 4.1.5 AWS Fargate

AWS Fargate je služba určená pre spúšťanie Docker kontajnerov bez nutnosti starostlivosti o vlastné hardvérové zdroje. Namiesto toho môže používateľ definovať požadované hardvérové prostriedky, ktoré sú nevyhnutné pre funkčnosť Docker kontajnera a AWS Fargate automaticky poskytne tieto zdroje. Kontajnery môžu byť nastavené tak, aby sa automaticky škálovali v závislosti od definovaných podmienok [33].

Služba AWS Fargate sa skladá z týchto základných komponentov [33]:

- **Klaster** - logické zoskupenie viacerých Docker kontajnerov, ktoré spolu súvisia, napríklad v rámci jednej aplikácie.
- **Definícia úlohy** - je textový súbor v JSON formáte, ktorý opisuje jeden alebo viac Docker kontajnerov tvoriacich aplikáciu a funguje ako plán pre aplikáciu, definuje parametre ako operačný systém, kontajnery, otvorené porty a dátové objemy pre kontajner v úlohách. Parametre závisia od potrieb konkrétnej aplikácie.
- **Úlohy** - sú inštancie definície úlohy, predstavujúci konkrétny kontajner spustený v službe AWS Fargate.
- **Služby** (z angl. Service), je komponent, ktorý slúži na riadenie a kontrolu spustených Docker kontajnerov. Je zodpovedný za automatické reštartovanie kontajnera v prípade poruchy alebo neaktivity na základe definície úlohy.

### 4.1.6 AWS ECR

Súvisiacou službou pre AWS Fargate je AWS ECR (Elastic Container Registry). Táto služba poskytuje funkcie pre uchovávanie Docker obrazov, ktoré sa využívajú v rámci infraštruktúry AWS. Medzi hlavné výhody používania AWS ECR patrí zabezpečenie prístupu k Docker obrazom, integrácia s ďalšími službami AWS, ako napríklad Fargate a Lambda a vysoká dostupnosť prostredníctvom automatického replikovania medzi datacentrami v rámci AWS cloudu. Táto služba je kľúčová pre distribuované globálne aplikácie, ktoré potrebujú vysokú dostupnosť a zabezpečenie. [34]

### 4.1.7 AWS Elastic Beanstalk

Amazon Elastic Beanstalk je webová služba od AWS, ktorá umožňuje vývojárom jednoduchý spôsob nasadenia a správy aplikácií v cloude. Tento nástroj sa využíva pre rýchle nasadenie aplikácií a webových stránok bez nutnosti zložitého konfigurovania a spravovania infraštruktúry. Elastic Beanstalk interne využíva služby, ktoré sme spomenuli v predchádzajúcich častiach tejto kapitoly, ako napríklad EC2 inštancie, Fargate, load balacere a podobne. Vďaka využitiu služby Elastic Beanstalk je však konfigurácia a správa takejto infraštruktúry výrazne jednoduchšia, čo je pre potreby tejto práce ideálne. [35]

Elastic Beanstalk ponúka širokú paletu možností pre nasadenie a správu aplikácií. Podporuje rôzne programovacie jazyky, ako sú Java, .NET, PHP, Python a

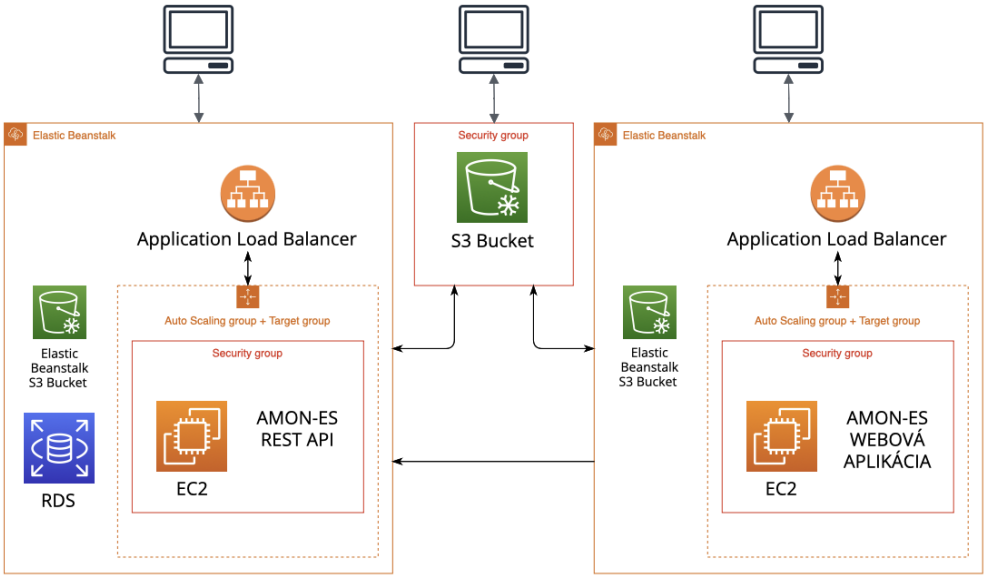
mnoho ďalších. V kontexte tejto práce je však najdôležitejšia možnosť využitia nástroja Docker. V tomto prípade stačí vytvoriť súbor nazvaný Dockerfile, ktorý definuje, ako je potrebné danú službu spustiť a Elastic Beanstalk je následne schopný spustiť ľubovoľný kód. Elastic Beanstalk tiež umožňuje prispôsobenie prostredia pre aplikáciu pomocou vlastných konfiguračných súborov, ktoré môžu slúžiť napríklad na Ngnix reverzného proxy. [35]

Jednou z najväčších výhod Elastic Beanstalk je jeho možnosť jednoduchého škálovania aplikácií. Služba umožňuje horizontálne aj vertikálne škálovanie, čo znamená, že vývojári môžu jednoducho pridať alebo odstrániť inštancie, čím zabezpečia dostatočnú kapacitu a dostupnosť aplikácie, prípadne jednoducho definovať silnejšiu inštanciu, na ktorej má aplikácia bežať. Škálovanie sa môže vykonávať manuálne alebo pomocou automatizovaných pravidiel založených na dátovej záťaži, ktoré umožňujú elastické prispôsobenie aplikácie podľa požiadaviek používateľov. [35]

## 4.2 Návrh serverfull architektúry

Pri návrhu novej architektúry pre aplikáciu AMON-ES sme sa sústredili na to, aby sme predišli podobným problémom ako v prípade serverless architektúry. Vzhľadom na to, že aplikácia pre monitorovanie AMON-ES dát je kontinuálnym projektom, na ktorom budú po ukončení tejto diplomovej práce pokračovať ako zamestnanci SAV, tak aj ďalší študenti, bolo cieľom to, aby bola novo navrhnutá architektúra jednoduchá na úpravu a doplnenie, aj bez hlbokých znalostí o jednotlivých AWS službách a zároveň bola jednoducho škálovateľná vzhľadom na aktuálne zaťaženie.

Ako je možné vidieť na obrázku 4.1, výsledná architektúra pozostáva z viacerých častí. Začneme s popisom REST API aplikácie AMON-ES. Základom tejto časti je EC2 inštancia, na ktorej je sputený Docker kontajner predstavujúci AMON-ES REST API službu. Táto EC2 inštancia je zaradená do Auto Scaling groupy, ktorá zabezpečuje automatické vytváranie prípadne odoberanie nových EC2 inštancií vzhľadom na aktuálnu záťaž na toto REST rozhranie. Ďalej je zaradená aj do Target groupy, ktorá sa stará o distribúciu komunikácie medzi spustenými EC2 inštanciami, ako aj o kontrolu správnej funkčnosti jednotlivých inštancií pomocou takzvaných health check-ov. Následne je do tejto architektúry pridaný Aplikačný load balancer, ktorý slúži, ako jednotný prístupový bod do aplikácie a ktorý pomocou target group-y distribuuje komunikáciu medzi koncovým klientom a EC2 inštanciami.




Obr. 4.1: Zobrazenie hlavných častí serverfull architektúry aplikácie na platforme AWS.

Všetky tieto služby sú súčasťou Elastic Beanstalk prostredia, ktoré umožňuje jednoduchú konfiguráciu všetkých vyššie spomenutých AWS služieb na jednom mieste. Rovnako ponúka možnosť jednoduchej úpravy existujúceho prostredia aplikácie pomocou CLI nástroja. Súčasťou tohto prostredia je aj služba RDS, ktorá slúži na ukladanie AMON dát a AAC metadát.

Druhou časťou je S3 bucket, ktorý slúži na ukladanie ako surových, tak aj spracovaných AAC dát. Tento S3 bucket nie je súčasťou Elastic Beanstalk prostredia a prístup k nemu je ošetrený správne zvolenou bezpečnostnou skupinou, ktorá zabezpečuje, že len EC2 inštancie určené na spúšťanie AMON-ES REST API doň môžu zapisovať nové dáta. Čítanie dát z tohto S3 bucketu je voľne prístupné pre webovú aplikáciu či širokú verejnosť.

Treťou a poslednou časťou je ďalšie Elastic Beanstalk prostredie, ktoré je určené na beh AMON-ES webovej aplikácie. Podobne ako v prípade AMON-ES REST API aj toto prostredie pozostáva z EC2 inštancie, auto scaling a target group-y a aplikačného load balanceru. AMON-ES služba získava dáta z AMON-ES REST API.

V oboch Elastic Beanstalk prostrediach sa nachádza aj S3 bucket, ktorý je využívaný pri úprave a nasadzovaní novej verzie aplikácie. Do tohto S3 bucketu sa s využitím CLI príkazu nahrala aktuálna verzia aplikácie a služba Elastic Beanstalk automaticky spustí proces nahrávania novej verzie aplikácie naprieč všetkými aktuálne spustenými EC2 inštanciami.

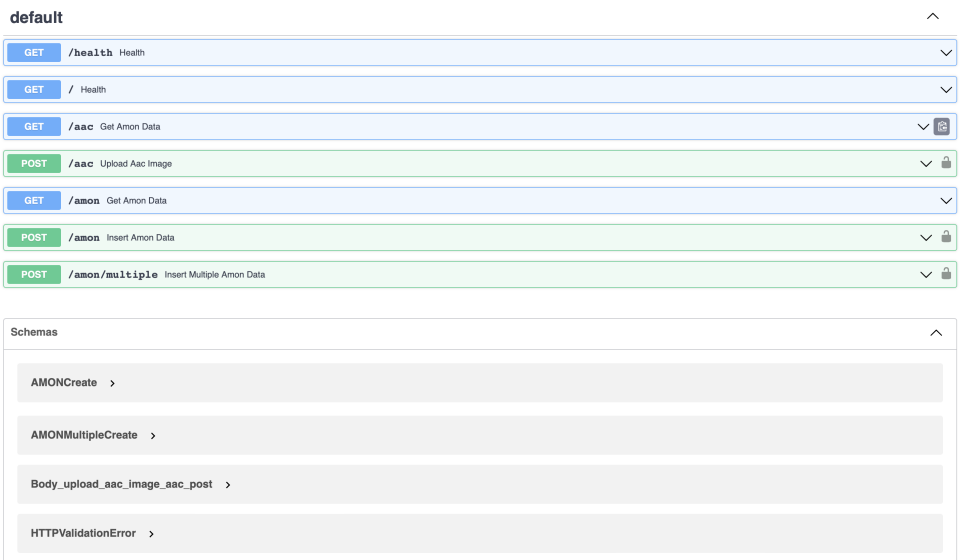
Rozdelenie aplikácie na dva samostatné Elastic Beanstalk prostredia vychá-

dzalo z požiadavky pracovníkov SAV, ktorí mali záujem len o aplikačný prístup k airglow dátam. Vďaka tomuto rozdeleniu sú obe časti aplikácie schopné škálovať svoje hardvérové zdroje podľa aktuálneho zaťaženia. V prípade potreby je možné obe tieto služby spojiť do jedného Elastic Beanstalk prostredia, ktoré bude spúšťať obe služby - REST API a WEB - naraz zo služby docker compose.

## 4.3 Úprava serverovej časti AMON-ES aplikácie

Vzhľadom na nový návrh architektúry bolo potrebné prepísať serverovú časť aplikácie AMON-ES. Keďže táto časť bola pomerne malá a bola založená na technológii Flask Rest Plus, ktorá už nie je ďalej podporovaná, rozhodli sme sa celú serverovú časť aplikácie prepracovať.

Nová verzia serverovej časti aplikácie je napísaná v knižnici Python s využitím technológie FastAPI. FastAPI je webový rámec určený na rýchle vytváranie REST rozhraní s využitím typových anotácií, ktoré sú súčasťou vyšších verzií jazyka Python. FastAPI, podobne ako Flask Rest Plus, ponúka možnosť automatického generovania dokumentácie OpenAPI 3, ale vďaka využitiu typových anotácií je celý proces tvorby dokumentácie automatizovaný. Dokumentácia je prístupná cez rozhranie Swagger UI a jej ukážku je možné vidieť na obrázku 4.2.




Obr. 4.2: Zobrazenie hlavných častí serverfull architektúry aplikácie na platforme AWS.

Ako je vidieť na obrázku, upravená verzia serverovej časti aplikácie sa skladá z týchto koncových bodov:

- **Nahrávanie nových AMON dát** - tento endpoint slúži na nahrávanie nových záznamov z detektora AMON. Štruktúra vstupných dát je identická s pôvodnou databázovou štruktúrou, ktorú je možné vidieť na obrázku 2.8. Tieto dáta sú následne uložené do Postgres databázy, ktorá je spravovaná AWS službou RDS.

- **Nahrávanie viacerých AMON dát** - Upravená verzia vyššie spomenutého koncového bodu, ktorá ponúka možnosť nahrať pole AMON záznamov.

- **Nahrávanie nových AAC obrázkov** - tento koncový bod slúži na nahrávanie nových obrázkov, pričom priamo počas procesu nahrávania je obrázok aj klasifikovaný na základe jeho kvality. Rovnako je aj vygenerovaný nový obrázok, ktorý je využívaný pre zobrazenie vo webovej aplikácii. Pôvodné aj novo vygenerované obrázky sú uložené do S3 bucketu, odkiaľ sú ďalej prístupné pre webovú aplikáciu či širokú verejnosť. Metadáta, ako napríklad čas získania daného AAC obrázku či triedu klasifikácie, sú uložené do Postgres databázy.

- **Získavanie AMON dát** - Tento koncový bod slúži na získavanie AMON dát na základe časového intervalu určeného začiatočným a koncovým dátumom a časom.

- **Získavanie AAC dát** - tento koncový bod slúži na získavanie AAC dát na základe dát na základe časového intervalu určeného začiatočným a koncovým dátumom a časom.

Mimo vyššie spomenuté koncové body, bol do aplikácie pridaný koncový bod pre takzvaný health check, ktorý je vykonávaný každých 30 sekúnd službou AWS target group a je využívaný na určenie toho či aplikácia správne funguje. Tento koncový bod po zavolaní vracia jednoduchú odpoveď s HTTP status kódom 200.

Keďže toto REST rozhranie bude verejne prístupné, všetky koncové body aplikácie určené pre prístup k dátam nebudú zabezpečené. Na zníženie možnosti zaťaženia aplikácie sú však časové rozsahy definujúce interval požadovaných dát obmedzené na sedem dní. Všetky koncové body aplikácie určené na zápis nových dát sú zabezpečené pomocou API kľúča, ktorý je potrebné definovať v autorizačnej hlavičke aplikácie. Tento kľúč je možné jednoducho upraviť pomocou environmentálnych premenných aplikácie v prípade potreby.

Prístup k Postgres databáze je zabezpečený pomocou knižnice SQLAlchemy, čo je populárna knižnice s otvoreným zdrojovým kódom v jazyku Python, ktorá umožňuje jednoducho pracovať s relačnými databázami. SQLAlchemy umožňuje

vytvárať, manipulovať a dotazovať sa na databázy pomocou Python tried, čo zjednodušuje prácu s dátami a zlepšuje prenositeľnosť kódu medzi rôznymi databázovými systémami.

Pre serverovú časť aplikácie bol následne vytvorený súbor Dockerfile, ktorý definoval spôsob spustenia aplikácie, ktorý bol následne využitý službou Elastic Beanstalk pre spustenie aplikácie v cloudovom prostredí na EC2 inštancií.

## 4.4 Úprava grafickej časti AMON-ES aplikácie

Grafická časť aplikácie AMON-ES musela byť čiastočne upravená, aby bola schopná komunikovať s novou serverovou časťou aplikácie. Zdrojový kód bol upravený s cieľom upraviť premenné cesty k jednotlivým zdrojom. Bola upravená cesta pre prístup k AAC obrázkom a upravená URL pre komunikáciu s REST rozhraním.

Okrem týchto drobných úprav bolo potrebné aktualizovať niektoré knižnice, ktoré boli zastaralé a stratili podporu. Rovnako bola upravená funkcia načítavania dát. V staršej verzii aplikácie bola táto funkcia obmedzená len na určitý časový rozsah. Po úprave je možné vo webovej aplikácii zvoliť ľubovoľný dátum. V prípade, že pre daný dátum nie sú prítomné AMON a AAC dáta, aplikácia na to upozorní.

Na záver bolo potrebné, rovnako ako v prípade serverovej časti aplikácie, pridať súbor Dockerfile, ktorý definuje Docker kontajner pre spustenie grafickej časti aplikácie. Tento kontajner je využívaný službou Elastic Beanstalk pre spustenie aplikácie v cloudovom prostredí.

## 4.5 Lokálne vývojové prostredie

Keďže jedným z cieľov bolo zlepšiť lokálny vývoj aplikácie AMON-ES, v tejto časti si opíšeme akým spôsobom sme to dosiahli.

V predchádzajúcich častiach kapitoly sme ukázali, že aplikácia AMON je rozdelená na dve základné časti, a to serverovú a grafickú. Obe tieto časti majú pridelený samostatný repozitár a sú jedna od druhej odseparované. Vďaka tejto seprácií je možné jednotlivé časti aplikácie vyvýjať a upravovať samostatne pričom každá ma svoju vlastnú git históriu. Obe tieto služby majú definovaný súbor Dockerfile, vďaka ktorému je možné jednoducho túto službu lokálne spustiť v prípade, že máme na lokálnom prostredí nainštalovanú službu Docker.

Napriek výhodám takejto separácie pre potreby lokálneho vývoja a možnosti jednoduchého spustenia, testovania, či dokonca nahrávania aplikácie do cloudo-

vého prostredia sme vytvorili samostatný projektový repozitár nazvaný **ae_compose**. Tento repozitár bol určený, ako hlavný prístupový bod, z ktorého je možné vyvíjať AMON-ES aplikáciu a bol rozdelený na tieto časti:

- Infraštruktúrna časť - tento pričinok obsahuj definície AWS infraštruktúry, ktoré nie je možné spravovať pomocou nástroja Elastic Beanstalk. Príkladom je S3 bucket určený na ukladanie AAC obrázkov.
- Časť služieb - tento priečinok obsahuje samotné služby AMON-ES aplikácie. V minimálnej konfigurácii sa tu nachádza serverová a grafická časť aplikácie.
- Konfiguračné súbory - Tu patria napríklad konfiguračné súbory pre lokálne spustenie aplikácie pomocou služby docker compose, skripty pre vytvorenie AWS infraštruktúry a pod.

Zdrojové kódy jednotlivých služieb sú do projektu pridané pomocou **git submodulov**. Git submoduly umožňujú definovať externé git repozitáre ako súčasť tohto repozitára. Pri ich definovaní, nástroj git vytvorí symbolický odkaz, ktorý špecifikuje url externého repozitára spolu s vetvou, na ktorú sa má odkazovať.

Využitie git submodulov umožňuje jednoduchý prístup ku zdrojovému kódu zvyšných častí aplikácie, pričom všetky tieto časti majú vlastný git repozitár. Pri vývoji je rovnako možné jednoducho upravovať kód týchto častí, pričom tieto zmeny sa uložia len do histórie verzií upravenej služby. Vďaka využitiu tohto nástroja, teda máme zároveň separátny projektový repozitár jednotlivých služieb, rovnako ako aj spôsob, ako ku nim pristupovať na jednom mieste.

Pre jednoduché spustenie celej aplikácie bol vytvorený docker compose súbor, ktorý definoval všetky potrebné Docker obrazy potrebné pre chod aplikácie. Mimo samotnú serverovú a grafickú časť aplikácie je v tomto docker compose súbore definovaná aj služba pre spustenie databázy Postgres. Následne je spustená serverová časť aplikácie, ktorá je pomocou nastavení enviromentálnych premenných nakonfigurovaná tak, aby pristupovala k lokálnej Postgres databáze.

Mimo tieto základné služby, bolo potrebné vyriešiť, ako simulovať S3 Bucket pre ukladanie a prístup k AAC obrázkom. Pre tento účel sme zvoli nástroj **LocalStack**. Táto služba umožňuje simulovať rôzne AWS služby v lokálnom prostredí. Je možné ju spustiť, ako jeden z definovaných kontajnerov pomocou nástroja Docker Compose a následne vytvoriť všetky potrebné služby, ktoré v aplikácii potrebujeme. V našom prípade sme ju použili najmä na vytvorenie S3 bucketu určeného

na ukladanie AAC obrázkov, vďaka čomu je možné pri lokálnom vývoji pristupovať k lokálnemu S3 bucketu pomocou identického rozhrania ako v prípade produkčnej verzie aplikácie.

Jednou z nevýhod tejto služby je, že bezplatná verzia neponúka perzistenciu medzi jednotlivými spusteniami, preto bolo potrebné upraviť službu tak, aby sa počas inicializácie kontajnera LocalStack vždy vytvorili AWS služby nanovo. Tento prístup však vedie k strate dát uložených v predchádzajúcich spusteniach aplikácie. Avšak v rámci lokálneho vývoja to nebol podstatný problém. V prípade, že by bola táto funkcionalita v budúcnosti potrebná, je možné doplniť inicializáciu služieb aj o nahratie vopred definovaných dát, ktoré by predstavovali akýsi štartovací bod aplikácie pre lokálny vývoj.

Vďaka využitiu všetkých vyššie spomenutých nástrojov je možné celú aplikáciu AMON-ES spustiť z jedného miesta príkazom *docker compose up*. Pri spustení tohto príkazu sa automaticky vytvorí lokálna databáza, rovnako ako aj S3 bucket, ktorý beží v lokálnom prostredí. Následne sa spustia serverová a grafická časť aplikácie. Zdrojové kódy oboch častí aplikácie je možné upravovať na jednom mieste, pričom obe majú vlastnú git históriu a samostatný git repozitár, na ktorom je možné v prípade potreby separátne pracovať. Na rozdiel od serverless architektúry, popísanej v predchádzajúcej kapitole, je tento proces rýchlejší a z pohľadu vývojára príjemnejší, keďže všetky časti aplikácie sú spustené lokálne, vďaka čomu je ich možné rýchlo upraviť a jednoducho otestovať.

## 4.6 Proces nahrávanie aplikácie do cloudového prostredie AWS

Proces vytvárania a úpravy architektúry na cloudovej platforme AWS sme sa snažili zjednodušiť najviac, ako to bolo možné. Väčšinu vyššie spomenutej architektúry je možné nakonfigurovať pomocou CLI nástroja služby Elastic Beanstalk. V tomto procese si užívateľ najskôr vytvorí Elastic Beanstalk aplikáciu, pričom počas procesu vytvárania sa mu zobrazí rad otázok, ako napríklad, ktoré dátové centrum AWS chce využiť. Následne je potrebné samostatne vytvoriť Elastic Beanstalk prostredie, ktoré slúži na samotný beh aplikácie. V tomto procese je znovu používateľovi zobrazený rad otázok, ako napríklad aký typ AWS load balancera si želá použiť. Pri procese vytvárania Elastic Beanstalk prostredia je možné pomocou CLI prepínačov zvoliť ďalšie možnosti, ako napríklad vytvorenie RDS databázy, typ a veľkosť vytvorenej databázy, typ EC2 inštancií, minimálny, maximálny a optimálny počet bežiacich EC2 inštancií, bezpečnostnú skupinu, ktorá sa má

jednotlivým inštanciám priradiť.

V rámci projektového repozitára **ae_compose**, ktorý bol spomenutý v prechádzajúcej časti, sú pripravené bash skripty, ktoré definujú všetky potrebné prepínače, ako pri vytváraní aplikácie, tak aj pri vytváraní jej prostredia. Medzi najdôležitejšie nastavenia v Elastic Beanstalk prostredí pre serverovú časť aplikácie, patrí nastavenie správnej bezpečnostnej skupiny EC2 inštancií tak, aby bol povolený zápis to S3 bucketu určeného pre ukladanie AAC obrázkov. Vzhľadom na to, že ani S3 bucket, ani samotná bezpečnostná skupina nie sú súčasťou Elastic Beanstalk prostredia bolo potrebné ich vytvoriť vopred.

Pre tento účel sme použili nástroj Terraform. Tento proces prebiehal podobne ako v prípade serverless architektúry, teda najskôr bolo potrebné inicializovať Terraform backend, ktorý bol určený na uchovávanie stavu spravovanej infraštruktúry. Tento pozostával zo samostatného S3 bucketu a DynamoDB databázy. Pre dosiahnutie jednoduchého vytvárania, sme pre Terrafom backend vytvorili samostatnú Terraform infraštruktúru, ktorú stačí inicializovať raz, na začiatku projektu. Následne sme definovali konfiguračné súbory pre vytvorenie S3 bucketu určeného pre AAC obrázky, definovali potrebné role tak, aby bolo čítanie verejne prístupné a zápis dovolený len špecifickej bezpečnostnej skupine. Následne sme vytvorili konfiguračný súbor pre vytvorenie profilu EC2 inštancie umožňujúcej zápis do S3 bucketu, ktorú je možné využiť pri vytváraní Elastic Beanstalk prostredia. Na záver sme pridali možnosť nahratia SSH kľúčov, ktoré je možné priradiť EC2 inštanciám v Elastic Beanstalk prostredí a vďaka ktorým je možné vzdialene k týmto inštanciám pristupovať.

### 4.6.1 Automatizácia procesu vytvárania cloudovej infraštruktúry

Celý proces vytvárania cloudovej infraštruktúry pozostával z viacerých krokov. Vzhľadom na to, že doterajší vývoj prebiehal na testovacom AWS účte a proces vytvárania produkčného AWS účtu prebiehal na pôde SAV, rozhodli sme sa proces vytvárania tejto infraštruktúry automatizovať, vďaka čomu bude v budúcnosti jednoduchšie replikovať túto infraštruktúru, a to až po vytvorení AWS účtu určeného pre produkčné použitie.

Pre účely automatizácie procesu nasadenia celej aplikácie na cloudovú platformu sme využili nástroj Ansible. Ansible je voľne dostupný nástroj určený na konfiguráciu serverov a nasadzovanie aplikácií. Jeho hlavným cieľom je zjednodušiť a zautomatizovať komplexné úlohy, ktoré sa týkajú konfigurácie a správy infraštruktúry. Ansible sa môže používať pre rôzne operačné systémy a cloudové platformy.

Ansible umožňuje spravovať konfigurácie, nasadiť a riadiť aplikácie na cloudových platformách, ako napríklad AWS, Azure alebo Google Cloud. Ansible môže byť použitý samostatne alebo v kombinácii s inými nástrojmi na automatizáciu, ako napríklad Terraform. V našom prípade môže byť Terraform použitý na vytvorenie infraštruktúry, zatiaľ čo Ansible môže byť použitý na riadenie konfigurácie a nasadzovanie aplikácií na tejto infraštruktúre. Oba tieto nástroje sa v našom prípade dopĺňajú.

Celý proces vytvárania infraštruktúry pozostával z automatického vytvorenia nových SSH kľúčov, vytvorenia infraštruktúry pomocou nástroja Terraform, vytvorenia Elastic Beanstalk aplikácie s využitím vyššie spomenutých bash skriptov a na záver vytvorenia Elastic Beanstalk prostredia nakonfigurovaného tak, aby využíval EC2 profil pre prístup ku S3 bucketu pre AAC obrázky.

## 4.7 Možnosti rozšírenia navrhnutej architektúry

Súčasná architektúra je vďaka využívaniu AWS služieb ako auto scaling a aplikačný load balancer, schopná automaticky reagovať na aktuálnu záťaž a jednoducho horizontálne škálovať spustené EC2 inštancie. Okrem horizontálneho škálovania je možné rovnako jednoducho škálovať aj vertikálne pomocou úpravy typu spustených EC2 inštancií. Obe tieto možnosti poskytujú dostatok priestoru pre škálovanie aplikácie AMON-ES v prípade potreby.

Vzhľadom na to, že aplikácia AMON-ES je kontinuálny projekt, s vysokou pravdepodobnosťou budú do aplikácie pridávané nové funkcie. Jednou z týchto zmien môže byť vylepšenie alebo úprava algoritmu pre analýzu obrazových AAC dát, ktorý je aktuálne spúšťaný priamo pri nahrávaní nových dát do serverovej časti aplikácie. V prípade, že nový algoritmus by vyžadoval viac času na analýzu, je možné v rámci Elastic Beanstalk prostredia pridať komponent nazývaný worker. Worker je v podstate Docker kontajner, ktorého úlohou je vykonať prácu, v tomto prípade spracovanie obrazových dát. Na rozdiel od klasických služieb bežiacich v Elastic Beanstalk prostredí, nie je nutné, aby bol worker spustený celý čas. Stačí, ak bude spustený vtedy, ak budú k dispozícii nové dáta. Existujúci koncový bod aplikácie by bolo potrebné upraviť tak, aby nevykonával spúšťanie analýzy, ale len jej registráciu na spracovanie. Pre tento účel je možné využiť napríklad službu AWS SQS (z angl. Simple Queue Service), ktorú je možné jednoducho pridať do Elastic Beanstalk prostredia.

Vďaka takejto úprave by bolo možné oddeliť proces nahrávania nových dát a samotného spracovania. Okrem toho, v prípade EC2 inštancie workeru je možné

definovať použitie tzv. Spot inštancií, ktoré sú na rozdiel od klasických EC2 inštancií lacnejšie, ale nemusia byť vždy dostupné. V prípade takejto analýzy to však nie je nutné. Ako je možné vidieť, takýmto spôsobom sme schopní kontinuálne pridávať nové komponenty do AMON-ES aplikácie bez výrazných narušení navrhnutej architektúry.

Druhou otázkou v prípade vyššie popísaného rozšírenia aplikácie je, ako vhodne rozšíriť lokálne prostredie tak, aby bolo stále možné spustiť a testovať aplikáciu AMON-ES v lokálnom prostredí. Možností je samozrejme viac, no jednoduchým príkladom môže byť napríklad pridanie novej služby do súboru docker compose určenej na analýzu obrazu a pridanie procesu pre periodické spúšťanie tejto služby spolu s využitím lokálnej služby SQS v nástroji LocalStack. Ako môžeme vidieť v takto upravenom lokálnom prostredí, používame rovnaké rozhranie pre SQS, ako v produkčnom prostredí a výsledný Docker kontajner pre spracovanie obrazu je identický s produkčným, možno s výnimkou niekoľkých environmentálnych premenných. Jednoduchou úpravou vieme dosiahnuť to, že aplikáciu AMON-ES dokážeme lokálne spustiť, testovať a následne nahrať na upravenú architektúru.

Medzi ďalšie plánované rozšírenia aplikácie AMON-ES patrí integrácia GNSS dát, dát zo stanice zbierajúcej dáta o počasí a TLE dát. Integrácia týchto dát by mala byť priamočiara vďaka tomu, že základné prvky architektúry AMON-ES aplikácie sú už pripravené.

## 4.8 Zhrnutie

V tejto kapitole sme sa venovali novému návrhu architektúry AMON-ES aplikácie. V úvode sme si predstavili AWS služby, ktoré boli v kontexte navrhnutej architektúry relevantné. Následne sme predstavili upravenú architektúru, pri ktorej bolo cieľom vytvoriť jednoduché a škálovateľné riešenie, ktoré by bolo možné jednoducho upraviť aj bez hlbokých znalostí o jednotlivých AWS službách. Navrhnutá architektúra pozostáva z viacerých častí, vrátane EC2 inštancií, Auto Scaling group, Target group, Aplikačného load balancera, Elastic Beanstalk prostredia a S3 bucketu. Elastic Beanstalk prostredie umožňuje jednoduchú konfiguráciu všetkých služieb na jednom mieste, spolu s možnosťou aplikácie jednoducho vertikálne a horizontálne škálovať vzhľadom na aktuálnu záťaž. Architektúra bola navrhnutá s ohľadom na kontinuálny vývoj aplikácie a vďaka vyššie spomenutým službám je možné túto aplikáciu jednoducho upravovať s využitím Elastic Beanstalk CLI rozhrania.

Následne sme si predstavili úpravy serverovej a grafickej časti aplikácie. Serverovú časť aplikácie sme celú prepísali s využitím webového rámca FastAPI, upravili jednotlivé koncové body REST rozhrania a upravili prístup k databáze, ktorou sa stala ako v lokálnom, tak aj v produkčnom prostredí, služba Postgres. Na grafickej časti aplikácie sme spravili drobné zmeny, ktoré umožňujú jej kontinuálne použitie aj v prípade novo pridaných dát. Obe tieto časti aplikácie sme s využitím git submodulov a Docker kontajnerov boli schopní využiť v novom projektovom repozitári, nazvanom ae_compose, ktorý slúži na jednoduché spustenie aplikácie, testovanie aplikácie na vytvorenie a úpravu cloudovej infraštruktúry.

Následne sme si popísali, ako je možné aplikáciu vyvíjať v lokálnom prostredí a ako funguje proces nahrávania aplikácie do cloudového prostredia s využitím služieb ako Terraform a Ansible. Následne sme ukázali spôsob, ako je možné túto architektúru jednoducho v prípade potreby doplniť o nové služby.

Využívanie Docker kontajnerov, ako základ jednotlivých komponentov aplikácie AMON-ES nám umožnilo vytvoriť robustnú architektúru schopnú adaptácie, rovnako, ako aj vytvoriť dobre použiteľné lokálne vývojové prostredie. Rovnako je vďaka Docker kontajnerom možné AMON-ES aplikáciu jednoduchšie preniesť napríklad na iné cloudové prostredie alebo na vlastný hardvér, keďže nie sú špecifické len pre AWS. Vďaka tomu je výsledná aplikácia pripravená na ďalší kontinuálny vývoj a pridávanie nových funkcií a komponentov.

# 5 Záver

V tejto práci sme sa venovali návrhu a implementácií softvérového riešenia pre zbieranie, uskladňovanie a monitorovanie airglow dát pochádzajúcich z pozorovacej stanice Ústavu experimentálnej fyziky SAV, v. v. i. umiestnenej na Astronomickom observatóriu na Kolonickom sedle na východe Slovenska. Vysvetlili sme, ako airglow vzniká, aké sú jeho charakteristiky, aké procesy pozemského a vesmírneho počasia ho ovplyvňujú. Rovnako sme si popísali, ako pozorovanie javu airglow súvisí so špecifickými nadmorskými výškami zemskej atmosféry, v ktorých vzniká. Vysvetlili sme, že jednou z motivácií štúdia javu airglow je porozumenie dynamiky hornej vrstvy atmosféry a toho, ako naň pôsobia vesmírne a pozemské faktory.

Opísali sme súčasný stav projektu AMON-ES. Detailnejšie sme predstavili, že pozorovacia stanica pozostáva z viacerých komponentov, ako sú samotné zariadenie AMON, zariadenie AAC na snímanie oblohy, GNSS prijímač či zariadenie na monitorovanie počasia. Opísali sme východiskový stav softvérového riešenia aplikácie, ktoré pozostáva z dvoch základných častí. Obe časti, serverová aj grafická, boli doposiaľ používané len v lokálnom prostredí a neboli nasadené na žiadne produkčné prostredie. Ďalej sme poukázali na to, že niektoré funkcie nevyhnutné pre kontinuálny chod aplikácie chýbali, ako napríklad nahrávanie nových AMON dát alebo AAC obrázkov.

V nasledujúcich kapitolách sme sa sústredili na možnosti využitia cloudovej platformy AWS pre túto aplikáciu. Prvý pokus využíval primárne serverless služby, ktoré platforma AWS ponúka. Navrhli sme architektúru riešenia, ktorá pozostávala z troch základných častí - častí pre spracovanie nových dát, REST aplikačného rozhrania a samotnej grafickej aplikácie. Pri implementácii tejto architektúry sme začali s čiastkovým riešením, a to práve so spracovaním AAC obrázkov, pričom v tomto procese bola čiastočne implementovaná prvá zo spomínaných častí aplikácie. Pri tomto procese sme sa mimo iné sústredili aj na nástroje umožňujúce definovať aplikačnú infraštruktúru ako kód, medzi ktoré patrí napríklad Terraform. Počas implementácie sme narazili na niekoľko problémov, z

ktorých najväčšie boli práve náročný vývoj serverless aplikácie v lokálnom prostredí, ktorý by aj pri dokončenom riešení vyžadoval výraznú znalosť využitých AWS služieb. Druhým problémom boli pamäťové obmedzenia lambda funkcií pri analýze a spracovaní AAC obrázkov, ktoré už v skorých fázach implementácie vyžadovali využitie Docker kontajnerov pri tejto serverless architektúre. Kombinácia týchto faktorov viedla k prehodnoteniu návrhu architektúry.

V poslednej kapitole práce sme sa venovali úprave architektúry s cieľom vytvoriť jednoduché, no zároveň robustné riešenie. Vďaka využitiu služby AWS Elastic Beanstalk, ktorá využíva ďalšie AWS služby ako Auto Scaling, Application Load Balancer či RDS a nástroju Terraform sa nám tento cieľ podarilo splniť. Výsledné riešenie aplikácie umožňuje jednoduché nasadenie a úpravu serverovej a grafickej časti aplikácie. Aplikáciu je možné škálovať horizontálne, a to vzhľadom na aktuálnu záťaž, rovnako ako aj vertikálne jednoduchou úpravou konfigurácie Elastic Beanstalk prostredia. Pri implementácii novo navrhnutej architektúry sme prepracovali serverovú časť aplikácie s využitím moderného webového rámca FastAPI, pridali novú vrstvu pre komunikáciu s databázou s využitím knižnice SQLAlchemy a doplnili chýbajúce koncové body pre pridávanie a spracovanie AMON a AAC dát. Grafickú časť aplikácie sme upravili tak, aby bolo možné jej využívanie aj pri kontinuálnom pridávaní nových dát, čo v predchádzajúcej verzii nebolo možné. Pri úprave týchto častí sme sa sústredili aj na zjednodušenie vývoja aplikácie v lokálnom vývojovom prostredí, čo sa nám vďaka využitiu Docker kontajnerov v prípade serverovej, aj grafickej časti aplikácie jednoducho podarilo pridaním konfigurácie pre službu Docker Compose. V závere poslednej kapitoly sme stručne ukázali, ako je možné výslednú architektúru aplikácie AMON-ES v prípade potreby upraviť a doplniť o nové funkcie. V čase písania tejto práce je aplikácia AMON-ES nasadená v cloudovom prostredí AWS Ústavu experimentálnej fyziky SAV, v. v. i. Aktuálne url adresy pre webovú aplikáciu AMON-ES, rovnako ako aj na REST API je možné nájsť na adrese https://github.com/spacelab-sk/amon-es.

# Literatúra


1. CHATTOPADHYAY, Rabindranath; MIDYA, S. Airglow emissions: Fundamentals of theory and experiment. *Indian Journal of Physics*. 2006, roč. 80.
2. AL., Mackovjak et. *Technical Note of Theoretical Study (TN-TS)* [online] [cit. 2022-06-22]. Dostupné z : `https://github.com/space-lab-sk/amon-es/blob/master/documentation/2019_SK2-09_TN-TS.pdf`.
3. AL., Mackovjak et. *The 2nd Technical Note of AMON-EN (TN2-AE)* [online] [cit. 2022-06-22]. Dostupné z : `https://github.com/space-lab-sk/amon-es/blob/master/documentation/2020_SK2-09_TN2-AE.pdf`.
4. PFAFF, Robert. The Near-Earth Plasma Environment. *Space Science Reviews*. 2012, roč. 168. ISBN 978-1-4614-5676-6. Dostupné z DOI: `10.1007/s11214-012-9872-6`.
5. MACKOVJAK, Š.; BOBÍK, P.; BALÁŽ, J.; STRHÁRSKÝ, I.; PUTIŠ, M.; GORODETZKY, P. Airglow monitoring by one-pixel detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2019, roč. 922, s. 150–156. ISSN 0168-9002. Dostupné z DOI: `https://doi.org/10.1016/j.nima.2018.12.073`.
6. AL., Mackovjak et. *The 1st Technical Note of AMON-EN (TN1-AE)* [online] [cit. 2022-06-22]. Dostupné z : `https://github.com/space-lab-sk/amon-es/blob/master/documentation/2019_SK2-09_TN1-AE.pdf`.
7. AMRICH, S.; MACKOVJAK, Š.; STRHÁRSKÝ, I.; BALÁŽ, J.; HANČIKOVSKÝ, M. Design and construction of hardware and software for autonomous observations of Transient Luminous Events. *Journal of Instrumentation*. 2021, roč. 16, č. 12, T12016. Dostupné z DOI: `10.1088/1748-0221/16/12/T12016`.
8. *Welcome to Flask!* [Online] [cit. 2023-03-10]. Dostupné z : `https://flask.palletsprojects.com/en/2.2.x/`.
9. *Welcome to Flask-RESTPlus's documentation!* [Online] [cit. 2023-03-10]. Dostupné z : `https://flask-restplus.readthedocs.io/en/stable/`.

10. *Swagger documentation* [online] [cit. 2023-03-10]. Dostupné z : https://flask-restplus.readthedocs.io/en/stable/swagger.html.

11. *Quick Start* [online] [cit. 2023-03-10]. Dostupné z : https://flask-restplus.readthedocs.io/en/stable/quickstart.html.

12. *OpenAPI Specification v3.1.0* [online]. 2021 [cit. 2023-03-10]. Dostupné z : https://spec.openapis.org/oas/latest.html#openapi-specification.

13. *About SQLite* [online]. 2018 [cit. 2023-03-10]. Dostupné z : https://sqlite.org/about.html.

14. *Introduction to Dash* [online] [cit. 2023-03-10]. Dostupné z : https://dash.plotly.com/introduction.

15. PLOTLY. *Introducing Dash* [online]. 2017 [cit. 2023-03-10]. Dostupné z : https://medium.com/plotly/introducing-dash-5ecf7191b503.

16. SERVICES, Amazon Web. *Overview of Amazon Web Services* [online] [cit. 2022-06-22]. Tech. spr. Amazon Web Services. Dostupné z : https://docs.aws.amazon.com/whitepapers/latest/aws-overview/aws-overview.pdf#introduction.

17. *What is AWS Lambda* [online] [cit. 2022-06-23]. Dostupné z : What%20is%20AWS%20Lambda.

18. *What is AWS Step Functions* [online] [cit. 2022-06-23]. Dostupné z : https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html.

19. *Amazon S3 Storage Classes* [online] [cit. 2022-06-23]. Dostupné z : https://aws.amazon.com/s3/storage-classes.

20. *What is Amazon DynamoDB?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

21. *Amazon DynamoDB pricing* [online] [cit. 2023-01-29]. Dostupné z : https://aws.amazon.com/dynamodb/pricing/?refid=1509de88-c72e-427e-a847-6a914fc95d08.

22. *What Is Amazon EventBridge?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html.

23. *What is AWS CloudFormation?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.

24. *What is Terraform?* [Online] [cit. 2023-01-29]. Dostupné z : https://developer.hashicorp.com/terraform/intro.

25. *State* [online] [cit. 2023-01-29]. Dostupné z : https://developer.hashicorp.com/terraform/language/state.

26. *Architecture & Concepts* [online] [cit. 2023-03-10]. Dostupné z : https://www.pulumi.com/docs/intro/concepts.

27. *Supported Languages* [online] [cit. 2023-03-10]. Dostupné z : https://www.pulumi.com/docs/intro/languages/.

28. *Auto Scaling groups* [online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html.

29. *What is Elastic Load Balancing?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html.

30. *How Elastic Load Balancing works* [online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html.

31. *Target groups for your Application Load Balancers* [online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-target-groups.html.

32. *What is Amazon Relational Database Service (Amazon RDS)?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html.

33. *What is AWS Fargate?* [Online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/AmazonECS/latest/userguide/what-is-fargate.html.

34. *What is Amazon Elastic Container Registry* [online] [cit. 2023-01-29]. Dostupné z : https://aws.amazon.com/ecr/.

35. *AWS Elastic Beanstalk Developer Guide* [online] [cit. 2023-01-29]. Dostupné z : https://docs.aws.amazon.com/pdfs/elasticbeanstalk/latest/dg/awseb-dg.pdf.

# Zoznam skratiek

**AAC** z angl. AMON All-Sky Camera.

**AMON** Airglow MONitor.

**AMON-ES** Airglow MONitor - Extended Station.

**AWS** Amazon Web Services.

**EUV** (z angl. Extreme ultraviolet).

**GNSS** (z angl. Global Navigation Satellite System).

**IaC** (z angl. Infrastructure as a code).

**ISS** (z angl. International Space Station).

**TLE** (z angl. Transient Luminous Events).