ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS
INFORMATIKAI KAR

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Szabó Márk**
2021 Hallgató törzskönyvi száma: **T/006586/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

SZAKDOLGOZAT
FELADATLAP

Hallgató neve: **Szabó Márk**
Törzskönyvi száma: T/006586/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

COVID-19 statisztikák elemzése és feldolgozása felhőalapú technológiák segítségével nagyvállalati döntéstámogatáshoz

Analyzing and Processing COVID-19 statistics using cloud computing technologies to aid enterprise decision-making

Intézményi konzulens: Dr. Fleiner Rita
Külső konzulens: Retezi Richárd, Cloud Solution Architect, Microsoft

Beadási határidő: 2021. december 15.

A záróvizsga tárgyai: Számítógép architektúrák
Big Data és üzleti intelligencia
specializáció

A FELADAT

A feladat a COVID-19 világjárvány, illetve annak statisztikáinak és kulcs indikátorainak megismerése, hatásainak elemzése nagyvállalatokra fókuszálva. Továbbá historikus és frissülő adatok feldolgozása, analizálása és vizualizációja felhőalapú technológiákkal, mint például magasan skálázható serverless megoldások, felhőalapú ETL folyamatok, gépi tanulási eszközök és felhős adattárházak. A feladat központi eleme egy nagyvállalati döntéstámogató Proof of Concept (PoC) megoldás architektúrájának megtervezése és lefejlesztése.

A dolgozatnak tartalmaznia kell:

- a COVID-19 világjárvány és hatásainak rövid összefoglalását,
- kapcsolódó nagyvállalati problémák ismertetését,
- felhasznált komponensek és szolgáltatások leírását és ezek opcionális alternatíváit,
- PoC megoldás architektúrális tervét és a megvalósítás folyamatát,
- COVID-19 statisztikák analizálásának eredményeit,
- és továbbfejlesztési lehetőségeket.

ÓBUDAI EGYETEM Neumann János Informatikai Kar 1. Budapest Ph.

Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2023. december 15.**

(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

külső konzulens

Fleiner Rita

intézményi konzulens

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2021. 12. 10.

.....

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun kód:	Tagozat:
Szabó Márk	[redacted]	Mérnökinformatikus BSc

Telefon:	Levelezési cím (pl: lakcím):
[redacted]	[redacted]

Szakdolgozat címe magyarul:

COVID-19 statisztikák elemzése és feldolgozása felhőalapú technológiák segítségével nagyvállalati döntéstámogatáshoz

Szakdolgozat címe angolul:

Analyzing and Processing COVID-19 statistics using cloud computing technologies to aid enterprise decision-making

Intézményi konzulens:	Külső konzulens:
Dr. Fleiner Rita	Retezi Richárd Cloud Solution Architect, Microsoft

Alk.	Dátum	Tartalom	Aláírás
1.	**2021. 02. 26.**	**Szakdolgozat témájának megbeszélése**	Fleiner Rita
2.	**2021. 03. 05.**	**Feladatlap megbeszélése**	Fleiner Rita
3.	**2021. 04. 29.**	**Irodalomkutatás megbeszélése**	Fleiner Rita
4.	**2021. 05. 06.**	**Dokumentumok beadásának megbeszélése**	Fleiner Rita

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat I. (BSc) tantárgy követelményét teljesítette, beszámolóra bocsátható.

Budapest, 2021. május 10.

Fleiner Rita
intézményi konzulens

KONZULTÁCIÓS NAPLÓ

Hallgató neve:	Neptun kód:	Tagozat:
Szabó Márk	[redacted]	Mérnökinformatikus BSc

Telefon:	Levelezési cím (pl: lakcím):
[redacted]	[redacted]

Szakdolgozat címe magyarul:

COVID-19 statisztikák elemzése és feldolgozása felhőalapú technológiák segítségével nagyvállalati döntéstámogatáshoz

Szakdolgozat címe angolul:

Analyzing and Processing COVID-19 statistics using cloud computing technologies to aid enterprise decision-making

Intézményi konzulens:	Külső konzulens:
Dr. Fleiner Rita	Retezi Richárd Cloud Solution Architect, Microsoft

Alk.	Dátum	Tartalom	Aláírás
1.	**2021. 09. 20.**	**Irodalomkutatás véglegesítése**	Fleiner Rita
2.	**2021. 10. 04.**	**Követelmény specifikáció, rendszerterv**	Fleiner Rita
3.	**2021. 11. 29.**	**Kódolás, eredmények összegzése**	Fleiner Rita
4.	**2021. 12. 06.**	**Dokumentumok beadásának megbeszélése**	Fleiner Rita

A Konzultációs naplót összesen 4 alkalommal, az egyes konzultációk alkalmával kell láttamoztatni bármelyik konzulenssel.

A hallgató a Szakdolgozat II. (BSc) tantárgy követelményét teljesítette, védésre bocsátható.

Budapest, 2021. december 10.

Fleiner Rita
intézményi konzulens

TARTALOMJEGYZÉK

1. BEVEZETÉS

A COVID-19 világjárvány és hatásainak végiggyűrűzése a gazdaságon komoly kihívások elé állította a kis és nagyvállalatokat szerte a világon. A mindennapi üzemeléssel kapcsolatos döntések is komoly, nagy mennyiségű, folyamatosan változó adaton alapuló statisztikai döntésekké alakultak egyik pillanatról a másikra. Dolgozatomban kifejezetten a nagyvállalatok problémáira fókuszáltam és azt jártam körül, milyen döntéstámogató rendszert lehet kialakítani, amivel megkönnyíthetőek ezek a mindennapi döntések.

Ehhez modern, felhőalapú technológiákat használtam, hogy a rendelkezésre álló adatokat analizáljam, feldolgozzam és vizualizáljam, illetve olyan megoldásokat hozzak létre, amik ezen nagyvállalatok döntéshozóit és munkavállalóit a kockázatértékelésben segítik.

Az adatnak komoly szerepe van a járványkezelésben, mint ahogy azt a későbbiekben tárgyalni fogom, jelentős költségek és potenciális bevételkiesés is származhat a vállalati járványkezelésből. Ha a megfelelő döntéshozók tényleg akkor és olyan döntéseket tudnak meghozni, amik a legoptimálisabbak, azzal limitálni lehet a felesleges kiadásokat és maximalizálni a profitot.

Ilyen döntés lehet például az irodák nyitása vagy zárása, a befogadóképesség limitációja vagy a személyes partner- és ügyféltalálkozók engedélyezése vagy tiltása.

Ezen döntések meghozatalában fontos szerepet kell játsszon a rendelkezésre álló valós idejű adatmennyiség a járvány aktuális lokális és globális állapotáról, trendekről, terjedéséről. Ezen adatok többé-kevésbé rendelkezésre is állnak – erről részletesebben az Adatforrások fejezetben értekezem.

Dolgozatom részeként egy olyan COVID-19 statisztikák elemzésére és feldolgozására alkalmas rendszert készítettem, mely a publikusan elérhető és frissülő releváns adatforrásokból szerzi be az adatokat, ezeket feldolgozza és megfelelő formátumra hozza, hogy utána analitikákat lehessen készíteni belőle, vagy akár automatikusan frissülő adatvizualizációs dashboard-okon jelenhessenek meg.

Mindezt a folyamat legelejétől a legvégéig automatizáltan és lokális (on-premises) infrastruktúra nélkül, kizárólag felhős technológiákból alakítottam ki a karbantartás és a költségek minimalizálása érdekében.

Az adatbetöltő rendszer szerepét egy menedzselt felhőalapú ETL megoldás látja el. Az adatok a betöltés után tisztításra kerülnek és megfelelő formátumra lesznek konvertálva, majd egy felhős skálázható adatbázisba töltődnek be.

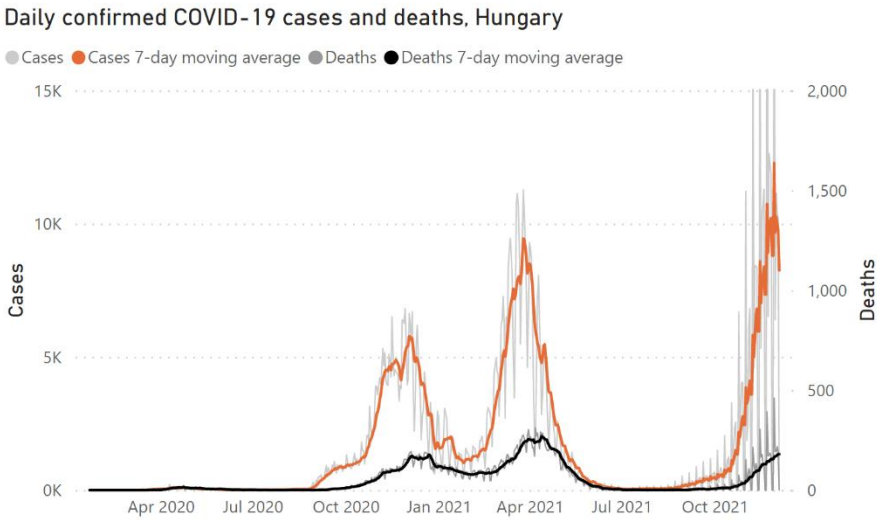
Innen kerülnek kiolvasásra az adatvizualizációhoz a riportok által, illetve Machine Learning modellek betanításához az erre szolgáló megoldás által.

A nagyvállalati felsővezetők és döntéshozók ezekből a könnyen feldolgozható riportokból és dashboard-okból hozhatják meg az adatokra alapozott döntéseiket.

2. A COVID-19 VILÁGJÁRVÁNY ÉS HATÁSAI

2019 decemberében diagnosztizálták az első SARS-CoV-2 vírus által fertőzöttet Kína Wuhan nevű tartományában. Magas reprodukciós rátája és a rendkívül mobilis globalizált világ segítségével a vírus néhány hónap leforgása alatt az egész világon elterjedt 2020 első felében.

Különböző országok különböző módszerekkel próbálkoztak védekezni és kordában tartani a járványt. Magyarországot az első hullám 2020 első és második negyedévében az átlaghoz képest kevéssé érintette, a napi új fertőzöttek száma az alacsonynak minősülő 3 számjegyű kategóriában maradt, míg a napi elhunytak száma nem lépte át a 20 főt.



1. ábra: Az első, második, harmadik és negyedik COVID-19 hullámok napi fertőzési és halálozási számai Magyarországon.

A nyári nyitás és a külföldi utazások korlátozásának hiánya – illetve a nyár folyamán azok támogatása – miatt a második és harmadik hullámokhoz képest az első elhanyagolható méreteket öltött. A fertőzöttek száma két nagyságrenddel haladta meg az első hullámot és a napi elhunytak száma olyan magas volt, hogy bizonyos statisztikák szerint Magyarországon halt meg a legtöbb ember népességarányosan a teljes járványra vetítve.

Hatásos gyógyszer és vakcina hiányában az egyetlen védekezési eszköz a 'social distancing', azaz közösségi távolságtartás lett, amihez a kis és nagyvállalatoknak is egyaránt alkalmazkodniuk kellett. A nagyvállalatok esetében specifikusan szinte univerzális home office, azaz otthoni munkavégzés került bevezetésre, ami számtalan új kihívás elé állította ezeket a piaci szereplőket.

A járvány közvetlen egészségre veszélyes velejárói mellett jelentős másodlagos hatásként jelent meg a kialakult gazdasági válság. 2020 márciusában történelmi mélységekbe zuhant a globális tőzsde a jövőre vonatkozó teljes bizonytalanság eredményeképp és

ugyan ezt a korábbi válságokhoz (mint például 2008) képest gyors visszapattanás követte, ez nem volt elmondható a tényleges gazdaságról.

A kormányok sosem látott fiskális serkentőkkel próbálták élénkíteni a gazdaságot, de ez rendkívül keveset segített azokon az iparágakon, melyek teljes forgalmukat veszítették el. Ilyen például a turisztika, vendéglátás, de például a fogyasztói légiközlekedés is. [1]

2021 elejére elérhetővé váltak korunk leggyorsabban kifejlesztett vakcinái, amelyek jelentős része élesben sosem próbált technológiát használ. A korábbi kihívások mellett megjelent egy teljesen új feladat: az oltások disztribúciója és a fokozatos nyitás megtervezése.

2.1. **Nagyvállalati kihívások**

A nagyvállalatok nevükből is eredően sok embert foglalkoztatnak, akik tradicionálisan nap mint nap bejárnak a munkahelyükre dolgozni. Ez érthető okokból nem egy olyan helyzet, amely a járvány elleni védekezést könnyen lehetővé teszi.

A vállalati vezetőknek a saját alkalmazottjaik és ügyfeleik biztonsága és a pénzügyi fennmaradás között kell navigálniuk, mindezt egy nagyon gyorsan változó környezetben, a napi operációk újratervezésével és komplikált állami támogató programok mellett kell megtenniük.

A vállalatok jól felfogott érdekei mellett egy társadalmi elvárás is megjelent, hogy ezen piaci szereplők is aktívan részt vegyenek a védekezésben. A munkahelyi járványkezelés több fronton valósul meg:

- A védekezés első vonala az ún. PPE, azaz egyéni védőfelszerelés. Kötelező maszkviselés zárt térben, kéz- és felületfertőtlenítés.
- Zárt terekben a távolságtartás, illetve kapacitás limitáció (minden második asztalhoz lehet csak ülni) adhattak potenciális megoldást.
- Ezeknél sokkal biztosabb eredményt ad a fizikai kapcsolatok tényleges csökkentése, mely az alkalmazottak részleges vagy teljes távoli munkavégzésbe való áthelyezése. Ez történhet rotációval (egyik héten a csapat egyik fele dolgozik az irodából, a következő héten a másik), illetve kritikusság alapján is (bolti eladó nem tud otthonról dolgozni, a HR-es viszont igen), illetve, ha a cég profilja ezt megengedi, a teljes állományra be lehet vezetni az otthoni munkavégzést párhuzamosan az iroda ideiglenes bezárásával.
- A határzárak közvetlen következményeként az üzleti utak szinte minden iparágban teljesen meg lettek szüntetve és le lettek mondva.
- Végül – ha a fent említettek nem elegek és feltétlen szükséges a fizikai érintkezés – néhány vállalat kötelező tesztelést vezetett be, amely vagy egy-egy esemény biztonságát szolgálták (stúdióban videó felvételhez mindenkit tesztelnek) vagy rendszeresen minden alkalmazottat leteszteltek.

Ezen intézkedéseknek viszont ára van és itt nem a tényleges költségek az elsődlegesek, sokkal inkább a hatékonyság csökkenéséből fakadó bevételkiesés, ami hozzáadódik a járvány miatti állami intézkedésekhez (kijárási- és gyülekezési tilalom, boltok bezárása). Ebből kifolyólag a vezetőknek szükségük van olyan rendszerekre, amelyek segítik meghozni a szükséges döntéseket, de csak azokat, amik feltétlen szükségesek.

2.2. **Üzleti igények**

A tárgyalt nagyvállalatok vezetői és döntéshozói több ezer fős szervezetek felett hoznak döntéseket, melyek közül a legkisebbek hatásai is dollár százezrekben mérhető nagyságrendűek.

A legfontosabb ilyen döntések, amelyeknek meghozatalában támogatásra van szükségük az üzleti teljesítményt potenciálisan negatívan érintő korlátozások. A teljesség igénye nélkül:

- Mikor feltétlen szükséges az ügyfélközpontú, személyes találkozókra épülő munkaerő kötelező otthoni munkavégzésre rendelése, mert túl nagy a veszélye egy lokális járvány gócpont kialakulásának vállalaton belül? Egy ilyen lokális gócpont kialakulása számos negatív következménnyel jár, köztük a munkaerő kiesésével, egészségük potenciális hosszútávú károsodásával, ügyfelek megfertőzésével, és különösen nagyvállalatok esetében negatív sajtóvisszhanggal.
- Mekkora az esélye, hogy a szervezetben találhatóak fertőzöttek, mennyi az ő várható számuk?
- Érdemes-e kor, illetve klinikai veszélyeztetettség alapján különböző módon kezelni az alkalmazottakat, mekkora az esélye egy-egy konkrét munkavállalóra nézve a különböző fertőzési kimeneteknek?
- Mikor és milyen korlátozások meghozatalával tudjuk biztosítani a kritikus infrastruktúra működését minden esetben? A járványhelyzet egy adott pontján a budapesti tömegközlekedés működőképessége is kérdésessé vált, azon egyszerű tényből kifolyólag, hogy olyan sok sofőr volt beteg, hogy például a metrók működése közel biztosíthatatlanná vált. Egy ilyen esetben az egész városra kiterjedő korlátozások jelenthettek csak elégséges lépést.
- Hogyan segíthetjük a munkavállalókat a személyes kockázatelemzésben?
- Milyen szintű átoltottság esetén lehet lazításokat eszközölni a szervezeten belül?

3. MEGKÖZELÍTÉSI MÓDOK ÉS MEGOLDÁSI LEHETŐSÉGEK ÁTTEKINTÉSE

3.1. Proof of Concept

A Proof of Concept (PoC) – magyarra fordítva a koncepció bizonyítása – egy olyan projekt típus, melyet egy új rendszer bevezetése előtt, az elképzelések validálása érdekében készítünk el. Egy tényleges, éles környezetben használt rendszer teljes funkcionalitásával nem kell rendelkeznie, elég csak a kritikus, nem biztosan, vagy nehezen megvalósítható komponensekből példát mutatni. Egy egyszerű példával élve, egy italautomata PoC-je lehet egy olyan IKEA-s furnérlemezből készült szekrénybe épített szerkezet is, amely csak egyetlen fajta üdítőt tud kiadni, nincs üvegajtaja, hűteni sem tud. Ezeket a feladatokat tudjuk, hogy el tudjuk végezni, üvegajtós hűtőszekrény létezik már, a szerkezetet csak be kell majd szerelnünk. Az elektronika, a fizetési és számlázási rendszer, ezek integrációja viszont egy olyan feladat, amit érdemes a PoC keretein belül elvégezni, mert ha nem tudunk olyan mechanikát készíteni, ami kiadja az üdítőt, akkor nem érdemes folytatnunk az italautomata fejlesztését.

Egy PoC eredménye, vagy részeredménye lehet az is, hogy az adott feladat a specifikált módon nem valósítható meg, és ezen eredmény is teljesen elfogadandó. Az előbbi példánál maradva, mondhatjuk például a PoC végén, hogy a készpénz felismerés nem megfelelően stabil, nem látunk rá könnyen megvalósítható megoldást, így az a javaslat, hogy az automatánál csak bankkártyával lehessen fizetni.

A Proof of Concept-et annak sikeressége esetén egy pilot projekt követi, ahol szoros megfigyelés alatt, de már élesben vagy közel-élesben használják. Szintén az automatás példával élve, az automata fel van töltve italokkal, pénzt is elfogad, de még csak a fejlesztő cég irodájában van felállítva, ahol nem kell vandalizmusra felkészülve lennie.

A pilot projekt után – az abból merített visszajelzéseket is beépítve a megoldásba – lehet elkezdeni a széleskörű bevezetést, ahol élesben sokan az eredeti célok elérése érdekében használják már a megoldást.

3.2. Az infrastruktúra absztrakciói

3.2.1. Felhőalapú számítástechnika [2]

A felhőalapú számítástechnika alatt olyan igény szerint elérhető, hálózatba kötött számítási kapacitást értünk, amely könnyen és gyorsan hozható létre a szolgáltató manuális interakciója nélkül.

A felhőalapú számítástechnika NIST által megfogalmazott alapvető karakterisztikái:

- Igény szerint, önkiszolgáló módon érhetőek el ez erőforrások, tehát például nem kell egy emailt írnunk az üzemeltetőknek, hogy új erőforrást provizionáljanak számunkra, hanem saját magunk meg tudjuk tenni ezt néhány konfigurációs paraméter megadása után.
- Hálózaton keresztül menedzselhetőek az erőforrások sztenderd protokollokon keresztül.
- Az erőforrásokat a felhő szolgáltató összevontan kezeli, így mindig az tudja igénybe venni őket, akinek aktuálisan szüksége van rá.
- Gyorsan és rugalmasan skálázhatóak az erőforrások mind kifelé, mind befelé a szükségleteknek megfelelően.
- Végezetül az erőforrások használata pontosan mérve van, így ténylegesen csak azután kell fizetnünk amennyit használunk.

3.2.2. **CAPEX és OPEX**

Egy tradicionális infrastruktúra kialakítás során elsősorban eszközbeszerzésről beszélünk. Ez az ún. CAPEX (capital expenditure), azaz tőkebefektetés kategóriába tartozik; még mielőtt használatba tudnánk venni az infrastruktúrát, egy nagyon jelentős költséget el kell könyvelnünk előre. Ezután a rendszer fenntartási költségei ugyan számottevőek (mind humán erőforrás költségben, mind a hardverek tárolása, energiaellátása, hűtése, fizikai biztonsága és esetleges cseréje), de a kezdeti tőkebefektetéshez képest alacsonyabbak.

Ezzel szemben egy felhős infrastruktúra létrehozásakor nem kell számolnunk semmilyen előzetes költséggel, a teljes infrastruktúrát OPEX-ből (operating expenditure), azaz működési költségekből fedezzük. Az erőforrások után csupán annyit kell fizetnünk amennyit ténylegesen el is használtunk.

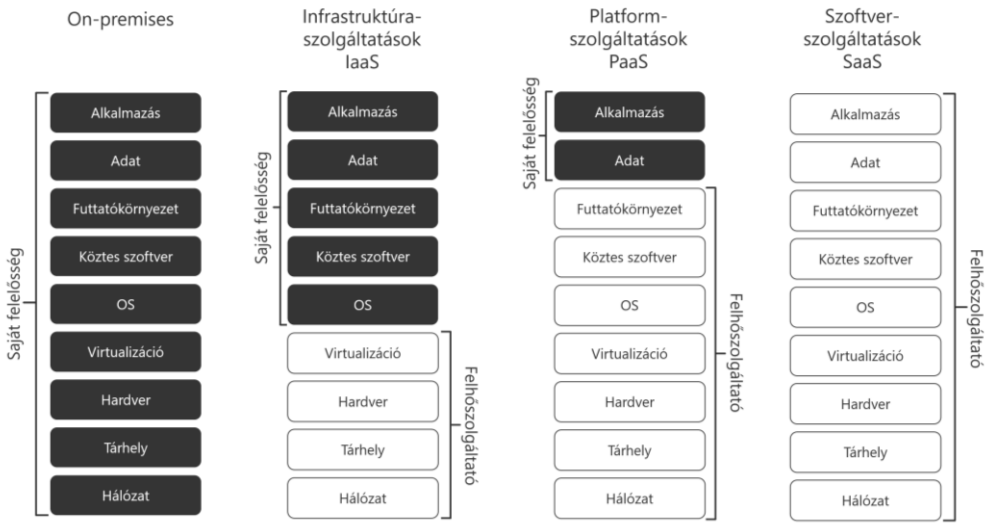
A fentiek alapján elmondható, hogy a felhőalapú infrastruktúra a legtöbb esetben költséghatékonyabb lehet, mint privát adatközpontok építése. Különösen igaz ez, ha a szervezetünk agilisen működik, megoldásokat pivotál és a normál működés része, hogy valami, ami nem felel meg az eredeti elképzeléseknek gyorsan leszerelésre kerül.

3.2.3. **Felhőszolgáltatók**

A három legjelentősebb felhőszolgáltató sorrendben az Amazon Web Services (AWS), a Microsoft Azure és a Google Cloud Platform (GCP). A jelenlegi legnagyobb és a piacon első AWS az infrastruktúra szolgáltatások terén rendelkezik komoly piaci előnnyel. A piac meredek növekedésével és bővülésével újabb szereplők is megjelentek, köztük az Oracle, az IBM vagy a kínai Alibaba.

3.2.4. **Felhős szolgáltatási modellek**

Az on-premises infrastruktúrákkal szemben – ahol természetesen mi felelünk minden részletért – a felhőszolgáltatók jelentős felelősségeket vállalnak át tőlünk, mindezeket pedig SLA-kal (Service Level Agreement, szolgáltatásszint-megállapodás) garantálják.



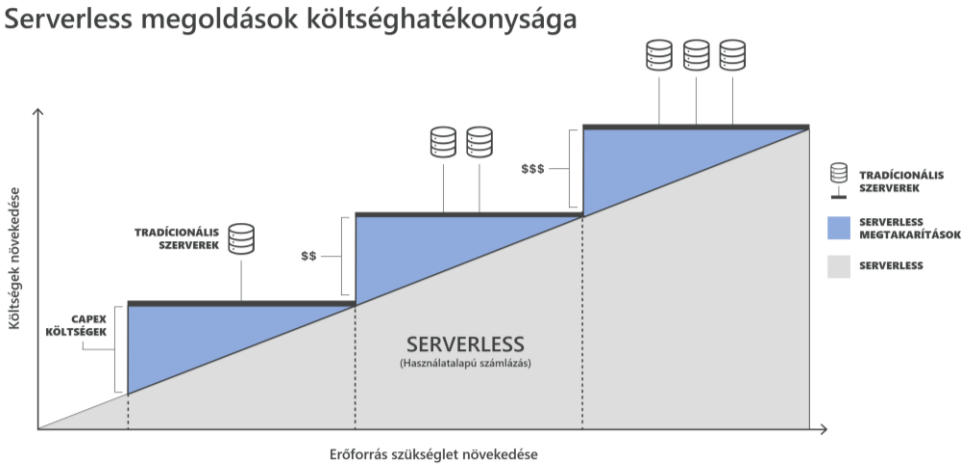
2. ábra: A felelősség megoszlása a felhőszolgáltató és az ügyfél között a különböző szolgáltatási modellek viszonylatában.

A felhő felé vezető út első állomásán az infrastruktúra szolgáltatások (Infrastructure-as-a-Service, IaaS) találhatóak, amiket például egy meglévő földi rendszer egyszerű átmozgatásával (lift&shift) is igénybe tudunk venni. Infrastruktúra szolgáltatásoknak nevezzük a különböző típusú virtuális gépeket, hálózati megoldásokat, tűzfalakat. A szolgáltató átvállalja tőlünk a fizikai adatközpont karbantartásának feladatait. Ő felel azért, hogy legyen energiaellátás, megfelelő hűtés, hálózati kapcsolat, a hardverek ne legyenek hibásak, illetve ha meghibásodnak, leállás nélkül (vagy minimális leállással) megvalósuljon a cseréjük; de például az adatközpont fizikai biztonságért is. Általában fegyveres őrök vigyáznak az adatközpontra, és oda bejutva is több állomásnyi biztonsági ellenőrzésen kell átesnünk, hogy ténylegesen a szerverek közelébe kerülhessünk. A hardver fölött menedzselve kapunk egy virtualizációt is és egy általunk választott operációs rendszert előre telepítve, de ennek frissítéseiért, biztonsági konfigurációiért már mi felelünk.

Egy szinttel több felelősséget vállal át a szolgáltató a platformszolgáltatások (Platform-as-a-Service, PaaS) esetében, ahol az operációs rendszeren át egészen a futtatókörnyezetig mindent a szolgáltató tart karban. Ilyen szolgáltatás például egy menedzselt adatbázis, egy webalkalmazások futtatására szolgáló megoldás beépített skálázódással és load balancer-rel. A nagy felhőszolgáltatók jelenleg a legtöbb erőforrást ezen PaaS megoldások fejlesztésébe, újak létrehozásába fektetik.

Még egy szerveződési szinttel feljebb találjuk a szoftverszolgáltatásokat (Software-as-a-Service, SaaS), ahol már saját kódot sem kell írnunk, az alkalmazás teljesen kész van, az ügyfélnek csak használnia kell azt. Jó példa erre a Microsoft vonatkozásában a Microsoft 365, ahol ugyan van egy admin felület, ahol a rendszergazda be tud konfigurálni különböző beállításokat, a rendszer maga viszont teljesen készen van és a kódjába nem lehet belenyúlni, személyre szabni azt.

3.2.5. **Serverless szolgáltatások**



3. ábra: Költségbecslés a serverless megoldások költséghatékonyságára a tradicionális szerverekkel szemben a Cloudflare által. [3]

A serverless szolgáltatások még mindig szervereken futnak, a különbség viszont a többi platformszolgáltatáshoz képest a hardvertől való totális szétcsatoltságban rejlik. Nem lényeges, hány szervert használunk, hogyan és mikor vannak skálázva, a hálózat vagy éppen a load balancing milyen módon van megoldva – egyedül az számít, hogy lefusson a kódunk. Míg a többi platformszoláltatás esetében valamilyen szinten meg kell határoznunk a rendelkezésre álló számítási kapacitást – ez alapján történik a számlázás is – a serverless megoldások esetében erre nincsen szükség, fizetni pedig a kód futásainak száma, illetve annak ideje alapján fizetünk majd. [4]

3.3. **Felhőszolgáltató választás szempontjai**

A szolgáltató választás legelső lépése magának a szolgáltatónak, mint vállalatnak a vizsgálata kell legyen. A pénzügyi stabilitáson, átlátható működésen és szervezeten, törvényi és különböző harmadik fél szabványainak való megfelelőségen és bizalmon túl rendkívül fontos a vendor üzleti és technológiai szakértelme és hozzáértése.

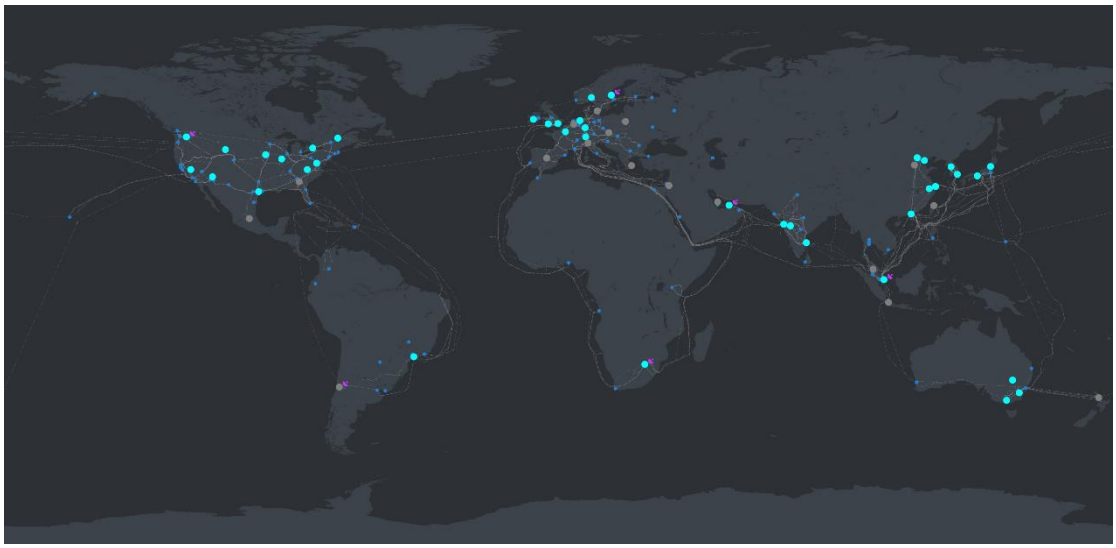
Ezek mellett fontos az adminisztratív támogatás is, a szolgáltatónak képesnek kell lennie SLA-kat biztosítani, a teljesítményt és rendelkezésre állást riportolni, ezeket, illetve az erőforrásokat monitorozni, és ezekről megfelelően alapos számlát készíteni a tényleges használat után.

A platform technikai képességeit vizsgálva azt kell tapasztalnunk, hogy a rendszer sztenderdeket használ, sztenderd interfészeken keresztül kommunikál, az erőforrások létrehozása és menedzsmentje egyszerű és érthet. Ezentúl a platformban bekövetkező változásokra is kialakított folyamatoknak és dokumentációknak kell léteznie.

Végül pedig kardinálisan fontos szempont a szolgáltató által vállalt biztonság, lévén, hogy a szerverek nincsenek többé a kezünkben. Ez magában foglalja az infrastruktúra tervezése általi biztonságát, a proaktív támadások elleni védelmet, különböző hozzáférés menedzsmentet. Emellett ide tartozik az adatok biztonsági mentése, de az adatközpontok fizikai biztonsága is üzleti folytonossági és katasztrófa-helyreállítási tervekkel. [5]

3.4. **Microsoft Azure**

Microsoft Azure, a Microsoft felhő platformja 2010 februárjában vált publikusan elérhetővé. Mind infrastruktúra-, platform- és szoftverszolgáltatásokkal rendelkezik, összességében több mint 200 darabbal, az egyszerű virtuális gépektől a különböző machine learning, blockchain és mixed reality megoldásokon át egészen a quantum számítógépekig.



4. ábra: A Microsoft Azure jelenlegi (kék) és épülő (szürke) adatközpont régiói, hálózati végpontjai (sötétkék) és műholdas földi állomási (lila).

A rendszer implementációja során a Microsoft Azure szolgáltatásait használtam, az egyes elemeket az alábbiakban részletezem. A potenciális alternatívák között az Amazon Web Services (AWS), a Google Cloud Platform (GCP) és az Alibaba Cloud is felmerülhet. Az

AWS-el szemben a Microsoft szervezeten belüli ismertsége, elterjedtsége és megbízhatósága szólt, a GCP esetében az árak, illetve a szolgáltatások fejlettsége döntött, az Alibaba Cloud esetében pedig az adatbiztonság megkérdőjelezhetősége, illetve a Kínai befolyás jelentett kizáró tényezőt.

Jelenleg több mint 60 régióval és több mint 100 fizikai adatközponttal a Microsoft Azure nagyobb hálózattal rendelkezik, mint bármelyik másik felhős infrastruktúra szolgáltató. Ezentúl a folyamatos bővülés jellemzi, a Közép-Kelet Európai régióban például jelenleg is épül egy adatközpont Lengyelországban és Görögországban, tőlünk nyugatabbra pedig Ausztriában és Észak-Olaszországban is. [6]

3.4.1. **Azure Data Factory**

A Microsoft új felhőalapú integrációs szolgáltatása – az Azure Data Factory – célja a régi SSIS (SQL Server Integration Services) package-ek leváltása ADF pipeline-okra. Ezenkívül kibővíti a hagyományos funkcionalitást tömérdek új funkcióval, például a gépi tanulással való zökkenőmentes integráció illetve a serverless megoldások. [7]

A Data Flow egy vizuálisan elkészített adattranszformáció, melynek segítségével kódolás nélkül vagy minimális kódolással tudunk transzformációs logikát készíteni. Az így készült Data Flow-k ADF-ben önmagukban nem indíthatóak, pipeline-okban futnak le, a tényleges egzekúciót pedig Apach Spark klaszterek végzik el. [8]

A Power Query a Microsoft által fejlesztett hatékony adattranszformációs és adatelemző motor, saját fejlesztői nyelvezettel. Egyaránt használható ETL folyamatok megvalósítására és adat előfeldolgozásra is. Több különböző Microsoft termékbe beépítve is megtalálható, mint például Excel, Power BI és Azure Data Factory. [9]

3.4.2. **Azure SQL Database**

A menedzselt Azure SQL Database számos előnnyel rendelkezik a földi párjával szemben, beleértve, de nem kizárólag, a serverless skálázhatóságot, az integrált gépi tanuláson alapuló Advanced Thread Protection elnevezésű biztonsági megoldást és az Active Directory integrációval történő fejlett hozzáférés-vezérlést. [10]

3.4.3. **Azure Synapse Analytics**

Az Azure Synapse Analytics az SQL Data Warehouse utódja, de annál sokkal több képességgel. Kombinálja az adatintegrációs, a nagyvállalati adattárház és big data analitikai feladatokat egyetlen megoldásba. Támogatja az SQL-t és az Apache Spark-ot is, így dolgozhatunk strukturált és nem-strukturált adatokkal is. A gépi tanulással, illetve BI-al való integráció már előre el van készítve nekünk, használatuk egyetlen

gombnyomásba kerül csak. Képes dedikált erőforráson és serverless kapacitáson is dolgozni.

3.4.4. **Azure Cosmos DB**

Az Azure Cosmos DB egy teljesen menedzselt NoSQL adatbázis, amit a Microsoft eredetileg a saját SaaS megoldásai alá tervezett belső használatra, és ebből lett később külön termék. SLA által garantálja az egyszámjegyű válaszidőt, illetve a 99.999% rendelkezésreállást. Globálisan skálázódik és szinkronizál, pontosan definiált konzisztenciaszintek mellett több-régiós párhuzamos írást is támogat.

Különböző API-ok közül választhatunk, így SQL, MongoDB, Cassandra és gráf adatbázis (Gremlin) támogatás is van.

3.4.5. **Azure Storage**

Az Azure Storage szolgáltatásnak 4 részegysége van: az Azure Files, amely egy egyszerű SMB-NFS hálózati fájlmegosztás, az Azure Queue Storage, ami egy meglehetősen költséghatékony queue megoldás, az Azure Table Storage, a Cosmos DB kistestvére igazából, egy olcsó, de nem olyan funkciógazdag NoSQL adatbázis és a számunkra érdekes Blob Storage, ahova a napi adatokat archiváljuk az ETL folyamat során. Ez egy REST API alapú végtelenül skálázható nem-strukturált objektum tároló.

Az Azure Data Lake az Azure Storage egy kifejezetten big data és ETL projektek megvalósításához kifejlesztett verziója. Nagyméretű fájlok tárolása és gyors feldolgozása mellett támogatja a Hadoop Distributed File System-et (HDFS), illetve az arra épülő alkalmazás ökoszisztémát is. Legfőbb célja a hagyományos adatsilók leváltása egy olyan megoldással, mely az adattároláson túl például beépített SQL alapú query motort és klaszteres adatfeldolgozást tesz lehetővé.

3.4.6. **Azure Functions**

A Microsoft Azure serverless megoldása az Azure Functions, aminek segítségével esemény-vezérelt állapot nélküli általában kicsi feladatokat ellátó funkciókat tudunk készíteni, de képes akár komplexebb állapottal rendelkező problémákat is orkesztrálni.

Támogatja a .NET mellett a Node.js, a Java, a Python nyelveket/framework-öket, de még a PowerShell scripteket is. Ezentúl a legújabb fejlesztésekkel már konténereket is lehet futtatni Functions-ben, így tehát igazából szinte bármilyen kóddal elboldogul.

3.4.7. **Azure Machine Learning**

Az Azure Machine Learning a platform központosított gépi tanulás megoldása, ahol az automatizált modell tanítástól, a Machine Learning Studio designer felületén kódolás nélküli betanításon át a Jupyter Nootebook-okban R-ban vagy Python-ban megírt egyedi modellek felhőben betanításáig több szolgáltatás van ötvözve.

Az ETL által betöltött és feldolgozott adatokon tanítom be a gépi tanulási modellünket az Automated Machine Learning segítségével.

3.4.8. **Azure Active Directory**

Mint minden felhő platform, az Azure is rendelkezik egy hozzáférés és identitás menedzsment rendszerrel. Ez a tradicionális földi Active Directory felhős verziója, ahol a Role-based access control (RBAC) segítségével pontosan azoknak és pontosan azokhoz az erőforrásokhoz lesz csak hozzáférésük, akiknek arra ténylegesen szükségük van.

Ezentúl a munkahelyi fiókokkal való single sign-on (SSO) támogatásához is az Azure Active Directory-val kell integrálódni.

Azon vállalatok, akik még mindig használnak földi Active Directory-t sincsenek kizárva az egységes identitás menedzsmentből, mert az Azure AD Connect segítségével többféleképp is lehetőség van szinkronizálni a felhasználói fiókokat a földi rendszer és a felhő között. Ilyenkor választhatunk, hogy felszinkronizáljuk a felhőbe a felhasználók jelszavainak hash-ét (Password hash sync), vagy autentikáció során továbbirányítjuk a felhasználókat minden alkalommal a földi AD szerverhez (pass-through authentication vagy federation).

3.4.9. **Microsoft Power BI**

A Microsoft Power BI szintén egy felhős termék, de mégsem az Azure márkanév alatt fut, hanem a Microsoft 365 programcsomag része, mint SaaS termék. Jelenleg a világ egyik vezető adatvizualizációs eszköze. Segítségével kódolás nélkül, könnyedén és gyorsan tudunk interaktív és automatikusan frissülő vizualizációkat készíteni, amit utána az üzleti döntéshozók riportokon keresztül érnek el.

Az ETL folyamat által betöltött és feldolgozott adatokat tartalmazó adatbázisra ráállítva készítettem el a Power BI riportokat, amik ütemezetten frissülnek minden nap.

4. FUNKCIONÁLIS SPECIFIKÁCIÓ

4.1. **Adatforrások**

Az adatok hozzáférhetőségét sajnálatos módon a magyar kormány nagy mértékben nehezítette azzal, hogy az adatokat nem megfelelően publikálták, azok sokszor hibásak voltak, illetve historikusan egyáltalán nem érhetőek el. A hivatalos oldalon naponta egyszer – általában reggel 7 és 11 között, de időnként csak délután – frissülnek az adatok, ahonnan a weboldal HTML kódjából lehet kiolvasni a legfrissebb számokat, de bizonyos adatokat csak képformátumú térképekről leolvasva lehet csak megszerezni. [11]

Valamennyi adatot beküldenek a WHO-nak (World Health Organization, Egészségügyi Világszervezet), a CDC-nek (United States Centers for Disease Control and Prevention, Amerikai Betegségmegelőzési és Járványvédelmi Központ), illetve az ECDC-nek (European Centre for Disease Prevention and Control, Európai Betegségmegelőzési és Járványvédelmi Központ), amelyből a Johns Hopkins University a világ szinte minden országára aggregálja a rendelkezésre álló adatot. Ezt publikusan elérhetővé teszik és minden nap frissítik. [12]

Az átoltottság aktuális statisztikáit az Our World in Data csapata összesíti hivatalos források alapján és teszi elérhetővé minden nap. [13]

Ezen adatok CSV (coma separated values, vesszővel elválasztott értékek) formátumú fájlban állnak a rendelkezésünkre, amiket az ETL rendszerek könnyen fel tudnak dolgozni ellenben a HTML weboldalakkal, ahol a kódból kell megszerezni a napi aktuális számokat és a frissítés dátumát.

Az adatok minden országra rendelkezésre állnak, a fölösleges adattárolás és számításvégzés elkerülése érdekében viszont csak azokkal az országokkal kell foglalkozni, ahol szervezetünk operációt végez. Ezen országok: Magyarország, Csehország, Szlovákia, Horvátország, Románia, Szerbia, Szlovénia, Montenegró és Bosznia-Hercegovina.

4.2. **Felhasználói felületek**

A rendszer elsődleges felhasználói felülete a generált és folyamatosan frissített riportok, amik a döntéshozók rendelkezésére állnak. Ezek a Microsoft Power BI Pro felhős szolgáltatáson keresztül kellenek, hogy elérhetőek legyenek, melynek használatához a felhasználók egyedi licenszelése szükséges.

A riportokhoz való hozzáférés csak autentikáció és autorizáció után lehetséges, tehát a felhasználónak igazolnia kell magát (autentikáció), majd a rendszer automatikusan

ellenőrzi, hogy a felhasználónak van-e ténylegesen hozzáférése az adott riporthoz, és csak ezek után enged hozzáférést.

4.3. Rendszer integrációs interfészek

A rendszernek egy REST API-on keresztül JSON formátumban elérhetővé kell tennie egy végpontot, ahol a vállalati chatbot le tudja kérdezni, – és a felhasználóknak ez alapján válaszolni – hogy egy adott helyzetben aktuálisan mennyi az esélye egy fertőzöttel való találkozásnak.

A végpont csak a vállalati belső hálóról (és ezzel együtt a site-to-site VPN-nel csatolt felhős virtuális hálózatról) szabad elérhető legyen, így túlterheléses támadás ellen nem kell külön védekezni, és lévén, hogy csak publikus adatokra alapoz, autentikációra sincs szükség.

4.4. Felhasználói profilok

A rendszer kiemelt felhasználói a döntéshozók, elsősorban az ő igényeiket kell kiszolgálni. Ezen felhasználók köre egy a vállalati címtárban található biztonsági csoport segítségével van meghatározva, ami a későbbiekben is bármikor változhat, az alkalmazás minden komponensének képesnek kell lennie a változások lekövetésére.

A vállalat meglévő chatbotján keresztül pedig a végfelhasználók köre kiterjed a vállalat összes alkalmazottjára is. Ők a megszokott módon, a vállalati Teams csevegőfelületén érhetik el a már bevezetett és működő vállalati információs chatbotot. A chatbot fejlesztőcsapata az alkalmazás elkészülte után fogja beintegrálni a chatbotba az új funkcionalitást.

4.5. Nem-funkcionális követelmények

A funkcionális követelményeken túl számos, nem közvetlenül a funkcionalitást érintő, ellenben a végleges megoldás működését nagyban befolyásoló minőségi tényezőről beszélhetünk – ezeket nem-funkcionális követelményeknek nevezzük. [14]

4.5.1. Teljesítmény

A rendszernek a webes sztenderdeknek megfelelő felhasználói élményt és teljesítményt kell nyújtania. Az alkalmazásnak reszponzívnak kell lennie, tehát a felhasználói interakcióra azonnali reakciót kell nyújtania. A háttérben történő adatbetöltésekről minden alkalommal tájékoztatni kell a felhasználót.

4.5.2. **Rendelkezésre állás**

Ugyan nincs szükség kiemelten magas rendelkezésre állású környezet kialakítására (HA, high availability), az üzleti döntések idő-szenzitivitása miatt fontos, hogy a rendszer mindig elérhető legyen, így az elvárt SLA minimum 99,9%.

Az infrastruktúra és SaaS komponensek egyedi SLA-kkal rendelkeznek, így például a Power BI Pro vagy a legtöbb Azure szolgáltatás 99.9%-os SLA-t garantál, ami kicsit szemléletesebben azt jelenti, hogy havonta 43 perc 49 másodperc állásidő fogadható el. [15]

4.5.3. **Biztonság**

A rendszer kialakítása során hangsúlyt kell fektetni az információbiztonságra, a rendszer komponensei csak megfelelő vállalati autentikáció után szabad, hogy elérhetőek legyenek. A hálózati komponensek lehetővé kell tegyék a csak belső hálózatról, vagy site-to-site VPN-nel összekapcsolt privát hálózatokból való elérhetőséget, mert a POC sikeressége esetén fejlesztett működő rendszernek ez követelménye lesz.

A szoftverbiztonságért a vállalati kiberbiztonsági csapat felel, a rendszert átadás előtt ők auditálják, illetve monitorozzák folyamatosan.

4.5.4. **Használhatóság**

Az egyik legfontosabb nem-funkcionális követelmény magának a rendszernek használhatósága. A felhasználók számára intuitív kell legyen a használat, a riportokban a különböző vizualizációk egyértelműek kellenek legyenek, ahol pedig nem azok, magyarázó szövegeket kell használni. A mindennapos használathoz nem szabad, hogy szükséges legyen felhasználói útmutató.

Ugyanitt ki kell térni a különböző adottságú felhasználókra is, például a gyengénlátóknak nagy kontrasztú módot kell lehetővé tenni, vakok számára a szövegfelolvasó programokkal kompatibilisen kell elkészíteni a felületeket, olyan szoftvereket kell választani, amik ezt támogatják (például képek helyett a képeken található tartalom leírása). Hallássérültek részére a videókat, hanganyagokat feliratozni szükséges, ugyan ez a jelen kontextusban nem releváns.

Az adatvizualizációhoz használt Microsoft Power BI rendszer ezeket mind támogatja, különös hangsúlyt fektettek a fejlesztése során a különböző nehézségekkel küzdő embertársainkra.

4.5.5. **Karbantarthatóság**

A vállalat IT szervezetét nem terhelhetjük komoly infrastruktúra üzemeltetési feladatokkal, így olyan felhős platformszolgáltatásokra kell épülnie a megoldásnak, amik karbantartása minimális erőforrást igényel. Ezek elsősorban a korábban részletezett platformszolgáltatások, ahol a felhőszolgáltató átvállalja a karbantartási feladatokat.

Az infrastruktúra karbantarthatóságán túl rendkívül fontos a fejlesztők által készített kódok minősége, a kód dokumentálása és egy teljes rendszer dokumentáció létezése is.

A Robert C. Martin által megfogalmazott „clean code”, azaz tiszta kód irányelvek közül a következőek a legfontosabbak: [16]

- Egy kód akkor lesz „jó”, ha egyszerű, mindenki számára **könnyen érthető** – tehát nem kell sokat beszélni róla.
- Az **elnevezéseknek egyértelműnek és beszédesnek kell lenniük**. Jelen kontextusban a tényleges kódon túl ez a felhős szolgáltatások elnevezése során fontos. Nem mindegy, hogy egy COVID-19 adatokat tároló adatbázisnak az a neve, hogy „covid”, vagy az, hogy „sqldb-covid-prod-westeu-001” amiben név alapján rögtön egyértelművé válik mindenkinek, hogy ez egy SQL adatbázis, COVID-19 adatokat tárol, Production, tehát éles környezetben működik és a West Europe adatközpontban található fizikailag Amszterdamban. Ez az elnevezés pontosan megfelel a Microsoft részletes és precíz elnevezéstanának is. [17]
- A kód **megfelelő kommentelése**, megjegyzésekkel való ellátása is egy fontos pont. A kommenteket nem azért kell írni, hogy megmagyarázzuk az enélkül érthetetlen „rossz” kódot, hanem a kód fontos részeire, körülményeire figyelem felhívásként.
- A **kód formázására** konkrét automatizálható előírásokat kell alkalmazni, hogy a többi fejlesztő által készített kód egységes kinézetű legyen.
- A **hibakezelés** egy sokszor nem megfelelően megvalósított eleme a projekteknek. Minden megoldást fel kell készíteni a különböző hálózati, tranziens és egyéb hibákra, ezek nem érhetik totálisan váratlanul a rendszert és a felhasználókat megfelelően kell erről tájékoztatni is. Ez alól felmentést élveznek a korábban tárgyalt POC alkalmazások, ahol a cél nem egy minden esetben tökéletesen működő alkalmazás, hanem egy koncepció validálása.
- Hasonló kivétel érvényes a tesztelésre is. A kód működésének biztosításához a fejlesztés idejében és a fejlesztés/karbantartás későbbi szakaszaiban is elengedhetetlen a **tesztelés**. Ez háromféleképp valósulhat meg:
    - Unit testing: a kód egyetlen pici részét kiemelve, minden más függőséget helyettesítve (mocking) teszteljük az adott bemenetre kapott eredményt.
    - Integration testing: az adott komponens dependenciáihoz való integrációját teszteljük vele nagyobb mennyiségű kódot lefedő tesztekkel.

- End-to-end testing: akárcsak a felhasználó, egy automatizált teszt során ugyanazokat a kattintásokat végzi el a teszt, mint amit a felhasználó csinálna, ezzel pontosan tesztelve, hogy az megfelelően működik-e.

Ezeken túl a karbantarthatóság véleményem szerint egyik legfontosabb aspektusa a verziókezelés, ami manapság szinte minden esetben Git segítségével valósul meg. Egy fához hasonlóan a fejlesztők különböző branch-eken (ágakon) dolgoznak, így nem zavarják egymás munkáját, majd az ezeken történő változásokat merge-ök és pull request-ek segítségével vezetik vissza a közös branch-be. Ezekre többféle metodológia is létezik, minél nagyobb egy projekt, annál komplexebb rendszert kell kialakítani. A jelen rendszer POC volta miatt ez kevésbé releváns. Az elkészült kódokat Git segítségével verziókezelni kell, de komplex branching stratégiára nincs szükség.

A verziókezelésen túl egy fejlesztőcsapat életének sarokkövét képezi a DevOps rendszer. Ilyen megtalálható a GitHub-ban is az Issues, illetve Actions képében, de a Microsoft Azure DevOps is egy hosszú múltra visszatekintő teljeskörű megoldás. A hibák, feladatok követésétől kezdve az automatizált fordításig, tesztelésig és telepítésig lefedik a teljes folyamatot, ez utóbbi elnevezése a Continuous Integration / Continuous Delivery (CI/CD).

5. ARCHITEKTÚRA TERVEZÉS

5.1. Azure Architecture Center

A felhő platformok extrém komplexitása miatt látszólag ugyanazon platform több különböző megoldása is megfelelő lehet egy adott problémára. A Microsoft Azure az egyik legnagyobb és legkomplexebb felhő platform, a megfelelő komponensek kiválasztásának elősegítésére ezért létrehozták az Architecture Center-t, ahol architektek és fejlesztők részletezett referencia architektúrákat tanulmányozhatnak. A rendszer elkészítése során is alkalmaztam ezt az eszközt. [18]

5.2. Infrastruktúra architektúra

5. ábra: Az infrastruktúra komponens architektúrája.

Az infrastruktúra három fő komponensből áll:

- Az adatok betöltését, frissítését és előkészítését végző ETL folyamat, amit egy Azure Data Factory (ADF) segítségével valósítottam meg. Ennek belső működését az alábbiakban részletezem.
- Az adattároló komponensek, melyek jelen esetben egy Azure Blob Storage, ahol a napi bejövő adatokat CSV-ben archiválom, illetve két Azure SQL Database az ETL különböző stádiumainak.

- Az adatpiacra kapcsolódó analitikai és adatvizualizációs megoldások, specifikusan egy Power BI riport és az Azure Machine Learning.

5.3. Azure Data Factory adatfolyam architektúra

Az Azure Data Factory feladata az adatok letöltése az eredeti forrásokról, majd ezek tisztítása, átalakítása és betöltése Azure SQL adatbázisokba. Ehhez külön ún. pipeline-okat hoztam létre, ezzel szétbontva alfeladatokra a megoldandó problémát. A pipeline-ok indítását és sorrendben történő végrehajtását egy orkesztrátor pipeline menedzseli, ez felel az ETL logika komponenseinek egységbe zárásáért és ütemezéséért.

5.3.1. Extract pipeline

Az adatbetöltő pipeline időzítve fut le minden nap 01:00 UTC időpontban és két elemből áll:

- Az első Copy activity bemenete egy HTTP data source, ami a naponta frissülő adatállományra mutat. A forrásadat CSV, azaz vesszővel elválasztott értékek formátumban van. Ezt a szöveges fájlt egy Azure Blob Storage-ba menti.
- A második Copy activity innen beolvassa az adatokat és egy Full load eredményeképp a Staging SQL adatbázisba tölti őket. A napi adatok mindig tartalmazzák visszamenőleg a járvány kitörése óta az összes adatot, és ezek utólag is változhatnak amikor az adatszolgáltató hatóságok rájönnek, hogy valahol nem megfelelő volt az adatszolgáltatás – ezért van szükség Full load-ra.

5.3.2. Transform pipeline

Az adattraszformációs pipeline az orkesztrátor segítségével indul el, amikor az Extract pipeline befejeződik.

Megtörténik az adatok tisztítása, a különböző adatforrások aggregálásra kerülnek, adat integráció vizsgálat történik (minden napra minden országnak pontosan egy rekordjának kell lennie).

5.3.3. Load pipeline

Az adatbetöltő pipeline szintén az orkesztrátor által kerül elindításra amint a Transform pipeline a feladatai végére ér.

Egy Data Flow segítségével az adatok a megfelelő adatpiac SQL adatbázisba kerülnek.

5.4. **Adatbázis architektúra**

A Staging adatbázis mindösszesen egy táblát tartalmaz, ide kerül összesítésre minden adat, ami egy ország egy napjához tartozik. Így tehát a kulcs az ország, illetve a dátum mezőkből tevődik össze.

Az adatpiac adatbázisa ezzel szemben egy csillagsémát valósít meg. A ténytábla maga a napi statisztika országokra lebontva, és két dimenzió táblában pedig az országokhoz tartozó információk, illetve a dátumok kibontva találhatóak.

5.5. **Analitika és adatvizualizáció**

A Power BI időzített adatforrás frissítéssel automatikusan betölti az adatpiacról a frissített adatokat és ezek az adatvizualizációkon is megjelennek.

Az Azure Machine Learning is erre az adatpiacra van rácsatlakoztatva, innen kerül betanításra a predikciós modell.

6. IMPLEMENTÁCIÓ FOLYAMATA

6.1. Felhős erőforrások létrehozása

A felhős implementáció első lépése az infrastruktúra létrehozása. Ehhez többféle – a felhőszolgáltató által elérhetővé tett – eszközt is használhatunk. A Microsoft Azure esetében ezek közé tartozik maga a Portál, ahol egy grafikus felületen (GUI, graphical user interface) „kattintgathatjuk" össze a rendszert, de emellett elérhetőek az úgynevezett Azure Resource Manager template-ek, azaz egy szöveges leírás az infrastruktúra komponenseiről és azok konfigurációiról. Ezeket az ipar sokszor Infrastructure as Code néven emlegeti, és ide tartoznak harmadik fél által készített, vendor-független [1] megoldások is, mint például a Terraform vagy az Ansible. Ezentúl Azure esetében PowerShell vagy Azure CLI (command line interface, parancssor) scriptekkel is lehet automatizálni, vagy akár közvetlenül használva az Azure REST API-ját, bármilyen nyelvből automatizálható az erőforrások provizionálását.

Én kényelmi szempontból a Portál segítségével hoztam létre az erőforrásokat, de a megoldás elkészülte után készítettem egy Azure Resource Manager template-et a későbbi automatizáció érdekében.

A Microsoft Azure esetében minden erőforrásnak pontosan egy címtárban (Azure Active Directory), egy előfizetésben és egy erőforráscsoportban kell lennie. Ezek a menedzsment hierarchia szintjei is, a menedzsment csoportokkal kiegészítve, melyek a címtár és az előfizetés között helyezkednek el opcionálisan.

Az előfizetés gyakorlatilag egy virtuális reprezentációja annak a szerződéses viszonynak a Microsofttal, ahogy az igénybe vett szolgáltatások után fizetünk. Ez lehet egy a Microsofttal közvetlenül aláírt Enterprise Agreement (ahol a számlázás egy közvetítő, ún. Licensing Solution Provider segítségével történik), egy Microsoft partneren keresztül például egy Cloud Solution Provider szerződés formájában, de akár közvetlenül a Microsoftnak bankkártyával vagy valamilyen szponzoráció keretein belül (Microsoft for Startups, Azure for Research, Azure for Students, stb.).

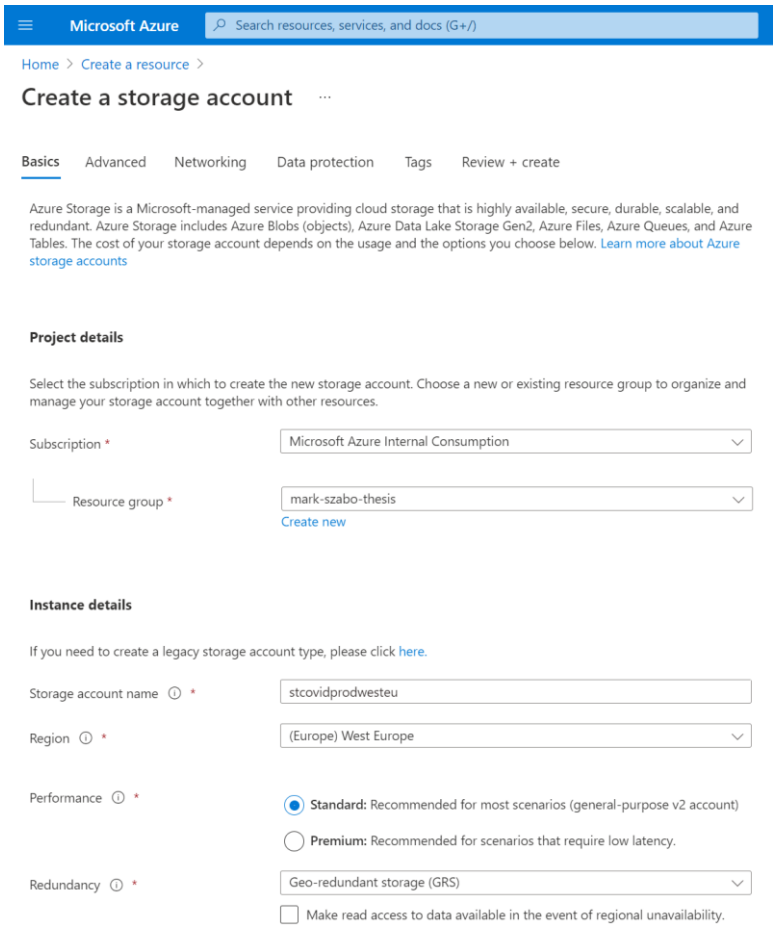
Az erőforráscsoportok pedig egy kizárólag logikai kapcsolatot hivatottak létrehozni a különböző erőforrások között, elsősorban az átláthatóság és a hozzáférés kezelés érdekében. A különböző projektek erőforrásait, a különböző környezeteket (prod, dev, stb.), a különböző fizikai lokációkat érdemes például különböző csoportokba elkülöníteni. Utólagos átjárásra természetesen van lehetőség, mind erőforrás-csoport, előfizetés és

[1] Egyik szolgáltatóhoz sem elköteleződő, köztük szükség esetén váltást könnyen lehetővé tevő.

címtár szinten is, de van számos megkötés, ami nehezíti a munkát, leállást okozhat egy ilyen migráció során.

6.1.1. **Storage Account**

Az ETL folyamat első fázisaként a forrásrendszerekből az adat az Azure Blob Storage-ben kerül archiválásra. Ehhez egy Storage Account-ot kell létrehoznunk, melynek a Blob Storage egy funkciója. A létrehozás során először egy előfizetést kell választanunk, ezzel határozzuk meg hogyan szándékozunk fizetni a szolgáltatásért. Az erőforráscsoport kiválasztása lépésnél lehőségünk van új csoportot is létrehoznunk, ha még nem tettük meg ez korábban. Az én esetemben a *mark-szabo-thesis* elnevezésű erőforrás-csoportban dolgozom.



6. ábra: Az Azure Storage Account konfigurációja a létrehozás során.

El kell neveznünk az erőforrást, ez fogja meghatározni az URL-jét is a tárolónak és a benne tárolt fájloknak. A régió kiválasztásánál számos szempont játszik szerepet. Általánosságban mondva érdemes a felhasználókhoz legközelebbi adatközpontot választani. Ez a közelség azonban nem mindig fizikailag értendő, inkább az átlagos

válaszidő a kulcs, azt szeretnénk, hogy a felhasználók minél gyorsabban választ kapjanak a felhő felé intézett kéréseikre. Magyarországon, illetve a Közép-Kelet Európai régióban – a lengyel és görög adatközpontok elkészültéig és üzembelépéséig – az Amszterdamban található West Europe régió az általánosságban leggyorsabban elérhető adatközpont, ahol minden szolgáltatás széleskörűen elérhető. Ez többek között az ott kilépő transzatlanti internetvezetékekre és az oda bekötött Európán belüli fő hálózatra vezethető vissza.

A teljesítmény opció mellett két lehetőség közül választhatunk, a HDD alapú Standard és az SSD-kre épülő Performance közül nekünk – lévén, hogy a felhasználók sose fognak közvetlenül interakciót létesíteni a Blob Storage-gel – teljesen megfelel a Standard is. Ezzel természetesen költséget is takaríthatunk meg.

Végül a redundancia esetében több lehetőség is a rendelkezésünkre áll:

- A **lokálisan redundáns tárolás** (LRS, Locally-redundant storage) esetében minden adat 3 példányban lesz eltárolva egyetlen fizikai adatközpontban. Ez a legolcsóbb és a lehetőségek közül a legkevésbé biztosított megoldás, csak a lemezekben bekövetkező hardverhibától véd.
- A **zóna-redundáns tároló** (ZRS, Zone-redundant storage) kétszer három példányban tárolja el az adatainkat, és a két zóna egymástól fizikailag, energetikailag, hűtésileg, internet-kapcsolat ügyileg szeparálva vannak – de még mindig lehet egy fizikai adatközpont lokáció egymáshoz közel eső telephelyei.
- A **georedundáns tárolás** (GRS, geo-redundant storage) ezt továbbfejlesztve a két zónát egymástól messze található, különböző adatközpont régiókban helyezi el, ehhez régió párokat alkalmaz (mint például West Europe – North Europe), így akár egy teljes adatközpont megsemmisülése esetén is biztonságban lennének az adataink.
- Újabban lehetőség van ezt még egy szinttel magasabbra emelni, és a **geo-zóna redundáns tároló** (GZRS, geo-zone-redundant storage) segítségével 2 külön adatközpont régióban, 2-2 zónában, 3-3-3-3 példányban, tehát összesen 12 példányban kerül eltárolásra az adat. Mindezen megoldások ugyan egy esetleges természeti katasztrófa vagy más fizikai és hardverhiba esetében segítenek elkerülni az adatvesztést, a leállás nélküli üzemeltetésben nem segítenek.
- Ehhez rendelkezésre állnak a **read-access geo-redundant** (RA-GRS) és **read-access geo-zone-redundant** (RA-GZRS) megoldások, amelyek esetében a másodlagos régióban található szinkronizált adatok is mindig elérhetőek egy másodlagos végponton keresztül. Így az elsődleges végpont elérhetetlensége esetében egy úgynevezett failover műveletet tudunk alkalmazás-szinten elvégezni, amely segítségével az alkalmazás átáll a másodlagos infrastruktúra komponensekre.

A GRS, GZRS, RA-GRA és RA-GZRS megoldások kritikus és különlegesen kritikus felhasználási esetekben indokoltak, ahol az adatveszteség felbecsülhetetlen károkat

okozna, a ZRS elsősorban biztonsági mentési felhasználásra ajánlott, az LRS pedig nem-kritikus feladatkörökben használható. Mivel az általunk készített rendszer önmagában sem kritikus rendelkezésre állású, a Blob Storage kiesése esetén pedig továbbra is használható az összes felhasználói felület és API, ezentúl kizárólag archivált publikusan elérhető adatokat tárolunk benne, nem indokolt a lokális redundancia fölötti megoldások költségnövelése.

Az Azure Storage számlázása teljesen fogyasztás alapú, pontosan annyit kell fizetnünk amennyi adatot tárolunk, illetve amennyi adatot kifelé forgalmazunk. Az adatközpontba befelé történő adatforgalom mindig ingyenes.

A konfigurációs lehetőségek között számos speciális, adatbiztonságra vonatkozó opció áll még rendelkezésre, mint például saját kulccsal történő titkosítás, virtuális hálózatba csatlakoztatás, de a jelen megoldás esetében ilyen magas adatbiztonsági elvárások hiányában ezekkel nem foglalkozunk.

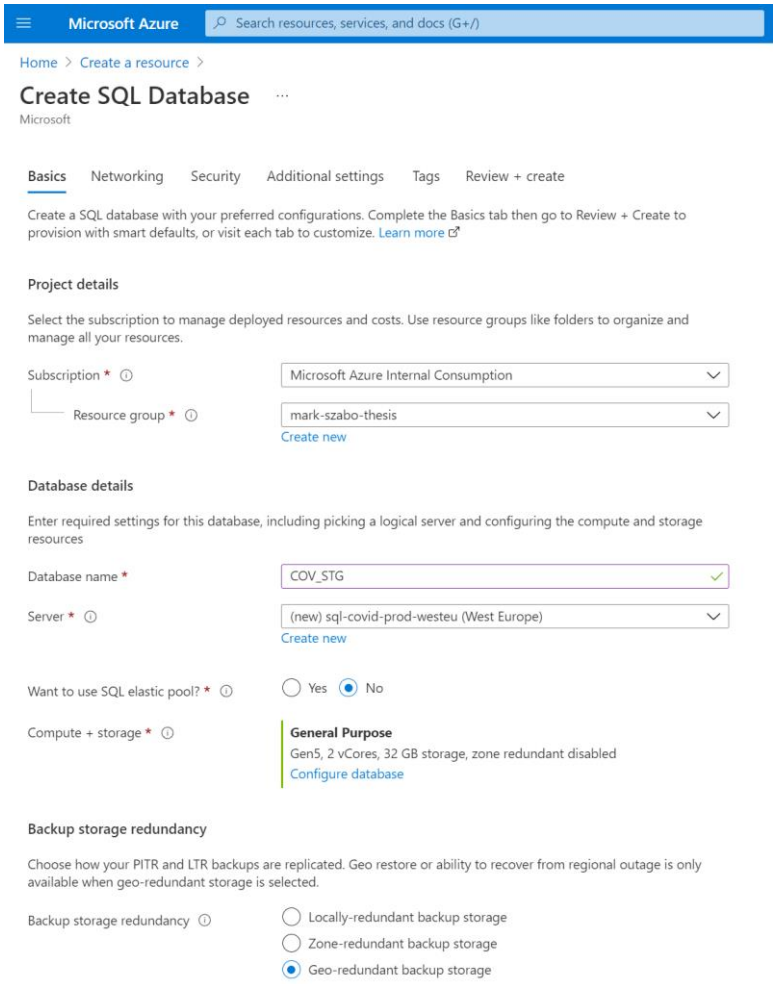
Ugyanitt a speciális beállítási lehetőségeknél kell az Azure Data Lake opciót is bekapcsolni.

6.1.2. **SQL szerver és adatbázisok**

Az SQL szerver esetében a Storage Account-hoz hasonlóan nem a hálózati és egyéb biztonsági funkciókra kerül a hangsúly, habár ezeket mind támogatja, és később szükség is lesz rájuk a POC után. Erről a továbbfejlesztési lehelőségek fejezetben részletesebben értekezem.

Mint minden erőforrás létrehozásakor, itt is ki kell választani az előfizetést és az erőforráscsoportot, majd az adatbázis létrehozása előtt egy szervert is készítenünk kell – amennyiben nem áll még rendelkezésünkre. Esetünkben két külön adatbázist készítettem a staging (COV_STG) és a data mart (COV_DM), azaz adatpiac környezeteknek, ezek viszont egy szerveren helyezkednek el. Az első adatbázis létrehozása során egy szerver is létrejött, a másodiknál már csak ki kellett a meglévőt választani.

A számítási kapacitás opciónál meg kell határozni az adatbázis számára rendelkezésre álló erőforrásokat. Mindkét adatbázis esetében egy kétmagos ötödik generációs hardvert választottam, ezek a POC során jelentős ráhagyással is biztosan elegendőek lesznek. Ezután a rendszert éles környezetbe állításakor felettébb egyszerűen fel tudjuk skálázni mindenféle adatmigráció nélkül, néhány kattintás segítségével.

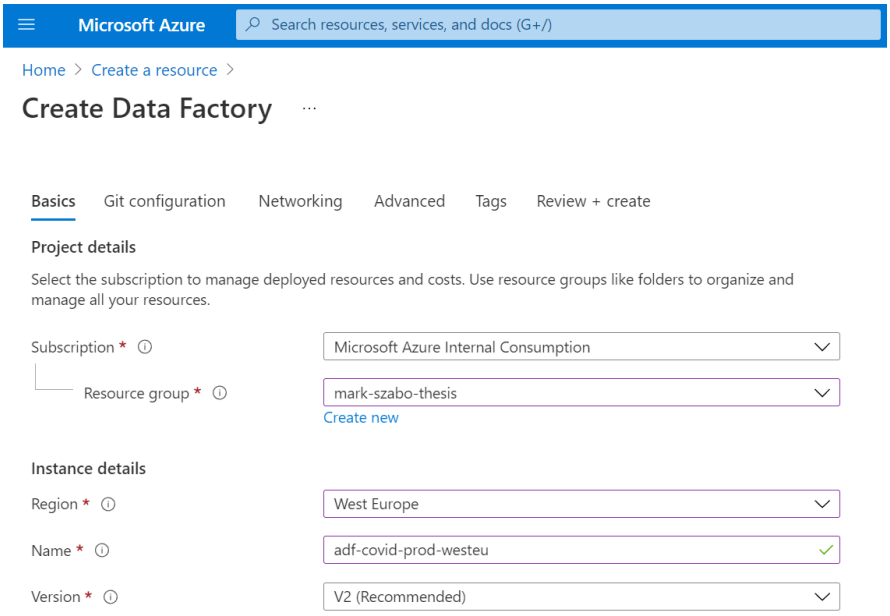


7. ábra: Az Azure SQL Database beállítási lehetőségei.

6.1.3. **Data Factory**

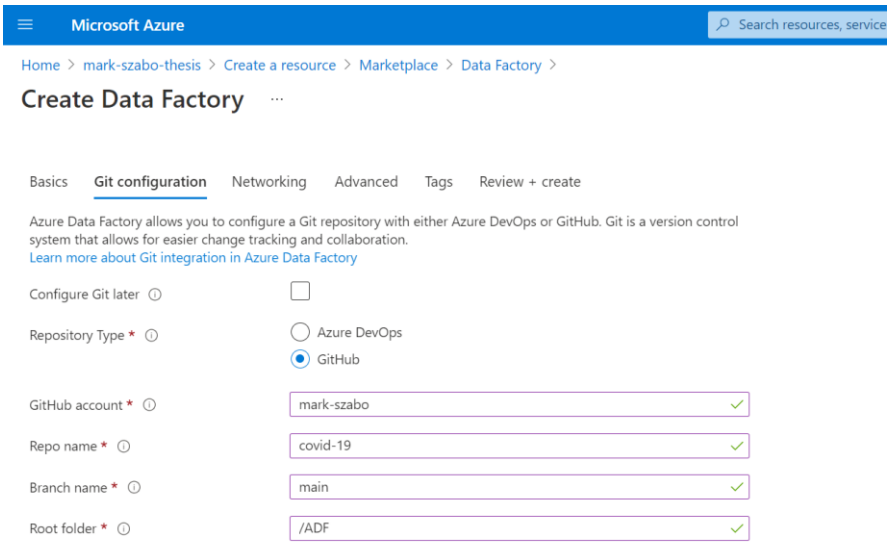
Az Azure Data Factory létrehozása során több lépésben is fontos beállításokat kell elvégeznünk. A korábban ismertetett módon működik az előfizetés, az erőforráscsoport, a régió és az erőforrás nevének meghatározása.

A verzió esetében az első és a jelenlegi második verzió között lehet választani. Erre a visszafelé kompatibilitás miatt van szükség, a második verzióban olyan változtatások kerültek bevezetésre, amik törést okoztak volna a már működő rendszerben. A Microsoft tradicionálisan kifejezetten ügyel ennek elkerülésére, így két független verziót hoztak létre. Egy zöldmezős projekt esetében nincs mi indokolja az elavult verzió használatát, így én is a v2-t választottam.



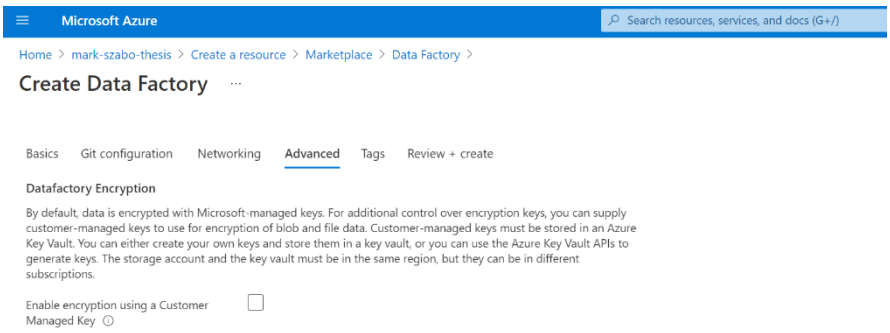
8. ábra: Az Azure Data Factory alapvető paraméterei a létrehozás során.

Második lépésben a verziókezelés beállítása következik. A specifikációban meghatározott módon minden kódot, jelen esetben a generált kódokat is git-ben kell tárolni. Ehhez a GitHub szolgáltatását vettem igénybe, mely a világ legnagyobb open-source (nyílt forráskódú) kód tárolója. A használatához engedélyt kellett adni az Azure részére a repository írására és meg kellett határozni melyik repository melyik branch-ére és milyen könyvtárba dolgozzon.



9. ábra: A verziókezelés beállítása.

A speciális beállítási konfigurációnál arra is van lehetőség, hogy mint a Blob Storage esetében saját magunk által menedzselt kulccsal titkosítsuk a feldolgozott adatokat.



10. ábra: Az Azure Data Factory speciális konfigurációs lehetőségei.

6.1.4. **Serverless Function**

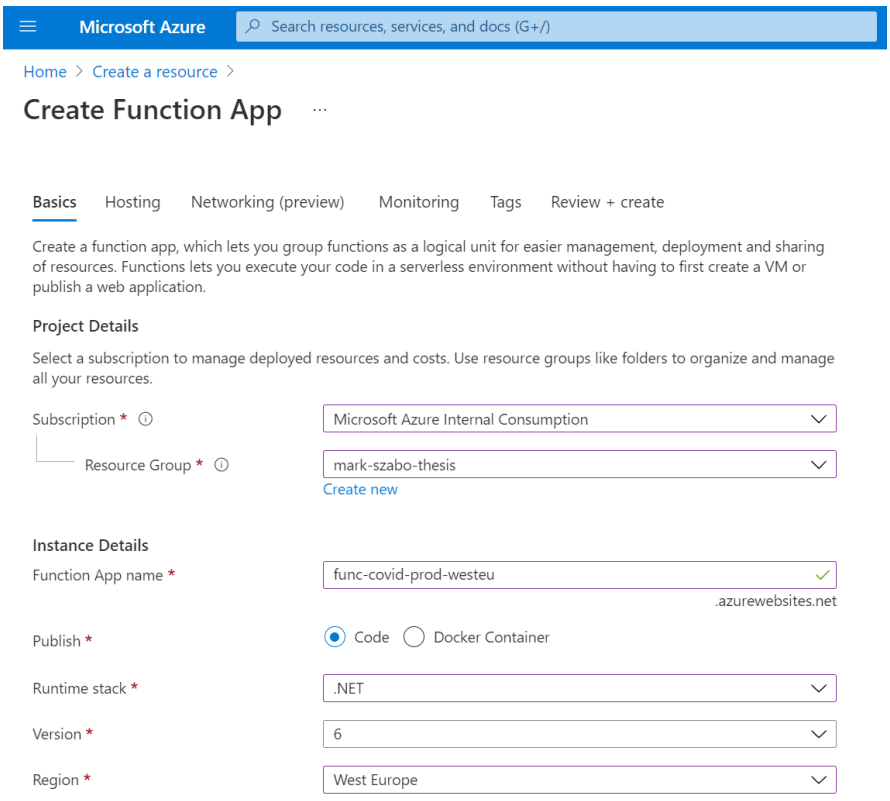
Az Azure Function esetében a szokásos konfigurációból kiemelendő, hogy az erőforrás nevéből képződik a webszolgáltatás URL-je, itt lesz majd elérhető az API végpont. Ezentúl ki kell választanunk, hogy közvetlenül kódot, vagy egy konténerizált alkalmazást szeretnénk majd futtatni. Előbbi esetében a Function app szolgáltatja a megfelelő futtatókörnyezetet, így ki kell ezeket is választani. Jelen esetben ez .NET 6, mely kiválasztásának szempontjairól később értekezem.

A Hosting, vagyis kiszolgálás lépésben ki kellett választani az alkalmazást kiszolgáló számítási kapacitás mivoltát. Itt több opció is rendelkezésre áll:

- A **Consumption plan** esetében ténylegesen csak azután kell fizetni ahányszor az adott funkció lefut, ez az igazi serverless működés. Automatikusan skálázódik és a legköltséghatékonyabb megoldás.
- A **Premium plan** a Consumption Plan limitációit hivatott feloldani, illetve többek között lehetővé teszi a virtuális hálózatokhoz való csatlakozást vagy egyedi linux képfájlokon történő futtatást.
- Az **App Service Plan** esetében elveszítjük a serverless működést, és egy előre provízionált, fix számítási kapacitáson fog lefutni minden kód. Ez viszont az eredményezi, hogy amennyiben egyáltalán nincs forgalom, akkor is ugyanúgy ki kell fizetni a kapacitást, illetve túlterhelés esetén limitációkba ütközhetünk és a megoldás elérhetetlenné válhat.

A jelen felhasználási igényeket a Consumption plan teljes mértékben kielégíti, futtatása elhanyagolható költséget generál. Amennyiben később a virtuális hálózatokhoz való csatlakozás szükségessé válik, a Premium plan-re való átállás triviális feladat lesz.

A monitorozási lehetőségeknél egy Azure Application Insight-ot hoztam létre, ahova a Function app a telemetriáit fogja eljuttatni, így a hibakeresést nagyban megkönnyítve.



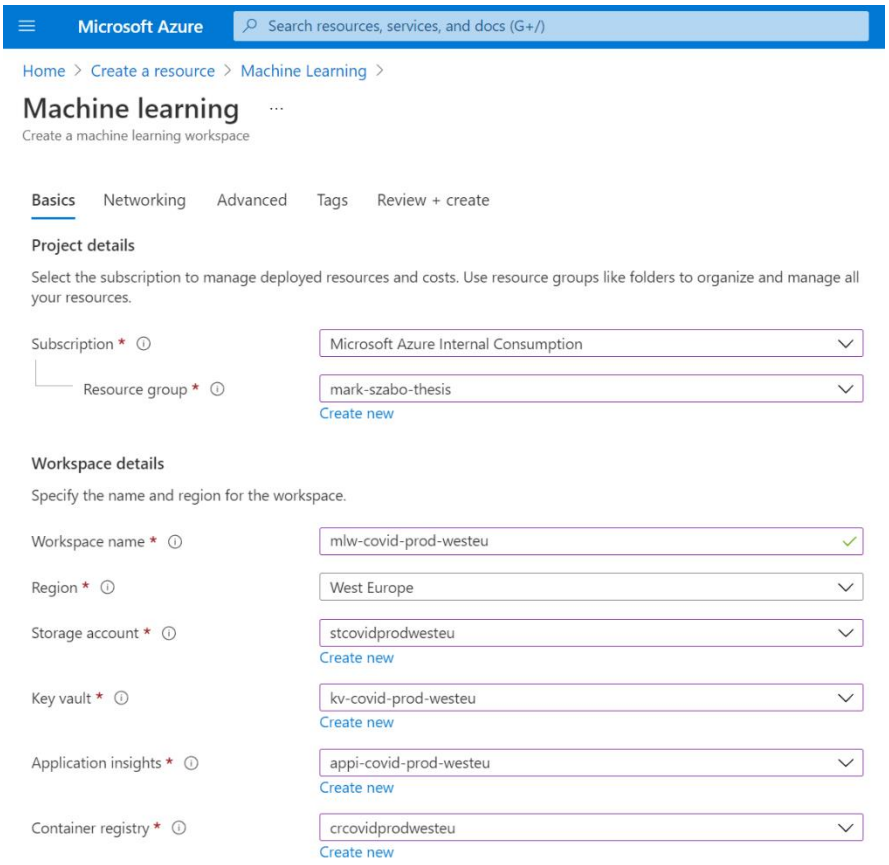
11. ábra: A serverless Function App beállításai.

6.1.5. **Machine Learning**

Utolsó elemként az Azure Machine Learning Workspace létrehozása során az eddigiekhez hasonló folyamaton megyünk végig, eltekintve a függőségek beállításától, mely egy eddig nem bemutatott elem. A korábban már létrehozott Storage Account és Application Insights mellett egy Container Registry-re és egy Key Vault-ra lesz szükség. Előbbiben az elkészült machine learning modellek tárolódnak konténerekbe csomagolva, utóbbiban pedig a kulcsok és tanúsítványok kapnak biztonságos helyet.

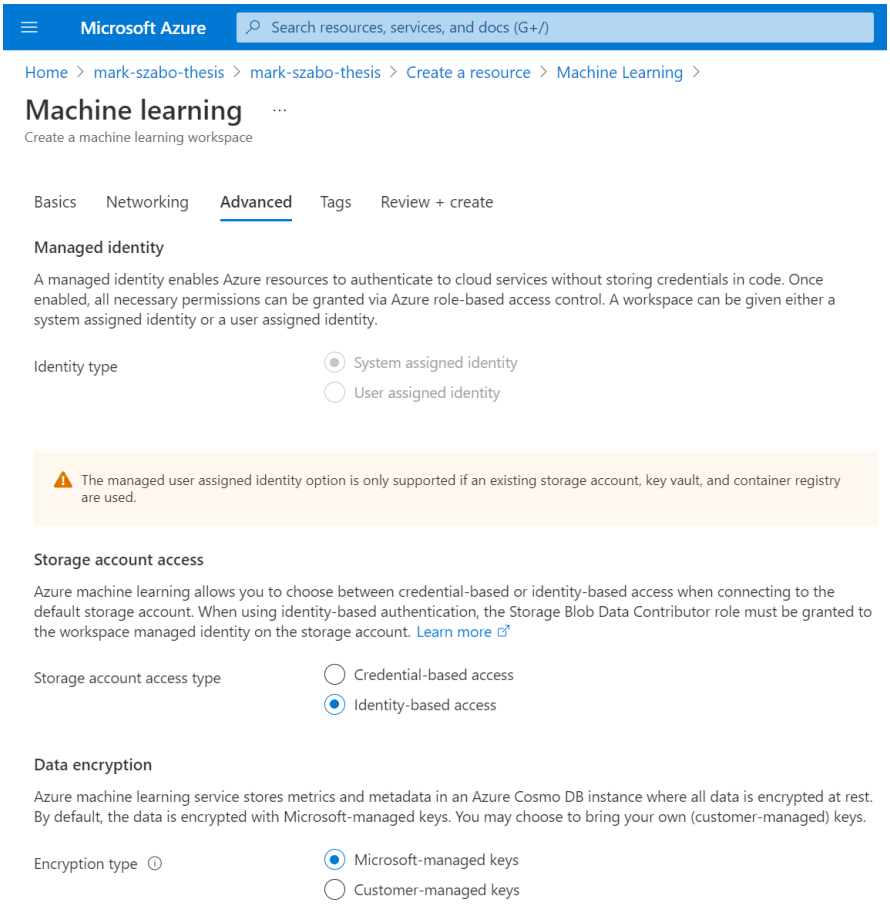
A speciális beállítási lehetőségek között itt is megtaláljuk a hozott kulccsal történő titkosítást, de emellett egy úgynevezett Managed Identity-t is létrehoz. A Managed Identity – és annak két fajtája, a Managed System Identity (MSI) és a User Assigned Managed Identity – olyan a platform által menedzselt identitás, amivel egyes szolgáltatás példányokat tudunk felruházni. Így tehát nem kell például a Machine Learning Workspace-nek megadni az SQL szerver jelszavát, hiszen a Workspace közvetlenül a saját nevében is el tudja érni az adatbázisban tárolt adatokat miután a korábban említett RBAC segítségével hozzáférést adtunk neki. Ezt a módszert alkalmazva megszűnik a kényszer a titkos kulcsok tárolására.

A Managed System Identiy esetében egyetlen infrastruktúra komponens példányhoz társítunk identitást, így annak életciklusa is szorosan összefonódik az identitásával. Amennyiben a példányt töröljük, az identitás is törlődik. Minden MSI pontosan egy példánnyal áll kapcsolatban.



12. ábra: Az Azure Machine Learning Workspace alapvető beállításai.

A User Assigned Managed Identity ezzel szemben egy általunk létrehozott infrastruktúra komponensektől független identitás, aminek életciklusa a komponensektől független, és egyszerre akár több komponens példány is rendelkezhet ugyanazzal az identitással.



13. ábra: Speciális, bizonsági konfugurációk.

6.2. **Adatbázis provizionálása**

Az adatbázis struktúra létrehozása, beleértve a táblákat és az indexeket, SQL scriptek segítségével történt, melyeket egy tetszőleges adatbázis kezelő program segítségével a megfelelő autentikáció után futtattam le.

6.3. **ETL folyamat Azure Data Factory segítségével**

Pipeline neve	**Forrás**	**Cél**	**Leírás**
PL_covid_orchestrator	HTTP források	COV_DM SQL adatbázis	Az ETL pipelineok megfelelő sorrendbeli indításáért felelős pipeline.
PL_covid_extract	GitHub HTTP	Blob Storage	Az adatok forrásrendszerekből a Blob Storage-ba történő

			letöltését és archiválását végző pipeline.
PL_covid_transform	Blob Storage	COV_STG SQL adatbázis	A már letöltött adatok transzformálását és staging adatbázisba töltését végző pipeline.
PL_covid_load	COV_STG SQL adatbázis	COV_DM SQL adatbázis	A csillagsémát előállító, adatokat aggregáló és data mart-ba töltő pipeline.

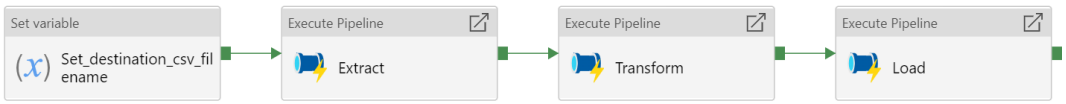
1. táblázat: Az ETL folyamat pipeline-jai.

6.3.1. **Orchestrator**

Az Orchestrator pipeline összesen 3 pipeline-t indító aktivitást tartalmaz, illetve ezek előtt egy változót állít be, amely a fájl nevét generálja le az aktuális dátum, illetve a futtatási azonosító alapján az alábbi módon:

```
@concat(formatDateTime(utcnow(), 'yyyy-MM-dd'), '_', pipeline().RunId, '.csv')
```

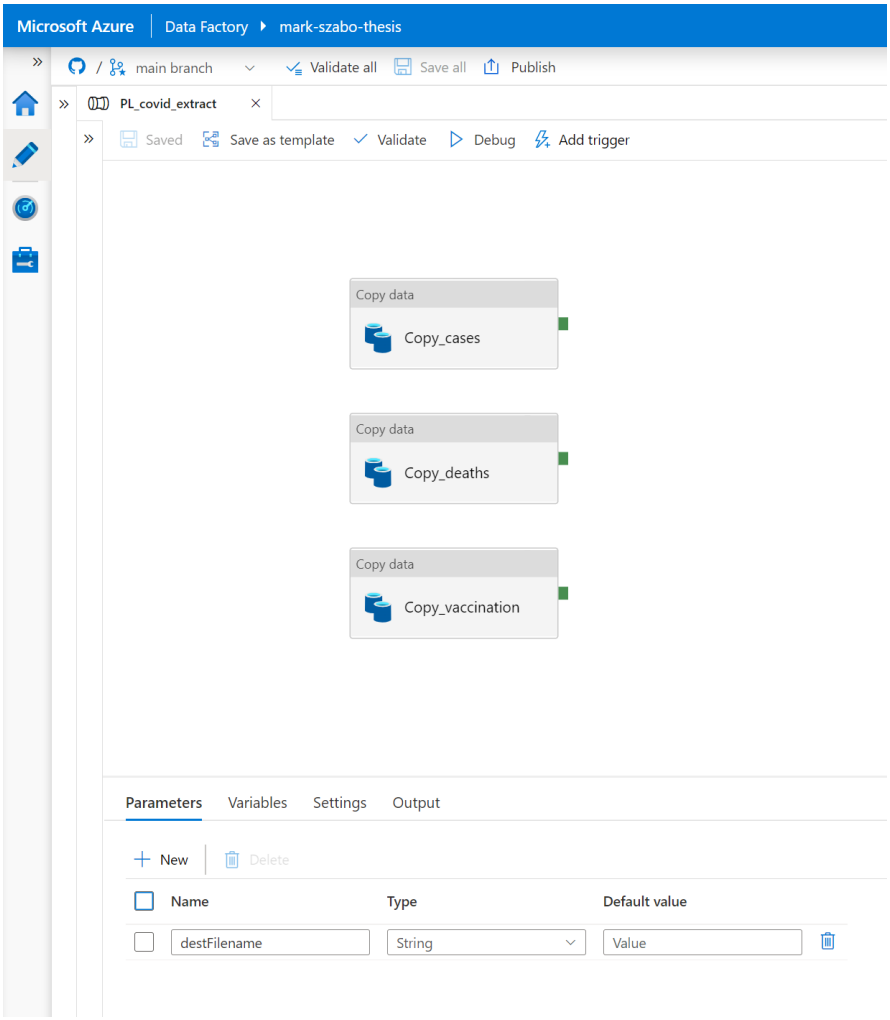
Az aktivitások sorba vannak kötve, így a változó beállítása után fog lefutni az Extract pipeline, annak befejeződése után a Transform és ugyanígy végül a Load. Ezen pipeline-ok paraméterként kapják meg az említett változót.



14. ábra: Az orkesztrátor pipeline.

6.3.2. **Extract**

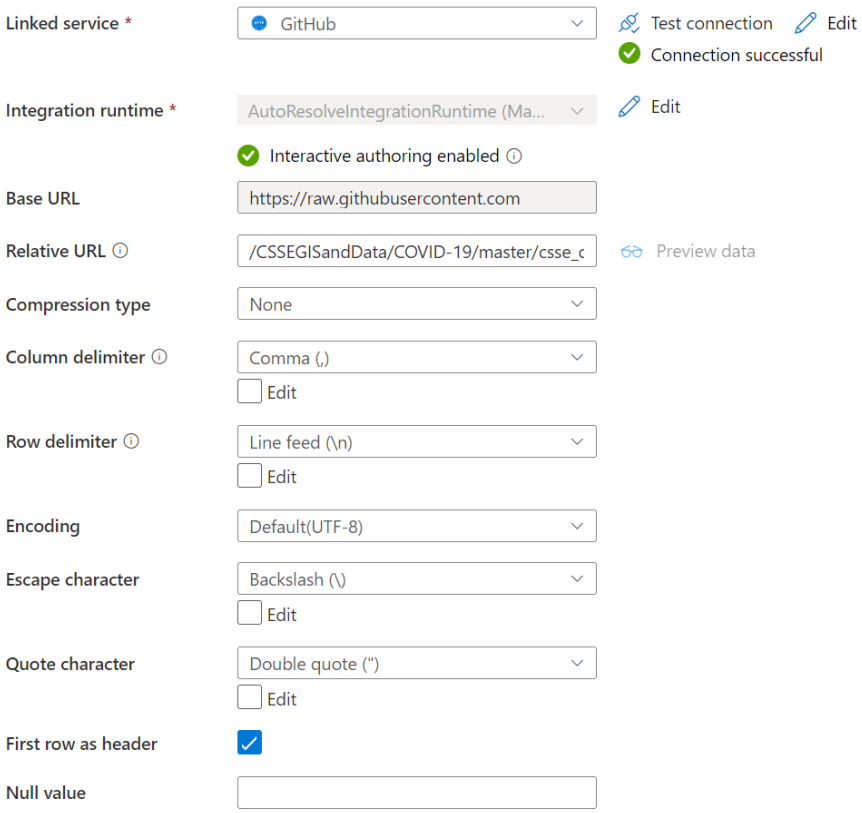
Az Extract pipeline 3 egymással párhuzamosan futó Copy aktivitást tartalmaz, melyek a forrás rendszerből HTTP-n keresztül letöltik a napi aktuális adatokat és ezeket ugyanazon Blob Storage konténer különböző virtuális mappáiba mentik el. A mentési fájlnevet az orkesztrátortól kapja egy pipeline paraméteren keresztül.



15. ábra: Az Extract pipeline áttekintése.

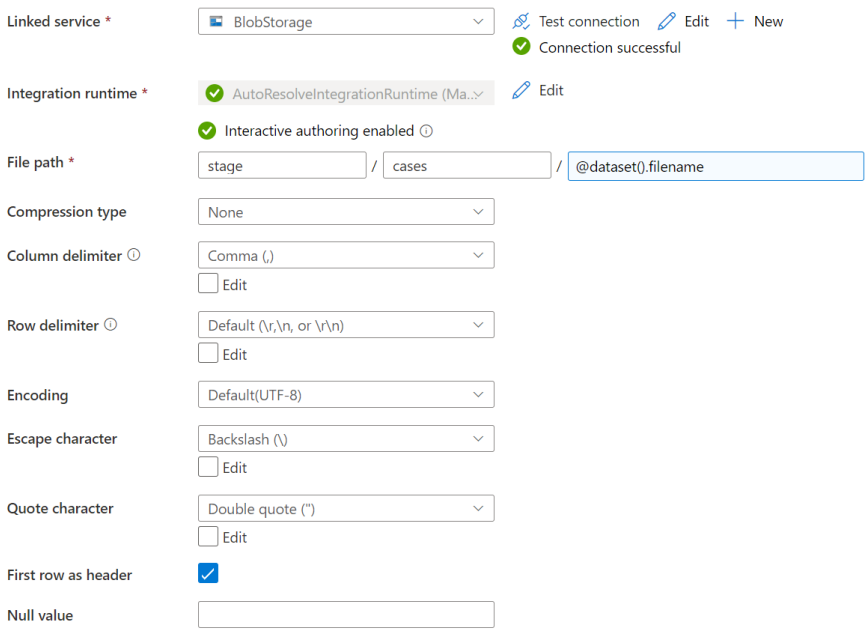
Az adatforrás konfigurációjánál be kellett állítani a szolgáltatást, ami az adatot rendelkezésre bocsájtja, illetve a hozzá tartozó autentikációt. A GitHub-on elérhetővé tett CSV formátumú adatok nem várnak el semmilyen autentikációt, így anonim módon le lehet tölteni őket.

Ezután a CSV fájl oszlop- és sorelválasztó karaktereit, enkódolását, tömörítését, első sorának oszlopcímként történő használatát és egyebeket lehetett konfigurálni. A rendelkezésre álló források angol típusú, vesszővel elválasztott adatokat tartalmaztak, így ennek megfelelően állítottam be.



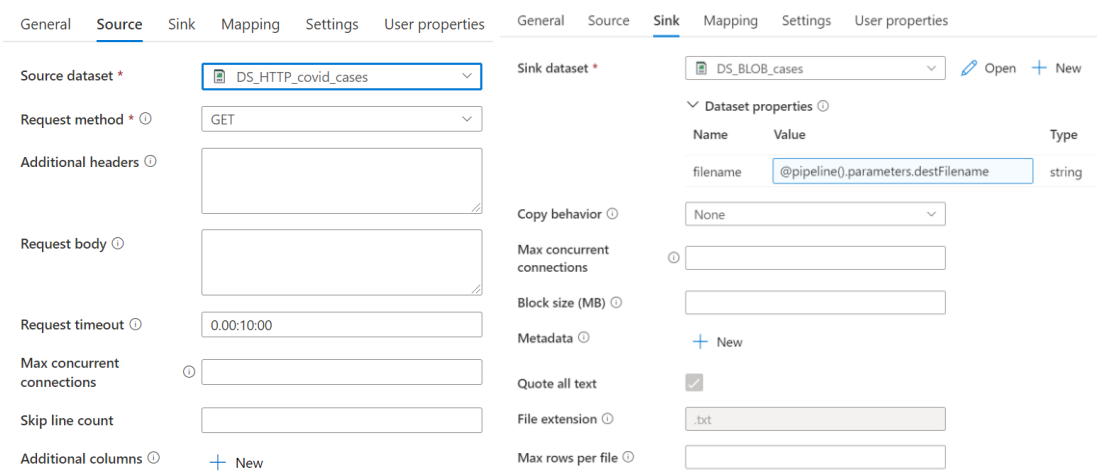
16. ábra: Az adat forrás konfigurációja.

A cél tároló mindhárom esetben ugyanazon Blob Storage, melynek a stage elnevezésű konténerében a *cases*, *deaths* és *vaccination* nevű virtuális mappákat használom. A Blob Storage nem rendelkezik mappaszerkezettel, a virtuális mappák gyakorlatilag a fájlnevekből épülnek fel a vizualizáció során az azokban található elválasztó karakterek segítségével. A forráshoz hasonlóan a CSV konfigurációját itt is el kell végezni. Ezt az Azure Data Factory önállóan kezeli.



17. ábra: A cél tároló konfigurációja.

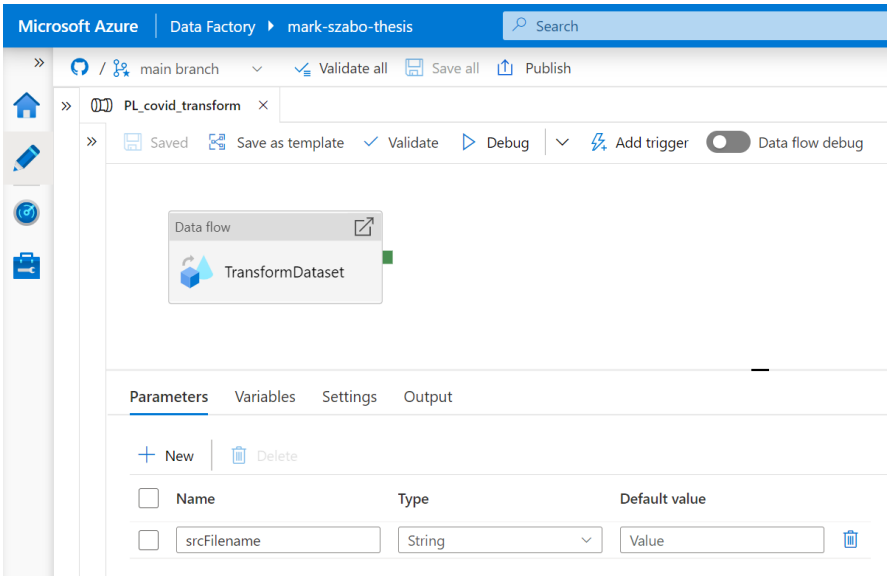
A Copy Data aktivitások konfigurációjánál csak a megfelelő forrás és cél tárolókat kellett kiválasztani, és autentikáció, illetve egyéb HTTP konfiguráció hiányában mindent az alapértelmezett értéken lehetett hagyni. A cél tárolónak dataset paraméterben tovább kell adni a pipeline paraméterben kapott generált fájlnevet.



18. ábra: A Copy Data aktivitás forrás (bal) és cél (jobb) beállításai.

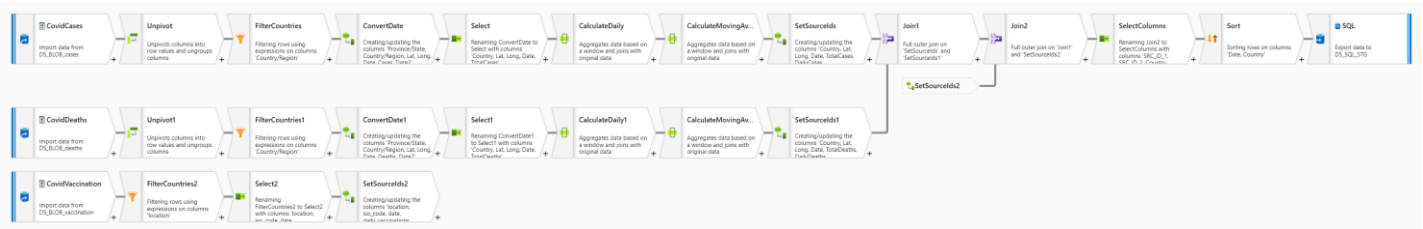
6.3.3. **Transform**

A Transform pipeline egyetlen Data Flow-t tartalmaz.



19. ábra: A Transform pipeline áttekintése.

A transzformációs Data Flow a Blob Storage-ből indul, ahonnan mindhárom forrást betölti (*cases*, *deaths* és *vaccination* adatszettek). Ezeket szükség szerint egy unpivot művelettel transzponálja, a kívánt országokra szűri, dátumokat konvertál, a megfelelő oszlopokra szűr, majd kiszámolja a napi különbségeket. Az így kapott feldolgozott adatokból 7 napos mozgóátlagot számol, beállítja a forrás azonosítókat, ezek alapján összefűzi őket, dátum szerint rendezi, majd az egészet a *staging* adatbázisba tölti. A következőkben ezeket a műveleteket részletezem.



20. ábra: A Transform Data Flow elemei.

Transzformáció neve	**Forrás**	**Típus**	**Leírás**
CovidCases	Blob Storage	Source	A forrásrendszerből, név szerint a Blob Storage-ből betölti az adatokat. Konfigurációja során fontos volt a schema drift-et bekapcsolni, ami lehetővé teszi, hogy a forrás oszlopainak változása esetén is megfelelően működjön. Erre a fertőzési és
CovidDeaths	Blob Storage	Source	
CovidVaccination	Blob Storage	Source	

			halálozási adatok miatt volt szükség, melyek minden nappal eggyel több oszlopot tartalmaznak.
Unpivot	CovidCases	Unpivot	Az unpivot művelet transzponálja és kibontja az adatot. A forrásrendszerekben minden sor egy ország, és a hozzá tartozó oszlopokban találhatóak az adatok idősorosan. Ebből az unpivot segítségével minden országnak minden napra létrehozunk egy külön sort.
Unpivot1	CovidDeaths	Unpivot	
FilterCountries	Unpivot	Filter	A specifikációnak megfelelően csak a szervezetünknek szükséges országokkal foglalkozunk, így minden más ország adatait figyelmen kívül hagyjuk.
FilterCountries1	Unpivot1	Filter	
FilterCountries2	CovidVaccination	Filter	
ConvertDate	FilterCountries	Derived column	A speciális dátumformátumban (’M/d/yy’) érkező adatokat manuálisan dátummá kell konvertálni. Ehhez egy Derived Column, azaz származtatott oszlop transzformációt használtam.
ConvertDate1	FilterCountries1	Derived column	
Select	ConvertDate	Select	A Select transzformáció segítségével a nem szükséges oszlopoktól szabadultam meg.
Select1	ConvertDate1	Select	
Select2	FilterCountries2	Select	
CalculateDaily	Select	Window	A Window egy speciális transzformáció, segítségével egymás utáni sorokkal végezhetünk műveletet egy mozgó ablak segítségével. A napi különbségek kiszámításához országok szerint csoportosítva, dátum szerint növekvő sorrendben rendezve, 1 nap
CalculateDaily1	Select1	Window	

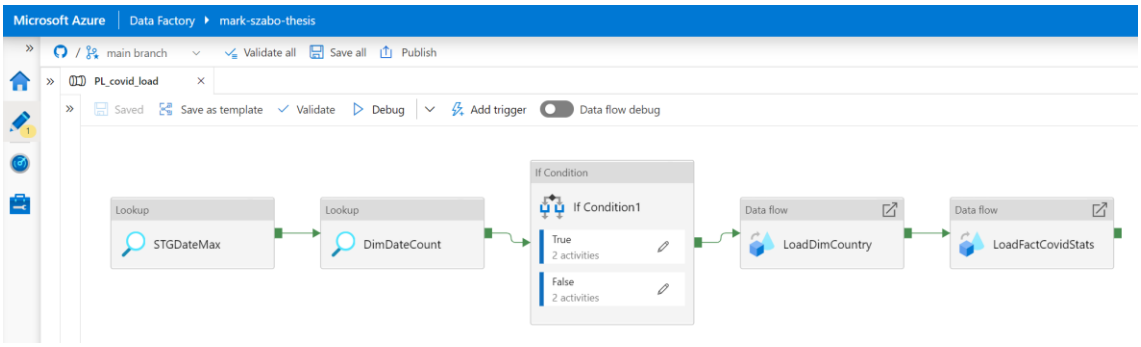
			hosszúságú ablakmérettel dolgozva az aktuális sortól visszafelé mérve az utolsó (azaz az aktuális nap) és a legelső (azaz az előző nap) különbségét vettem.
CalculateMoving-Average	CalculateDaily	Window	A napi különbségekhez hasonlóan a 7 napos mozgóátlag kiszámolásához is a Window transzformációt vettem igénybe, ahol ugyanúgy ország szerint csoportosítva és dátum szerint növekvő sorrendbe rendezve az adatot, 7 nap hosszúságú ablakot használva a múltban, az aktuális nappal bezárólag átlagot számoltam és azt egészre kerekítettem.
CalculateMoving-Average1	CalculateDaily1	Window	
SetSourceIds	CalculateMoving-Average	Derived column	A forrásazonosítók beállítását szintén egy származtatott oszloppal oldottam meg, ahol az első számú azonosító (SRC_ID_1) az ország neve változtatás nélkül, a másodlagos (SRC_ID_2) pedig a dátum 'yyyyMMdd' formátumban szöveggé konvertálva.
SetSourceIds1	CalculateMoving-Average1	Derived column	
SetSourceIds2	Select2	Derived column	
Join1	SetSourceIds és SetSourceIds1	Join	A napi és országbontásúra hozott fertőzési és halálozási adatokat egy full outer join segítségével egyesítettem a két forrásazonosító mentén.
Join2	Join1 és SetSourceIds2	Join	Az eredetileg is napi és országbontású vakcinációs adatokat szintén full outer join segítségével szintén a két forrásazonosító mentén egyesítettem.

SelectColumns	Join2	Select	A join-ok által keletkező duplikált oszlopokat egy Select transzformáció segítségével elliminnáltam.
Sort	SelectColumns	Sort	Az egységesített adatszettet dátum és ország szerint rendezi.
SQL	Sort	Sink	Végül az adatokat a *staging* adatbázisba tölti.

2. táblázat: Az adattranszformációs Data Flow transzformációi.

6.3.4. **Load**

A Load pipeline a csillagséma kialakításáért és a *staging* adatbázisból a adatpiacba (*DM*) való adattöltésért felel.



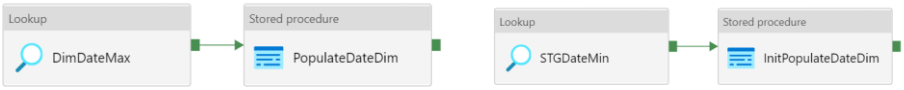
21. ábra: A Load pipeline áttekintése.

A pipeline két Lookup aktivitással indul, melyek egyszerű SQL lekérdezéseket futtatnak le. Az első megszerzi a *staging* adatbázisban található legnagyobb dátumot, ez fogja megadni az új legutolsó napot. A második a *DM* adatbázis *DimDate* táblájában megszámolja az értékek számát.

Ezután egy elágazás következik: amennyiben a *DimDate* tábla nem üres (a második legkérdezés eredménye nagyobb, mint nulla) a *DimDate* tábla maximuma és a *staging*-ben található korábban kinyert maximum között kell legenerálnunk a dátumokat és ezeket a dátum dimenzióba tölteni. Ehhez először a korábbiakban ismertetett módon egy Lookup-ot használtam, majd egy erre a célra írt tárolt eljárással generálom a dátumokat.

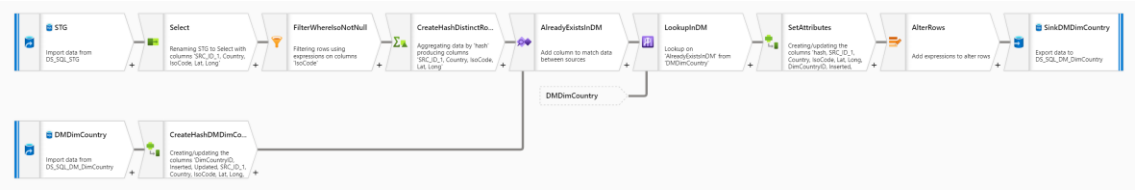
Amennyiben a *DimDate* tábla viszont üres, a *staging*-ben található legkisebb és legnagyobb érték között kell a dátumokat generálni. Ehhez szintén egy Lookup-ot és ugyanazt a tárolt eljárást használtam más paraméterezéssel.

Ennek segítségével az adatpiacon mindig csak a tényleges dátumok lesznek megtalálhatóak és nincs szükség a dátumok egy arbitrális jövőbeli időpontig történő legenerálására.



22. ábra: Az elágazás igaz (bal) és hamis (jobb) ága.

Mindezek után két Data Flow következik, sorrendben először az ország dimenzió, majd a ténytábla betöltésére. A sorrend kötött, ellenkező esetben megtörténhet, hogy egy – az ország dimenzióban nem létező – értékre próbálnánk hivatkozni a ténytáblában.



23. ábra: Az ország dimenzió Data Flow elemei.

A *LoadDimCountry* Data Flow a *staging* adatbázisból a *DM* adatbázis *DimCountry* táblájába tölti az adatokat. Hasonlóképpen a *LoadFactCovidStats* Data Flow a *DM FactCovidStats* táblájába tölti a ténytábla adatait.

Mindkét esetben csak az új és változott rekordokat töltjük át a *DM*-be. Ehhez a Slowly Changing Dimension Type 1 módszert alkalmaztam, ahol az adatokat felülírjuk, a korábbi verziókat nem tároljuk.

Transzformáció neve	**Forrás**	**Típus**	**Leírás**
STG	Staging adatbázis	Source	A *staging* adatbázisból érkező adatok forrásként.
DMDimCountry	DM adatbázis	Source	A *DM* adatbázis ország dimenziójában jelenleg is megtalálható adatok.
Select	STG	Select	A *staging* adatbázisból leszűrt, országokra vonatozó oszlopok.
FilterWhere-IsoNotNull	Select	Filter	Az egyik forrás nem tartalmazza az ISO kódokat, így a sorok egy részében ezek a mezők nincsenek

			kitöltve. Ezekre nincs szükség, így kiszűrjük őket.
CreateHash-DistinctRows	FilterWhere-IsoNotNull	Aggregate	SHA2 algoritmussal generált hash az ország adataiból.
CreateHash-DMDimCountry	DMDimCountry	Derived column	
AlreadyExists-InDM	CreateHash-DistinctRows és CreateHash-DMDimCountry	Exists	Egy Exists transzformáció segítségével a nem talált hash-ekre szűr.
LookupInDM	AlreadyExists-InDM és DMDimCountry	Lookup	Az attribútumok (specifikusan a beszúrás dátuma) nem található meg a staging adatbázisban, így ezt egy Lookup segítségével a *DM*-ből szerzi meg.
SetAttributes	LookupInDM	Derived column	A beszúrás és frissítés attribútumait frissíti. A beszúrás értéke abban az esetben változik, amennyiben a rekord még nem létezik, így az előző Lookup nem tér vissza semmilyen eredménnyel.
AlterRows	SetAttributes	Alter row	Megjelöli az összes rekordot frissítésre (lévén, hogy már csak a változást tartalmazó rekordok érnek el ehhez a lépéshez).
SinkDM-DimCountry	AlterRows	Sink	Végül az adatokat a *DM* adatbázisba tölti.

3. táblázat: Az ország dimenzió Data Flow transzformációi.

24. ábra: A ténytáblát feltöltő Data Flow elemei.

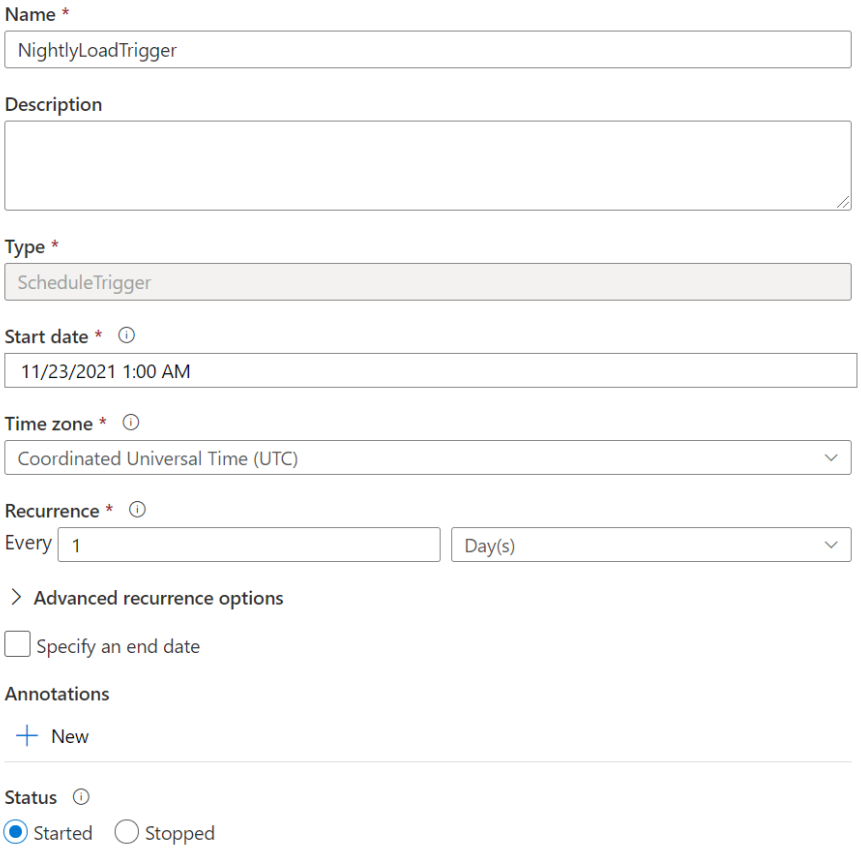
Transzformáció neve	**Forrás**	**Típus**	**Leírás**
STG	Staging adatbázis	Source	A *staging* adatbázisból érkező adatok forrásként.
DMFact-CovidStats	DM adatbázis	Source	A *DM* adatbázis ténytáblájában jelenleg is megtalálható adatok.
DMDimCountry	DM adatbázis	Source	A *DM* adatbázis ország dimenziójában található már frissített adatok.
DMDimDate	DM adatbázis	Source	A *DM* adatbázis dátum dimenziójában található már frissített adatok.
Select	STG	Select	A *staging* adatbázisból leszűrt, a ténytáblába való oszlopok.
CreateHash	Select	Derived column	SHA2 algoritmussal generált hash a ténytábla adataiból.
CreateHashDM-FactCovidStats	DMFact-CovidStats	Derived column	
AlreadyExists-InDM	CreateHash és CreateHashDM-FactCovidStats	Exists	Egy Exists transzformáció segítségével a nem talált hash-ekre szűr.
LookupInDM	AlreadyExists-InDM és DMFact-CovidStats	Lookup	Az attribútumok (specifikusan a beszúrás dátuma) nem található meg a *staging* adatbázisban, így ezt egy Lookup segítségével a *DM*-ből szerzi meg.
SetAttributes	LookupInDM	Derived column	A beszúrás és frissítés attribútumait frissíti. A beszúrás értéke abban az esetben változik, amennyiben a rekord még nem létezik, így az előző Lookup nem tér vissza semmilyen eredménnyel.

LookupCountry	SetAttributes és DMDimCountry	Lookup	Forrásazonosító alapján megkeresi a megfelelő rekordot az ország dimenzióból.
LookupDate	LookupCountry és DMDimDate	Lookup	Forrásazonosító alapján megkeresi a megfelelő rekordot az dátum dimenzióból.
AlterRows	LookupDate	Alter row	Megjelöli az összes rekordot frissítésre (lévén, hogy már csak a változást tartalmazó rekordok érnek el ehhez a lépéshez).
SinkDMFact-CovidStats	AlterRows	Sink	Végül az adatokat a *DM* adatbázisba tölti.

4. táblázat: Az ténytábla Data Flow transzformációi.

6.3.5. **Trigger**

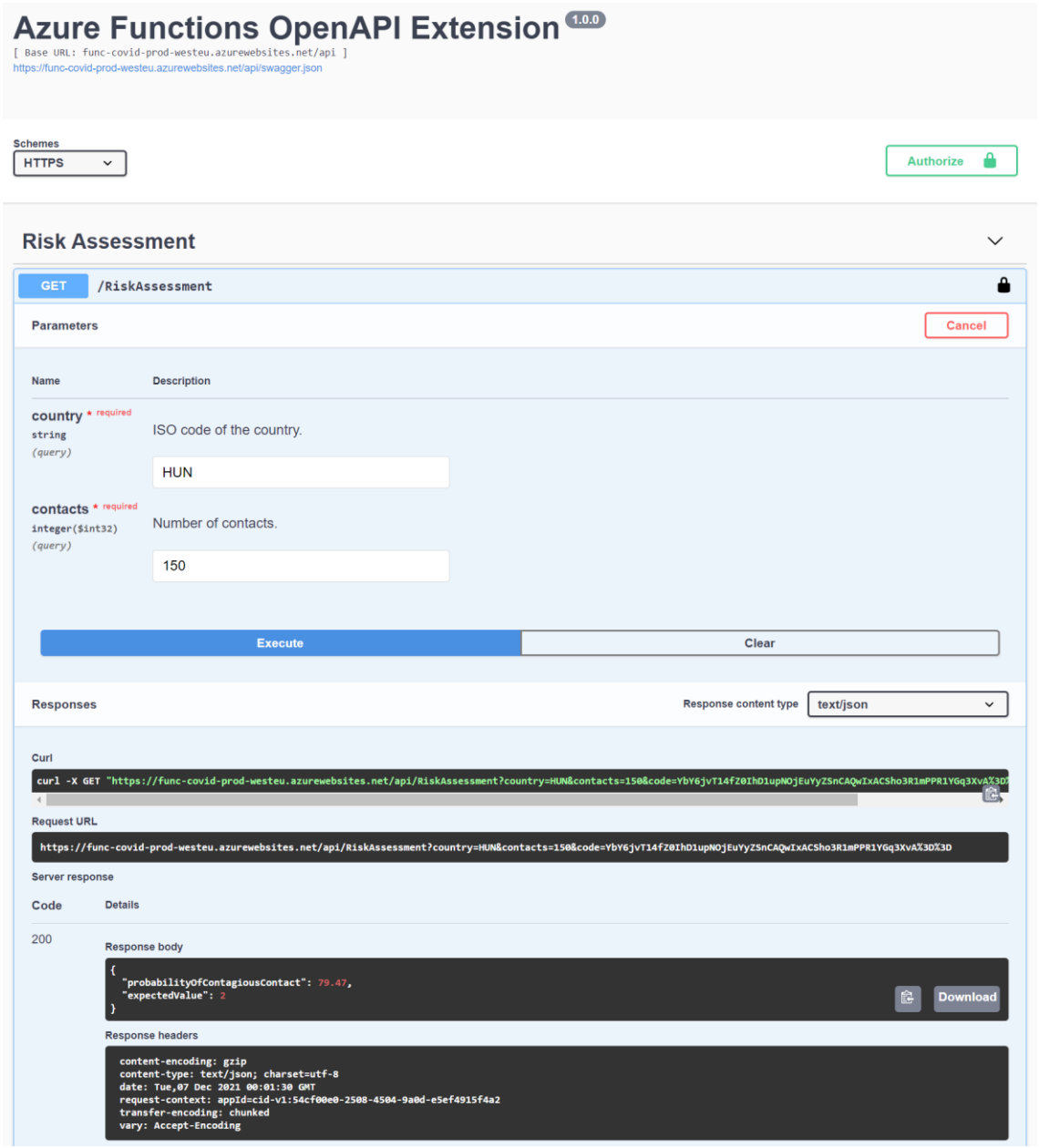
A specifikáció szerinti időzített töltés egy Trigger segítségével indul el minden éjszaka.



25. ábra: Az éjszakai adattöltés indításáért felelős Trigger konfigurációja.

6.4. **API végpont Azure Functions segítségével**

Az Azure Function app implementálásához először létrehoztam a projektet Visual Studio Code-ban. Ehhez az Azure Functions VS Code bővítményt használtam. A projekt generálása során meg kellett adni a keretrendszert (esetemben .NET 6), a sablont (HTTP trigger with OpenAPI), az autentikáció típusát (Function) és a projekt, illetve a namespace nevét.



26. ábra: A funkció által hosztolt API végpont Swagger dokumentációja és tesztelő felülete.

A keretrendszer kiválasztása során a saját kompetenciáim és a .NET 6 LTS[2] mivolta voltak meghatározóak. A specifikációban meghatározott módon egy API végpontot kell rendelkezésre bocsájtani, így a HTTP trigger volt a megfelelő választás, ehhez a Swagger dokumentáció generálásához az OpenAPI[3] szabványt támogató HTTP trigger sablont használtam. Követelmény volt a biztonságos működés, így a beépített Function autentikációt használva egy API kulcs segítségével válik elérhetővé a függvény.

A Function a HTTP hívás paramétereiből megkapja az ország és kontaktusszám bemeneti változókat, majd ehhez egy SQL binding segítségével hozzátölti az éppen aktuális aktív fertőzött számokat a *DM* adatbázisból. Ezt az elmúlt 14 nap napi fertőzött számait összegezve kapja meg. Ezekből az alábbi matematikai formula segítségével számolja ki a legalább egy fertőzött valószínűségét:

$$p = 1 - \left(1 - \frac{f_a}{n_o}\right)^k$$

Ahol f_a az aktuális fertőzöttek száma, n_o az ország népessége, k a kontaktusszám és p a valószínűség arra, hogy legalább egy aktív fertőzött található a kontakt személyek között.

Az SQL binding segítségével nem ütköztem nehézségekbe az SQL adatbázishoz való csatlakozás során, elég volt csak az SQL lekérdezést, illetve a connection string-re mutató alkalmazás beállítás nevét megadni, az Azure Function így a kész eredményt adja csak vissza függvényparaméterként.

A Function app forráskódján a Git verziókezelő egy külön branch-én dolgoztam és a forráskód GitHub-on van tárolva.

6.5. **Járvány-prediktor az Azure Machine Learning szolgáltatással**

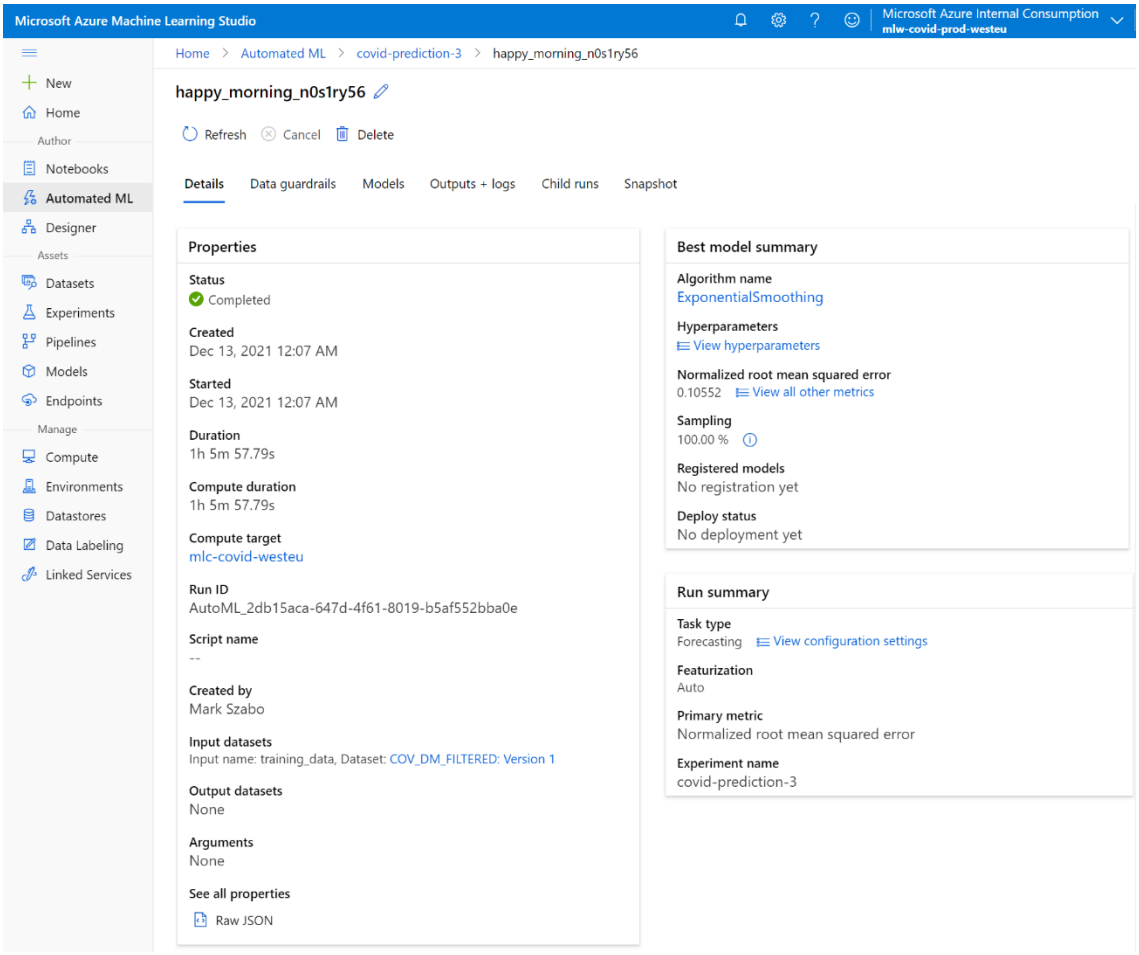
A járványadatok előrejelzéséhez az Azure Machine Learning Automated ML megoldását használtam, aminek segítségével kódolás nélkül lehet gépi tanulási modelleket készíteni. Ilyenkor a rendszer ugyanazon a tanítási adatszetten többféle algoritmust is kipróbál, majd ezeket leteszteli, és a legjobb modellt tartja csak meg.

---

[2] Long Term Support (LTS), hosszútávon támogatott verzió. .NET esetében minden második verzió LTS, ezek 3 évig támogatottak az első stabil megjelenéstől számítva. A közte lévő verziók mindösszesen 18 hónapig támogatottak, így például az előző .NET 5 verzió támogatottsága 2022 májusban megszűnik, míg a .NET 6 egészen 2024 novemberig javítva lesz. [19]

[3] Az OpenAPI Specification (OAS) egy olyan szabvány, ami egy sztenderd, programozási nyelv-független interfész leírást definiál HTTP API-ok számára. Segítségével mind emberileg, mind programmatikusan értelmezhetővé és felfedezhetővé válnak a szolgáltatás képességei és annak használata. A Swagger pedig az OpenAPI implementációját és felhasználását segítő eszközök gyűjteménye, ahogy annak egy példája a fenti ábrán is látható. [20]

A teszteléshez k-fold cross validation-t használtam, aminek segítségével ugyanazon az adaton tudtam tanítani és tesztelni is a modellt. Az algoritmus az adatszett k darabra bontása után egyesével végigiterál a szétbontott adatokon, majd az aktuális rész kivételével a többit a tanításhoz használja fel, az aktuálisat pedig a teszteléshez. Így k alkalommal fut le a tanítás-tesztelés folyamat és végül a legjobb modell marad csak meg.

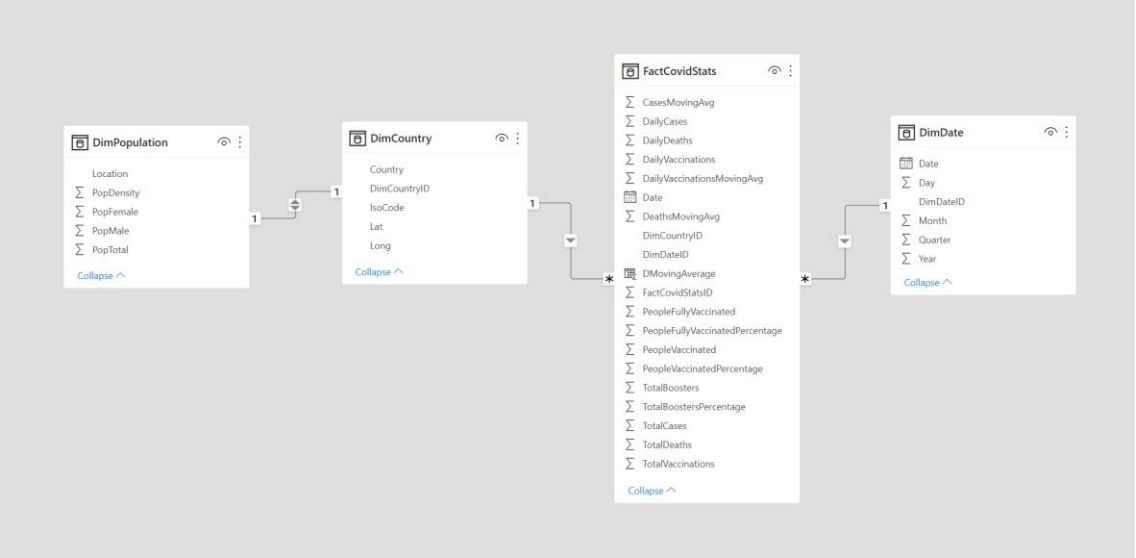


27. ábra: Az Automated ML tanításának eredménye.

Az így betanított modellel elért normalizált négyzetes középhiba 0.10552, ami egy elfogadható eredmény. Az éles használat során egy kézzel finomhangolt Azure ML Notebook-ban írt modell lenne a célravezető.

6.6. **Riportolás Power BI segítségével**

A Power BI riport elkészítéséhez a Power BI Desktop alkalmazásra volt szükség, ahová megfelelő autentikáció után betöltöttem az adatokat a *DM* adatbázisból. A Power BI automatikusan létrehozta az idegen kulcsok alapján a kapcsolatokat a táblák között.



28. ábra: Tábla kapcsolatok vizualizációja Power BI-ban.

Az országokra vetített lakosságarányos fertőzési és halálozási adatokhoz szükségem volt az aktuális populációkra országok szerint. Ezt az ENSZ publikusan elérhető adatforrásaiból oldottam meg egy Web Source segítségével. Betöltés közben a Power BI leszűri az adatokat az éppen aktuális évre, majd a számunkra érdekes országokra és végül 1000-rel szoroztam a számadatokat tartalmazó mezőket, lévén, hogy a forrásrendszerben 1000 emberben voltak megadva az adatok.

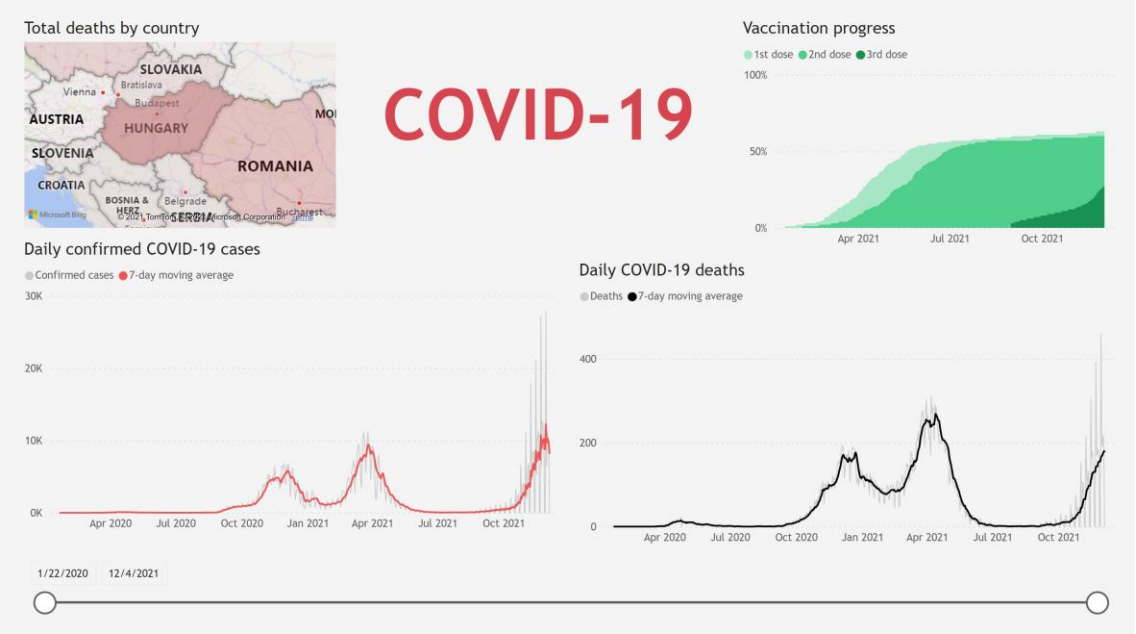
Természetesen megoldás lehetett volna erre az is, hogy az ETL folyamat során betöltöm ezeket az adatokat a *DM* adatbázisba, de a POC keretein belül minél több megvalósítási lehetőséget szerettem volna prezentálni, és így lehetőségem nyílt a közvetlenül Power BI-ban történő adatgazdagítást is bemutatni.

A Power BI funkcionalitásának prezentációjára a halálozási adatok 7 napos mozgóátlagát egy DAX lekérdezés segítségével egy kalkulált oszlopban újraszámoltam:

```
DeathsMovingAverage =
CALCULATE (
  AVERAGE( FactCovidStats[DailyDeaths] ),
  DATESINPERIOD ( FactCovidStats[Date], LASTDATE ( FactCovidStats[Date] ), -7, DAY ),
  ALLEXCEPT ( FactCovidStats, FactCovidStats[DimCountryID] )
)
```

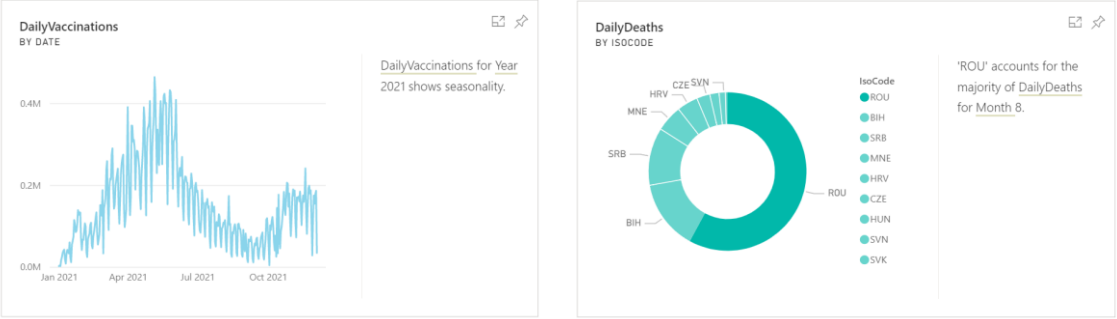
A riportok között megtalálhatóak a kiszámított mozgóátlagok és a hozzájuk tartozó tényleges értékek is egy idővonalon, illetve a beoltott társadalom százalékos értékben. Az országok között a térképen való kattintással lehet szűrni, ahol emellett a népességarányos halálozás látható, illetve egy csúszka segítségével a járvány kezdetétől tetszőleges időpontra lehet szűrni.

Ezek mind a céges belsős adatokkal kiegészítve jelenthetnek a jövőben majd nagy hozzáadott értéket, ha látjuk, hogy egy megfelelően nagy számú szervezetben az átlagoshoz képest mennyire tudunk jobban védekezni a különböző korlátozásokkal.



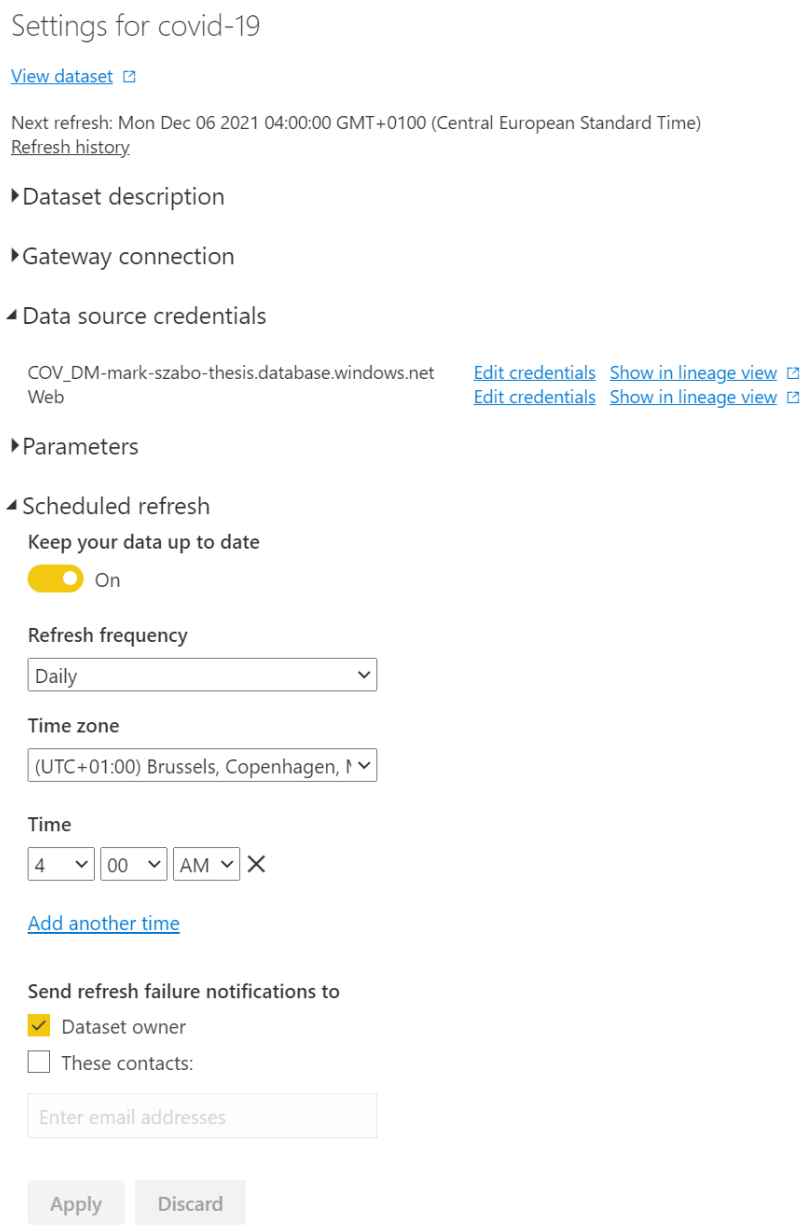
29. ábra: Power BI riport az adatpiacon található adatokból.

Az elkészült riportot a Power BI Pro szolgáltatásba publikáltam, hogy a döntéshozók egyszerűen egy böngészőből elérjék a kimutatásokat. Ennek járulékos nyeresége volt, hogy a Quick Insights segítségével a felhős szolgáltatás automatikusan legenerált néhány érdekes kimutatást.



30. ábra: Kivonat a Power BI Pro Quick Insights megoldása által automatikusan generált analitikákból.

Mindezek használhatósága merőben limitált lenne amennyiben az adathalmaz nem frissülne automatikusan. Szerencsére a Power BI Pro szolgáltatás megoldást nyújt erre, és az adatforrások újra-autentikálása után beállítottam egy napi frissítést hajnali 4-re, amikorra már az ETL folyamat biztosan lezárul. Amennyiben ez valamilyen hiba miatt mégsem sikerülne, a rendszer automatikusan egy emailben tájékoztat erről.



31. ábra: A Power BI Pro időzített automatikus frissítése az adatpiacról.

7. EREDMÉNYEK

A szakdolgozatomban elkészített Proof of Concept megoldás során arra a kérdésre próbáltam választ adni, hogy megvalósítható-e egy felhőalapú COVID-19 adatokat feldolgozó és analizáló rendszer, amely hatékonyan tudja segíteni a nagyvállalati döntéshozókat aktuális adatalapú döntések meghozatalában. Az elkészült rendszer alapján kijelenthető, hogy a jelenlegi technológiák megfelelően, megbízhatóan alkalmazhatóak egy ilyen támogató megoldás felállításához, illetve az ehhez szükséges komponensek rendelkezésre állnak.

A korábban felvezetett nagyvállalati vezetők így képessé válnak megalapozott adatalapú döntéseket hozni az üzleti teljesítményt jelentősen érintő korlátozások és lazítások terén, melyek potenciálisan a cég sikerét és akár munkavállalók ezreinek megélhetését határozzák meg.

Az elkészült rendszer segít megválaszolni az üzleti igényként megfogalmazott kérdéseket, mint például, hogy mikor ér el egy kritikus szintet a munkavállalók közti fertőzöttség valószínűsége, mikor érdemes azon munkavállalókat otthoni munkavégzésre kötelezni, akik munkájához a személyes találkozások nem feltétlenül szükségesek, vagy milyen szintű az átoltottság a vállalaton belül, illetve országosan vagy a régióban, ez alapján mikor érdemes lazítani az aktuális fertőzési számokat figyelembe véve. A rendszer a vállalati belsős chatboton keresztül képes őket a személyes kockázatelemzésükben is segíteni, illetve a belsős adatbázist integrálva az olyan kérdésekre is választ kaphatunk, hogy a különböző klinikai veszélyeztetettségű alkalmazottakat (kor vagy egészségügyi állapot miatt) érdemes-e és megoldható-e külön kezelni.

A POC sikeres kimenetele alapján megalapozott a rendszer bevezetése nagyvállalti környezetben. Az ehhez szükséges lépéseket a továbbfejlesztési lehetőségek alatt részletezem.

8. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

A járvány előrehaladtával a rendszer karbantartása és frissítése elengedhetetlen lesz, ahogy azt a természetéből adódóan állandóan mutálódó vírushelyzet és a rá adott védekezési intézkedések indokolják. A rendelkezésre álló adatok körének bővülésével egyre jobb analitikákat és predikciókat vagyunk képesek készíteni – de ehhez ezt a rendszernek is le kell követnie.

Új mezők bevezetésére lesz szükség az ETL folyamatba, ezek alapján új vizualizációk készítése a prezentációs rétegben és amennyiben az indokolt, a machine learning modellek újratanítása többféle adaton.

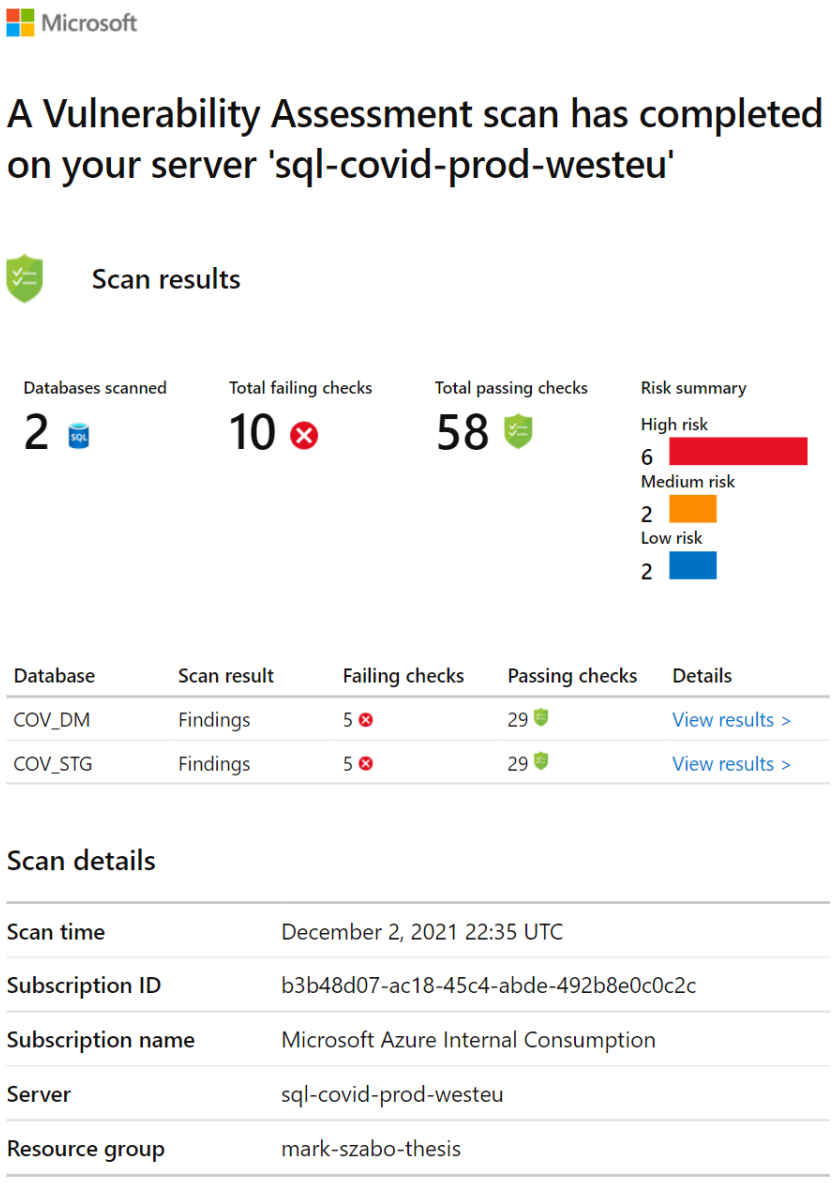
A technológiaválasztás egyik szempontja volt a továbbfejlesztésre való lehetőség, így tehát a választott komponensek mind könnyen lehetővé teszik a folyamatok módosítását. Az Azure Data Factory a legtöbb helyen automatikusan kezeli a séma módosulását és ennek megfelelően tölti át az adatokat a forrás rendszerekből az adatpiacokba. Az Azure Storage egy nem strukturált tároló, így az bármilyen változást a beérkező adatokban automatikusan kezel. Az SQL adatbázis esetében a sémát módosító DDL[4] parancsok segítségére lesz szükség. A Power BI ez alapján automatikusan frissíti az adatbázis struktúrát, az új vizualizációk létrehozása során ezeket azonnal tudjuk használni – értelemszerűen a vizualizációk létrehozásához manuális beavatkozásra lesz szükség. A machine learning és az API a kódok frissítésével lehetséges, természetesen itt is lehet a meglévő kódra építkezni, semmit sem kell teljesen újra elkészíteni egy frissítés során.

A rendszer legfőbb továbbfejlesztési célja a pozitív POC hatására a megoldás éles környezetbe helyezése. Ehhez a legfontosabb a nagyvállalati biztonsági sztenderdeknek való megfelelés. Elsősorban a komponensek virtuális hálózatba helyezésére lenne szükség, így ezek egy privát hálózaton, a publikus internettől elzárva kommunikálnának. A Power BI esetében egy virtual network data gateway-re lenne szükség, egy olyan komponensre, ami a virtuális hálózaton belül helyezkedik el és lehetővé teszi a Power BI Pro számára a biztonságos kommunikációt. A vállalati chatbot pedig egy site-to-site VPN segítségével csatlakozhatna a publikusan el nem érhető Function app-hoz.

Ezentúl az SQL szerveren az automatikus sérülékenységi vizsgálatot bekapcsolva ajánlásokat kaphatunk az adatbázisok biztonságosabbá tételére. Az alább található ábrán például egy ilyen vizsgálat eredménye látható, ahol a linkekre való kattintás után felhívja a figyelmünket, hogy az SQL auditálásnak bekapcsolva kellene lennie, a szerver szintű tűzfalszabályok túlzottan megengedőek és nincsenek követve, a *dbo* felhasználót nem szabadna általános operáció során használni és ajánlott lenne minden felhasználót követni,

[4] Data Definition Language, olyan SQL parancsok, melyek az adatbázis sémát definiálják, módosítják.

akinek hozzáférése van az adatbázishoz. De emellett azt is tudtunkra adja, hogy megfelelő hosszúságú aszimmetrikus kulcsok vannak használva az adatbázis titkosítására, a guest felhasználónak helyesen nincsenek jogai, vagy hogy az SQL Threat Detection (proaktív védelmi megoldás) be van kapcsolva.



32. ábra: Az SQL szerver által küldött email értesítés a potenciális biztonsági résekről.

Mindezeken túl, a megoldás fejlesztése közben – az oltások széleskörűen elérhetővé válásának hatására – felmerült üzleti igényként egy immunitás-követő rendszer készítése, amivel az alkalmazottak személyes kockázatértékelését segítenénk elő azzal, hogy meg tudja osztani másokkal, illetve követni tudja, hogy ki mikor és milyen vakcinával lett beoltva, így tehát mikor szerzett immunitást és az meddig tart. Természetesen a rendszer kizárólag önkéntes alapon működne.

9. ÖSSZEGZÉS

A 2020-as és 21-es év központi témája a COVID-19 világjárvány volt, mely világszerte több százmillió embert fertőzött meg, milliók haltak meg a komplikációiban. Közvetlen következményeként globális gazdasági visszaesés és recesszió következett be, amelynek következtében milliók vesztették el – átmenetileg vagy véglegesen – munkájukat, és esetleg otthonaikból is ki kellett költözniük, mivel nem tudták fizetni a jelzálog- vagy bérleti díjakat. Röviden, a vírus – amely túl kicsi ahhoz, hogy szabad szemmel is látható legyen – elég erős volt ahhoz, hogy újraértelmezze a munkánkat, az életünket, a gazdaságunk működését és alapvetően az életünk szinte minden szegmensét. Ebben a szakdolgozatban megkíséreltem elemezni, feldolgozni és vizualizálni a világjárvány statisztikáit, és feltárni a hatalmas adatmennyiségben rejtőző okokat, hogy megértsük az események alakulását. A legmodernebb felhőalapú számítástechnikai technológiákat használtam arra, hogy a bonyolult és számításigényes feladatokat a bolygó túlsó oldalán lévő óriási adatközpontokba helyezzem át, amelyeket olyan technológiai óriások tartanak fenn, mint a Microsoft, a Google vagy az Amazon. Hozzáadott értékként a legtöbb ilyen új felhőalapú megoldás sokkal fejlettebb, mint régi, lokálisan telepített változataik, így lehetőségem nyílt olyan funkciókat használni, mint felügyelet nélküli, automatikusan frissülő riportok és vizualizációk, vagy felhős ETL folyamok integrált gépi tanulási és serverless skálázódási képességekkel. A Proof of Concept eredményeként bebizonyítottam, hogy egy felhőalapú végponttól végpontig tartó adatfeldolgozási megoldás előnyös lehet egy multinacionális vállalat számára a döntéshozatalban egy olyan világjárványban, ahol gyors átfutási idők és közel valós idejű elemzések szükségesek. Következő lépésként a megoldás belső bevezetése tűzhető ki a specifikált biztonsági fejlesztésekkel és a belső munkavállalói statisztikákkal való integrációval.

10. SUMMARY

The COVID-19 disease outbreak and pandemic were 2020's and 2021's central theme, causing hundreds of millions worldwide to contract the virus, millions to die and as a direct consequence, a global economic downturn and recession, where millions lost their jobs – temporarily or permanently – and potentially were forced out of their homes as they couldn't keep up with their mortgage or rent payments. Briefly, the virus – too small to be seen with the naked eye – was powerful enough to redefine how we work, live our life, the way our economy works and basically nearly every segment of our life. In this thesis I attempted to analyze, process, and visualize the statistics of the pandemic and uncover the underlying reasons hidden in the massive amounts of data to understand the way events panned out. I used cutting-edge cloud computing technologies to offload complex and computationally heavy tasks to giant datacenters on the other side of the planet, maintained by tech giants, like Microsoft, Google, or Amazon. As an added value, most of these new cloud-based solutions are far superior to their legacy on-premises versions, so I could take advantage of these advanced features, including, but not limited to, unattended automatically updating dashboards and visualizations, integrated machine learning and serverless scaling capabilities in ETL flows. As a result of the Proof of Concept, I proved that a cloud-based end-to-end data processing solution can benefit a multinational enterprise in their decision making in a pandemic where quick turnaround times can be achieved and near real-time analytics are needed. As a next step the solution can be rolled out internally with the indicated security improvements and connections to the internal employee statistics.

11. KÖSZÖNETNYILVÁNÍTÁS

Köszönettel tartozom konzulensemnek, Dr. Fleiner Ritának a szakdolgozatom készítése során nyújtott támogatásáért és Retezi Richárdnak, a Microsoft Cloud Solution Architect-jének az Azure Data Platform terén nyújtott szakmai segítségéért. Ezentúl Kun Szabolcsnak, a Nokia IT Technical Engineer-jének az Azure Data Factory tudásának megosztásáért és végezetül barátaimnak, akik külső szemléletmódot és összehasonlítási alapot biztosítottak számomra.

12. RÖVIDÍTÉSEK

- AAD: Azure Active Directory
- AD: Active Directory
- ADF: Azure Data Factory
- ARM: Azure Resource Manager
- AWS: Amazon Web Services
- CAPEX: Capital Expenditure
- CDC: United States Centers for Disease Control and Prevention
- CI/CD: Continuous Integration / Continuous Delivery
- CLI: Command Line Interface
- COVID-19: Coronavirus Disease 2019
- CSP: Cloud Solution Provider
- CSV: Coma Separated Values
- DDL: Data Definition Language
- EA: Enterprise Agreement
- ECDC: European Centre for Disease Prevention and Control
- ETL: Extract-Transform-Load
- GCP: Google Cloud Platform
- GRS: geo-redundant storage
- GUI: Graphical User Interface
- GZRS: geo-zone-redundant storage
- HA: High Availability
- HDD: Hard Disk Drive
- HDFS: Hadoop Distributed File System
- HTML: HyperText Markup Language
- IaaS: Infrastructure-as-a-Service
- JSON: JavaScript Object Notation
- LRS: Locally-redundant storage
- LSP: Licensing Solution Provider
- LTS: Long Term Support
- ML: Machine Learning
- MSI: Managed System Identity
- NFS: Network File System
- OAS: OpenAPI Specification
- OPEX: Operating Expenditure
- PaaS: Platform-as-a-Service
- PoC: Proof of Concept
- RA-GRS: Read-Access Geo-Redundant

- RA-GZRS: Read-Access Geo-Zone-Redundant
- RBAC: Role-Based Access Control
- REST API: Representational State Transfer Application Programming Interface
- SaaS: Software-as-a-Service
- SLA: Service Level Agreement
- SMB: Server Message Block
- SQL: Structured Query Language
- SSD: Solid State Drive
- SSIS: SQL Server Integration Services
- SSO: Single Sign-On
- URL: Uniform Resource Locator
- UTC: Coordinated Universal Time
- VPN: Virtual Private Network
- WHO: World Health Organization
- ZRS: Zone-redundant storage

13. IRODALOMJEGYZÉK

[1] A. Kentikelenis, D. Gabor, I. Ortiz, T. Stubbs, M. McKee és D. Stuckler, „Softening the blow of the pandemic: will the International Monetary Fund and World Bank make things worse?," *The Lancet Global Health, 8*. kötet, 6. szám, pp. 758-759, 2020.

[2] P. Mell és T. Grance, „The NIST definition of cloud computing.," Gaithersburg, MD, 2011.

[3] „What Is Serverless Computing? | Serverless Definition | Cloudflare.," [Online]. Elérhető: https://www.cloudflare.com/learning/serverless/what-is-serverless/. [Hozzáférés dátuma: 2021. Május 10.].

[4] P. G. López, M. S. Artigas, G. . París, D. B. Pons, Á. R. Ollobarren és D. A. Pinto, „Comparison of Production Serverless Function Orchestration Systems.," *arXiv: Distributed, Parallel, and Cluster Computing*, 2018.

[5] Microsoft Corporation, „How do I choose a cloud service provider?," [Online]. Elérhető: https://azure.microsoft.com/en-us/overview/choosing-a-cloud-service-provider/. [Hozzáférés dátuma: 2021. Május 10.].

[6] Microsoft Corporation, „Azure global infrastructure," [Online]. Elérhető: https://azure.microsoft.com/en-us/global-infrastructure/. [Hozzáférés dátuma: 2021. November 14.].

[7] Microsoft Corporation, „Azure Data Factory, Data Integration in the Cloud," 2018. [Online]. Elérhető: https://azure.microsoft.com/mediahandler/files/resourcefiles/azure-data-factory-data-integration-in-the-cloud/Azure_Data_Factory_Data_Integration_in_the_Cloud.pdf.

[8] Microsoft Corporation, „Mapping data flows in Azure Data Factory," [Online]. Elérhető: https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-overview. [Hozzáférés dátuma: 2021. November 14.].

[9] Microsoft Corporation, „What is Power Query?," [Online]. Elérhető: https://docs.microsoft.com/en-us/power-query/power-query-what-is-power-query. [Hozzáférés dátuma: 2021. November 14.].

[10] Microsoft Corporation, „Azure SQL Documentation," 2020. [Online]. Elérhető: https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview.

[11] Nemzeti Népegészségügyi Központ, „Tájékoztató oldal a koronavírusról,” [Online]. Elérhető: https://koronavirus.gov.hu/. [Hozzáférés dátuma: 2021. Május 9.].

[12] Johns Hopkins University, "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University," Johns Hopkins University, [Online]. Elérhető: https://github.com/CSSEGISandData/COVID-19. [Hozzáférés dátuma: 2021. Május 9.].

[13] Our World in Data, „Data on COVID-19 (coronavirus) by Our World in Data,” [Online]. Elérhető: https://github.com/owid/covid-19-data/tree/master/public/data. [Hozzáférés dátuma: 2021. Május 9.].

[14] M. Glinz, „On non-functional requirements.,” *15th IEEE international requirements engineering conference (RE 2007),* pp. 21-26, 2007.

[15] Microsoft Corporation, „Service Level Agreements (SLA) for Online Services,” [Online]. Elérhető: https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services. [Hozzáférés dátuma: 2021. November 14.].

[16] R. C. Martin, Clean code: a handbook of agile software craftsmanship., Pearson Education, 2009.

[17] Microsoft Corporation, „Define your naming convention,” [Online]. Elérhető: https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming. [Hozzáférés dátuma: 2021. November 14.].

[18] Microsoft Corporation, "Azure Architecture Center," 2020. [Online]. Elérhető: https://docs.microsoft.com/en-us/azure/architecture/.

[19] Microsoft Corporation, „.NET and .NET Core Support Policy,” [Online]. Elérhető: https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core. [Hozzáférés dátuma: 2021. November 14.].

[20] Linux Foundation, „OpenAPI Specification v3.1.0,” 15 Február 2021. [Online]. Elérhető: https://spec.openapis.org/oas/latest.html.

14. ÁBRAJEGYZÉK