Tervezői dokumentáció

Adatok tárolását, feldolgozását, megjelentését támogató informatikai rendszer feladat 3. fázisához kapcsolódóan

Tartalomjegyzék

1 A feladat meghatározása

Feladat megnevezése: adatok tárolását, feldolgozását, megjelentését támogató informatikai rendszer

Fázis 1: Az elektroszmog szakterület backend moduljának elkészítése mely megvalósítja az adatok fogadását, tárolását, elemzését, lekérdezhetővé tételét, push értesítést és felhasználókezelést. A fejlesztés során mikroszolgáltatásokra épülő szerver oldali architektúrát alkalmazunk, melynek révén biztosított lesz az egyes szerver oldali komponensek egységbe zárása, skálázhatósága. Adattárolásra open-source relációs és noSQL adatbázisokat alkalmazunk.

Fázis 2: Az elektroszmog tématerület webes kliensének fejlesztése, mely lehetővé teszi az összegyűjtött és elemzett adatok megjelenítését, monitorozást, nyomonkövetést modern webes felületen. A kliens felületen kereshetővé válnak az egyes területek térképes nézetben. Jól használható térképek készülnek az összegyűjtött adatokra alapozottan, melyek szemléltetik az adott területen mért elektroszmog mértékét.

Fázis 3: Az elektroszmog szakterület backend moduljának második változata: elemzés, lekérdezhetővé tétel, push értesítés és felhasználókezelés.

Az elektroszmog tématerület webes kliensének második változata: monitorozás, nyomonkövetés modern webes felületen.

Jelen dokumentum a harmadik fázisban történt kutatási-fejlesztési feladatokat összegzi.

2 Az elkészített szoftverrendszer specifikációja

2.1 A projekt célja

Az Elektroszmog projekt célja az 30Hz – 6GHz között terjedő elektromágneses spektrumban történő térerősség mérések készítése és ezen mérési eredmények tárolása, elemzése és vizualizációja. A dokumentumban tárgyalt projekt az adatok mérőeszközről történő fogadását, feldolgozását, elemzését és vizualizálását tartalmazza. A mérőeszközök és mérési fájlok készítését harmadik fél végzi. A méréseket mobil mérőállomások készítik, az adatok megtekintésére és elemzésére modern webes felületeken lesz lehetőség.

Az elemzések segítségével megállapítható adott pontokban a különböző frekvenciákon mérhető térerősségek alakulása az idő függvényében, vagy adott frekvenciák észlelhetősége pozíciótól függően. Az adatok használhatók az elektromágneses sugárzás egészségügyi hatásainak vizsgálatára, mobiltelefon, rádió és TV műsor szolgáltatók és egyéb rádiós szolgálatok lefedettségének elemzésére, elektromágneses zavarforrások körüli térerősség eloszlás feltérképezésére, stb.

2.2 Követelmények

2.2.1 Adatfogadás

A rendszernek fogadnia kell a mérő rendszerből érkező adatfájlokat és ezeket fel kell tudnia dolgozni. Az adatfájlok formátumát a mérőeszközt készítő partnerrel folyamatosan egyeztetjük, CSV alapú formátumot használunk.

Az első mérföldkőig az adatok betöltése manuális, vagy félig automatizált módon történik. Mivel az adatok automatikus továbbítása mobil telefonhálózaton, vagy valamilyen más rádiós összeköttetésen keresztül volna lehetséges, az ennek során használt rádiójelek befolyásolnák a mérések eredményét. Ezért az adatfájlokat a rendszer nem kérdezi le automatikusan a mérőautóról, azokat kézzel kell egy kinevezett mappába eljuttatni, ahonnan a rendszer felolvassa őket.

A rendszer legyen képes egy, a rendszerrel azonos gépen lévő mappából mérési fájlokat automatizáltan felolvasni. A rendszernek érzékelnie kell, ha a mappába új fájl került és be kell azt olvasnia.

A feldolgozott fájlokat a rendszer módosított névvel, az eredeti mappában meg kell tartsa.

2.2.2 Adatfeldolgozás

A rendszer az adatokat a tároláshoz és későbbi elemzéshez hatékony formába kell, hogy alakítsa. Ezen feldolgozás során ki kell számítania bizonyos értékeket és el kell végeznie elemzéseket, ezek a következők:

- Mérés frekvenciatartományainak meghatározása a file fejléce alapján
- Mérés útvonalát befoglaló minimális poligon számítása
- Térerősség négyzetösszeg logaritmikus képlet alapján, a kezelő által megadott frekvenciasávban:
    - képlet:

$$E_{eredő} = \sqrt{\sum_{i=1}^{n} E(f_i)^2}$$

- ahol: *i* a kijelölt frekvenciasávban adódó amplitúdósűrűség spektrum helyi maximumainak sorszáma
- f_i az i-edik helyi maximum frekvenciája
- $E(f_i)$ az elektromos térerősség az f_i frekvencián, [*dBµV/m*]
- $E_{eredő}$ az összegzés során adódó helyettesítő térerősség, [*dBµV/m*]

  - A mérés során adódó *dBµV/m* logaritmikus térerősség mértékegységet használva:
    - $E_{eredő}^{dB\mu V/m} = 10 \cdot log \sum_{i=1}^{n} invlog \frac{E^{dB\mu V/m}(f_i)}{10}$

- Legnagyobb térerősséggel jelenlevő frekvencia meghatározása

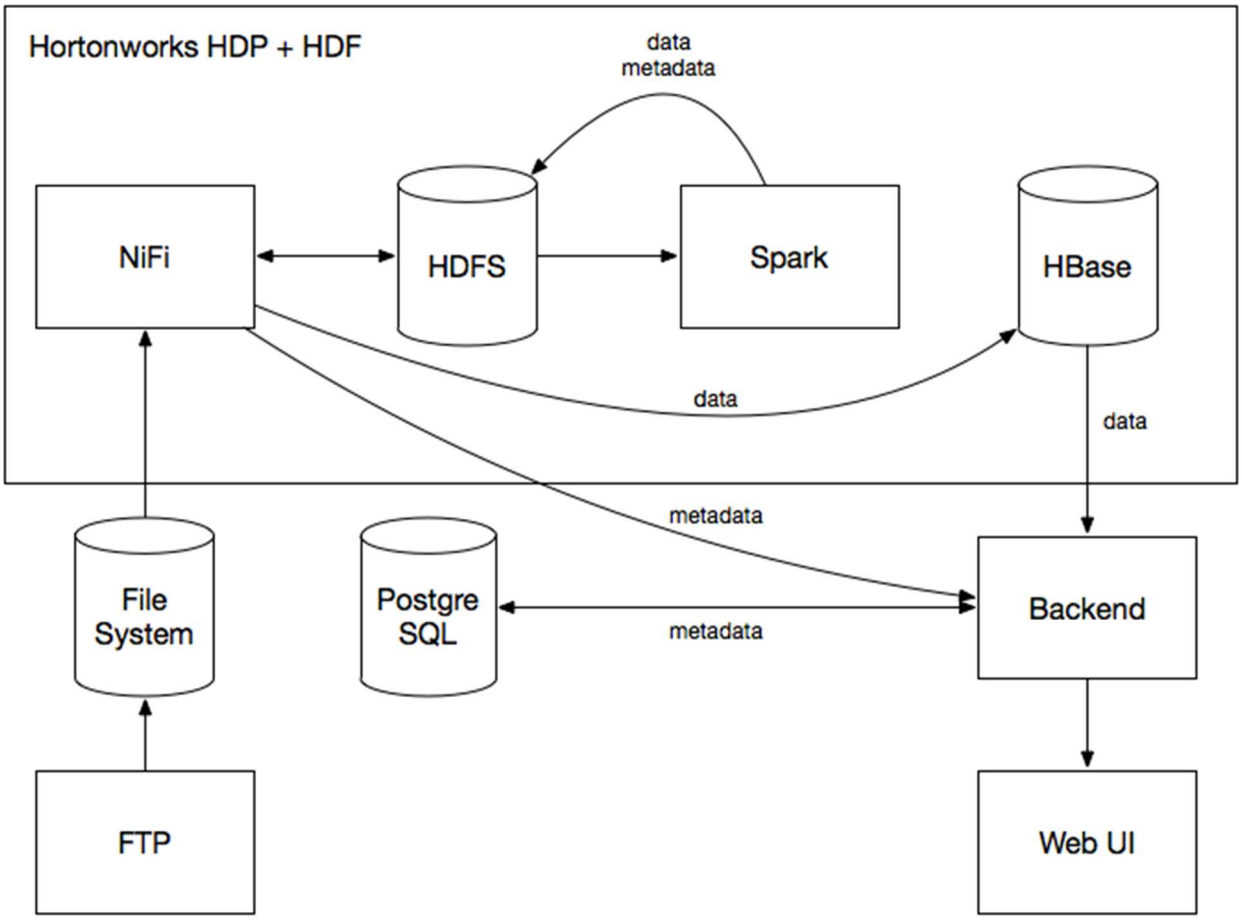
3 Rendszerterv

Az elektroszmog projekt során, hasonlóan más IoT projektekhez olyan bejövő adatok feldolgozására van szükség, amelyek nagy gyakorisággal érkezhetnek, és méretükben hamar túllépik a hagyományos, relációs adatbázisokon alapuló eszközök határait. Ezen megfontolásból a rendszer architektúráját olyan módon kellett kialakítani, hogy a nagyméretű adatok betöltése, tárolása és megjelenítése hatékonyan, gyors válaszidővel és megbízhatóan megtörténjen.

A fejlesztéshez a korábbi fázisokban bemutatott IoT platform, a SensorHUB megoldásait vettük alapul, amelyek az ilyen típusú kihívások megoldására optimalizáltak. A következőkben bemutatásra kerül a rendszer architektúrája, az abban megjelenő adatfolyamok, illetve a használt adattárolási megoldások.

3.1 Architektúra

A rendszer architektúráját az alábbi ábra szemlélteti.



Ábra 1 Architektúra diagram

A rendszer számos komponensből áll melyek együttes működése lehetővé teszi a nagyméretű adathalmazok begyűjtését, tárolását és feldolgozását. Az egyes komponensek rövid összefoglalóját a következőkben tárgyaljuk.

3.1.1 FTP

Az adatfájlokat FTP-n keresztül lehet eljuttatni a rendszerbe. Ez a technológia lehetővé teszi a nagyméretű fájlok egyszerű eljuttatását forrásrendszerből a célrendszerbe.

3.1.2 File system

Az FTP-n feltöltött fájlok a mester gép helyi fájlrendszerébe kerülnek eltárolásra további feldolgozást megelőzően.

3.1.3 Hortonworks HDP, HDF

A projekt jellegéből adódóan nagyméretű adathalmazokat kell a rendszernek kezelnie. Ilyen feladatokra ideális megoldást kínálnak a big data szoftverek, melyek tipikusan gyorsan keletkező, nagyméretű adatok tárolását és feldolgozását segítik, jellemzően elosztott módon. Ezeknek a szoftvereknek az ökoszisztémáját Hadoopnak nevezik. A Hortonworks az egyik legnagyobb Hadoop disztribútor, aki számos big data szoftvert konfigurál össze két nagy termékébe a HDF-be és HDP-be. A projekt során mindkét termékből használunk komponenseket.

3.1.3.1 Nifi

Az Apache NiFi egy az Apache Software Foundation által karbantartott szoftver, mely segítségével adatfolyamokat menedzselhetünk és automatizálhatunk. A projekt igen népszerű, többek között azon oknál fogva, hogy számos adatforrással és célponttal tud dolgozni, valamint kiterjedt lehetőségeket biztosít az adatok feldolgozására is. Felhasználási területei igen széleskörűek, mi egyfajta ETL (Extract Transform Load, adatmozgatás és transzformáció rendszerek közt) eszközként fogunk rá tekinteni, amely segít az adatok különböző forrásokból történő betöltésében, előfeldolgozásában.

Fontos fogalmak:

1. ***FlowFile**:* Egy FlowFile lényegében egy csomagként fogható fel, amely a rendszerben halad az egyes adatfolyamok mentén. Minden FlowFile két elemből áll össze, a metaadatokat tartalmazó attribútumokból, és a FlowFilehoz tartozó adat tartalmából.
2. ***FlowFile Processor**:* A lényegi munkát a Processorok végzik el. Feladatuk lehet az adat transzformálása, routeolása, vagy betöltése valamilyen külső rendszerbe. A Processorok hozzáférnek a FlowFileok attribútumaihoz és tartalmához is.
3. ***Connection**:* Az egyes Processorokat valamilyen módon össze kell kötni, ebben segítenek a Connectionök. Annak érdekében, hogy a különböző sebességgel működő Processorok összeköthetők legyenek, a köztük lévő kapcsolatok egyfajta várakozási sorként is működnek, melyek paraméterei konfigurálhatók.
4. ***Flow Controller**:* Egyfajta ütemezőként működik, amely az egyes Processorok számára fenntartott szálakat és erőforrásokat kezeli.
5. ***Process Group**:* Feldolgozási egység, amely tartalmazhat Processorokat és Connectionöket. Fogadhat, illetve küldhet adatot az Input és Output portjain keresztül. Tipikusan a különböző absztrakciós szinten mozgó feldolgozási elemek egységbe foglalására használjuk.

3.1.3.2 HDFS

A HDFS vagy Hadoop Distributed File System egy elosztott hibatűrő fájlrendszer, ami nagy adathalmazok tárolására lett kifejlesztve. A rendszer több fizikai számítógép merevlemezei felett valósít meg egy virtuális fájlrendszert a felhasználók elől elrejtve azt, hogy a valóságban nem egy gépen dolgoznak. A HDFS a hibatűrő funkcióját replikációval vagy hibajavító kódok használatával éri el. Klaszter mérettől és beállításoktól függően, de elmondható, hogy együkét gép kiesése nem jelent problémát, adataink továbbra is biztonságban lesznek, a rendszer pedig automatikusan újra replikálja azokat.

Felépítését tekintve egy kitüntetett és számos tároló gépből áll az architektúrája. A NameNode, amiből jellemzően egy van egy installációban felelős a metaadatok tárolásáért. Ez a gép tartja nyilván, hogy a HDFS-en tárolt fájlok darabja hol vannak, egy olvasási vagy írási művelet során ez a gép tudja megmondani, hogy hol keressük az elérni kívánt fájlt vagy hova írhatjuk az új fájl blokkjait. A DataNode-ok feladata az adatok tárolása, ez a NameNode által hozzájuk irányított fájlblokkok tárolásában és a NameNode felé heartbeatek küldésében merül ki.

Fontos megemlíteni, hogy az adatok nem a NameNode-on áramlanak keresztül, így az nem jelent szűk keresztmetszetet, míg az adatok több példányban történő tárolása jó hatással van az olvasási sebességre, hiszen az egyes blokkokat más-más merevlemezről felolvasva a sebesség megtöbbszöröződhet.

3.1.3.3 Spark

Az Apache Spark egy elosztottan működni képes adatfeldolgozó motor big data környezetekhez. Alap koncepciója, hogy úgynevezett adatpárhuzamos problémák esetén, ahol ugyan azt a műveletet kell elvégezni egymástól független adatrekordokon egyszerű megoldást kínáljon. Java, Scala vagy Python nyelven, funkcionális stílusban írt kódot tud futtatni a klaszter erőforrásait kihasználva, aminek köszönhetően nagyméretű adathalmazokkal is könnyen megbirkózik. Kihasználja az adatlokalitást, azaz egy fájlt lehetőség szerint azon a gépen fog feldolgozni ahol az tárolva van, ezzel is csökkentve a költséges és lassú hálózati kommunikációt.

A Spark programozás központi fogalma az RDD - Resilient Distributed Dataset, ami egy programozásból jól ismert listaként képzelhető el, azzal a különbséggel, hogy ennek a listának a partíciói különböző fizikai gépeken vannak. Ezeken az RDD adatstruktúrákon hajthatjuk végre a Spark számos funkcionális metódusát, melyeket két nagy csoportra bonthatunk: transzformációkra és akciókra. Transzformáció során az RDD-ből egy másik RDD jön létre, ilyen a map, vagy filter művelet, amikkel az RDD elemeit tudjuk átalakítani vagy megszűrni. Akció esetén az RDD-ből jellemzően valamilyen skalár érték képződik, ilyen a count, sum stb.

3.1.3.4 HBase

Az Apache HBase egy a Google által kiadott cikk alapján épített nagy adathalmazok tárolására felkészített adatbázisredszer. Az HBase a HDFS-t használja az adatok tárolására, a HDFS biztosítja az adatok megbízható tárolását. A rendszer adatstruktúrája és működése merőben eltér a hagyományos SQL alapú adatbázisoktól, így egész más megközelítést igényel a tervezéstől kezdve. Elmondható, hogy a legtöbb NoSQL alapú adatbázisnál az adatséma tervezés első lépése a jövőben futtatandó lekérdezések

átgondolása, ezek alapján kell megtervezni a modelleket, gyakran duplikáció, redundáns adattárolás árán is.

Az HBase a CAP tétel alapján CP kategóriába eső rendszer. A CAP tétel kimondja, hogy egy elosztott adattároló megoldás a C (consistency - konzisztencia), A (availability - rendelkezésreállás), P (partition tolerability - partícionálás tűrés) csak kettőnek felelhet meg. Ez alapján az HBase konzisztens és partícionálás tűrő, ami azt jelenti, hogy bármely időpillanatban egy adat kiolvasás a legfrissebb értéket vagy hibát fog adni, valamint a rendszer működőképes marad a csomópontok közti kapcsolatmegszakadás vagy üzenetvesztés esetén is. Az elosztott adatbázisoknál mindig fennáll a kérdés, hogy hálózati hiba, azaz partícionálás esetén a konzisztencia az elsőleges vagy az, hogy a kliens mindenképp választ kapjon, akár akkor is ha az nem a legfrissebb adat.

Adatmodelljét tekintve bigtable alapú rendszer. Ez egy oszlop alapú tárolást alkalmazó struktúra, szokás wide column storenak is nevezni. Adatsruktúrája legegyszerűbben egy a programozásból ismert map adatstruktúrával írható le: Map<rowkey, Map<column, data>>. HBase ben sorok kerülnek tárolásra, előre meghatározott séma nincs. Minden sornak van egy kulcsa (rowkey), a soron belül oszlopcsaládok és azokban oszlopok találhatók, melyek teljesen dinamikusan hozhatók létre. Egy adatcellát egyértelműen azonosít a sorkulcs, oszlopcsalád és oszlopnév hármas. Ez alapján egy kulcs - érték tárnak is felfogható.

3.1.4 PostgreSQL

A rendszer a mérések metaadatat (hely, idő, berendezés konfiguráció), SQL alapokon egy PostgreSQL szerverben tárolja. Ezen adatok alapján lehet a webes felületen szűrni, keresni a mérések közt. Az SQL egy nagyon elterjedt és közkedvelt nyelv és tárolási megoldás, ami jó alapot biztosít a metaadatokon való szűréshez.

3.1.5 Backend

Java Spring technológiával készült szerver alkalmazás felelős az a webes grafikus felület kéréseinek kiszzlgálásáért. Az SQL szerverben tárolt metaadat fájlok alapján szűréseket futtat és az ezekhez tartozó nyers adatokat HBase-ből kinyerve továbbítja a grafikus felületnek.

3.1.6 Web UI

A webes felületen keresztül van lehetőség az alkalmazás funkcióinak elérésére. A mérések között azok metaadatai alapján szűrhetünk és számos diagramon, térképen, grafikonon vizualizált adatokat tekinthetünk meg.

3.2 Adatfolyam

A rendszerbe került adatok számos lépésen haladnak át míg tárolásra kerülnek HBase-ben. A folyamat során szétválogatásra kerülnek a mérés beállításait tartalmazó metaadatok és maguk amért értékek, valamint olyan formába konvertálódnak, ami hatékonyan tárolható és a későbbiekben jól szűrhető.

Az adatok kezdetben CSV formában kerülnek feltöltésre az FTP kiszolgálóra, ami egy a master szerveren kialakított mappába helyezi őket. Ezt a mappát egy NiFi flow figyeli és óránként beolvassa az új fájlokat,

amiket átmásol a HDFS-re. Itt már hozzáférhetők a Spark számára feldolgozásra. A Spark programot a NiFi indítja a fájlok másolását követően, futásának eredménye egy metaadat és egy mérési adat fájl, amiket HDFS-be tárol. A mérési adatokat tároló fájlt a NiFi betölti HBase-be, míg a metaadat fájlt egy HTTP POST kéréssel elküldi az alkalmazásszervernek, ami eltárolja azokat az operációs SQL adatbázisba.

Miután az adatok a fenti módon tárolásra kerültek, a szerver a webes kliens kéréseit kiszolgálva a PostgreSQL és HBase ben tárolt adatok alapján válaszol.

3.3 Adattárolás

Az elkészült szoftverrendszer által tárolt adatok két fő csoportra bonthatók az adatok felhasználási jellege és formátuma szerint:

- Metaadatok
- Mérési adatok.

A metaadatok tartalmazzák azokat az operatív információkat, amelyekre a mérések közötti keresés során szükség lehet. A mérési adatok a mérőműszerekből kinyert nyers adatok transzformációjával állnak elő. Itt kerülnek eltárolásra az egyes mérési pontokhoz tartozó időbélyegek, földrajzi pozíciók, valamint a vizsgált frekvenciákon megfigyelhető elektromágneses térerősség értékek.

Ezen adatok különböző adattárolási eljárásokat kívánnak meg, hiszen míg a metaadatok kis mennyiségben állnak rendelkezésre, de rendkívül gyors, véletlenszerű elérést követelnek meg, addig a mérési adatok több gigabyte méretűek is lehetnek egy mésére vonatkozóan, valamint azokat a szekvenciális adatelérésre kell optimalizálni.

A metaadatok tárolásához ideális megoldás egy relációs adatbáziskezelő használata. Az adatbázisban egyetlen tábla sorai jelképezik a méréseket, azok nevével, kezdő- és végidőpontjaival, mérési tartományával, befoglaló poligonjukkal. Utóbbi tárolása miatt olyan DBMS megválasztására van szükség, amely a geospatiális adatok tárolását és lekérdezését hatékonyan meg tudja valósítani. Ezen feltételrendszer mentén került kiválasztásra a PostgreSQL adatbáziskezelő rendszer. Az alábbi SQL DDL utasítás írja le a metaadatokat tartalmazó tábla sémáját:

```
create table measurement_session
(
     id varchar(255) not null
           constraint measurement_session_pkey
                 primary key,
     end_date integer,
     end_frequency integer,
     end_point bytea,
     frequency_range integer,
     is_stationary boolean,
     name varchar(255),
     point bytea,
     polygon bytea,
     start_date integer,
     start_frequency integer,
     start_point bytea
);
```

A mérési adatok jellege azonban nem illeszkedik jól a relációs adatmodellhez, nagy mennyiségű adat tárolása esetén a tárigény és a beolvasási idő olyan mértékben növekedne, amely a rendszer használhatóságát nagyban lerontaná. Ezen adatok tárolására az Apache Software Foundation által karbantartott Apache HBase adatkezelő rendszer felhasználása mellett döntöttünk.

Az Apache HBase egy nyílt forráskódú, nem-relációs adatbázis, amely a Hadoop Distributed File System (HDFS) felett fut. Előnye, hogy nagy adathalmazokhoz biztosít valós idejű olvasási és írási hozzáférést. Az Elektroszmog rendszer üzemeltetése során nagy segítséget nyújt, hogy az HBase adatbázisok lineárisan skálázhatók, így könnyedén kezelhetők vele több milliárd sort és több millió oszlopot tartalmazó adathalmazok is. Az HBase adatbáziskezelőre nagy mértékben támaszkodik a SensorHUB IoT platform, az integráció rendkívül könnyen megoldható.

Az HBase adatbázistáblák oszlopai úgynevezett oszlopcsaládokba szerveződnek, amelyekbe azokat az adatokat érdemes szervezni, amelyek tipikusan együtt kerülnek felolvasásra. Az oszlopcsaládokon belül az oszlopok nevei és száma teljesen kötetlen, ezzel nagyfokú rugalmasságot biztosítva.

A mérési adatok tárolása olyan módon kerül tárolásra, hogy az egy mérésen belül lévő egyes mérési pontokat reprezentálják a tábla egyes sorai. Egy sor két oszlopcsaládra van felbontva, ezek egyike a mérési pont leírását tartalmazza úgy, mint az időbélyeg vagy az adott pillanatban a mérőautó földrajzi pozíciója. A második oszlopcsalád a mérési frekvenciákat, és a hozzájuk tartozó térerősség értékeket tartalmazza.

F column family			M column family				
timestamp	latitude	longitude	80000	80010	...	...	120000
1523872700	47.497913	19.040236	1.35	1.24	...	...	12.88

Ábra 2 Példa adatsor a mérési adatokat tartalmazó adatbázisból

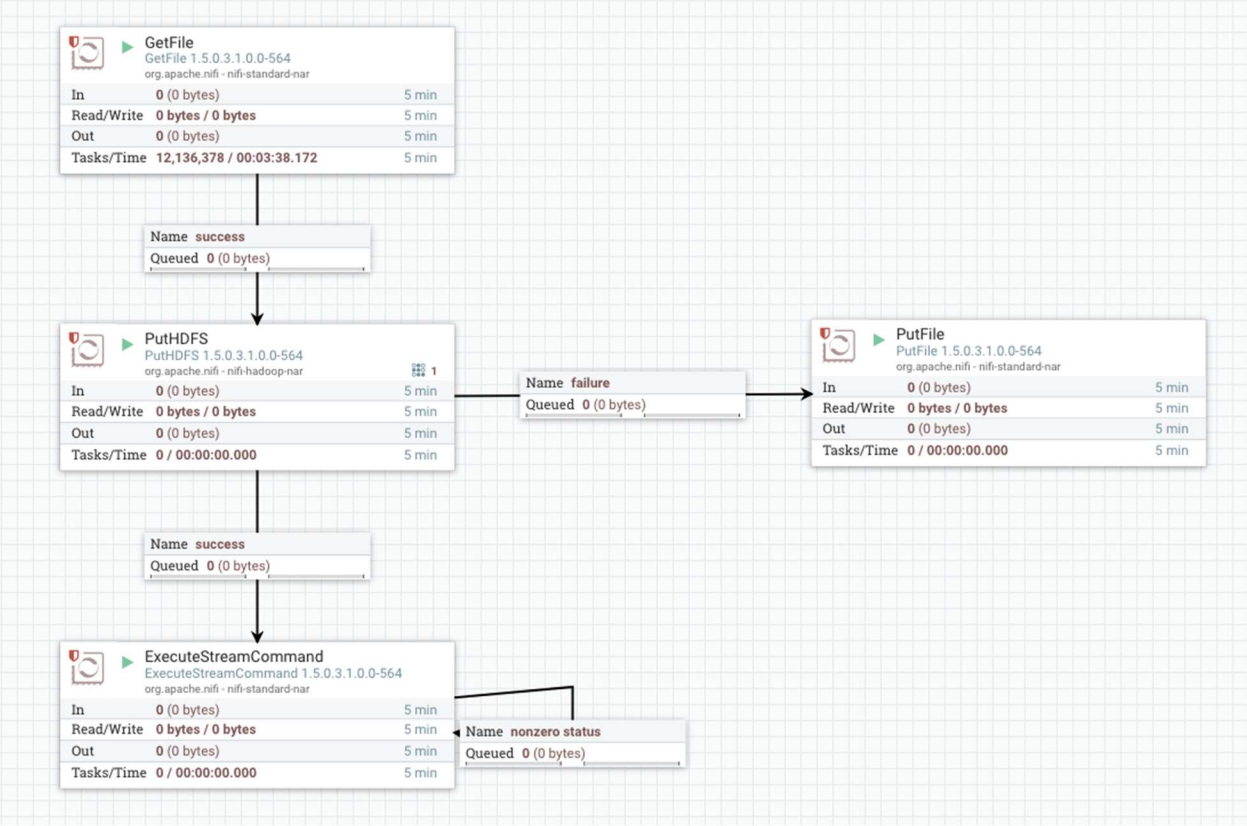
4 Elkészült komponensek fejlesztői dokumentációja

4.1 Apache NiFi adatfolyam

A rendszerarchitektúrának megfelelően a clusteren belül történő adatmozgatások az Apache NiFi adatfolyamkezelő alkalmazáson keresztül kerültek megvalósításra. A rendszerben három különböző adatfolyam került bevezetésre, a következőkben ezek kerülnek bemutatásra.

4.1.1 Nyers adatok betöltése és előfeldolgozása

Az FTP protokollon keresztül beérkező adatok először a cluster master nodeján kerülnek eltárolásra, ezt a megbízható, elosztott tárolás érdekében szükséges átmozgatni a HDFS fájlrendszerre, valamint ezen lépés után elindítható az adatok előfeldolgozása a következő fejezetben ismertetett Apache Spark adatfeldolgozó modul segítségével.



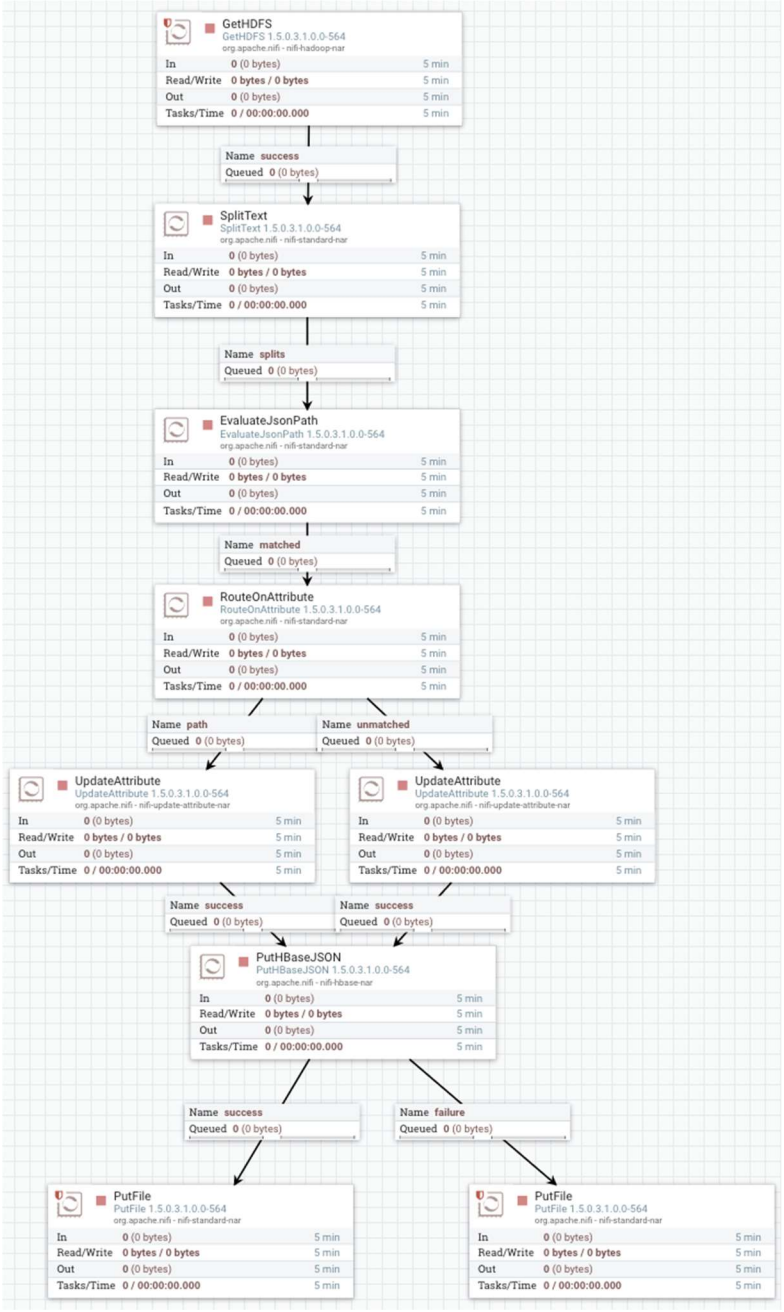
Ábra 3 Nyers adatok betöltéséért és előfeldolgozásáért felelős adatfolyam

A fenti ábrán látható az adatfolyam sematikus felépítése. Első lépésben a GetFile processor segítségével a master node fájlrendszeréről felolvasásra kerülnek a nyers adatok, majd az azt követő PutHDFS processzor elmenti azokat a HDFS-re. A GetFile processzor a beérkező adatokat a felolvasás után törli, ezért fontos, hogy abban az esetben, ha a HDFS-re írás sikertelen, azok kerüljenek újra visszaírásra a master nodera, az adatvesztések elkerülése érdekében.

Amennyiben a fájlok már elérhetők a HDFS-en, elindulhat az adatok előfeldolgozása, amelyet az ExecuteStreamCommand processor indít. A processor feladata, hogy a bejövő adatok hatására futtasson egy bash scriptet, amely a háttérben egy megfelelően konfigurált spark-submit parancsot ad ki. Amennyiben az adatfeldolgozás sikertelen, az ExecuteStreamCommand processor újra próbálkozik.

4.1.2 Előfeldolgozott adatok eltárolása az HBase adatbázisban

A mérési adatok az előfeldolgozás után elmenthetőek az HBase adatbázisba. A következő adatfolyam ennek a megvalósításáért felel.



Ábra 4 Előfeldolgozott adatok betöltése HBase adatbázisba

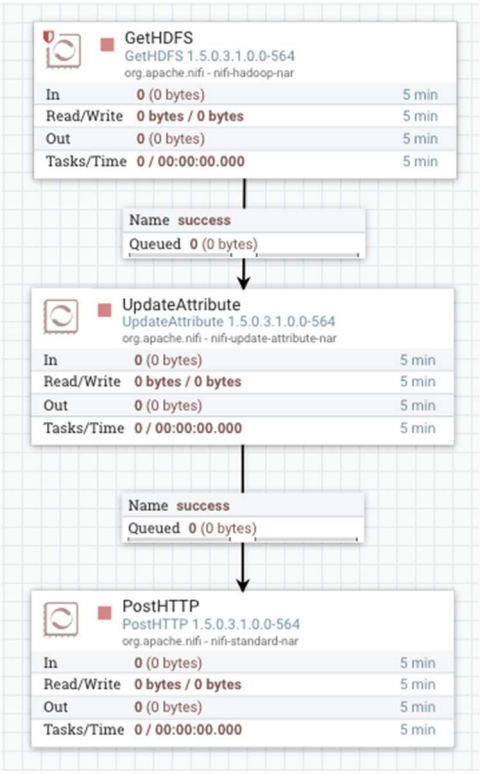
Az első, GetHDFS processor előre meghatározott időközönként ellenőrzi a számára bekonfigurált HDFS könyvtárat, és amennyiben ott új adatállományokat talál, azokat beolvassa. Az előfeldolgozott adatfájl soronként egy HBase oszlopcsalád oszlopait, és az azokhoz tartozó értékeket tartalmazza, így a következő lépésben szükséges a fájl sorokra bontása.

Az adatszerkezet két oszlopcsaládból áll, az első a mérési ponthoz tartozó paramétereket tartalmazza, a második pedig a konkrét mérési frekvenciákat és a mért értékeket. Ezeket az EvaluateJsonPath segítségével különböztethetjük meg, majd a RouteOnAttribute és UpdateAttribute műveletek segítségével beállítunk egy-egy attribútumot, amelyek alapján a PutHBaseJSON processor egyszerűen be tudja azokat szúrni az HBase adatbázisba.

Mind a sikeresen, mind a sikertelenül feldolgozott állományok újra tárolásra kerülnek a HDFS fájlrendszeren.

4.1.3 Metaadatok mentése az operációs adatbázisba

Az előfeldolgozás során előállnak az egyes mérésekről olyan metaadatok is, amelyek az alkalmazás operatív működéséhez szükségesek. Továbbá, kiszámolásra kerülnek aggregált értékek is, mint az eredő térerősség, a legjelentősebb frekvencia, valamint mozgó mérések esetén a mérési pontokat minimálisan befoglaló polygon is. Ezen adatok a korábbiakban ismertetettek szerint egy relációs adatbázisban kerülnek eltárolásra.



Ábra 5 Metaadatok tárolásáért felelős adatfolyam

A jelen adatfolyam implementációja jelentősen egyszerűbb a korábbiaknál. A GetHDFS processor segítségével történik a metainformációkat tartalmazó könyvtár figyelése, majd az ott megjelenő adatokat a PostHTTP processor segítségével a backend modulon keresztül az adatbázisba mentjük. Az UpdateAttribute processor segítségével kerül beállításra a HTTP kérésben található Authorization header értékének beállítása.

4.2 Apache Spark adatfeldolgozó modul

A rendszerbe beérkező nyers adatokat egy, az Apache Spark keretrendszer segítségével implementált adatfeldolgozó modul dolgozza fel. Az adatfeldolgozó feladata, hogy a beérkező mérési adatokból kinyerje azokat a metainformációkat, amelyek a mérés konfigurációját jellemzik, kiszámolja a méréshez tartozó aggregált értékeket, valamint mozgó mérések esetén meghatározza a mérési pontokat befoglaló minimális poligont. Továbbá, az egyes mérési pontokat átalakítja olyan formátumba, hogy azok később egyszerűen betölthetők legyenek a korábban részletezett Apache HBase adatbázisba.

Az Apache Spark egy nyílt forráskódú elosztott adatfeldolgozást segítő keretrendszer, amely az Apache Software Foundation által van karbantartva. A keretrendszer nagy mértékben megkönnyíti az elosztott adatfeldolgozó algoritmusok implementálását, a segítségével a számításokat végző clustert egy egységes interfészen keresztül programozhatjuk a Scala, Java, Python és R nyelvek valamelyikében. Az elektroszmog projektben szükséges adatfeldolgozó modul Java nyelven került implementálásra.

Az Apache Spark nagy mértékben integrálásra került a főbb Hadoop disztribúciókba, így minimális konfigurációt igényel a használata, az adatfeldolgozó algoritmusok futtatása.

Az általunk fejlesztett algoritmus bemeneteként a nyers adatállomány elérési útját, valamint a kimeneti fájlok kívánt elérési újtát várja parancssori argumentumként:

```
usage: gnu
 -if,--input-file <arg>              Input file to process
 -l,--local                          Runs on local environment; if
                                     present, metadata output will be
                                     saved to local file system instead of
                                     HDFS
 -mop,--metadata-output-path <arg>   Destination directory for measurement
                                     metadata
 -op,--output-path <arg>             Destination directory for processed
                                     measurement data
```

Az adatfeldolgozó program parancssori argumentumai

Az elosztott program kimenete JSON fájlok formájában jelenik meg. A két kimeneti állomány egyike a metaadatokat tároló adatbázisba mentendő, míg a másik a mérési adatokat tartalmazó HBase adatbázisba mentendő információkat tartalmazza.

Az algoritmus a futása során elsőként egy universally unique identifier (UUID) stringet generál a mérés számára, amely segítségével a későbbiekben egyértelműen azonosítható lesz. Ezek után megkezdődik a nyers adatfájl beolvasása a HDFS fájlrendszerről. A beolvasott adatokat egy JavaRDD objektum

reprezentálja, amelyen keresztül a szükséges adattranszformációk leírhatók. Az egyes transzformációs lépések minden JavaRDD-t egy újabb JavaRDD példányba visznek át, így végeredményben a transzformációs lépések ábrázolhatók egy irányított, körmentes gráf formájában.

A beolvasás során az adathalmaz a bemeneti CSV fájl sorai szerint kerül felbontásra, így a későbbiekben az egyes sorokat elosztott módon, jól párhuzamosítva dolgozhatjuk fel. Az első transzformációs lépés során ezeket a sorokat értelmezzük és alakítjuk át egy-egy Java objektummá. Ekkor a mérés minden mérési pontját egy Java objektum jelképezi. Ezen feldolgozási lépés segít az adatok további, objektum-orientált szemléletnek megfelelő kezelésében.

A mérési pontokat reprezentáló objektumokon definiált `getJSONs()` metódus segítségével olyan JSON formátumra alakíthatjuk, amelyet a PutHBaseJSON NiFi processor könnyedén el tud majd tárolni az HBase adatbázisban, így ezt a reprezentációt ekkor már el is menthetjük az HDFS fájlrendszerre.

Az mérési adatokon azonban további számítások elvégzése szükséges, ki kell számolni az eredő térerősséget, a legmagasabb térerősség értékkel rendelkező frekvenciát, valamint mozgó mérések esetén a mérési pontokat minimálisan befoglaló poligont is. A következő lépésként tehát ezek történnek meg. A minimális befoglaló poligon meghatározása a Graham scan algoritmus Spark keretrendszerre portolt változatával történik meg, O(n log n) időkomplexitással, így a feldolgozás jól skálázható marad.

A méréshez tartozó metaadatok előállítása során azokat is JSON formátumban exportálja a program, a parancssori argumentumként megadott elérési útra.

4.3 Backend modul

A projekt második fázisában elkészített adatmegjelenítő modul egy single page webalkalmazás, amely a számára szükséges adatokat egy HTTP REST interfészen keresztül kérdezi le. Ennek az interfésznek az implementálása egy Java nyelven írt backend modul formájában történt meg, amelyhez az iparágban elterjedten használt Spring keretrendszer került felhasználásra.

A modul feladata, hogy a megjelenítő alkalmazástól érkező HTTP kérések hatására a szükséges információkat felolvassa a rendszerben lévő adatbázisokból, azokat JSON formátumba szerializálja, majd átadja a kliens számára. Továbbá, a mérési adatokat csak belépett felhasználók érhetik el, ezért fontos, hogy az alkalmazás gondoskodjon az authentikációról és authorizációról is.

Az alkalmazás a fő funkciói mellett monitoring információkat is szolgáltat magáról, így lehetővé téve, hogy az alkalmazás üzemeltetése során folyamatosan elérhetők legyenek a legfontosabb teljesítmény paraméterek.

A Spring keretrendszer számos modulja közül az Elektroszmog alkalmazás fejlesztése során a következők felhasználásra:

- **Core:** A keretrendszer legalacsonyabb szintű szolgáltatásai, melyre minden más alprojekt és alkalmazás épít. Legfontosabb feladata, hogy a függőséginjektálás végrehajtása, az objektumok életciklusának kezelése, illetve a konfigurációs paraméterek felolvasása.

- **Boot:** Spring alkalmazások fejlesztéséhez nyújt segítséget, megvalósítja az automatikus konfigurációt, a komponens szkennelést, valamint a monitoring paraméterek kiajánlását.
- **Data (JPA):** A legtöbb alkalmazásban szükség van adatok perzisztens tárolására, ehhez nyújt segítséget a Spring Data modul. A Spring Data JPA az objektum-relációs leképzés megvalósításával és a Repository minta használatával könnyíti meg a relációs adatbázisok használatát.
- **Security:** A keretrendszer egyik legösszetettebb modulja, amely egy alkalmazásban felmerülő legtöbb authentikációs és authorizációs problémára megoldást ad. Számos, ezen kérdésekkel kapcsolatos technológiához beépített támogatást nyújt (például: LDAP, OAuth2, Basic Auth), de könnyen ki is bővíthető bármilyen egyedi megoldás használatát lehetővé téve.
- **Web MVC:** A Java Servlet technológiára épített modul, amely megkönnyíti a Spring alapú webalkalmazások fejlesztését. Alapját a DispatcherServlet objektum adja, amely várakozik a bejövő HTTP kérésekre, majd azokat a kérés paramétereinek megfelelő Controller objektumhoz továbbítja, amelyben a kéréshez tartozó alkalmazáslogika implementációja található. A DispatcherServlet futtatja továbbá a definiált elő- és utófeldolgozó logikákat, mint például az előállított válasz sorosítása.
- **Hadoop:** A Spring Hadoop modul célja, hogy megkönnyítse a Spring alkalmazások kommunikációját más, Hadoop alapú szolgáltatásokkal, mint például a HDFS vagy az HBase. A projekt során elkészült alkalmazás ezen projekt segítségével tud adatokat lekérdezni az HBase adatbázisból.

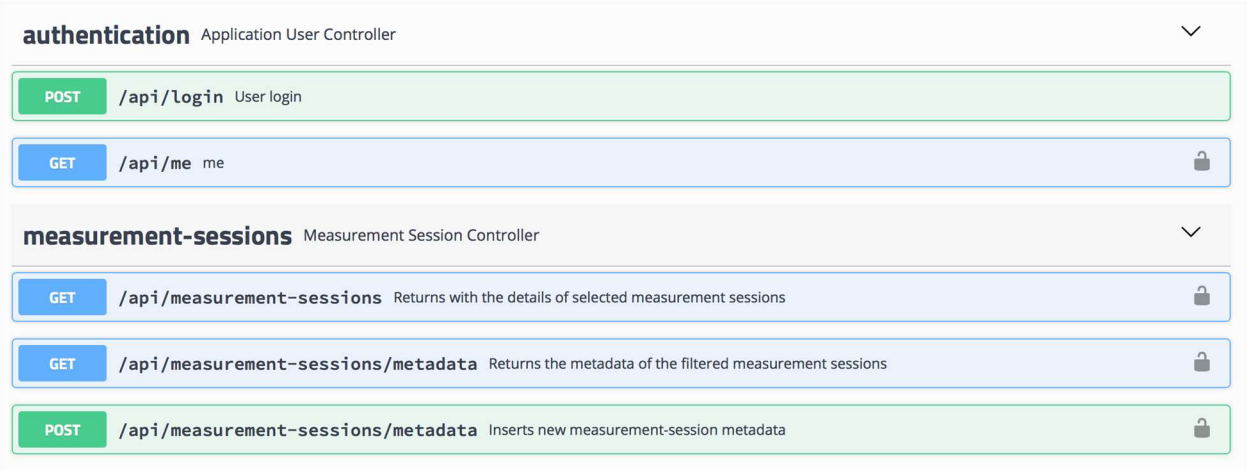
Az alkalmazás egyszerű három rétegű architektúrát követ, ahol a következő rétegek kerülnek megkülönböztetésre:

- **Web:** A HTTP kérések feldolgozásáért felelős réteg, feladata a beérkező kérések validálása, a felhasználói jogosultságok ellenőrzése, majd azok továbbítása a Service réteg felé
- **Service:** Az üzleti logika megvalósításának helye
- **DAL:** Adatelérési réteg, feladata, hogy a Service rétegben használt objektumok számára szükséges adatokat az alkalmazás által elért adatbázisokból lekérdezze, azokat objektumokra leképezze.

Az alkalmazás az HBaseben tárolt adatokat az adatbázis Java interfészén keresztül tudja lekérdezni, majd azokat a RowMapper interface megvalósításával objektumokká leképezni.

Fontos, hogy a backend modul a Hadoop disztribúció részeként elérhető ZooKeeper szerverrel kommunikálva szerzi meg az HBase nodeok elérési útjait, ezzel is biztosítva a szolgáltatás magas rendelkezésre állását.

Az alkalmazás által kiajánlott REST végpontok dokumentációja a Swagger API leíró segítségével történt meg, amelyet az alkalmazás a /swagger-ui.html URI alatt oszt meg. A dokumentáció automatikusan frissül a Web réteg Controllereinek megváltoztatásának hatására, így annak aktualizálására ilyenkor nincsen szükség.



Ábra 6 Az alkalmazáshoz generált Swagger dokumentáció

5 Üzemeltetési leírás

5.1 Rendszerkövetelmények

5.1.1 Hardver

A rendszer hardverigényeit elsősorban a Hadoop platform alkalmazása befolyásolja. Ez a rendszer számos Java folyamatot futtat, elosztottan működik, inicializálást követően is jelentős erőforrásigényei vannak. Ezek terheléssel természetesen növekednek. Azt, hogy mennyivel már az adatok mennyisége, a műveletek számításigénye határozza meg. A projekt keretében felmérésünk alapján a megfelelő hosszútávú működéshez az alábbi hardverkonfiguráció szükséges.

Minimum 4 fizikai szervergép az alábbi konfigurációval:

- 36 mag
- 96 GB ram
- 2 TB HDD

A szervergépek között korlátozások nélküli gigabites hálózati kapcsolatnak kell lennie. A telepítés során internetelérésre van szükség.

5.1.2 Szoftver

A projekt telepítése Linux alapú operációs rendszert igényel, egész pontosan CentOS 7 -et, a klaszter összes gépén. A gépeknek ezen kívül FQDN-el kell rendelkezniük. Ez azt jelenti, hogy minden gépnek egy teljesen kvalifikált domain neve van (pl.: worker1.elektroszmog.hu) és a többi gép ez alapján eléri azt. Ez megvalósítható DNS szerver beállításával vagy a gépek hosts fájljainak megfelelő kitöltésével is.

5.2 Szoftverkomponensek telepítése

A Hadoop cluster telepítése után az Elektroszmog projekthez szükséges szoftverkomponensek telepítése egyszerűen megoldható, ehhez nyújt segítséget a forráskódokkal együtt mellékelt infrastructure repository. A következőkben ezen telepítési folyamat kerül bemutatásra.

5.2.1 HBase adatbázis inicializálása

A telepítési folyamat első lépéseként az HBase adatbázisban szükséges a megfelelő adattáblák és névterek létrehozása. Ehhez segítségként az infrastructure repository tartalmaz egy egyszerű bash scriptet, amely automatizáltan megteszi ezt.

A script bemeneti paraméterként az HBase telepítés bin könyvtárát, a létrehozandó namespace, tábla és oszlopcsaládok nevét várja, a következő formában:

```
./hbase_init.sh /home/username/hbase-2.0.0/bin test_namespace test_table cf1 cf2
```

Fontos figyelni arra, hogy a script eldob minden névteret és táblát, amelynek neve egyezik az általunk megadottakkal, így amennyiben a script nem megfelelően paraméterezett, a futtatása adatvesztéshez is vezethet!

A script sikeres futtatása után a szükséges névtér, tábla és oszlopcsaládok rendelkezésre állnak.

5.2.2 Backend modul telepítése és elindítása

A backend modul telepítése és futtatása együtt történik meg a PostgreSQL adatbázissal a telepítési folyamat megkönnyítésének érdekében. Mindkét komponens a master nodeon fut, egymástól független docker containerekben, azonos docker networkbe szervezve.

A telepítést megkönnyítendő, az infrastructure repository tartalmaz egy docker-compose.yaml fájlt, amely a kívánt környezetet írja le. Ezen állományon belül szükséges beállítani a komponensek konfigurációs paramétereit.

A backend szolgáltatáshoz tartozó konfiguráció:

- **DB_USERNAME:** A PostgreSQL adatbázis eléréshez szükséges felhasználónév
- **DB_PASSWORD:** A PostgreSQL adatbázis eléréséhez szükséges jelszó
- **DB_NAME:** A PostgreSQL adatbázis, amelyet a backend használhat
- **DB_HOST:** A PostgreSQL adatbázist futtató host elérési útja
- **DB_PORT:** A PostgreSQL adatbázis által használt port száma
- **HIBERNATE_CONFIG:** Az alkalmazásban használt ORM megoldás, a Hibernate adatbázis-inicializáló módja
- **JWT_SECRET:** Az authentikációs tokenek aláírásához használt titkos kulcs
- **JWT_EXPIRATION_TIME:** Az authentikációs tokenek érvényességi ideje másodpercben
- **HBASE_ZK_HOST:** A ZooKeeper hostok elérései vesszőkkel elválasztva, ahonnan az HBase kiszolgálók címei lekérhetők
- **HBASE_ZK_PORT:** A ZooKeeper hostok által használt port
- **HBASE_MEASUREMENTS_TABLE:** A mérési értékeket tartalmazó HBase tábla neve namespace:table formátumban
- **HBASE_METADATA_CF_NAME:** A mérési pontokhoz tartozó metaadatokat tartalmazó oszlopcsalád neve
- **HBASE_FREQUENCIES_CF_NAME:** A mérési frekvenciákat és értékeket tartalmazó oszlopcsalád neve

A PostgreSQL adatbázishoz tartozó konfigurációs paraméterek:

- **POSTGRES_USER:** Az induláskor létrehozott felhasználó neve
- **POSTGRES_PASSWORD:** Az induláskor létrehozott felhasználó jelszava
- **POSTGRES_DB:** Az induláskor létrehozott adatbázis neve

A konfigurációs paraméterek beállítása után az alkalmazás a következő paranccsal indítható.

```
docker-compose up -d
```

Az indítás sikerességét a következő paranccsal ellenőrizhetjük.

```
docker ps
```

5.2.3 Spark előfeldolgozó modul telepítése

A Spark előfeldolgozó modul telepítése rendkívül egyszerű, mindössze egy .jar állomány előállítására van szükség a dataprocessor repositoryban található programból:

```
./gradlew jar
```

Az így előállított .jar állományt ezek után egyszerűen fel kell másolnunk az Ambari UI Files View nézetének segítségével a HDFS-re.

Az előfeldolgozót a spark-submit program segítségével indíthatjuk el:

```
export SPARK_MAJOR_VERSION=2
spark-submit \
  --class hu.esmog.DataProcessor \
  --master yarn \
  --deploy-mode cluster \
  hdfs:///user/admin/dataprocessor-1.0-SNAPSHOT.jar \
    -input-file hdfs:///user/elektrosmog/raw_data/$1 \
    -output-path hdfs:///user/elektrosmog/processed_data \
    -metadata-output-path /user/elektrosmog/processed_metadata
```

5.2.4 NiFi adatfolyamok inicializálása

A NiFi adatfolyamok beállítása az infrastructure repositoryban megtalálható XML formátumú template állományok feltöltésével történik meg. A templateket importálnunk kell a NiFi adatfolyamra, majd az összes processort kiválasztva a bal oldali menün megjelenő start gombbal indíthatók el.

5.3 Szoftverkomponensek állapotfigyelése

A rendszer monitorozó komponenseket két részre bonthatjuk, az egyik a Hadoop klaszter felügyeletét látja el, míg a másik a backend alkalmazásból szolgáltat üzemeltetési információkat.

5.3.1 Hortonworks HDP & HDF

A Hadoop klaszterhez tartozó webes menedzsment felület az Ambari, lehetőséget ad a Hadoop komponensek monitorozására. Itt szolgáltatásonként megtekinthetjük a legfontosabb metrikákat a

- HDFS
- HBase
- Yarn
- Spark
- NiFi

erőforrás használatáról. Az Ambari a beállításoknak megfelelően riasztásokat is generál, amiket a felületein vagy emailben hoz tudtunkra. További metrikákat monitorozhatunk az Ambarihoz előkonfigurált Grafana dashboardokon. Ezek a felületek nem csak a Hadoop komponensek, de az azokat futtató hoszt gépek metrikáit is figyelik, mint a CPU, memória vagy tárhely használat.

5.3.2 Backend

Az alkalmazás szerver Spring actuator alapú végpontokat kínál az alkalmazásmetrikák lekérésére. Ezek a végpontok egyszerűen beköthetők egy Grafana vagy Nagios rendszerbe, ahol megvalósul a valós idejű monitorozás és felügyelet.