

Kodolányi János Egyetem

SZAKDOLGOZAT

BÓCZ-KEMÉNY VIKTOR BÉLA

ÜZEMMÉRNÖK-INFORMATIKUS ALAKÉPZÉSI SZAK

Budapest

2025

Kodolányi János Egyetem

Informatikai Tanszék

Az adminisztratív hatékonyság növelése

*A Microsoft Azure SDK alkalmazása az Azure DevOps felületén történő
adminisztratív feladatok egyszerűsítésére*

Konzulens: Dr. Pitlik László

Készítette: Bócz-Kemény Viktor Béla

Üzemmérnök-informatikus alapképzési szak

Budapest

2025

Tartalom

1.	Bevezetés.....	1
1.1	Probléma.....	1
1.2	Célok	1
1.3	Célcsoportok.....	2
1.4	Hasznosság	2
1.5	Motiváció	4
1.6	Dolgozat felépítése	4
1.6.1	Szoftvertervezés	5
1.6.2	Szoftvermegvalósítás	5
1.6.3	Szoftvertesztelés.....	5
1.6.4	A dolgozat szerkezetéről	5
2.	Szakirodalmi alapok.....	9
2.1	Felhőalapú szolgáltatások	11
2.2	Microsoft Azure.....	13
2.3	Magyarországi munkaerőköltség alakulása az IT iparágban	14
2.4	Szerverek és szerveroldali technológiák	16
2.5	Eseményfolyam-kezelő technológiák	17
2.6	Webkliensek és kliensoldali technológiák.....	18
2.7	Konténerizáció: üzemeltetés és karbantartás.....	19
2.7.1	Docker	21
2.7.2	Docker Compose	23
2.7.3	Kubernetes.....	23
2.8	Verziókezelő rendszerek.....	25
2.8.1	CVCS verziókezelő rendszerek.....	26
2.8.2	DVCS verziókezelő rendszerek.....	27
2.9	Statikus kódelemzés	28

2.10	Adatbázisok	28
2.11	Folyamatos Integráció, Folyamatos Szállítás	30
2.12	Tantárgyak szakdolgozati kapcsolata	30
2.12.1	Európai civilizáció és identitás	31
2.12.2	Szakterületi jogi ismeretek	31
2.12.3	Matematikai alapok	32
2.12.4	Adatszerkezetek és algoritmusok	33
2.12.5	Hálózatok és számítógép architektúrák	34
2.12.6	Operációs rendszerek	34
2.12.7	Programozási alapelvek és módszertanok	35
2.12.8	Rendszermodellezés	35
2.12.9	Emberi viselkedés és kommunikáció	35
2.12.10	Felhasználói interfészek és vizualizáció	35
2.12.11	Vezetési és vállalkozási ismeretek	35
2.12.12	Rendszertervezés	36
2.12.13	Programozás I., Programozás II., Programozás III.	36
2.12.14	Komplex társadalomtudományi ismeretek	36
2.12.15	Vállalati gazdaságtan	36
2.12.16	Adatbázisok I., Adatbázisok II.	37
2.12.17	Szoftverüzemeltetés	37
2.12.18	Szoftvertesztelés	37
2.12.19	Szoftverarchitektúrák	38
2.12.20	Innovatív információs és kommunikációs technológiák az IT-biztonság kapcsán	38
2.12.21	IT-biztonsági fejlesztések minőség- és projektmenedzsmentje	39
2.12.22	Mesterséges intelligenciák az IT biztonság területén	40
2.12.23	Tudásmenedzsment az IT biztonság területén	40

2.12.24	Az elektronika fizikai alapjai	40
2.12.25	Elektronikus áramkörök	40
3	Saját fejlesztés bemutatása	41
3.1	A szoftver tervezése	41
3.1.1	Szoftvert alkotó komponensek	42
3.1.2	Szoftverfejlesztést és üzembehelyezést támogató eszközök	45
3.1.3	Felhasznált technológiák összesítése	49
3.2	A szoftver megvalósítása.....	51
3.2.1	Webfelület	51
3.2.2	Szerveroldal.....	79
3.2.3	Eseményfolyam-kezelő	82
3.2.4	Adatbázis	86
3.2.5	Szoftverfejlesztést és üzembehelyezést támogató folyamatok.....	90
3.3	Tesztelés	103
3.3.1	Környezeti változó lekérdezés	104
3.3.2	Környezeti változó manipulálás	104
3.3.3	Secret lekérdezés	104
3.3.4	Secret manipulálás.....	104
3.3.5	Módosítások lekérdezése.....	105
3.3.6	Pipeline futtatás	105
3.3.7	Utolsó szoftververzió lekérdezés	106
3.3.8	Pull request létrehozás.....	106
3.3.9	Pull request lekérdezés	106
3.3.10	Tesztelési összegzés	106
4.	Vita	106
4.1	Magasabb szintű kódminőség és biztosabb működés	106
4.2	Egyszerűbb technológiai portfólió	107

4.3	Nyelvi korlátok megszüntetése, lokalizálás	107
5.	Konklúzió	108
6.	Összegzés	109
7.	Jövőkép.....	110
8.	MELLÉKLETEK	111
	Irodalomjegyzék.....	112
	Ábrajegyzék	117
	Táblázatjegyzék.....	122
	Rövidítések jegyzéke.....	123
	KIVONAT	124
	ABSTRACT	126
	Introduction	127
	Problem	127
	Goal	128
	Target audiences	128
	Usefulness	129
	Structure of thesis.....	130
	Software design	131
	Software implementation	131
	Software testing.....	131
	Detailed structure of the thesis	131
	Discussion	135
	Conclusion.....	136
	Summary	137
	Vision	138
	BlueBird IT Salary Guide.....	140
	Copilot támogatás.....	142

ChatGPT támogatás.....	147
Library API swagger ábra formátumban	150
API swagger ábra formátumban.....	152
Tesztesetek.....	154
Tesztelés eredménye.....	169
Példaeset környezeti változó módosítására	193

1. Bevezetés

1.1 Probléma

A szakdolgozati munkám a környezeti változók kezelésének komplexitására jelent megoldást, amely Microsoft Azure környezetben történő szoftverkitelepítéshez szükséges.

A környezeti változó alatt azokra a szoftver futtatásához elengedhetetlen metaadatokra gondolok, amelyek meghatározzák és lehetővé teszik a szoftvertermék helyes működését. Ilyen környezeti változó például az adatbáziskapcsolathoz szükséges jelszó vagy az adatbázisszerver IP címe. A változók értékét az Azure DevOps (Development and Operations) segítségével át lehet adni akár egy Kubernetes manifest YAML (YAML Ain't Markup Language / Yet Another Markup Language) fájlban, akár CLI (Command Line Interface) alapú utasítás paraméterének is, a felhasználás természetét a szoftverkezelő határozza meg.

Ezeknek a változóknak a számossága vállalati környezetben (például microservice architektúrával, szeparált környezetekkel) könnyen elérhetik a tízezres, vagy akár százezres nagyságrendet is.

A környezeti változók kezelésének komplexitása nagyban függ a kialakított kezelési stratégiától, amelyet a szoftverfejlesztő (vállalat, individuum) választott. Két fő oka lehet a magas kognitív komplexitásnak: az egyik a hibásan (tehát duplikációra és redundanciára hajlamos) kialakított környezeti változók kezelési stratégiájából, míg a másik az adott vállalati szoftvertermék növekedéséből (tehát sikerességéből, új fejlesztési igények létrejöttéből) fakad. Hozzáteszem, ez szintén egy hibásan kialakított kezelési stratégia következménye.

Mindemellett megállapítható bizonyos metrikák (módosítás sebessége, módosításhoz szükséges műveletek számossága) alapján, hogy az Azure DevOps felületén a környezeti változók módosítása nehezekebb, mint amilyen egy erre a problémakörre megoldást kínálni tudó szoftvernél lehetséges lenne. Ennek részletezését a következő fejezetekben végzem el.

1.2 Célok

A szakdolgozati munkám fő célja, hogy a felhasználó képes legyen az Azure DevOps felületén szükséges változómódosításokat minél egyszerűbben és minél nagyobb számosságban

elvégezni. A felhasznált Azure SDK (Software Development Kit) azonban további, a környezeti változók manipulálásán és lekérdezésén kívüli funkcionalitást is kínál, aminek segítségével tovább automatizálhatóak az üzemeltetéssel és szoftverfejlesztéssel járó feladatok. Ennek köszönhetően a szoftvertermék tulajdonosa költséget tud optimalizálni, a fejlesztett eszköz felhasználói pedig gyorsabban, egyszerűbben, nagyobb biztonsággal és kisebb hibázási lehetőséggel képesek ellátni a feladataikat. A potenciális felhasználók természetéről a következő fejezetben térek ki.

1.3 Célcsoportok

„Sok más technológiai munkakörhöz hasonlóan a DevOps mérnöki munkakör értelmezése is szubjektív, annak jelentése, szükséges készségei és alapvető felelősségei valószínűleg különbözni fognak vállalatról vállalatra. A valós DevOps mérnöki szerepkör függ az adott szervezet céljaitól, növekedési szakaszától, iparágától és a számítástechnika előrehaladásától.”
(Harrison, 2021)

Ahogy erről ebben a szakmai cikkben is olvasható (Harrison, 2021), a szoftverüzemeltetésre és szoftverkarbantartásra vonatkozó feladatok eltérően lehetnek elosztva a munkakörök között egy-egy informatikai vállalatnál. Ennek következtében célcsoportként azokat a munkaköröket adom meg, amelyek betöltőinek elvi lehetősége van arra, hogy az előbb említett felelősségi körökkel rendelkezzenek (üzemeltetés és karbantartás) és Microsoft által biztosított felhőalapú ökoszisztémában a munkájukhoz az Azure DevOps felületét használják. Ilyenek lehetnek például:

- Szoftverfejlesztők
- Rendszeradminisztrátorok
- Rendszerkarbantartók
- DevOps mérnökök

Tehát a fent leírtak alapján kijelenthető, hogy a szakdolgozati problémafelvetés alapján elkészített szoftver azoknak az informatikai szakembereknek nyújthat segítséget, akik Microsoft ökoszisztémában használnak felhőalapú szolgáltatásokat és felhasználói az Azure DevOps felületének.

1.4 Hasznosság

A dolgozat hasznosságát a szakdolgozathoz csatolt 14. melléklet segítségével támasztom alá.

Az említett mellékletben egy szimulált példán keresztül hatvanhétszeres időbeli különbséget realizáltam a manuális és a szakdolgozati projekt által elvégzett művelet között. A szükséges módosítást elvégeztem egy variable group-on, a mért idő alapján a művelet teljeskörű elvégzése manuális módosítások esetén 3502 másodpercet, míg az automatizált módosítások 52,46 másodpercet vettek igénybe (Lásd 14. melléklet és 1. ábra).



1. ábra: Szimulált hasznossági példa végeredményének diagrambéli reprezentációja (saját szerkesztés)

Amint az a Bluebird IT staffing vállalat által készített fizetési kimutatásában is olvasható, „*egy kezdő rendszerüzemeltető vagy DevOps mérnök is bruttó 650 és 800 ezer forint között keresett Magyarországon átlagosan a 2024-es évre vonatkozóan*” (Bluebird International, 2025, 7. melléklet). Ezekből a bruttó értékekből is megállapíthatóvá válik, hogy a példában említett művelet automatizált elvégzése is több ezer forint megtakarítást jelent a szoftvert megvásárló vállalat számára.

Amennyiben feltételezzük, hogy akár napi több, a példában említett művelethez hasonló módosítást is el kell végezni a környezeti változókon, a szoftver megtérülése könnyen bizonyíthatóvá válik a fenti példán keresztül.

Amint az a Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen által készített tanulmányban is olvasható, „*az informatikában megjelenő munkahelyi kiegészítő okai között szerepelnek a repetitív feladatok és a munkaterhelés is*” (Tulili et al., 2023). A szakdolgozati projekt vállalati szintű felhasználásával ezen indikátorok hatásai is szignifikánsan csökkenthetővé válnak, hiszen a munkaterhelés és a repetitív feladatok is redukálhatóvá válnak a szoftver felhasználásával.

1.5 Motiváció

Jelenleg szerveroldali fejlesztőként dolgozom Microsoft ökoszisztémában és még gyakornokként rengeteget kellett környezeti változók módosításával foglalkoznom. Azt a feladatot kaptam a feletteseimtől, hogy a környezeti változók módosítására keressek valamiféle automatizáló megoldást, annak érdekében, hogy a gyakornokainkat tehermentesíteni tudjuk.

Ennek okán létre is hoztam egy csapaton belül használt apró szoftvert, ami CLI alapon képes a környezeti változókat manipulálni és lekérdezni. Mindazonáltal éreztem, hogy ebben a témában nagyobb lehetőségek is rejlenek, így megírtam a saját, széleskörűbb megoldásokat kínáló szoftveremet, amelyet ebben a dolgozatban kívánok bemutatni.

1.6 Dolgozat felépítése

A dolgozat felépítésének bemutatását ebben a fejezetben teszem meg.

Az [1. Bevezetés](#) főfejezetet követően a [2. Szakirodalmi alapok](#) főfejezet következik, amelyben a szakdolgozati téma kapcsán felmerülő szakirodalmi háttér bemutatását végzem el. Emellett külön megvizsgálom, hogy a választott téma és megvalósítás tud-e, és ha igen, miben tud kapcsolódni a képzési mintatantervben szereplő tantárgyakkal (Lásd [2.9 Tantárgyak szakdolgozati kapcsolata](#)).

A [3. Saját fejlesztés bemutatása](#) főfejezetben bemutatom a probléma megoldására készítendő szoftver tervezési és elkészítési folyamatát, illetve magát a kész produktumot is prezentálom.

Az elkészített projektet a következő fejezetek mentén fogom bemutatni: szoftvertervezés (Lásd [3.1 A szoftver tervezése](#)), szoftvermegvalósítás (Lásd [3.2 A szoftver megvalósítása](#)) és szoftvertesztelés (Lásd [3.3 Tesztelés](#)).

A magam következtetéseit és saját munkámmal kapcsolatos kritikai észrevételeimet (demonstrálva az önreflexióra való hajlandóságot) a [4. Vita](#) és az [5. Konklúzió](#) fejezetekben írom le.

A [6. Összegzés](#) fejezetben a választott téma és elkészített munka szempontjából legértékesebb megállapításokat, következtetéseket fogom összegezni.

A [7. Jövőkép](#) fejezetben azok a fejlesztési lehetőségek szerepelnek, melyek végrehajtása belátható időn belül realitássá válhatnak, amennyiben erre megrendelői akarat jelentkezik a jövőben.

1.6.1 Szoftvertervezés

A [3.1 A szoftver tervezése](#) fejezetben bemutatom, hogy milyen technológiák, könyvtárak és architekturális megközelítések felhasználásával, illetve milyen platformra készítettem el az alkalmazást. Ebben a szegmensben kifejtésre kerül az is, hogy milyen érvek mentén döntöttem a felsorolt szoftvereszközök és megközelítések használata mellett.

A fejezetben kitérek a választott Folyamatos Integráció, Folyamatos Szállítás, tehát CI/CD (Continuous Integration/Continuous Deployment) stratégiára, illetve ezen stratégia összetevőire is.

1.6.2 Szoftvermegvalósítás

A [3.2 A szoftver megvalósítása](#) fejezetben bemutatom és értelmezem a szoftver fejlesztése során elkészült dokumentációt. A fejezetben ábrákkal és képkivágásokkal fogom szemléltetni az elkészült munka egyes elemeit, a különböző funkcionálisokat és a fejlesztési munkálatok stációit.

Továbbá bemutatom, hogy a választott CI/CD eszközök milyen tényleges hozzáadott értéket képviselnek a megvalósított szoftver vonatkozásában.

1.6.3 Szoftvertesztelés

A [3.3 Tesztelés](#) fejezetben bemutatom, hogy milyen technológiákat és módszertanokat használtam az elkészült szoftver tesztelése során. Ebbe beleértendőek a statikus kódelemző technológiák, a felhasznált keretrendszer által biztosított tesztelési könyvtárak, illetve a CI/CD által biztosított tesztlépések is.

A szoftver kapcsán manuális tesztelésre is van lehetőség, így egy elkészült tesztforgatókönyv és annak végrehajtását is be fogom mutatni.

1.6.4 A dolgozat szerkezetéről

Ebben az alfejezetben a szakdolgozat tartalmi és formai elemeire fogok kitérni, illetve bemutatom, hogy pontosan milyen elvárásai voltak a megrendelőnek a szakdolgozat alapjait jelentő szoftverrel szemben.

Ezen alfejezet és a [3. Saját fejlesztés bemutatása](#) főfejezet vizsgálatával a megrendelő és a szakdolgozat mindenkor olvasója képessé válik annak eldöntésére, hogy az elkészített munka megfelel-e, és ha igen, milyen minőségben felel meg a vele szemben támasztott követelményeknek.

1.6.4.1 Megrendelői elvárások

A megrendelői elvárásokat ebben az alfejezetben fogom összesíteni és erre az összesítésre fogok hivatkozni a későbbi fejezetekben.

A termék megrendelőjének fő céljai:

- a felhasználó képes legyen az Azure DevOps felületén szükséges változómódosításokat minél egyszerűbben és minél nagyobb számosságban elvégezni
- további cél a megrendelő elvárásai szegmensben megfogalmazott egyéb fejlesztési igények leszállítása, amivel a fő cél indítékának (manuális erőforrás felszabadítása automatizációval) még hatékonyabb elérésére törekszem

A termék megrendelője elvárja a következőket:

- Szoftver zárt forráskódjának megírása, verziókezelő használatával
- Forráskódból fordított állományok előállítása és leszállítása (lehetőleg konténerizált megoldással), amelyek bármely operációs rendszeren futtathatóak
- A szoftver rendelkezzen webes felhasználói interfésszel, modern keretrendszer használatával (JavaScript keretrendszer a preferált)
- A szoftvernek kizárólag angol nyelven szabad megjelenítenie a webfelületet (ennek következtében angolnyelvű oldalakat fogok prezentálni a saját munka bemutatása főfejezetben)
- A szoftver microservice architektúrát valósítson meg (.NET ökoszisztéma preferált)
- IT-biztonsági szempontok érvényesítése
- Tesztforgatókönyv és teszteredmények bemutatása és átadása a fejlesztés átvételének követelménye
- Funkcionális elvárások:
 - o Azure környezetben tapasztalt adminisztratív feladatok automatizálása és az ezzel kapcsolatos használt erőforrások minimalizálása
 - o Azure DevOps variable group-ban tárolt környezeti változók manipulálása és lekérdezése jelentős volumenű érintettségben egyszerű utasításokkal
 - o Azure DevOps-ban tárolt secretok lekérdezése, törlése és reaktiválása (recover)
 - o Azure DevOps-ban létrehozott key vault teljes tartalmának átmásolása egy másik key vault-ba (gyors adatfeltöltés)
 - o Azure DevOps-ban végzett, környezeti változókra, secretokra és key vault másolásokra vonatkozó módosítások naplózása

- Táblázatos megjelenítése a következő, módosítással kapcsolatos adatoknak:
 - Környezeti változó módosítás esetén:
 - Azure organizáció
 - Azure projekt
 - Felhasználó
 - Variable group regex parancs
 - Környezeti változó neve
 - Módosítás típusa
 - Dátum
 - Key vault másolás esetén:
 - Adatforrás Key vault
 - Cél Key vault
 - Felhasználó
 - Dátum
 - Azure Secret módosítás esetén:
 - Key vault neve
 - Felhasználó
 - Secret név regex parancs
 - Módosítás típusa
 - Dátum
- Azure DevOps-ban létrehozott pull request-ek lekérdezése
- Egyszerűsített Azure DevOps pull request létrehozása
- Azure DevOps build pipeline-ok egyszerűsített futtatása
- Azure DevOps-ban létrehozott szoftververziók egyszerűsített lekérdezése (utolsó verzió lekérdezése jelentős volumenben)
- Az elkészült szoftver törekedjen a következő eszközök használatára, amennyiben lehetséges:
 - széles fejlesztői közösséggel rendelkező
 - nyílt forráskódú
 - ingyenesen elérhető

A fenti megrendelői elvárások listából következik, hogy az alábbiak nem voltak elvárások a szoftverrel kapcsolatban:

- Szoftverüzemeltetés dedikált környezeten/környezeteken
- Konténermenedzsment
- Kódminőség riportálás, kódanalízis

1.6.4.2 Kutatási megkötések, avagy a szakdolgozatból tudatosan kihagyott elemek

Ebben az alfejezetben felsorolásszinten rögzítem, hogy a szakdolgozat milyen aspektusokat nem szándékozik taglalni a témát illetően.

A szakdolgozatnak nem témája (alapvetően terjedelmi okokból):

- az Azure DevOps részletes bemutatása és funkcionalitásának mélyenszántó ismertetése (csak amennyi a dolgozat megértéséhez feltétlenül szükséges)
- más, a piacon elérhető felhőszolgáltató részletes bemutatása és funkcionalitásának mélyenszántó ismertetése

A szakdolgozatból tudatosan kihagyott, így a dolgozattal szemben nem elvárható:

- Azure által biztosított IT-biztonsági szolgáltatásoknak a kiterjesztése, bővítése, megerősítése (a meglévő IT-biztonsági szintet természetesen nem kompromittálva, beépített biztonsági lépések felhasználásának megkövetelése, például: personal access token és egyéb hitelesítő adat)

Ennek tükrében a Microsoft Azure funkcionalitásainak ismertetését csak a feltétlenül szükséges mértékig fogom elvégezni, azokra a kérdésekre nem fogok magyarázatot keresni, hogy például a környezeti változókhoz tartozó funkcionalitásoknál miért nem szükséges kliens azonosítót (client id) megadni.

1.6.4.3 Formai megoldások a szakdolgozatban

A szakdolgozat követ bizonyos formai mintázatokat a szakdolgozati téma és megírt szoftver alaposabb megértése érdekében. Ilyenek a következők:

- Egyértelműen lehatárolt és strukturált fejezetek:
 - o Bevezetés főfejezet strukturálva és figyelemfelkeltően megírva, fenntartva az olvasóban a kíváncsiságot a dolgozattal kapcsolatosan.
 - o Élesen szeparálva a szakirodalmi háttérrel a dolgozat többi elemétől, kapcsolat legfeljebb a fejezetreferenciák esetén fellelhetőek (így demonstrálva a szükséges szakirodalmi kapcsolatot az elvégzett munkával)

- Élesen szeparálva és részletesen kibontva a szakdolgozati téma és szakirodalmi háttér alapján elvégzett fejlesztési munkát. A szakirodalmi háttérre való reflektálás elengedhetetlen része a munka bemutatásának, így a vonatkozó fejezetreferenciák elhelyezésre kerültek.
- Végkövetkeztetések (vita és összegzés szegmensekkel kiegészítve) és potenciális jövőképek ábrázolása a saját munka bemutatása után
- Részletes és gyakori vizuális reprezentációk:
 - Ábrák
 - Táblázatok
- Részletes tesztforgatókönyv, demonstrálva a szoftver helyes működését
- A dolgozat magyar nyelvű
- A dolgozat hipertextes ugrópontokat tartalmaz a digitális Olvasó támogatására
- Az idézetek „*dőltbetűvel és idézőjelben*” azonnali forrásmegjelölés mellett kerültek be a dolgozatba
- A dolgozat további formázásuktól (vastagon szedett szövegek, aláhúzások, lábjegyzetek) mentes
- A csatolt mellékletek segítik az Olvasót a dolgozat teljeskörű megértésében

2. Szakirodalmi alapok

Ebben a fejezetben bemutatom a szakdolgozati munkám szakirodalmi vonatkozásait, ezzel alátámasztva a dolgozat tudományos igényű megalapozottságát.

A [2.1 Felhőalapú szolgáltatások](#) fejezetben röviden áttekintem a felhőalapú számítástechnikát és a technológiával járó előnyöket és hátrányokat, illetve a felhőalapú számítástechnikában élen járó nagy informatikai vállalatok felhőalapú megoldásaira is ki fogok térni.

A [2.2 Microsoft Azure](#) című fejezetben bemutatom a Microsoft által biztosított felhőalapú szolgáltatásokat, ugyanis a szakdolgozat fő profilja az említett cég szolgáltatásaival összefüggő adminisztratív feladatok redukálása, racionalizálása.

Mivel az előbb említett feladatredukció anyagi ösztönzővel is együtt járhat az általam elkészített szoftvert használó vállalat számára (erre az [1.4 Hasznosság](#) dolgozatszegmensben rá is mutattam), így érdemes kitérni a szakdolgozat keretében az elmúlt évek magyarországi munkaerőköltségeinek alakulására az informatika területén. Ezt a [2.3 Magyarországi munkaerőköltség alakulása az IT iparágban](#) fejezetben fogom megtenni.

Általánosan bemutatásra kerülnek a szerverek és a modern szerveroldali technológiák a [2.4 Szerverek és szerveroldali technológiák](#) alfejezetben.

Tekintettel arra, hogy megrendelői igény mutatkozott webes felületre (Lásd [1.6.4.1 Megrendelői elvárások](#)) is az [1.1 Probléma](#) fejezetben taglaltakra megoldást kínáló szoftver kapcsán, így szükségesnek tartom, hogy kitérjek a webkliensekre és a modern webes technológiákról is. Ezt a [2.5 Webkliensek és kliensoldali technológiák](#) alfejezetben teszem meg.

Ahogy arról az [1.6.4 A dolgozat szerkezetéről](#) fejezetben is írtam, a megrendelővel kötött megállapodás értelmében a szoftver elkészítőjének (tehát nekem) a következő köteleességeket kell teljesítenie:

- szoftver (elvárt magas szakmai minőségben történő) megírása
- forráskódból fordított futtatható állományok előállítása és leszállítása

Ennek következtében az üzemeltetés és karbantartás felelősségi körei a megrendelővel kötött megállapodás szerint a megrendelőhöz tartoznak. Ettől eltekintve meggyőződésem, hogy a szoftver piaci térnyerésének érdekében fontos ezeken a területeken is megoldással szolgálnia az elkészült munkának. Ezen elhatározás okán az [2.6 Konténerizáció: üzemeltetés, karbantartás](#) fejezetben általános érvénnyel bemutatom a konténerizációval kapcsolatos szoftverüzemeltetés és szoftverkarbantartás szakirodalmi háttereit.

A [2.7 Verziókezelő rendszerek](#) szegmensben kitérek a verziókezelés általános jellemzőire és előnyeire, valamint bemutatok 1-2 szabadon elérhető verziókezelő rendszert is.

Tekintve, hogy a megrendelői elvárások (Lásd [1.6.4.1 Megrendelői elvárások](#)) következtében adatperzisztenciában is érintett a szakdolgozati munka alapját szolgáltató szoftver, így az [2.8 Adatbázisok](#) fejezetben kitérek arra, hogy miért is lehet fontos egy szoftver vonatkozásában az adatperzisztencia. Emellett bemutatom a különböző adatbázistípusokat és az adatbázisokkal történő kommunikációt lehetővé tevő általános érvényű lekérdezőnyelvet, az SQL-t (Structured Query Language) is.

A [2.9 Tantárgyak szakdolgozati kapcsolata](#) fejezetben az egyetemi képzés mintatantervében szereplő tantárgyak vonatkozásában is megvizsgálom a szakdolgozati munkát, keresve a kapcsolódási pontokat az adott tantárgy és a választott téma között. Ezzel adva egy interdiszciplinárisabb megközelítést a szakdolgozatnak, így törekedve a tágabb értelmezési keret elérésére.

2.1 Felhőalapú szolgáltatások

A felhőalapú szolgáltatások áttekintése előtt fontosnak tartom bemutatni és a szakirodalmi háttér alapján áttekinteni, hogy pontosan mit is értünk „felhő” alatt, vagy pontosabb: mit értünk felhőalapú számítástechnika alatt.

A felhőalapú számítástechnika fogalmának sokáig nem volt elfogadott definíciója, ahogyan erről a számítási felhő jövőjéről értekező szakirodalomban is olvastam:

„A cloud computing fogalma alatt nagyvonalakban a számítógépes erőforrások (hardver, szoftver-platform, adatbázis-szerver, felhasználói szoftvercsomag) interneten keresztüli felhasználását és azoknak, mintegy szolgáltatás-szintű megfizettetését érti az informatikával és számítástechnikával foglalkozók nagy többsége. Hogy mégis mennyire kiforratlan még ez a paradigma (fogalom, modell), álljon itt a 2008-as Cloud Summit Executive 2008 konferencia elnökének a bevezetőben tett szellemes megjegyzése (story-ja), miszerint húsz embert kért meg, hogy definiálja a cloud computing-t, és huszonkét különböző választ kapott.” (Imre & Ármin, 2014)

A hivatkozott szövegrészlet alapján megállapítható, hogy ezen új számítástechnikai paradigmával kapcsolatban nem csak az átlagembernek van fogalmi irányvesztettsége (jelesül a mindennapokban hallható „Fellőtte a receptet a felhőbe a gyógyszerár.” jellegű mondatokból érezhető bizonytalanság a mondat tárgyát képező jelenség kapcsán), hanem a szakma jeles képviselőinél is érezhető volt egy ehhez hasonló fogalmi zavar az elmúlt évtizedekben. Az alfejezet további hivatkozásai alapján (amelyek később megírt tanulmányokból származnak) megfigyelhető egyfajta értelmezési letisztulás.

Ugyanebben a dolgozatban meghatároznak viszont egy olyan definíciót, amit könnyen megérthetőnek és definitívnek találtam:

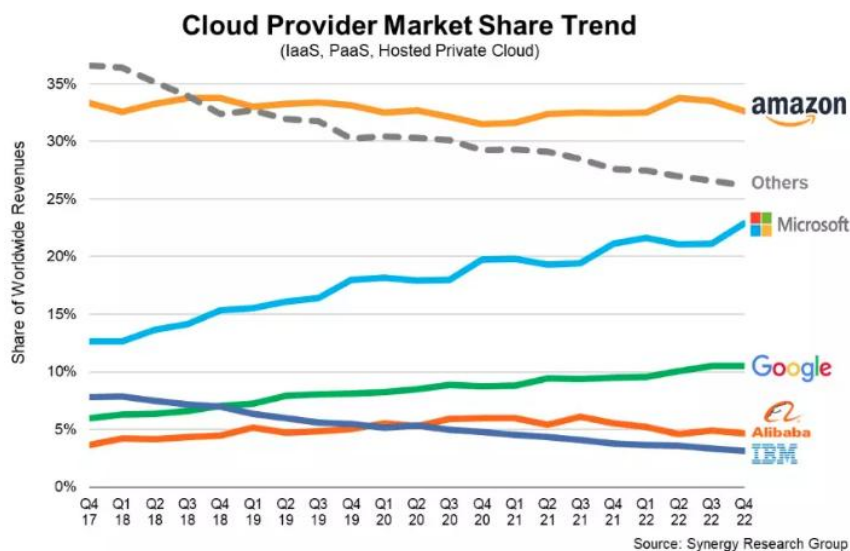
„A cloud computing olyan adatfeldolgozási stílus jelent, amelyben masszívan skálázható (hangolható) számítástechnikai lehetőségeket biztosítanak szolgáltatásként interneten keresztül külső fel használók számára.” (Imre & Ármin, 2014)

Rubóczki Edit Szilvia a doktori disszertációjában arra a következtetésre jut, hogy *„Az évtized legforradalmibb és talán legnépszerűbb IT infrastruktúra megoldása a felhő technológia.”* (Rubóczki Edit Szilvia, 2019)

Rubóczki Edit Szilvia disszertációjában én már érzékelek egyfajta értelmezési letisztulást a felhőalapú technológiák kapcsán a Cloud Summit Executive 2008 konferencián elhangzott megállapításokhoz képest.

A felhőalapú számítástechnikában élen járnak az alábbi nagy informatikai techvállalatok (Lásd 2. ábra), például:

- Microsoft Corporation (erről a vállalatról a [2.2 Microsoft Azure](#) fejezetben részletesebben is írok a szakdolgozati érintettsége miatt), a Microsoft Azure-t biztosítja felhőszolgáltatásként
- Amazon Inc, az AWS (Amazon Web Services) szolgáltatást biztosítja, mint felhőalapú megoldást
- Google LLC, a Google Cloud Platform tulajdonosa és üzemeltetője.



2. ábra: A három fő felhőszolgáltató vállalat piaci megoszlása (forrás: <https://www.cloudzero.com/blog/cloud-service-providers>)

A 2. ábra vonatkozásában érdemes rögzíteni, hogy az X tengely a globális bevételekből származó részesedés százalékos arányát mutatja, míg az Y tengely az évet, illetve az adott év negyedét jelenti (17 Q4 → 2017 utolsó negyedéve). A vonalakhoz (vonlak: egy tech vállalat mindenkori eredményeit reprezentálja az X és Y tengely vonatkozásában) tartozó színek pedig a különböző vállalatokhoz tartozó értékek hatékonyabb vizuális elválasztását szolgálják.

Az olvasott szakirodalmi összefoglalók alapján véleményem szerint könnyen lehet érvelni amellet, hogy egy vállalatnak miért érdemes lecserélnie a lokálisan futtatott szervereit egy „pay as you go” felhőszolgáltatásra. Amennyiben egy vállalat lerázhhatja magáról a meglévő

szervereinek üzemeltetését és skálázásának körülményességét adott esetben a meglévő költségének töredékéért, miért ne állna át a „felhőre”?

A hivatkozott szakirodalmi kitekintések alapján könnyen arra a következtetésre lehet jutni, hogy a felhőalapú szolgáltatások iránt a továbbiakban is igen nagy (és növekvő) lesz a kereslet, ugyanis „*a cloud computing jól skálázható, igény szerinti, webes hozzáférésű számítástechnikai erőforrások biztosítása költséghatékony módon, rugalmas igényváltozások, szabványosítás, modularizáció és virtualizáció megvalósításával*” (Imre & Ármin, 2014).

2.2 Microsoft Azure

A Microsoft az Azure szolgáltatáscsomaggal biztosítja a maga felhőalapú megoldásait, amelynek segítségével bizonyos kimutatások szerint mostanra globális piacvezető szerepet is elért. „*A Microsoft jelenleg is az egyik legnagyobb szereplő a felhőalapú szolgáltatások piacán, becslések szerint globálisan a 60-70 százalékát fedik le, és ehhez jön még hozzá, hogy övé az Xbox és a Windows, az Azure révén pedig általánosságban is a felhőalapú számítástechnika éllovasának számít.*” (András Mizsur, 2023)

Az Azure szolgáltatások bemutatására és használatára a Microsoft létrehozott több tutorial-t és guide-ot is (<https://azure.microsoft.com/en-us/get-started/?msocid=1d6c3332c4bd67993bb52663c5a0666a>), illetve különböző edukációs platformokon is van lehetőség elsajátítani az Azure platform használatához szükséges tudást (<https://www.udemy.com/courses/search/?src=ukw&q=microsoft+azure>). Ahogyan a két hivatkozott weboldalon is látható, díjköteles és ingyenes tartalmak is áttekinthetők.

A Microsoft hivatalos weboldalán megismerhetők az Azure biztosította lehetőségek tárháza (<https://azure.microsoft.com/en-us>). A teljesség igénye nélkül, áttekintés jelleggel én is ismertetnék néhány elérhető funkcionalitást:

- Azure Portal: Felhőalapú szolgáltatások erőforrásmenedzsmentjéért felelős webplatform (<https://learn.microsoft.com/en-us/azure/azure-portal/azure-portal-overview>)
- Azure Functions: Szervermentes felhőalapú függvények fejlesztése és üzemeltetése (<https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp>)
- Azure SQL: Felhőalapú adatbázisszerver szolgáltatás (<https://azure.microsoft.com/en-us/products/azure-sql>)

- Azure AI Services: Felhőalapú mesterséges intelligencia szolgáltatások beépített kis nyelvi modellekkel (<https://azure.microsoft.com/en-us/products/ai-services>)

2.3 Magyarországi munkaerőköltség alakulása az IT iparágban

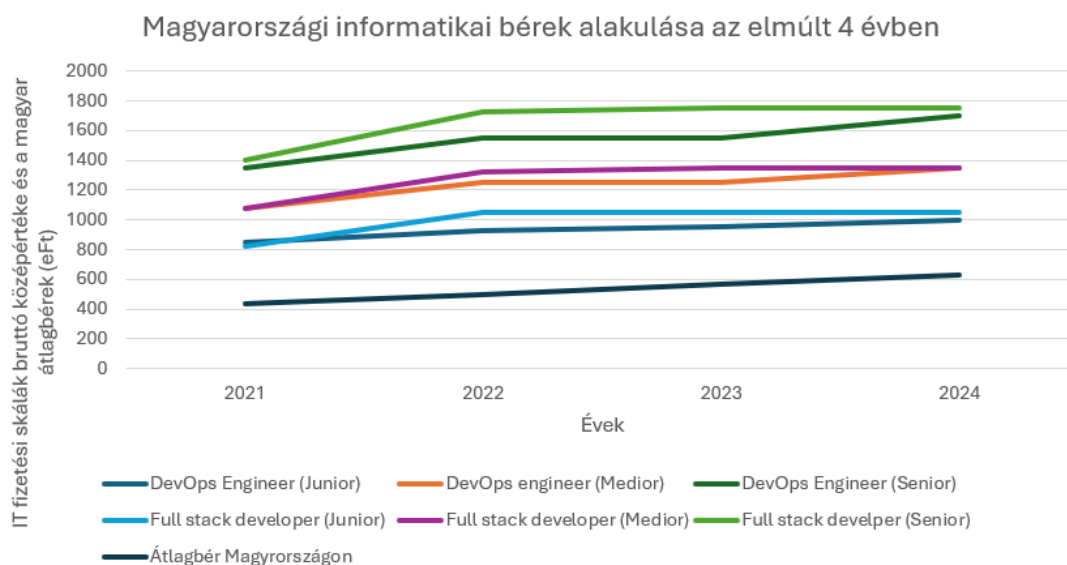
Mivel az [1.4 Hasznosság](#) című fejezetben utaltam a szakdolgozat alapjául szolgáló szoftver munkaerőköltségre gyakorolt lehetséges reduktív hatásaira is, ennek okán fontosnak tartottam kitérni a magyarországi informatikai munkaerőköltségek alakulására is a dolgozatban.

„Milyenek a fizetések az IT szektorban?”

Blogunkban időszakonként aktualizáljuk a munkakörönkénti és tapasztalati szint szerinti bruttó béreket.

Az általunk megadott fizetések nem reprezentatív felmérés eredményeként jöttek létre, hanem az alapján, amit nap mint nap tapasztalunk. Ismerjük az IT szakemberek által elvárt béreket és a megrendelőink által kialakított bérsávokat. Ezekből az adatokból dolgozunk és azokat a béreket tartjuk a legfontosabbnak, amelyben megrendelőink megállapodnak jelöltjeinkkel.” (Bluebird International, 2025)

A bevezetésben kifejtettem, hogy melyek a dolgozat célcsoportjai, így ebből a szakmalistából találomra kiválasztottam a BlueBird IT Salary Guide kiadványaiból (<https://bluebird.hu>) a DevOps mérnök és a Full stack fejlesztő szakmákra vonatkozó bruttó fizetési értékeket a szemléltetéshez. Az összefüggések mélyebb megértése érdekében a Központi Statisztikai Hivatal adatai alapján feltüntettem az adott évre vonatkozó magyarországi bruttó átlagbéreket is (<https://www.ksh.hu>).



3. ábra: Az elmúlt 4 év informatikai bértrendjei és a magyar átlagbérek a BlueBird Salary Guide kiadványok (forrás: <https://bluebird.hu/it-fizetesek/>) és a KSH jelentései alapján (forrás: <https://www.ksh.hu/keresetek/>) (saját szerkesztés)

A 3. ábrán látható, hogy bár a feltüntetett informatikai bérekre vonatkozó értékek az elmúlt 4 évben nem növekedtek szignifikánsan, az mégis kijelenthető, hogy ezek az értékek jelentősen felülmúlták a KSH által közölt magyar átlagbérek értékeit (Lásd 7. melléklet).

Ezen a ponton érdemes kiemelni, hogy az „erőforrás” kifejezésnek az informatika világában jellemvonása a többalakúság. Erről Dr. Pitlik László is értekezett az Informatikai erőforrások című publikációjában:

"Az informatika elsődleges erőforrásai az adatok, (pl. matematikai, szervezési) módszerek, valamint a technológia (az IT). Másodlagos erőforrásai az egyén (az ún. „szakértői” tudás) és az emberek szerveződése (kooperáció). Mint látható az IT csak egy (bár karakterisztikus) szelete az informatikának. Az informatika tehát egyfajta világnézet, mely értelmében a rendelkezésre álló nem IT-jellegű erőforrásokat és az információs technológiákat úgy kell kapcsolatba hozni egymással, hogy adott célok esetén a végeredmény hatékonyabb (jelen esetben számvitel-szervezési megoldást) jelentsen, mint IT eszközök nélkül.” (Dr. Pitlik, 2025)

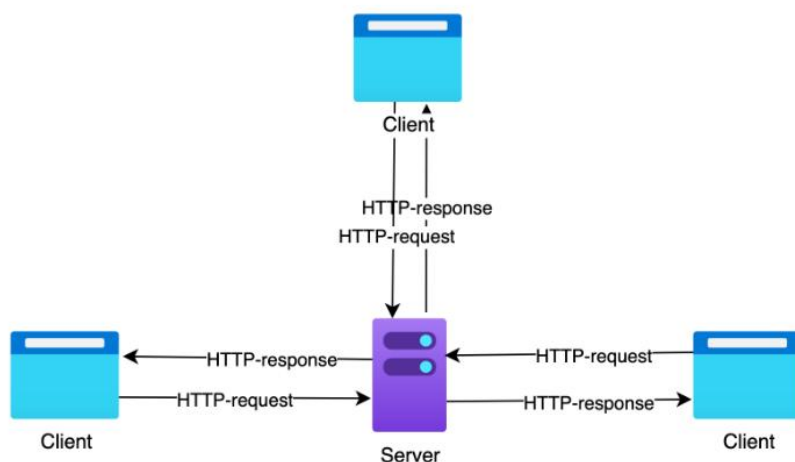
A munkaerő egy ilyen típusú erőforrás (az informatikától függetlenül is), így a felette való rendelkezés profitabilitási kérdés is egyben, ahogyan akár egy szoftverlicenz beszerzésének kérdésköre is az. Ennek mentén megfontolandó lehet egy, a szakdolgozati szoftver beszerzését fontolgató, profitorientált vállalkozás számára, hogy a fejezethez tartozó ábrák és mellékletek ismeretében ezen erőforrását az automatizálás lehetőségének vonatkozásában racionalizálja.

2.4 Szerverek és szerveroldali technológiák

„A kissé megerősített személyi számítógépeket vagy munkaállomásokat gyakran alkalmazzák hálózati kiszolgálókként (szerverként) mind helyi hálózaton (többnyire egy vállalaton belül), mind pedig az interneten. Ezek lehetnek egy- vagy többprocesszorosak, és több gigabájt memória, több száz gigabájt merevlemez és nagy sebességű hálózati kapcsolat jellemző rájuk. Némelyik akár kérések ezreit is képes kezelni másodpercenként.” (Andrew S. Tanenbaum, 2006)

„Az alkalmazásszerverek olyan szerverprogramok, amelyek alkalmazásprogramok számára biztosítanak végrehajtási környezetet egy osztott hálózaton. Pontosabban az alkalmazásszerver jelenti az összes konfiguráció elsődleges futási környezetét, és itt történik az alkalmazások tényleges végrehajtása. Az alkalmazásszerver a webszerverrel együttműködve ad vissza dinamikus, személyre szabott válaszokat a kliensek kéréseire.” (International Business Machines Corporation, 2021)

Ahogy az a fenti hivatkozásokban is olvasható, a kliens-szerver architektúrában a szerver hajtja végre azokat a procedúrákat és eljárásokat, amelyeknek a végrehajtása az előbb említett architektúrában a szerveroldalnak a felelőssége. Ilyen például az adatbáziskapcsolat létrehozása, az adatbáziskezelés vagy egy levelezőklienssel való kommunikáció (Lásd 4. ábra).



4. ábra: Kliens-szerver architektúra (forrás: <https://hyperskill.org/learn/step/25834>)

A 4. ábrán látható kliens-szerver architektúra vonatkozásában érdemes tisztázni, hogy pontosan mi is látható az ábrán. A központi helyen látható server a szerverre, a vele valamilyen interakcióba lépő client elemek a kliensekre és a nyilak pedig a HTTP protokollon alapuló

kommunikációra utal. A szerver felé mutató nyilak a HTTP kérésekre (például: kérem a weboldal tartalmát) és a kliensek felé mutató nyilak pedig a szerver által biztosított HTTP válaszokra utal (például egy weboldal HTML struktúrája)

Ahogy ezen a szakmai blogon is felsorolásra kerül (Trung Tran, 2024), szerveroldali fejlesztések elvégzéséhez több programnyelv és keretrendszer áll rendelkezésre, a teljesség igénye nélkül az alábbiak:

- .NET keretrendszer és a keretrendszerrel kompatibilis programnyelvek:
 - o C# programnyelv
 - o F# programnyelv
 - o Visual Basic programnyelv
 - o PowerShell szkriptnyelv
 - o IronPython programnyelv
- Spring Boot Java programozási nyelvvel integrálva
- FastAPI Python programozási nyelvvel integrálva

2.5 Eseményfolyam-kezelő technológiák

„Az eseménystreaming az alkalmazásokból, adatbázisokból és IoT-eszközökből származó valós idejű adatok rögzítésének és különböző célállomásokra történő továbbításának gyakorlata azonnali feldolgozás és tárolás, illetve valós idejű elemzés és analitikai jelentések készítése céljából.

Az eseményfolyam-feldolgozás (ESP) egyik legfontosabb funkciója, az eseményfolyam-feldolgozás lehetővé teszi az IT-infrastruktúrák számára a nagy, folyamatos eseményáramok kezelését azáltal, hogy az adatokat az esemény vagy változás bekövetkezésekor dolgozza fel.”(IBM Foundation, 2025)

Ahogy a forrás is kifejti, az eseményfolyam-kezelő szoftverhez csatlakozó komponensek képesek valós időben feldolgozni az eseményhez köthető adatsomagot.

A hivatkozott forrás kitér az eseményfolyam-kezelő legfontosabb komponenseinek ismertetésére is, amelyeknek bemutatása fontos a szakdolgozat vonatkozásában: *„Az események természetesen az eseménystreaming központi elemei; azonban egy sor más összetevő teszi lehetővé, hogy a streaming szolgáltatások olyan gyorsan és hatékonyan tudják feldolgozni az eseményeket, ahogyan azt ők teszik. Az egyéb létfontosságú összetevők közé tartoznak:*

Brókerek (Brokers)

A brókerek vagy üzenetközvetítők azok a kiszolgálók, amelyek az eseménystreaming platformokat futtatják. Az üzenetközvetítők lehetővé teszik az alkalmazások, rendszerek és szolgáltatások számára az egymással való kommunikációt és az információcserét azáltal, hogy az üzeneteket a hivatalos üzenetküldési protokollok között átalakítják. (...)

A brókerek clustereket is alkothatnak - a brókerek együtt dolgoznak a könnyebb terheléelosztás és skálázhatóság érdekében.

Topikok (Topics)

A topikok olyan kategorizálások vagy feednevek, amelyekben az események közzététele történik, és amelyek segítségével az események rendszerezhetők és szűrhetők a platformon belül. Az események „tárgyaként” szolgálnak, lehetővé téve a fogyasztók számára, hogy feliratkozzanak a témákra, és csak a releváns eseményeket kapják meg.

Partíciók (Partitions)

A topikok tovább oszthatók partíciókra, lehetővé téve, hogy egy témakörből egyszerre több fogyasztó (Consumer) olvasson anélkül, hogy az egyes partíciók sorrendje megszakadna.”
(IBM Foundation, 2025)

A fenti hivatkozás fontos a szakdolgozat szempontjából, ugyanis a skálázhatóság egy, a szakdolgozati szoftver működésének szempontjából igen jelentős funkcionálitása a működés tekintetében.

Fontosnak tartottam ezen fogalmak tisztázását a későbbi fejezetek vonatkozásában (Lásd [3.1.1.3 Eseményfolyam-kezelő platform](#), [3.2.3 Eseményfolyam-kezelő](#)).

2.6 Webkliensek és kliensoldali technológiák

„A világháló az interneten egymással hivatkozásokkal összekötött dokumentumok rendszere. A dokumentum itt egy tágabb értelemben vett tartalom. Általában egy weboldalt értünk alatta, amely HTML (HyperText Markup Language) formában tartalmazza a megfelelően strukturált információkat, amely azonban további hivatkozásokat tartalmazhat HTML állományokra, stílusállományokra, képekre, hangokra, videókra, stb. Dokumentum tehát lehet HTML állomány, kép, hang, videó, egyszerű szöveges állomány, tömörített állomány, tulajdonképpen bármi, amit a weben el tudunk érni. Az eléréshez megfelelő módon kell a tartalomra hivatkozni, ezt határozza meg az URL (Uniform Resource Locator) szabvány. A dokumentumokat a web

szerverek teszik elérhetővé az interneten keresztül az ún. HTTP (HyperText Transfer Protocol) protokoll segítségével. A világhálón megjelenő weboldalakat pedig megfelelő ügyfél (kliens) program, a webböngésző segítségével tekinthetjük meg.

A világhálón a dokumentumok elérése kliens-szerver architektúrában valósul meg. Egy dokumentum kérését mindig a böngésző (a kliens) kezdeményezi. A böngésző címsorába beírva a tartalom URL-jét, vagy egy URL-t tartalmazó hivatkozást aktiválva a böngésző a kérést a HTTP protokollon keresztül elküldi a szervernek, ahol a tartalom található. A szerver feldolgozza a kérést, majd a megfelelő feladatok elvégzése után visszaküldi a kívánt tartalmat a böngészőnek, amely megjeleníti azt (ha tudja). A folyamat sematikus ábrázolását az alábbi ábra mutatja.”(Elte Informatikai Kar, 2025)

A fenti idézett sorokból következik, hogy egy webes alapú, felhasználói vizualizációt szolgáló szoftverkomponens létrehozásához webböngészőre és a webböngészőre megírt szoftver megírására van szükség.

Ahogy ezt az alábbi szakmai blogon is olvasható (Brian, 2025), számos webtechnológia áll rendelkezésre ezen feladat elvégzésére. A teljesség igénye nélkül ezek az alábbiak:

- Javascript programnyelv a következő keretrendszerekkel:
 - ReactJS keretrendszer
 - Angular keretrendszer
 - Vue.js keretrendszer
- Python programnyelv a következő elérhető keretrendszerekkel:
 - Django keretrendszer
 - Flask keretrendszer

2.7 Konténerizáció: üzemeltetés és karbantartás

Az üzemeltetés és karbantartás szakirodalmi megalapozását (a szakdolgozati témával és a megoldandó problémával összhangban) a konténerizáció vonatkozásában teszem meg. Ez amiatt is fontos, mert a szoftverüzemeltetés és a futtatott szoftverek karbantartása olyan szerteágazó témaegyüttesek, amelyeknek részletes elemzése (ezen tárgykörök volumenét tekintve) nem lehet ennek a szakdolgozatnak a fókuszában.

Mindazonáltal az előre definiált mozgástér is bőven ad lehetőséget ezen témakör szakirodalmi háttérének bemutatására.

„Bár a konténerek és a virtuális gépek elsőre hasonlóknak tűnnek, van néhány fontos különbség, amelyeket a legkönnyebb diagramok segítségével elmagyarázni.

Az 1-1. ábra három alkalmazást mutat, amelyek külön virtuális gépeken futnak egy adott szervergépén. A virtuális gépek létrehozásához és futtatásához a hypervisorra van szükség, amely szabályozza a hozzáférést a mögöttes operációs rendszerhez és hardverhez, valamint szükség esetén értelmezi a rendszerhívásokat. Minden egyes virtuális gépnek szüksége van az operációs rendszer, a futtatandó alkalmazás és a támogató könyvtárak teljes másolatára.

Ezzel szemben az 1-2. ábra azt mutatja, hogyan futtatható ugyanaz a három alkalmazás egy konténerizált rendszerben. A virtuális gépekkel ellentétben a gazdaszerver kernelét megosztják a futtatott konténerekkel. Ez azt jelenti, hogy a konténerek mindig ugyanazt a rendszermagot futtatják, mint a gazdaszerver. Az Y és Z alkalmazások ugyanazokat a könyvtárakat használják, és megoszthatják ezeket az adatokat ahelyett, hogy redundáns másolatokkal rendelkeznének. A konténerekben belül futó folyamatok azonban egyenértékűek a gazdán lévő natív folyamatokkal, és nem viselik a hipervizoros futtatással járó terheket.” (Adrian, 2015)

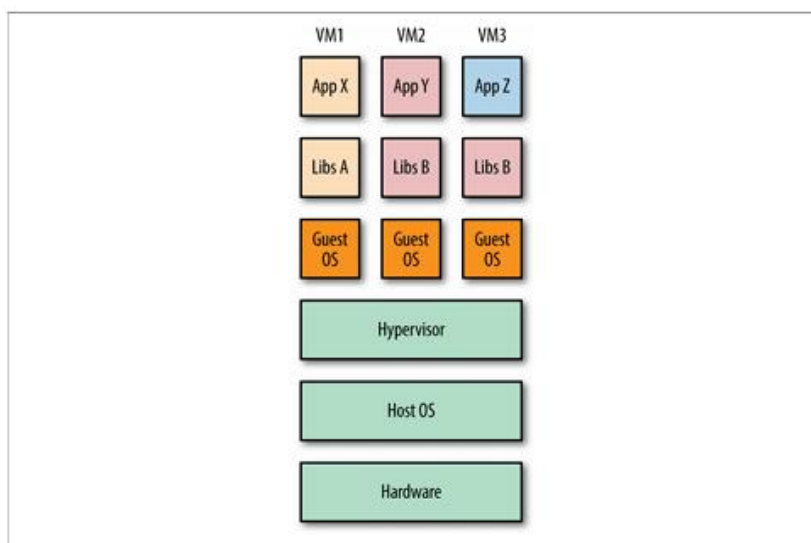


Figure 1-1. Three VMs running on a single host

5. ábra: Szakirodalmi hivatkozásban említett 1-1. ábra (forrás: (Adrian, 2015))

Az 5. ábrán egy olyan struktúra látható, ahol az applikációk külön virtuális gépeken futnak. A zöld elemek a virtuális gépek szervereként szolgáló számítógép komponenseire, míg a narancssárga pedig a számítógépre telepített virtuális gépeken futó operációs rendszerre utal. A kék, piros és sárga elemek pedig a három futtatott applikáció eltérő elemeire utalnak (futtatható

állomány, kapcsolódó könyvtárak). Erre az ábrára utalt a hivatkozott szakirodalom is a szakirodalomban található 1.1 ábra néven.

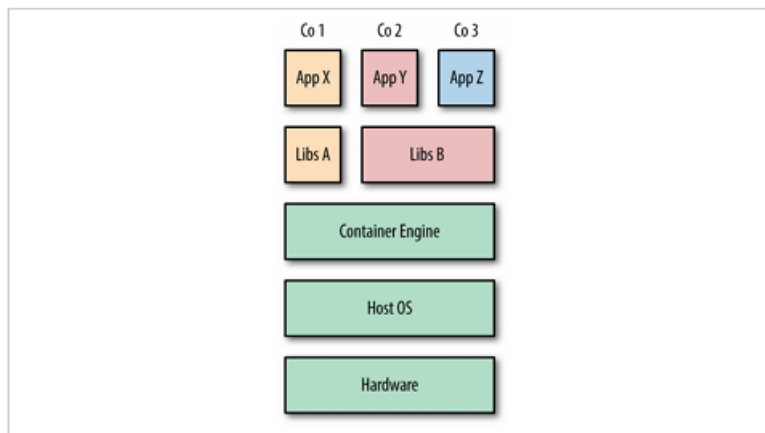


Figure 1-2. Three containers running on a single host

6. ábra: Szakirodalmi hivatkozásban említett 1-2. ábra (forrás: (Adrian, 2015))

A 6. ábrán egy olyan struktúra látható, ahol az applikációk külön konténerekben futnak. A zöld elemek a konténerek szervereként szolgáló számítógép komponenseire, míg a kék, piros és sárga elemek pedig a három futtatott applikáció eltérő elemeire utalnak (futtatható állomány, kapcsolódó könyvtárak). Erre az ábrára utalt a hivatkozott szakirodalom is a szakirodalomban található 1.2 ábra néven.

Az IBM hivatalos weboldalán található cikkben (<https://www.ibm.com/think/topics/containerization>) több jelentős konténerizációs platformot is megemlítenek, például:

- „Docker”
- „CoreOS rkt”
- „Mesos Containerizer”
- „LXC Linux Containers”
- „OpenVZ”

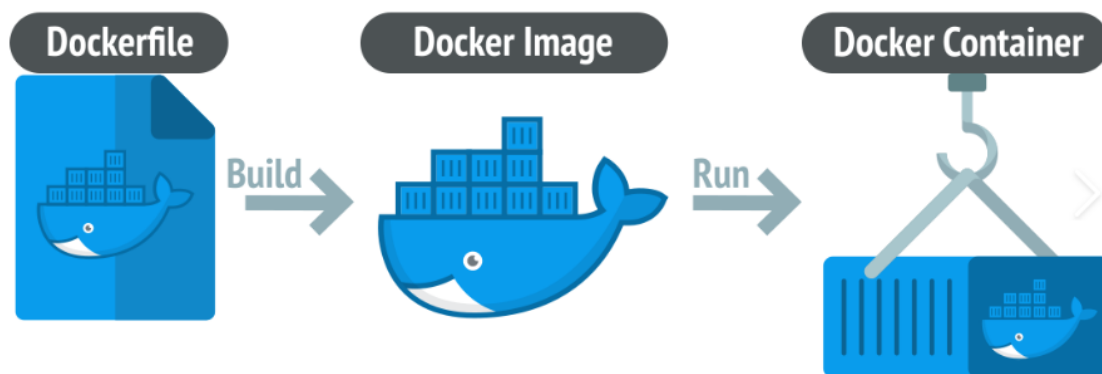
Ahogy az említett cikkben is kifejtik, a Docker a legismertebb platform ezek közül, így ennek a technológiának a részletesebb bemutatására vállalkozom.

2.7.1 Docker

Ahogy az a School of Computer Science and Engineering, KLE Technological University publikációjában is olvasható (Potdar et al., 2020), a konténerizációs technológiák megjelenésével és széleskörű használatával a szoftverfejlesztési és szoftvertelepítési ciklusok

jelentősen egyszerűsödtek és időbeli tényezőjükben lerövidültek, így lehetővé téve a gördülékenyebb és gyorsabb szállítást a szoftvert megrendelő(k) részére.

„A Docker egy olyan platform, amely az alkalmazások futtatására alkalmas, konténereknek nevezett könnyű egységek formájában. A konténerek mindenhol elterjedtek a szoftverekben, a felhőben futó szerver nélküli funkcióktól a vállalatok stratégiai tervezéséig. A Docker egyre inkább az üzemeltetők és a fejlesztők alapvető kompetenciájává válik az egész iparágban - a Stack Overflow 2019-es felmérésében a Docker az emberek által „legkeresettebb” technológiaként szerepelt.”(Elton Stoneman, 2020)



7. ábra: Docker konténer létrehozásának egyszerűsített folyamatábrája (forrás: <https://linuxiac.com/what-is-docker-container/>)

A 7. ábrán látható, hogy egy szoftvert miként lehet konténerizáltan futtatni.

Stációk:

- Dockerfile:
 - „A konténerekre való áttérés némi befektetést igényel: a meglévő telepítési lépéseket Dockerfiles nevű szkriptekké kell alakítania.” (Elton Stoneman, 2020)
 - Dockerfile egy szkriptfájl, amiben utasítások vannak megadva a konténer létrehozását illetően. Tekinthető egyfajta utasításokat tartalmazó konfigurációs fájlnek is.
- Docker Image:
 - „Az image tartalmazza az alkalmazás összes tartalmát, valamint a Docker számára az alkalmazás indítására vonatkozó utasításokat.” (Elton Stoneman, 2020)
 - A Dockerfile elkészítése után a Docker CLI használatával kiadott build utasítás létrehozza a Docker Image-t, ami a konténer „alaprajza”
- Docker Container:

- „A Docker konténer ugyanaz a koncepció, mint egy fizikai konténernél - gondolj rá úgy, mint egy dobozra, amelyben egy alkalmazás található. A dobozon belül az alkalmazás szempontjából úgy tűnik, mintha egy saját számítógép venné körül: saját gépneve és IP-címe is van.” (Elton Stoneman, 2020)
- Az image létrehozása után a Docker CLI használatával kiadott run utasítás létrehozza és elindítja a Docker Container-t

A világ egyik jelentősebb vezető konténerizációs eszköze a Docker platform, amely nyílt forráskódú, használatával költséghatékonyság is elérhető (amennyiben ez szempont a technológiát felhasználó számára), és széleskörű fejlesztői közösség tartja karban és fejleszti az tovább eszközt.(Mirantis Inc., 2025) Mindemellett a platformfüggetlenséget is elérhető a Docker használatával, ugyanis a konténerizált szoftver minden olyan ökoszisztémán futtatható, amelyen elérhető a Docker Engine.(Mirantis Inc., 2025)

2.7.2 Docker Compose

„A Docker Compose egy eszköz több konténerből álló alkalmazások definiálására és futtatására. Ez a kulcs az áramvonalas és hatékony fejlesztéshez és telepítéshez.

A Compose leegyszerűsíti a teljes alkalmazáshalmaz vezérlését, megkönnyítve a szolgáltatások, hálózatok és volume-ok kezelését egyetlen, érthető YAML konfigurációs fájlban. Ezután egyetlen paranccsal létrehozhatja és elindíthatja az összes szolgáltatást a konfigurációs fájlból.” (Docker Inc., 2025)

A Docker ökoszisztémának része a Docker Compose (Docker Inc., 2025), ami egy YAML konfigurációs fájl alapú szoftver. Ennek a szoftvernek a segítségével a konfigurált utasítások alapján egy parancs kiadásával komplett projekteket lehet elindítani, így tehermentesítve a kitelepítő szakembert a komplikált docker utasítások menedzselésétől. Ezt a szolgáltatást is díjmentesen biztosítja a Docker.

2.7.3 Kubernetes

„A Kubernetes egy konténerek futtatására szolgáló platform. Gondoskodik a konténerizált alkalmazások indításáról, a frissítések menedzseléséről, a szolgáltatási szintek fenntartásáról, az igényeknek megfelelő skálázásról, a hozzáférés biztosításáról és még sok másról. A Kubernetes két alapvető koncepciója az API, amelyet az alkalmazások definiálására használ, és a cluster, amely az alkalmazásokat futtatja. A cluster olyan egyedi kiszolgálók összessége,

amelyek mindegyike egy konténerfuttatóval, például a Dockerrel lett konfigurálva, majd a Kubernetes segítségével egyetlen logikai egységgé lett egyesítve.” (Elton Stoneman, 2021)

A Cloud Native Computing Foundation által karbantartott és támogatott Kubernetes fő célja a konténermenedzsment. Ahogyan ez a dokumentációban is szerepel (<https://kubernetes.io/docs>), a Kubernetes egy olyan konténermenedzselő szoftver, amely (a teljesség igénye nélküli felsorolás szerint) képes automatikusan skálázni, monitorozni és műveleteket végrehajtani a rábízott konténereken.

„Az alkalmazások skálázásának alapkonceptiója egyszerű: futtass több Podot. A Kubernetes absztrahálja a hálózatot és a tárolást a számítási rétegtől, így sok Podot futtathatsz, amelyek ugyanannak az alkalmazásnak a másolatai, és egyszerűen csak csatlakoztathatod őket ugyanahhoz az absztrakcióhoz. A Kubernetes ezeket a Podokat replikáknak nevezi, és egy multi-node clusterben ezek a node-ok között lesznek elosztva. Ez a skálázás minden előnyét biztosítja - nagyobb kapacitás a terhelés kezeléséhez, és magas rendelkezésre állás hiba esetén - mindezt egy olyan platformon, amely másodpercek alatt képes fel- és lefelé skálázódni.”(Elton Stoneman, 2021)

Továbbá beépített terheléelosztási funkcióval rendelkezik, így garantálva, hogy a beérkező forgalmat automatikusan elosztja a Kubernetes Clusterben elérhető (Kubernetes Pod entitásokban futtatott) konténerek között. YAML fájlokban lehet definiálni azokat a Kubernetes erőforrásokat, amelyek szükségesek a szoftver futtatásához.

Előfordulhat szoftverüzemeltetés során, hogy az adott ökoszisztémában elérhetetlenné válik egy futtatott Docker konténer. Sem a Docker, sem a Docker Compose nem monitorozza, nem *menedzseli* a már létrehozott konténereket, konténermenedzselő eszköz használata nélkül ebben az esetben az üzemeltetőnek manuálisan (vagy ha írt ilyen esetekre szkriptet, akkor azt használva) kell az ökoszisztémában új konténert létrehozni és elindítani. Amennyiben az új konténer konfigurációi módosultak, arról értesíteni kell a többi konténert is (feltételezve a köztük meglévő kommunikációt), ez pedig praktikus azt jelenti, hogy ezen konténerek konfigurációit is módosítani kell. Egy konténermenedzselő szoftver ennek a potenciális probléma- és feladatkörnek a felelősségét veszi le az üzemben tartó válláról.

Érdekes még a Kubernetesrel kapcsolatos alapfogalmakat rögzíteni a szakdolgozati szekcióban:

„A konténer egy virtualizált környezet, amely jellemzően egyetlen alkalmazáskomponenst

futtat. A Kubernetes a konténert egy másik virtualizált környezetbe csomagolja: a Podba. A Pod egy számítási egység (...)” (Elton Stoneman, 2021)

„A podok túl egyszerűek ahhoz, hogy önmagukban hasznosak legyenek - egy alkalmazás elszigetelt példányai, és minden pod egy Node-hoz van hozzárendelve. Ha ez a Node offline lesz, akkor a Pod elveszik, és a Kubernetes nem helyettesíti azt.” (Elton Stoneman, 2021)

„Itt jönnek a képbe a kontrollerek. A controller egy olyan Kubernetes-erőforrás, amely más erőforrásokat kezel - a Kubernetes API-val együttműködve figyeli a rendszer aktuális állapotát, összehasonlítja azt az erőforrásainak kívánt állapotával, és elvégzi a szükséges változtatásokat. Sokféle vezérlő létezik, de a Podok kezeléséhez a legfontosabb a Deployment, amely megoldja az imént ismertetett problémákat.” (Elton Stoneman, 2021)

A hivatkozott fogalmak (Pod, Deployment és Node) értelmezési keretbe rendelése szükséges lépés, ugyanis használni fogom ezen alapfogalmakat a szakdolgozati munka üzemeltetésre vonatkozó szegmenseinek bemutatásánál.

2.8 Verziókezelő rendszerek

A harmadik Computer Science and Computational Intelligence nemzetközi konferencia által kiadott, a verziókezelő rendszerekre vonatkozó kiadványában is olvasható (Zolkifli et al., 2018), hogy a verziókezelő rendszerek széles körben alkalmazott eszközök a szoftverfejlesztés területén.

„A verziókezelő rendszereket (VCS) sok szoftverfejlesztő használja a projektfejlesztések során, mivel segít nekik a forráskódok kezelésében, és lehetővé teszi számukra, hogy a projekt minden verzióját megőrizzék, amelyen dolgoztak. Ez az objektumok fejlesztésének irányítása, szervezése és koordinálása felé vezető út. A szoftverfejlesztésben a szoftverfejlesztőknek együtt kell működniük egymással, hogy egy jobb projektet fejlesszenek ki. Így a VCS nagyon hasznos, mert olyan együttműködési keretrendszert is támogat, amely megkönnyíti a szoftverfejlesztők hatékony együttműködését. VCS nélkül az együttműködés nagy kihívást jelent.” (Zolkifli et al., 2018)

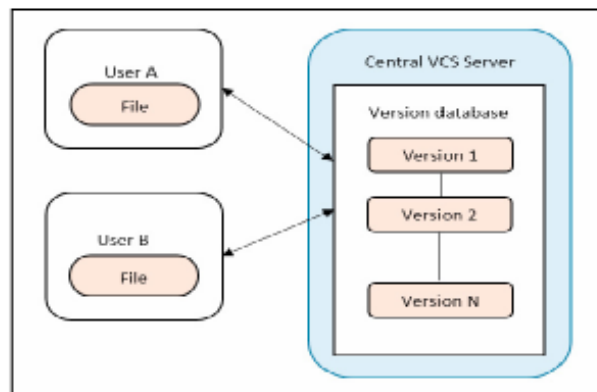
Lényeges ezen rendszerek használata, ugyanis ezekkel az eszközökkel lehetősége nyílik a fejlesztőknek a kezelt forráskód egyszerű menedzselésére, szeparálva a forráskód különböző verzióit. A verziószeparálás funkcionalitásának köszönhetően a szoftverfejlesztők közötti kollaboráció is egyszerűsödik, több fejlesztő szakember is dolgozhat egyazon forráskódon egyidőben.

Két típusú verziókezelő rendszer különböztethető meg: centralizált (CVCS, tehát Centralized Version Control System) és decentralizált (DVCS, tehát Distributed Version Control System) verziókezelő.

2.8.1 CVCS verziókezelő rendszerek

„A CVCS egy olyan megközelítés, amely lehetővé teszi a fejlesztők számára, hogy együttműködve dolgozzanak. Ez tárolja a fájlok előzményeinek egy mesterpéldányát, és annak érdekében, hogy elolvassa, lekérje, új változtatásokat rögzítsen egy bizonyos verzióhoz, kapcsolatba kell lépniük egy szerverrel.”(Zolkifli et al., 2018)

A CVCS verziókezelők centralizált modellt valósítanak meg, amelynek fő komponense egy központi forráskód tárhely (repository) (Lásd 8. ábra). Ezt a tárhelyet egy távoli szerveren érhetik el a felhasználók, a távoli szerver elérése nélkül viszont a kollaboratív munka tulajdonképpen ellehetetlenül. Ezen rendszerek másik jelentős hiányossága, hogy a forráskód naplózása egy szerveren történik meg, aminek vannak bizonyos IT-biztonsági kockázatai. Ilyen biztonsági kockázat például egy szerver permanens elérhetetlenné válása.



8. ábra: CVCS verziókezelő rendszer (forrás: <https://www.sciencedirect.com/science/article/pii/S1877050918314819>)

A 8. ábrán látható, hogy a két hasonló alakú, fehér háttérű elem a felhasználókra, míg a részben kék háttérű elem pedig a centrális repository-ra utal. A nyilak érintkezési pontjai alapján látható, hogy a két felhasználó mind egy tárhelyre kapcsolódik, mégpedig a központi tárhelyre.

Népszerű centralizált verziókezelő rendszerek például:

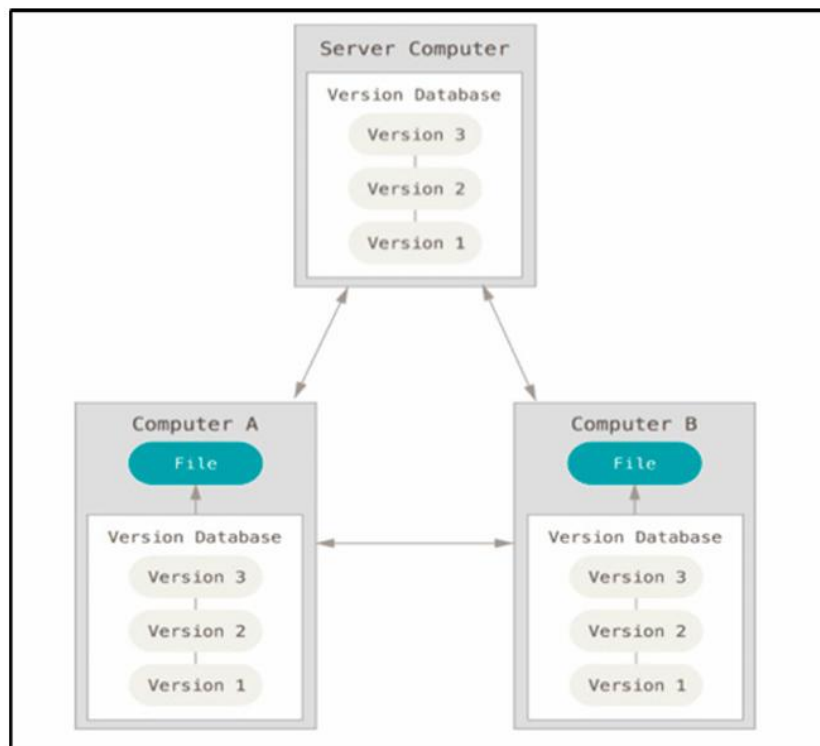
- „Concurrent Version System”
- „Subversion”
- „Perforce”

2.8.2 DVCS verziókezelő rendszerek

„(...) A DVCS-t úgy tervezték, hogy mindkét módon működjön, mivel minden gépen helyben tárolja a fájlok teljes előzményeit, és szükség esetén a felhasználó által végrehajtott helyi módosításokat is képes szinkronizálni a szerverre, hogy a módosításokat meg lehessen osztani az egész csapattal.” (Zolkifli et al., 2018)

A DVCS verziókezelők központi tárhely nélkül valósítják meg a maguk modelljét, ezen centralizált tárhely helyett minden felhasználónak saját lokális tárhelye van a számítógépén. A projekt naplóbejegyzéseinek másolata minden felhasználó lokális gépén megtalálható, így a felhasználó a saját munkáját és a korábbi bejegyzések másolatát is könnyen tudja szinkronizálni a távoli szerverrel.

Ennek köszönhetően, ha az egyik szerver permanensen elérhetetlenné válik és a korábbi fejlesztések kollaborációja megtörtént, úgy bármelyik kliens tárhelyén tárolt adattal megtörténhet az adathelyreállítás. Következésképpen minden „klón” tárhely az összes adat teljes biztonsági mentéseként működik.



9. ábra: DVCS verziókezelő rendszer (forrás: <https://www.sciencedirect.com/science/article/pii/S1877050918314819>)

A 9. ábrán látható, hogy a két, az ábra alsó részén elhelyezkedő elem a felhasználók számítógépeire, míg az ábra felső részén elhelyezkedő elem pedig a szerverszámítógépre utal.

A nyilak érintkezési pontjai alapján látható, hogy a számítógépeken megtalálhatóak a fájlok és verziókezelési naplóbejegyzések teljes másolata. Ennek köszönhetően (ahogyan azt a szakirodalmi idézetben is olvashattuk) képes mindegyik másolat biztonsági másolatként funkcionálni a többi aktor részére.

Népszerű decentralizált verziókezelők például:

- „Git”
- „Mercurial”
- „Bazaar”

2.9 Statikus kódelemzés

„A szoftverfejlesztés egyik alappillére az analitikus módszerek alkalmazása a forráskód felülvizsgálatára a végrehajtási hibák kijavítása érdekében.”(Gomes et al., 2009)

„A statikus kódelemzés a számítógépes szoftverek olyan elemzése, amelyet a szoftverből fordított programok tényleges végrehajtása nélkül végeznek, ellentétben a dinamikus elemzéssel (a szoftverek tesztelése a programok végrehajtásával). Az esetek többségében az elemzést a forráskód valamelyik változatán, a többi esetben pedig az objektumkód valamelyik formáján végzik. A kifejezést általában az automatizált szoftvereszközzel végzett elemzésre alkalmazzák, az emberi elemzést programmegértésnek, programértésnek vagy kódvizsgálatnak nevezik.”
(Gomes et al., 2009)

A tanulmány szerint a szoftverfejlesztés során értékes információkat lehet kinyerni a statikus kódelemzés eredményei alapján, így a program futtatása nélkül találhat meg a fejlesztő potenciális hibákat és ezáltal a szoftverminőséget is javíthatja.

2.10 Adatbázisok

„Az adatbázisok és az adatmodellezés párhuzamosan fejlődtek az 1960-as évektől kezdődően.

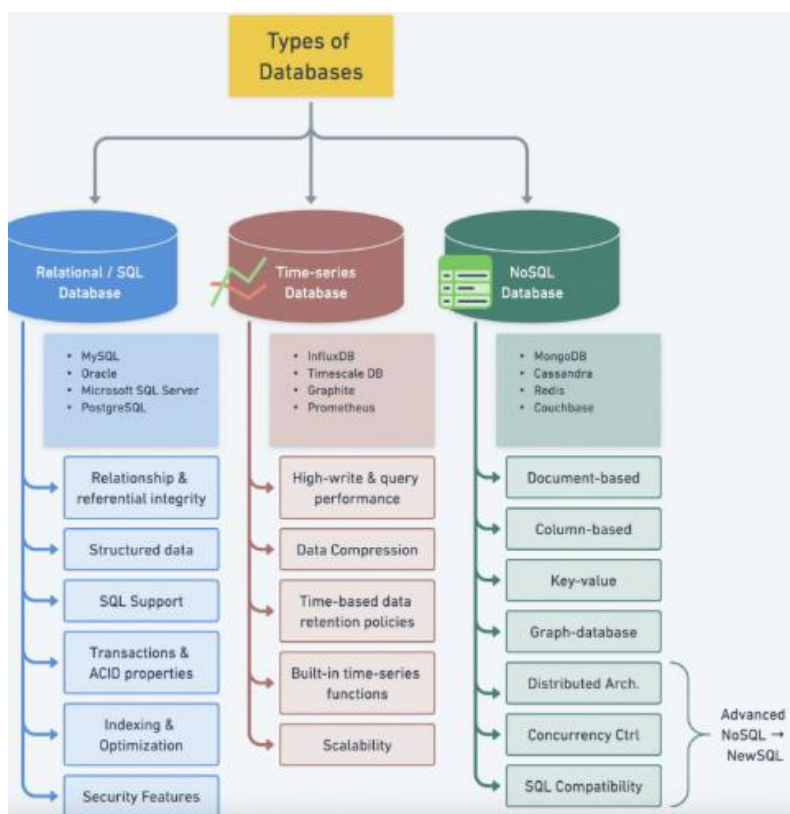
Ekkorra már komolyabb információs rendszerek fejlesztését igényelték a felhasználók, és az eddigi kevés adat és kis tárolókapacitás helyett komolyabb adatmennyiségek kezelése és tárolása vált szükségessé. Az addig szokásos számítások és különböző feldolgozások után az adatok kerültek középpontba.

Több elméleti modell keletkezett adatbázis-kezelők fejlesztésére. Ezekből három terjedt el: a hálós, a hierarchikus és a relációs adatbázisok.” (Bánné Varga, 2022)

„A relációs adatbázis a leggyakoribb típus, amelyben az adatok olyan táblákba vannak szervezve, amelyek minden egyes entitásról információkat tárolnak, és előre meghatározott kategóriákat tartalmaznak sorokban és oszlopokban. Ezek a strukturált adatok egyszerre hatékonyak és rugalmasan elérhetők.

Relációs adatbázis például az SQL Server, az Azure SQL, a MySQL, a PostgreSQL és a MariaDB.” (<https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-are-databases>)

„A szabványosítás következménye, hogy napjainkban az SQL a relációs adatbázis-kezelés általánosan elfogadott nyelvéné vált. Ma minden jelentősebb adatbázis-kezelő rendszer SQL alapú, köztük van az IBM DB2, az Oracle 18, az Informix, a Sybase, amely a mai Microsoft SQL Server 2017 alapja lett, a MySQL Server 8, hogy csak a legismertebbeket említsük. Az egyes megvalósítások között vannak kisebb különbségek, de a legfontosabb kérdésekben, a nyelv „filozófiájában” nincs eltérés.” (Bódy, 2019)



10. ábra: Adatbázistípusok (forrás: <https://www.devopsschool.com/blog/types-of-databases/>)

A 10. ábrán az adatbázistípusok (hengeralakú elemek) vizuális reprezentációja látható a hozzájuk tartozó implementációkkal (négyzet alakú elemek közvetlenül a hengeralakú elemek alatt) és jellemzőikkel (sötétebb színű kerettel ellátott téglalapok).

A tárolni kívánt strukturált adatok perzisztens hozzáférhetőségének elérése érdekében relációs adatbázisok használatára van szükség. A tárolt adatbázis entitásokat pedig a szabványosított SQL lekérdezőnyelv használatával lehet az adatbázisból a legegyszerűbben lekérdezni.

Ahogy az Bódy könyvében is olvasható, több lekérdező dialektus is létezik, amelyek SQL alapúak. Ennek köszönhetően az említett funkcionalitásbeli különbségek alapján kiválasztható és a feladat szakmai színvonalon történő elvégzéséhez célszerű is kiválasztani az adott feladatra legalkalmasabb SQL lekérdező dialektust.

2.11 Folyamatos Integráció, Folyamatos Szállítás

„Számos, a piacon elfogadott automatizálási gyakorlat létezik:

- A CI - az a gyakorlat, amely arra utasítja a fejlesztőket, hogy naponta többször rögzítsék a változtatásokat a közös tárolóba/fővonalba.*
- CDE - az a gyakorlat, amely arra törekszik, hogy minden rendelkezésre álló build stabil, ellenőrzött és ellenőrzött legyen, hogy egyetlen gombnyomással élesbe lehessen tolni.*
- CD - az a gyakorlat, amely tovább bővíti a CDE-t, biztosítva, hogy a build ne csak készen álljon és ellenőrizve legyen, hanem automatikusan élesben is átkerüljön a termelési környezetbe, hogy a valós felhasználók azonnal láthassák a változásokat.*

Az összes fent említett gyakorlat összefügg egymással, és a gyors és hibamentes telepítések lehetővé tétele révén a több piaci hely fogyasztására törekvő automatizálás iránti vágy következménye.” (Bobrovskis & Jurenoks, 2018)

Ebben a kutatásban értekeznek (Bobrovskis & Jurenoks, 2018) a Folyamatos Integráció, Folyamatos Szállítás elvekről és az alkalmazásukkal járó előnyökkel a forráskód módosítások esetében. Ezen elvek alkalmazása esetén az új funkcionalitások lefejlesztésével és leszállításával a szoftverintegráció megtörténik, a szoftver biztonságos működése biztosított, a megrendelői oldalról elvárt minőség pedig garantált lesz.

2.12 Tantárgyak szakdolgozati kapcsolata

Az alábbi fejezetben bemutatom a szakdolgozat összefüggéseit (amennyiben vannak összefüggések) a képzési mintatantervben szereplő tantárgyakkal.

2.12.1 Európai civilizáció és identitás

„Az EU egy sor munkaügyi szabályt vezetett be az erős szociális védelem biztosítása érdekében. Ezek közé tartoznak a munkafeltételek - például a bérekre, a munkaidőre, a részmunkaidőre, a munkavállalók jogaira - vonatkozó minimumkövetelmények, a foglalkoztatásuk fontos szempontjairól való tájékoztatás és a munkavállalók kiküldetése. Ezek az európai szociálpolitikák egyik sarokkövévé váltak.”(European Parliament, 2019)

A kiegyensúlyozott munka-magánélet viszony és a kielégítő munkahelyi körülmények mind kiemelt prioritást élveznek az Európai Unió munkavállalópolitikájában, ahogyan erről az Európa Parlament weboldalán is részletesen írnak (European Parliament, 2019). Véleményem szerint az európaiság sarokkövévé vált a minőségi életre való törekvés, az idézett weboldalon megjelent manifesto is ezt támasztja alá. Mivel a szakdolgozati projekt az automatizálással adminisztratív terheket vesz le a munkavállaló válláról, így ez lényegében összhangban van az EU törekvéseivel.

2.12.2 Szakterületi jogi ismeretek

Az összefüggésvizsgálat első lépéseként megkérdeztem a ChatGPT nevű generatív mesterséges intelligenciát arról, hogy a tantárgy milyen módon kapcsolódhat a szakdolgozat témájával. Az alábbi listát generálta (Lásd 9. melléklet):

„Az alábbiakban néhány példát hozok arra, hogy milyen módon lehet összefüggésbe hozni a két területet:

- 1. **Adatvédelem és biztonság:** Az adminisztratív hatékonyság növelése az Azure SDK és DevOps eszközökkel kiterjedhet olyan területekre is, mint az adatvédelem és az információbiztonság. A jogi ismeretek segíthetnek megérteni és betartani a vonatkozó adatvédelmi jogszabályokat és előírásokat, például a GDPR-t.*
- 2. **Szabályozási megfelelés:** Az adminisztratív folyamatok egyszerűsítése során fontos biztosítani, hogy minden tevékenység megfeleljen a vonatkozó jogszabályoknak és szabályozásoknak. A Szakterületi jogi ismeretek tárgy keretében megtanult szabályozási és megfelelési kérdések itt központi szerepet játszhatnak.*
- 3. **Szerződések és licencelés:** Az Azure és DevOps használata során különböző szerződések és licenelési feltételek érvényesek lehetnek, amelyek jogi ismereteket igényelnek. A tantárgy során megszerzett tudás segíthet eligazodni ezekben a kérdésekben.*

4. **Jogi felelősség és kockázatkezelés:** Az adminisztratív feladatok automatizálása és az IT eszközök használata jogi felelősséget is magával hozhat. A jogi ismeretek segíthetnek azonosítani és kezelni a lehetséges jogi kockázatokat.

Ezek alapján úgy gondolom, hogy a Szakterületi jogi ismeretek tantárgy hasznos alapot nyújthat a szakdolgozatod elkészítéséhez és az adminisztratív feladatok hatékonyabbá tételéhez a Microsoft Azure SDK és Azure DevOps alkalmazásával.”

A felsorolás mind a négy eleménél könnyen beláthatóak a kapcsolódási pontok, mindazonáltal az alábbi jogi ügyekkel foglalkozó oldal blogbejegyzése (dr. Dobos, 2016) is a ChatGPT megállapításait támasztja alá:

„A szoftverek használatára vonatkozó általános szabályokat a szerzői jogi törvény (1999. évi LXXVI. törvény; a továbbiakban: Sztj.) tartalmazza, míg a konkrét felhasználói jogviszonyt a legtöbb esetben egyedi licencszerződések szabályozzák. A védelmi rendszer kiterjed a fejlesztő és a felhasználó viszonyára, valamint a konkurens fejlesztők egymás közötti kapcsolataira. Tekintettel arra, hogy a védelem eszközei között a jogi szabályozási megoldások központi szerepet játszanak, ezért az alábbiakban a szoftver fejlesztésével és felhasználásával kapcsolatos jogi szabályok kerülnek részletesebb bemutatásra.” (dr. Dobos, 2016)

A generált válasz és a referált forrás megállapításai mentén a [3.2 A szoftver megvalósítása](#) fejezetben bemutatom a dolgozat vonatkozásában felmerült szakterületi jogi aspektusokat.

2.12.3 Matematikai alapok

Ahogy arra a [3.2 A szoftver megvalósítása](#) fejezetben is bemutattam, a fejlesztési folyamat során használtam mesterséges intelligencia alapokon működő nyelvi modelleket (Copilot és ChatGPT), amelyek matematikai alapokon végzi az adatok elemzését és feldolgozását. Lásd 8. melléklet és 9. melléklet.

Kapcsolódik a statikus kódelemzéshez (Lásd [2.8 Statikus kódelemzés](#)) is a matematika a dolgozatot illetően, ugyanis a kódelemzés elvégzéséhez analízis szükséges.

„Minden Dockerfile-t úgy kell optimalizálni, hogy az utasítások aszerint legyenek sorrendben, hogy milyen gyakran változnak: a Dockerfile elején legyenek azok az utasítások, amelyek valószínűleg nem változnak, a végén pedig azok, amelyek valószínűleg változnak. A cél az, hogy a legtöbb buildnek csak az utolsó utasítást kelljen végrehajtania, minden másra a gyorsítótárat használva. Ez időt, lemezterületet és

hálózati sávszélességet takarít meg, amikor elkezd megosztani a képeket.” (Elton Stoneman, 2020)

Ahogy a fenti hivatkozásban is olvasható, egy Dockerfile (Lásd [2.6.1 Docker](#)) strukturálásához analizálni kell a szükséges lépéseket és azok módosulási probabilitását is ki kell számítani az effektív futási idők eléréséhez.

2.12.4 Adatszerkezetek és algoritmusok

„Mint látni fogjuk, a jó architektúra alapja az objektumorientált tervezés (OO) elveinek megértése és alkalmazása.

(...)

Az adatelrejtést (encapsulation) azért említik az objektumorientált tervezés definíciójának részeként, mert az objektumorientált tervezési nyelvek lehetővé teszik az adatok és funkciók egyszerű és hatékony adatelrejtést.

(...)

Az öröklődés (inheritance) egyszerűen a változók és függvények egy csoportjának újbóli deklarálása egy körülvevő hatókörön belül.

(...)

A lényeg, hogy a többalakúság (polymorphism) a mutatók (pointer) függvényekre való alkalmazását jelenti. A programozók a Von Neumann-architektúrák 1940-es évek végi bevezetése óta használják a függvényekre mutató mutatókat a polimorf viselkedés eléréséhez.”(Martin, 2018)

A fenti idézet eszmefuttatás Robert C. Martin Clean Architecture című könyvéből egyértelművé teszi a fejlesztő szakember számára, hogy ezen elvek alkalmazásával magasabb kódminőségű adatszerkezetek és algoritmusok írhatóak.

A szakdolgozati munkában használt adatszerkezetek és algoritmusok szakirodalmi háttérét a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam, a megvalósítás a [3.2.1 Webfelület](#) és [3.2.2 Szerveroldal](#) fejezetekben, az adatbázisentitás adatszerkezeti megvalósítása pedig a [3.3.4 Adatbázis](#) fejezetben kerültek bemutatásra.

2.12.5 Hálózatok és számítógép architektúrák

A szakdolgozati munkában megjelenő hálózatokról és számítógép architektúrákról a szakirodalmi háttérrel a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

A Hálózatok és számítógép architektúrák tantárgy megvalósult vonatkozásai pedig a [3.1.2.6 Konténerizáció, konténerizáltan futtatható állományok tárolása](#), a [3.1.2.7 Kitelepítést és futtatást támogató egyéb technológiák](#), a [3.2.6 Üzemeltetés](#) és a [3.3.6 Üzemeltetés](#) fejezetekben találhatók.

Ezekben a fejezetekben bemutattam, hogy az elkészült szoftver a kitelepítést megkönnyítő technológiák használatával bármilyen számítógép architektúrán képes futni, amennyiben a fejezetekben tárgyalt körülmények (például Docker Engine) állnak fenn. Ennek az elérése prioritás volt a szoftver tervezése kapcsán.

Mindemellett a hálózatok témaköre is részben taglalásra kerül az említett fejezetekben, ugyanis a Docker Compose és a Kubernetes is létrehoz belső hálózatot, amin keresztül a konténerek, illetve a konténereket tartalmazó Kubernetes Podok kommunikálni tudnak egymással.

2.12.6 Operációs rendszerek

A szakdolgozati munkában megjelenő, operációs rendszerekkel kapcsolatos szakirodalmi háttérrel a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

Az Operációs rendszerek tantárgy megvalósult vonatkozásai pedig a [3.1.2.6 Konténerizáció, konténerizáltan futtatható állományok tárolása](#), a [3.1.2.7 Kitelepítést és futtatást támogató egyéb technológiák](#), a [3.2.6 Üzemeltetés](#) és a [3.3.6 Üzemeltetés](#) fejezetekben találhatók. Ezekben a fejezetekben bemutattam, hogy az elkészült szoftver a kitelepítést megkönnyítő technológiák használatával bármilyen operációs rendszeren képes futni, amennyiben a fejezetekben tárgyalt körülmények (például Docker Engine) állnak fenn. Ennek az elérése prioritás volt a szoftver tervezése kapcsán.

2.12.7 Programozási alapelvek és módszertanok

A szakdolgozati munkában megjelenő programozási alapelvek és módszertanok szakirodalmi háttérét a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

A szakdolgozati munkában használt programozási alapelvek és módszertanok a szoftvertervezésre vonatkozó fejezetben (Lásd [3.1 A szoftver tervezése](#)) és a tényleges programozásra kitérő fejezetekben (Lásd [3.2.1 Webfelület](#) és [3.2.2 Szerveroldal](#)) kerültek bemutatásra.

2.12.8 Rendszermodellezés

A szakdolgozati munkában megjelenő rendszermodellezés szakirodalmi háttérét a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

Lásd [3.1 A szoftver tervezése](#) fejezet, ahol részletesen kitérek a dolgozat alapjául szolgáló szoftver modellezésére.

2.12.9 Emberi viselkedés és kommunikáció

Az [1.4 Hasznosság](#) fejezetben kitértem az automatizálható feladatok és a munkahelyi kiegésző összefüggéseire, amely a tantárgy emberi viselkedés aspektusára vonatkoztatható. Mindemellett az emberi kommunikációnak is szerepe volt a szakdolgozati szoftver megírása során, hiszen a fejlesztési folyamat során használtam mesterséges intelligencia alapokon működő nyelvi modelleket (Copilot és ChatGPT), amelyekkel chatüzenetben kommunikáltam (prompt-ok megadása). Lásd 8. melléklet és 9. melléklet.

2.12.10 Felhasználói interfészek és vizualizáció

Lásd [3.1.1.1 Webfelület](#), [3.2.1 Webfelület](#) és [3.3.1 Webfelület](#) fejezeteket. A tantárgy szakirodalmi kapcsolódása pedig a [2.5 Webkliensek és kliensoldali technológiák](#) fejezetben található.

2.12.11 Vezetési és vállalkozási ismeretek

Nem kapcsolódik a szakdolgozathoz.

2.12.12 Rendszertervezés

A szakdolgozati munkában megjelenő rendszertervezés szakirodalmi háttérét a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

Lásd [3.1 A szoftver tervezése](#) fejezet, ahol részletesen kitérek a dolgozat alapjául szolgáló szoftver rendszertervére.

2.12.13 Programozás I., Programozás II., Programozás III.

A szakdolgozati munkában megjelenő programozási feladatok szakirodalmi háttérét a [2.2 Szerverek és szerveroldali technológiák](#), a [2.5 Webkliensek és kliensoldali technológiák](#), a [2.7 Verziókezelő rendszerek](#), valamint a [2.8 Adatbázisok](#) fejezetekben taglaltam.

Programozási feladatok természetesen a szoftver megvalósítása során merültek fel, elsődlegesen a webfelület és a szerveroldali fejlesztések során, illetve tágabb értelemben a szoftverüzemeltetésre való felkészítés kapcsán (konténerizációval és konténermenedzsmenttel kapcsolatos feladatok). Lásd [3.2.1 Webfelület](#), [3.2.2 Szerveroldal](#) és [3.2.6 Üzemeltetés](#) fejezeteket.

2.12.14 Komplex társadalomtudományi ismeretek

A dolgozat társadalomtudományi aspektusairól az [1.4 Hasznosság](#) és a [2.3 Magyarországi munkaerőköltség alakulása az IT iparágban](#) fejezetekben írtam. Lásd még 7. melléklet.

2.12.15 Vállalati gazdaságtan

A dolgozat vállalati gazdaságtan szakirodalmi aspektusairól az [1.4 Hasznosság](#) és a [2.3 Magyarországi munkaerőköltség alakulása az IT iparágban](#) fejezetekben írtam. Lásd még 7. melléklet.

A megvalósításról pedig a [3.1 A szoftver tervezése](#) és [Összegzés](#) fejezetekben lehet olvasni. Törekedtem arra, hogy a szoftver működési költségeit (a megbízóval kötött megegyezés mentén) minél alacsonyabban tartsam. A további potenciális üzemeltetési költségeket nem tekintve (amelyek nem voltak részei a megbízóval kötött megegyezésnek) ezt sikerült is teljesíteni, hiszen a szoftver ingyenesen elérhető eszközöket használ, ahogy erről az [Összegzés](#) fejezetben is írtam.

2.12.16 Adatbázisok I., Adatbázisok II.

A szakdolgozati munkában megjelenő, adatbázisokra vonatkozó feladatok szakirodalmi háttérét a [2.8 Adatbázisok](#) fejezetben taglaltam.

Lásd [3.1.1.4 Adatbázis](#), [3.2.4 Adatbázis](#), [3.3.4 Adatbázis](#) fejezeteket a megvalósulás megtekintéséhez, ezekben a fejezetekben tárgyaltam a dolgozat adatbázisokhoz kapcsolódó aspektusait.

2.12.17 Szoftverüzemeltetés

A szakdolgozati munkában megjelenő szoftverüzemeltetés szakirodalmi háttérét a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetben taglaltam.

Lásd [3.1.2.6 Konténerizáció, konténerizáltan futtatható állományok tárolása](#), [3.2.6 Üzemeltetés](#) és [3.3.6 Üzemeltetés](#) fejezeteket.

2.12.18 Szoftvertesztelés

„Az első és talán legfontosabb alapelv a teszteléssel kapcsolatban a nyomkövethetőség. Ez egyrészt fontos a hibák pontos helyének felderítése szempontjából, másrészt pedig a tesztek későbbi megismételhetősége miatt. Egy teszt nem tekinthető elfogadhatónak, ha később nem ismételhető meg. (...)

A következő alapelv, hogy a teszteket már jóval a teszt konkrét futtatása előtt meg kell tervezni. Ugyanúgy, ahogy elkészítjük a rendszertervet, el kell készíteni egy tesztervet is, ami magában foglalja a tesztesetek pontos leírását, és azt, hogy mi az elvárt eredmény a tesztelés során. Ettől eltérni csak nagyon kivételes esetben lehet. (...)

A Pareto elv kimondja, hogy a hibák 80%-a a kód 20%-ban fordul elő. Ez valószínűleg azon alapszik, hogy a hibák általában csoportosan, csomósodva fordulnak elő. (...)

Egy korábban említett alapelvet itt is megemlítünk, hiszen így lesz teljes a felsorolás. Ez az elv nem más, minthogy egyetlen modul vagy rendszer sem tesztelhető hiánytalanul, minden részletre kiterjedően.”(Horváth, 2007)

Szakirodalmi háttérrel biztosít még a tantárgy kapcsán a [2.8 Statikus kódelemzés](#) fejezet is, ugyanis a futtatás nélküli statikus analízis is kiszűrhet potenciális hibákat a megírt szoftverben, ahogyan ezt a hivatkozott fejezetben is taglalom.

Az idézett alapelvek gyakorlati alkalmazását a [3.3 Tesztelés](#) fejezetben fogom bemutatni, melyben részletesen tárgyalom a szoftver tesztelését.

2.12.19 Szoftverarchitektúrák

A szakdolgozati munkában megtalálható szoftverarchitektúrákról a szakirodalmi háttérrel a [2.2 Szerverek és szerveroldali technológiák](#), [2.5 Webkliensek és kliensoldali technológiák](#), valamint a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetekben taglaltam.

A gyakorlati megvalósításhoz: Lásd [3.1 A szoftver tervezése](#) fejezet, ahol részletesen kitérek a dolgozat alapjául szolgáló szoftver architektúrájára.

2.12.20 Innovatív információs és kommunikációs technológiák az IT-biztonság kapcsán

Az összefüggésvizsgálat első lépéseként megkérdeztem a Copilot nevű generatív mesterséges intelligenciát arról, hogy a tantárgy milyen módon kapcsolódhat a szakdolgozat témájával. Az alábbi listát generálta (Lásd 8. melléklet): „Néhány példa az innovatív ICT-re:

„Felhőalapú számítástechnika: Az olyan szolgáltatások, mint a Microsoft Azure, Amazon Web Services (AWS) és Google Cloud, lehetővé teszik a vállalatok számára, hogy skálázható és rugalmas IT-infrastruktúrát építsenek ki anélkül, hogy saját szervereket kellene fenntartaniuk.

Mesterséges intelligencia (AI) és gépi tanulás (ML): Az AI és ML alkalmazások képesek automatizálni a folyamatokat, előrejelzéseket készíteni és elemzéseket végezni, ami javíthatja a döntéshozatalt és a termelékenységet.

Internet of Things (IoT): Az IoT-eszközök összekapcsolják a különböző berendezéseket és rendszereket, lehetővé téve az adatok valós idejű gyűjtését és elemzését. Ezáltal hatékonyabbá válik az eszközök kezelése és a folyamatok optimalizálása.

5G és vezeték nélküli technológiák: Az 5G technológia gyorsabb és megbízhatóbb vezeték nélküli kapcsolatokat tesz lehetővé, ami támogatja az olyan alkalmazásokat, mint az okosvárosok, autonóm járművek és távorvoslás.

Big Data és adatvizualizáció: *A nagy mennyiségű adat elemzése és vizualizálása lehetővé teszi az adatokban rejlő értékek feltárását és az alapvető üzleti döntések meghozatalát.*”

Az alábbi IT szakmai blogbejegyzés (Awati, 2023) is a Copilot megállapításait támasztja alá:

„Az IKT magában foglalja az internetalapú és a vezeték nélküli hálózatok által működtetett mobil szférát. Ide tartoznak az olyan elavult technológiák, mint a vezetékes telefonok, a rádió és a televízió - amelyek mindegyike továbbra is széles körben használatos a mai IKT csúcstechnológiák, például a mesterséges intelligencia és a robotika mellett.

Az internet, a dolgok internete, a metaversum, a virtuális valóság és a közösségi média szintén az IKT részét képezik, akárcsak a felhőalapú számítástechnikai szolgáltatások, a videokonferencia- és együttműködési eszközök, az egységes kommunikációs rendszerek és a mobil kommunikációs hálózatok. Az olyan feltörekvő, folyamatban lévő vagy még csak kialakulóban lévő technológiák, mint az 5G/6G, a Web3 és a kvantumszámítástechnika szintén az IKT-univerzum részét képezik.” (Awati, 2023)

Ennek tükrében a szakirodalmi háttér a [2.1 Felhőalapú szolgáltatások](#) és a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) fejezetek biztosítják.

A gyakorlati megvalósításhoz: Lásd a [3.2 A szoftver megvalósítása](#) fejezetben található, IT-biztonságra vonatkozó alfejezetek ([3.2.1.8 IT-Biztonság](#), [3.2.2.8 IT-Biztonság](#), [3.2.3.1 IT-Biztonság](#), [3.2.4.1 IT-Biztonság](#) és [3.2.5.7 IT-Biztonság](#)) illetve a mesterséges intelligenciára vonatkozó alfejezeteket is ([3.2.1.9 Mesterséges intelligencia használata](#), [3.2.2.9 Mesterséges intelligencia használata](#), [3.2.3.2 Mesterséges intelligencia használata](#) és [3.2.5.8 Mesterséges intelligencia használata](#)).

2.12.21 IT-biztonsági fejlesztések minőség- és projektmenedzsmentje

A verziókezelő rendszerek használata képes egy bizonyos szintű projektmenedzsmentet és fejlesztési minőséget biztosítani (ahogyan ez a szakirodalmi kitekintésben is szerepel, Lásd [2.8 Verziókezelő rendszerek](#)), ennek okán gyakorlati megvalósításhoz: Lásd [3.2.5 Verziókezelés stratégia](#).

A gyakorlati megvalósítás IT-biztonsági aspektusaihoz: Lásd a [3.2 A szoftver megvalósítása](#) fejezetben található, IT-biztonságra vonatkozó alfejezetek ([3.2.1.8 IT-](#)

[Biztonság](#), [3.2.2.8 IT-Biztonság](#), [3.2.3.1 IT-Biztonság](#), [3.2.4.1 IT-Biztonság](#) és [3.2.5.7 IT-Biztonság](#)).

2.12.22 Mesterséges intelligenciák az IT biztonság területén

Az elmúlt éveket jellemző, a generatív mesterséges intelligenciákat érintő technológiai áttörések tükrében felelőtlenség volna, ha egy piaci viszonyokra készülő szoftver (amelynek markáns IT-biztonsági vonatkozásai is vannak) nem veszi igénybe az említett generatív MI eszközöket. Ez már csak a fejlesztést végző szakember(ek) vonatkozásában is fontos lehet, ahogyan erről Dr. Rikk János és Dr. Pitlik László is írnak a vizsgálatuk következtetésében: „*A végkövetkeztetés nem lehet más, mint az, hogy ezek a kompetenciák, melyek itt és most tetten érésre kerültek, mindenképpen integrálandók az emberekéivel. Az a humán egyed, aki nem tud jobbra válni a ChatGPT hatására végső soron akár pl. érettségét sem illene, hogy szerezhessen.*” (Dr. Rikk & Dr. Pitlik, 2023)

A gyakorlati megvalósításhoz: Lásd a [3.2 A szoftver megvalósítása](#) fejezetben található, IT-biztonságra vonatkozó alfejezetek ([3.2.1.8 IT-Biztonság](#), [3.2.2.8 IT-Biztonság](#), [3.2.3.1 IT-Biztonság](#), [3.2.4.1 IT-Biztonság](#) és [3.2.5.7 IT-Biztonság](#)) és mesterséges intelligencia felhasználására vonatkozó alfejezetek ([3.2.1.9 Mesterséges intelligencia használata](#), [3.2.2.9 Mesterséges intelligencia használata](#), [3.2.3.2 Mesterséges intelligencia használata](#) és [3.2.5.8 Mesterséges intelligencia használata](#)) együttes értelmezése.

2.12.23 Tudásmenedzsment az IT biztonság területén

A gyakorlati megvalósításhoz: Lásd a [3.2 A szoftver megvalósítása](#) fejezetben található, IT-biztonságra vonatkozó alfejezetek ([3.2.1.8 IT-Biztonság](#), [3.2.2.8 IT-Biztonság](#), [3.2.3.1 IT-Biztonság](#), [3.2.4.1 IT-Biztonság](#) és [3.2.5.7 IT-Biztonság](#)).

2.12.24 Az elektronika fizikai alapjai

Nem kapcsolódik a szakdolgozathoz.

2.12.25 Elektronikus áramkörök

Nem kapcsolódik a szakdolgozathoz.

3 Saját fejlesztés bemutatása

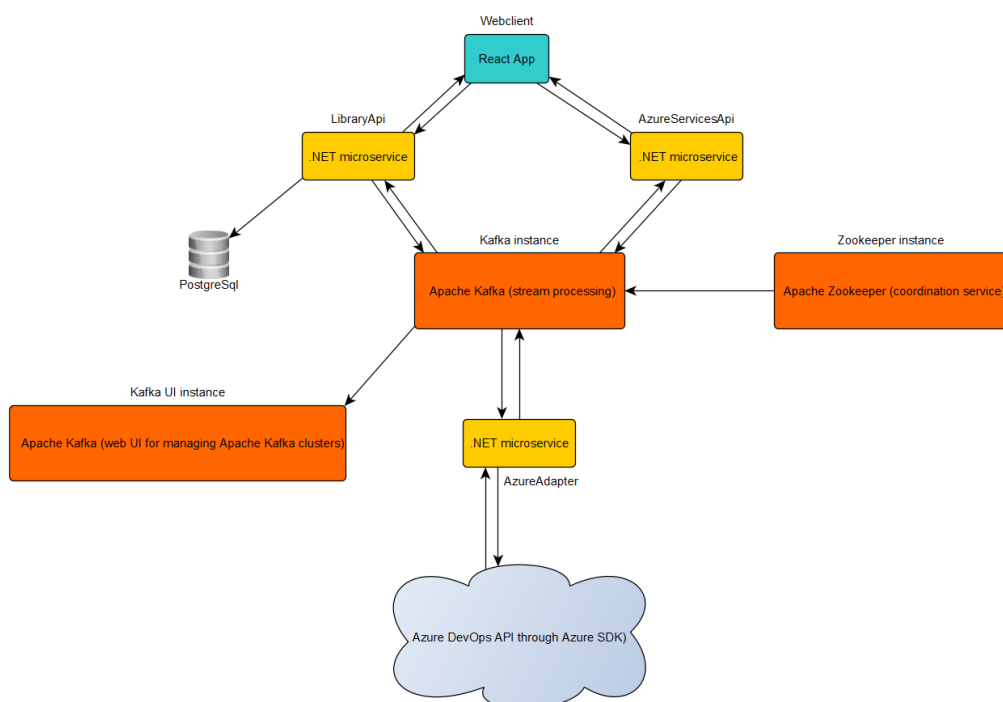
Ebben a főfejezetben részletesen bemutatom a szakdolgozati téma vonatkozásában elvégzett munka különböző stációit.

3.1 A szoftver tervezése

Elsődleges célom volt egy olyan költséghatékony és modern szoftver létrehozása, amely a már megfogalmazott felhasználói/megrendelői igényeket (Lásd [1.6.4.1 Megrendelői elvárások](#)) is kielégíti, illetve képes igazodni a jövőbeli célok és igények eléréséhez is.

Az előző bekezdésben megfogalmazott elvárások eléréséhez elengedhetetlen, hogy a céloknak megfelelő szoftverarchitektúrát válasszam és implementáljam.

Ennek megfelelően egy, a kitűzött céloknak (Lásd [1.6.4.1 Megrendelői elvárások](#)) megfelelő szoftver egy webfelületből, a webfelülettel és az egymással is kommunikáló szerveroldali komponensekből, ezen szerveroldali kommunikációt biztosító eseményfolyam-kezelő platformból és adatbázispéldányból kell állnia a már taglalt szakirodalmi áttekintés alapján (Lásd 11. ábra).



11. ábra: Szoftverarchitektúra (saját szerkesztés)

A 11. ábrához jelmagyarázat is tartozik, ugyanis az eltérő színeknek, alakzatoknak és nyilaknak van jelentéstartalma. Az alakzatok elhelyezkedése kizárólag a könnyebb értelmezés

irányvonala mentén történt meg, így ez tekinthető egy önkényes rendezésnek. Az alakzatok színeinek, formájának és a nyilak irányvonalának megtartásával más elrendezésben is létrehozhattam volna a szoftverarchitektúra ábráját.

A fenti ábrán külön színjelöléssel jelöltem az eltérő technológiában íródott komponenseket. A narancssárga szín jelöli a Java programnyelvben íródott szoftvereket. A sárga szín jelöli a .NET keretrendszerben íródott C# projekteket. A kék szín jelöli a NodeJS alapon futó, JavaScript programnyelvben íródott szoftvert. Az alsó halványkék, felhőalakú rész jelöli a publikus, ökoszisztémán kívüli internetet. A szürkés, hordószerű ikon a PostgreSQL adatbázist jelenti.

A nyilak a két komponens közötti kommunikáció meglétét és a kommunikáció irányait jelöli. Mivel ez egy egyszerűbb, szemléltető ábra, így külön nem részleteztem az ábrán a kommunikáció természetét (például milyen protokoll társul az adott kommunikációhoz).

Az alakzatokhoz tartozó szövegek nem egységesek abban az értelemben, hogy bizonyos komponensekhez részletesebb magyarázat szükségeltetett az ábra egészének értelmezéséhez. Az alakzat határain belülre olyan információk kerültek, amelyek elsődlegesen a szoftver technológiai természetére vonatkoznak. Az alakzaton kívülre olyan információ került, ami az alakzatot betöltött funkciójában értelmezi.

3.1.1 Szoftvert alkotó komponensek

3.1.1.1 *Webfelület*

A szakirodalmi háttér áttekintése után (Lásd [2.5 Webkliensek és kliensoldali technológiák](#)) a kliensoldali (webfelület) megvalósításhoz a ReactJS (React Community (Meta), 2025) keretrendszert választom, amely komponensalapú, dinamikus felhasználói felületek készítésére és kezelésére is alkalmas. Két fő oka van annak, amiért ezt a Javascript keretrendszert használtam:

- A ReactJS technológia integrált, modern frontendfejlesztési megoldást kínál, funkcionalitásai okán pedig (pl.: virtuális DOM, komponensalapú működés) a weboldal performanciája, illetve a funkciókra és hibajavításokraallokált fejlesztési idő is optimalizált (Suraj Surve, 2021; Devinder Suthwal, 2020).
- Ebben a keretrendszerben volt létező, több éves fejlesztői tapasztalatom, így a termék kliensoldali komponens elkészítési ideje szignifikánsan redukálhatóvá vált. Mindemellett kiemelném, hogy a technológia tanulási görbéje és részletes

dokumentációja is a keretrendszer használatára ösztönzi a modern webfejlesztésben elmélyedni tervező fejlesztőket.

Az elkészült React applikáció konténerizáltan futtatható a Docker használatával (Mirantis Inc., 2025). Ezen technológia használatával a webprojekt futtatható bármilyen eszközön, amelyen a Docker elérhető, tekintet nélkül az operációs rendszerre vagy processzor szignatúrára. A konténerizációnak köszönhetően a webalkalmazás szükség szerint skálázható (Elton Stoneman, 2020).

3.1.1.2 *Szerveroldal*

A szakirodalmi háttér áttekintése után (Lásd [2.4 Szerverek és szerveroldali technológiák](#)) a szerveroldali fejlesztések C# programnyelven, a .NET keretrendszer használatával lesznek elkészítve, microservice architektúra implementálásával a skálázhatóság és a kognitív komplexitás redukálása érdekében (Microsoft Inc., 2025b).

„Egy modul egy, és csakis egy szereplőnek lehet felelős.” (Martin, 2018)

Az SRP (Single Responsibility Principle) elvét szem előtt tartva egy microservice felel az Azure DevOps konnektivitásért (Martin, 2018). Ezt a megközelítést a jövőbeni megrendelt fejlesztések esetén is meg fogom tartani, például új felhőszolgáltatóhoz (például AWS) történő csatlakozás esetén is.

Az elkészült microservice architektúrájú .NET projektek is konténerizáltan fognak futni a Docker használatával (Mirantis Inc., 2025). Ennek köszönhetően ezek a konténerek is futtathatóak lesznek bármilyen szerveren, amelyen a Docker elérhető, tekintet nélkül az operációs rendszerre vagy processzor szignatúrára (Elton Stoneman, 2020). Többek között a konténerizációnak is köszönhető, hogy a .NET komponensek is szükség szerint skálázhatóak.

3.1.1.3 *Eseményfolyam-kezelő platform*

Az eseményfolyam-kezelésről szóló szakirodalmi kitekintés a [2.5 Eseményfolyam-kezelő technológiák](#) fejezetben található.

A szerveroldali komponensek közötti kommunikációt az Apache Kafka nevű nyílt forráskódú elosztott eseményfolyam-kezelő platform fogja megvalósítani. Azért esik a választásom ennek a platformnak a használatára, mert a felhasználása ingyenes, illetve a szoftver eseményfolyam-kezelő technológiája alkalmassá teszi a kommunikáció skálázhatóságára, ellentétben a HTTP alapú kommunikációval. Mindemellett magas rendelkezésreállás és gyors üzenetküldés jellemzi ezt a szoftvert (LinkedIn Inc., 2025).

A platform működése szempontjából szükséges az Apache Zookeeper egy futtatható példányának elérése is a Kafka szolgáltatás számára, ugyanis a Zookeeper kontrollálja és karbantartja a vezető-követő kapcsolati hálózatot a Kafkában található partíciók között. Emellett a feladatai közé tartoznak még a különböző Kafka topikok konfigurálása és karbantartása is (Elin Vinka, 2018).

A leszállított termékhez tartozik még egy olyan (Docker által konténerizáltan) futtatható komponens is, amely alkalmas arra, hogy az Apache Kafka szolgáltatáson keresztül megvalósult kommunikációt monitorozni lehessen. Az ebben rejlő lehetőségek kiaknázása nem tárgya a szakdolgozati munkának, de a jövőben szándékomban áll optimalizálni a szerverkomponensek közötti kommunikációt ennek a monitoring eszköznek a felhasználásával (Provectus, 2025).

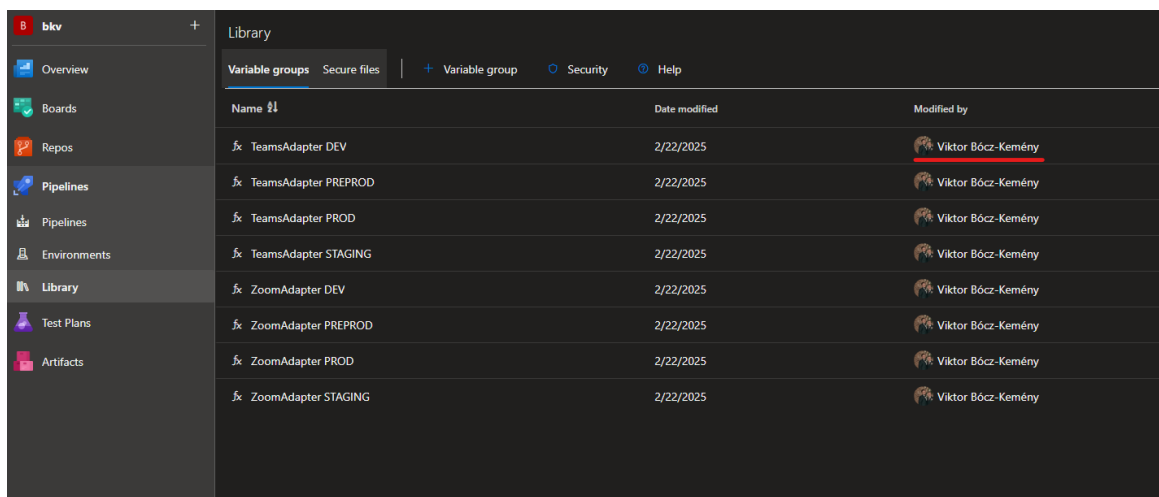
„A Kafka üzenetméret-korlátozás az a maximális méret, amellyel egy üzenet sikeresen eljuthat egy Kafka topic-ba. Az Apache Kafka alapértelmezés szerint kis méretű üzenetekre van optimalizálva, de konfigurációs lehetőségeket is biztosít az üzenetméretek beállításához. A Kafka alapértelmezett üzenetméret-korlátja 1 MB-ra van beállítva. Ennek a korlátozásnak az a célja, hogy a hálózati torlódások csökkentése és a rendszer erőforrásainak kímélése révén fenntartsa a magas átviteli teljesítményt.” (Apache Kafka Message Size Limit: Best Practices & Config Guide, 2025)

Tehát a fenti hivatkozásból következik, hogy az Apache Kafka maximális átviteli kapacitása üzenetenként 1 MB, konfiguráció módosítás esetén több is lehet.

3.1.1.4 Adatbázis

Mivel kiemelt szempont volt a költséghatékonyság és a nyílt forráskódú szoftverek használata a szoftver megtervezésekor, így adatperzisztáláshoz a PostgreSQL-re esik a választásom (Szakirodalmi kitekintés a [2.8 Adatbázisok](#) fejezetben). Korábban volt tapasztalatom ezzel az SQL dialektussal, így a használata nem esett nehezemre.

Az Azure DevOps nagy hiányossága, hogy a környezeti változókon és secreteken elvégzett módosításokat nem olyan minőségben és részletességben követi nyomon, mint ahogyan az felhasználói oldalról elvárható lenne. A felületen mindösszesen annyi jelenik meg, hogy ki módosította utoljára az adott környezeti változó csoportot. Az nem derül ki ennyiből, hogy melyik változót milyen értékről mire módosítottam (Lásd 12. ábra).



Library			
Variable groups Secure files + Variable group Security Help			
Name	Date modified	Modified by	
TeamsAdapter DEV	2/22/2025	Viktor Bócz-Kemény	
TeamsAdapter PREPROD	2/22/2025	Viktor Bócz-Kemény	
TeamsAdapter PROD	2/22/2025	Viktor Bócz-Kemény	
TeamsAdapter STAGING	2/22/2025	Viktor Bócz-Kemény	
ZoomAdapter DEV	2/22/2025	Viktor Bócz-Kemény	
ZoomAdapter PREPROD	2/22/2025	Viktor Bócz-Kemény	
ZoomAdapter PROD	2/22/2025	Viktor Bócz-Kemény	
ZoomAdapter STAGING	2/22/2025	Viktor Bócz-Kemény	

12. ábra: Variable group utolsó módosító (saját szerkesztés)

A 12. ábrán kiemeltém azt a szegmenst, ahol az utolsó módosító neve látható az Azure DevOps felületén. Az Azure DevOps felülete ennyi információt biztosít a felhasználó számára a korábbi módosítással kapcsolatban.

A szakdolgozati munka egyik célja (Lásd [1.6.4.1 Megrendelői elvárások](#)), hogy a módosítások monitorozása a webes felületen megtörténjen, ennek részleteit [A szoftver megvalósítása](#) fejezetben fogom részletezni.

3.1.2 Szoftverfejlesztést és üzembehelyezést támogató eszközök

3.1.2.1 Verziókezelés

Ahogy erre már a szakirodalmi kitekintésben is kitértem (Lásd [2.7 Verziókezelő rendszerek](#)), a verziókezelésnek köszönhetően a fejlesztő képessé válik arra, hogy a forráskódjából különböző verziókat, elágazásokat hozzon létre, így megőrizve egy új fejlesztés esetén is a meglévő funkciók működőképességét. Mindemellett a disztribúció és a kollaboráció lehetőségei is igen fontos képességei a verziókezelésnek, ezen funkciók erősíthetik az IT-biztonságot és a minőségbiztosítást is.

Véleményem szerint szükség van verziókezelésre egy professzionális, piacra termelő fejlesztés esetén, így ennek a használata egyértelmű volt a számomra. Verziókezeléshez a Git nyílt forráskódú, elosztott verziókezelő szoftvert választottam. Főként a népszerűsége és ingyenes felhasználhatósága miatt választottam a Git-et, ezen funkciókról a Git dokumentációban tájékozódhat az érdeklődő részletesebben (Chacon & Straub, 2025). Ezt a szoftvert több éve használom, így van szerzett tapasztalatom is, amit a fejlesztési idő lerövidítésében kamatoztathattam.

A Git verziókezelés mellett Git branch stratégiát is választani kell, én a Git Flow stratégiát fogom használni.

„A Git flow egy népszerű Git branching stratégia, amelynek célja a release egyszerűsítése, és amelyet Vincent Driessen szoftverfejlesztő vezetett be 2010-ben. Alapvetően a Git flow lényege, hogy a munkát különböző típusú Git branch-ekre különítjük el.” (Git Kraken, 2022)

A Git Flow stratégia egyszerű koncepciót biztosít számomra a verziókezelést illetően, felépítésében egyértelműsíti a különböző Git branch-ek eltérő funkcionalitását és felelősségi köreit.

3.1.2.2 *Forráskódtárolás*

A megrendelő részéről elvárás volt, hogy a szoftver zárt forráskódú legyen (Lásd [1.6.4.1 Megrendelői elvárások](#)), így a forráskód tárolására olyan megoldást kellett keressek, ahol ez lehetséges (akár az ingyenes verzió keretében is, amennyiben létezik ilyen). Mindemellett a platformfüggetlenség is szempont volt, így a felhőszolgáltatók által kínált, integrált kódtárolási szolgáltatások sem voltak opciók az esetemben.

A GitHub ingyenesen biztosítja az előző bekezdésben megfogalmazott elvárásokat, ahogyan ez a GitHub Education weboldalon is részletesen olvasható (Intro to GitHub - GitHub Education, 2025), így ennek tudatában ezt a platformot választottam a forráskód tárolására. A választott verziókezelő szoftver (Git) kapcsán egyébként is adta magát, hogy a Git-hez integrálódó, platformfüggetlen szolgáltatót válasszak.

3.1.2.3 *Statikus kódanalízis*

Statikus kódanalízis használata nem volt elvárás a megrendelői oldalról (Lásd [1.6.4.1 Megrendelői elvárások](#)), viszont a szakirodalmi háttérből levonható következtetések (Lásd [2.8 Statikus kódelemzés](#)) szerint a szoftverfejlesztés során értékes információkat lehet kinyerni a statikus kódelemzés eredményei alapján, így a program futtatása nélkül találhat meg a fejlesztő potenciális hibákat és ezáltal a szoftverminőséget is javíthatja. Ennek okán a használata hasznos lehet a szoftver megvalósítása vonatkozásában.

Statikus kódelemzéshez a SonarCloud-ot fogom használni, ugyanis a felhasználása ingyenes és a GitHub is támogatja a beépített CI/CD pipeline esetén a szoftverrel történő integrációt. A SonarCloud dokumentációja (SonarSource Inc., 2025) részletezi a CI/CD folyamat összekapcsolhatóságát annak érdekében, hogy egyéni kódelemzői stratégiát alakíthasson ki a platformot használó szakember.

3.1.2.4 Folyamatos Integráció, Folyamatos Szállítás tervezés

A szakirodalmi háttér a CI/CD vonatkozásában a [2.11 Folyamatos Integráció, Folyamatos Szállítás](#) fejezetben taglaltam.

Ahogy az a korábbi fejezetben is írtam, a GitHub biztosít beépített CI/CD szolgáltatást, így kézenfekvő lenne ennek (díjmentes) felhasználása. Mivel korábban nem írtam CI/CD pipeline-t, így a vonatkozó GitHub blogposzt (Douglas, 2025) alapján fogom implementálni a workflow-t.

3.1.2.5 Könyvtárcsomagok kezelése

„A probléma megoldásának talán legkézenfekvőbb módja az adatok és a funkciók szétválasztása. A három osztály megosztja a hozzáférést az EmployeeData adathoz, ami egy egyszerű adatszerkezet, amely nem tartalmaz metódusokat (7.3. ábra). Mindegyik osztály csak az adott funkcióhoz szükséges forráskódot. A három osztály nem tudhat egymásról. Így minden véletlen duplikáció elkerülhető.” (Martin, 2018)

Ahogy az Robert C. Martin könyvében (Martin, 2018) és a fenti idézetből is olvasható is olvasható, a minőségi szoftver megalkotásához elengedhetetlen, hogy a fejlesztő törekedjen a kódduplikáció és kódredundancia minimalizálására. A szerveroldali fejlesztés esetén felmerült, hogy bizonyos kódrészleteket újrahasznosíthatóvá tudok tenni, így adta magát a kérdés, hogy milyen csomagkezelő megoldást válasszak erre a feladatra.

A NuGet csomagkezelő szoftver ingyenes és kifejezetten .NET alapú könyvtárcsomagok létrehozására és felhasználására alkalmas. Mindemellett a szoftvert Microsoft ökoszisztémára optimalizálták és maga a Microsoft a karbantartó vállalat, ahogy ez a szoftver dokumentációjában is olvasható (Microsoft Inc., 2025a).

3.1.2.6 Konténerizáció, konténerizáltan futtatható állományok tárolása

A konténerizációról és a konténerizáció területén vezető technológiának számító Docker platformról már a [2.6 Konténerizáció: üzemeltetés és karbantartás](#) szakirodalmi fejezetben már értekeztem. A Docker használatával a korábbi fejezetekben megfogalmazott irányelveket (költséghatékonyság, nyílt forráskódú és széleskörű fejlesztői közösség által támogatott szoftver felhasználása) meg lehet valósítani a konténerben futtathatóság esetében is. Mindemellett a platformfüggetlenséget is sikerülhet elérni a Docker használatával, ugyanis a konténerizált szoftver minden olyan ökoszisztémán futtatható, amelyen elérhető a Docker Engine. Ezen érvek mentén a konténerizáció megvalósításához a Docker platformot választom.

Elton Stoneman könyvét (Elton Stoneman, 2020) a Docker dokumentációját tartalmazó hivatalos weboldal (*Docker Docs*, 2025) is ajánlja, így a fejlesztés megkezdése előtt több fejezetet is elolvastam vonalvezetés gyanánt.

Mivel a DockerHub az alapértelmezett szolgáltató, így én is ezt a platformot tervezem használni a projekt megvalósításához. A DockerHub kínál fizetős és ingyenes szolgáltatásokat, de a szakdolgozat esetében (a projekt volumenét és komplexitását tekintve) elegendő lesz az ingyenes verzió használata is.

3.1.2.7 Kitelepítést és futtatást támogató egyéb technológiák

Ebben a szekcióban azokról a technológiákról és azoknak a szakdolgozati projektbe történő beemeléséről írok, amelyek nem tartalmi elemei a megrendelővel kötött szerződésnek (Lásd [1.6.4.1 Megrendelői elvárások](#)). A megrendelő nem rendelte meg ezeket a funkcionalitásokat, ugyanis a szoftver üzemeltetése a megrendelői feladatkörbe tartozik a megállapodás szerint. Véleményem szerint viszont az elkészült projekt vállalati szintű terméké válásához elengedhetetlen, hogy a lenti két alfejezetben felmerülő kérdésekre/problémákra a szakdolgozati munka is megoldást tudjon nyújtani.

Docker Compose

Annak érdekében, hogy a konténerek létrehozását ne manuálisan kelljen felparaméterezni és végrehajtani, keresnem kellett egy olyan technológiát, amely támogatja a több konténerből álló szoftverek kitelepítésének egyszerűsítését, automatizációját. A Docker Compose technológiára esett a választásom (Szakirodalmi háttérért lásd [2.7.2 Docker Compose](#)).

A Compose automatikusan létrehoz egy Docker hálózatot (docker network), amin keresztül képessé válnak a konténerek kommunikálni egymással. Ez a funkcionalitás természetesen paraméterezhető.

Kubernetes

A szakirodalmi háttér tanulmányozása során (Lásd [2.6.2 Kubernetes](#)) arra a következtetésre jutottam, hogy szakmailag fontos a konténerek elindításán és futtatásán kívül a konténerek menedzselésének biztosítása is, mert ahogyan erre a hivatkozott forrás is kitér, a Kubernetes képes gondoskodni „(...) a konténerizált alkalmazások indításáról, a frissítések menedzseléséről, a szolgáltatási szintek fenntartásáról, az igényeknek megfelelő skálázásról (...)” (Elton Stoneman, 2021)

A döntésben fontos szerepet játszottak a szakirodalmi kitekintőben felsorolt körülmények is a Kuberneteset illetően, mint például:

- nyílt forráskódú a szoftver
- ingyenesen elérhető
- széleskörű közösség tartja karban a platformot

3.1.3 Felhasznált technológiák összesítése

Ebben az alfejezetben összesítem a szakdolgozati projekthez használni tervezett technológiákat. Az 1. táblázatban feltüntettem a felhasználás célját és a használt verziószámot is.

Technológia/ szoftvereszköz	Felhasználás célja	Verzió	Indoklás
ReactJS	Webfelület	v22.13.1	Utolsó stabil verzió (forrás: https://nodejs.org/en/download)
.NET keretrendszer	Szerveroldal	v8.0	Utolsó LTS (long time support) verzió használata (forrás: https://dotnet.microsoft.com/en-us/download/dotnet), továbbá IT-biztonsági szempontok (non-root felhasználó, forrás: (Lander, 2023))
Git	Verziókezelés	v2.48.1	Utolsó stabil verzió (forrás: https://git-scm.com/downloads/win)
Docker Engine	Konténerizáció	v28.0.1	Utolsó stabil verzió (forrás: https://docs.docker.com/desktop/release-notes/)
Docker Compose	Multikonténer applikációkezelés	v2.33.1	Utolsó stabil verzió (forrás: https://docs.docker.com/desktop/release-notes/)
Kubernetes	Konténerkezelő	kliens: v1.30.5, szerver: v1.32.0	Utolsó stabil verzió (forrás: https://kubernetes.io/releases/)
PostgreSQL	Adatbázis	Utolsó verzió	Dockerfile-ban definiált tag okán a keretrendszer az utolsó stabil verziót használja (Lásd 3.2.6 Üzemeltetés).
Apache Kafka	Eseményfolyam- kezelő	Utolsó verzió	Dockerfile-ban definiált tag okán a keretrendszer az utolsó stabil verziót használja (Lásd 3.2.6 Üzemeltetés).
Apache Zookeeper	Eseményfolyam- kezelő kontrollig	Utolsó verzió	Dockerfile-ban definiált tag okán a keretrendszer az utolsó stabil verziót használja (Lásd 3.2.6 Üzemeltetés).

1. táblázat: Felhasznált technológiák (saját szerkesztés)

3.2 A szoftver megvalósítása

Ebben a fejezetben a szoftver megvalósítására fogok kitérni, taglalva a szoftver különböző rétegeit és elemeit, emellett bemutatva fejlesztési folyamat során alkalmazott eszközöket is (például: CI/CD folyamat, statikus kódelemzés).

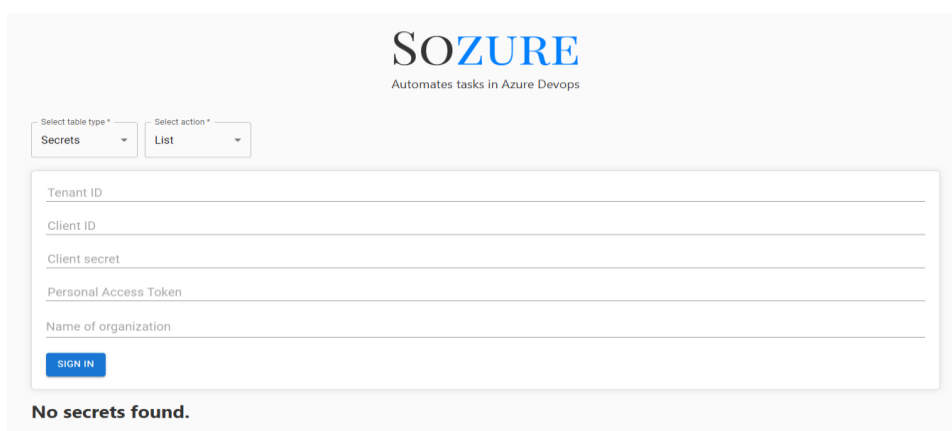
Ahogy erről az [1.6.4.2 Kutatási megkötések, avagy a szakdolgozatból tudatosan kihagyott elemek](#) alfejezetben is írtam, a szoftver felhasználása feltételezi és meg is követeli a megrendelőtől, avagy tágabb értelemben véve a felhasználótól az Azure DevOps ismeretét.

3.2.1 Webfelület

Először a webfelületet fogom bemutatni, kitérve a dedikált felelősségi köreiket ellátó komponensekre. Az elsődleges cél az elkészített webfelület ismertetése, a funkciók szerveroldali működésére a [3.2.2 Szerveroldal](#) alfejezetben térek ki.

3.2.1.1 Bejelentkezés

A szoftver használatának első lépésénél a weboldal egy bejelentkező felülettel fogadja a felhasználót (Lásd 13. ábra és 14. ábra).

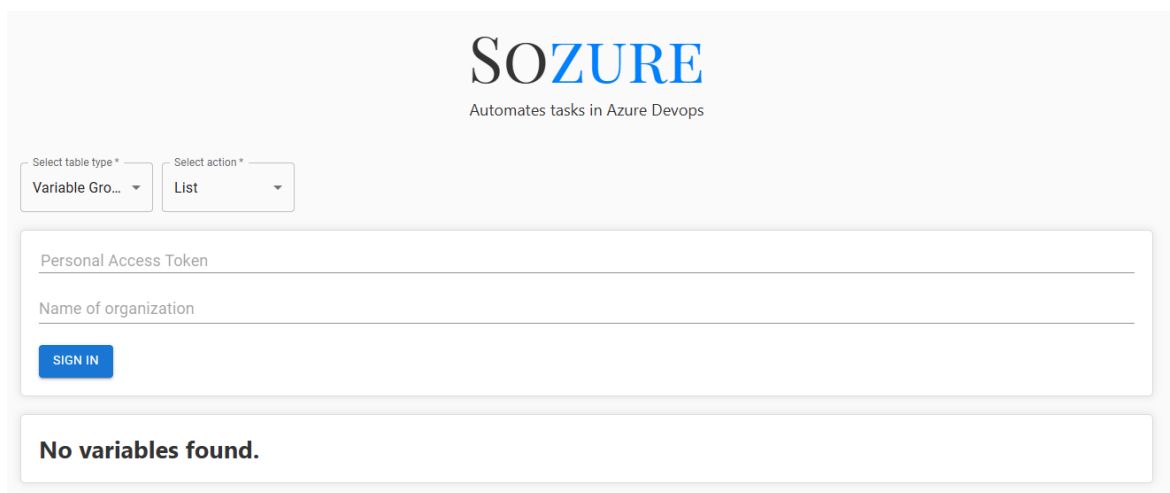


13. ábra: Secret bejelentkező felület (saját szerkesztés)

Jelmagyarázat a 13. ábrához:

- Table type: funkcionalitás témacsoport és a hozzá tartozó adatlap kiválasztása
- Action: funkcionalitás és a hozzá tartozó adatlap kiválasztása
- Tenant ID: Bérő azonosító, Azure organizációhoz tartozó Azure Portal hitelesítő adata
- Client ID: Kliens azonosító, Azure organizációhoz tartozó Azure Portal hitelesítő adata
- Client Secret: Kliens titok, Azure organizációhoz tartozó Azure Portal hitelesítő adata
- Personal Access Token: Személyes token, egyfajta jelszó, amit a felhasználóhoz kell generálni
- Name of organization: Azure organizáció
- Sign in: Bejelentkezést végrehajtó gomb

Az Azure secret és key vault funkcionalitásokhoz szükségesek a fenti adatok, ezen adatok megadásával lehet a szerveroldalon elvégzett lekérdezésekkel végrehajtani a bejelentkezést.



The screenshot shows the SOZURE application interface. At the top, the logo 'SOZURE' is displayed in blue, with the tagline 'Automates tasks in Azure Devops' below it. Below the logo, there are two dropdown menus: 'Select table type *' with 'Variable Gro...' selected, and 'Select action *' with 'List' selected. Below these, there are two input fields: 'Personal Access Token' and 'Name of organization'. A blue 'SIGN IN' button is positioned below the 'Name of organization' field. At the bottom of the form, a message states 'No variables found.'

14. ábra: Környezeti változó bejelentkezési felület (saját szerkesztés)

Jelmagyarázat a 14. ábrához:

- Personal Access Token: Személyes token, egyfajta jelszó, amit a felhasználóhoz kell generálni
- Name of organization: Azure organizáció
- Sign in: Bejelentkezést végrehajtó gomb

A környezeti változók lekérdezéséhez és manipulálásához (az Azure által biztosított API (Application Programming Interface) logikájának köszönhetően) a secret és key vault funkcionalitásokhoz szükséges adatok töredékét is elegendő megadni. Ezen adatok megadásával lehet a szerveroldalon elvégzett lekérdezésekkel végrehajtani a bejelentkezést.

A szükséges belépési adatokat a felhasználó az Azure DevOps organizáció tulajdonosától tudja beszerezni, illetve szükség esetén generáltatni.

Fontos megemlíteni, hogy ezen bejelentkeztetési logika is erősíti a szoftver IT-biztonsági aspektusát, elkerülve az illetéktelen manipulációkat az Azure organizáción belül. Mindazonáltal a felelősség a felhasználóé, hogy miként őrzi meg az Azure-ben létrehozott hitelesítő adataira. Az elkészült szoftver nem tárolja a felhasználók hitelesítő adatait.

3.2.1.2 Környezeti változók lekérdezése, manipulálása

A szükséges hitelesítő adatok megadása után a következő felület fogadja a felhasználót (Lásd 15. ábra).

The screenshot shows the SOZURE web application interface. At the top, the logo "SOZURE" is displayed with the tagline "Automates tasks in Azure Devops". Below the logo, there are two dropdown menus: "Select table type *" with "Variable Gro..." selected, and "Select action *" with "List" selected. A message bar says "Hi Viktor Bócz-Kemény! To change Azure organization, click here:" followed by two buttons: "BACK TO AUTHENTICATION" and "SEE CHANGES HISTORY". The main form area contains a "Select Azure project" dropdown with "bkv" selected. Below this are three input fields: "Name (regex) of variable group", "Key of variable", and "Value (regex) of variable [OPTIONAL]". There are two checkboxes: "Secret included" and "Key's name should be regex". A "SEND REQUEST" button is located at the bottom right of the form. A message box at the bottom states "No variables found."

15. ábra: Környezeti változók lekérdezése, adatlap (saját szerkesztés)

Jelmagyarázat a 15. ábrához:

- Legfelső téglalapban megjelenő szöveg: felhasználó köszöntése a bejelentkezést követően
- Legfelső téglalapban megjelenő gombok: baloldali visszairányít a bejelentkezés adatlapra, a jobboldali elirányít a módosítási naplóbejegyzések oldalra
- Azure project: Azure projekt neve
- Name (regex) of variable group: Variable group neve
- Key of variable: Környezeti változó neve
- Value (regex) of variable [OPTIONAL]: Környezeti változó értéke
- Secret included: Variable group-pal összekapcsolt secret megjelenítése, igen/nem
- Key's name should be regex: Környezeti változó nevének reguláris kifejezésként történő megadása, igen/nem
- Send request: Lekérdezést kiváltó parancs kiadása a gomb megnyomása esetén

A 15. ábrán látható, hogy a felület köszönti a felhasználót és fel is ajánlja a lehetőséget a bejelentkezési felületre való visszatérésre, illetve a módosítási napló megtekintésére. A módosítási naplóhoz tartozó funkcionalitásról a [3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése](#) alfejezetben írok.

Környezeti változó lekérdezése

A lekérdezés elvégzéséhez szükséges lesz megadni a következő paramétereket (Lásd 15. ábra):

- Azure projekt neve
 - Azure projekt: Azure organizáción belül létező, forráskód tárolásáért, CI/CD folyamatokért és kitelepítésért felelős Microsoft Azure entitás
- Variable group neve
 - Reguláris kifejezés (RegEx) alkalmazható
- Környezeti változó neve
 - Reguláris kifejezés (RegEx) alkalmazható, amennyiben az ábrán látható adatlap két bepípálható checkbox eleme közül a másodikat bepípálja a felhasználó
- Környezeti változó értéke
 - Opcionális paraméter
 - Reguláris kifejezés (RegEx) alkalmazható
- Variable group-pal összekapcsolt secret megjelenítése vagy sem
 - Ábrán látható adatlap két bepípálható checkbox eleme közül az első

- Fontos: itt csak a secret összekapcsoltsága állapítható meg, a secret értékét a secret menüpont alatt lehet megtekinteni!

Példa lekérdezést elvégeztem annak érdekében, hogy a megjelenő eredményfelületet is bemutathassam (Lásd 16. ábra):

Select Azure project
bkv

.*

.*

Value (regex) of variable [OPTIONAL]

☐ Secret included

☒ Key's name should be regex

SEND REQUEST

Matched variables (Found variables: 16)

Project	Variable Group Name	Variable key	Variable value	Operation
bkv	TeamsAdapter PROD	DatabaseProvider	PostgreSql	[Copy] [Edit] [Delete]
bkv	TeamsAdapter PROD	Logging	Serilog	[Copy] [Edit] [Delete]
bkv	TeamsAdapter PREPROD	DatabaseProvider	PostgreSql	[Copy] [Edit] [Delete]
bkv	TeamsAdapter PREPROD	Logging	Serilog	[Copy] [Edit] [Delete]
bkv	TeamsAdapter DEV	DatabaseProvider	PostgreSql	[Copy] [Edit] [Delete]

1-5 of 16 < >

16. ábra: Környezeti változók lekérdezése, eredmény (saját szerkesztés)

Jelmagyarázat a 16. ábrához:

- A 16. ábrán is megtalálható elemek megegyeznek jelentésükben a 15. ábrához tartozó jelmagyarázatban szereplőkkel
- Matched variables (Found variables: x): A beadott paraméterek alapján visszatérő eredmény számossága
- Project: Azure projekt neve
- Variable group name: Variable group neve
- Variable key: Környezeti változó neve
- Variable value: Környezeti változó jelenlegi értéke
- Operation: További tevékenységeket (érték kimásolása vágólapra, módosítás, törlés) indítani képes gombok

A megjelenő eredményfelület mutatja, hogy összesen hány találatot kap a felhasználó a lekérdezése alapján, illetve a táblázatban megjeleníti az adott környezeti változó alábbi adatait:

- Azure projekt neve
- Variable group neve
- Környezeti változó neve
- Környezeti változó értéke

Lehetőség van az adott sorhoz tartozó gombok megnyomásával módosítani és törölni is az adott sorhoz tartozó környezeti változót, továbbá a felhasználó ki is tudja másolni a környezeti változó aktuális értékét.

Környezeti változó hozzáadása

Környezeti változó hozzáadásához az alábbi adatlap tartozik (Lásd 17. ábra):

17. ábra: Környezeti változó hozzáadás, adatlap (saját szerkesztés)

Jelmagyarázat a 17. ábrához:

- A 17. ábrán is megtalálható elemek megegyeznek jelentésükben a 15. ábrához tartozó jelmagyarázatban szereplőkkel
- New variable's key: Új környezeti változó neve
- New variable's value: Új környezeti változó értéke

A hozzáadás elvégzéséhez szükséges lesz megadni a következő paramétereket (Lásd 17. ábra):

- Azure projekt neve

- Variable group neve
 - o Reguláris kifejezés (Regex) alkalmazható
- Új környezeti változó neve
- Új környezeti változó értéke

Az adatlap kitöltése és a kérés elküldése után a felhasználó visszakapja a megerősítő felületet, ezzel erősítve a szoftver IT-biztonsági követelményeit (Lásd 18. ábra):

Select Azure project
bkv

*
NewKafkaBroker

kafka-broker-value

SEND REQUEST

Add variable to these variable groups (Found variable groups: 8).

Project	Variable group name	
bkv	TeamsAdapter PROD	X
bkv	TeamsAdapter PREPROD	X
bkv	TeamsAdapter DEV	X
bkv	TeamsAdapter STAGING	X
bkv	ZoomAdapter PROD	X

1-5 of 8 < >

Are you sure you want to add variables?

YES NO

18. ábra: Környezeti változó hozzáadás, megerősítő felület (saját szerkesztés)

Jelmagyarázat, 18. ábra:

- A 18. ábrán is megtalálható elemek megegyeznek jelentésükben a 17. ábrához tartozó jelmagyarázatban szereplőkkel
- Add variable to these variable groups (Found variable groups: x): A beadott paraméterek alapján visszatérő eredmény számossága
- Project: Azure projekt neve
- Variable group name: Variable group neve
- Are you sure you want to add variables? yes/no: A művelet végrehajtását megerősítő kérdés igen/nem válaszadási lehetőséggel

Látható a megerősítő felületen, hogy a hozzáadás elvégzése előtt még testreszabhatja a felhasználó a műveletet, tehát például ki is veheti azokat a variable groupokat (adott sor végén látható törlés gomb) a javaslati listából, amelyeken nem szeretné végrehajtani a műveletet.

Emellett meg is szakíthatja a műveletet a felhasználó az alsó igen/nem kérdés megválaszolásával, amennyiben ez a felhasználói szándék.

Megerősítés és sikeres végrehajtás esetén az alábbi sikerképernyő jelenik meg (Lásd 19. ábra):

Select Azure project
bkv

Success

Name (regex) of variable group

Key of variable

Value (regex) of variable [OPTIONAL]

☐ Secret included

☐ Key's name should be regex

SEND REQUEST

Matched variables (Found variables: 8)

Project	Variable Group Name	Variable key	Variable value	Operation
bkv	TeamsAdapter PROD	NewKafkaBroker	kafka-broker-value	[Icons]
bkv	TeamsAdapter PREPROD	NewKafkaBroker	kafka-broker-value	[Icons]
bkv	TeamsAdapter DEV	NewKafkaBroker	kafka-broker-value	[Icons]
bkv	TeamsAdapter STAGING	NewKafkaBroker	kafka-broker-value	[Icons]
bkv	ZoomAdapter PROD	NewKafkaBroker	kafka-broker-value	[Icons]

1 - 5 of 8

19. ábra: Környezeti változó hozzáadás, sikeres művelet képernyő (saját szerkesztés)

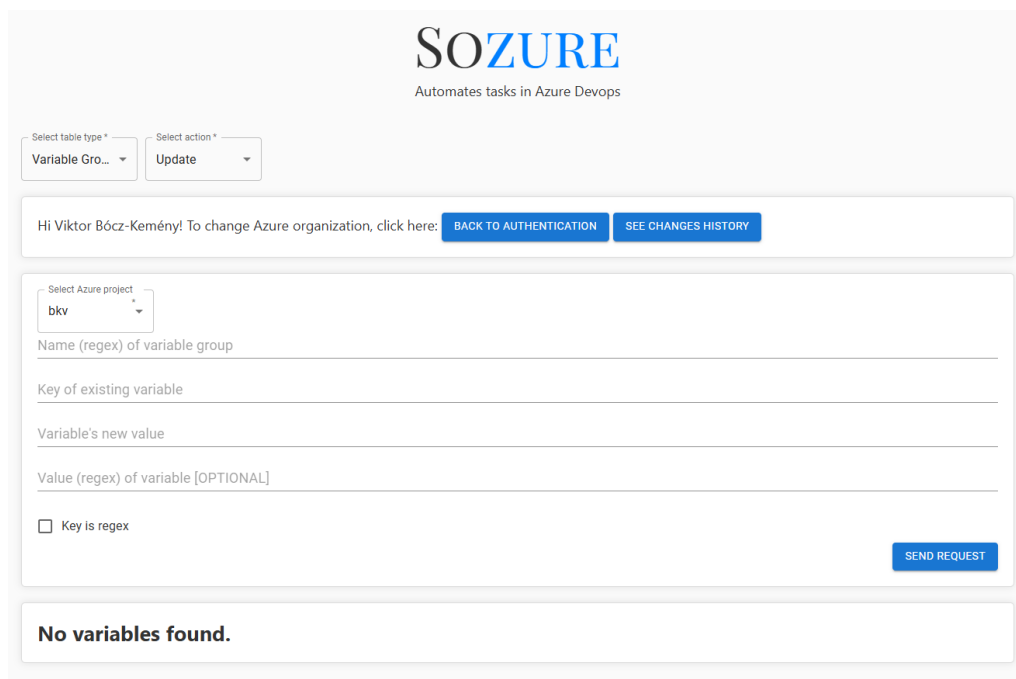
Jelmagyarázat, 19. ábra:

- A 19. ábrán is megtalálható elemek megegyeznek jelentésükben a 16. ábrához tartozó jelmagyarázatban szereplőkkel
- Success: Siker pop-up doboz a művelet sikeres végrehajtása esetén
 - o Érdeemes kiemelni, hogy a műveletek sikeressége/sikertelensége esetén a 19.ábrán látható pozícióban jelennek meg a visszaigazoló üzenetek, amelyek a következők lehetnek
 - Sikeresség esetén
 - Success: sikeres végrehajtás
 - Sikertelenség esetén:
 - Already contains: Létrehozni kívánt entitás (környezeti változó) már létezik a dedikált környezetben
 - Unauthorized: Hibás bejelentkezési adatok
 - Resource not found: Ismeretlen Azure erőforrás (a szakdolgozat esetében Azure organizáció) lett megadva

- Project does not exist: Azure projekt nem létezik a megadott Azure organizációban
- No subscriptions found: Azure Portal-hoz tartozó hitelesítési adatok nem stimmelnek
- Failed request. Consider a more specified query. Size of response message could be too large: A lekérdezési paraméterek alapján visszakapott válasz meghaladja a keretrendszer beállított/maximális átviteli kapacitását
- Branches do not exist in the git repository: Git branch nem létezik a megadott Git repository-hoz

Környezeti változó módosítása

Környezeti változó módosításához az alábbi adatlap tartozik (Lásd 20. ábra):



20. ábra: Környezeti változó módosítása, adatlap (saját szerkesztés)

Jelmagyarázat, 20. ábra:

- A 20. ábrán is megtalálható elemek megegyeznek jelentésükben a 15. ábrához tartozó jelmagyarázatban szereplőkkel
- Key of existing variable: Környezeti változó neve
- Variable's new value: Környezeti változó új értéke

A módosítás elvégzéséhez szükséges lesz megadni a következő paramétereket (Lásd 19. ábra):

- Azure projekt neve
- Variable group neve
 - o Reguláris kifejezés (RegEx) alkalmazható
- Környezeti változó neve
 - o Reguláris kifejezés (RegEx) alkalmazható, amennyiben az ábrán látható adatlap checkbox elemét bepipálja a felhasználó
- Környezeti változó új értéke
- Környezeti változó jelenlegi értéke
 - o Reguláris kifejezés (RegEx) alkalmazható
 - o Opcionális

A kérés elküldése esetén a hozzáadást megerősítő felülettel megegyező felület fog megjelenni (Lásd 18. ábra).

A művelet sikeressége esetén a hozzáadási művelet sikerképernyőjével megegyező felület fog megjelenni (Lásd 19. ábra).

Környezeti változó törlése

Környezeti változó törléséhez az alábbi adatlap tartozik (Lásd 21. ábra):

21. ábra: Környezeti változó törlése, adatlap (saját szerkesztés)

Jelmagyarázat 21. ábra:

- A 21. ábrán is megtalálható elemek megegyeznek jelentésükben a 20. ábrához tartozó jelmagyarázatban szereplőkkel

A módosítás elvégzéséhez szükséges lesz megadni a következő paramétereket (Lásd 19. ábra):

- Variable group neve
 - o Reguláris kifejezés (RegEx) alkalmazható
- Környezeti változó neve

A kérés elküldése esetén a hozzáadást megerősítő felülettel megegyező felület fog megjelenni (Lásd 18. ábra).

A művelet sikeressége esetén a hozzáadási művelet sikerképernyőjével megegyező felület fog megjelenni (Lásd 19. ábra).

3.2.1.3 *Secretek lekérdezése, manipulálása*

A szükséges hitelesítő adatok megadása után a következő felület fogadja a felhasználót (Lásd 22. ábra).

The screenshot shows the SOZURE web application interface. At the top, the logo 'SOZURE' is displayed with the tagline 'Automates tasks in Azure DevOps'. Below the logo, there are two dropdown menus: 'Select table type *' with 'Secrets' selected, and 'Select action *' with 'List' selected. A message bar indicates: 'Hi Viktor Bócz-Kemény! To change Azure organization, click here: BACK TO AUTHENTICATION SEE CHANGES HISTORY'. Below this, there is a section for 'Select KeyVault *' with 'KeyVault-BKV' selected. A text input field for 'Secret name (regex)' is present, followed by a checkbox labeled 'Get deleted secrets'. A 'SEND REQUEST' button is located at the bottom right of the form. At the bottom of the page, the text 'No secrets found.' is displayed.

22. ábra: Secret lekérdezés, adatlap (saját szerkesztés)

Jelmagyarázat, 22. ábra:

- A 22. ábrán is megtalálható elemek megegyeznek jelentésükben a 21. ábrához tartozó jelmagyarázatban szereplőkkel
- Secret key vault: Azure key vault neve
- Secret name (regex): Secret neve
- Get deleted secrets: törölt secretet is vissza szeretné-e kapni a felhasználó, igen/nem

Secret lekérdezése

A lekérdezés elvégzéséhez szükséges lesz megadni a következő paramétereket (Lásd 22. ábra):

- Azure key vault neve
- Secret neve
 - o Reguláris kifejezés (RegEx) alkalmazható
- Törölt secretokat is vissza szeretné-e kapni a felhasználó, checkbox

Példa lekérdezést elvégeztem annak érdekében, hogy a megjelenő eredményfelületet is bemutathassam (Lásd 23. ábra):

The screenshot shows the Azure Key Vault search interface. At the top, there are two dropdown menus: 'Select table type *' with 'Secrets' selected, and 'Select action *' with 'List' selected. Below these is a message: 'Hi Viktor Bócz-Kemény! To change Azure organization, click here:' followed by two buttons: 'BACK TO AUTHENTICATION' and 'SEE CHANGES HISTORY'. Underneath is a 'Select KeyVault *' dropdown with 'KeyVault-BKV' selected. A search input field contains the pattern '.*'. There is a checkbox labeled 'Get deleted secrets' which is currently unchecked. A 'SEND REQUEST' button is on the right. Below the search area, it says 'Matched secrets (Found secrets: 3)'. A table displays the results with columns: 'Key vault', 'Secret name', 'Secret value', and 'Operation'. The table has three rows of data. Each row has two buttons in the 'Operation' column: a document icon and a trash can icon. At the bottom left, it says 'Page: 1/1'.

Key vault	Secret name	Secret value	Operation
KeyVault-BKV	Secret1	Secret1	[Document Icon] [Trash Icon]
KeyVault-BKV	Secret2	Secret2	[Document Icon] [Trash Icon]
KeyVault-BKV	Secret3	'\$AA...'	[Document Icon] [Trash Icon]

23. ábra: Secret lekérdezés eredményfelülete (saját szerkesztés)

Jelmagyarázat, 23. ábra:

- A 23. ábrán is megtalálható elemek megegyeznek jelentésükben a 22. ábrához tartozó jelmagyarázatban szereplőkkel
- Key vault: Key vault neve
- Secret name: Secret neve
- Secret value: Secret érték
- Operation: További tevékenységeket (érték kimásolása vágólapra, törlés) indítani képes gombok

A megjelenő eredményfelület mutatja, hogy összesen hány találatot kap a felhasználó a lekérdezése alapján, illetve a táblázatban megjeleníti az adott secret alábbi adatait:

- Key vault neve
- Secret neve
- Secret értéke

Lehetőség van az adott sorhoz tartozó gombok segítségével törölni a sorhoz tartozó secretet, továbbá ki tudja másolni a secret aktuális értékét.

Secret törlése

Secret törléséhez az alábbi adatlap tartozik (Lásd 24. ábra):

24. ábra: Secret törlése, adatlap (saját szerkesztés)

Jelmagyarázat, 24. ábra:

- A 24. ábrán is megtalálható elemek megegyeznek jelentésükben a 22. ábrához tartozó jelmagyarázatban szereplőkkel

A secret törléséhez szükséges lesz megadni a következő paramétereket (Lásd 24. ábra):

- Azure key vault neve
- Secret neve

Az adatlap kitöltése és a kérés elküldése után a felhasználó visszakapja a megerősítő felületet, ezzel erősítve a szoftver IT-biztonsági követelményeit (Lásd 25. ábra):

SOZURE
Automates tasks in Azure Devops

Select table type *
Secrets

Select action *
Delete

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select KeyVault *
KeyVault-BKV

Secret1

SEND REQUEST

Matched secrets (Found secrets: 1).

Key vault	Secret name	Secret value
KeyVault-BKV	Secret1	Secret1

Page: 1/1

Are you sure you want to delete secret?

YES NO

25. ábra: Secret törlés megerősítő felület (saját szerkesztés)

Jelmagyarázat, 25. ábra:

- A 25. ábrán is megtalálható elemek megegyeznek jelentésükben a 24. ábrához tartozó jelmagyarázatban szereplőkkel
- Key vault: Key vault neve
- Secret name: Secret neve
- Secret value: Secret értéke
- Are you sure you want to delete secret? yes/no: A művelet végrehajtását megerősítő kérdés igen/nem válaszadási lehetőséggel

A felhasználó megszakíthatja a műveletet az alsó igen/nem kérdés megválaszolásával, amennyiben ez a felhasználói szándék.

Megerősítés és sikeres végrehajtás esetén visszatér a felület a kilistázó adatlaphoz (Lásd 22. ábra).

Secret reaktíválása

Secret reaktíválásához a secret törlésével megegyező adatlapot kell kitölteni (Lásd 25. ábra).

A kérés elküldése esetén a secret törléséhez tartozó megerősítő felülettel megegyező felület fog megjelenni (Lásd 25. ábra).

A művelet sikeressége esetén, a secret törléséhez hasonlóan visszatér a felület a kilistázó adatlaphoz (Lásd 22. ábra).

Key vault másolás

Key vault másolásához az alábbi adatlap tartozik (Lásd 26. ábra):

26. ábra: Key vault másolása, adatlap (saját szerkesztés)

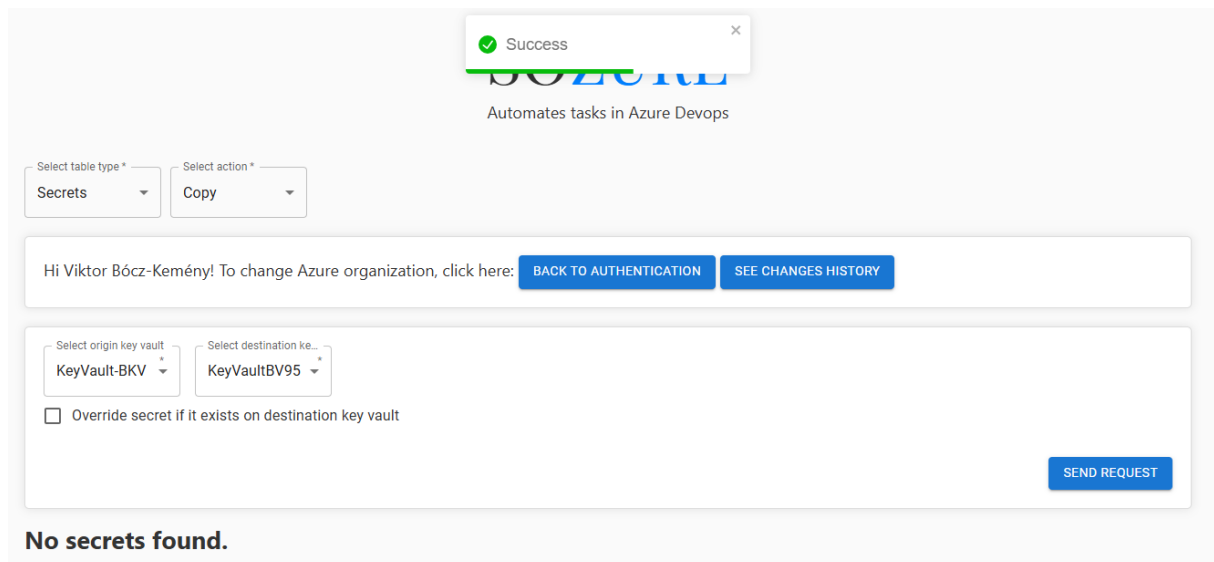
Jelmagyarázat, 26. ábra:

- A 26. ábrán is megtalálható elemek megegyeznek jelentésükben a 25. ábrához tartozó jelmagyarázatban szereplőkkel
- Origin key vault: Adatforrás key vault
- Destination key vault: Cél key vault
- Override secret if it exists on destination key vault: Felülírja-e az eszköz a cél key vault-ban található secret értékét, amennyiben meglévőt másolna át a felhasználó az említett key vault-ba (checkbox)

A key vault másolásához szükséges lesz megadni a következő paramétereket (Lásd 26. ábra):

- Adatforrás key vault
- Cél key vault
- Felülírja-e az eszköz a cél key vault-ban található secret értékét, amennyiben meglévőt másolna át a felhasználó az említett key vault-ba (checkbox)

A művelet jóváhagyása esetén a következő eredményképernyő fogadja a felhasználót (Lásd 27. ábra):



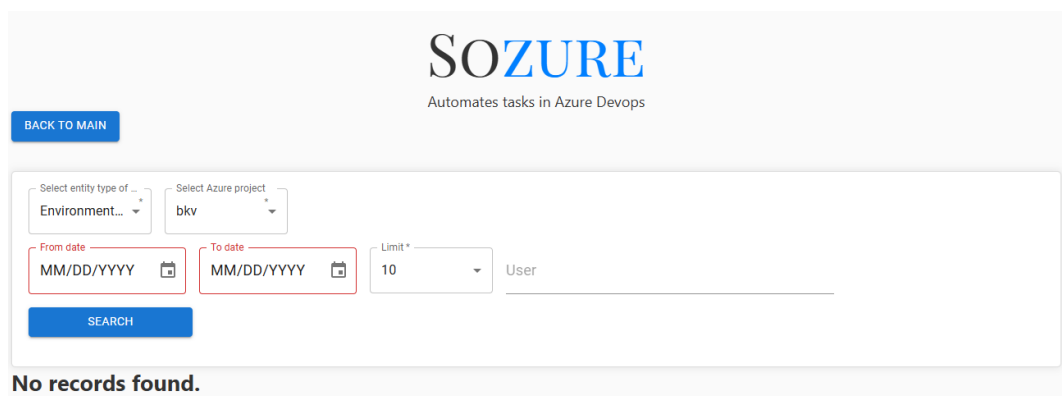
27. ábra: Key vault átmásolás sikerképernyő (saját szerkesztés)

Jelmagyarázat, 27. ábra:

- A 27. ábrán is megtalálható elemek megegyeznek jelentésükben a 26. ábrához tartozó jelmagyarázatban szereplőkkel
- Success: Sikeres végrehajtás

3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése

Ahogy a felület is mutatja a 15. ábrán, a felhasználó (a bejelentkezést követően) megtekintheti a korábbi módosításokat a See changes history gomb megnyomásával. A gomb megnyomásával a következő felület jelenik meg (Lásd 28. ábra):



28. ábra: Szoftverben elvégzett módosítások megtekintése (saját szerkesztés)

Jelmagyarázat, 28. ábra:

- Back to main gomb: Visszatérés a 15. ábrán is látható felületre

- Modification type: Választható módosítás típus
- Azure project: Választható Azure projekt
- From date: Lekérdezés kezdődátuma
- To date: Lekérdezés végdátuma
- Limit: Találatbéli limitáció
- User: Felhasználószintű lekérdezés
- Search gomb: művelet végrehajtása

Környezeti változó módosítások lekérdezése

Az adatlap (Lásd 28. ábra) kitöltésével és a művelet végrehajtásával megjelenik a következő találati felület (Lásd 29. ábra):

Records (found 9 elements):

Organization	Project	User	Variable group regex	Key	Operation type	Date
bevikt95	bkv	Viktor Bócz-Kemény	.*	NewKafkaBroker	Add	Tue, 11 Mar 2025 17:41:45 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PREPROD	AdminUser	Update	Tue, 11 Mar 2025 15:25:44 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Delete	Tue, 11 Mar 2025 15:25:35 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Update	Tue, 11 Mar 2025 15:25:30 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Update	Tue, 11 Mar 2025 15:25:24 GMT

Page: 1/2

29. ábra: Korábbi környezeti változó módosítások lekérdezése (saját szerkesztés)

Jelmagyarázat, 29. ábra:

- A 29. ábrán is megtalálható elemek megegyeznek jelentésükben a 28. ábrához tartozó jelmagyarázatban szereplőkkel
- Records (found x elements): A beadott paraméterek alapján visszatérő eredmény számossága
- Organization: Azure organizáció
- Project: Azure projekt
- User: Módosítást elvégző felhasználó
- Variable group regex: Megadott variable group neve (akár reguláris kifejezés formájában is)
- Key: Környezeti változó neve

- Operation type: Művelet típusa
- Date: Módosítás dátuma

Secret módosítások lekérdezése

Az adatlap (Lásd 28.ábra) kitöltésével és a művelet végrehajtásával megjelenik a következő találati felület (Lásd 30. ábra):

The screenshot shows the SOZURE application interface. At the top, there's a logo 'SOZURE' and the tagline 'Automates tasks in Azure DevOps'. Below it, a 'BACK TO MAIN' button is visible. The main search area contains several filters: 'Select modification type' (set to 'Secrets'), 'Select Azure vault' (set to 'KeyVault-BKV'), 'From date' (03/01/2025), 'To date' (03/14/2025), 'Limit' (10), and a 'User' field. A 'SEARCH' button is at the bottom of the filter section. Below the filters, it says 'Records (found 4 elements):'. A table follows with 5 columns: 'Key vault's name', 'User', 'Secret name regex', 'Operation type', and 'Date'. The table contains 4 rows of data. At the bottom left, it says 'Page: 1/1'.

Key vault's name	User	Secret name regex	Operation type	Date
KeyVault-BKV	Viktor Bócz-Kemény	Secret1	Delete	Tue, 11 Mar 2025 20:27:27 GMT
KeyVault-BKV	Viktor Bócz-Kemény	Secret1	Recover	Tue, 11 Mar 2025 20:26:36 GMT
KeyVault-BKV	Viktor Bócz-Kemény	Secret1	Delete	Tue, 11 Mar 2025 20:26:15 GMT
KeyVault-BKV	Viktor Bócz-Kemény	Secret1	Delete	Tue, 11 Mar 2025 13:43:16 GMT

30. ábra: Korábbi secret módosítások lekérdezése (saját szerkesztés)

Jelmagyarázat, 30. ábra:

- A 30. ábrán is megtalálható elemek megegyeznek jelentésükben a 28. ábrához tartozó jelmagyarázatban szereplőkkel
- Records (found x elements): A beadott paraméterek alapján visszatérő eredmény számossága
- Key vault's name: Key vault neve
- User: Módosítást elvégző felhasználó
- Secret name regex: Megadott secret neve (akár reguláris kifejezés formájában is)
- Operation type: Művelet típusa
- Date: Módosítás dátuma

Key vault másolások lekérdezése

Az adatlap (Lásd 28.ábra) kitöltésével és a művelet végrehajtásával megjelenik a következő találati felület (Lásd 31. ábra):

SOZURE

Automates tasks in Azure Devops

[BACK TO MAIN](#)

Select modification type
 Key vault co...

From date
 03/01/2025

To date
 03/14/2025

Limit *
 10

User

[SEARCH](#)

Records (found 3 elements):

Original key vault	Destination key vault	User	Date
KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	Tue, 11 Mar 2025 20:47:36 GMT
KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	Tue, 11 Mar 2025 20:47:31 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Tue, 11 Mar 2025 13:43:52 GMT

Page: 1/1

31. ábra: Korábbi key vault másolások lekérdezése (saját szerkesztés)

Jelmagyarázat, 31. ábra:

- A 31. ábrán is megtalálható elemek megegyeznek jelentésükben a 28. ábrához tartozó jelmagyarázatban szereplőkkel
- Records (found x elements): A beadott paraméterek alapján visszatérő eredmény számossága
- Original key vault: Adatforrás key vault
- Destination key vault: Cél key vault
- User: Módosítást elvégző felhasználó
- Date: Módosítás dátuma

3.2.1.5 Pipeline futtatás

Ahogy a felület is mutatja a 15. ábrán, a felhasználó (a bejelentkezést követően) megtekintheti build pipeline futtató adatlapot az első lenyíló mezőben kiválasztva a „Run pipelines” (Pipeline futtatás) opciót (Lásd 32. ábra):

The screenshot shows the SOZURE web application interface. At the top, the logo 'SOZURE' is displayed in blue, with the tagline 'Automates tasks in Azure Devops' below it. The main content area has a light gray background. At the top of this area, there are two dropdown menus: 'Select table type *' with 'Run pipelines' selected, and 'Select action *' with 'Build' selected. Below these is a white box containing a greeting 'Hi Viktor Bócz-Kemény! To change Azure organization, click here:' followed by two blue buttons: 'BACK TO AUTHENTICATION' and 'SEE CHANGES HISTORY'. Below this is another white box with a 'Select Azure project' dropdown menu showing 'bkv' and a blue button 'GET BUILD PIPELINES'. At the bottom, there is a search bar with the text 'Search' and a message 'There are no build pipelines.'

32. ábra: Build pipeline futtatásának első adatlapja, pipeline lekérések (saját szerkesztés)

Jelmagyarázat, 32. ábra:

- A 32. ábrán is megtalálható elemek megegyeznek jelentésükben a 15. ábrához tartozó jelmagyarázatban szereplőkkel
- Azure project: Azure projekt neve
- Get build pipelines gomb: Művelet végrehajtására szolgáló gomb

A végrehajtó gomb megnyomásával megjelenik a build pipeline lista az eredménytáblában (Lásd 33. ábra):

The screenshot shows the SOZURE web application interface. At the top, the logo 'SOZURE' is displayed with the tagline 'Automates tasks in Azure Devops'. Below the logo, there are two dropdown menus: 'Select table type *' with 'Run pipelines' selected, and 'Select action *' with 'Build' selected. A message bar indicates 'Hi Viktor Bócz-Kemény! To change Azure organization, click here:' with two buttons: 'BACK TO AUTHENTICATION' and 'SEE CHANGES HISTORY'. Below this, there is a 'Select Azure project' dropdown menu with 'bkv' selected, and a 'GET BUILD PIPELINES' button. The main content area shows a search bar and a section titled 'Found pipelines: 1'. Below this, there is a table with the following structure:

Pipeline	Source type	Source	Action
Company.Team.TestApp	Set source type * tag	Set source * refs/tags/0...	RUN PIPELINE

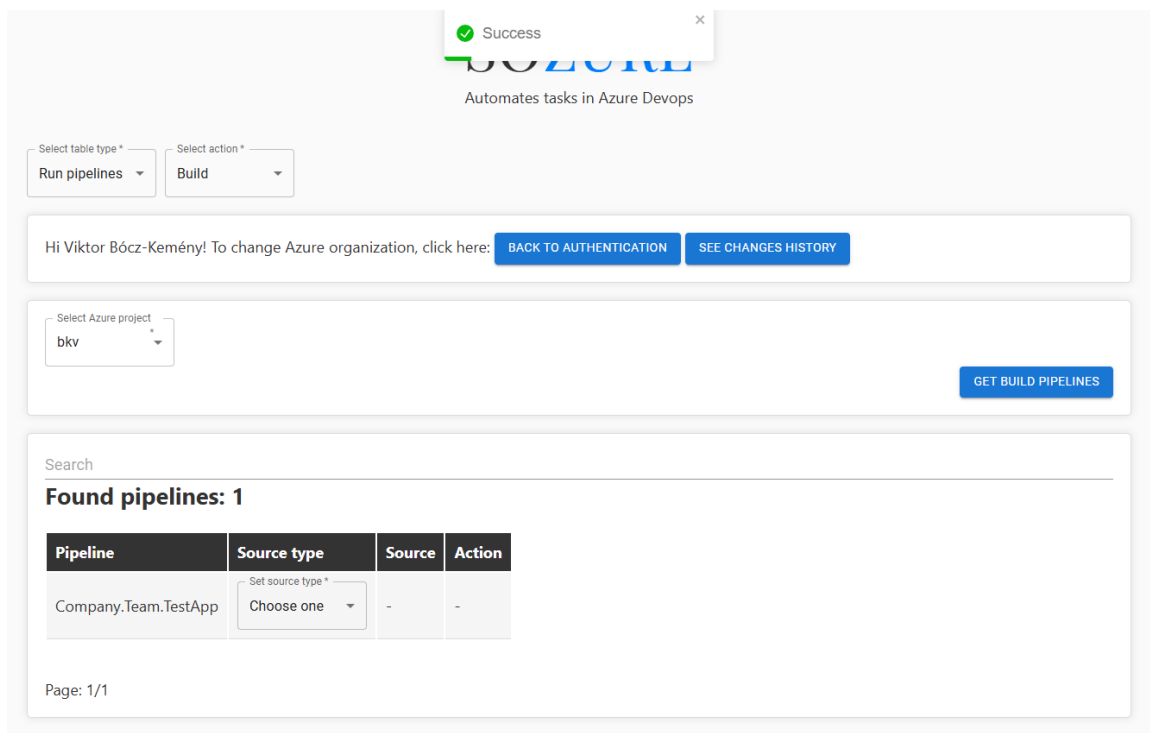
At the bottom of the table, it says 'Page: 1/1'.

33. ábra: Build pipeline lekérdezés eredménytáblája, pipeline futtatás adatlapja (saját szerkesztés)

Jelmagyarázat, 33. ábra:

- A 33. ábrán is megtalálható elemek megegyeznek jelentésükben a 32. ábrához tartozó jelmagyarázatban szereplőkkel
- Pipeline: Build pipeline neve
- Source type: Forrástípus, lehet branch vagy tag
- Source: Forrás, lehet egy branch, vagy egy tag is
- Action oszlopban megtalálható run pipeline gomb: végrehajtást elindító gomb

A run pipeline gomb megnyomásával visszatér a felület az eredménytáblában lévő alapértelmezett adatlapra (Lásd 34. ábra):

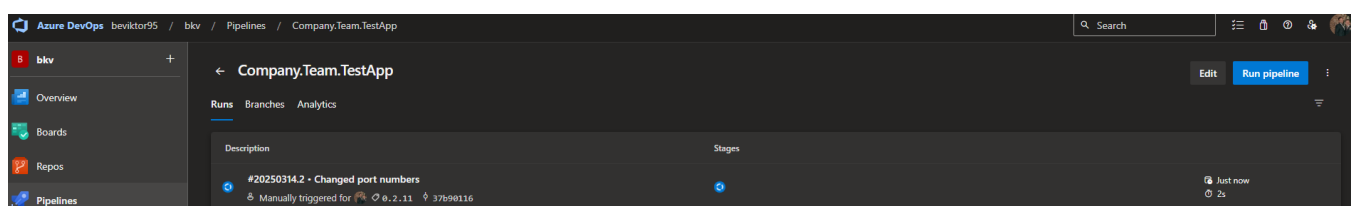


34. ábra: Sikeres végrehajtás, alapértelmezett build pipeline futtató eredménytábla adatlap (saját szerkesztés)

Jelmagyarázat, 34. ábra:

- A 33. ábrán is megtalálható elemek megegyeznek jelentésükben a 32. ábrához tartozó jelmagyarázatban szereplőkkel
- Siker pop-up doboz

Mindeközben látható, hogy az Azure DevOps felületén elindította a pipeline-t (Lásd 35. ábra):



35. ábra: Azure DevOps felületén elindított pipeline (saját szerkesztés)

3.2.1.6 Utolsó szoftververziók lekérdezése

Ahogy a felület is mutatja a 15. ábrán, a felhasználó (a bejelentkezést követően) megtekintheti az utolsó stabil szoftververziót az első lenyíló mezőben kiválasztva a „Tags” opciót (Lásd 36. ábra):

SOZURE
Automates tasks in Azure Devops

Select table type *
Tags

Select action *
List

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select Azure project
bkv

[GET REPOSITORIES](#)

Search
There are no repositories.

36. ábra: Utolsó stabil szoftververzió lekérdezése (saját szerkesztés)

Jelmagyarázat, 36. ábra:

- A 36. ábrán is megtalálható elemek megegyeznek jelentésükben a 15. ábrához tartozó jelmagyarázatban szereplőkkel
- Azure project: Azure projekt
- Get repositories gomb: Művelet végrehajtása

Az adatlaphoz (Lásd 36. ábra) tartozó műveleti gombbal elvégezzük a lekérdezést, a következő eredménytábla jelenik meg (Lásd 37.ábra):

SOZURE
Automates tasks in Azure Devops

Select table type *
Tags

Select action *
List

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select Azure project
bkv

[GET REPOSITORIES](#)

Search
Found repositories: 2

Repository	Latest tag
bkv	-
Company.Team.TestApp	0.2.11

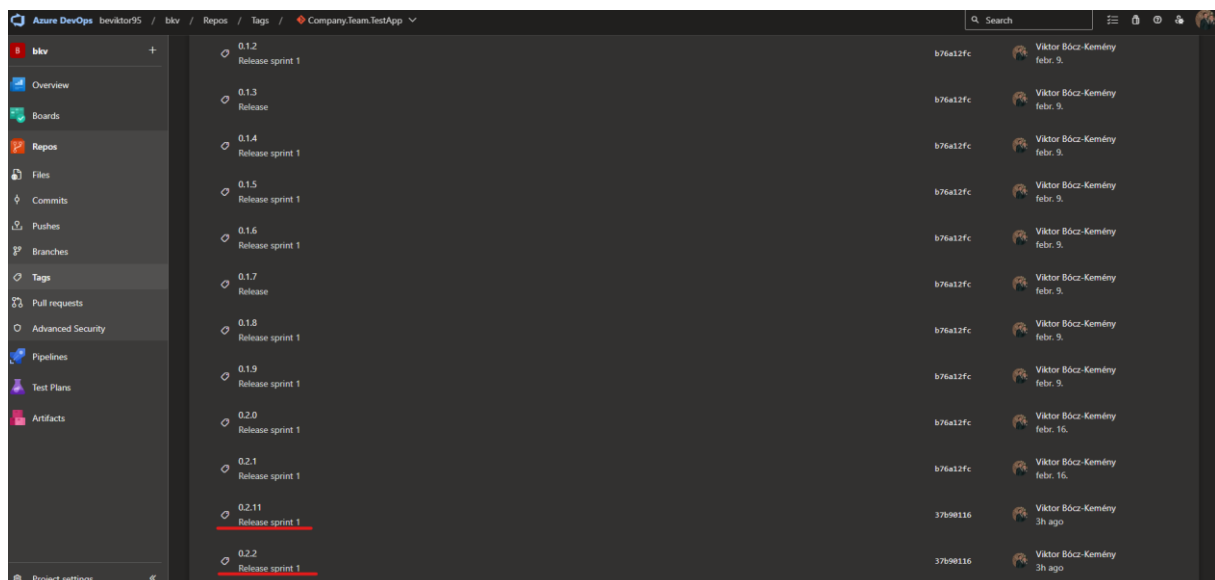
Page: 1/1

37. ábra: Utolsó stabil szoftververzió lekérdezésének eredménye (saját szerkesztés)

Jelmagyarázat, 37. ábra:

- A 37. ábrán is megtalálható elemek megegyeznek jelentésükben a 36. ábrához tartozó jelmagyarázatban szereplőkkel
- Search mező: dinamikusan kereshető a beadott input alapján a repository listában, ami filterezi a találatot az alsó traktusban megjelenő táblázatban
- Found repositories: x: A beadott paraméterek alapján visszatérő eredmény számossága
- Repository: Forráskód repository
- Latest tag: Utolsó stabil szoftververzió

Ezen a ponton érdemes megemlíteni, hogy a szoftver működése kijavítja az Azure DevOps felületén megjelenített stabil verziók sorrendiségét. Az Azure DevOps felületén az alábbi módon vannak felsorolva a stabil verziószámok (Lásd 38. ábra):



38. ábra: Azure DevOps felület - stabil verziószámok hibás rendezése (saját szerkesztés)

A képen látható, hogy a verziószámok növekvő sorrendben történő rendezése az intenciója a felületnek, azonban ez hibásan működik: a 0.2.11 verziószámnak kellene az utolsó helyen állnia, közvetlenül a 0.2.2 verziószám után. Bár a megírt szoftver csak az utolsó stabil verziót jeleníti meg (harmóniában a megrendelői igényekkel, Lásd [1.6.4.1 Megrendelői elvárások](#)), belátható, hogy az elkészített szoftver (Lásd 37. ábra) javítja ezt a hiányosságot, hiszen a helyes utolsó verziót jeleníti meg.

3.2.1.7 Pull request létrehozás, lekérdezés

Létrehozás

Ahogy a felület is mutatja a 15. ábrán, a felhasználó (a bejelentkezést követően) megtekintheti a pull request-et létrehozó adatlapot az első lenyíló mezőben kiválasztva a „Pull request” opciót, majd a „Create” opciót kiválasztva a második lenyíló mezőben (Lásd 38. ábra):

39. ábra: Repository lekérdezés adatlapja az új pull request létrehozásához (saját szerkesztés)

Jelmagyarázat, 39. ábra:

- A 39. ábrán is megtalálható elemek megegyeznek jelentésükben a 36. ábrához tartozó jelmagyarázatban szereplőkkel

Az adatlaphoz (Lásd 39. ábra) tartozó műveleti gombbal elvégezzük a lekérdezést, a következő eredménytábla jelenik meg (Lásd 40. ábra):

SOZURE

Automates tasks in Azure DevOps

Select table type *

Pull requests

Select action *

Create

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select Azure project *

bkv

GET REPOSITORIES

Search

Found repositories: 2

Azure project	Repository	Source branch	Target branch	Title	Autocomplete	Send
bkv	bkv	CREATE PR	-	-	-	-
bkv	Company.Team.TestApp	Select source br... develop	Select target br... main	Thesis test: develop into i	☐	SEND

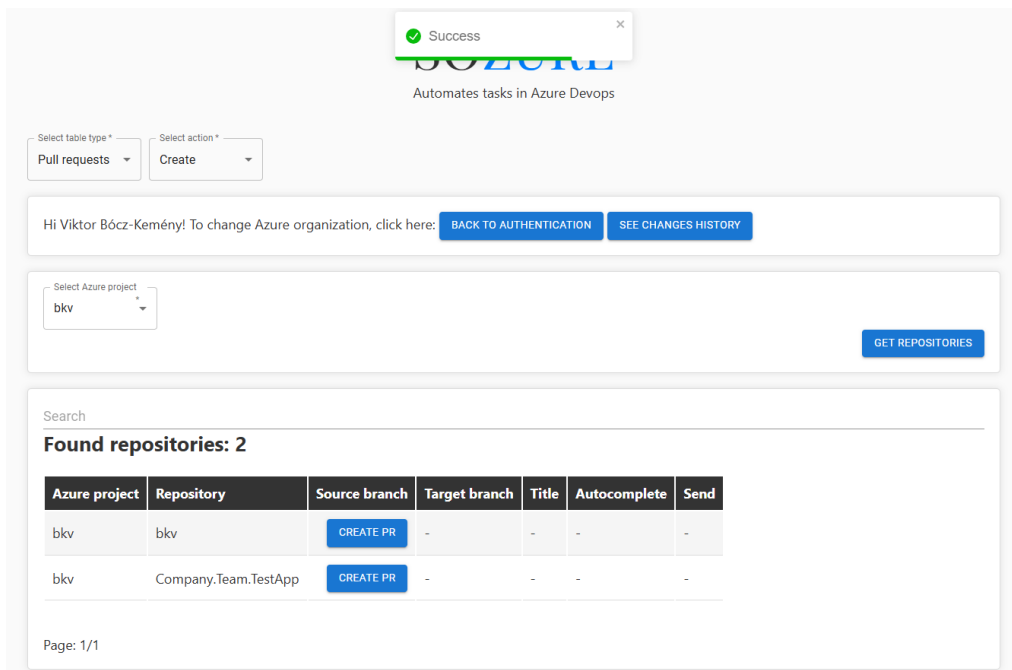
Page: 1/1

40. ábra: Repository lekérdezésének eredménye a pull request létrehozásához és a létrehozás adatlapja (saját szerkesztés)

Jelmagyarázat, 40. ábra:

- 40. ábrán is megtalálható elemek megegyeznek jelentésükben a 39. ábrához tartozó jelmagyarázatban szereplőkkel
- Azure project: Azure projekt neve
- Repository: Git repository neve
- Source branch:
 - o Create PR gomb, amivel elindítható az adatlap kitöltése
 - o Source branch lenyíló mező: Forrás branch, amiből indítva a pull request-et szeretné létrehozni a felhasználó
- Target branch: Cél branch, amit módosítani szeretne a felhasználó a pull request létrehozásával
- Title: Pull request cím
- Autocomplete: Azure DevOps által szolgáltatott pull request autocomplete opció bekapcsolása
- Send: Műveletet végrehajtó gomb

A művelet végrehajtás megnyomása esetén a következő felület fogadja a felhasználót (Lásd 41. ábra):



41. ábra: Pull request létrehozása utáni sikerképernyő (saját szerkesztés)

Jelmagyarázat, 41. ábra:

- 41. ábrán is megtalálható elemek megegyeznek jelentésükben a 40. ábrához tartozó jelmagyarázatban szereplőkkel
- Success: a 41. ábrán is megtalálható elemek megegyeznek jelentésükben a 19. ábrához tartozó jelmagyarázatban szereplőkkel

Létrejött a pull request, amit a lekérdezés szekcióban le is fogok ellenőrizni.

Lekérdezés

Ahogy a felület is mutatja a 15. ábrán, a felhasználó (a bejelentkezést követően) megtekintheti a pull request-et lekérdező adatlapot az első lenyíló mezőben kiválasztva a „Pull request” opciót, majd a „List” opciót kiválasztva a második lenyíló mezőben (Lásd 34. ábra, megegyezik a pull request létrehozására irányuló repository lekérdező adatlappal).

A művelet végrehajtására szolgáló gomb megnyomásával megjelenik az eredményképernyő (Lásd 42. ábra):

Select table type *
Pull requests

Select action *
List

Hi Viktor Bócz-Kemény! To change Azure organization, click here:
[BACK TO AUTHENTICATION](#)
[SEE CHANGES HISTORY](#)

Select Azure project
bkv

GET PULL REQUESTS

Search

Found pull requests: 1

Project	Repository	Title	Created by	Created	Size	Approvers	Navigate
bkv	Company.Team.Te...	Thesis test: develo...	Viktor Bócz-Kemény	Today	Medium		

1-1 of 1 < >

42. ábra: Pull request lekérdező képernyő eredménytáblája (saját szerkesztés)

Jelmagyarázat, 42. ábra:

- 42. ábrán is megtalálható elemek megegyeznek jelentésükben a 40. ábrához tartozó jelmagyarázatban szereplőkkel
- Found pull requests: x: A beadott paraméterek alapján visszatérő eredmény számossága
- Project: Azure projekt neve
- Repository: Git repository neve
- Title: Pull request címe
- Created by: Pull request létrehozója
- Size: Pull request mérete módosult fájlok szerint
- Approvers: Jóváhagyók
- Navigate gomb: a megnyomásával megnyitható a böngészőben külön lapon

Látható, hogy a létrehozott pull requestet a legkérdezés során már vissza is kapom eredményként.

3.2.1.8 IT-Biztonság

A webfelület elkészítésekor figyelembe vettem IT-biztonsági aspektusokat, amelyeket ebben az alfejezetben fogok kifejteni.

A bejelentkezés során (Lásd 13. ábra és 14. ábra) elvárja a felület a felhasználótól a bejelentkezési adatok megadásának szükségességét, illetve minden felhasználónak a saját bejelentkezési adataival van csak lehetősége a belépésre. Nincs admin, vagy szuper felhasználó, akiknek a belépési adatait beégettem volna a felületre (akár az input mezőkbe) ezzel kényszerítve a felhasználót a bejelentkezésre.

Továbbá a beviteli mezőknél több helyen is restriktíóval éltem, például az Azure projektek és Azure key vault-ok nem szabadszöveges írható mezők, hanem lenyíló (ügynevezett drop-down) elemek (Lásd 15. ábra és 29. ábra). Ennek megvan a maga kényelmi aspektusa is (tudja a felhasználó, hogy miből kell választani), viszont irányítom is a felhasználót ezzel, megadva a szoftver nagyobb működésbiztonságát.

Mindemellett a művelet sikerességéről, vagy sikertelenségéről is tájékoztatja a felület a felhasználót a művelet végeredményét illetően (Lásd 19. ábra jelmagyarázat), ezzel garantálva a többletinformáció adta biztonsági garanciákat.

3.2.1.9 Mesterséges intelligencia használata

A webfelület létrehozásához használtam a ChatGPT nevű generatív mesterséges intelligenciát, erre vonatkozó prompt megtalálható a 9. mellékletben (Szoftververziók verziószám alapján történő rendezése című szekció).

A stabil verziók verziószám alapján történő újrendezésének algoritmusírását kiszerveztem az említett MI eszköznek, az eredmény megtalálható a hivatkozott szekcióban. A megírt függvény eredményességét a [3.2.1.6 Utolsó szoftververziók lekérdezése](#) fejezetben bemutattam.

3.2.2 Szerveroldal

Ebben a fejezetben bemutatom a megvalósult szoftver szerveroldali komponenseit (Lásd [3.2.2.1 Kiszolgáló microservice-k bemutatása](#)), ezen kívül kitérve az IT-biztonsági (Lásd [3.2.2.2 IT-Biztonság](#)) és a mesterséges intelligenciára vonatkozó aspektusokra is (Lásd [3.2.2.3 Mesterséges intelligencia használata](#)).

3.2.2.1 Kiszolgáló microservice-k bemutatása

A [3.1 A szoftver tervezése](#) fejezetben felvázoltam a szoftverarchitektúrát (Lásd 11.ábra), ennek az architektúrának a szerveroldali megvalósítását fogom bemutatni ebben az alfejezetben.

Ahogy a 11. ábrán is látható, a webkliens két API-hoz is csatlakozik HTTP kommunikáció megvalósításával:

- A bejelentkezés, környezeti változó, secret és key vault funkcionalitásokhoz tartozó generált API swagger dokumentációja elérhető a 10. mellékletben.
- A többi funkcionalitáshoz tartozó generált API swagger dokumentációja pedig a 11. mellékletben található.

Az említett API Swagger mellékletekben megtalálhatóak JSON formátumban azok a modellek, amiket a webkliens oldaláról vár a szerveroldal bemeneti paraméterként, illetve amiket a szerveroldal válaszol a küldött kérésre. Ennek köszönhetően a webkliens elkészítője (amennyiben a felhasználó interfészt fejlesztő személye eltérne a szerveroldal fejlesztőjétől) értesül az elvárt bemeneti és kimeneti struktúrákról is, illetve az olvasó is ellenőrizni tudja, hogy pontosan milyen kérés-válasz struktúrákkal dolgozik a szerver. Mindezt a forráskód feltárása nélkül (Lásd [1.6.4.1 Megrendelői elvárások](#)).

Az Adapter biztosítja a kapcsolatot az Azure DevOps API (az Azure SDK használatával) szolgáltatásaival, így kiszolgálva az említett két API-t.

A kapcsolat kiindulópontja az alábbi C# osztály, amely a szükséges hitelesítési adatok megadásával létrehozza az Azure kapcsolatot, majd az előre definiált kliensek kikérésével az Azure API szolgáltatások is elérhetővé válnak (Lásd 43.ábra és jelmagyarázat):

```

1 reference
public class HttpClientProvider : IHttpClientProvider
{
    private VssConnection _connection = null!;
    private ClientSecretCredential _secretCredential = null!;
    private Uri _secretUri = null!;

    25 references
    public void Setup(string organization, string pat)
    {
        var uriString = $"https://dev.azure.com/{organization}";
        Uri uri;
        Uri.TryCreate(uriString, UriKind.Absolute, out uri!);

        var credentials = new VssBasicCredential(string.Empty, pat);
        _connection = new VssConnection(uri, credentials);
    }

    7 references
    public void Setup(
        string keyVaultName,
        string tenantId,
        string clientId,
        string clientSecret
    )
    {
        _secretUri = new Uri($"https://{keyVaultName.ToLower()}.vault.azure.net/");
        _secretCredential = new ClientSecretCredential(tenantId, clientId, clientSecret);
    }

    27 references
    public async Task<T> GetClientAsync<T>(Cancellation token cancellationToken = default) where T : VssHttpClientBase
    {
        return await _connection.GetClientAsync<T>(cancellationToken);
    }

    7 references
    public SecretClient GetSecretClient()
    {
        return new(_secretUri, _secretCredential);
    }
}

```

43. ábra: Saját készítésű HttpClientProvider felelős az Azure kapcsolatért (saját szerkesztés)

Jelmagyarázat, 43. ábra:

- class HttpClientProvider: IHttpClientProvider
 - o HttpClientProvider elnevezésű osztály, amely implementálja az IHttpClientProvider interfész deklarált függvényeit
- VssConnection _connection, ClientSecretCredential _secretCredential és Uri _secretUri
 - o A kapcsolat létrehozásáért felelős külső dependencia osztályok
- Setup függvények felsorolt paraméterekkel
 - o A kapcsolat létrehozásáért felelős saját készítésű függvények, melyek az implementált logika alapján felhasználják a külső dependencia osztályokat
- GetClientAsync és GetSecretClient függvények
 - o Biztosítják az Azure kliens osztályokat az egyéb absztrakciós rétegekben elhelyezkedő szervíz típusú osztályoknak, amelyek pedig az API modulokat szolgálják ki az elvárt funkcionalitás működéséért

Mivel a megrendelő zárt forráskódú szoftvert rendelt meg (Lásd [1.6.4.1 Megrendelői elvárások](#)), emiatt a forráskód részletesebb ismertetése nem lehetséges a szakdolgozatban, viszont az előre leegyeztetett minőségbiztosítási folyamat és a megrendelő beleegyezése mentén harmadik fél is megtekintheti a forráskód bizonyos elemeit.

Mindazonáltal a fenti kódrészlet és a biztosított mellékletek és ábrák demonstrálhatták a szakirodalmi háttér adta teóriák gyakorlati megvalósítását (Lásd [2.4 Szerverek és szerveroldali technológiák](#), [2.10.4 Adatszerkezetek és algoritmusok](#), [2.10.7 Programozási alapelvek és módszertanok](#), [2.10.13 Programozás I.](#), [Programozás II.](#), [Programozás III.](#)).

A webfelület ismertetéséhez (Lásd [3.2.1 Webfelület](#)) hasonló, funkcionalitásszintű bemutatást nem tartottam indokoltnak ebben a fejezetben, ugyanis az említett fejezet bemutathatatlan lett volna a rendelkezésreálló szerveroldal nélkül. Ennek okán az ott elmondottak érvényesek (lehetnek) a szerveroldali fejezetre is.

3.2.2.2 IT-biztonság

Ahogy ezt a [3.1.3 Felhasznált technológiák összesítése](#) fejezetben is bemutattam, a szerveroldali fejlesztések .NET 8 verzióval készültek. Ahogy erről az alábbi Microsoft blog posztban is olvasni lehet: „*A .NET 8-tól kezdve minden Linux konténer image tartalmazni fog egy nem root felhasználót. A .NET konténereket így egyetlen sor kóddal nem root felhasználóként is hosztolhatja. Ez a platformszintű változás biztonságosabbá teszi az alkalmazásokat, a .NET pedig az egyik legbiztonságosabb fejlesztői ökoszisztémává válik. Ez egy kis változás, amelynek nagy hatása van a digitális biztonságra.*” (Lander, 2023)

A fenti körülmények okán IT-biztonsági szempontok tudnak érvényesülni a szerveroldali szoftver vonatkozásában is.

3.2.2.3 Mesterséges intelligencia használata

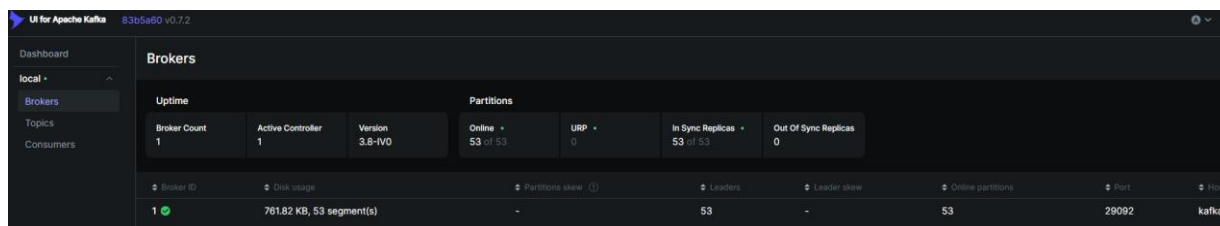
A fejlesztés során felmerült, hogy az API projekteket Minimal API vagy Controller megközelítéssel valósítsam-e meg (forrás a kérdéskör kapcsán: <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/apis>). Ebben kértem ki a Copilot nevű generatív mesterséges intelligencia segítségét. A diskurzus a 8. mellékletben olvasható, végül a Minimal API megközelítést választottam.

3.2.3 Eseményfolyam-kezelő

Ezen alfejezetben bemutatom a felhasználásra kerülő, a szerveroldali kommunikáció szempontjából nélkülözhetetlen eseményfolyam-kezelő platform megvalósítását. Szakirodalmi

kitekintés a [2.5 Eseményfolyam-kezelő technológiák](#) fejezetben, a szoftver tervezésének eseményfolyam-kezelésére vonatkozó aspektusai pedig a [3.1.1.3 Eseményfolyam-kezelő platform](#) fejezetben található.

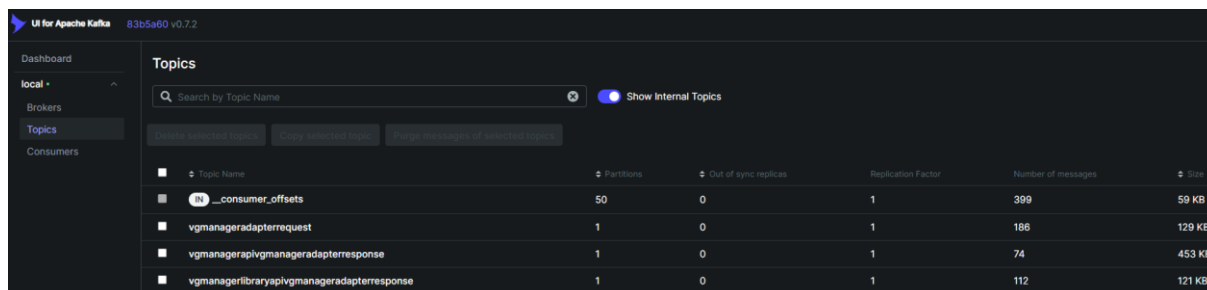
Ennek a bemutatását a futtatott Apache Kafka példány által biztosított felhasználói interfész segítségével fogom elvégezni (Lásd 44. ábra, 45. ábra és 46. ábra):



44. ábra: Kafka UI kezelőfelület, brókerek (saját szerkesztés)

Jelmagyarázat, 44. ábra:

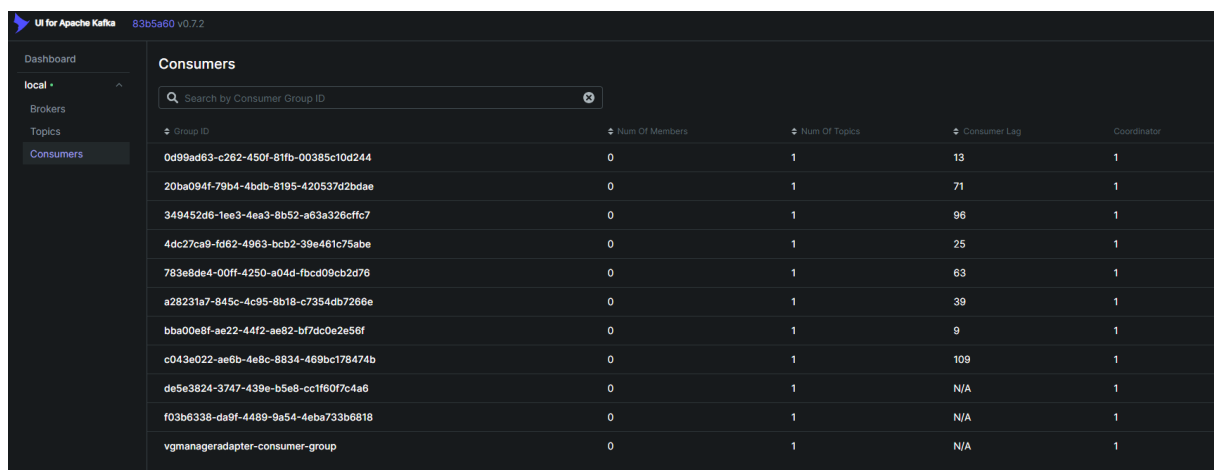
- Láthatóak a szakirodalmi háttérben (Lásd [2.5 Eseményfolyam-kezelő technológiák](#)) taglalt komponensek, így azok jelentésének részletes magyarázatára itt nem vállalkozom.
 - Látható, hogy elérhető egy bróker, ami az üzenetközvetítésért felelős
 - Látható, hogy elérhetőek partíciók (53 darab), amelyek a topikok szétdarabolt reprezentánsai
 - Látható, hogy a host szervernek kafka lett az elnevezése (általám lett elnevezve, a futtatási konfigurációban beállítva)
 - Látható az is, hogy a fenti entitások online elérhetőek, így csatlakozni is lehet rájuk.



45. ábra: Kafka UI kezelőfelület, topikok (saját szerkesztés)

Jelmagyarázat, 45. ábra:

- Láthatóak a szakirodalmi háttérben (Lásd [2.5 Eseményfolyam-kezelő technológiák](#)) taglalt komponensek, így azok jelentésének részletes magyarázatára itt nem vállalkozom.
 - o Topikok, amelyek a partíciók tömbösített változataként is tekinthetők. Ezekre az entitásokra „iratkoznak fel” a Kafkára csatlakozó komponensek (2 db API és 1 db Adapter)
 - A topikok elnevezése önkényesen történt, a felhasználó fejlesztő megadhat bármilyen nevezéktant



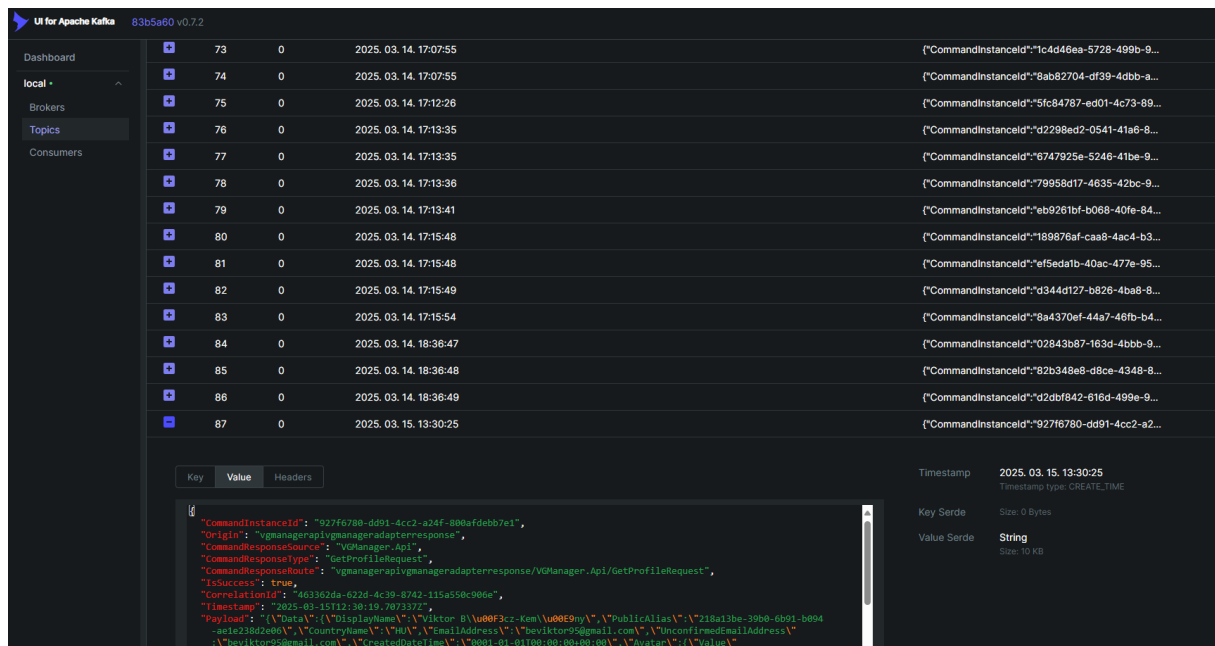
Group ID	Num Of Members	Num Of Topics	Consumer Lag	Coordinator
0d99ad63-c282-450f-81fb-00385c10d244	0	1	13	1
20ba094f-79b4-4bdb-8195-420537d2bdae	0	1	71	1
349452d6-1ee3-4ea3-8b52-a63a326cfc7	0	1	96	1
4dc27ca9-fd62-4963-bcb2-39e461c75abe	0	1	25	1
783e8de4-00ff-4250-a04d-fbcd09cb2d76	0	1	63	1
a28231a7-845c-4c95-8b18-c7354db7266e	0	1	39	1
bba00e8f-ae22-44f2-ae82-bf7dc0e2e56f	0	1	9	1
c043e022-ae6b-4e8c-8834-469bc178474b	0	1	109	1
de5e3824-3747-439e-b5e8-cc1f60f7c4a6	0	1	N/A	1
f03b6338-da9f-4489-9a54-4eba733b6818	0	1	N/A	1
vgmanageradapter-consumer-group	0	1	N/A	1

46. ábra: Kafka UI kezelőfelület, fogyasztók (saját szerkesztés)

Jelmagyarázat, 46. ábra:

- Láthatóak a szakirodalmi háttérben (Lásd [2.5 Eseményfolyam-kezelő technológiák](#)) taglalt komponensek, így azok jelentésének részletes magyarázatára itt nem vállalkozom.
 - o Láthatóak a fogyasztók, akik a topikokra „iratkoznak fel”
 - IT-biztonsági szempontból itt a regisztrált komponensek vagy el vannak maszkírozva random generált GUID (Globally Unique Identifier) azonosítókkal
 - Vagy pedig fogyasztói csoportba (pl.: vgmanageradapter-consumer-group) vannak rendezve, amiből következtetni lehet, hogy pontosan kinek az üzeneteire vannak csatlakozva (2 db API fog ebbe a csoportba tartozni ebben az esetben)

Az üzenetek megtekintésére is van lehetőség az említett kezelőfelületen, be is mutatnék egy példát (Lásd 47. ábra):



47. ábra: Kafka UI kezelőfelület, üzenetek (saját szerkesztés)

Jelmagyarázat, 47. ábra:

- Maradva a topikok felületen (Lásd 45. ábra)
 - Lenyithatóak a topikokhoz tartozó üzenetek
 - Az egyik üzenet tartalmát részben mutatom csak meg IT-biztonsági okokból (azonosítók láthatóak a teljes üzenetben)
 - Mindenesetre látható, hogy a válasz, ami az Adapterből érkezett a topikra, az egyik API példánynak lett szánya feldolgozás céljából

Bemutatásra kerülhetne még a szerveroldali komponensek Kafka konfigurációja is, viszont ennek bemutatása nem lehetséges a megrendelő által lefektetett, zárt forráskódra vonatkozó klauzula miatt (Lásd [1.6.4.1 Megrendelői elvárások](#)).

3.2.3.1 IT-Biztonság

Az említett fogyasztók kiléte nem derül ki a Kafka implementációja során, mert vagy GUID azonosítókkal vannak titkosítva, vagy pedig csoportokba vannak rendezve (amennyiben egy topikra többen is fel vannak iratkozva ez szükséges), ami szintén ellehetetleníti a kilétük megismerését.

3.2.3.2 *Mesterséges intelligencia használata*

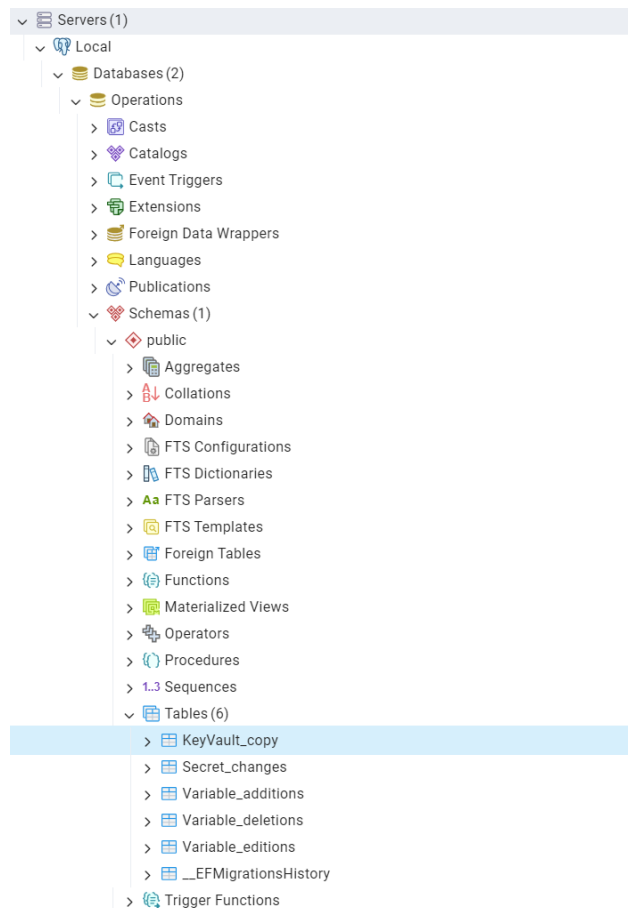
A fejlesztés során felmerült, hogy pontosan milyen követelmények szükségeltetnek az Apache Kafka példányának futtatásához. Ebben kértem ki a Copilot nevű generatív mesterséges intelligencia segítségét. A diskurzus a 8. mellékletben olvasható.

3.2.4 Adatbázis

Ezen fejezetben bemutatom az adatbázisra vonatkozó szakdolgozati aspektusok gyakorlati megvalósítását. Szakirodalmi háttér a [2.10 Adatbázisok](#) fejezetben, a szoftvertervezési vonatkozások pedig a [3.1.1.4 Adatbázis](#) fejezetben olvasható.

A gyakorlati megvalósításban az adatbáziskezelésért felelős szerveroldali komponens a Library API lett, ugyanis ebben az API komponensben valósul meg a környezeti változó, secret és key vault funkcionalitásokkal kapcsolatos üzleti logika. Az adattáblákban eltárolt adatok pedig ezen üzleti logikákkal vannak összefüggésben (perzisztálni kell az említett entitásokra vonatkozó módosításokat), ahogyan ez a megrendelői elvárásokban is olvasható (Lásd [1.6.4.1 Megrendelői elvárások](#)).

Ahogyan a [3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése](#) fejezetben is látható, a webfelületen megjelennek bizonyos naplóbejegyzések a korábbi módosításokat illetően. Az eltárolt adatokat a következő adatbázisban és adattáblákban perzisztáltuk (bemutatva ezen adatokat a pgAdmin adatbáziskliens használatával) (Lásd 48. ábra):



48. ábra: Adatbázis és kapcsolódó adattáblák pgAdmin adatbáziskliens használatával (saját szerkesztés)

Jelmagyarázat, 48. ábra:

- Local: Lokálisan futtatott PostgreSQL szerverpéldányra utal
- Databases: Elérhető adatbázisok, a szakdolgozat szempontjából az Operations nevű adatbázis lesz érdekes
 - o Operations adattáblák:
 - KeyVault_copy: key vault másolások
 - Secret_changes: Secret módosítások
 - Variable_additions: Változó hozzáadás
 - Variable_deletions: Változó törlés
 - Variable_editions: Változó felülírás
 - __EFMigrationsHistory: Adatmigrációt segítő adattábla

A felsorolt adattáblák beszédes elnevezésűek (talán az utolsót leszámítva), így azok részletesebb ismertetését nem tartom szükségesnek. Mindenesetre az egyik adattábla rekordjait

érdeemes megjeleníteni, hogy láthassuk, szinkronicitás mutatkozik a [3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése](#) fejezetben látottakkal (Lásd 49. ábra):

The screenshot shows the pgAdmin interface with the 'Operations/postgres@Local' tab active. The 'Query' tab displays the following SQL query:

```
1 SELECT "Id", "KeyVaultName", "User", "SecretNameRegex", "ChangeType", "Date"
2 FROM public."Secret_changes";
```

The 'Data Output' tab shows the results of the query in a table with 7 columns: Id, KeyVaultName, User, SecretNameRegex, ChangeType, and Date. The table contains 4 rows of data.

Id	KeyVaultName	User	SecretNameRegex	ChangeType	Date
d1732bc9-3dd5-4cee-b128-1bfc169fba...	KeyVault-BKV	Viktor Bócz-Kemény	Secret1	1	2025-03-11 13:43:16.2349+00
53b0c5a8-3390-4641-9413-3930cbfe5a...	KeyVault-BKV	Viktor Bócz-Kemény	Secret1	1	2025-03-11 20:26:15.198596+00
678d1056-daf4-4aa0-9ccb-edfb347a42...	KeyVault-BKV	Viktor Bócz-Kemény	Secret1	0	2025-03-11 20:26:36.002846+00
658accba-9faa-4790-bd4c-4f67360a325f	KeyVault-BKV	Viktor Bócz-Kemény	Secret1	1	2025-03-11 20:27:27.889563+00

49. ábra: Adatbázis, secret módosítások lekérdezése pgAdmin adatbáziskliens használatával (saját szerkesztés)

Jelmagyarázat, 49. ábra:

- 49. ábrán is megtalálható elemek megegyeznek jelentésükben a 48. ábrához tartozó jelmagyarázatban szereplőkkel
- Látható, hogy a lekérdezés (Select SQL utasítás) eredménye szinkronban van a 30. ábrán látható webes reprezentációval

3.2.4.1 IT-Biztonság

A szoftverhez tartozó adatbázis adatbáziskapcsolatának létrehozásához felhasználó és jelszó szükséges, amelyek megléte nélkül nem lehetséges az adatbázis elérése (Lásd 50. ábra):

The screenshot shows a 'Register - Server' window with the 'Connection' tab selected. The form includes the following fields and controls:

- Host name/address: An empty text input field.
- Port: A text input field containing '5432'.
- Maintenance database: A text input field containing 'postgres'.
- Username: A text input field containing 'postgres'.
- Kerberos authentication?: A toggle switch that is currently turned off.
- Password: An empty password input field.
- Save password?: A toggle switch that is currently turned off.
- Role: An empty text input field.
- Service: An empty text input field.

A red error message bar at the bottom of the form area displays the text: 'Name' cannot be empty. The dialog box has a title bar with a close button (X) and a tabbed interface with tabs for General, Connection, Parameters, SSH Tunnel, Advanced, and Tags. At the bottom of the dialog are buttons for Close, Reset, and Save, along with information and help icons.

50. ábra: Adatbázis, kapcsolat létrehozása (saját szerkesztés)

Jelmagyarázat, 50. ábra:

- Host name: szerver neve
- Port: Szerverhez tartozó publikus port elérése
- Username: Felhasználó
- Password: Jelszó
- A kliens által előre kitöltött mezők nem biztos, hogy helyesek (például: port, username)
- Látható, hogy a jelszó kitöltendő mezőként szerepel a kapcsolatlétrehozás során (ellenben nincsen feltüntetve az alsó figyelmeztető sávban, mint kitöltendő mező, ugyanis nem minden adatbázis esetében elvárt a jelszó használata)

3.2.5 Szoftverfejlesztést és üzembehelyezést támogató folyamatok

Ebben a fejezetben a szoftverfejlesztést és üzembehelyezést támogató folyamatokat mutatom be, amelynek részei: Git verziókezelés (Lásd [3.2.5.1 Git verziókezelés](#) fejezet) a forráskód verziózáshoz, CI/CD folyamat (Lásd [3.2.5.2 CI/CD folyamat](#) fejezet) a folyamatos integrációért, statikus kódelemzés (Lásd [3.2.5.3 Statikus kódelemzés](#) fejezet) a minőség magasabb szintű biztosításáért.

Továbbá az üzembehelyezést segítő megoldásokat is bemutatom, amelyek a Docker, Docker-Compose és Kubernetes használatát jelenti.

Ezek mellett kitérek az IT-biztonsági (Lásd [3.2.5.7 IT-Biztonság](#) fejezet) és mesterséges intelligenciára (Lásd [3.2.5.8 Mesterséges intelligencia használata](#) fejezet) vonatkozó aspektusokra is.

3.2.5.1 *Git verziókezelés*

A vonatkozó szakirodalmi háttér a [2.8 Verziókezelő rendszerek](#) fejezetben taglaltam, illetve a tervezési fejezetben (Lásd [3.1.2.1 Verziókezelés](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

Ahogy erről a hivatkozott tervezési fejezetben is írtam, a Git Flow stratégiát választottam branching stratégiának, így a main és develop branch-ek mellett feature branch-eket használtam a különböző funkcionálisok implementálásához, illetve hotfix branch-eket a felmerülő programhibák elhárításához.

A távoli tárhely szolgáltatáshoz pedig a GitHub platformot használtam, szeparáltan minden megírt szoftverkomponens számára.

3.2.5.2 *CI/CD folyamat*

A vonatkozó szakirodalmi háttér a [2.11 Folyamatos Integráció, Folyamatos Szállítás](#) fejezetben taglaltam, illetve a tervezési fejezetben (Lásd [3.1.2.4 Folyamatos Integráció, Folyamatos Szállítás tervezés](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

Az implementált build pipeline-ok több, a kódminőséget, a kitelepítést és az integrációt is biztosító műveletet végeznek el a beállított branch státuszmódosulások esetén. Az összes pipeline bemutatására nem vállalkozom a terjedelmük miatt, viszont egyes pipeline-ok bemutatását mindenképp megteszem.

Szerveroldali forráskódintegrációt biztosító build pipeline (Lásd 51. ábra):

```
1  # This workflow will build a .NET project
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-net
3
4  name: Build and run tests
5
6  on:
7    push:
8      branches: [ "main", "develop" ]
9    pull_request:
10     branches: [ "main", "develop" ]
11
12  jobs:
13    build:
14
15     runs-on: ubuntu-latest
16
17     steps:
18     - uses: actions/checkout@v3
19
20     - name: Setup .NET
21       uses: actions/setup-dotnet@v3
22       with:
23         dotnet-version: 8.0.x
24
25     - name: Restore dependencies
26       working-directory: ./src
27       run: dotnet restore
28
29     - name: Build
30       working-directory: ./src
31       run: dotnet build --no-restore
32
33     - name: Test
34       working-directory: ./src
35       run: dotnet test --no-build --verbosity normal
```

51. ábra: Forráskód integrációt biztosító pipeline, szerveroldal (saját szerkesztés)

Jelmagyarázat, 51. ábra:

- 15. sor: runs-on: ubuntu-latest: A build pipeline futtat egy virtuális gépet, amelynek az operációs rendszere Ubuntu utolsó stabil verziója lesz
- 6. sortól a 10. sorig: Amennyiben a Git push parancs, vagy pull request létrehozása történik az alábbi branch listán (develop és main), az alábbi lépések történjenek meg:
 - o 20. sortól a 23. sorig: Tegye elérhetővé a pipeline-t futtató virtuális gépen a .NET 8 keretrendszer-t. Sikertelenség esetén a pipeline hibára fut
 - o 25. sortól a 27. sorig: Adja ki a dotnet restore parancsot, aminek a segítségével az összes, a szoftverkomponens által használt, harmadik fél által készített szoftvercsomagot telepítse. Sikertelenség esetén a pipeline hibára fut.
 - o 29. sortól a 31. sorig: Fordítsa a forráskódot MSIL (Microsoft Intermediate Language) forráskódállományra, így ellenőrizve, hogy nincs fordítási hiba az új verzióban. Sikertelenség esetén a pipeline hibára fut

- 33. sortól a 35. sorig: Lefuttatja a megírt, forráskódban található teszteket. Sikertelenség esetén a pipeline hibára fut.

Üzemeltetési és kitelepítési integrációt biztosító, szerveroldalra vonatkozó pipeline (Lásd 52. ábra):

```

1  name: Build and Push Docker Image
2  on:
3    push:
4      branches:
5        - main
6
7  jobs:
8    push_to_registry:
9      name: Push Docker image to Docker Hub
10     runs-on: ubuntu-latest
11     steps:
12       - name: Check out the repo
13         uses: actions/checkout@v3
14
15       - name: Log in to Docker Hub
16         uses: docker/login-action@f4ef78c080cd8ba55a85445d5b36e214a81df20a
17         with:
18           username: ${ secrets.DOCKERHUB_USERNAME }
19           password: ${ secrets.DOCKERHUB_PASSWORD }
20
21       - name: Extract metadata (tags, labels) for Docker
22         id: meta
23         uses: docker/metadata-action@9ec57ed1fcd8f14dcef7dfbe97b2010124a938b7
24         with:
25           images: boczvikt/vgmanager-backend
26
27       - name: Build and push Docker image
28         uses: docker/build-push-action@3b5e8027fcad23fda98b2e3ac259d8d67585f671
29         with:
30           context: ./src/
31           file: ./src/Dockerfile
32           push: true
33           tags: boczvikt/vgmanager-backend:latest
34           labels: ${ steps.meta.outputs.labels }

```

52. ábra: Üzemeltetési és kitelepítési integrációt biztosító pipeline, szerveroldal (saját szerkesztés)

Jelmagyarázat, 52. ábra:

- 11. sor: runs-on: ubuntu-latest: A build pipeline futtat egy virtuális gépet, amelynek az operációs rendszere Ubuntu utolsó stabil verziója lesz
- 2. sortól a 5. sorig: Amennyiben a Git push művelet történik a main branch-en, az alábbi lépések történjenek meg:
 - 15. sortól a 19. sorig: Jelentkezzen be a GitHub secret csoportban eltárolt DockerHub hitelesítő adatokkal a DockerHub fiókba.

- 21. sortól a 25. sorig: Kivonatolja a metaadatokat a Docker image elkészítéséhez és felküldéséhez.
- 27. sortól a 34. sorig: A docker build utasítás megadásával a projektben található Dockerfile segítségével készítsen Docker image-t, majd a megadott és bejelentkezett DockerHub fiókra töltse fel az elkészült image-t.

Ennek az 52. ábrának jelentősége lesz a [3.2.5.4 Docker](#), a [3.2.5.5 Docker-Compose](#) és a [3.2.5.6 Kubernetes](#) fejezetek szempontjából, ugyanis az említett ábra és hozzá tartozó CI/CD folyamat már az üzemeltetést segíti elő.

Statikus kódanalízist lehetővé tevő és így a kódminőséget erősítő pipeline (Lásd 52. ábra):

```
name: SonarCloud
on:
  push:

jobs:
  build:
    name: Sonar
    runs-on: windows-latest
    steps:
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: 17
          distribution: 'zulu' # Alternative distribution options are available.

      - uses: actions/checkout@v3
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis

      - name: Install Sonarqube-related dependencies
        working-directory: ./src
        run: |
          dotnet tool install --global dotnet-sonarscanner
          dotnet tool update dotnet-reportgenerator-globaltool -g

      - name: Analyze
        working-directory: ./src
        run: |
          dotnet sonarscanner begin /k:"BoczyVgmanager_backend" /o:"Boczy" /d:sonar.host.url="https://sonarcloud.io" /d:sonar.login="${{ secrets.SONAR_TOKEN }}" /d:sonar.coverageReportPaths=".\\sonarqubecoverage\\SonarQube.xml"
          dotnet build
          dotnet test --no-build --collect:"XPlat Code Coverage"
          reportgenerator --reports:"\\TestResults\\coverage.cobertura.xml" --targetdir:sonarqubecoverage --reporttypes:SonarQube
          dotnet sonarscanner end /d:sonar.login="${{ secrets.SONAR_TOKEN }}"
```

53. ábra: Statikus kódanalízist biztosító pipeline, szerveroldal (saját szerkesztés)

Jelmagyarázat, 53. ábra:

- 8. sor: runs-on: windows-latest: A build pipeline futtat egy virtuális gépet, amelynek az operációs rendszere a Windows utolsó stabil verziója lesz
- 2. sortól a 4. sorig: Amennyiben a Git push művelet történik bármelyik branch-en, az alábbi lépések történjenek meg:
 - 10. sortól a 14. sorig: Tegye elérhetővé a JDK-t (Java Development Kit) a Sonar kódanalizáló szoftver működtetéséhez.
 - 20. sortól a 24. sorig: Sonar függőségeinek letöltése a Sonar kódanalizáló szoftver működtetéséhez.
 - 26. sortól a 33. sorig:
 - Végezze el lokálisan az analízist (kódfordítással, tesztek futtatásával).

- Készítsen riportot.
- A GitHub secret csoportban eltárolt Sonar hitelesítő adatokkal töltsse fel a Sonar profilba a riportot.

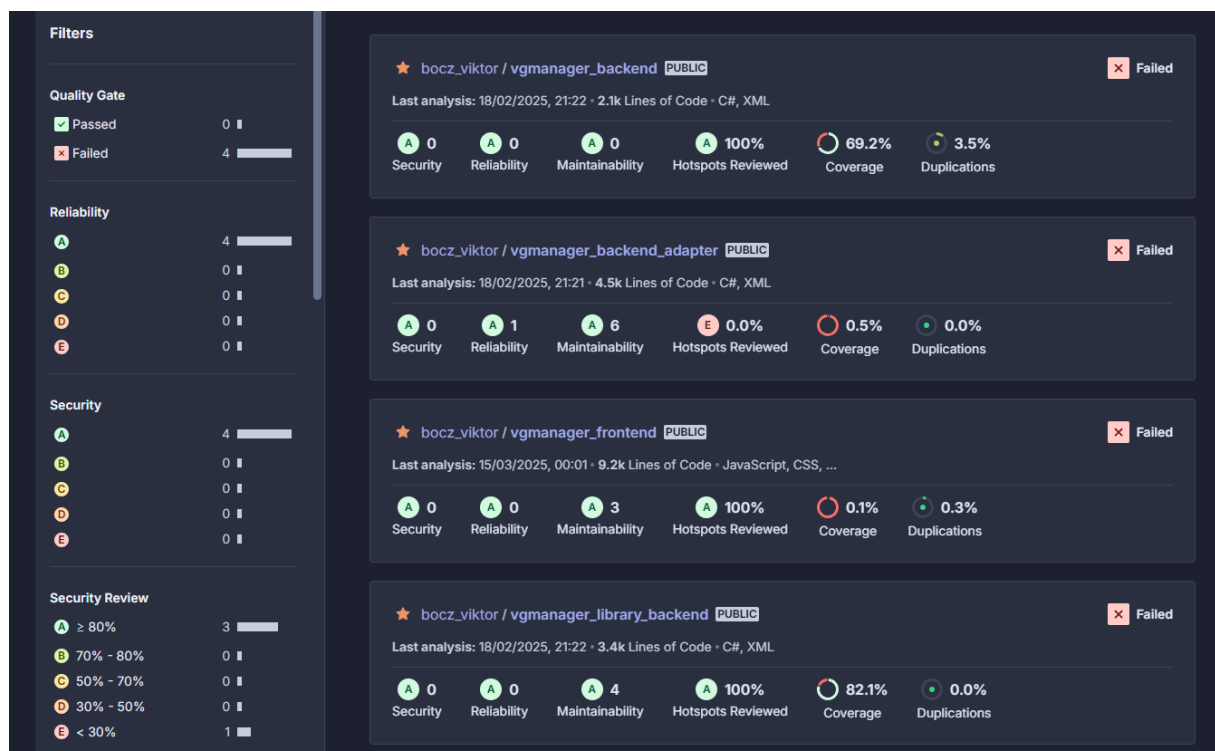
Ennek az 53. ábrának jelentősége lesz a [3.2.5.3 Statikus kódelemzés](#) fejezet szempontjából, ugyanis az említett ábra és hozzá tartozó CI/CD folyamat már a statikus kódelemzést teszi lehetővé.

A webfelület megvalósítását szolgáló projekt esetén is létre vannak hozva ezen pipeline-ok ReactJS-hez tartozó megfelelői.

3.2.5.3 Statikus kódelemzés

A vonatkozó szakirodalmi háttérrel a [2.9 Statikus kódelemzés](#) fejezetben taglalтам, illetve a tervezési fejezetben (Lásd [3.1.2.3 Statikus kódelemzés](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

Ebben az alfejezetben bemutatom a statikus kódelemzésre használt megoldásomat, ami a SonarCloud platformot jelenti. Az alábbi ábra szolgál ennek bemutatásául (Lásd 54. ábra):



54. ábra: Statikus kódelemzés analíziseket összesítő kezelőfelület (saját szerkesztés)

Jelmagyarázat, 54. ábra:

- Mindegyik kártyaelem a jobboldali traktusban egy-egy implementált projektet jelent

- A kártyaelem alsó traktusában találhatóak az analízis különböző metrikáit és a kiértékelés eredményét is
 - o Security: Kódbiztonság
 - o Reliability: Forráskód megbízhatósága
 - o Maintainability: Forráskód karbantartottsága
 - o Hotspots reviewed: További kódbiztonsági aspektus
 - o Coverage: Tesztlefedettség
 - o Duplications: Kódduplikáció

Statikus kódanalízis használata nem volt elvárás a megrendelői oldalról (Lásd [1.6.4.1 Megrendelői elvárások](#)), így ennek részletesebb bemutatása nem is volna szinkronban a szakdolgozat fő kérdéseivel. Mindenesetre megemlítendő, hogy ennek az aspektusnak a részletesebb kidolgozása és bemutatása lehetne egy jövőbeli feladat is, amennyiben a megrendelő nyitott a további együttműködésre.

3.2.5.4 Docker

A vonatkozó szakirodalmi háttérrel a [2.7.1 Docker](#) fejezetben taglaltam, illetve a tervezési fejezetben (Lásd [3.1.2.6 Konténerizáció, konténerizáltan futtatható állományok tárolása](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

Ebben az alfejezetben bemutatom a használt konténerizációs eszközt, ami a Dockert jelenti. Az alábbi, Dockerfile implementációkról szóló ábrák szolgálnak ennek bemutatásul (Lásd 55. ábra és 56. ábra):

```
# Build Stage
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY . .
RUN dotnet restore "./VGManager.Adapter.Api/VGManager.Adapter.Api.csproj"
RUN dotnet publish "./VGManager.Adapter.Api/VGManager.Adapter.Api.csproj" -c release -o /app --no-restore

# Serve Stage
FROM mcr.microsoft.com/dotnet/aspnet:8.0
WORKDIR /app
COPY --from=build /app ./

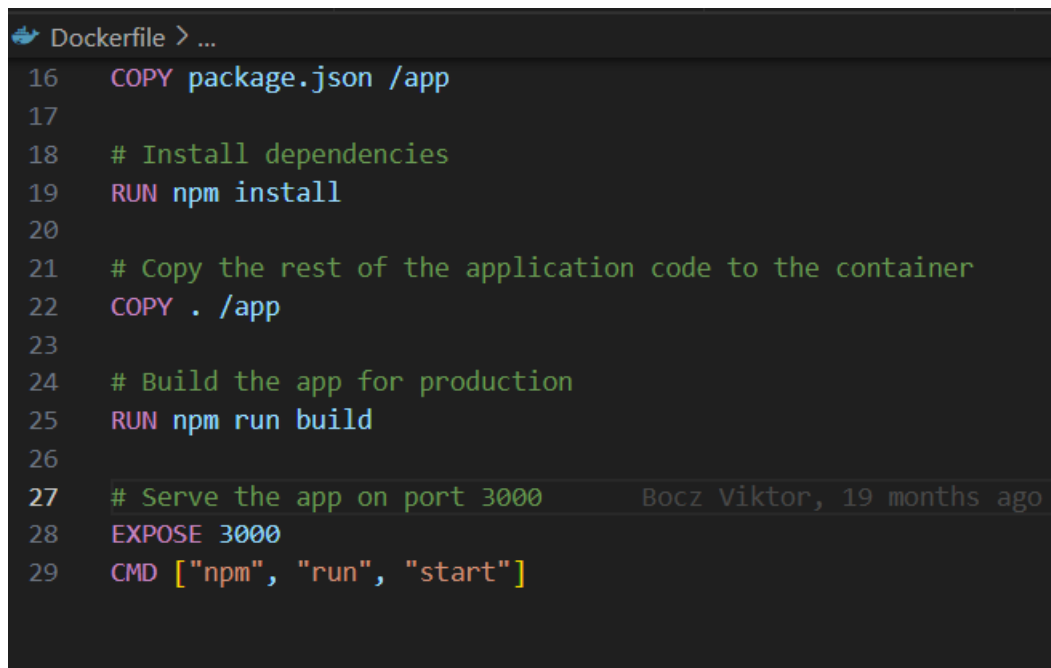
EXPOSE 5004

ENTRYPOINT ["dotnet", "VGManager.Adapter.Api.dll"]
```

55. ábra: Szerveroldali komponens Dockerfile tartalom (saját szerkesztés)

Jelmagyarázat, 55. ábra:

- Első FROM utasítással kezdődő sor: Az alap, .NET SDK-t tartalmazó konténer image kiválasztása, amely tartalmaz minden eszközt, amely szükséges a forráskódban definiált tesztek futtatásához, a forráskód fordításához és a fordított állomány publikálásához. Erre fog ráépülni a szoftver fordításához szükséges logikákat implementáló image rétegek.
- WORKDIR utasítással kezdődő sor: Meghatározza, hogy melyik, az image-ben megtalálható vagy meg nem található könyvtár (ebben az esetben létre is hozza) legyen az image munkaterülete
- COPY utasítással kezdődő sor: Átmásolja a forráskódot tartalmazó könyvtárakat az alap konténer image-ben meghatározott munkaterületre, fordítási célból
- RUN dotnet restore és publish sorok: Fordítja a forráskódot futtatható állománnyá
- Második FROM utasítással kezdődő sor: Az alap, .NET futtató környezetet tartalmazó konténer image kiválasztása, amivel a fordított MSIL állományokat futtatni lehet.
- Második WORKDIR utasítással kezdődő sor: Meghatározza, hogy melyik, az image-ben megtalálható vagy meg nem található könyvtár (ebben az esetben létre is hozza) legyen az image munkaterülete
- Második COPY utasítással kezdődő sor: Átmásolja a forráskódot tartalmazó könyvtárakat az alap konténer image-ben meghatározott munkaterületre, futtatási célból
- EXPOSE utasítással kezdődő sor: Megnyitja a numerikus paraméterként megadott konténer portot konténeren kívüli kommunikációra
- ENTRYPOINT utasítással kezdődő sor: Megadja a szoftver belépési ponthoz tartozó fordított DLL fájlját, dotnet utasítással



```
Dockerfile > ...
16 COPY package.json /app
17
18 # Install dependencies
19 RUN npm install
20
21 # Copy the rest of the application code to the container
22 COPY . /app
23
24 # Build the app for production
25 RUN npm run build
26
27 # Serve the app on port 3000 Bocz Viktor, 19 months ago
28 EXPOSE 3000
29 CMD ["npm", "run", "start"]
```

56. ábra: Webkliens komponens Dockerfile tartalom (saját szerkesztés)

Jelmagyarázat, 56. ábra:

- Az 56. ábrán is megtalálható elemek megegyeznek jelentésükben a 55. ábrához tartozó jelmagyarázatban szereplőkkel
- CMD utasítással kezdődő sor: Megadja a konténer futtatásához szükséges futtatási utasítást (elindítja a webservert)

A hivatkozott két ábra alapján elkészítettem az általam megírt szoftverek Docker konténerizációját, ennek (és a külső függőségekhez tartozó konténerek elindításának) köszönhetően a szakdolgozati munka futtathatóvá válik bármely szerveren, amely rendelkezik Docker Engine-nel (Lásd 57. ábra és 58. ábra):

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers

Give feedback

View all your running containers and applications. [Learn more](#)

Container CPU usage

5.19% / 2000% (20 CPUs available)

Container memory usage

1.67GB / 15.1GB

Show charts

Q

Search

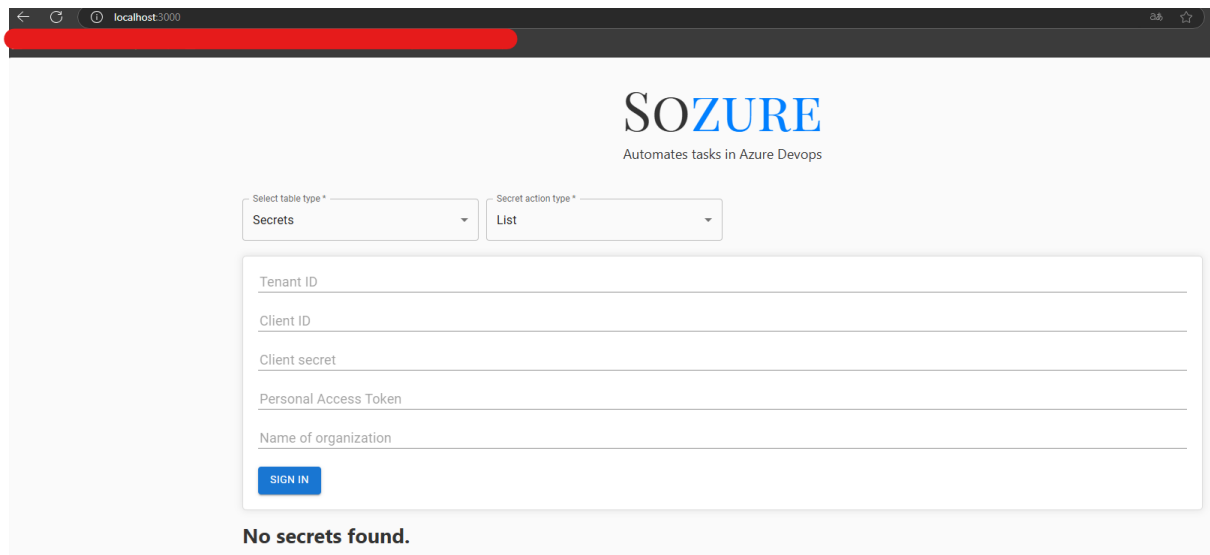
Only show running containers

☐	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
☐	☐ minikube	d37c2e9cf8ca	k8s-minikube/kicbase-v0.0.2	0.22 Show all ports (5)	0%	15 days ago	
☐	vg_folder	-	-	-	5.21%	2 minutes ago	
☐	☒ frontend	a4f2f98879d	boczviktor/vgmanager-fron	3000:3000 ?	0.01%	2 minutes ago	
☐	☒ library-backend	33e7f846494b	boczviktor/vgmanager-libra	5002:5002 ? Show all ports (2)	0.18%	2 minutes ago	
☐	☒ backend	429e21c183fa	boczviktor/vgmanager-bad	5000:5000 ? Show all ports (2)	0.21%	2 minutes ago	
☐	☒ backend-adapter	f4ab79475c71	boczviktor/vgmanager-bad	5004:5004 ? Show all ports (2)	0.34%	2 minutes ago	
☐	☒ kafka-ui	14cfbb3249cb	provectuslabs/kafka-ui:late	8080:8080 ?	0.14%	2 minutes ago	
☐	☒ kafka	b4e7f7910898	confluentinc/cp-kafka:lates	29092:29092 ? Show all ports (2)	4.13%	2 minutes ago	
☐	☒ db	e8dc11fbcba0	postgres:latest	5432:5432 ? Show all ports (2)	0.03%	2 minutes ago	
☐	☒ zookeeper	c668809b3b6e	confluentinc/cp-zookeeper:	22181:2181 ?	0.17%	2 minutes ago	

57. ábra: Szakdolgozati projekt komponenseinek futtatott konténerei (saját szerkesztés)

Jelmagyarázat, 57. ábra:

- A Containers oldalon a futó vagy épp leállított státuszban lévő konténerek láthatóak
- Saját készítésű szoftverből fordított konténerek:
 - o frontend: Webkliens
 - o library-backend: Szerveroldal
 - o backend: Szerveroldal
 - o backend-adapter: Szerveroldal
- Külső függőségekhez tartozó konténerek:
 - o db: Adatbázis
 - o zookeeper: Kafka függőség a rendelkezésre álláshoz
 - o kafka: Eseményfolyam-kezelő példány
 - o kafka-ui: Eseményfolyam-kezelő példányhoz tartozó monitoring és controlling példány (webes megjelenítéssel)



58. ábra: Elérhető webkliens és szoftver a szerver (localhost, saját gép) 3000-es portján (saját szerkesztés)

Jelmagyarázat, 58. ábra:

- A futtatott konténereknek köszönhetően elérhető a szakdolgozati munka a szerveren

3.2.5.5 Docker-Compose

A vonatkozó szakirodalmi háttérrel a [2.7.2 Docker Compose](#) fejezetben taglaltam, illetve a tervezési fejezetben (Lásd [3.1.2.7 Kitelepítést és futtatást támogató egyéb technológiák](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

A Docker-Compose YAML konfigurációs fájl nem mutatható be ábrával, mert tartalmaz érzékeny, IT-biztonsági szempontok alapján be nem mutatható adatokat (például adatbázis kapcsolat jelszó).

Azzal mindenesetre kiegészíteném a [3.2.5.4 Docker](#) fejezetet, hogy az 57. ábrán látható konténereket a Docker-Compose YAML fájl futtatásával indítottam el, elkerülve a Docker image-k manuális elkészítését, majd az ebből készülő konténerek manuálisan elvégzett konfigurálását és indítását. Ennek tükrében az 57. ábra szolgálhat egyfajta bizonyítékként a Docker-Compose eszköz használatára.

Kiemelendő, hogy ezen üzemeltetési megoldások nem lettek megrendelve a megrendelő által (Lásd [1.6.4.1 Megrendelői elvárások](#)), ezen feladatok a jövőbeli további megrendelések reményében kerültek elvégzésre.

3.2.5.6 *Kubernetes*

A vonatkozó szakirodalmi háttérrel a [2.7.3 Kubernetes](#) fejezetben taglaltam, illetve a tervezési fejezetben (Lásd [3.1.2.7 Kitelepítést és futtatást támogató egyéb technológiák](#)) a munka elvégzéséhez szükséges döntéseket is meghoztam.

A Kubernetes konténermenedzselő szoftver szakdolgozati implementációját két ábrán keresztül fogom bemutatni (Lásd 59.ábra és 60. ábra).

Mivel a Kubernetes manifest fájlok tartalmazzak érzékeny, IT-biztonsági szempontok alapján be nem mutatható adatokat (például adatbázis kapcsolat jelszó), így a webkliens manifestjének (önkényesen kiválasztva a manifestek közül, bemutató gyanánt) azon részeit fogom bemutatni, amelyek nem tartalmazzák az említett adatokat (Lásd 59. ábra):

```

38 ---
39 apiVersion: apps/v1
40 kind: Deployment
41 metadata:
42   name: frontend
43   labels:
44     app: frontend
45   annotations:
46     configmap.reloader.stakater.com/reload: "frontend-config"
47   namespace: default
48 spec:
49   replicas: 1
50   selector:
51     matchLabels:
52       app: frontend
53   template:
54     metadata:
55       labels:
56         app: frontend
57     spec:
58       containers:
59         - name: frontend
60           image: boczviktorg/vgmanager-frontend:latest
61           ports:
62             - containerPort: 3000
63           envFrom:
64             - configMapRef:
65               name: frontend-config
66           resources:
67             requests:
68               cpu: 0.02
69               memory: 128Mi
70               ephemeral-storage: 1Mi
71             limits:
72               cpu: 0.3
73               memory: 320Mi
74               ephemeral-storage: 100Mi
75 ---

```

59. ábra: Webkliens Kubernetes manifest Deployment-je (saját szerkesztés)

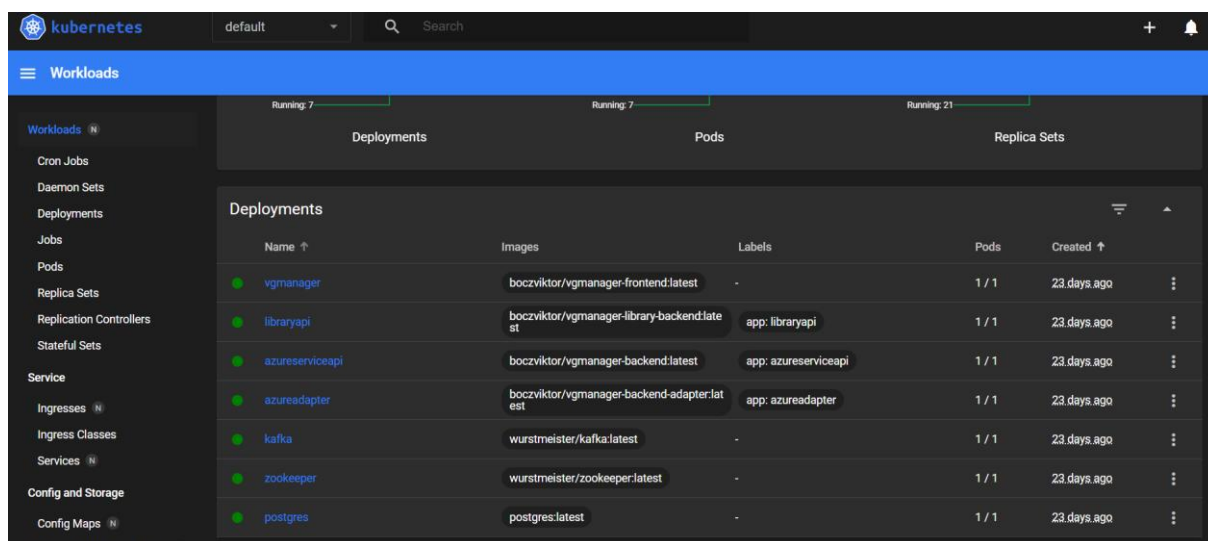
Jelmagyarázat, 59. ábra:

- Kubernetes Deployment jelentése a Kubernetes ökoszisztémában: Kubernetes Podok és ReplicaSet-ek kontrollíngjával foglalkozik (Lásd [2.7.3 Kubernetes](#))
- Fontosabb részletek az ábra vonatkozásában:
 - o metadata:

- Deploymenthez kapcsolódó metaadatok beállítása
 - replicas: 1: Egy példányt kell futtatnia a Deploymentnek
 - containers szekció:
 - Meghatározza, hogy milyen néven jöjjön létre a Kubernetes Deployment és a hatásköre alá tartozó Kubernetes podok
 - image: Honnan töltsse le a Docker image-t, amiből a Podban konténert fog készíteni és futtatni
 - envFrom: Honnan szerezze a konfigurációkat (érzékeny adatok!)
 - resources: Milyen erőforrások allokálhatóak a Kubernetes Deploymentnek

A fenti példához hasonlóan megírásra kerültek ezen Kubernetes manifestek az üzemeltetés új szintre emeléséhez.

A Kubernetes manifestek alapján történő Kubernetes Cluster elindítás után a következő kezelőfelületet lehet megtekinteni a Kuberneteshez kapcsolódó kiegészítő szoftver (minikube) használatával (Lásd 60. ábra):



60. ábra: Lokálisan futtatott Kubernetes Cluster a szakdolgozati szoftver vonatkozásában (saját szerkesztés)

Jelmagyarázat, 60.ábra:

- A bemutatás szempontjából a Deployments szegmens érdekes
 - Látható, hogy a kafka-ui konténer kivételével (amelynek implementálása még hiányzik, mivel a komponens inszignifikáns a szoftver egészének működését nézve) mindegyik szoftverkomponensünknek van elérhető Kubernetes Deployment-je (szoftverkomponensek ellenőrzéséhez lásd 57. ábra)

- Látható továbbá az is, hogy mindegyik Kubernetes Deployment-hez tartozik 1-1 futó Kubernetes Pod is
- A Cluster futtatásának végeredménye pedig megegyezik az 58 ábrán látottakkal.

Kiemelendő, hogy ezen üzemeltetési megoldások nem lettek megrendelve a megrendelő által (Lásd [1.6.4.1 Megrendelői elvárások](#)), ezen feladatok a jövőbeli további megrendelések reményében kerültek elvégzésre.

3.2.5.7 IT-Biztonság

A szoftverfejlesztést és üzembehelyezést támogató folyamatok fejezetben számos IT-biztonsági aspektust lehet felfedezni, amelyek a következők:

- Git verziókezelés: Implementált Git branching stratégia lehetővé tette, hogy a szeparált branch-ek következtében a forráskódintegritás és a kollaboráció lehetősége biztosított legyen. Ezek pedig IT-biztonsági követelmények.
- CI/CD folyamat: A CI/CD pipeline-ok használnak bizonyos hitelesítési adatokat, ahogyan erről a fejezetben is írtam (pl.: SonarCloud és DockerHub fiókok belépési adatai). Ezen adatok biztonságát a GitHub által szolgáltatott érzékenyadat-menedzsment megoldással biztosítottam.
- Statikus kódelemzés:
 - Privát SonarCloud fiókon statikus kódanalízist vettem igénybe, amelynek eredményét csak az láthatja, akinek jogosultságot adok erre.
 - Az analízis metrikái között több IT-biztonságra vonatkozó is megtalálható (pl.: Security, Reliability)

3.2.5.8 Mesterséges intelligencia használata

Korábban nem használtam sokat a Docker technológiát, így a Dockerfile-ok megírását a Copilot nevű generatív mesterséges intelligenciára bíztam. A diskurzus a 8. mellékletben olvasható, a kapott eredményeket pedig testreszabás után felhasználtam.

3.3 Tesztelés

Ebben a fejezetben tesztesetekkel fogom bemutatni a szoftver pontos és stabil működését. A teszteseteket és az eredményeket mellékletként kapcsolom a szakdolgozathoz, az alfejezetekben csak a vizsgálat eredményét fogom összegezni.

3.3.1 Környezeti változó lekérdezés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.2 Környezeti változó manipulálás

3.3.2.1 Környezet változó hozzáadás

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.2.2 Környezeti változó módosítás

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.2.3 Környezeti változó törlés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.3 Secret lekérdezés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.4 Secret manipulálás

3.3.4.1 Secret törlés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.4.2 Secret reaktiválása

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.4.3 Key vault másolás

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.5 Módosítások lekérdezése

3.3.5.1 Környezeti változó módosítások lekérdezése

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.5.2 Secret módosítások lekérdezése

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.5.3 Key vault másolások lekérdezése

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.6 Pipeline futtatás

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.7 Utolsó szoftververzió lekérdezés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.8 Pull request létrehozás

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.9 Pull request lekérdezés

Vonatkozó tesztforgatókönyv a 12. mellékletben található.

Vonatkozó tesztforgatókönyv a 13. mellékletben található.

A hivatkozott mellékletek alapján a tesztet eredményességét sikeresnek tekintem.

3.3.10 Tesztelési összegzés

A fenti alfejezetek eredményeit figyelembe véve kijelenthető, hogy a tesztelési forgatókönyvben foglaltak sikeresen végrehajtásra kerültek és a teszteredmények alapján a szoftver megfelelt a tesztforgatókönyvben foglaltaknak.

4. Vita

Ebben a zárófejezetben a szakdolgozattal kapcsolatos önreflexív meglátásaimat fogom összegezni, lehetőséget biztosítva a szakmai vitának.

4.1 Magasabb szintű kódminőség és biztosabb működés

Véleményem szerint magasabb szintű kódminőséget eredményezett volna, amennyiben sikerül beszédesebb hibaüzenetekkel ellátni a felületet, ezzel erősítve a felhasználó intuitív felülethasználatát. Az említett kódminőségjavulást erősítette volna, ha sikerül azokat a folyamatokat eliminálni, ahol vagy nem is érkezik visszajelzés egy, az eszközhasználat során fellépő hiba esetén, vagy ami rosszabb: félrevezető hibaüzenetet kap a felhasználó. Ezekre példát könnyen találni a saját munka bemutatása során.

A szoftver megvalósítása és tesztelése során felmerült, hogy más, alternatív folyamatstruktúrák, webfelületrendezési irányelvek implementálásával tovább lehet növelni a feladatelvégzéssel járó erőforrásmegtakarítást. Magyarán szólva az adott feladat elvégzése kevesebb időt és/vagy számítási kapacitást igényelne.

Bár nem volt közvetlen elvárása a megrendelőnek a kódminőség analízisen és megírt teszteken alapuló biztosítását illetően, ez mit sem von le abból a tényből, hogy az analízis használata és a tesztírás gyakorlata hozzájárul egy szoftver magasabb szintű kódminőségéhez és stabilabb működéséhez.

4.2 Egyszerűbb technológiai portfólió

A szakdolgozati szoftver technológiai portfóliója a szakdolgozatban kifejtett körülmények és kutatási eredmények alapján meghozott döntések következménye. Felmerül a kérdés, hogy ugyanezen bemeneti paraméterek meglétével juthatott volna más következtetésre is a szoftver elkészítője? Véleményem szerint juthatott volna.

Természeteszerű, hogy egy problémára több eszköz is szolgálhat megoldásként, példának említtve a szakdolgozatban is tárgyalt egyéb, konkurens keretrendszereket.

A döntés meghozatalában a szakmai megfelelőség követelményének determinációja mellett (amelyet a szakdolgozat szoftvertervezési fázisában is kifejtettem) segítettek saját fejlesztői tapasztalataim, amelyek (természeteszerűleg) bizonyos technológiai eszközökhöz kötnek. Figyelembevéve a szoftver elkészítésének időfaktorát és a mielőbbi szoftverleszállítás hasznosságát, úgy döntöttem, hogy (a szakmai elvárásoknak kötelezően megfelelően) az ismert eszközt választom az ismeretlen helyett.

4.3 Nyelvi korlátok megszüntetése, lokalizálás

Mindezek mellett jelen vannak az elkészített szoftver nyelvi korlátai, amelyek az angol nyelv ismeretére kényszerítik a felhasználót. Felmerülhet az olvasóban (ahogy felmerült a szakdolgozat írójában is), hogy hasznos lenne a szoftvert lokalizálni a potenciális, angolul nem beszélő felhasználók részére.

Ezen fejlesztési igény követelményeinek meghatározása természetesen a megrendelőt terhelik, amennyiben a megrendelő is hasonló következtetésre jut lokalizálás témakörében.

5. Konklúzió

A szakdolgozatom alapját szolgáltató webalkalmazás fő célja (Lásd [1.6.4.1 Megrendelői elvárások](#)) a repetitív feladatok automatizálása annak érdekében, hogy:

- a megrendelő (vállalat vagy magánszemély) minimalizálni tudja a feladat elvégzéséhez szükséges erőforrásokat
- a programfelhasználó minél kevesebb idő alatt, a lehető legkisebb hibázási eséllyel végezhesse el a feladatát
- mindezt a lehető legalacsonyabb üzemeltetési költségen végrehajtva.

A [4. Vita](#) fejezetben felmerültek olyan, a szakdolgozat megvalósításával kapcsolatos felvetések, amelyekre érdemes és kell is reflektálni ebben a fejezetben:

- Magasabb szintű kódminőség és biztosabb működés (Lásd [4.1 Magasabb szintű kódminőség és biztosabb működés](#)):
 - o A hivatkozott szakmai felvetés alapján véleményem szerint az elkészített projektben nagyobb hangsúlyt is kaphatott volna a kódanalízis és a különböző tesztstratégiák részletesebb és magasabb színvonalú implementálása.
 - o A meg nem valósulás hátterében húzódó okként a megrendelői akarat hiányát tudnám megnevezni, márpedig ennek kell elsődleges rendezőelvként jelentkeznie egy professzionális munkavégzés vonatkozásában.
 - o A fenti két gondolatmenet alapján azt a konklúziót vonnám le, hogy a felvetés jogos szakmai felvetés és a megrendelő nyitottságát feltételezve ez indukálhat egy jövőbeli fejlesztést.
- Egyszerűbb technológiai portfólió (Lásd [4.2 Egyszerűbb technológiai portfólió](#)):
 - o A döntés meghozatalában a szakmai megfelelőség követelményének determinációja mellett (amelyet a szakdolgozat szoftvertervezési fázisában is kifejtettem) segítettek saját fejlesztői tapasztalataim, amelyek (természetesen) bizonyos technológiai eszközökhöz kötnek. Figyelembevée a szoftver elkészítésének időfaktorát és a mielőbbi szoftverleszállítás hasznosságát, úgy döntöttem, hogy (a szakmai elvárásoknak kötelezően megfelelően) az ismert eszközt választom az ismeretlen helyett.
 - o Konklúzióként tehát azt a megállapítást kell tegyem, hogy a hivatkozott fejezetben is ismertetett faktorok alapján a felhasznált technológiai portfólió kiválasztását tekintem továbbra is a legoptimálisabbnak.

- Nyelvi korlátok megszüntetése, lokalizálás (Lásd [4.3 Nyelvi korlátok megszüntetése, lokalizálás](#)):

- o A hivatkozott fejezetben is ismertetett gondolatmenet alapján azt a konklúziót vonnám le, hogy a felvetés jogos szakmai felvetés és a megrendelő nyitottságát feltételezve ez indukálhat egy jövőbeli fejlesztést.

Konklúzióként rögzítendő még, hogy a különböző fejezetekben leírt, költségekkel kapcsolatos megállapítások aggregálásával kijelenthető, hogy a szoftvert használata az általam biztosított technológiai környezettel díjmentesen igénybevehető.

Mindemellett annyit fontosnak tartok kiemelni még konklúzióként a tesztelési forgatókönyvben foglaltak sikeres végrehajtása (Lásd [3.3.10 Tesztelési összegzés](#)) és a megrendelői igények (Lásd [1.6.4.1 Megrendelői elvárások](#)) teljesítése tudatában, hogy a szakdolgozati témafelvetésre íródott szoftver sikeresen megoldást nyújt a felvetett problémára (Lásd [1.1 Probléma](#)).

6. Összegzés

Ezen Összegzés fejezetben a teljes dolgozathoz idézetként kiemelhető legértékesebb mondatokat fogom bemutatni, így summázva a szakdolgozat következtetéseit.

„A környezeti változó alatt azokra a szoftver futtatásához elengedhetetlen metaadatokra gondolok, amelyek meghatározzák és lehetővé teszik a szoftvertermék helyes működését. Ilyen környezeti változó például az adatbáziskapcsolathoz szükséges jelszó vagy az adatbázisszerver IP címe. (...)

Ezeknek a változóknak a számossága vállalati környezetben (például microservice architektúrával, szeparált környezetekkel) könnyen elérhetik a tízezres, vagy akár százezres nagyságrendet is.

A környezeti változók kezelésének komplexitása nagyban függ a kialakított kezelési stratégiától, amelyet a szoftverfejlesztő (vállalat, individuum) választott. Két fő oka lehet a magas kognitív komplexitásnak: az egyik a hibásan (tehát duplikációra és redundanciára hajlamos) kialakított környezeti változók kezelési stratégiájából, míg a másik az adott vállalati szoftvertermék növekedéséből (tehát sikerességéből, új fejlesztési igények létrejöttéből) fakad. Hozzáteszem, ez szintén egy hibásan kialakított kezelési stratégia következménye.” (Lásd [1.1 Probléma](#))

„A szakdolgozati munkám fő célja, hogy a felhasználó képes legyen az Azure DevOps felületén szükséges változómódosításokat minél egyszerűbben és minél nagyobb számosságban elvégezni.” (Lásd [1.2 Célok](#))

„A szakdolgozati projekt vállalati szintű felhasználásával ezen indikátorok hatásai is szignifikánsan csökkenthetővé válnak, hiszen a munkaterhelés és a repetitív feladatok is redukálhatóvá válnak a szoftver felhasználásával.” (Lásd [1.4 Hasznosság](#))

„A fenti alfejezetek eredményeit figyelembe véve kijelenthető, hogy a tesztelési forgatókönyvben foglaltak sikeresen végrehajtásra kerültek és a teszteredmények alapján a szoftver megfelelt a tesztforgatókönyvben foglaltaknak.” (Lásd [3.3.10 Tesztelési összesség](#))

„Konklúzióként rögzítendő még, hogy a különböző fejezetekben leírt, költségekkel kapcsolatos megállapítások aggregálásával kijelenthető, hogy a szoftvert használata az általam biztosított technológiai környezettel díjmentesen igénybevehető.” (Lásd [5. Konklúzió](#))

7. Jövőkép

„Minden sikeres projektben szerepelnie kell az átadott szolgáltatás későbbi visszajelzésekre alapuló folyamatos fejlesztésének.” (Györgydeák, 2021)

Több lehetőség és irányvonal is kínálkozik az alkalmazás jövőbeni fejlesztését illetően.

A legkézenfekvőbb és implementálásukban a legegyszerűbb jövőbeli fejlesztési feladatok a [4. Vita](#) és [5. Konklúzió](#) fejezetekben megfogalmazott állításokból következő fejlesztési igények lehetnek.

Mindemellett az adatperzisztenciában rejlő potenciális funkcionalitások is lehetnek a közeljövő fejlesztéseinek a fókuszában, mint például:

- Részletesebb módosítási előzményrekordok
- Részletesebb lekérdező felület a pontosabb naplóbejegyzés találatokért
- Műveletvisszaállítási funkcionalitás a naplóbejegyzések alapján

Van lehetőség újabb funkcionalitások implementálására a Microsoft Azure SDK mélyebb és széleskörűbb integrációjának és felhasználásának segítségével. A teljesség igénye nélkül a következő, az említett SDK által biztosított funkcionalitásokkal egészíthető ki a projekt:

- Agilis munkamódszertannal összefüggésben lévő feladatok automatizálása, folyamatok egyszerűsítése (például: Azure DevOps Boards elemek nagyszámú státuszváltoztatásai)
- Azure container registry-k lekérdezése és manipulálása

Továbbá van lehetőség a többi nagy felhőszolgáltató által biztosított API funkcionálisok feltárására is. Ezek a következők:

- Amazon Web services (AWS)
- Google Cloud Platform
- Alibaba
- IBM Cloud

A felsorolt felhőszolgáltatók közül az AWS és a Google Cloud Platform is biztosít elérhető API funkcionálisokat, amelyeknek a felhasználásával a termék platformfüggetlenné és széleskörűbben alkalmazhatóvá válna. Feltételezhető, hogy a termék profitabilitási és értékesíthetőségi mutatói jelentősen növekednének, amennyiben a platformfüggetlenítési irányba lesz továbbfejlesztve az alkalmazás.

8. MELLÉKLETEK

1. melléklet: [Irodalomjegyzek.pdf](#)
2. melléklet: [Abrajegyzek.pdf](#)
3. melléklet: [Tablázatjegyzek.pdf](#)
4. melléklet: [Roviditesek_jegyzeke.pdf](#)
5. melléklet: [Kivonat.pdf](#)
6. melléklet: [Dolgozat_angol_nyelvu_kivonata.pdf](#)
7. melléklet: [Munkaero_koltseg_alakulasa_az_it_iparagban.pdf](#)
8. melléklet: [Copilot_promptok.pdf](#)
9. melléklet: [ChatGPT_promptok.pdf](#)
10. melléklet: [Library_API_swagger.pdf](#)
11. melléklet: [API_swagger.pdf](#)
12. melléklet: [Tesztforгатokonyv.pdf](#)
13. melléklet: [Teszteredmenyek.pdf](#)
14. melléklet: [Kornyezeti_valtozo_modositasi_osszehasonlitas.pdf](#)

1. MELLÉKLET

Irodalomjegyzék

Adrian, M. (2015). *Using Docker DEVELOPING AND DEPLOYING SOFTWARE WITH CONTAINERS* (First). O'Reilly Media. <https://www.oreilly.com/library/view/using-docker/9781491915752/>

András Mizsur. (2023). Az Európai Unió zöld utat adott minden idők legnagyobb videójátékos üzletének. *Telex*. <https://telex.hu/techtud/2023/05/15/microsoft-europai-unio-activision-blizzard-videojatek-felvasarlas>

Andrew S. Tanenbaum. (2006). *Számítógép architektúrák* (Bohus Mihály, Trans.). Panem Könyvkiadó Kft. <https://www.antikvarium.hu/konyv/andrew-s-tanenbaum-szamitogep-architekturak-676209-0>

Apache Kafka Message Size Limit: Best Practices & Config Guide. (2025). Confluent. <https://www.confluent.io/learn/kafka-message-size-limit/>

Awati, R. (2023, November). *What is ICT (Information and Communications Technology)? | Definition from TechTarget*. <https://www.techtarget.com/searchcio/definition/ICT-information-and-communications-technology-or-technologies>

Bánné Varga, G. (2022). *Fogalmi adatmodellezés és relációs adatbázis-tervezés* (Kótai Kata, Ed.). Typotex Kiadó. https://www.typotex.hu/book/12733/banne_varga_gabriella_fogalmi_adatmodellezes_es_relacios_adatbazis-tervezes

Bluebird International. (2025). *BLUEBIRD IT SALARY GUIDE HUNGARY 2024*. <https://bluebird.hu/it-fizeteseik/>

Bobrovskis, S., & Jurenoks, A. (2018). *A Survey of Continuous Integration, Continuous Delivery and Continuous Deployment*. <https://ceur-ws.org/Vol-2218/paper31.pdf>

Bódy, B. (2019). *SQL példákon keresztül* (II). Jedlik Oktatási Stúdió. https://joskiado.hu/termek/az_sql_peldakon_keresztul_masodik_kiadas.html

- Brian, K. (2025, January 3). *The Top 8 Programming Languages for Web Development in 2025 - DEV Community*. <https://dev.to/bkthemes/the-top-8-programming-languages-for-web-development-in-2025-317n>
- Chacon, S., & Straub, B. (2025). *Pro Git Book* (Second). Apress. <https://git-scm.com/book/en/v2>
- Devinder Suthwal. (2020, November 30). *Medium - React Virtual DOM vs Real DOM*. <https://medium.com/devinder/react-virtual-dom-vs-real-dom-23749ff7adc9>.
<https://medium.com/devinder/react-virtual-dom-vs-real-dom-23749ff7adc9>
- Docker Docs*. (2025). <https://docs.docker.com/>
- Docker Inc. (2025, March 16). *Docker Compose | Docker Docs*.
<https://docs.docker.com/compose/>
- Douglas, B. (2025). *How to build a CI/CD pipeline with GitHub Actions in four simple steps - The GitHub Blog*. <https://github.blog/enterprise-software/ci-cd/build-ci-cd-pipeline-github-actions-four-steps/>
- dr. Dobos, I. (2016, November 28). *A szoftverfejlesztési szerződések jogi alapjai*.
<https://doboslegal.eu/a-szoftverfejlesztesi-szerzodesek-jogi-alapjai/>.
<https://doboslegal.eu/a-szoftverfejlesztesi-szerzodesek-jogi-alapjai/>
- Dr. Pitlik, L. (2025, March 13). *Informatikai erőforrások*. https://miau.my-x.hu/mediawiki/index.php/Informatikai_er%C5%91forr%C3%A1sok
- Dr. Rikk, J., & Dr. Pitlik, L. (2023). *ChatGPT-kísérletek - programozás*.
<https://m2.mtmt.hu/gui2/?mode=browse¶ms=publication;34071055>
- Elin Vinka. (2018, July 4). *What is Zookeeper and why is it needed for Apache Kafka? - CloudKafka, Apache Kafka*. <https://www.cloudkafka.com/blog/cloudkafka-what-is-zookeeper.html>
- Elte Informatikai Kar. (2025). *Bevezetés a kliens- és szerveroldali webalkalmazások készítésébe / Bevezetés (1. lecke)*.
https://webprogramozas.inf.elte.hu/tananyag/wf2/lecke1_lap1.html
- Elton Stoneman. (2020). *Learn Docker in a Month of Lunches*. Manning Publications Co.
<https://www.manning.com/books/learn-docker-in-a-month-of-lunches>

- Elton Stoneman. (2021). *Learn Kubernetes in a Month of Lunches*. Manning Publications Co.
<https://www.manning.com/books/learn-kubernetes-in-a-month-of-lunches>
- European Parliament. (2019, May 14). *How the EU improves workers' rights and working conditions* | Topics | European Parliament.
<https://www.europarl.europa.eu/topics/en/article/20190506STO44344/how-the-eu-improves-workers-rights-and-working-conditions>
- Git Kraken. (2022, June 17). *What is Git Flow*. <https://www.gitkraken.com/learn/git/git-flow>
- Gomes, I., Morgado, P., Gomes, T., & Moreira, R. (2009). *An overview on the Static Code Analysis approach in Software Development*.
- Györgydeák, B. P. (2021). *A Szoftverfejlesztés Gazdaságtana* [Kodolányi János Egyetem].
https://miau.my-x.hu/miau/276/kje/szd1/a_szoftverfejleszttes_gazdasagtana.pdf
- Harrison, C. (2021, September 28). *How "DevOps Engineer" became the most in-demand job title*. <https://www.harrisonclarke.com/blog/how-devops-engineer-became-the-most-in-demand-job-title>
- Horváth, L. (2007). *Szoftvertesztelés a gyakorlatban* [Eötvös Loránd Tudományegyetem].
<https://people.inf.elte.hu/holaci/Diploma/Szoftvertesztel%20a%20gyakorlatban.pdf>
- IBM Foundation. (2025, March 15). *What Is Event Streaming?*
<https://www.ibm.com/think/topics/event-streaming>
- Imre, P., & Ármin, P. (2014). *Az informatika jövője a felhőben van*. Budapesti Műszaki Egyetem. https://www.vmtt.org.rs/mtn2010/492_502_Petkovics_A.pdf
- International Business Machines Corporation. (2021, April 14). *Alkalmazásszerverek - IBM dokumentáció*. <https://www.ibm.com/docs/hu/i/7.2?topic=serving-application-servers>
- Intro to GitHub - GitHub Education*. (2025).
https://education.github.com/experiences/intro_to_github
- Lander, R. (2023, February 21). *Secure your .NET cloud apps with rootless Linux Containers - .NET Blog*. <https://devblogs.microsoft.com/dotnet/securing-containers-with-rootless/>
- LinkedIn Inc. (2025, February 24). *Apache Kafka*. <https://kafka.apache.org/>

- Martin, R. C. (2018). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. In *Robert C. Martin series*. Prentice Hall.
<https://www.oreilly.com/library/view/clean-architecture-a/9780134494272/>
- Microsoft Inc. (2025a). *NuGet documentation* | *Microsoft Learn*.
<https://learn.microsoft.com/hu-hu/nuget/>
- Microsoft Inc. (2025b, February 24). *Microservices architecture design - Azure Architecture Center* | *Microsoft Learn*. <https://Learn.Microsoft.Com/En-Us/Azure/Architecture/Microservices/>. <https://learn.microsoft.com/en-us/azure/architecture/microservices/>
- Mirantis Inc. (2025, February 24). *Docker official documentation*.
<https://www.docker.com/>
- Potdar, A. M., Narayan, D. G., Kengond, S., & Mulla, M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. *Procedia Computer Science*, 171, 1419–1428.
<https://doi.org/10.1016/j.procs.2020.04.152>
- Provectus. (2025). *provectus/kafka-ui: Open-Source Web UI for Apache Kafka Management*.
<https://github.com/provectus/kafka-ui>
- React Community (Meta). (2025, February 24). *React official documentation*.
<https://react.dev/>
- Rubóczki Edit Szilvia. (2019). *A felhasználók biztonságtudatosságának jelentősége a publikus felhőszolgáltatások nagyvállalatoknál történő bevezetésekor* [Óbudai Egyetem]. https://lib.uni-obuda.hu/sites/lib.uni-obuda.hu/files/Ruboczki_Edit_Szilvia_tezisfuzet.pdf
- SonarSource Inc. (2025). *SonarQube Cloud Documentation*.
<https://docs.sonarsource.com/sonarqube-cloud/>
- Suraj Surve. (2021, February 18). *FreeCodeCamp - Why you should use React.js for web development*. <https://www.freecodecamp.org/news/why-use-react-for-web-development>
- Trung Tran. (2024, May 27). *Top Server-side Scripting Languages & Frameworks - Orient Software*. <https://www.orientsoftware.com/blog/server-side-scripting-languages/>

- Tulili, T. R., Capiluppi, A., & Rastogi, A. (2023). Burnout in software engineering: A systematic mapping study. In *Information and Software Technology* (Vol. 155). Elsevier. <https://doi.org/10.1016/J.INFSOF.2022.107116>
- Zolkifli, N. N., Ngah, A., & Deraman, A. (2018). Version Control System: A Review. *Procedia Computer Science*, 135, 408–415. <https://doi.org/10.1016/j.procs.2018.08.191>

2. MELLÉKLET

Ábrajegyzék

1. ábra: Szimulált hasznossági példa végeredményének diagrambéli reprezentációja (saját szerkesztés).....	3
2. ábra: A három fő felhőszolgáltató vállalat piaci megoszlása (forrás: https://www.cloudzero.com/blog/cloud-service-providers)	12
3. ábra: Az elmúlt 4 év informatikai bértrendjei és a magyar átlagbérek a BlueBird Salary Guide kiadványok (forrás: https://bluebird.hu/it-fizeteseik/) és a KSH jelentései alapján (forrás: https://www.ksh.hu/keresetek) (saját szerkesztés).....	15
4. ábra: Kliens-szerver architektúra (forrás: https://hyperskill.org/learn/step/25834)	16
5. ábra: Szakirodalmi hivatkozásban említett 1-1. ábra (forrás: (Adrian, 2015))	20
6. ábra: Szakirodalmi hivatkozásban említett 1-2. ábra (forrás: (Adrian, 2015))	21
7. ábra: Docker konténer létrehozásának egyszerűsített folyamatábrája (forrás: https://linuxiac.com/what-is-docker-container/)	22
8. ábra: CVCS verziókezelő rendszer (forrás: https://www.sciencedirect.com/science/article/pii/S1877050918314819).....	26
9. ábra: DVCS verziókezelő rendszer (forrás: https://www.sciencedirect.com/science/article/pii/S1877050918314819).....	27
10. ábra: Adatbázistípusok (forrás: https://www.devopsschool.com/blog/types-of-databases/)	29
11. ábra: Szoftverarchitektúra (saját szerkesztés)	41
12. ábra: Variable group utolsó módosító (saját szerkesztés).....	45
13. ábra: Secret bejelentkező felület (saját szerkesztés).....	51
14. ábra: Környezeti változó bejelentkezési felület (saját szerkesztés).....	52
15. ábra: Környezeti változók lekérdezése, adatlap (saját szerkesztés)	53
16. ábra: Környezeti változók lekérdezése, eredmény (saját szerkesztés)	55
17. ábra: Környezeti változó hozzáadás, adatlap (saját szerkesztés)	56
18. ábra: Környezeti változó hozzáadás, megerősítő felület (saját szerkesztés)	57
19. ábra: Környezeti változó hozzáadás, sikeres művelet képernyő (saját szerkesztés)	58
20. ábra: Környezeti változó módosítása, adatlap (saját szerkesztés)	59
21. ábra: Környezeti változó törlése, adatlap (saját szerkesztés)	60

22. ábra: Secret lekérdezés, adatlap (saját szerkesztés).....	61
23. ábra: Secret lekérdezés eredményfelülete (saját szerkesztés)	62
24. ábra: Secret törlése, adatlap (saját szerkesztés).....	63
25. ábra: Secret törlés megerősítő felület (saját szerkesztés)	64
26. ábra: Key vault másolása, adatlap (saját szerkesztés)	65
27. ábra: Key vault átmásolás sikerképernyő (saját szerkesztés).....	66
28. ábra: Szoftverben elvégzett módosítások megtekintése (saját szerkesztés).....	66
29. ábra: Korábbi környezeti változó módosítások lekérdezése (saját szerkesztés)	67
30. ábra: Korábbi secret módosítások lekérdezése (saját szerkesztés).....	68
31. ábra: Korábbi key vault másolások lekérdezése (saját szerkesztés).....	69
32. ábra: Build pipeline futtatásának első adatlapja, pipeline lekérések (saját szerkesztés)	70
33. ábra: Build pipeline lekérdezés eredménytáblája, pipeline futtatás adatlapja (saját szerkesztés).....	71
34. ábra: Sikeres végrehajtás, alapértelmezett build pipeline futtató eredménytábla adatlap (saját szerkesztés).....	72
35. ábra: Azure DevOps felületén elindított pipeline (saját szerkesztés)	72
36. ábra: Utolsó stabil szoftververzió lekérdezése (saját szerkesztés)	73
37. ábra: Utolsó stabil szoftververzió lekérdezésének eredménye (saját szerkesztés).....	73
38. ábra: Azure DevOps felület - stabil verziószámok hibás rendezése (saját szerkesztés).....	74
39. ábra: Repository lekérdezés adatlapja az új pull request létrehozásához (saját szerkesztés)	75
40. ábra: Repository lekérdezésének eredménye a pull request létrehozásához és a létrehozás adatlapja (saját szerkesztés)	76
41. ábra: Pull request létrehozása utáni sikerképernyő (saját szerkesztés)	77
42. ábra: Pull request lekérdező képernyő eredménytáblája (saját szerkesztés)	78
43. ábra: Saját készítésű HttpClientProvider felelős az Azure kapcsolatért (saját szerkesztés)	81
44. ábra: Kafka UI kezelőfelület, brókerek (saját szerkesztés)	83
45. ábra: Kafka UI kezelőfelület, topikok (saját szerkesztés)	83
46. ábra: Kafka UI kezelőfelület, fogyasztók (saját szerkesztés).....	84
47. ábra: Kafka UI kezelőfelület, üzenetek (saját szerkesztés)	85
48. ábra: Adatbázis és kapcsolódó adattáblák pgAdmin adatbáziskliens használatával (saját szerkesztés).....	87

49. ábra: Adatbázis, secret módosítások lekérdezése pgAdmin adatbáziskliens használatával (saját szerkesztés).....	88
50. ábra: Adatbázis, kapcsolat létrehozása (saját szerkesztés)	89
51. ábra: Forráskód integrációt biztosító pipeline, szerveroldal (saját szerkesztés)	91
52. ábra: Üzemeltetési és kitelepítési integrációt biztosító pipeline, szerveroldal (saját szerkesztés).....	92
53. ábra: Statikus kódanalízist biztosító pipeline, szerveroldal (saját szerkesztés)	93
54. ábra: Statikus kódelemzés analíziseket összesítő kezelőfelület (saját szerkesztés)	94
55. ábra: Szerveroldali komponens Dockerfile tartalom (saját szerkesztés).....	95
56. ábra: Webkliens komponens Dockerfile tartalom (saját szerkesztés)	97
57. ábra: Szakdolgozati projekt komponenseinek futtatott konténerei (saját szerkesztés)	98
58. ábra: Elérhető webkliens és szoftver a szerver (localhost, saját gép) 3000-es portján (saját szerkesztés).....	99
59. ábra: Webkliens Kubernetes manifest Deployment-je (saját szerkesztés)	101
60. ábra: Lokálisan futtatott Kubernetes Cluster a szakdolgozati szoftver vonatkozásában (saját szerkesztés).....	102
61. ábra: Az elmúlt 4 év informatikai bértrendjei és a magyar átlagbérek a BlueBird Salary Guide kiadványok (https://bluebird.hu/it-fizetese/) és a KSH jelentései alapján (https://www.ksh.hu/keresetek) (saját szerkesztés)	140
62. ábra: Library API - módosítások és azure projekt lekérdezéssel kapcsolatos végpontok (saját szerkesztés).....	150
63. ábra: Library API - Secret műveletekre vonatkozó végpontok (saját szerkesztés)	150
64. ábra: Library API - Környezeti változó műveletekre vonatkozó végpontok (saját szerkesztés).....	150
65. ábra: Build pipeline, git fájl és git pull request lekérdezéssel kapcsolatos végpontok (saját szerkesztés).....	152
66. ábra: Git repository, git verzió, profil és release pipeline vonatkozású végpontok (saját fejlesztés).....	153
67. ábra: Hitelesítési adatok maszkolt formátumban (saját szerkesztés)	154
68. ábra: Keresett környezeti változót tartalmazó variable group Azure DevOps felületen (saját szerkesztés).....	155
69. ábra: ZoomAdapter PROD variable group jelenlegi értéklístája (saját szerkesztés)	157
70. ábra: Lekérdezés tárgyát képező key vault jelenlegi secret listája (saját szerkesztés)	159
71. ábra: KeyVaultBV95 key vault jelenlegi állapota Azure Portalon (saját szerkesztés)	161

72. ábra: KeyVault-BKV key vault jelenlegi állapota Azure Portalon (saját szerkesztés).....	162
73. ábra: Környezeti változó hozzáadási műveleteket naplózó adattábla perzisztált eredményei (saját szerkesztés).....	163
74. ábra: Secret módosítások naplóbejegyzései (saját szerkesztés)	164
75. ábra: Key vault másolások naplóbejegyzései (saját szerkesztés).....	165
76. ábra: Automatikusan létrehozott git repository, nincsenek tagek (saját szerkesztés	166
77. ábra: Általam létrehozott Git repository és az ahhoz tartozó stabil verziók (saját szerkesztés).....	166
78. ábra: Azure DevOps-ban található build pipeline-ok (saját szerkesztés).....	167
79. ábra: Sikeres bejelentkezés után üdvözlő képernyő (saját szerkesztés).....	169
80. ábra: Sikertelen bejelentkezés (saját szerkesztés)	170
81. ábra: Környezeti változó sikeres lekérdezés, teszteredmény (saját szerkesztés)	171
82. ábra: Sikertelen környezeti változó lekérdezés (saját szerkesztés)	172
83. ábra: Sikeres környezeti változó hozzáadás megjelenik a szoftver kezelőfelületén (saját szerkesztés).....	173
84. ábra: Sikeres környezeti változó hozzáadás megjelenik az Azure DevOps felületén is (saját szerkesztés).....	173
85. ábra: Sikertelen környezeti változó hozzáadás (saját szerkesztés).....	174
86. ábra: Sikeres környezeti változó módosítás (saját szerkesztés)	175
87. ábra: Sikeres környezeti változó módosítás megjelenik az Azure DevOps felületén is (saját szerkesztés).....	175
88. ábra: Sikertelen környezeti változó módosítás, nincs a paramétereknek megfelelő környezeti változóra találat (saját szerkesztés)	176
89. ábra: Sikeres környezeti változó törlés (saját szerkesztés).....	177
90. ábra: Sikeres környezeti változó törlés után nem található az Azure DevOps-ban a SearchTestScenarioKey2 környezeti változó (saját szerkesztés)	177
91. ábra: Sikertelen környezeti változó törlés, nincs a paramétereknek megfelelő környezeti változóra találat (saját szerkesztés)	178
92. ábra: Sikeres secret lekérdezés, eredmény az eredménytáblában megjelenítve (saját szerkesztés).....	179
93. ábra: Sikertelen secret lekérdezés (saját szerkesztés)	180
94. ábra: Secret törlés utáni Azure Portal állapot (saját szerkesztés)	180
95. ábra: Sikertelen secret törlés (saját szerkesztés).....	181
96. ábra: Sikeres reaktiválást követő key vault állapot (saját szerkesztés)	182

97. ábra: Sikertelen reaktíválási művelet (saját szerkesztés)	182
98. ábra: Sikeres key vault másolás sikerképernyő (saját szerkesztés).....	183
99. ábra: Sikeres key vault másolás utáni Keyvault-BKV key vault állapota (saját szerkesztés)	183
100. ábra: Sikertelen key vault másolás eredménye az Azure Portalon (saját szerkesztés)...	184
101. ábra: Környezeti változó módosítások naplóbejegyzései, első oldal (saját szerkesztés)	185
102. ábra: Környezeti változó módosítások naplóbejegyzései, második oldal (saját szerkesztés).....	185
103. ábra: Secret módosítások naplóbejegyzései, első oldal (saját szerkesztés).....	186
104. ábra: Secret módosítások naplóbejegyzései, második oldal (saját szerkesztés).....	186
105. ábra: Key vault másolások naplóbejegyzései, első oldal (saját szerkesztés)	187
106. ábra: Key vault másolások naplóbejegyzései, második oldal (saját szerkesztés)	187
107. ábra: Sikeres utolsó stabil verziók lekérdezése (saját szerkesztés)	188
108. ábra: Sikeres pipeline futtatás a webkliensen (saját szerkesztés).....	189
109. ábra: Sikeres pipeline futtatás megjelenik az Azure Devops felületén (saját szerkesztés)	189
110. ábra: Sikeres pull request létrehozás webkliensen (saját szerkesztés)	190
111. ábra: Sikeresen létrehozott pull request megjelenik az Azure DevOps felületén (saját szerkesztés).....	190
112. ábra: Sikertelen pull request létrehozás, nem léteznek branch-ek a repository-hoz (saját szerkesztés).....	191
113. ábra: Táblázatba rendezett pull request lista (saját szerkesztés).....	192
114. ábra: Azure Devops-on található pull request lista (saját szerkesztés).....	192
115. ábra: variable group kikeresése Library menüpontban, szűrve a variable group nevére	194
116. ábra: környezeti változó értékének módosítása, majd mentése.....	194
117. ábra: bejelentkezés a Sozure felületére personal access token és azure organization név megadással	195
118. ábra: kiválasztjuk a megfelelő űrlapot a legördülő menük segítségével, majd ki is töltjük	196
119. ábra: Szükség esetén módosítunk, majd jóváhagyjuk/elutasítjuk az elvégezni kívánt műveletet	196

3. MELLÉKLET

Táblázatjegyzék

1. táblázat: Felhasznált technológiák (saját szerkesztés)

4. MELLÉKLET

Rövidítések jegyzéke

- YAML: YAML Ain't Markup Language / Yet Another Markup Language (egy újabb jelölési nyelv, kubernetes manifestek írására is használják)
- SDK: Software Development Kit (elérhető és felhasználható szoftvercsomag)
- CLI: Command Line Interface (parancssori interfész)
- CI: Continuous Integration (folyamatos integráció)
- CD: Continuous Deployment(folyamatos telepítés)
- DevOps: Development and Operations (Fejlesztés és Üzemeltetés)
- API: Application Programming Interface (Alkalmazásprogramozási interfész)
- SQL: Structured Query Language (Strukturált lekérdező nyelv)
- AWS: Amazon Web Services (Amazon webszolgáltatások)
- HTML: HyperText Markup Language (Hiperszöveg jelölőnyelv)
- URL: Uniform Resource Locator (Egységes erőforrás helymeghatározó)
- HTTP: HyperText Transfer Protocol (Hiperszöveg átviteli protokoll)
- DVCS: Distributed Version Control System (Elosztott verziókezelő rendszer)
- CVCS: Centralized Version Control System (Centralizált verziókezelő rendszer)
- SRP: Single Responsibility Principle (Egyetlen felelősség elve)
- LTS: Long Time Support (Hosszútávú támogatás)
- bkV: Automatikusan létrehozott Azure projekt a saját Azure organizációmban (Bócz-Kemény Viktor névre utalhat)
- GUID: Globally Unique Identifier (Globálisan egyedi azonosít)
- MSIL: Microsoft Intermediate Language (Microsoft köztes nyelv)
- JDK: Java Development Kit (Java fejlesztési csomag)

5. MELLÉKLET

KIVONAT

A célom egy olyan szoftver létrehozása és bemutatása, amely Microsoft Azure környezetben képes bizonyos repetitív feladatok automatizálására és elvégzésük felgyorsítására egyszerűsített utasítások megadásával.

Célcsoportok:

1. A szoftver célcsoportja: Azok a Microsoft Azure-t használó fejlesztők, rendszeradminisztrátorok, karbantartók, vagy DevOps mérnökök, akik rendszeresen használják az Azure DevOps felületét.
2. A szakdolgozat célcsoportja: Folyamatoptimalizáló szakemberek, akik főként Microsoft Azure ökoszisztémával foglalkoznak és érdeklődnek a repetitív és adminisztratív természetű feladatok automatizálása iránt.

Az elkészült szoftver szerveroldali komponensei .NET keretrendszerrel készültek (.NET 8.0 verzió), a komponensek közötti kommunikációt az Apache Kafka nevű elosztott eseményfolyam-kezelő platform valósítja meg. Az adatbáziskezelést PostgreSQL nyílt forráskódú, objektum-relációs adatbázis-kezelő rendszer használatával valósítottam meg.

Az Apache Kafka kommunikáció használata, illetve a kódduplikáció elkerülése érdekében könyvtárlétrehozásra és megosztott könyvtárhasználatra is szükség volt a szerveroldali komponensek esetén. Ehhez a NuGet .NET alapú csomagkezelő rendszert vettem igénybe.

A projekt kliensoldalát React nyílt forráskódú Javascript könyvtár használatával készítettem el.

A forráskód verziókezeléséhez a Git verziókezelő rendszert vettem igénybe, a forráskód tárolására és a potenciális kollaborálás lehetőségéhez a GitHub webplatformot választottam. Ezen platformon képes voltam az előre definiált build pipeline yaml fájlok használatával kialakítani a saját CI/CD stratégiámat is, ami a projekt folyamatos integrációját hivatott megvalósítani.

Az elkészült szoftver konténerizáltan futtatható a Docker konténerizációs technológia használatával. A megírt kubernetes manifestek segítségével futtatható a produktum konténer

alapú alkalmazáskezelő szoftver segítségével is, ami a szakdolgozati munka esetében Kubernetes-t jelent.

Statikus kódelemzéshez a SonarCloud platformot használtam. Ennek köszönhetően a megírt forráskód minősége a statikus kódelemzés kritériumai szerint biztosítottá váltak, hiszen a kiválasztott elemzési beállítások segítségével a platform valós időben tájékoztatott a változtatásokkal járó riportmódosulásokról is.

Kiemelt szempont volt, hogy a szoftver költségeit minimális szinten tartsam. A fent részletezett szoftverfejlesztési eszközök és stratégiák segítségével a produktum a felhasznált technológiák tekintetében teljesen ingyenes tudott maradni. Ennek okán a terméket magasabb profit margin értékkel lehet értékesíteni.

6. MELLÉKLET

A dolgozat angol nyelvű kivonata

English extract of the thesis

ABSTRACT

My goal is to create and demonstrate a software that can automate and accelerate certain repetitive tasks in a Microsoft Azure environment by providing simplified instructions.

Target groups:

1. Target audience for the software: Developers, system administrators, maintainers, or DevOps engineers using Microsoft Azure who regularly use the Azure DevOps interface.
2. Target audience of the thesis: Process optimization professionals who are mainly working with Microsoft Azure ecosystem and are interested in automating tasks of repetitive and administrative nature.

The server-side components of the developed software are built using the .NET framework (.NET version 8.0), the communication between the components is implemented by a distributed event flow management platform called Apache Kafka. The database management was implemented using PostgreSQL, an open source, object-relational database management system.

In order to use Apache Kafka communication and avoid code duplication, library creation and shared directory usage was required for the server-side components. For this I used the NuGet .NET based package management system.

The client side of the project was built using React open source Javascript library.

For version control of the source code I used the Git version control system, and for source code storage and potential collaboration I used the GitHub web platform. On this platform, I was also able to develop my own CI/CD strategy using predefined build pipeline yaml files, which was designed to achieve continuous integration of the project.

The resulting software can be run in containers using Docker containerization technology. The written Kubernetes manifests can also be used to run the product using container-based application management software, which in the case of this thesis is Kubernetes.

For static code analysis, I used the SonarCloud platform. This ensured the quality of the written source code according to the static code analysis criteria, as the platform also informed me in real time about the report modifications due to the selected analysis settings.

A key consideration was to keep the cost of the software to a minimum. The software development tools and strategies detailed above allowed the product to remain completely free of charge in terms of the technologies used. As a result, the product can be sold with a higher profit margin.

Introduction

Problem

My thesis is a solution to the complexity of managing environment variables for software deployment in a Microsoft Azure environment.

By environment variables, I mean the metadata that is essential for the software to run and that defines and enables the software product to function correctly. Examples of such environment variables are the password required for database connection or the IP address of the database server. The value of these variables can be passed via Azure DevOps (Development and Operations) either to a Kubernetes manifest YAML (YAML Ain't Markup Language / Yet Another Markup Language) file or as a parameter of a CLI (Command Line Interface) based command, the nature of the usage is determined by the software manager.

The number of these variables in an enterprise environment (e.g. microservice architecture, segregated environments) can easily reach tens of thousands or even hundreds of thousands.

The complexity of managing environment variables is highly dependent on the management strategy chosen by the software developer (company, individual). There are two main reasons for high cognitive complexity: one is due to the incorrectly (i.e. duplication and redundancy prone) designed management strategy for environmental variables, and the other is due to the growth (i.e. success, new development needs) of the enterprise software product. I would add that this is also a consequence of a flawed management strategy.

Nevertheless, certain metrics (speed of modification, number of operations required to modify) indicate that modifying environment variables in the Azure DevOps interface is more difficult than would be possible in a software solution to this problem. This will be discussed in more detail in the following chapters.

Goal

The main goal of my thesis is to enable the user to make the necessary changes to the Azure DevOps interface as simple and as numerous as possible. However, the Azure SDK (Software Development Kit) used also offers additional functionality beyond manipulating and querying environment variables, which can be used to further automate the tasks involved in deployment and software development. This allows the owner of the software product to optimise costs and the users of the developed tool to perform their tasks faster, easier, with greater security and less chance of error. The nature of potential users is discussed in the next chapter.

Target audiences

"As with many other technology jobs, the definition of a DevOps engineering job is subjective, and its meaning, required skills and core responsibilities are likely to vary from company to company. The actual DevOps engineering role will depend on the goals, growth stage, industry, and advances in computing of the organization." (Harrison, 2021)

As discussed in this technical article (Harrison, 2021), software operations and software maintenance responsibilities may be distributed differently across job functions in an IT company. As a consequence, I will target those jobs whose holders have the theoretical possibility to have the aforementioned responsibilities (operations and maintenance) and use the Azure DevOps platform for their work in a Microsoft cloud ecosystem. Examples include:

- Software developers
- System Administrators
- System Administrators
- DevOps engineers

So, based on the above, it can be stated that the software developed based on the thesis problem statement can help IT professionals who use cloud services in Microsoft ecosystem and are users of Azure DevOps platform.

Usefulness

The usefulness of the thesis is supported by Annex 14 attached to the thesis.

In this annex, I have simulated a sixty-seven-fold difference in time between the manual operation and the operation performed by the thesis project. I performed the required modification on a variable group, and the measured time to complete the operation was 3502 seconds for manual modifications, while the automated modifications took 52.46 seconds (See Annex 14 and Figure 1).



As can be seen from the salary statement prepared by the IT staffing company Bluebird, "a beginner system administrator or DevOps engineer could earn between 650 and 800 thousand HUF gross in Hungary on average in 2024" (Bluebird International, 2025, Annex 7). From these gross figures it can be seen that even the automated performance of the operation mentioned in the example saves the company buying the software thousands of HUF.

If we assume that several daily modifications of environmental variables similar to the one mentioned in the example have to be performed, the payback of the software can be easily demonstrated by the above example.

As stated in a study by the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, "*repetitive tasks and workload are among the main causes of job burnout in IT*" (Tulili et al., 2023). By using the thesis project at the enterprise level, the impact of these indicators can be significantly reduced, as both workload and repetitive tasks can be reduced by using the software.

Motivation

I'm currently working as a server-side developer in Microsoft ecosystem and as an intern I had to deal with a lot of environment variables. I was tasked by my superiors to find some kind of automation solution to modify environment variables in order to reduce the workload of our interns.

For this reason, I created a small software used within the team, which can manipulate and query the environment variables on a CLI basis. However, I felt that there was greater potential in this area, so I wrote my own software offering a wider range of solutions, which I intend to present in this thesis.

Structure of thesis

The structure of the thesis is presented in this chapter.

Following the main chapter 1 Introduction, I will present the literature background of the thesis topic in the main chapter 2 Literature. In addition, I will specifically examine whether the chosen topic and its implementation can, and if so how it can, be related to the subjects in the model curriculum (see 2.9 Relationship of subjects to the thesis).

In section 3, Presentation of my own development, I will describe the design and development process of the software to be developed to solve the problem and present the finished product itself.

I will present the completed project along the following chapters: software design (See 3.1 Designing the software), software implementation (See 3.2 Implementing the software) and software testing (See 3.3 Testing).

My conclusions and critical comments on my own work (demonstrating a willingness to self-reflect) are described in Discussion 4 and Conclusion 5.

In Chapter 6, Conclusion, I will summarise the most valuable findings and conclusions for the chosen topic and the work I have done.

In Chapter 7, Vision for the Future, I will outline the options for improvement that could become a reality in the foreseeable future, if there is a willingness to do so on the part of the client.

Software design

In chapter 3.1 Software design, I describe the technologies, libraries and architectural approaches I used to build the application and the platform I used to build it. In this segment, I will also explain the rationale behind my decision to use the software tools and approaches listed.

In this chapter I will also discuss the chosen Continuous Integration/Continuous Deployment (CI/CD) strategy and the components of this strategy.

Software implementation

In chapter 3.2 Software implementation, I present and interpret the documentation produced during the development of the software. In this chapter I will illustrate with diagrams and screenshots the different elements of the completed work, the different functionalities and the stages of the development work.

Furthermore, I will show how the chosen CI/CD tools represent the real added value of the implemented software.

Software testing

In chapter 3.3 Testing, I describe the technologies and methodologies I used to test the software. This includes the static code analysis technologies, the testing libraries provided by the framework used and the test steps provided by CI/CD.

Manual testing of the software is also possible, so I will present a completed test scenario and its implementation.

Detailed structure of the thesis

In this subsection, I will describe the content and form of the thesis, and the exact expectations the client had of the software on which the thesis is based.

By examining this sub-chapter and the main chapter 3, Presentation of my own development, the client and the reader of the thesis will be able to decide whether the work produced meets the requirements and, if so, to what extent.

Client expectations

I will summarise the client's expectations in this sub-chapter and refer to this summary in later chapters.

The main objectives of the customer of the product:

- The user should be able to make the required variable changes in the Azure DevOps interface as easily and as numerous as possible
- an additional objective is to deliver other development needs as defined in the customer's expectations segment, which I will use to achieve the main objective (freeing up manual resources through automation) even more effectively

The customer of the product expects the following:

- Writing closed source code for the software, using a version manager
- Generation and delivery of compiled source code files (preferably with a containerized solution) that can be run on any operating system
- Software should have a web user interface using a modern framework (JavaScript framework preferred)
- The software must display the web interface in English only (therefore I will present English pages in the main chapter on the presentation of my work)
- The software should implement a microservice architecture (.NET ecosystem is preferred)
- IT security considerations
- Test scenario and test results to be presented and handed over as a requirement for adoption of the development
- Functional requirements:
 - Automation of administrative tasks experienced in Azure environment and minimization of resources used in this context
 - Manipulation and querying of environment variables stored in Azure DevOps variable groups with significant volume of involvement using simple statements
 - Query, delete and reactivate (recover) secrets stored in Azure DevOps
 - Copy the entire contents of a key vault created in Azure DevOps to another key vault (fast data upload)
 - Logging changes made in Azure DevOps to environment variables, secrets and key vault copies
 - Tabular display of the following change-related data:
 - Environment variable when modified:
 - Azure organization

- Azure project
- User
- Variable group regex command
- Environment variable name
- Change type
- Date
- Key for vault copy:
 - Data source Key vault
 - Target Key vault
 - User
 - Date
- Azure Secret in case of modification:
 - Key vault name
 - User
 - Secret name regex command
 - Modification type
 - Date
- Query pull requests created in Azure DevOps
- Create a simplified Azure DevOps pull request
- Simplified execution of Azure DevOps build pipelines
- Simplified query of software versions created in Azure DevOps (query last version in significant volume)
- The built software should aim to use the following tools where possible:
 - a broad developer community
 - open source
 - freely available

From the above list of customer expectations, it can be inferred that the following were not expectations of the software:

- Software operation in dedicated environment(s)
- Container management
- Code quality reporting, code analysis

Research constraints, or elements deliberately omitted from the thesis

In this subsection, I list the aspects of the thesis that I do not intend to cover.

The thesis does not have a topic (basically for reasons of length):

- A detailed introduction to Azure DevOps and an in-depth description of its functionality (only as much as is strictly necessary to understand the thesis)
- a detailed presentation and in-depth description of the functionality of other cloud providers available on the market

Deliberately omitted from the thesis, so not expected of the thesis:

- Extend, expand and enhance the IT security services provided by Azure (without compromising the existing IT security level, of course, requiring the use of built-in security measures, e.g.: personal access token and other credentials)

In light of this, I will only describe the Microsoft Azure functionality to the extent strictly necessary, I will not seek to explain why, for example, the functionality of environment variables does not require a client id.

Formal solutions in the thesis

The thesis follows certain formal patterns in order to provide a more thorough understanding of the thesis topic and the software written. These include the following:

- Clearly delimited and structured chapters:
 - o Introduction main chapter written in a structured and attention-grabbing way, maintaining the reader's curiosity about the thesis.
 - o Sharply separating the literature background from the rest of the thesis, with links found at most in the chapter references (thus demonstrating the necessary literature link to the work done)
 - o Sharply separating and detailing the development work carried out on the basis of the thesis topic and literature background. Reflection on the literature background is an essential part of the presentation of the work, so relevant chapter references have been included.
 - o Conclusions (with discussion and summary segments) and potential future scenarios after the presentation of own work
- Detailed and frequent visual representations:
 - o Figures
 - o Tables

- Detailed test scenario, demonstrating the correct functioning of the software
- The paper is in Hungarian
- The essay includes hypertext stepping stones to support the Digital Reader
- Citations are in italics and in quotation marks with immediate acknowledgement of the source
- The essay is free of additional formatting (bold, underlining, footnotes)
- Appendices are included to assist the reader in fully understanding the essay

Discussion

In this concluding chapter, I will summarise my self-reflective insights on the thesis, providing an opportunity for professional discussion.

Higher code quality and more secure operation

In my opinion, a higher level of code quality would have resulted if more verbose error messages could have been provided in the interface, thus enhancing the intuitive interface use by the user. The above-mentioned code quality improvement would have been reinforced if processes could have been eliminated where either no feedback is given in case of an error during the use of the tool, or worse, a misleading error message is given to the user. Examples of these are easy to find when presenting your own work.

During the implementation and testing of the software, it was found that implementing other alternative process structures and web interface design guidelines could further increase the resource savings associated with task completion. In other words, performing a given task would require less time and/or computing power.

Although there was no direct expectation from the client to ensure code quality based on analysis and written tests, this does not detract from the fact that the use of analysis and the practice of writing tests contribute to a higher level of code quality and more stable operation of a software.

Simpler technology portfolio

The technology portfolio of the thesis software is the result of decisions taken on the basis of the circumstances and research results presented in the thesis. The question arises: could the software designer have reached a different conclusion given the same input parameters? In my opinion, he could have.

It is natural that a problem can be solved by several tools, such as the other competing frameworks discussed in this thesis.

In making this decision, in addition to the determination of the technical adequacy requirement (which I also explained in the software design phase of the thesis), I was helped by my own development experience, which (naturally) binds me to certain technological tools. Taking into account the time factor of the software development and the usefulness of developing the software as soon as possible, I decided to choose the known tool (as a professional requirement) instead of the unknown one.

Language barrier removal, localisation

In addition to all this, there are language limitations of the software developed, which force the user to know English. It may occur to the reader (as it did to the author of this thesis) that it would be useful to localise the software for potential non-English speaking users.

It is of course up to the client to determine the requirements of this development need, should the client come to a similar conclusion on localisation.

Conclusion

The main objective of the web application that is the basis of my thesis (see 1.6.4.1 Customer requirements) is to automate repetitive tasks in order to:

- the client (company or individual) can minimize the resources needed to complete the task
- the program user can perform the task in the least possible time and with the lowest possible chance of error
- all at the lowest possible operating cost.

Discussion 4 has raised some issues related to the implementation of the thesis which are worth and should be reflected in this chapter:

- Higher code quality and more secure operation (See 4.1 Higher code quality and more secure operation):
 - o Based on the technical suggestion cited, I believe that the project could have placed more emphasis on code analysis and a more detailed and higher quality implementation of different test strategies.

- I would point to the lack of customer willingness as the reason for the failure, and this should be the primary guiding principle for professional work.
- On the basis of the above two lines of thought, I would conclude that the suggestion is a legitimate professional suggestion and, assuming the client is open, it could induce future development.
- Simpler technology portfolio (See 4.2 Simpler technology portfolio):
 - In addition to the determination of the requirement of professional adequacy (which I also explained in the software design phase of this thesis), my own development experience, which (naturally) binds me to certain technological tools, helped me in making this decision. Taking into account the time factor of the software development and the usefulness of developing the software as soon as possible, I decided to choose the known tool (as a professional requirement) instead of the unknown one.
 - In conclusion, I must therefore conclude that, on the basis of the factors described in the chapter referred to above, I still consider the choice of the technology portfolio used to be the most optimal.
- Language barrier removal, localisation (see 4.3 Language barrier removal, localisation):
 - On the basis of the thought process also described in the referenced chapter, I would conclude that the suggestion is a legitimate technical suggestion and, assuming the openness of the client, it could induce a future development.

As a conclusion, it should also be noted that, by aggregating the cost-related findings described in the different chapters, it can be stated that the use of the software can be claimed free of charge with the technological environment I have provided.

Nevertheless, I consider it important to highlight as a conclusion that the software written for this thesis has successfully solved the problem raised (see 1.1 Problem), in the knowledge that the testing scenario (see 3.3.10 Testing summary) and the customer requirements (see 1.6.4.1 Customer requirements) have been met.

Summary

In this Conclusion section, I will present the most valuable sentences that can be quoted from the whole thesis, thus summarizing the conclusions of the thesis.

"By environment variable, I mean the metadata that is essential for the software to run and that defines and enables the software product to function correctly. Examples of such environment variables are the password required to connect to the database or the IP address of the database server (...)

The number of these variables in an enterprise environment (e.g. microservice architecture, distributed environments) can easily reach tens of thousands or even hundreds of thousands.

The complexity of managing environment variables is highly dependent on the management strategy chosen by the software developer (company, individual). There are two main reasons for high cognitive complexity: one is due to the incorrectly (i.e. duplication and redundancy prone) designed management strategy for environmental variables, and the other is due to the growth (i.e. success, new development needs) of the enterprise software product. I would add that this is also a consequence of a flawed management strategy." (See 1.1 Problem)

"The main goal of my thesis is to enable the user to make the necessary changes to the variables in the Azure DevOps interface as simple and as numerous as possible." (See 1.2 Objectives)

"By using the thesis project at enterprise level, the impact of these indicators can be significantly reduced, as workload and repetitive tasks can be reduced by using the software." (See 1.4 Usefulness)

„Taking into account the results of the above subsections, it can be stated that the test scenario was successfully implemented and the test results show that the software met the requirements of the test scenario” (See 3.3.10 Test summary)

"As a conclusion, it should also be noted that, by aggregating the cost-related findings described in the different chapters, it can be stated that the software can be used free of charge in the technological environment provided by me." (See Conclusion 5)

Vision

"Every successful project should include continuous improvement of the service delivered based on feedback afterwards." (Györgydeák, 2021)

There are several options and directions for the future development of the application.

The most obvious future development tasks, and the easiest to implement, may be the development needs that follow from the statements made in Discussion 4 and Conclusion 5.

Nevertheless, the potential functionalities inherent in data persistence may also be a focus for near future development, such as:

- More detailed modification history records
- More detailed query interface for more accurate log entry hits
- More advanced search results for more detailed search results

There is potential to implement additional functionality through deeper and broader integration and use of the Microsoft Azure SDK. Without being exhaustive, the following functionalities provided by this SDK can be added to the project:

- Automation of tasks related to Agile work methodology, simplification of processes (e.g.: Azure DevOps Boards elements with a large number of status changes)
- Query and manipulate Azure container registries

There is also the possibility to explore API functionalities provided by other major cloud providers. These are:

- Amazon Web services (AWS)
- Google Cloud Platform
- Amazon Cloud
- IBM Cloud

Of the listed cloud providers, AWS and Google Cloud Platform also provide available API functionality that could be used to make the product platform independent and more widely deployable. It can be assumed that the profitability and saleability of the product would increase significantly if the application is further developed in a platform-independent direction.

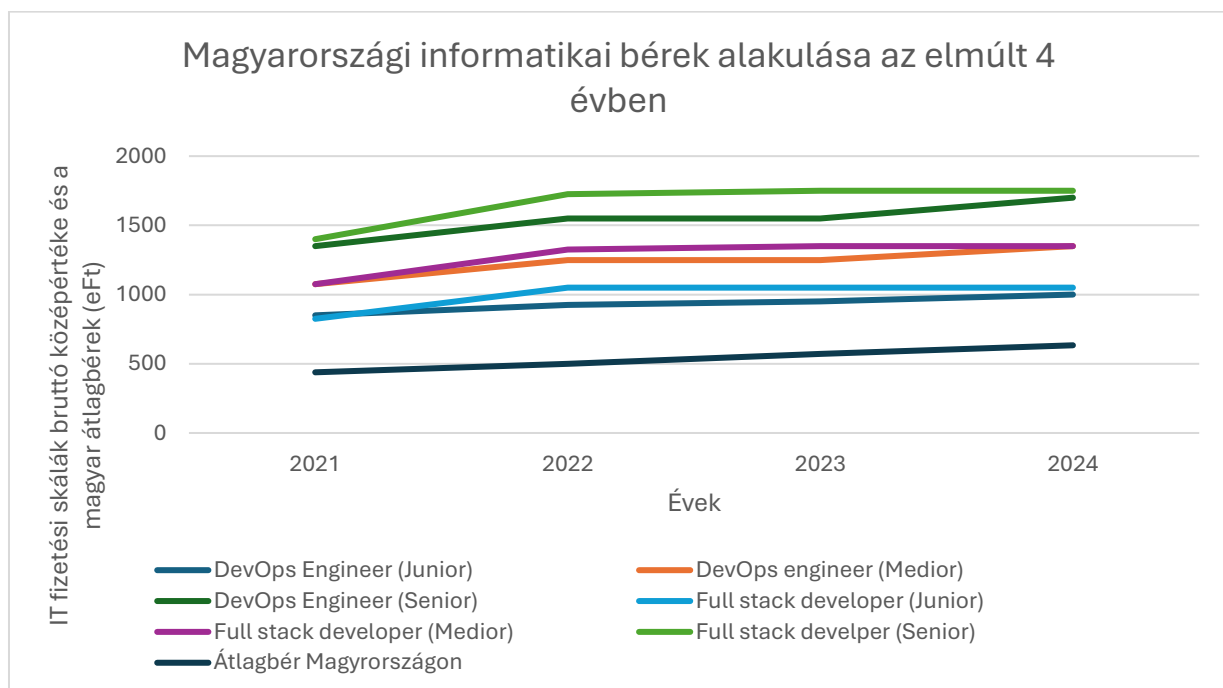
7. MELLÉKLET

Magyarországi munkaerőköltség alakulása az IT iparágban

BlueBird IT Salary Guide

A melléklet tartalmának alapjául a BlueBird IT Salary Guide kiadványainak kimutatásai szolgálnak az elmúlt 4 évből, ezzel szemléltetve az informatika területén tapasztalt bértrendeket. Azért választottam a BlueBird kimutatásait, mert könnyen visszakereshetőek ezek a kiadványok az említett IT staffing vállalat weboldalán és több informatikai szakmára kiterjedten is lehetett összehasonlításokat végezni (<https://bluebird.hu>).

A szakdolgozatban kifejtettem, hogy melyek a dolgozat célcsoportjai, így ebből a szakmalistából taláломra kiválasztottam a DevOps mérnök és a Full stack fejlesztő szakmákat a szemléltetés érdekében. Az összefüggések mélyebb megértése érdekében a Központi Statisztikai Hivatal adatai alapján feltüntettem az adott évre vonatkozó magyarországi bruttó átlagbéreket is (<https://www.ksh.hu>).



61. ábra: Az elmúlt 4 év informatikai bértrendjei és a magyar átlagbérek a BlueBird Salary Guide kiadványok (<https://bluebird.hu/it-fizeteseik/>) és a KSH jelentései alapján (<https://www.ksh.hu/keresetek/>) (saját szerkesztés)

Az 1. ábrán látható, hogy bár a feltüntetett informatikai bérekre vonatkozó értékek az elmúlt 4 évben nem növekedtek szignifikánsan, az mégis kijelenthető, hogy ezek az értékek jelentősen felülmúlták a KSH által közölt magyar átlagbérek értékeit.

8. MELLÉKLET

Copilot promptok és válaszok

Copilot támogatás

Innovatív információs és kommunikációs technológiák

Prompt: „Mit értünk Innovatív információs és kommunikációs technológiák alatt?”

Copilot generált válasza: „Az innovatív információs és kommunikációs technológiák (ICT) azokat a fejlett technológiákat és rendszereket jelentik, amelyek lehetővé teszik az információ hatékony gyűjtését, feldolgozását, tárolását és továbbítását. Ezen technológiák különféle területeken alkalmazhatók, beleértve az üzleti életet, oktatást, egészségügyet és a mindennapi életet is.

Néhány példa az innovatív ICT-re:

1. **Felhőalapú számítástechnika:** Az olyan szolgáltatások, mint a Microsoft Azure, Amazon Web Services (AWS) és Google Cloud, lehetővé teszik a vállalatok számára, hogy skálázható és rugalmas IT-infrastruktúrát építsenek ki anélkül, hogy saját szervereket kellene fenntartaniuk.
2. **Mesterséges intelligencia (AI) és gépi tanulás (ML):** Az AI és ML alkalmazások képesek automatizálni a folyamatokat, előrejelzéseket készíteni és elemzéseket végezni, ami javíthatja a döntéshozatalt és a termelékenységet.
3. **Internet of Things (IoT):** Az IoT-eszközök összekapcsolják a különböző berendezéseket és rendszereket, lehetővé téve az adatok valós idejű gyűjtését és elemzését. Ezáltal hatékonyabbá válik az eszközök kezelése és a folyamatok optimalizálása.
4. **5G és vezeték nélküli technológiák:** Az 5G technológia gyorsabb és megbízhatóbb vezeték nélküli kapcsolatokat tesz lehetővé, ami támogatja az olyan alkalmazásokat, mint az okosvárosok, autonóm járművek és távorvoslás.

5. **Big Data és adatvizualizáció:** A nagy mennyiségű adat elemzése és vizualizálása lehetővé teszi az adatokban rejlő értékek feltárását és az alapvető üzleti döntések meghozatalát.

Ezek a technológiák folyamatosan fejlődnek, és új lehetőségeket nyitnak meg az információ és a kommunikáció területén. Ha bármelyik konkrét téma mélyebben érdekel, szívesen beszélgetek róla tovább!”

Minimal API vagy Controller .NET ökoszisztémában

Prompt: „.NET keretrendszerben Minimal API-t vagy Controller-t érdemesebb használni egy szakdolgozati projekt esetén?”

Copilot generált válasza: „Ez attól függ, hogy milyen típusú projektet készítesz a szakdolgozatodhoz, és mi a célod. Mind a Minimal API, mind a Controller-alapú megközelítésnek megvannak az előnyei és hátrányai. Segítek eligazodni:

Minimal API

- **Előnyök:**
 - **Egyszerűség:** Gyorsan és kevés kóddal hozhatsz létre egy REST API-t.
 - **Könnyű indulás:** Ideális, ha kisebb, egyszerű projekteket készítesz.
 - **Korszerű:** A .NET 6-tól kezdve elérhető és modern API-építési megközelítést kínál.
 - **Kisebb erőforrás-igény:** Minimalizált a kód, ami kevesebb memóriát használhat.
- **Hátrányok:**
 - **Komplexitásnál limitált:** Ha a projekt nagyobb és bonyolultabb, akkor nehezebb skálázni.
 - **Kisebb rugalmasság:** Kevésbé strukturált, így összetettebb projektekhez kevésbé ideális.”

Apache Kafka példány futtatásának követelményei

Prompt: „Mi szükségeltetik egy Apache Kafka példány futtatásához?”

Copilot generált válasza: „Egy Apache Kafka példány futtatásához az alábbi alapvető dolgokra van szükséged:

1. **Szerver(ek):** Legalább egy számítógépre vagy virtuális gépre van szükség, amely futtatja a Kafka brokert és a Zookeeper-t (vagy Raft protokollra épülő MetaStore-t újabb verziókban). Kisebb rendszerekhez egyetlen szerver is elegendő, de nagyobb rendszerekhez több broker szükséges a fürtözéshez.
2. **Java telepítve:** A Kafka a Java Runtime Environment-et (JRE) vagy Java Development Kit-et (JDK) használja. Általában egy stabil verziójú OpenJDK (pl. 11-es vagy újabb) megfelel.
3. **Operációs rendszer:** Kafka többnyire Linux rendszeren fut a legjobban, de Windows vagy MacOS alatt is működhet.
4. **Hálózati beállítások:** A Kafka kommunikációhoz stabil hálózati kapcsolatra van szükség. Nyitva kell lennie a Kafka broker(ek) által használt port(ok)nak (alapértelmezett: 9092).
5. **Konfigurációs fájlok:** Meg kell határozni a Kafka és a Zookeeper (vagy MetaStore) konfigurációs paramétereit, például logok helyét, adatfájlok elérési útját és memóriabeállításokat.
6. **Hardver:** A Kafka nagy mennyiségű adat feldolgozására készült, ezért nagyobb terhelés esetén megfelelő CPU, RAM (legalább 8GB) és tárhely szükséges.
7. **Kafka bináris fájlok:** Le kell tölteni az Apache Kafka hivatalos verzióját az Apache Kafka weboldaláról.

Amennyiben érdekel a pontos telepítési lépésekkel kapcsolatban is további részletezés, szívesen segítek!”

Dockerfile generálás .NET és ReactJS alapokon futó szoftvereknek

Prompt: „Generálj nekem egy Dockerfile-t egy .NET alapú API-nak”

Copilot generált válasza: „Természetesen! Íme egy példa Dockerfile egy .NET alapú API-hoz:

```

dockerfile

# Build szakasz

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build

WORKDIR /src

# Másoljuk a projekt fájlokat és telepítjük a függőségeket

COPY *.csproj ./

RUN dotnet restore

# Másoljuk az összes forrást és buildeljük az alkalmazást

COPY . ./

RUN dotnet publish -c Release -o /app

# Runtime szakasz

FROM mcr.microsoft.com/dotnet/aspnet:8.0

WORKDIR /app

# Az alkalmazás binárisainak másolása a build szakaszból

COPY --from=build /app .

# Az alkalmazás indítási parancsa

ENTRYPOINT ["dotnet", "YourApi.dll"]

```

Ha szeretnéd testreszabni a Dockerfile-t vagy hozzáadni extra funkciókat, mint például környezeti változók, jelöld meg, és segíték!”

Prompt: „Most generálj egy Dockerfile-t egy ReactJS projekthez”

Copilot generált válasza: „Rendben! Íme egy példa Dockerfile egy ReactJS projekt számára:

```

dockerfile

# Base image a Node.js-hez (verzió változtatható, ha szükséges)

```



```
FROM node:18 AS build

# Mappa beállítása az alkalmazás fájljainak

WORKDIR /app

# package.json és package-lock.json másolása

COPY package*.json ./

# Függőségek telepítése

RUN npm install

# Az összes forráskód másolása

COPY . .

# Az alkalmazás buildelése a production környezetre

RUN npm run build

# Nginx használata az épített fájlok hosztolására

FROM nginx:alpine

WORKDIR /usr/share/nginx/html

# Buildelt fájlok másolása a build szakaszból

COPY --from=build /app/build .

# Nginx indítás

ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

Ezzel a parancsokkal elérheted az alkalmazást a böngészőben a <http://localhost:3000> címen. Szívesen segítek, ha van még kérdésed!”

9. MELLÉKLET

ChatGPT promptok és válaszok

ChatGPT támogatás

Szakterületi jogi ismeretek tantárgy a szakdolgozati témához fűződő összefüggésvizsgálata

Prompt: „Milyen kapcsolatban állhat a Szakterületi jogi ismeretek tantárgy az alábbi szakdolgozati témához: Az adminisztratív hatékonyság növelése - A Microsoft Azure SDK alkalmazása az Azure DevOps felületén történő adminisztratív feladatok egyszerűsítésére”

ChatGPT generált válasza: „Szakdolgozati témád, amely az adminisztratív hatékonyság növelésére összpontosít az Azure SDK és Azure DevOps alkalmazásával, számos ponton kapcsolódhat a Szakterületi jogi ismeretek tantárgyhoz. Az alábbiakban néhány példát hozok arra, hogy milyen módon lehet összefüggésbe hozni a két területet:

1. **Adatvédelem és biztonság:** Az adminisztratív hatékonyság növelése az Azure SDK és DevOps eszközökkel kiterjedhet olyan területekre is, mint az adatvédelem és az információbiztonság. A jogi ismeretek segíthetnek megérteni és betartani a vonatkozó adatvédelmi jogszabályokat és előírásokat, például a GDPR-t.
2. **Szabályozási megfelelés:** Az adminisztratív folyamatok egyszerűsítése során fontos biztosítani, hogy minden tevékenység megfeleljen a vonatkozó jogszabályoknak és szabályozásoknak. A Szakterületi jogi ismeretek tárgy keretében megtanult szabályozási és megfelelési kérdések itt központi szerepet játszhatnak.
3. **Szerződések és licencelés:** Az Azure és DevOps használata során különböző szerződések és licencelési feltételek érvényesek lehetnek, amelyek jogi ismereteket igényelnek. A tantárgy során megszerzett tudás segíthet eligazodni ezekben a kérdésekben.
4. **Jogi felelősség és kockázatkezelés:** Az adminisztratív feladatok automatizálása és az IT eszközök használata jogi felelősséget is magával hozhat. A jogi ismeretek segíthetnek azonosítani és kezelni a lehetséges jogi kockázatokat.

Ezek alapján úgy gondolom, hogy a Szakterületi jogi ismeretek tantárgy hasznos alapot nyújthat a szakdolgozatod elkészítéséhez és az adminisztratív feladatok hatékonyabbá tételéhez a Microsoft Azure SDK és Azure DevOps alkalmazásával. Remélem, ez segít eligazodni a kapcsolódási pontok között!”

Szoftververziók verziószám alapján történő rendezése

Prompt: „Írj egy függvényt Javascript programnyelvben, ami a verziószámok alapján növekvő sorrendbe rendezi a verziószámokat tartalmazó listát. Egy példa egy verziószámra refs/tags/0.1.0”

ChatGPT generált válasza: „Itt van egy példa egy olyan JavaScript függvényre, amely egy verziószámokat tartalmazó listát növekvő sorrendbe rendez:

javascript

```
function compareVersions(a, b) {  
  
    const partsA = a.split('/').pop().split('.').map(Number);  
  
    const partsB = b.split('/').pop().split('.').map(Number);  
  
  
    for (let i = 0; i < Math.max(partsA.length, partsB.length); i++) {  
  
        const diff = (partsA[i] || 0) - (partsB[i] || 0);  
  
        if (diff !== 0) return diff;  
  
    }  
  
    return 0;  
  
}
```

```
function sortVersions(versionList) {  
  
    return versionList.sort(compareVersions);  
  
}
```

// Használat:

```
const versionList = [  
  "refs/tags/0.1.0",  
  "refs/tags/1.0.0",  
  "refs/tags/0.10.0",  
  "refs/tags/0.2.0"  
];
```

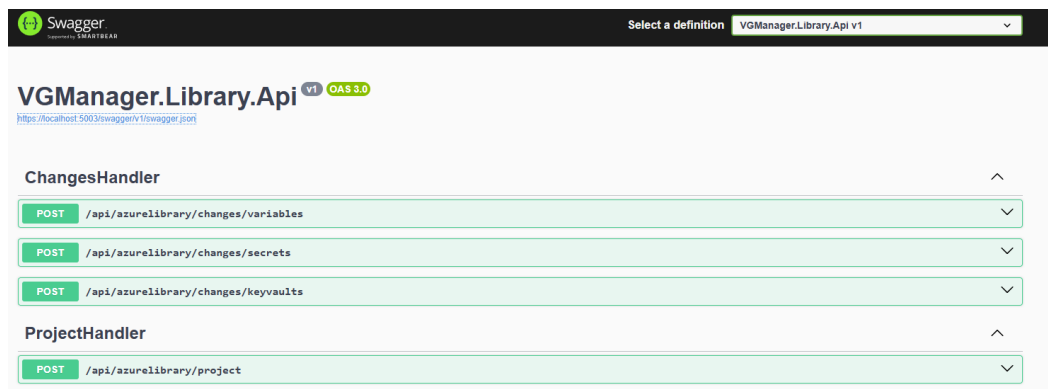
```
const sortedVersionList = sortVersions(versionList);  
  
console.log(sortedVersionList);
```

Ez a kód először kettébontja a verziószámokat a '/' karakter mentén, majd a '.' karakter mentén is, és számokká alakítja azokat. Ezután az `Array.sort()` metódust használja a rendezéshez. Remélem, segít! Ha bármilyen kérdésed van, nyugodtan kérdezz!”

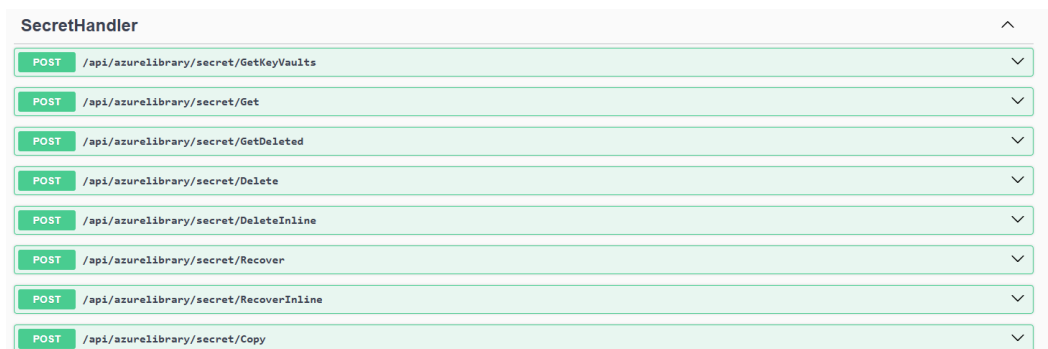
10. MELLÉKLET

Library API Swagger

Library API swagger ábra formátumban



62. ábra: Library API - módosítások és azure projekt lekérdezéssel kapcsolatos végpontok (saját szerkesztés)



63. ábra: Library API - Secret műveletekre vonatkozó végpontok (saját szerkesztés)



64. ábra: Library API - Környezeti változó műveletekre vonatkozó végpontok (saját szerkesztés)

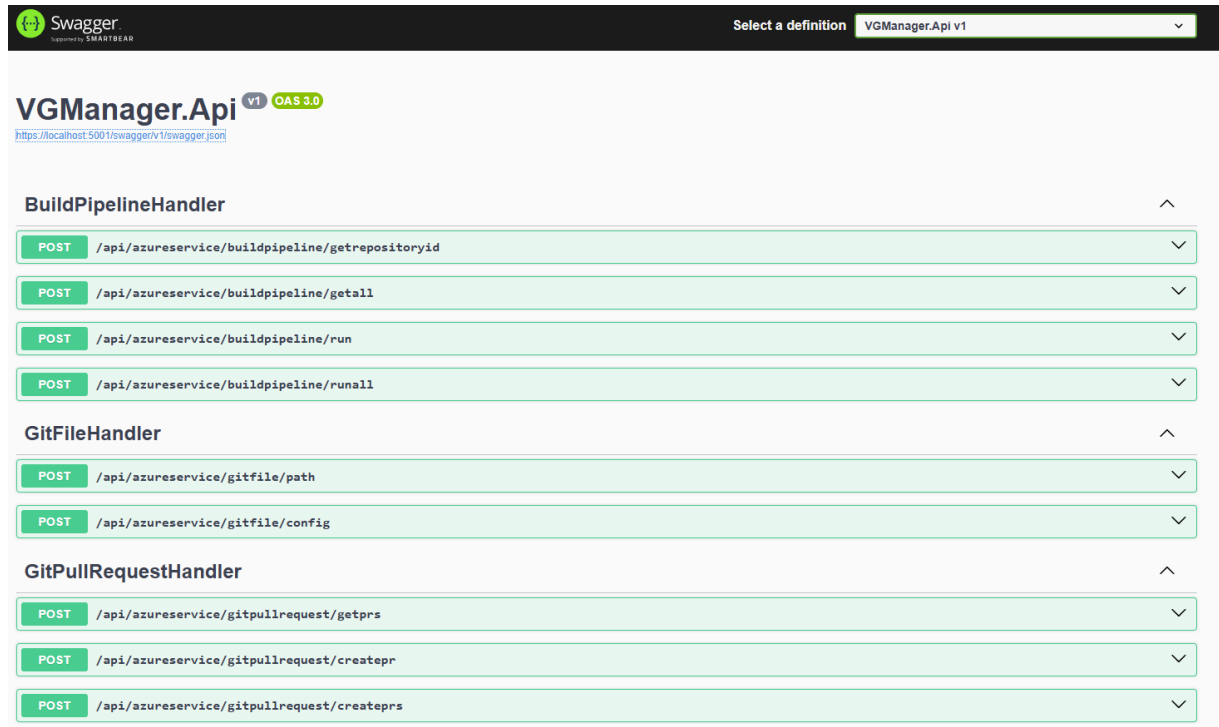
Generált Swagger JSON

Swagger JSON generálása is lehetséges, annak terjedelme miatt ezt nem jelenítem meg a dolgozatban.

11. MELLÉKLET

API Swagger

API swagger ábra formátumban



65. ábra: Build pipeline, git fájl és git pull request lekérdezéssel kapcsolatos végpontok (saját szerkesztés)

GitRepositoryHandler		^
POST	/api/azureservice/gitrepository	▼
POST	/api/azureservice/gitrepository/variables	▼
GitVersionHandler		^
POST	/api/azureservice/gitversion/branches	▼
POST	/api/azureservice/gitversion/tags	▼
POST	/api/azureservice/gitversion/latesttags	▼
POST	/api/azureservice/gitversion/tagcreation	▼
ProfileHandler		^
POST	/api/azureservice/profile	▼
ReleasePipelineHandler		^
POST	/api/azureservice/releasepipeline/environments	▼
POST	/api/azureservice/releasepipeline/variableGroups	▼
POST	/api/azureservice/releasepipeline/projects	▼

66. ábra: Git repository, git verzió, profil és release pipeline vonatkozású végpontok (saját fejlesztés)

Generált Swagger JSON

Swagger JSON generálása is lehetséges, annak terjedelme miatt ezt nem jelenítem meg a dolgozatban.

12. MELLÉKLET

Tesztforgatókönyv

Tesztesetek

Bejelentkezés

Sikeres művelet

Teszt eset:

- Feladat:
 - Hitelesítési adatokkal be kell tudni jelentkezni az alkalmazásfelületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.1 Bejelentkezés)
 - Hitelesítési adatok maszkolással felsorolva (Lásd 67. ábra):

```
REACT_APP_TENANT_ID="7e3[REDACTED]f2b5"  
REACT_APP_CLIENT_ID="572[REDACTED]36bd"  
REACT_APP_CLIENT_SECRET="85n[REDACTED]Man5"  
REACT_APP_PAT="GKU[REDACTED]xuzC"  
REACT_APP_ORGANIZATION="beviktör95"
```

67. ábra: Hitelesítési adatok maszkolt formátumban (saját szerkesztés)

- Elvárt követelmény:
 - Az alkalmazás bejelentkeztet, elérhetőek a funkcionálisok.
 - A felület üdvözlí a felhasználót

Sikertelen művelet

Teszt eset:

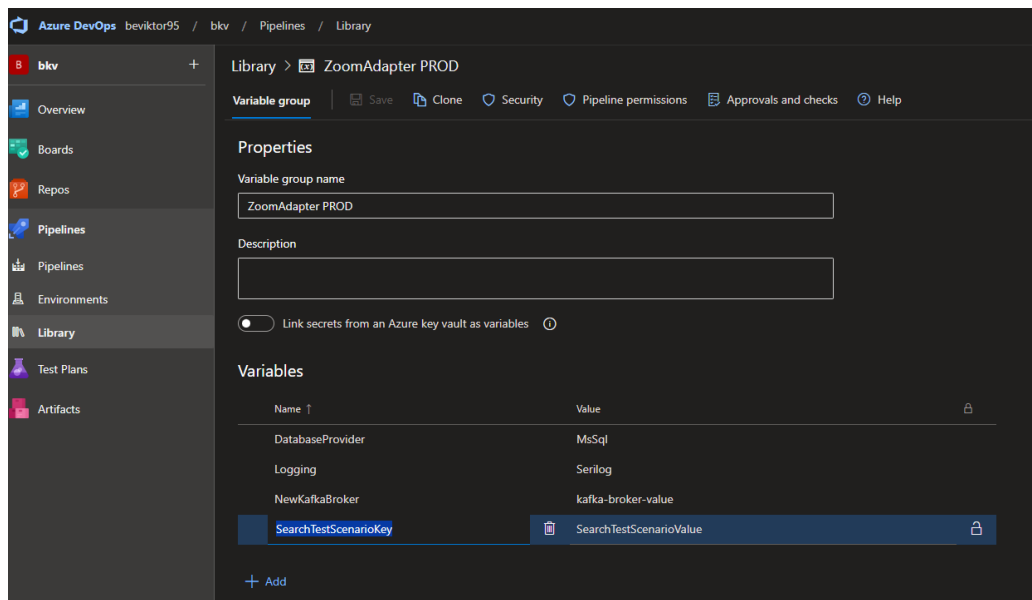
- Feladat:
 - Bejelentkezés - Sikeres művelet fejezetben megadott feladat, csak módosított (tehát helytelen) hitelesítési adatokkal
- Elvárt eredmény:
 - A webkliens nem enged túl a bejelentkező felületen
 - Hibaüzenet jelenik meg a helytelen bejelentkezési adatok miatt

Környezeti változó lekérdezés

Sikeres művelet

Teszt eset:

- Feladat:
 - o A bejelentkezést követően a szoftvernek ki kell listáznia a SearchTestScenarioKey környezeti változót és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.2 Környezeti változók lekérdezése, manipulálása)



68. ábra: Keresett környezeti változót tartalmazó variable group Azure DevOps felületen (saját szerkesztés)

- Elvárt eredmény a 68. ábra alapján:
 - o Egy találat megjelenítése: ZoomAdapter PROD variable group tartalmazza a környezeti változót

Sikertelen művelet

Teszt eset:

- Feladat:
 - o Környezeti változó lekérdezés - Sikeres művelet fejezetben megadott feladat, csak időközben az adapter serveroldali komponenshez tartozó konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.

- Elvárt eredmény:
 - Hibaüzenet jelenik meg a lekérdezés sikertelenségéről
 - Nem jelenik meg a keresés eredménye a felületen

Környezeti változó manipulálás

Környezeti változó hozzáadás

Sikeres művelet

Teszt eset:

- Feladat:
 - A bejelentkezést követően a szoftvernek hozzá kell adnia a SearchTestScenarioKey2 környezeti változót a ZoomAdapter PROD variable group-hoz és megjelenítenie az eredményt a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.2 Környezeti változók lekérdezése, manipulálása), illetve az Azure DevOps-ban is ellenőrizni kell az eredményt.
- Elvárt eredmény:
 - Megjelenik az új, hozzáadott környezeti változó az eredménylistában
 - Azure DevOps-ban látható az új környezeti változó (említett variable group aktuális állapotáért lásd: 68. ábra)

Sikertelen művelet

Teszt eset:

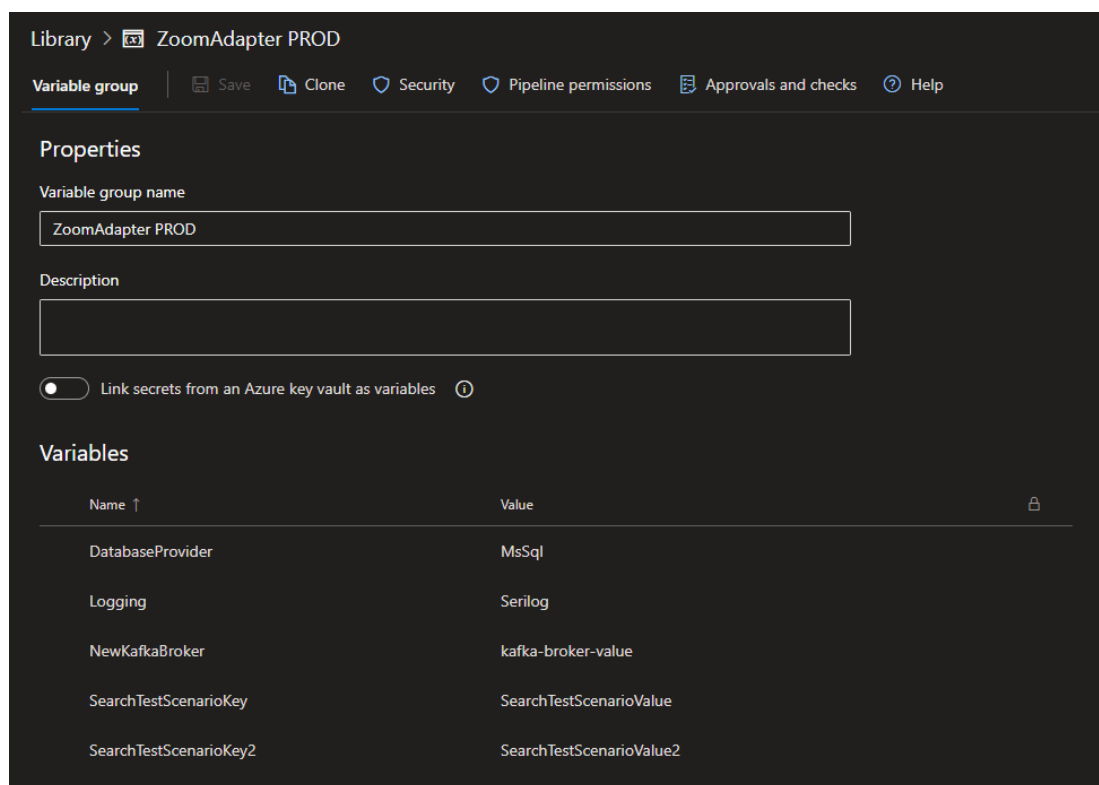
- Feladat:
 - Környezeti változó manipulálás - Sikeres művelet fejezetben megadott feladat (csak SearchTestScenarioKey3 környezeti változóval), időközben azonban az eseményfolyam-kezelő konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.
- Elvárt eredmény:
 - Hibaüzenet jelenik meg a hozzáadás sikertelenségéről
 - Nem jelenik meg az új környezeti változó az Azure DevOps felületén

Környezeti változó módosítás

Sikeres művelet

Teszt eset:

- Feladat:
 - A bejelentkezést követően a szoftvernek módosítania kell a SearchTestScenarioKey2 környezeti változót a ZoomAdapter PROD variable group-ban a következő értékre: ModifiedSearchTestScenarioValue2 a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.2 Környezeti változók lekérdezése, manipulálása).
 - Meg kell jelenítenie az eredményt a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.2 Környezeti változók lekérdezése, manipulálása), illetve az Azure DevOps-ban is ellenőrizni kell az eredményt.



69. ábra: ZoomAdapter PROD variable group jelenlegi értéklísta (saját szerkesztés)

- Elvárt eredmény:
 - Megjelenik a módosított környezeti változó az eredménylistában

- Azure DevOps-ban látható a környezeti változó új értéke, tehát módosul a 69. ábrán látható utolsó környezeti változó értéke

Sikertelen művelet

Teszteset:

- Feladat:
 - Környezeti változó módosítás - Sikeres művelet fejezetben megadott feladat, annyi különbséggel, hogy olyan környezeti változót szeretnénk módosítani (például: SearchTestScenarioKey3), ami nem is létezik a variable group-ban.
- Elvárt eredmény:
 - Nem jelenik meg egy találat sem a találati listában a műveletjövőahagyási felületen.
 - Nem jelenik meg módosulás a variable group-ban az Azure DevOps felületén.

Környezeti változó törlés

Sikeres művelet

Teszteset:

- Feladat:
 - A bejelentkezést követően a szoftvernek törölnie kell a SearchTestScenarioKey2 környezeti változót a ZoomAdapter PROD variable group-ból a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.2 Környezeti változók lekérdezése, manipulálása).
- Elvárt eredmény:
 - Sikeres státuszú visszajelzés érkezik a szoftver webfelületéről
 - Azure DevOps-ban már nem látható a törölni kívánt környezeti változó

Sikertelen művelet

Teszteset:

- Feladat:
 - Környezeti változó törlés - Sikeres művelet fejezetben megadott feladat, annyi különbséggel, hogy olyan környezeti változót szeretnénk törölni, ami már nem is létezik a variable group-ban (például: SearchTestScenarioKey2).
- Elvárt eredmény:

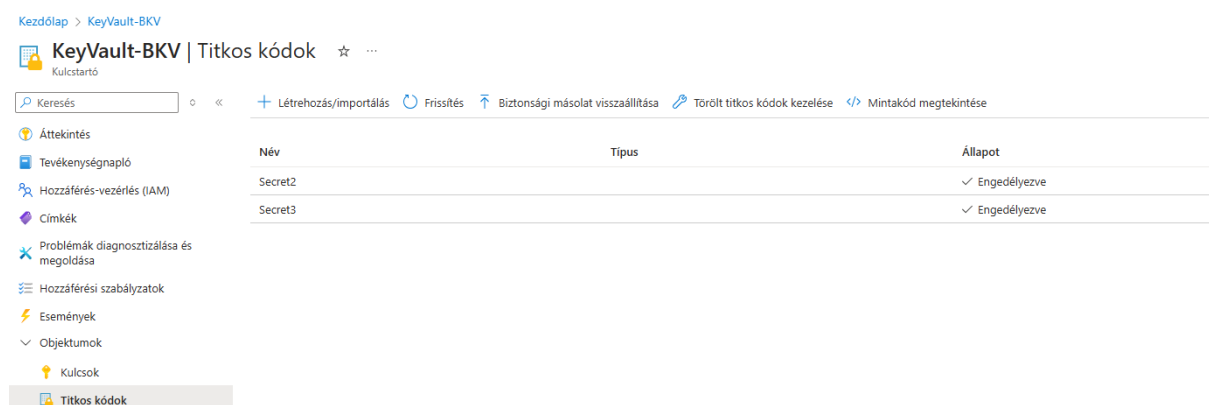
- Nem jelenik meg egy találat sem a találati listában a műveletjövőhagyási felületen.
- Nem jelenik meg módosulás a variable group-ban az Azure DevOps felületén.

Secret lekérdezés

Sikeres művelet

Teszt eset:

- Feladat:
 - A bejelentkezést követően a szoftvernek ki kell listáznia az összes aktív secret-et a KeyVault-BKV nevű key vaultból és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.3 Secretok lekérdezése, manipulálása)



70. ábra: Lekérdezés tárgyát képező key vault jelenlegi secret listája (saját szerkesztés)

- Elvárt eredmény a 70. ábra alapján:
 - Secret2 és Secret3 secretok megjelenítése a kezelőfelületen

Sikertelen művelet

Teszt eset:

- Feladat:
 - Secret lekérdezés - Sikeres művelet fejezetben megadott feladat, csak időközben az adapter szerveroldali komponenshez tartozó konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.
- Elvárt eredmény:
 - Hibaüzenet jelenik meg a lekérdezés sikertelenségéről

- Nem jelenik meg egy találat sem a találati listában a műveletjövőhagyási felületen.

Secret manipulálás

Secret törlés

Sikeres művelet

Teszteset:

- Feladat:
 - A bejelentkezést követően a szoftvernek törölnie kell a Secret2 secret-et a KeyVault-BKV key vault-ból a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.3 Secretok lekérdezése, manipulálása).
- Elvárt eredmény:
 - Azure Portal-ban már nem látható a törölni kívánt secret

Sikertelen művelet

Teszteset:

- Feladat:
 - Secret manipulálás - Sikeres művelet fejezetben megadott feladat, csak időközben az adapter szerveroldali komponenshez tartozó konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.
- Elvárt eredmény:
 - Hibaüzenet jelenik meg a művelet sikertelenségéről
 - Azure Portal-ban nem törlődik a törölni kívánt secret

Secret reaktiválás

Sikeres művelet

Teszteset:

- Feladat:
 - A bejelentkezést követően a szoftvernek reaktiválnia kell a Secret2 secret-et a KeyVault-BKV key vault-ban a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.3 Secretok lekérdezése, manipulálása).
- Elvárt eredmény:

- Azure Portal-ban újra látható a reaktivált secret

Sikertelen művelet

Teszt eset:

- Feladat:
 - Secret reaktiválás - Sikeres művelet fejezetben megadott feladat, csak időközben az adapter szerveroldali komponenshez tartozó konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.
- Elvárt eredmény:
 - Hibaüzenet jelenik meg a művelet sikertelenségéről
 - Azure Portal-ban nem reaktiválódik a törölni kívánt secret

Key vault másolás

Sikeres művelet

Teszt eset:

- Feladat:
 - A bejelentkezést követően a szoftvernek át kell másolnia a KeyVaultBV95 key vault-ból a KeyVault-BKV key vault-ba a secretokat a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.3 Secretok lekérdezése, manipulálása).

Kezdőlap > KeyVaultBV95

KeyVaultBV95 | Titkos kódok ☆ ...

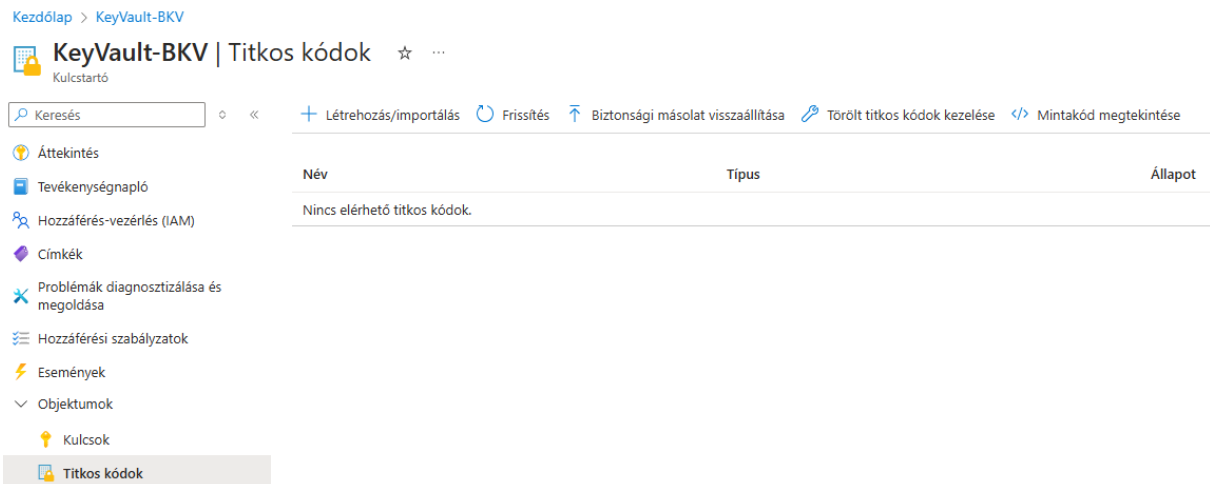
Kulcstartó

Keresés 🔍 ◊ « + Létrehozás/importálás ↻ Frissítés ↑ Biztonsági másolat visszaállítása 🔗 Törölt titkos kódok kezelése </> Mintakód megtekintése

Név	Típus	Állapot
Secret1		✓ Engedélyezve
Secret2		✓ Engedélyezve
Secret3		✓ Engedélyezve

🔍 Áttekintés
 📅 Tevékenységnapló
 🔑 Hozzáférés-vezérlés (IAM)
 🏷️ Címkék
 🔧 Problémák diagnosztizálása és megoldása
 ⚙️ Hozzáférési szabályzatok
 ⚡ Események
 ▼ Objektumok
 🔑 Kulcsok
 🗂️ Titkos kódok

71. ábra: KeyVaultBV95 key vault jelenlegi állapota Azure Portalon (saját szerkesztés)



72. ábra: KeyVault-BKV key vault jelenlegi állapota Azure Portalon (saját szerkesztés)

- Elvárt eredmény 71.ábra és 72. ábra alapján:
 - o KeyVault-BKV key vault secret halmaz tartalmában megegyezik a KeyVaultBV95 key vault secret halmazzal

Sikertelen művelet

Teszteset:

- A teszteléshez visszaállítottam a Key vault másolás - Sikeres művelet fejezetben beállított key vault állapotokat, így az 71. ábra és a 72. ábra tükrözi ennél a tesztelésnél is az alapállást.
- Feladat:
 - o Key vault másolás - Sikeres művelet fejezetben megadott feladat, csak időközben az adapter szerveroldali komponenshez tartozó konténer (manuális előidézéssel) leáll és elérhetetlenné válik a szolgáltatás.
- Elvárt eredmény:
 - o Azure Portal-ban nem másolódnak át az átmásolni kívánt secret-ek

Módosítások lekérdezése

Környezeti változó módosítások lekérdezése

Sikeres művelet

Teszteset:

- Feladat:

- A bejelentkezést követően a szoftvernek ki kell listáznia a 2025.03.10-től 2025.03.17-ig rögzített környezeti változó hozzáadási műveletek naplóbejegyzéseit és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése)

Query Query History

```
1 SELECT "Organization", "Project", "User", "VariableGroupFilter", "Key", "Date"
2 FROM public."Variable_additions";
```

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1

	Organization text	Project text	User text	VariableGroupFilter text	Key text	Date timestamp with time zone
1	bevikt95	bkv	Viktor Bócz-Kemény	.*	NewKafkaBroker	2025-03-11 17:41:45.370572+00
2	bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PROD	SearchTestScenarioKey2	2025-03-17 17:18:36.503749+00

73. ábra: Környezeti változó hozzáadási műveleteket naplózó adattábla perzisztált eredményei (saját szerkesztés)

- Elvárt eredmény a 73. ábra alapján:
 - Két hozzáadással kapcsolatos találat megjelenítése webkliensen

Secret módosítások lekérdezése

Sikeres művelet

Tesztetset:

- Feladat:
 - A bejelentkezést követően a szoftvernek ki kell listáznia a 2025.03.10-től 2025.03.17-ig rögzített secret módosítások naplóbejegyzéseit és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése)

Query Query History Scratch Pad x

```

1 SELECT "Id", "KeyVaultName", "SecretNameRegex", "ChangeType", "User", "Date"
2 FROM public."Secret_changes";

```

Data Output Messages Notifications

Showing rows: 1 to 7 Page No: 1 of 1

	Id [PK] text	KeyVaultName text	SecretNameRegex text	ChangeType integer	User text	Date timestamp with time zone
1	d1732bc9-3dd5-4cee-b128-1bfc169fba...	KeyVault-BKV	Secret1	1	Viktor Bócz-Kemény	2025-03-11 13:43:16.2349+00
2	53b0c5a8-3390-4641-9413-3930cbe5...	KeyVault-BKV	Secret1	1	Viktor Bócz-Kemény	2025-03-11 20:26:15.198596+00
3	678d1056-daf4-4aa0-9ccb-edfb347a42...	KeyVault-BKV	Secret1	0	Viktor Bócz-Kemény	2025-03-11 20:26:36.002846+00
4	658accba-9faa-4790-bd4c-4f67360a32...	KeyVault-BKV	Secret1	1	Viktor Bócz-Kemény	2025-03-11 20:27:27.889563+00
5	583a3740-3c5e-44d6-af78-9c3925f0d5...	KeyVault-BKV	Secret2	1	Viktor Bócz-Kemény	2025-03-17 18:39:44.57465+00
6	91370581-0daf-47e8-abb3-68ec5a7a3...	KeyVault-BKV	Secret2	0	Viktor Bócz-Kemény	2025-03-17 18:51:48.10414+00
7	f2d264f1-24d4-4c07-8baf-64a8712bfb44	KeyVault-BKV	Secret2	1	Viktor Bócz-Kemény	2025-03-17 18:56:37.16194+00

74. ábra: Secret módosítások naplóbejegyzései (saját szerkesztés)

- Elvárt eredmény a 74. ábra alapján:
 - o Hét secret módosítással kapcsolatos találat megjelenítése webkliensen

Key vault másolások lekérdezése

Sikeres művelet

Teszttesetek:

- Feladat:
 - o A bejelentkezést követően a szoftvernek ki kell listáznia a 2025.03.10-től 2025.03.17-ig rögzített key vault másolások naplóbejegyzéseit és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.4 Környezeti változó, secret és key vault módosítások lekérdezése)

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × Operations/postgres@Local* ×

Operations/postgres@Local

Query Query History Scratch Pad ×

```

1 SELECT "Id", "OriginalKeyVault", "DestinationKeyVault", "User", "Date"
2 FROM public."KeyVault_copy";

```

Data Output Messages Notifications

Showing rows: 1 to 9 Page No: 1 of 1

	Id [PK] text	OriginalKeyVault text	DestinationKeyVault text	User text	Date timestamp with time zone
1	2f83924e-2eb1-416e-8503-05fd25baf5e1	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-11 13:43:52.107085+00
2	64bc688b-464a-4b9b-a937-5bd0c5e13...	KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	2025-03-11 20:47:31.192897+00
3	ba847416-e404-48ea-9a13-0a65613cb...	KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	2025-03-11 20:47:36.544317+00
4	49b5896e-da2d-401f-b8ea-8d4393a2a3...	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 19:15:02.538069+00
5	9dba3007-ca74-4f1b-9137-a48f4910d9...	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 21:48:33.91071+00
6	9abb62fe-6b97-4415-b47b-33af1f5f1599	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 21:49:08.021432+00
7	7b748943-8e8b-49fa-9938-f9d8e9fe919f	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 21:50:05.843918+00
8	b2d2c54d-5157-487b-b87b-07ef8c5f76...	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 21:50:23.987857+00
9	ea6b4880-70b2-4ebf-8a51-f7dfd24ed080	KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	2025-03-17 21:51:35.152273+00

75. ábra: Key vault másolásk naplóbejegyzései (saját szerkesztés)

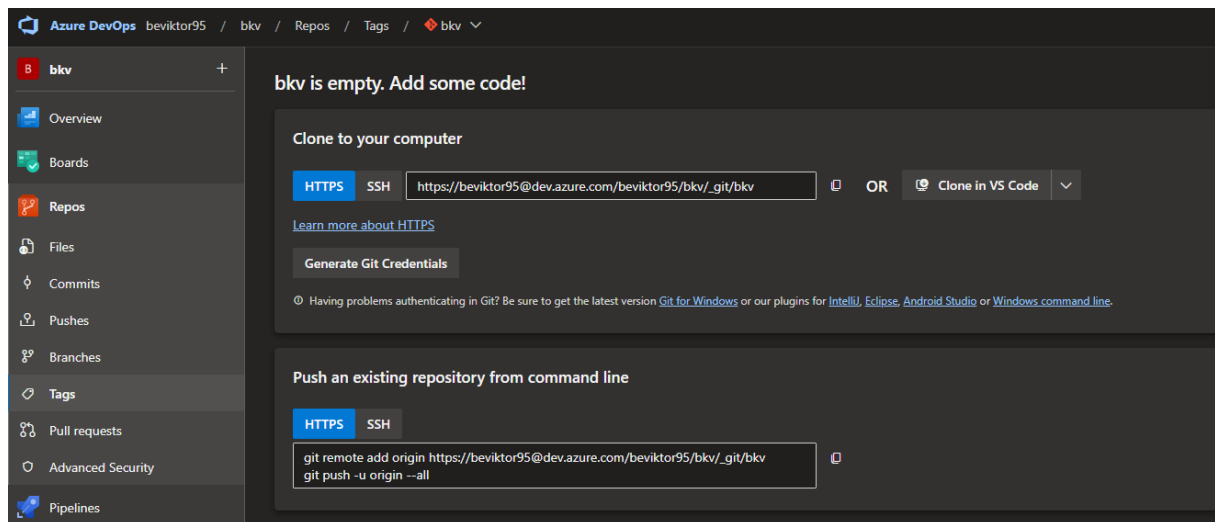
- Elvárt eredmény a 75. ábra alapján:
 - o 9 db key vault másolással kapcsolatos találat megjelenítése webkliensen

Utolsó szoftververzió lekérdezése

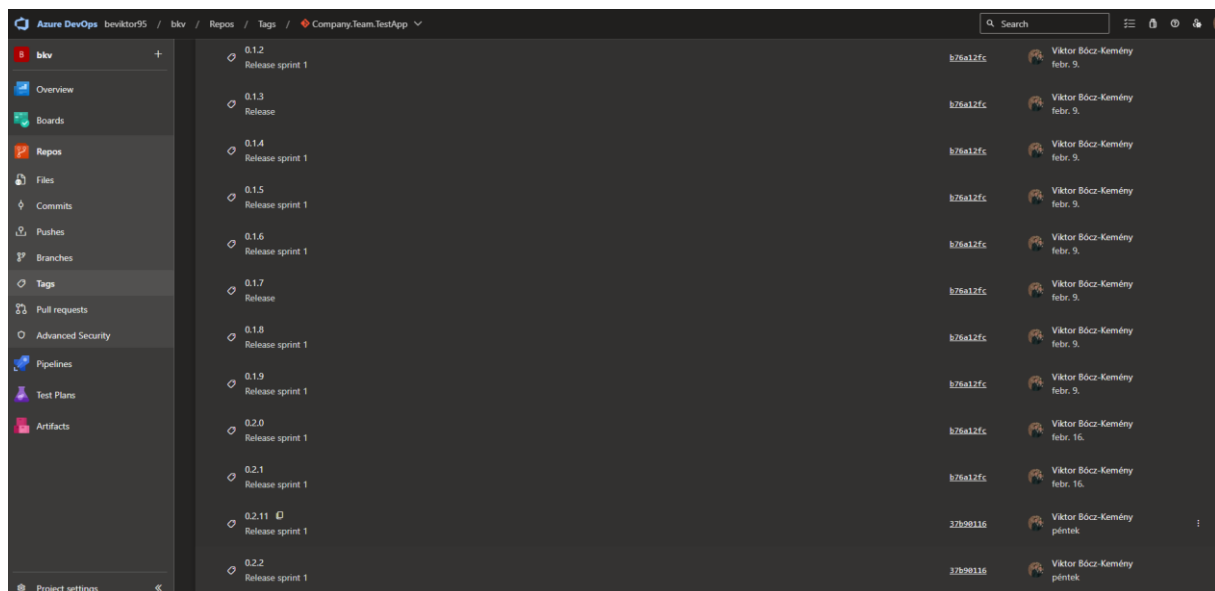
Sikeres művelet

Tesztetset:

- Feladat:
 - o A bejelentkezést követően a szoftvernek ki kell listáznia a bkv néven futó Azure projekt git repository-jainak utolsó stabil verzióját a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.6 Utolsó szoftververziók lekérdezése)



76. ábra: Automatikusan létrehozott git repository, nincsenek tagek (saját szerkesztés)



77. ábra: Általam létrehozott Git repository és az ahhoz tartozó stabil verziók (saját szerkesztés)

- Elvárt eredmény a 76. ábra és 77. ábra alapján:
 - o 2 elemű lista tér vissza a git repository tematikájú táblázatban eredményként
 - o A bkvs nevű repository-nál nincs stabil verzió (tag), a Company.Team.TestApp nevű repository-nál az utolsó stabil verzió a 0.2.11-es

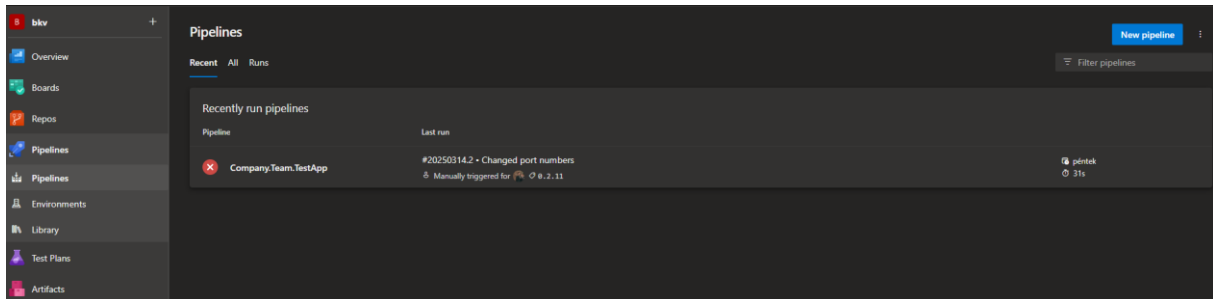
Pipeline futtatása

Sikeres művelet

Tesztesetek:

- Feladat:

- A bejelentkezést követően a szoftvernek ki kell listáznia a build pipeline-okat és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.5 Pipeline futtatás)
- A kilistázott build pipeline-ok esetén el kell tudni indítani a pipeline futtatást.



78. ábra: Azure DevOps-ban található build pipeline-ok (saját szerkesztés)

- Elvárt eredmény:
 - Build pipeline-ok kilistázásra kerülnek (jelen esetben egy darab, lásd 78. ábra)
 - A listaelemeken elindítható a pipeline futtatás

Pull request létrehozás

Sikeres művelet

Tesztesetek:

- Feladat:
 - A bejelentkezést követően a szoftvernek ki kell listáznia a git repository-kat és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.5 Pipeline futtatás)
 - A kilistázott git repository-k esetén el kell tudni indítani a pull request létrehozás műveletet.
- Elvárt eredmény:
 - Git repository lista megjelenik a lekérdezés során
 - Pull request létrehozása sikeres

Sikertelen művelet

Teszteset:

- Feladat:

- Pull request létrehozás - Sikeres művelet fejezetben megadott feladat, csak a kiválasztott git repository-hoz nem tartoznak branch-ek.
- Elvárt eredmény:
 - Hibaüzenet jelenik meg a művelet sikertelenségéről
 - Azure DevOps-ban nem jön létre a pull request

Pull request lekérdezés

Sikeres művelet

Tesztesetek:

- Feladat:
 - A bejelentkezést követően a szoftvernek ki kell listáznia az aktív pull request találati listát és megjelenítenie a szakirodalomban bemutatott felületen a szakirodalomban bemutatott folyamat alapján (Lásd a szakdolgozatban: 3.2.1.5 Pipeline futtatás)
- Elvárt eredmény:
 - Git pull request lista megjelenik a lekérdezés során

13. MELLÉKLET

Teszteredmények

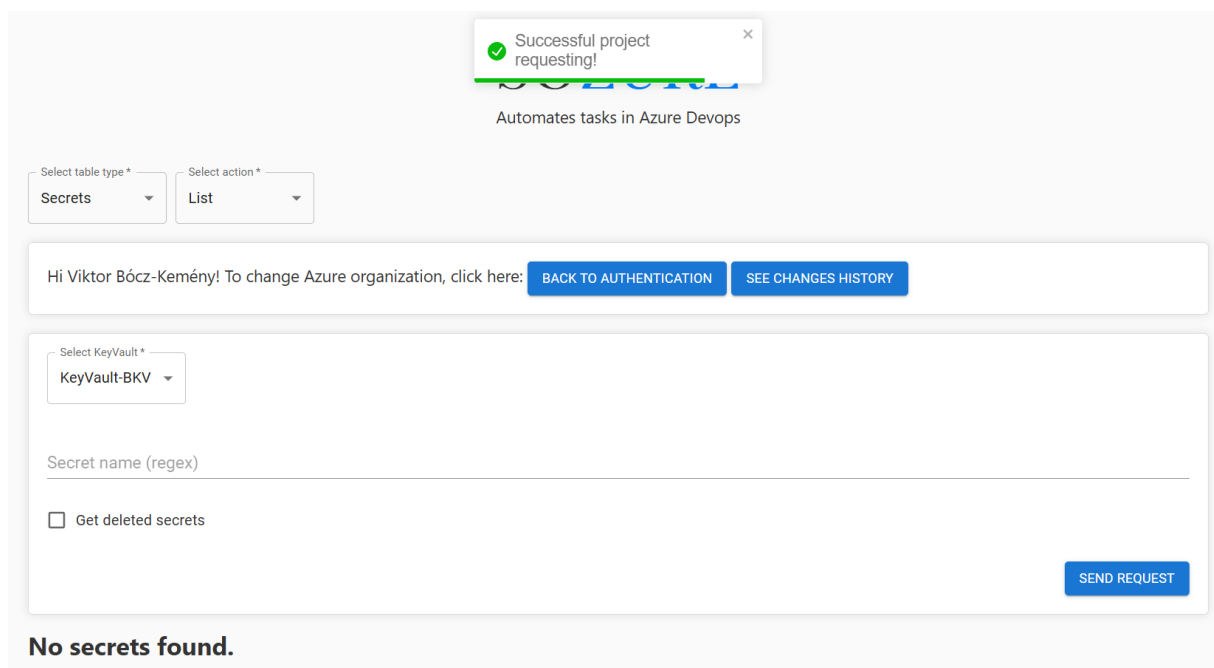
Tesztelés eredménye

Bejelentkezés

Sikeres művelet

Teszteset:

- Elvárt eredmény:
 - Az alkalmazás bejelentkeztet, elérhetőek a funkciók.
 - A felület üdvözlí a felhasználót



The screenshot shows a web interface for Azure DevOps. At the top, a green notification box says "Successful project requesting!". Below it, the text "Automates tasks in Azure DevOps" is visible. The main form has two dropdown menus: "Select table type *" with "Secrets" selected, and "Select action *" with "List" selected. Below these is a message: "Hi Viktor Bócz-Kemény! To change Azure organization, click here:" followed by two buttons: "BACK TO AUTHENTICATION" and "SEE CHANGES HISTORY". Underneath is a section for "Select KeyVault *" with "KeyVault-BKV" selected. There is a text input field for "Secret name (regex)". A checkbox labeled "Get deleted secrets" is present. A blue "SEND REQUEST" button is at the bottom right. At the bottom of the form, it says "No secrets found."

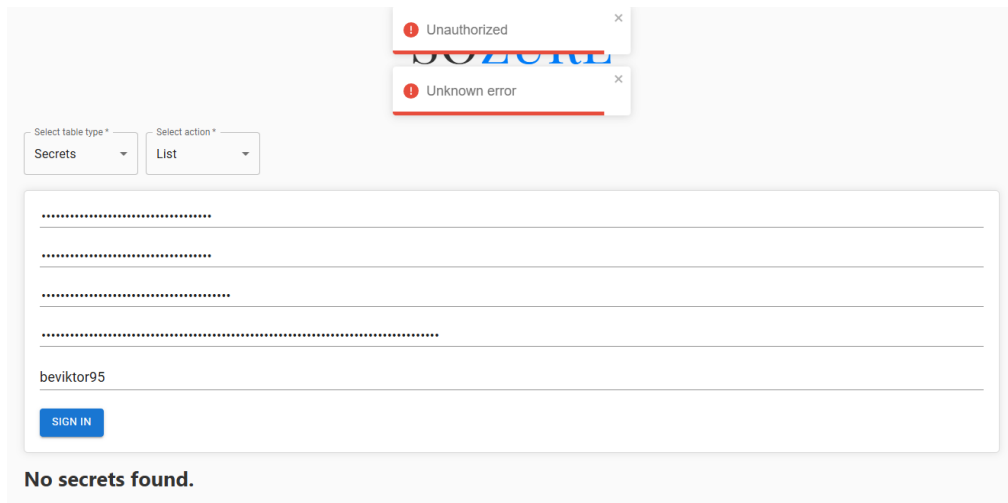
79. ábra: Sikeres bejelentkezés után üdvözlő képernyő (saját szerkesztés)

- Tényleges eredmény:
 - 79. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszteset:

- Elvárt eredmény:
 - o A webkliens nem enged túl a bejelentkező felületen
 - o Hibaüzenet jelenik meg a helytelen bejelentkezési adatok miatt



80. ábra: Sikertelen bejelentkezés (saját szerkesztés)

- Tényleges eredmény:
 - o 80. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Környezeti változó lekérdezés

Sikeres művelet

Tesztet:

- Elvárt eredmény:
 - o Egy találat megjelenítése: ZoomAdapter PROD variable group tartalmazza a környezeti változót

Select table type *
Variable Gro...

Select action *
List

Hi Viktor Bócz-Kemény! To change Azure organization, click here:
[BACK TO AUTHENTICATION](#)
[SEE CHANGES HISTORY](#)

Select Azure project
bkv

*

SearchTestScenarioKey

Value (regex) of variable [OPTIONAL]

☐ Secret included
☐ Key's name should be regex

SEND REQUEST

Matched variables (Found variables: 1)

Project	Variable Group Name	Variable key	Variable value	Operation
bkv	ZoomAdapter PROD	SearchTestScenarioKey	SearchTestScenarioValue	

1-1 of 1

81. ábra: Környezeti változó sikeres lekérdezés, teszteredmény (saját szerkesztés)

- Tényleges eredmény:
 - o 81. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszteset:

- Elvárt eredmény:
 - o Hibaüzenet jelenik meg a lekérdezés sikertelenségéről
 - o Nem jelenik meg a keresés eredménye a felületen

82. ábra: Sikertelen környezeti változó lekérdezés (saját szerkesztés)

- Tényleges eredmény:
 - 82. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel (hibaüzenetben rögzített javaslat félrevezető lehet a felhasználó számára, potenciálisan javítandó működés is lehet).

Környezeti változó manipulálás

Környezeti változó hozzáadás

Sikeres művelet

Tesztetset:

- Elvárt eredmény:
 - Megjelenik az új, hozzáadott környezeti változó az eredménylistában
 - Azure DevOps-ban látható az új környezeti változó

✓ Success

Select table type *
 Variable Gro...

Select action *
 List

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select Azure project
 bkv

Name (regex) of variable group

Key of variable

Value (regex) of variable [OPTIONAL]

☐ Secret included
☐ Key's name should be regex

[SEND REQUEST](#)

Matched variables (Found variables: 1)

Project	Variable Group Name	Variable key	Variable value	Operation
bkv	ZoomAdapter PROD	SearchTestScenarioKey2	SearchTestScenarioValue2	📄 ✎ 🗑️

83. ábra: Sikeres környezeti változó hozzáadás megjelenik a szoftver kezelőfelületén (saját szerkesztés)

b kv
+

- [Overview](#)
- [Boards](#)
- [Repos](#)
- [Pipelines](#)
- [Pipelines](#)
- [Environments](#)
- [Library](#)
- [Test Plans](#)
- [Artifacts](#)

Library > ZoomAdapter PROD

[Save](#)
[Clone](#)
[Security](#)
[Pipeline permissions](#)
[Approvals and checks](#)
[Help](#)

Properties

Variable group name

Description

☐ Link secrets from an Azure key vault as variables ⓘ

Variables

Name ↑	Value	
DatabaseProvider	MsSql	
Logging	Serilog	
NewKafkaBroker	kafka-broker-value	
SearchTestScenarioKey	SearchTestScenarioValue	
SearchTestScenarioKey2	SearchTestScenarioValue2	

84. ábra: Sikeres környezeti változó hozzáadás megjelenik az Azure DevOps felületén is (saját szerkesztés)

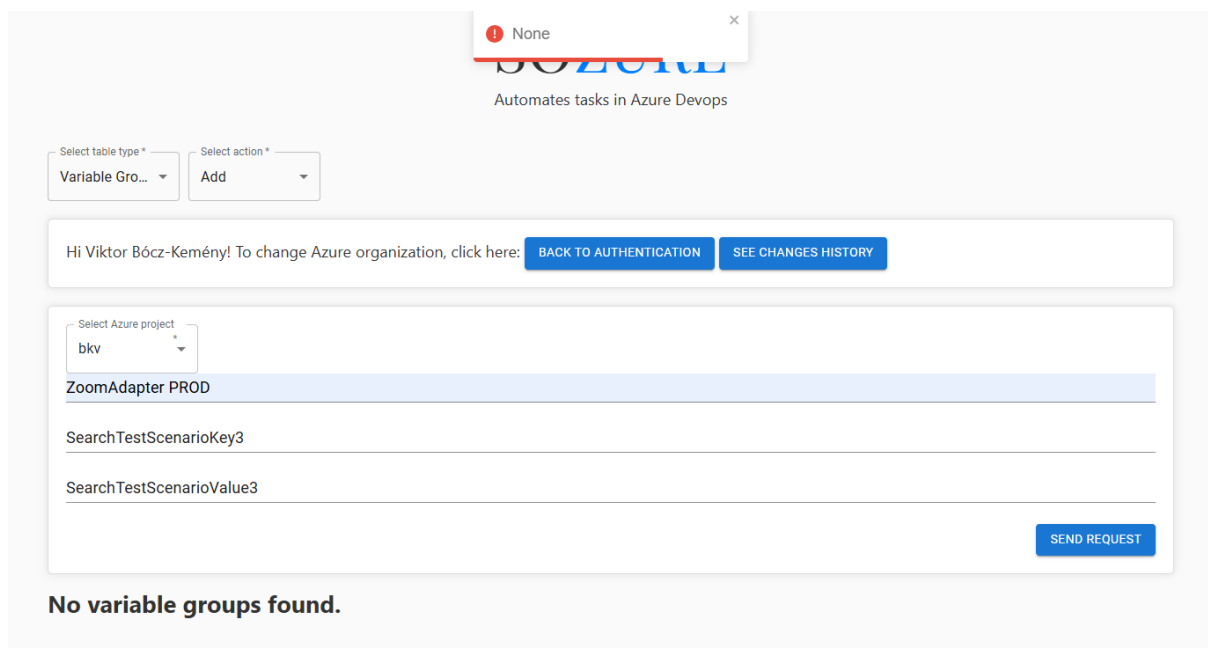
- Tényleges eredmény:

- 83. ábra és 84. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszt eset:

- Elvárt eredmény:
 - Hibaüzenet jelenik meg a hozzáadás sikertelenségéről
 - Nem jelenik meg az új környezeti változó az Azure DevOps felületén



85. ábra: Sikertelen környezeti változó hozzáadás (saját szerkesztés)

- Tényleges eredmény:
 - Variable group állapota nem változott, az állapota megegyezik a 84. ábrán látott állapottal
 - 84. ábra és 85. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Környezeti változó módosítás

Sikeres művelet

Teszt eset:

- Elvárt eredmény:
 - Megjelenik a módosított környezeti változó az eredménylistában

- Azure DevOps-ban látható a környezeti változó új értéke

The screenshot shows the 'Variable Groups' interface in Azure DevOps. At the top, there's a success message: 'Success'. Below it, a notification bar says: 'Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)'. The main form is for editing a variable group named 'bkv'. It has fields for 'Name (regex) of variable group', 'Key of variable', and 'Value (regex) of variable [OPTIONAL]'. There are checkboxes for 'Secret Included' and 'Key's name should be regex'. A 'SEND REQUEST' button is at the bottom right. Below the form, a section titled 'Matched variables (Found variables: 1)' shows a table with one variable:

Project	Variable Group Name	Variable key	Variable value	Operation
	ZoomAdapter PROD	SearchTestScenarioKey2	ModifiedSearchTestScenarioValue2	

At the bottom right of the table, it says '1-1 of 1' with navigation arrows.

86. ábra: Sikeres környezeti változó módosítás (saját szerkesztés)

The screenshot shows the 'Variable Groups' interface in Azure DevOps, specifically the 'ZoomAdapter PROD' variable group. The top navigation bar includes 'Library > ZoomAdapter PROD' and tabs for 'Variable group', 'Save', 'Clone', 'Security', 'Pipeline permissions', 'Approvals and checks', and 'Help'. The 'Properties' section shows the 'Variable group name' as 'ZoomAdapter PROD' and a 'Description' field. There's a toggle switch for 'Link secrets from an Azure key vault as variables'. The 'Variables' section shows a table with five variables:

Name ↑	Value
DatabaseProvider	MsSql
Logging	Serilog
NewKafkaBroker	kafka-broker-value
SearchTestScenarioKey	SearchTestScenarioValue
SearchTestScenarioKey2	ModifiedSearchTestScenarioValue2

At the bottom left, there's a '+ Add' button.

87. ábra: Sikeres környezeti változó módosítás megjelenik az Azure DevOps felületén is (saját szerkesztés)

- Tényleges eredmény:

- 86.ábra és 87. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszteset:

- Elvárt eredmény:
 - Nem jelenik meg egy találat sem a találati listában a műveletjövőahagyási felületen.
 - Nem jelenik meg módosulás a variable group-ban az Azure DevOps felületén.

The screenshot shows the 'Update' action for a 'Variable Group' in the Azure DevOps interface. At the top, there's a message: 'Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION] [SEE CHANGES HISTORY]'. Below this, there's a 'Select Azure project' dropdown menu with 'bkv' selected. The main area displays a list of variables: 'ZoomAdapter PROD', 'SearchTestScenarioKey3', and 'ModifiedSearchTestScenarioValue2'. The 'Value (regex) of variable [OPTIONAL]' field is empty. There's a checkbox for 'Key is regex' which is unchecked. A 'SEND REQUEST' button is at the bottom right. At the bottom of the interface, a message states: 'No variables found.'

88. ábra: Sikertelen környezeti változó módosítás, nincs a paramétereknek megfelelő környezeti változóra találat (saját szerkesztés)

- Tényleges eredmény:
 - Variable group jelenlegi állapotát továbbra is tükrözi a 87. ábra
 - 87. ábra és 88. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Környezeti változó törlés

Sikeres művelet

Teszteset:

- Elvárt eredmény:
 - Sikeres státuszú visszajelzés érkezik a szoftver webfelületéről

- Azure DevOps-ban már nem látható a törölni kívánt környezeti változó

89. ábra: Sikeres környezeti változó törlés (saját szerkesztés)

Name ↑	Value
DatabaseProvider	MsSql
Logging	Serilog
NewKafkaBroker	kafka-broker-value
SearchTestScenarioKey	SearchTestScenarioValue

90. ábra: Sikeres környezeti változó törlés után nem található az Azure DevOps-ban a SearchTestScenarioKey2 környezeti változó (saját szerkesztés)

- Tényleges eredmény:
 - 89. ábra és 90. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszt eset:

- Elvárt eredmény:
 - o Nem jelenik meg egy találat sem a találati listában a műveletjövahagyási felületen.
 - o Nem jelenik meg módosulás a variable group-ban az Azure DevOps felületén.

The screenshot shows the Azure DevOps Variable Group management interface. At the top, there are two dropdown menus: 'Select table type *' with 'Variable Gro...' selected, and 'Select action *' with 'Delete' selected. Below these is a message: 'Hi Viktor Bócz-Kemény! To change Azure organization, click here:' followed by two buttons: 'BACK TO AUTHENTICATION' and 'SEE CHANGES HISTORY'. The main section has a 'Select Azure project' dropdown with 'bkv' selected. Below it is a table with one row: 'ZoomAdapter PROD'. Under the table is a search bar with 'SearchTestScenarioKey2' entered. A 'SEND REQUEST' button is on the right. At the bottom, a message box says 'No variables found.'

91. ábra: Sikertelen környezeti változó törlés, nincs a paramétereknek megfelelő környezeti változóra találat (saját szerkesztés)

- Tényleges eredmény:
 - o Variable group jelenlegi állapotát továbbra is tükrözi a 90. ábra
 - o 90. ábra és 91. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Secret lekérdezés

Sikeres művelet

Teszt eset:

- Elvárt eredmény:
 - o Secret2 és Secret3 secretok megjelenítése a kezelőfelületen

Success
×

Automates tasks in Azure Devops

Select table type *

Select action *

Secrets

List

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#) [SEE CHANGES HISTORY](#)

Select KeyVault *

KeyVault-BKV

.*

☐ Get deleted secrets

SEND REQUEST

Matched secrets (Found secrets: 2).

Key vault	Secret name	Secret value	Operation
KeyVault-BKV	Secret2	Secret2	📄 🗑️
KeyVault-BKV	Secret3	`\$AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...`	📄 🗑️

Page: 1/1

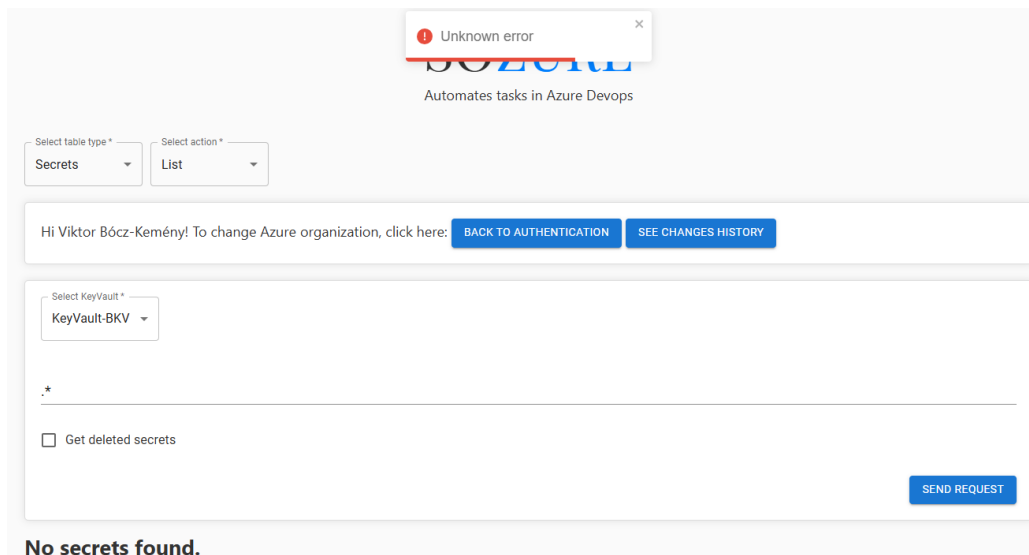
92. ábra: Sikeres secret lekérdezés, eredmény az eredménytáblában megjelenítve (saját szerkesztés)

- Tényleges eredmény:
 - 92. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszt eset:

- Elvárt eredmény:
 - Hibaüzenet jelenik meg a lekérdezés sikertelenségéről
 - Nem jelenik meg egy találat sem a találati listában a műveletjövőahagyási felületen.



93. ábra: Sikertelen secret lekérdezés (saját szerkesztés)

- Tényleges eredmény:
 - 93. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

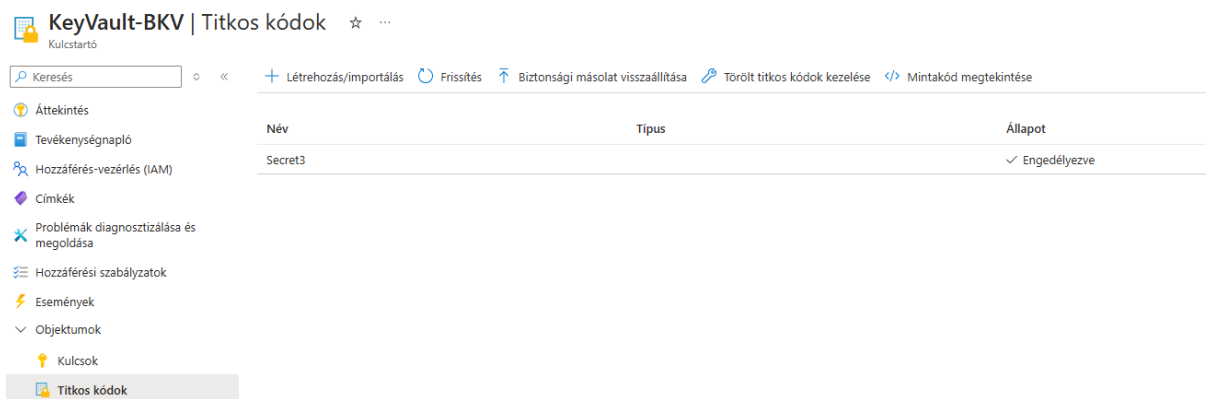
Secret manipulálás

Secret törlés

Sikeres művelet

Tesztetset:

- Elvárt ererdmény:
 - Azure Portal-ban már nem látható a törölni kívánt secret



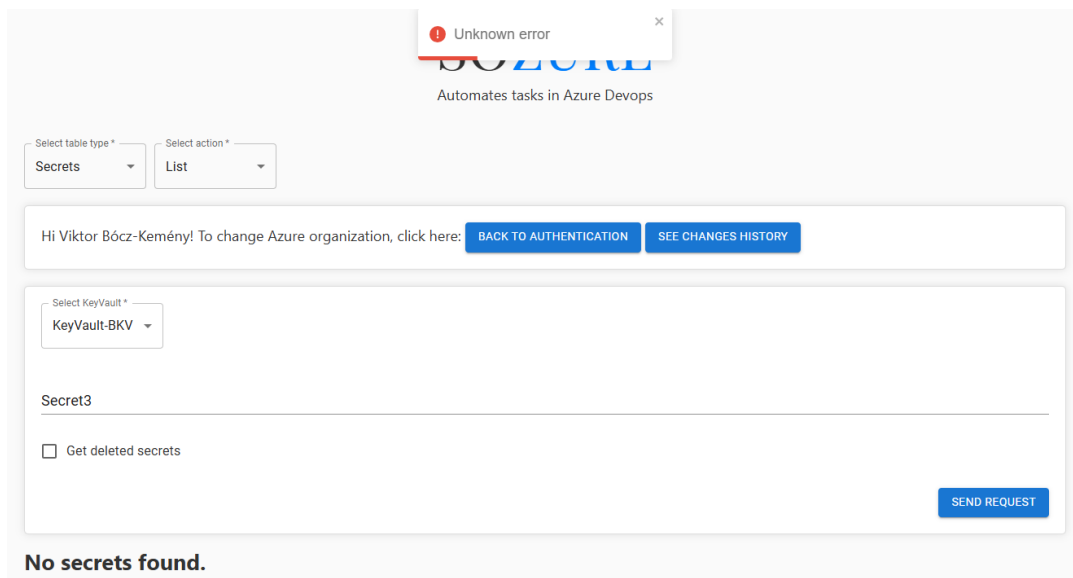
94. ábra: Secret törlés utáni Azure Portal állapot (saját szerkesztés)

- Tényleges eredmény:
 - 94. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Tesztet:

- Elvart eredmény:
 - Hibaüzenet jelenik meg a művelet sikertelenségéről
 - Azure Portal-ban nem törlődik a törölni kívánt secret



95. ábra: Sikertelen secret törlés (saját szerkesztés)

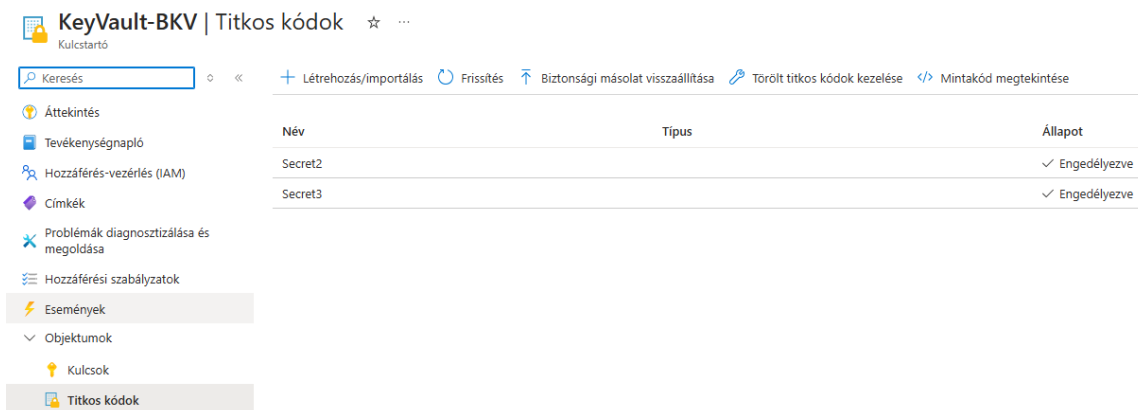
- Tényleges eredmény:
 - Key vault állapota megegyezik a 94. ábrán látottakkal
 - 94. ábra és 95. ábra alapján a tényleges eredmény megegyezik az elvart eredménnyel.

Secret reaktiválás

Sikeres művelet

Tesztet:

- Elvart eredmény:
 - Azure Portal-ban újra látható a reaktivált secret.



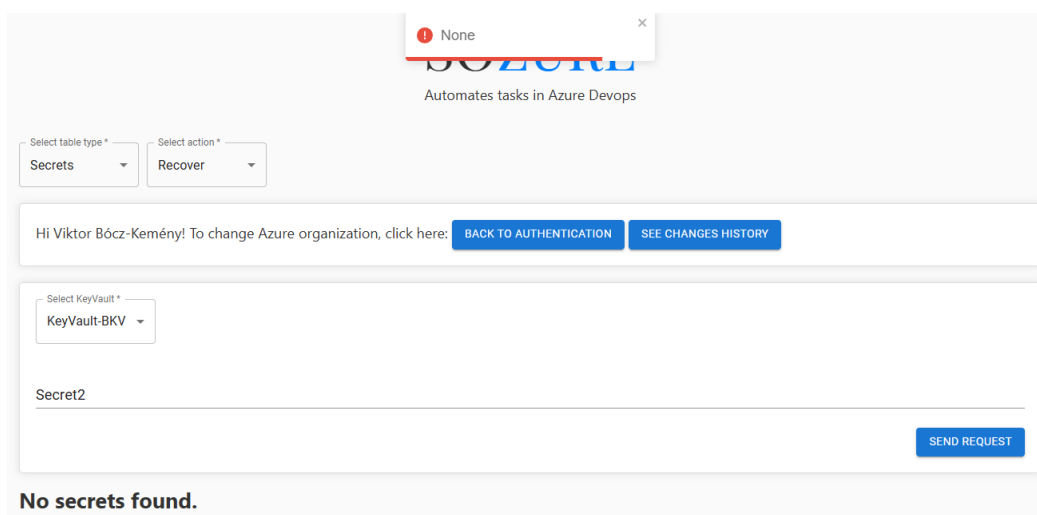
96. ábra: Sikeres reaktíválást követő key vault állapot (saját szerkesztés)

- Tényleges eredmény:
 - 96. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Tesztetset:

- Elvárt ererdmény:
 - Hibaüzenet jelenik meg a művelet sikertelenségéről
 - Azure Portal-ban nem reaktíválódik a törölni kívánt secret



97. ábra: Sikertelen reaktíválási művelet (saját szerkesztés)

- Tényleges eredmény:
 - Key vault állapota megegyezik a 94. ábrán látottakkal
 - 94. ábra és 97. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Key vault másolás

Sikeres művelet

Tesztetek:

- Elvárt eredmény:
 - o KeyVault-BKV key vault secret halmaz tartalmában megegyezik a KeyVaultBV95 key vault secret halmazzal

98. ábra: Sikeres key vault másolás sikerképernyő (saját szerkesztés)

Név	Típus	Állapot
Secret1	Secret	✓ Engedélyezve
Secret2	Secret	✓ Engedélyezve
Secret3	Secret	✓ Engedélyezve

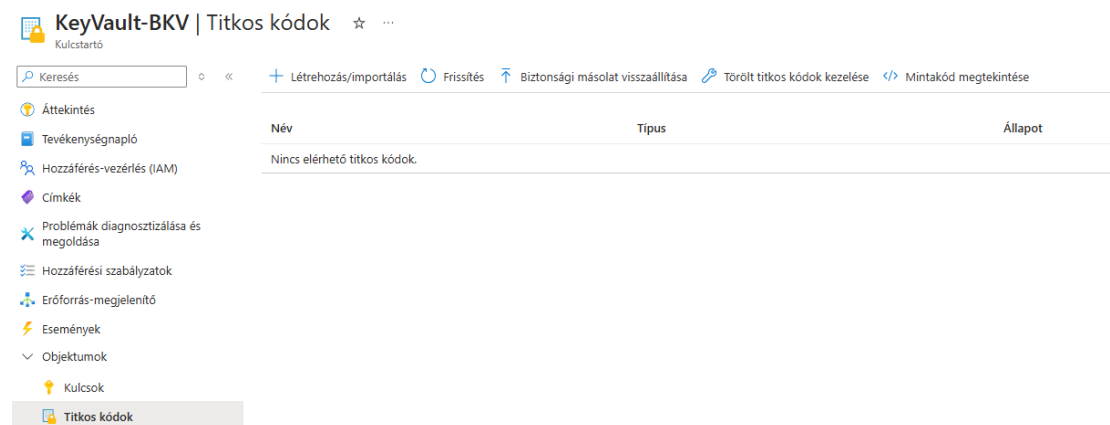
99. ábra: Sikeres key vault másolás utáni Keyvault-BKV key vault állapota (saját szerkesztés)

- Tényleges eredmény:
 - o 98. ábra és 99. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Tesztetek:

- Elvárt eredmény:
 - o Azure Portal-ban nem másolódnak át az átmásolni kívánt secret-ek



100. ábra: Sikertelen key vault másolás eredménye az Azure Portalon (saját szerkesztés)

- Tényleges eredmény:
 - o 100. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Módosítások lekérdezése

Környezeti változó módosítások lekérdezése

Sikeres művelet

Tesztet:

- Elvárt eredmény:
 - o Két hozzáadással kapcsolatos találat megjelenítése webkliensen

SOZURE

Automates tasks in Azure Devops

[BACK TO MAIN](#)

Select modification t...
 Environment...

Select Azure project
 bkv

From date
 03/10/2025

To date
 03/17/2025

Limit *
 10

User

[SEARCH](#)

Records (found 10 elements):

Organization	Project	User	Variable group regex	Key	Operation type	Date
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PROD	SearchTestScenarioKey2	Delete	Mon, 17 Mar 2025 17:58:46 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PROD	SearchTestScenarioKey2	Update	Mon, 17 Mar 2025 17:39:04 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PROD	SearchTestScenarioKey2	Add	Mon, 17 Mar 2025 17:18:36 GMT
bevikt95	bkv	Viktor Bócz-Kemény	.*	NewKafkaBroker	Add	Tue, 11 Mar 2025 17:41:45 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter PREPROD	AdminUser	Update	Tue, 11 Mar 2025 15:25:44 GMT

Page: 1/2

[PREVIOUS](#)
[NEXT](#)

101. ábra: Környezeti változó módosítások naplóbejegyzései, első oldal (saját szerkesztés)

SOZURE

Automates tasks in Azure Devops

[BACK TO MAIN](#)

Select modification t...
 Environment...

Select Azure project
 bkv

From date
 03/10/2025

To date
 03/17/2025

Limit *
 10

User

[SEARCH](#)

Records (found 10 elements):

Organization	Project	User	Variable group regex	Key	Operation type	Date
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Delete	Tue, 11 Mar 2025 15:25:35 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Update	Tue, 11 Mar 2025 15:25:30 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter DEV	Logging	Update	Tue, 11 Mar 2025 15:25:24 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter STAGING	Logging	Update	Tue, 11 Mar 2025 15:25:17 GMT
bevikt95	bkv	Viktor Bócz-Kemény	ZoomAdapter STAGING	Logging	Update	Tue, 11 Mar 2025 15:25:10 GMT

Page: 2/2

[PREVIOUS](#)
[NEXT](#)

102. ábra: Környezeti változó módosítások naplóbejegyzései, második oldal (saját szerkesztés)

- Tényleges eredmény:
 - o 101. ábra és 102. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel, hiszen 2 db hozzáadási művelet található a naplóbejegyzések közt.

Secret módosítások lekérdezése

Sikeres művelet

Teszteteset:

- Elvárt eredmény:
 - o Hét secret módosítással kapcsolatos találat megjelenítése webkliensen

The screenshot shows the SOZURE web application interface. At the top, the logo 'SOZURE' is displayed with the tagline 'Automates tasks in Azure Devops'. Below the logo is a 'BACK TO MAIN' button. The main search area contains several filters: 'Select modification t...' with a dropdown set to 'Secrets', 'Select Azure vault *' with a dropdown set to 'KeyVault-BKV', 'From date' set to '03/10/2025', 'To date' set to '03/17/2025', 'Limit *' set to '10', and a 'User' input field. A 'SEARCH' button is located below these filters. The results section is titled 'Records (found 7 elements):' and contains a table with 5 columns: 'Key vault's name', 'User', 'Secret name regex', 'Operation type', and 'Date'. The table lists 5 records, all from 'Viktor Bócz-Kemény' on 'KeyVault-BKV'. The first record is a 'Delete' operation on 'Secret2' at 'Mon, 17 Mar 2025 18:56:37 GMT'. The second is a 'Recover' operation on 'Secret2' at 'Mon, 17 Mar 2025 18:51:48 GMT'. The third is a 'Delete' operation on 'Secret2' at 'Mon, 17 Mar 2025 18:39:44 GMT'. The fourth is a 'Delete' operation on 'Secret1' at 'Tue, 11 Mar 2025 20:27:27 GMT'. The fifth is a 'Recover' operation on 'Secret1' at 'Tue, 11 Mar 2025 20:26:36 GMT'. Below the table, it says 'Page: 1/2'. On the right side, there are 'PREVIOUS' and 'NEXT' buttons.

103. ábra: Secret módosítások naplóbejegyzései, első oldal (saját szerkesztés)

The screenshot shows the SOZURE web application interface, similar to the previous one. The search filters are the same: 'Secrets', 'KeyVault-BKV', '03/10/2025' to '03/17/2025', '10' limit, and an empty 'User' field. The results section is titled 'Records (found 7 elements):' and contains a table with 5 columns: 'Key vault's name', 'User', 'Secret name regex', 'Operation type', and 'Date'. The table lists 2 records, both from 'Viktor Bócz-Kemény' on 'KeyVault-BKV'. The first record is a 'Delete' operation on 'Secret1' at 'Tue, 11 Mar 2025 20:26:15 GMT'. The second is a 'Delete' operation on 'Secret1' at 'Tue, 11 Mar 2025 13:43:16 GMT'. Below the table, it says 'Page: 2/2'. On the right side, there are 'PREVIOUS' and 'NEXT' buttons.

104. ábra: Secret módosítások naplóbejegyzései, második oldal (saját szerkesztés)

- Tényleges eredmény:
 - o 103. ábra és 104. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel, hiszen 7 db secret módosítással kapcsolatos rekord található a naplóbejegyzések közt.

Key vault másolások lekérdezése

Sikeres művelet

Tesztesetek:

- Elvárt eredmény:
 - 9 db key vault másolással kapcsolatos találat megjelenítése webkliensen

The screenshot shows the SOZURE web application interface. At the top, the logo 'SOZURE' is displayed with the tagline 'Automates tasks in Azure Devops'. Below the logo is a 'BACK TO MAIN' button. The main search area contains a dropdown menu for 'Select modification L...' with 'Key vault co...' selected. Below this are input fields for 'From date' (03/10/2025), 'To date' (03/17/2025), and 'Limit' (10), followed by a 'User' field. A 'SEARCH' button is positioned below these fields. The search results are displayed in a table with the heading 'Records (found 9 elements):'. The table has four columns: 'Original key vault', 'Destination key vault', 'User', and 'Date'. The first five rows of the table are visible, showing copies from 'KeyVaultBV95' to 'KeyVault-BKV' by 'Viktor Bócz-Kemény' on March 17, 2025. At the bottom left, it says 'Page: 1/2', and at the bottom right, there are 'PREVIOUS' and 'NEXT' buttons.

Original key vault	Destination key vault	User	Date
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Mon, 17 Mar 2025 21:51:35 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Mon, 17 Mar 2025 21:50:23 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Mon, 17 Mar 2025 21:50:05 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Mon, 17 Mar 2025 21:49:08 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Mon, 17 Mar 2025 21:48:33 GMT

105. ábra: Key vault másolások naplóbejegyzései, első oldal (saját szerkesztés)

This screenshot shows the second page of the search results in the SOZURE application. The interface is identical to the first page, but the table of results is different. The heading 'Records (found 9 elements):' is still present. The table has the same four columns: 'Original key vault', 'Destination key vault', 'User', and 'Date'. The first four rows of the table are visible, showing copies from 'KeyVault-BKV' to 'KeyVaultBV95' by 'Viktor Bócz-Kemény' on March 11, 2025. At the bottom left, it says 'Page: 2/2', and at the bottom right, there are 'PREVIOUS' and 'NEXT' buttons.

Original key vault	Destination key vault	User	Date
KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	Tue, 11 Mar 2025 20:47:36 GMT
KeyVault-BKV	KeyVaultBV95	Viktor Bócz-Kemény	Tue, 11 Mar 2025 20:47:31 GMT
KeyVaultBV95	KeyVault-BKV	Viktor Bócz-Kemény	Tue, 11 Mar 2025 13:43:52 GMT

106. ábra: Key vault másolások naplóbejegyzései, második oldal (saját szerkesztés)

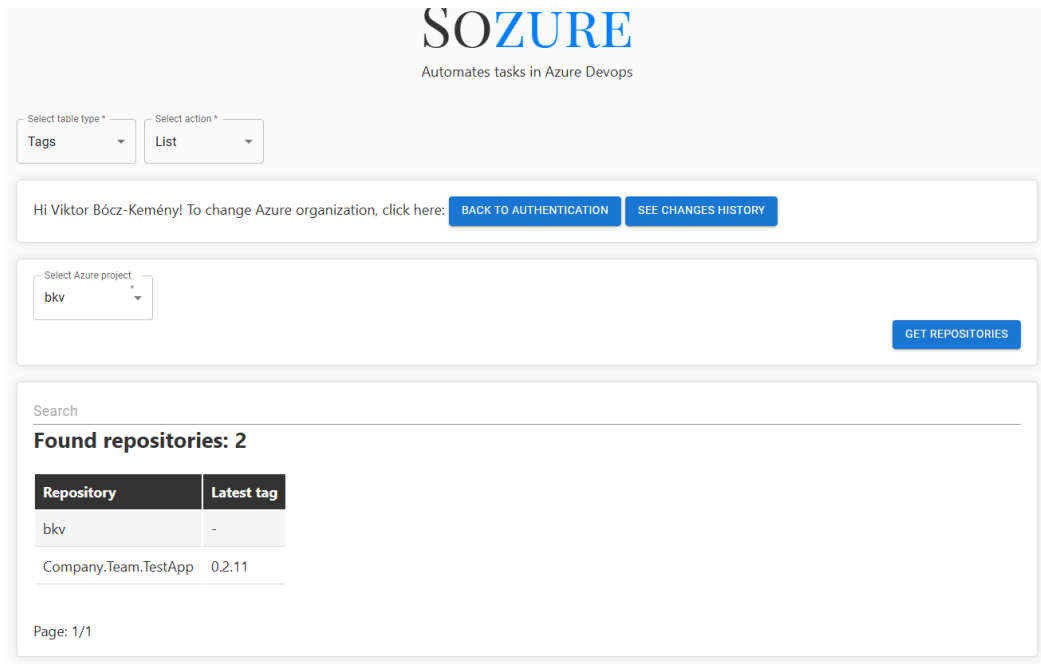
- Tényleges eredmény:
 - 105. ábra és 106. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel, hiszen 9 db key vault másolással kapcsolatos rekord található a naplóbejegyzések közt.

Utolsó szoftververzió lekérdezés

Sikeres művelet

Tesztetset:

- Elvárt eredmény:
 - o 2 elemű lista tér vissza a git repository tematikájú táblázatban eredményként
 - o A bkv nevű repository-nál nincs stabil verzió (tag), a Company.Team.TestApp nevű repository-nál az utolsó stabil verzió a 0.2.11-es



107. ábra: Sikeres utolsó stabil verziók lekérdezése (saját szerkesztés)

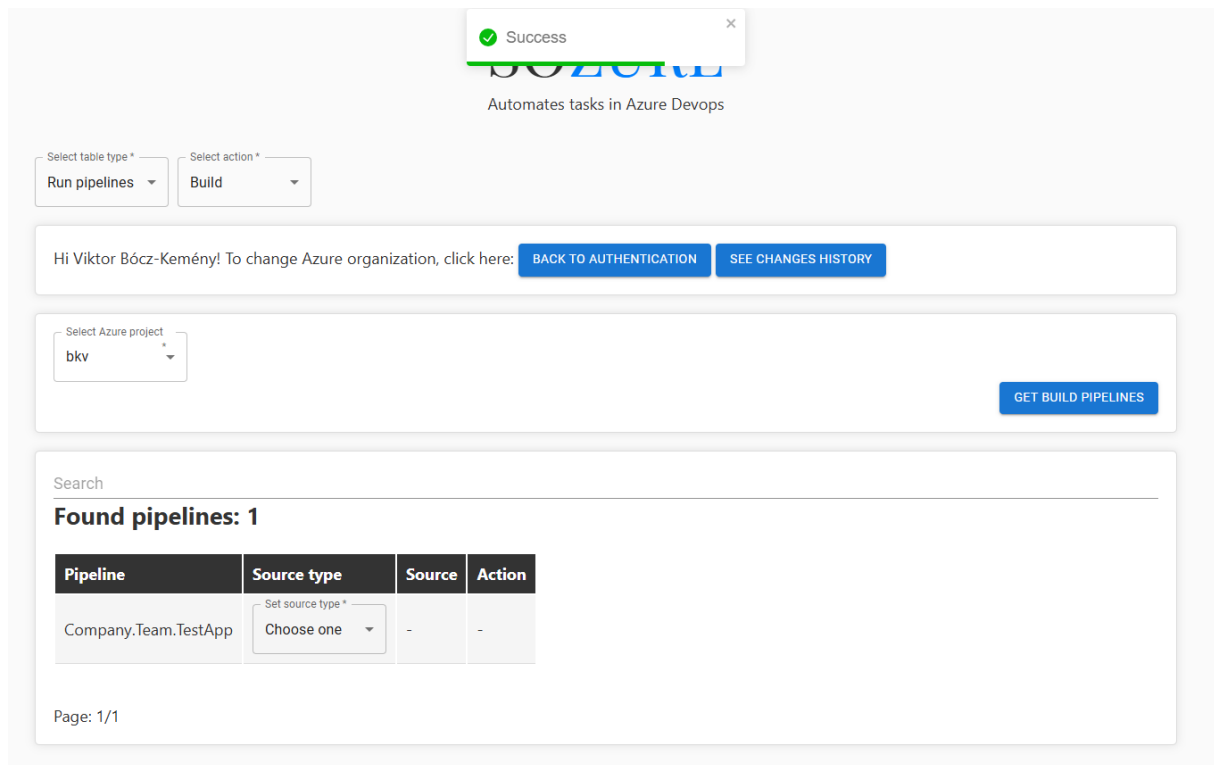
- Tényleges eredmény:
 - o 107. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Pipeline futtatása

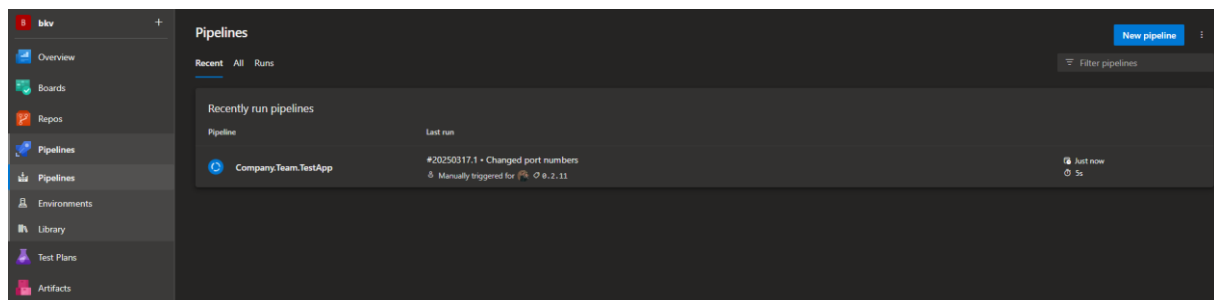
Sikeres művelet

Tesztetset:

- Elvárt eredmény:
 - o Build pipeline-ok kilistázásra kerülnek
 - o A listaelemeken elindítható a pipeline futtatás



108. ábra: Sikeres pipeline futtatás a webkliensen (saját szerkesztés)



109. ábra: Sikeres pipeline futtatás megjelenik az Azure DevOps felületén (saját szerkesztés)

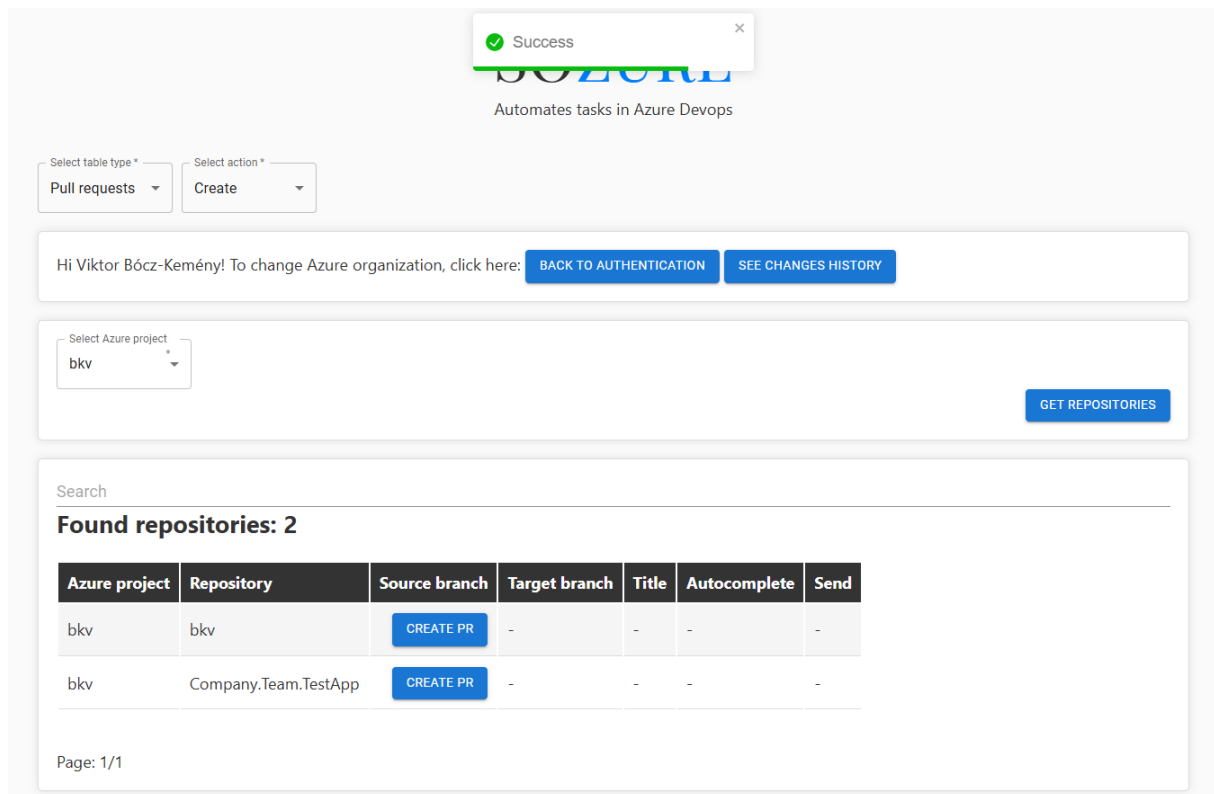
- Tényleges eredmény:
 - o 108. ábra és 109. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Pull request létrehozás

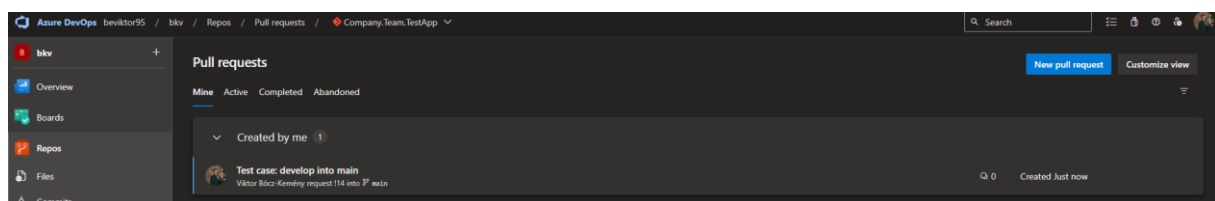
Sikeres művelet

Tesztesetek:

- Elvárt eredmény:
 - o Git repository lista megjelenik a lekérdezés során
 - o Pull request létrehozása sikeres



110. ábra: Sikeres pull request létrehozás webkliensen (saját szerkesztés)



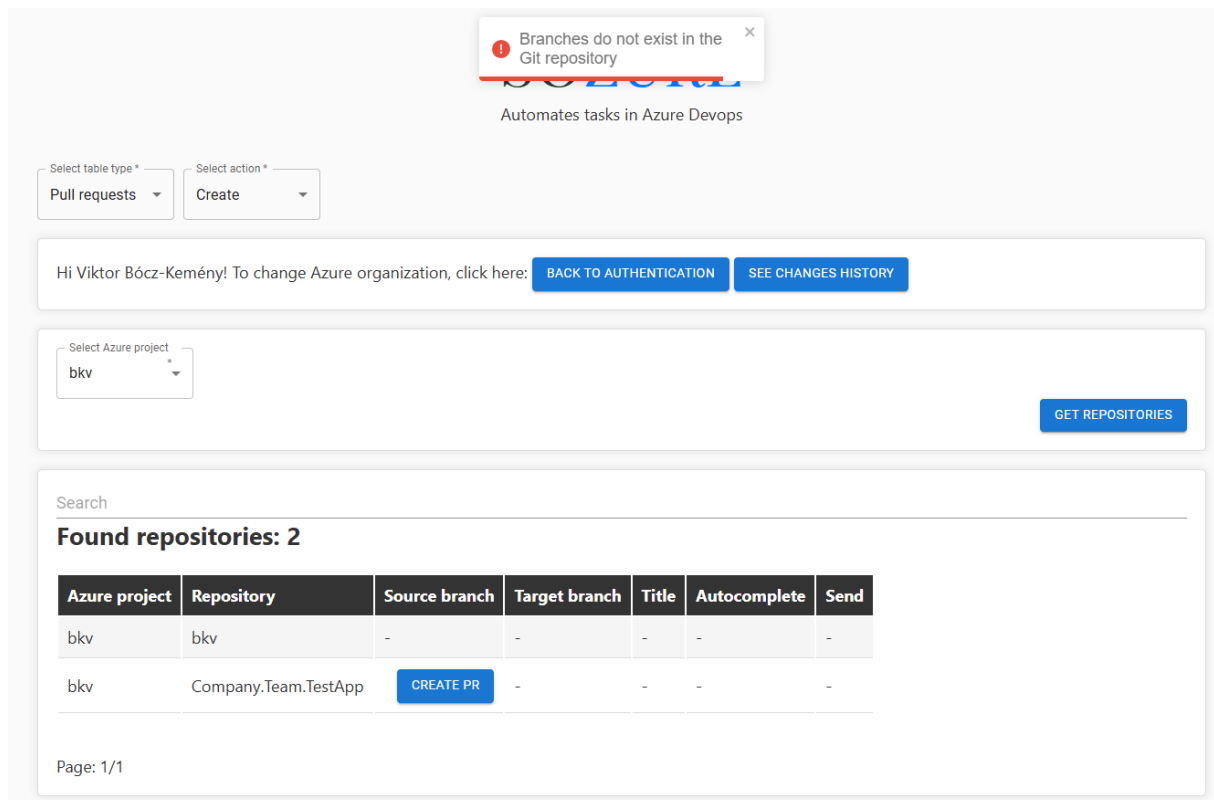
111. ábra: Sikeresen létrehozott pull request megjelenik az Azure DevOps felületén (saját szerkesztés)

- Tényleges eredmény:
 - o 110. ábra és 111. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Sikertelen művelet

Teszt eset:

- Elvárt eredmény:
 - o Hibaüzenet jelenik meg a művelet sikertelenségéről
 - o Azure DevOps-ban nem jön létre a pull request



112. ábra: Sikertelen pull request létrehozás, nem léteznek branch-ek a repository-hoz (saját szerkesztés)

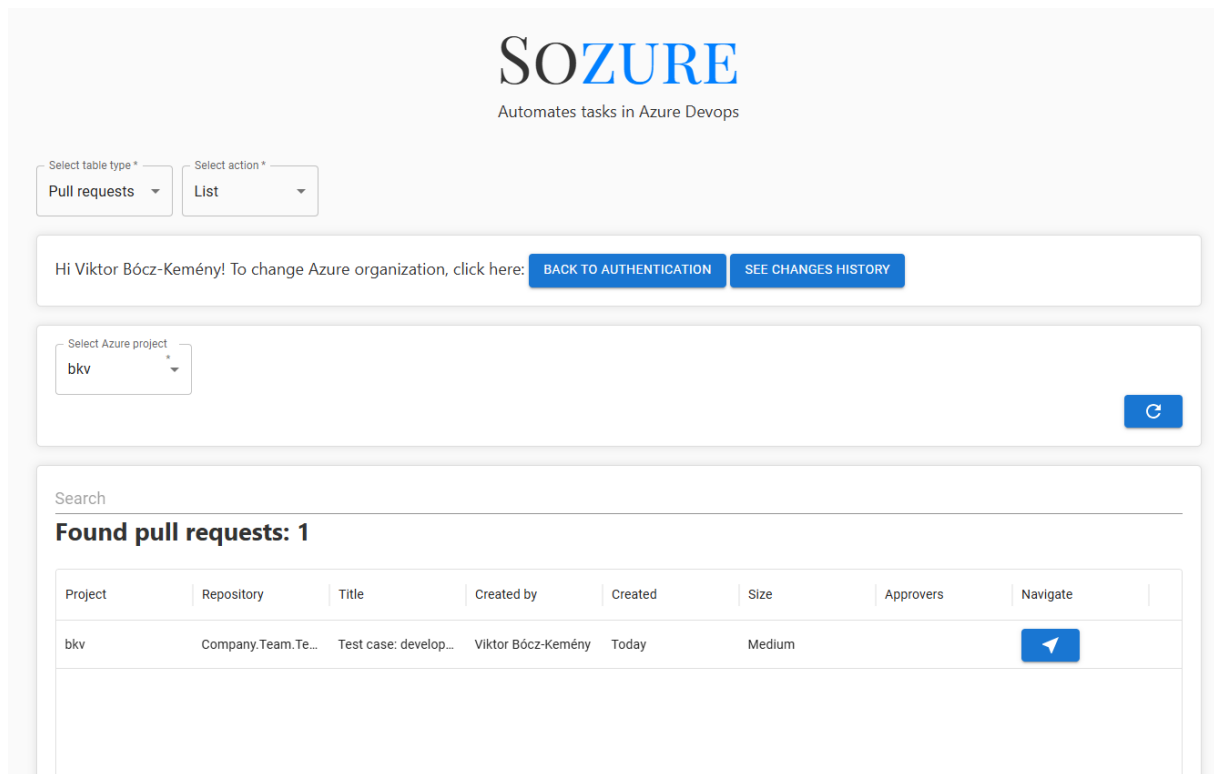
- Tényleges eredmény:
 - o 112. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

Pull request lekérdezés

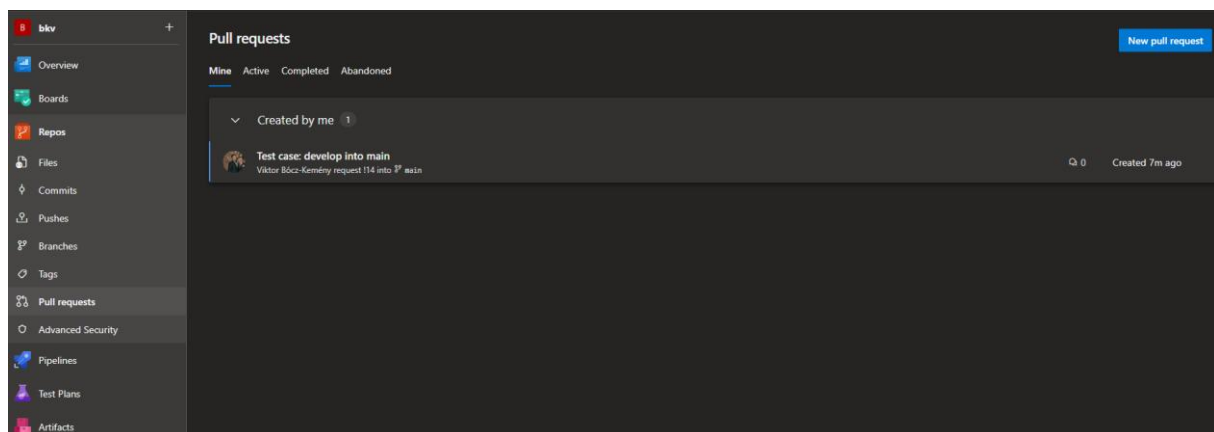
Sikeres művelet

Tesztesetek:

- Elvárt eredmény:
 - o Git pull request lista megjelenik a lekérdezés során



113. ábra: Táblázatba rendezett pull request lista (saját szerkesztés)



114. ábra: Azure Devops-on található pull request lista (saját szerkesztés)

- Tényleges eredmény:
 - o 113. ábra és 114. ábra alapján a tényleges eredmény megegyezik az elvárt eredménnyel.

14. MELLÉKLET

Változómódosítás erőforrásigényének összehasonlítása manuális és automatizált folyamat esetén

Példaeset környezeti változó módosítására

Az Azure projektünkben 172 darab variable group található, ezek között szerepelnek olyan groupok, amelyek a „DatabaseProvider” környezeti változót is tartalmazzák. A példánk esetében előre nem tudjuk megmondani azt, hogy mely variable groupokban található az említett változó.

Mivel a vállalatunk egységesen áll MongoDB használatra, így az a feladatunk, hogy ennek a változónak az értékét egységesen állítsuk be a „MongoDB” értékre a „DatabaseProvider” esetén.

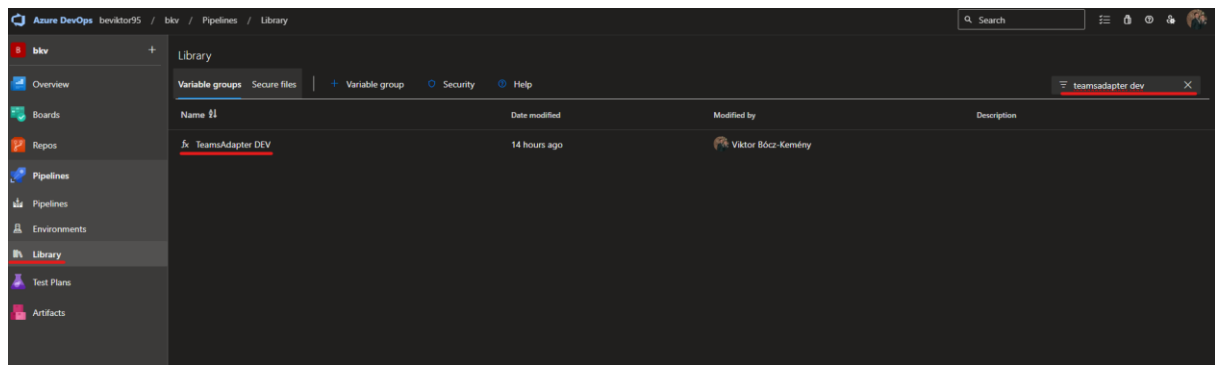
Manuális folyamat

Ebben a fejezetben áttekintem a manuális módosítás lépéseit, illetve a szimulált feladat erőforrásigénye kapcsán is méréseket fogok elvégezni. A mért értékek alapján pedig esztimációt készítek a hasonló típusú feladatok tervezhetősége érdekében.

Lépések

Variable group kikereséses

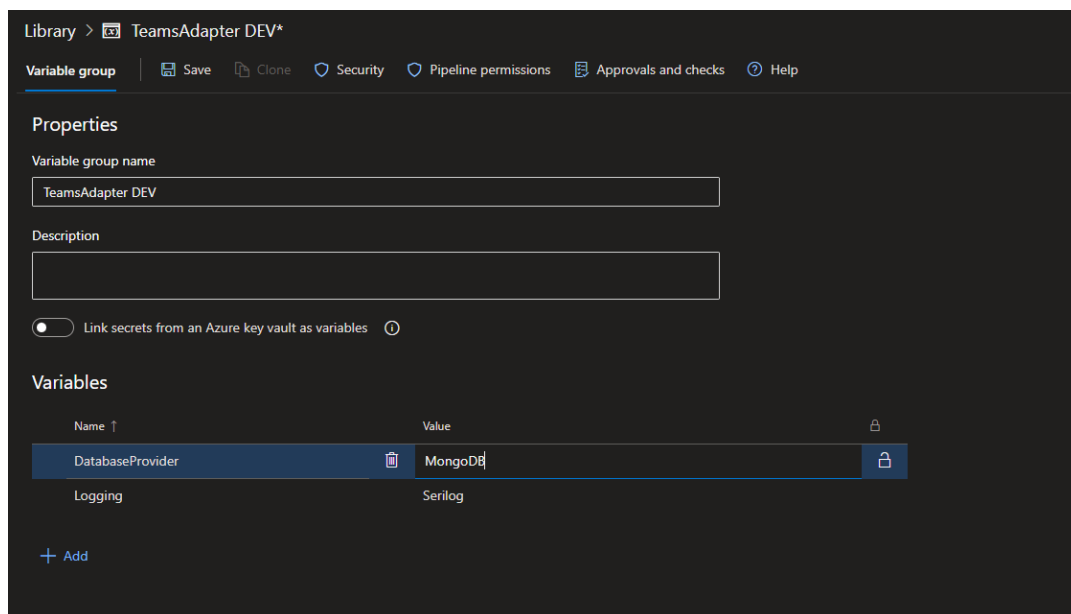
Első lépésként ki kell keressük a Library menüpontban a környezeti változót tartalmazó variable groupot. Ezen a felületen több száz group is megtalálható vállalati környezetben, így a névre szűrés előfeltétele a keresett variable group megtalálásához.



115. ábra: variable group kikeresése Library menüpontban, szűrve a variable group nevére

Környezeti változó módosítása

Miután kiválasztottuk a keresett variable groupot, módosíthatjuk a keresett változó értékét és menthetjük a groupot. Mivel egy groupban tárolhatunk (és vállalati környezetben tárolunk is) akár több tucat környezeti változót is, így a könnyebb kereshetőség érdekében a CTRL + F billentyűparanccsal könnyen megtaláljuk a keresett környezeti változót.



116. ábra: környezeti változó értékének módosítása, majd mentése

Művelet elvégzése alatt mért értékek

A fenti műveletet 20,36 másodperc alatt végeztem el. Az erőforrásigény szempontjából elképzelhető leg pesszimistább forgatókönyv szerint mindegyik group tartalmazza ezt a környezeti változót, így a módosítást mindenhol el kell végezni. Ebben az esetben körülbelül 3502 másodperc szükséges a feladat elvégzéséhez, ami kerekítve 58 percnak felel meg.

Automatizált folyamat

Ebben a fejezetben áttekintem az automatizált módosítás lépéseit, illetve a szimulált feladat erőforrásigénye kapcsán is méréseket fogok elvégezni. A mért értékek alapján pedig esztimációt készítek a hasonló típusú feladatok tervezhetősége érdekében.

Lépések

Bejelentkezés a felületre

A bejelentkezési felületen megadjuk a hitelesítéshez szükséges belépési adatokat (PAT – personal access token, Azure szervezet neve).

117. ábra: bejelentkezés a Sozure felületére personal access token és azure organization név megadással

Megfelelő űrlap kiválasztása és kitöltése

Bejelentkeztetés és hitelesítés után köszönt minket a felület. Ezután kiválasztjuk a módosításhoz szükséges űrlapot, amit ki is töltünk. Az űrlap reguláris kifejezések alapján működik, így eszerint adhatjuk meg a bemeneti paramétereinket.

SOZURE

Automates tasks in Azure Devops

Select table type *
 Variable Groups

VG action type *
 Update

Hi Viktor Bócz-Kemény! To change Azure organization, click here: [BACK TO AUTHENTICATION](#)

Select Azure project *
 bkv

*. *

DatabaseProvider

MongoDB

Value (regex) of variable [OPTIONAL]

☐ Key is regex

[SEND REQUEST](#)

118. ábra: kiválasztjuk a megfelelő űrlapot a legördülő menük segítségével, majd ki is töltjük

Utasítás jóváhagyása, vagy testreszabása

Az utasítás végrehajtása előtt egy összegző felület jelenik meg, ahol adott esetben módosítani, majd véglegesíteni tudjuk a műveletet.

MongoDB

Value (regex) of variable [OPTIONAL]

☐ Key is regex

[SEND REQUEST](#)

Variables to be updated (Total: 8)

Project	Variable group name	Variable key	Variable value	
bkv	TeamsAdapter PROD	DatabaseProvider	PostgreSql	X
bkv	TeamsAdapter PREPROD	DatabaseProvider	PostgreSql	X
bkv	TeamsAdapter DEV	DatabaseProvider	PostgreSql	X
bkv	TeamsAdapter STAGING	DatabaseProvider	PostgreSql	X
bkv	ZoomAdapter PROD	DatabaseProvider	PostgreSql	X

1-5 of 8 < >

Are you sure you want to update variables?

[YES](#)
[NO](#)

119. ábra: Szükség esetén módosítunk, majd jóváhagyjuk/elutasítjuk az elvégezni kívánt műveletet

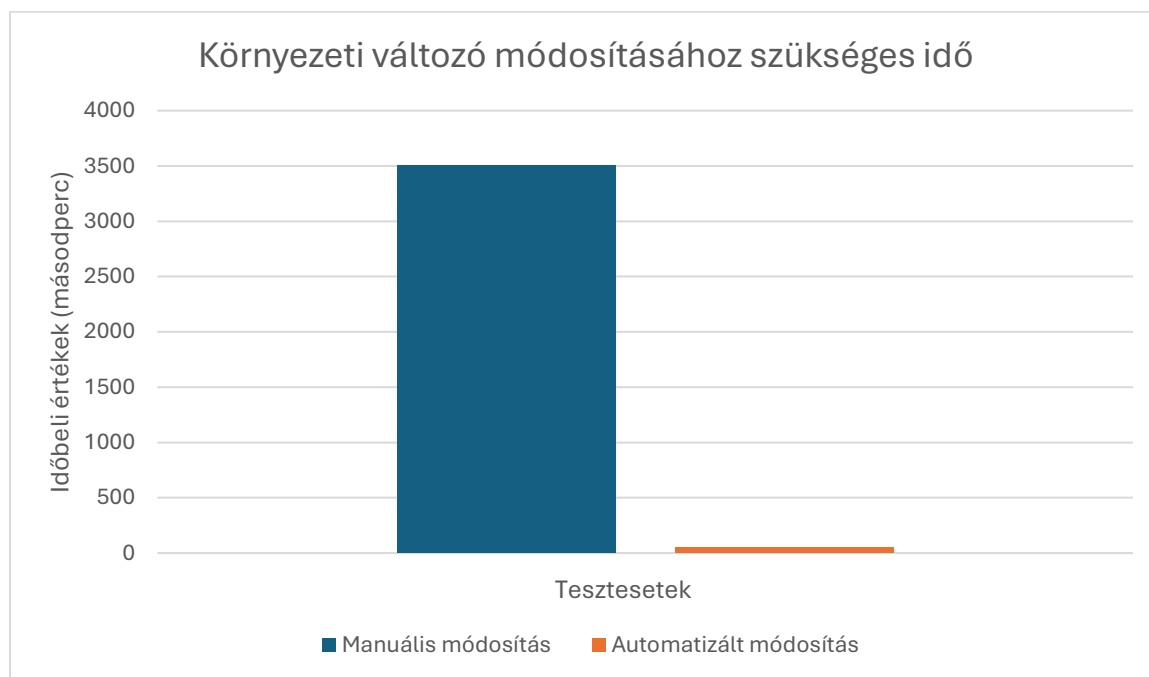
Művelet elvégzése alatt mért értékek

A fenti műveletet 52,46 másodperc alatt végeztem el. Az erőforrásigény szempontjából elképzelhető leg pesszimistább forgatókönyv szerint mindegyik group tartalmazza ezt a környezeti változót, így a módosítást mindenhol el kell végezni. Ez a tényező nem befolyásolja a műveletünk elvégzésének sebességét, így számolhatunk az egy perc alatti végrehajtási időigénnyel.

Összegzés

A fent részletezett példán keresztül bemutattam a manuális és az automatizált változómódosítás közötti erőforrásigény nagyságrendi különbségét.

A két művelet elvégzése között hatvanhét-szeres időbeli különbséget mértem. Ezen módosítások akár napi rendszerességgük is lehetnek, hiszen nagyvállalati ökoszisztémában több tízezer környezeti változó karbantartása is elképzelhető.



A manuális módosítások esetén felmerülhet a módosítást végző munkatárs pontatlansága, illetve a kiégés veszélye is a feladat repetitív jellege miatt. Ennek következtében a hibázási lehetőséggel, illetve az ezen típusú feladatok kollégák közti rotálásával is számolnunk kell a feladat feltervezése esetén.