ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

5. sz. melléklet

# SZAKDOLGOZAT

**OE-NIK** Hallgató neve: **Góg Máté**

**2022** Hallgató törzskönyvi száma: **T/005633/FI12904/N**

Óbudai Egyetem
Neumann János Informatikai Kar
Kiberfizikai Rendszerek Intézet

# SZAKDOLGOZAT FELADATLAP

Hallgató neve: **Góg Máté**
Törzskönyvi száma: T/005633/FI12904/N
Neptun kódja: [redacted]

A dolgozat címe:

**Magyar Államkincstár TÉBA rendszerének Tértivevénnyel kapcsolatos jogerősítés folyamatok kezelése**
**Management of legal confirmation processes related to the Return Certificate of the TÉBA system of the Hungarian State Treasury**

Intézményi konzulens: Rusznák Attila
Külső konzulens:

Beadási határidő: 2022. május 15.

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

**A feladat**

A dolgozat célja, hogy bemutassa a Magyar Államkincstár, Siebel alapú és a FileNet online dokumentumtároló (továbbiakban FileNet) rendszer összeköttetésével megvalósult Országos Nyugdíjbiztosítási Főigazgatóság Családtámogatási főosztály (továbbiakban ONYF) által használt CRM (továbbiakban TÉBA) rendszer Tértivevények jogerősítési folyamatait, illetve azok feldolgozását, üzleti logikájának bemutatását és lefejlesztését webes környezetre. A rendszer alapos megértéséhez végig kell vennünk a FileNet és TÉBA két irányú kommunikációt. Ez megoldható lett volna a Siebel alapokon is, viszont ez rendkívül visszavetette volna a fejlesztés ütemét. Ahhoz, hogy a kapott küldeményeket, fel tudjuk dolgozni, ismernünk kell a központi nyomtatással kinyomtatott, vagy elektronikus tértivevénnyel feladott küldemények készítési, illetve kiküldési folyamatait.

**A dolgozatnak tartalmaznia kell**:

- a tértivevény fogadásának bemutatását,
- a központi nyomtatással kinyomtatott, elektronikus tértivevénnyel feladott küldemények készítési, kiküldési folyamatát,
- az eTértivevény és Tértivevény fejlesztési igényeket,
- a papír alapú kézbesítés kezelését,
- végül az elektronikus kézbesítés kezelését

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

Fleiner R.

.....
Dr. Fleiner Rita
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

Ruscsák AAik

.....
intézményi konzulens

# HALLGATÓI NYILATKOZAT

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

Budapest, 2022.05.09.

Gög Máté

hallgató aláírása

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

9. sz. melléklet

# KONZULTÁCIÓS NAPLÓ

| Hallgató neve: | Neptun kód: | Tagozat: |
|---|---|---|
| Góg Máté | [redacted] | BSc |

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Magyar Államkincstár TÉBA rendszerének Tértivevénnyel kapcsolatos jogerősítési folyamatok kezelése

Szakdolgozat / Diplomamunka[2] címe angolul:

Management of legal confirmation processes related to the Return Certificate of the TÉBA system of the Hungarian State Treasury

| Intézményi konzulens: | Külső konzulens: |
|---|---|
| Rusznák Atilla | ..... |

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2021.09.27. | Témaegyeztetés, feladatkiírás | Rusznák Atilla |
| 2. | 2021.11.15. | Formai követelmények megbeszélése | Rusznák Atilla |
| 3. | 2021.11.29. | Szakdolgoazrészlet vélaményezése | Rusznák Atilla |
| 4. | 2021.12.09. | Írásos beszámoló véleményezése | Rusznák Atilla |

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

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 20 21. december 9.

Rusznák Atilla
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

9. sz. melléklet

# KONZULTÁCIÓS NAPLÓ

| Hallgató neve: | Neptun kód: | Tagozat: |
|---|---|---|
| Góg Máté | [redacted] | BSc |

Telefon: Levelezési cím (pl: lakcím):

[redacted]

Szakdolgozat / Diplomamunka[1] címe magyarul:

Magyar Államkincstár TÉBA rendszerének Tértivevénnyel kapcsolatos jogerősítési folyamatok kezelése

Szakdolgozat / Diplomamunka[2] címe angolul:

Management of legal confirmation processes related to the Return Certificate of the TÉBA system of the Hungarian State Treasury

| Intézményi konzulens: | Külső konzulens: |
|---|---|
| Rusznák Atilla | ..... |

Kérjük, hogy az adatokat nyomtatott nagybetűkkel írja!

| Alk. | Dátum | Tartalom | Aláírás |
|---|---|---|---|
| 1. | 2022.02.10. | Szakdolgozat 1 eredményeinek megbeszélése | Rusznák Atila |
| 2. | 2022.04.15. | Képek számának optimalizálása | Rusznák Atila |
| 3. | 2022.04.29. | Egyéb formai kérdések megvitatása | Rusznák Atila |
| 4. | 2022.05.07. | Utolsó ellenőrzés | Rusznák Atila |

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

A hallgató a Szakdolgozat I. / Szakdolgozat II. (BSc) vagy Diplomamunka 1 / Diplo-mamunka 2 / Diplomamunka 3 / Diplomamunka 4[3] tantárgy követelményét teljesítette, beszámolóra / védésre[4] bocsátható.

Budapest, 2022. 05. 09.

Rusznák Atila
.....
intézményi konzulens

[1] Megfelelő aláhúzandó!
[2] Megfelelő aláhúzandó!
[3] Megfelelő aláhúzandó!
[4] Megfelelő aláhúzandó!

# Tartalomjegyzék

**1. Bevzetés** **3**

**2. A rendszer elméleti működése** **5**
2.1. Papír alapú kézbesítés . . . . . 5
2.1.1. Sikeres kézbesítés . . . . . 6
2.1.2. Sikertelen kézbesítés / 1 . . . . . 6
2.1.3. Sikertelen kézbesítés / 2 . . . . . 7
2.2. Elektronikus kézbesítés . . . . . 8
2.2.1. Sikeres kézbesítés . . . . . 8
2.2.2. Sikertelen kézbesítés . . . . . 9

**3. Fejlesztőeszközök** **10**
3.1. Mi is a CRM? . . . . . 10
3.2. Oracle CRM . . . . . 10
3.2.1. Oracle Siebel CRM alapjai . . . . . 11
3.2.2. Oracle Siebel CRM hátrányai . . . . . 12

**4. Tervezés** **14**
4.1. Mezők bemutatása . . . . . 15
4.2. Ügyintézési határidő számítása általános esetben . . . . . 16
4.3. Ügyintézési határidő számítása egyéb ügyaltípusok esetén . . . . . 16
4.3.1. Otthonfelújítási támogatás ügyaltípus . . . . . 16
4.3.2. KRESZ ügyaltípus . . . . . 17
4.3.3. Nyelvvizsga ügyaltípus . . . . . 17
4.4. Kimenő dokumentumok hatása . . . . . 17
4.4.1. Tényállás tisztázása . . . . . 17
4.4.2. Eljárás felfüggesztés, szünetelés . . . . . 17
4.5. Eljárás lezárása . . . . . 18
4.5.1. Eljárás lezárása kimenő dokumentum alapján . . . . . 18
4.5.2. Eljárás lezárása manuálisan . . . . . 18
4.6. Sablonok importálása . . . . . 19
4.6.1. CSV fájl létrehozása . . . . . 19
4.6.2. .Rtf és .Xlf file-ok . . . . . 20
4.6.3. Sablonfeltöltés működése . . . . . 20
4.6.4. Sablon mezőinek frissítése . . . . . 20
4.6.5. Technikai leírás a tervezett működésről . . . . . 21

**5. Megvalósítás** **22**
5.1. TÉBA környezetek . . . . . 22
5.1.1. DEV környezet bemutatása . . . . . 22

5.1.2. TESZT környezet bemutatása . . . . . 22
5.1.3. PROD környezet bemutatása . . . . . 23
5.2. Fejlesztési folyamatok . . . . . 23

**6. Sablonimportálás fejlesztési folyamatainak bemutatása 24**
6.1. Feldolgozandó adatok rendszerezése . . . . . 24
6.2. Szkript megírása . . . . . 24

**7. Ügyintézési határidőszámítás 29**
7.1. Régebbi kódok revíziója . . . . . 29
7.2. Üzleti logika implementálása . . . . . 29
7.3. Auditálás . . . . . 34
7.4. Tértivevények . . . . . 35
7.4.1. SQL script megírása . . . . . 36
7.4.2. Hosszútávú megoldás implementálása Siebelben . . . . . 36

**8. Tesztelés 39**

**9. Továbbfejlesztési lehetőségek 41**
9.1. Automatizált tesztelés . . . . . 41
9.2. További automatizáció . . . . . 41

**10. Összefoglalás 42**

**11. Summary 43**

# 1. Bevzetés

Szakdolgozatom témája, a Magyar Államkincstár (továbbiakban MÁK), Siebel alapú és a FileNet online dokumentumtároló (továbbiakban FileNet) rendszer összeköttetésével megvalósult Országos Nyugdíjbiztosítási Főigazgatóság Családtámogatási főosztály (továbbiakban ONYF) által használt Customer Relationship Managemant (továbbiakban TÉBA) rendszerben Tértivevények jogerősítései folyamatait, illetve azok feldolgozását, kezelését, üzleti logikájának bemutatását és fejlesztését webes környezeten. A dolgozat főleg a TÉBA rendszerre fog fókuszálni, a FileNet csak a teljesség kedvéért kerül említésre.

A következő fejezetben (2. „A rendszer elméleti működése") részletesen bemutatom, a Tértivevénnyel kapcsolatos jogerősítési folyamatok működését, üzleti logikáját. Továbbá részletezésre fog kerülni a digitalizálásból adódó hasonlóságok különbségek. Ez egy adott, termékben is használt példa felhasználásával kerül részletezésre és bemutatásra.

Az elméleti működés bemutatása után (3. "Fejlesztőeszközök") kerül bemutatásra, az általunk fejlesztéshez használt, TÉBA rendszer mögött megbúvó Siebel CRM környezetet, annak előnyeit, hátrányait, komplexitását elemezve és összegezve. Többek között részletezve lesz az adatstruktúra felépítése, az erre épülő Backend és Frontend. Szó lesz ezen felül az eScriptről, Visual Basicről, amik a környezet kódolási nyelvei, illetve a Siebel által nyújtott Backend tulajdonságairól és felépítéséről.

Az ezt követő fejezetben (4. "Tervezés") végig vesszük a megrendelő által felállított követelményeket, az üzleti logikát. Ez a rész egy fejlesztési dokumentációhoz lesz hasonló tartalmában. Összeszedjük az elvárt működést lépésről lépésre, kellően részletesen, de a megrendelő által is érthető nyelven megfogalmazva, folyamatábrákkal tarkítva.

Ötödik pontban (5. "Megvalósítás") részletezni fogom, a fejlesztés fázisait. Bemutatásra fog kerülni a projekten belüli fejlesztési folyamatok és azokat a lépcsőket, amelyeken végig kell mennie egy-egy prejektnek, hogy kikerülhessen az éles környezetre

Hatodik pontban (6. "Sablonimportálás fejlesztési folyamatainak bemutatása") a sablonimportálás fejlesztési folyamatát fogom részletezni. A fejezetben bemutatásra kerül a fejlesztések bemutatása, a logika részletezése, és az ez alapján megírt kódok bemutatása szöveseges formában.

Hetedik fejezetben (7. "Ügyintézési határidőszámítás") bemutatásra kerül, a fejlesztés fő része, az Ügyintézési határidőszámítás fejlesztésének bemutatása. A részletezés során időrendben fogok haladni, az ügyek és eljárások létrejöttétől, az eljárs futása alatt előjövő különböző változtatások és komplikációkon át, az eljárás lezárásáig. A végén bemutatásra

kerül, hogy hogy a régi ügyekl esetén az integráció, hogy került megoldásra.

Nyolcadik pontban (8. ”Folyamatok tesztelése”) pedig az eddigi részletezett pontok megvalósításának végtermékének tesztelését fogom végezni, amit Tesztelési jegyzőkönyv stílusban fogok megvalósítani. Tartalmazni fog egy megfelelően részletes és szeleskörű esethalmazt.

Kilencedik és egyben az utolsó előtti tartalmi pontban (9. ”Továbbfejlesztési lehetőségek”) bemutatásra kerülnek az általam észrevett továbbfejlesztési lehetőségek. Ezek vonatkozni fognak az általam kilakított részekre, illetve az egész TÉBA rendszerre is.

Tizedik és tizenegyedik pontban (10. ”Összefoglalás”, 11. ”Summary”), pedig a szakdolgozatomat fogom összegezni magyarul és angolul is. Kiemelem a tervezés és megvalósítás szempontjából a számomra legérdekesebb és legnagyobb kihívást jelentő részeket.

A szakdolgozat végén található ”Függelékek” rész alatt lesznek találhatóak a fejlesztéshez szükséges, azonban az üzleti logika megértéséhez nem szükséges adatok és kódok. A feltűntetett kódrészletek kerültek ki végül az éles környezetre.

# 2. A rendszer elméleti működése

A TÉBA nem áll közvetlen kapcsolatban a központi postai rendszerrel, hanem a FileNet nevű alkalmazáson keresztül csatlakoznak egymáshoz, ezért a későbbi dokumentumok kezelése is ezen a programon keresztül történik meg. Miután a TÉBA átadja az iratot, illetve a hozzá tartozó metaadatokat a FileNet-nek, az visszaküld a TÉBA irányába bizonyos státuszokat, ami tartalmazza az irat nyomtatásának, kiküldésének és kézbesítésének állapotáról szóló adatokat.

A FileNet-ből érkező, nyomtatásra vonatkozó státuszváltozások egyrészt a TÉBA-ban generált ”Kimenő dokumentum” típusú feladathoz kapcsolódó ”Változáskövetés” nézeten láthatóak, másrészt pedig a dokumentum állapota is változik ezen beérkező információk alapján. A nyomtatásra vonatkozó kimenő dokumentum állapotok lehetnek ”Nyomtatás alatt”, illetve ”Nyomtatva”

A kiküldésre vonatkozó státuszok, a nyomtatásra vonatkozókhoz hasonlóan, a kiküldéssel kapcsolatos státuszváltozások is megtekinthetőek a változáskövetés nézeten, illetve a kimenő dokumentum állapotváltozásain keresztül is. Állapota lehet üres (null érték), vagy ”Kiküldve” értékű is.

A korábban kiküldésre került küldemények esetén a kézbesítési értesítések hatására létre-hozásra kerül adott ügy alatt egy Bejövő dokumentum típusú feladat FileNet Tértivevény típussal, ami becsatolásra kerül a kapcsolódó kimenő dokumentum típusú feladathoz is. Amennyiben egy irat kapcsán kézbesítés státusz üzenet érkezik a TÉBA-ba, a státuszban szereplő adatok alapján létrehozásra kerül egy FileNet Tértivevény típusú bejövő dokumentum feladat, melyet a kapcsolódó Kimenő dokumentum feladat alatt is lehetősége van megtekinteni a felhasználóknak, a következő adatokkal együtt, ami az ”Átvétel módja”, ”Átvétel dátuma” és a ”Véglegessé válás” dátuma.

## 2.1. Papír alapú kézbesítés

A Magyar Posta Zrt. által átadott, és a Filenetben letárt adatállományban szereplő kézbesítési értékeket érdemes 3 különböző kategóriába csoportosítani, mert egy-egy kategórián belül eltérő kezelés indokolt.

### 2.1.1. Sikeres kézbesítés

Ha az e-tértivevény a következő státuszokkal érkezik vissza:

- Kézbesítve címzettnek
- Kézbesítve meghatalmazottnak
- Kézbesítve gondnoknak/gyámnak

akkor a TÉBA-ban a következő információkat kell látnunk:

- ”Sikeres kézbesítés” jelzés,
- ”tértivevény átvétele”,
- ”véglegessé válás dátuma”

A TÉBA az átvétel dátuma alapján automatikusan beállítja a véglegessé válás dátumát, az ügy állapota pedig automatikusan „Kész” -re módosul. A csatolt e-tértivevény képi megjelenítése a következő jogcímeket tartalmazza: 1 Címzett, 2 Meghatalmazott, 3 Helyettes átvevő, 4 Közvetett kézbesítő.

### 2.1.2. Sikertelen kézbesítés / 1

A sikertelen kézbesítés eseteit 2 kategóriába rendezhetjük a szerint, hogy az általános közigazgatási rendtartásról (későbbiekben Ákr.) fűz-e hozzá jogkövetkezményt vagy sem. Ebben a 2-es pontban azok szerepelnek, amelyekhez fűz. Ilyennek minősül:

- Átvétel megtagadva
- Küldeményt nem kereste
- Elköltözött
- Ismeretlen helyre költözött

Ilyenkor a TÉBA-ban szerintem a következő információkat kell látnunk:

- ”Sikertelen kézbesítés” jelzés,

- a sikertelen kézbesítés ezen négy jogcíme közül a megfelelő megjelenítése,
- kézbesítési fikció alapjául szolgáló esemény dátuma (ide kerül be a tértivevényen szereplő azon dátum, amihez a következő pont szerint dátumot automatikusan be kell állítani),
- véglegessé válás dátuma, ami az egyes jogcímek alapján eltérő szabály szerint:
  - ”Átvétel megtagadva” esetén a véglegessé válás dátuma megegyezik azzal a nappal, ami a tértivevényen szerepel,
  - ”Küldeményt nem kereste” esetén a véglegessé válás napja az e-térti szerinti „2. értesítőt elhelyeztem” mezőben szereplő nap plusz öt munkanap,
  - ”Elköltözött” és „Ismeretlen helyre költözött” esetén a véglegessé válás napja az e-térti szerinti dátum plusz öt munkanap.

### 2.1.3. Sikertelen kézbesítés / 2

Amennyiben az Ákr. nemfűz hozzá jogkövetkezményt és a következő információkat látjuk a TÉBA rendszerben:

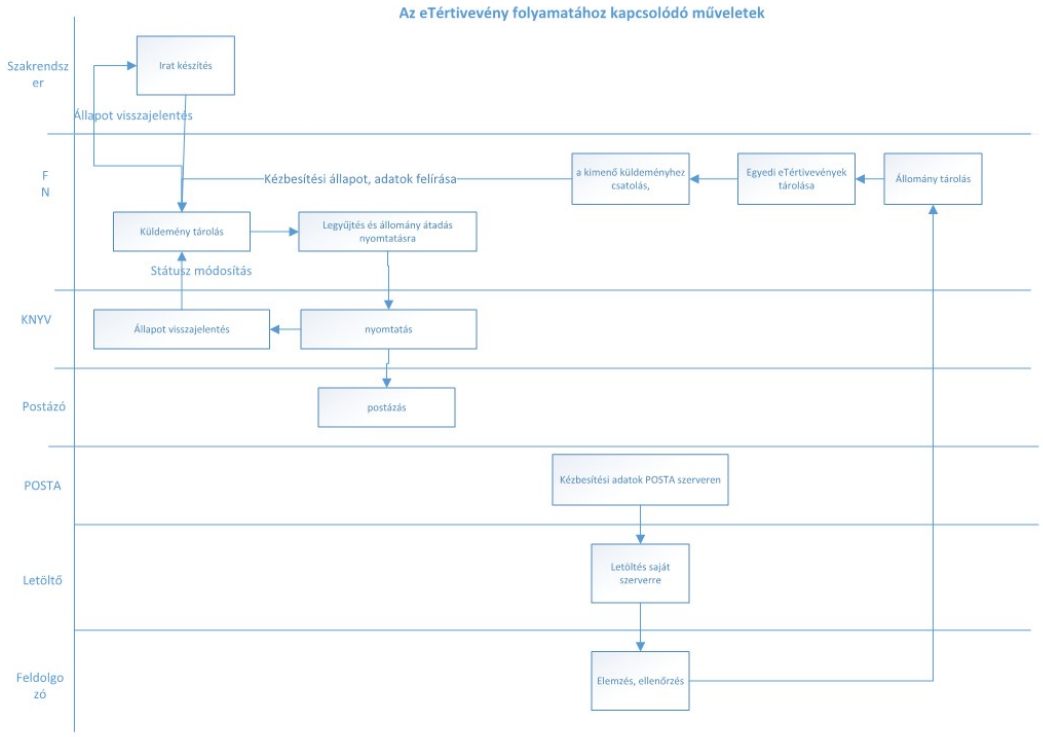
- ”Nem kézbesíthető” (Az e-tértivevény képi megjelenítésén ilyen nem szerepel, ezt a fogalmat a 335/2012. Kormány rendelet sem ismerni, és egyébként ez egy gyűjtőkategória, amiben több másik jogcím is benne.)
- ”Meghalt” (Az e-tértivevény képi megjelenítése és a 335/2012. Kormány rendelet szerinti megfogalmazás a „Bejelentve: meghalt/megszűnt.)
- ”Cím nem azonosítható” (Ez ugyanígy szerepel a képi megjelenésen és a rendeletben is; a fogalom-meghatározás szerint: a küldemény címzése vagy címe nem megfelelő, vagy a cím nem létező, továbbá, ha a címhely azonosításra nem alkalmas, vagy az nem egyértelmű.)
- ”Kézbesítés akadályozott” (Ez szintén ugyanígy szerepel mindenhol. fogalom-meghatározása: a levélszekrénybe történő elhelyezés, személyes átadással történő kézbesítés vagy az értesítő hátrahagyása nem lehetséges.)

Ezekre az esetekre nézve az Ákr. kézbesítési fikciót nem állít fel. Itt lényegében olyan meghiúsult kézbesítésekről van szó, ahol az e-tértivevény visszaérkezését követően nem alkalmazható az automatikus „Kész” -re állítás, hanem egy feladatot kell generálni az ügyintézőnek. A „Meghalt” jelzés esetén megeshet, hogy eddig nem tudtunk a halálesetről,

ezért utána kell járni, létezi-e halotti anyakönyvi kivonat az adott halálesetről, amennyiben igen, annak az esetlegesen már teljesített kifizetésre is hatása lehet (vagy éppen vissza fog érkezni az összeg is). A többi esetben pedig valami hiba csúszott a címzésbe, és azt javítva, a küldeményt ismét kiküldve lehet orvosolni. Ezek az esetek előfordulási valószínűségé rendkívül alacsony, de szükséges a kezelésük.

## 2.2. Elektronikus kézbesítés

Az elektronikus kézbesítésnél jóval kevesebb esetet kell kezelni, azonban itt sem elegendő, ha csak a sikeres kézbesítést jelenítjük meg a TÉBA rendszerben.




1. ábra. eTértivevény folyamatokhoz kapcsolódó műveletek

[1]

### 2.2.1. Sikeres kézbesítés

A hivatalos elérhetőségre kézbesített küldemény kézbesítettnek minősül, amennyiben a hivatalos elérhetőséget biztosító szolgáltató a küldemény ügyfél által történő átvételét igazolja vissza, az igazolásban feltüntetett időpontban. Ezesetben a TÉBA rendszerben a következő adatoknak kell látszódnia, amik a „Sikeres kézbesítés” jelzés, kézbesítés

dátuma, véglegessé válás dátuma, ami az elektronikus kézbesítés végett megegyezik a kézbesítés dátumával.

### 2.2.2. Sikertelen kézbesítés

Átvétel megtagadva jelzés esetén, a szolgáltatótól visszajön a megtagadásra vonatkozó igazolás, és a küldemény kézbesítettnek minősül az ezen igazolásban feltüntetett időpontban.

Amennyiben a hivatalos elérhetőséget biztosító szolgáltató azt igazolja vissza, hogy a küldeményt a címzett kétszeri értesítése ellenére nem vette át, a második értesítés igazolásban feltüntetett időpontját követő ötödik munkanapon a kézbesítés sikertelen lesz.

# 3. Fejlesztőeszközök

A következő fejezetben bemutatásra kerül az Oracle Customer Relationship Management (CRM) megoldása, a Sibel rendszer és annak felépítése. Az eScript bemutatása, alap szintaktikák ismertetése. Ezen felül részletezve lesznek a rendszer erősségei és a gyengeségei is.

## 3.1. Mi is a CRM?

Magyarul az „Ügyfélkapcsolat-kezelés" rövidítése, ami lehetővé teszi a cégek számára, hogy egy központi helyről, egyszerűen kezelhessék az üzleti kapcsolataikat ügyfeleikkel, illetve elérhessék az ehhez szükséges ügyféladatokat és információkat . Segít, hogy az összes fiókinformációhoz, a leadekhez, az értékesítési lehetőségekhez és sok egyéb adathoz hozzáférjen a cég, akar több munkatársa is egyszerre, így segítve elő az ügyfelek kategorizálását és az értékes információk kinyerését.

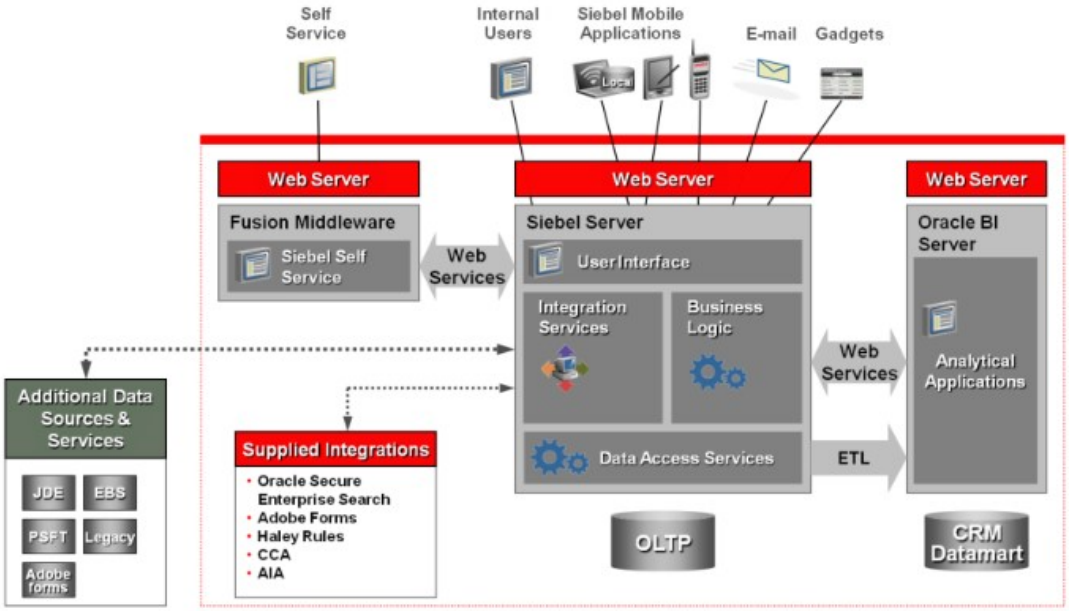
A szakdolgozatom, az átlagtól eltérő CRM rendszer kiegészítéséről fog szólni, mivel az Államkincstár működése nem piaci verseny alapú, és versenytárs hiányában egyedi igényeknek és törvényeknek kell megfelelni. Ezen kívül a működési igények is eltérnek, sokkal szélleskörűbb logika feladatok megalkotásra volt szükség, hogy a rendszer működőképes legyen.
[2][3]

## 3.2. Oracle CRM

A 1990-es években adták ki, így mondhatni az egyik, ha nem a legrégebbi CRM megoldás a világon. Mára elvesztette a piacvezetői szerepét, de kiforrottsága és nagyvállalati használatra való tervezése miatt, a mai napig használt megoldás a piacon. Egyik legnagyobb előnye a nagymennyiségű adatok kezelése.

Ezen indokok miatt tökéletes választás a projekt megvalósítására, hiszen a Magyar Állam-kincstár rendszerében az ország lakosságának adatait tartalmazza, ami bőven kimeríti a nagymennyiségű adat fogalmát.

Tekinthetünk a Oracle Siebel megoldására mint egy „Integrated Stack Selection”-re, ami, a „Best of Breed Selection”-nel szemben egy olyan szolgáltató, ami minden vagy a legtöbb igényhez biztosítja az eszközt, ami külön-külön nem lenne a legjobb megoldás, viszont mint összesség, sokkal jobban optimalizált, és az integrációs problémák sem lassítják a fejlesztést.




2. ábra. Siebel multi-tier architektúrája

[4] [5]

### 3.2.1. Oracle Siebel CRM alapjai

Az adatbázistól, a backenden át a frontendig, a Siebel Tools az, ami hozzáférést biztosít számunkra, mint fejlesztői környezet. Itt tudjuk a fejlesztéshez szükséges részeket, mint Siebel Objektumokat kezelni.

Adatbázisnak az Oracle SQL Developer lett választva. Részben a kompatibilitás, részben pedig a nagyvállalati felhasználás miatt. A létrehozott relációs táblákat, Business Komponensekbe rendezzük, amiket Business Objektumokba csoportosítunk. Ezekre a komponensekre épülnek az Appletek, amik a különböző adatvizualizációkat szolgáltatják. A megjelenítendő Appleteket, nézetekbe (View) rendezzük, amit hozzáadunk a kívánt ablakhoz (Screen).

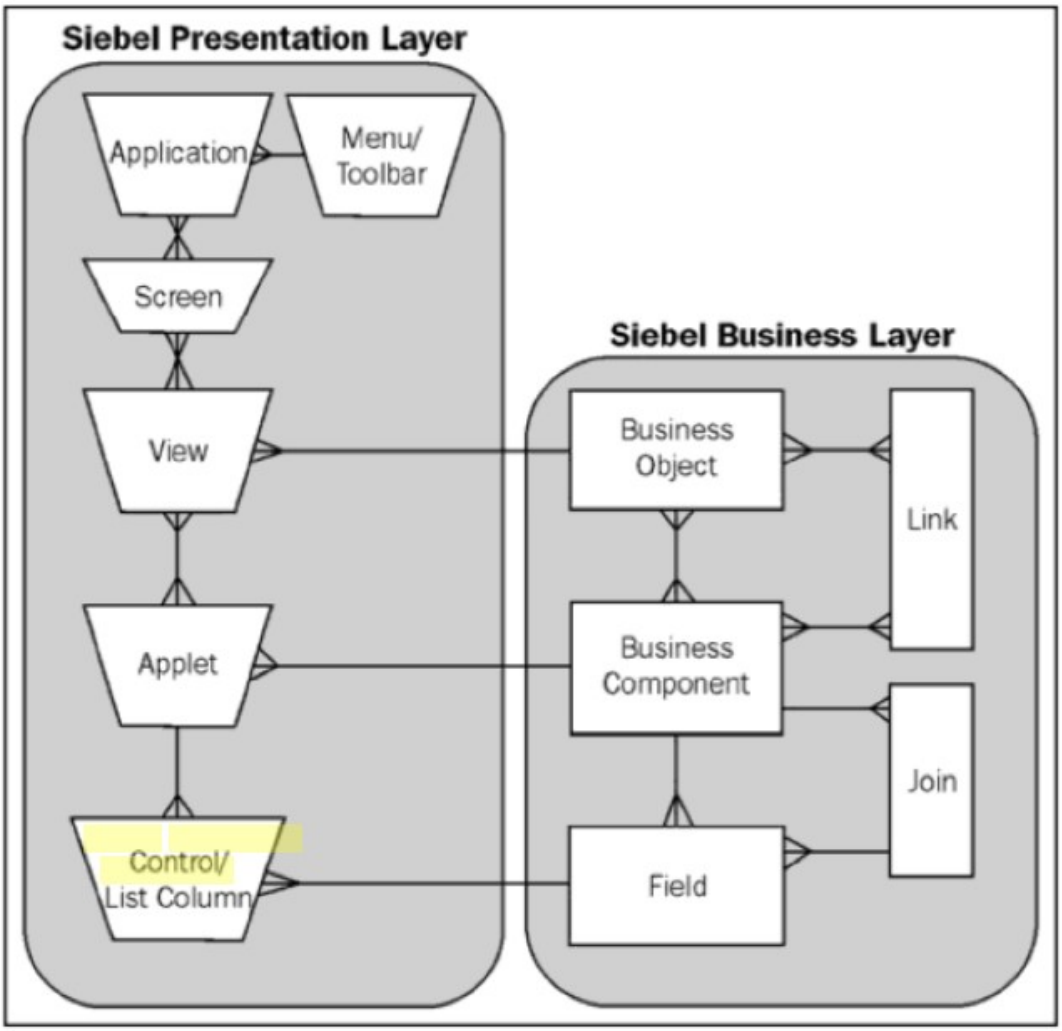
Ezekhez a Siebel Objektumokhoz létre lehet hozni különböző scripteket, az üzleti logika lekódolásához. A scriptek túlnyomó többsége a Business Komponensbe, illetve a Business Service-be kerül, utóbbi egy olyan kódrészlet szokott lenni, amit akár több Komponensben is felhasználunk, így a Clean Code-ra törekvés miatt oda írjuk ezeket a részeket.

Ezek a kódok íródhatnak Visual Basic-ben, vagy eScript nyelven. Mára már az eScript lett a leghasználtabb, ami az ECMA-262 standard implementációja, ami a rendkívül népszerű JavaScript alapja is.
[6]

### 3.2.2. Oracle Siebel CRM hátrányai

Kiforrottsága ellenére, érződik, hogy egy régi szoftverről beszélünk. Észrevehető a folyamatos fejlődés, de akadnak olyan tulajdonságai, ami más rendszerekben sokkal fejlettebb.

Egyik, talán legnagyobb hátránya, a többfejlesztős projektek. Ahhoz, hogy szerkeszteni tudjunk valamit a Siebel Tools-ban, le kell zárnunk, az adott objektumot, ami a munkálatok alatt, más fejlesztőknek elérhetetlenné válik, a zárolás idejére, így a fejlesztés kevésbé lesz ütemes, mint lehetne. Illetve a fejlesztői környezet működése sem mindig zökkenőmentes. Gyakran akadnak hibák, kisebb fagyások, aminek orvoslása csak egy újraindítással lehetséges.

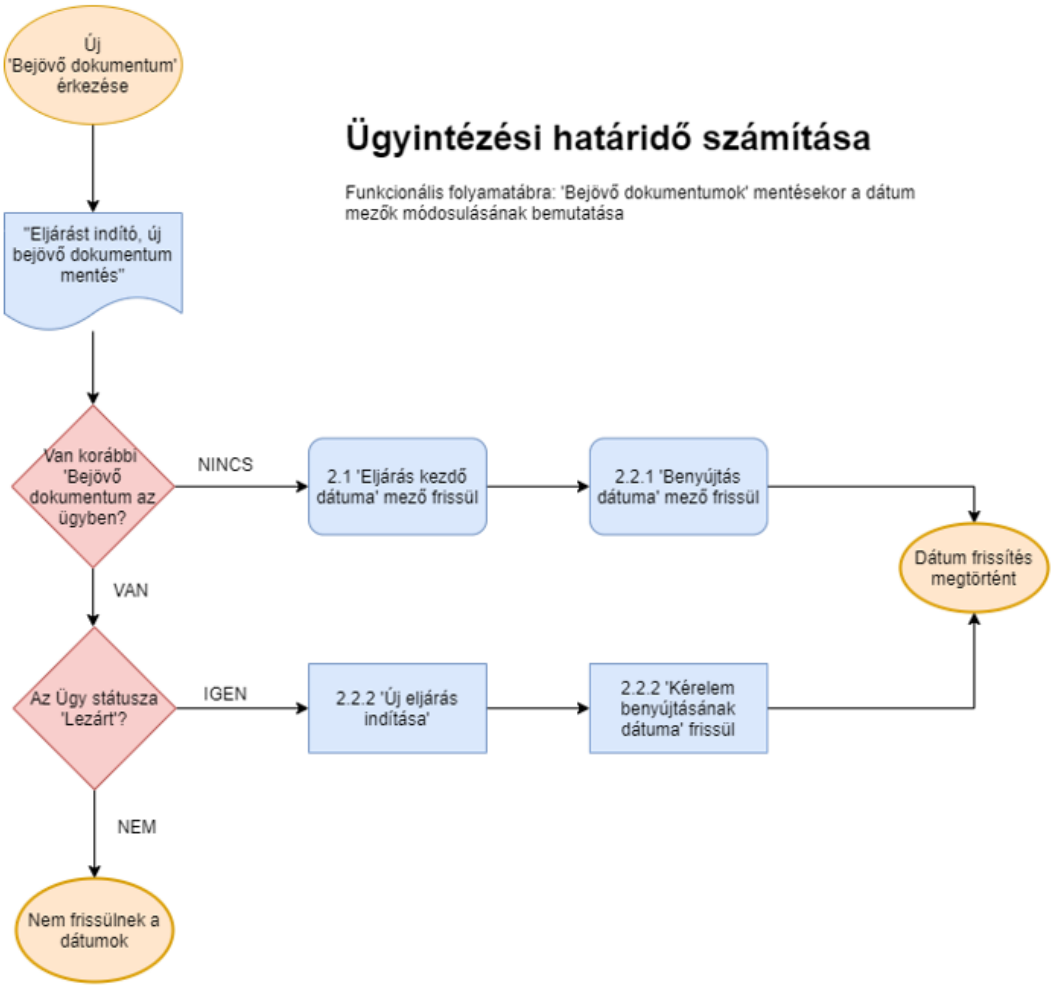



3. ábra. Siebel Applikáció felépítése

# 4. Tervezés

A tervezési fázisban szó lesz a megrendelő által megfogalmazott igényekről, illetve az átalakítandó működés bemutatásáról természetes nyelven. A fejezet nem definiálja a rendszer belső működését; nem tartalmazza annak a meghatározását, hogyan lesznek a rendszer funkciói implementálva (5. fejezet).

A konkrét feladat, az Ügyintézési határidőszámítás teljes refaktorálása. Minden személynek lehetnek ügyei, és minden ügyön belül több eljárás lehetséges, azonban egyszerre csak egy. Ezek az eljárások egy Bejövő dokumentum érkezésére indulnak el, aminek a feladási módja lehet Postai (Tértivevény), illetve SZÜF (eTértivevény).




4. ábra. Siebel Applikáció felépítése

## 4.1. Mezők bemutatása

Az alábbi pontban kerül bemutatásra egy adott személy ügyét részletező Appleten látható mezők, és a hozzájuk tartozó elvért működés ismertetése.

Mezők és működésük:

- Eljárás kezdő dátuma: A mező elvárt működése, az eljárást indító 'Bejövő dokumentum' mentésekor az abban szereplő 'Érkezés dátuma' kerül be az ügybe, mint az ügy 'Eljárás kezdő dátuma'. Azonos eljárásban további 'Bejövő dokumentum' mentésekor nem módosul az 'Eljárás kezdő dátuma' mező. Abban az esetben, ha az eljárást indító bejövő dokumentum 'Érkezés dátuma' változik, akkor az a mező is frissül.

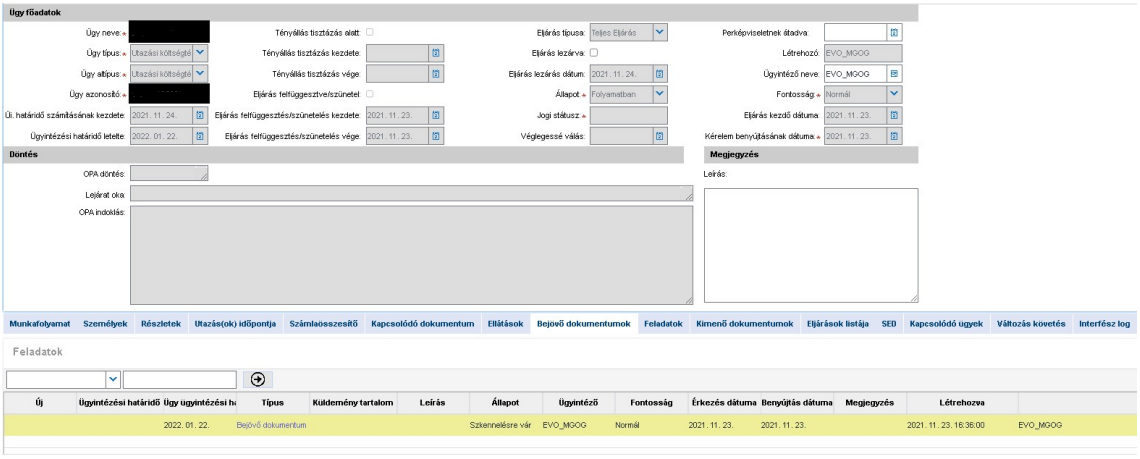
- Benyújtás dátuma: A mező elvárt működése: Ha az 'Ügyhöz tartozó' 'Bejövő dokumentum' típusú feladat 'Benyújtás dátuma' megváltozik és nincs korábban létrehozott 'Bejövő dokumentum' az ügyben, akkor a mező megkapja a 'Benyújtás dátuma' mező értékét.

- Eljárás típusa: Az új 'Eljárás típusa' mező egy lista (List Of Value, továbbiakban LOV), amely kizárólag olvasható, a rendszer által automatikusan kerül beállításra és az alábbi értékeket tartalmazza: 'Automatikus', 'Sommás', 'Teljes eljárás'. A mező alapértelmezett értéke: 'Sommás' lesz. 'Automatikus' értéket vesz fel automatikus feldolgozás esetén és a rendszer az ügyintézési határidő végét is beállítja. 'Teljes eljárás' értéket vesz fel dokumentum kiküldés és a következő iratkategóriák esetén: 'Jogsegély', 'Szakhatósági állásfoglalás', 'Eljárás Felfüggesztés/Szünetelés', 'Tényállás tisztázása'.

- Ügyintézési határidő számításának kezdete: 'Eljárás kezdő dátuma' értékhez hozzáadunk egy napot. Amennyiben változik az ügy 'Eljárás kezdő dátuma', akkor ez a mező is frissül, azaz új értéket vesz fel. A mező kizárólag csak olvasható tulajdonságú!

- Ügyintézési határidő letelte: A mező elvárt működése: Amennyiben változik az 'Ügyintézési határidő számításának kezdete' mező, akkor ez a mező is frissül, azaz új értéket vesz fel.

Látványterv a létrehozni kívánt oldalról:




5. ábra. TÉBA tervezett felülete

## 4.2. Ügyintézési határidő számítása általános esetben

”Általános” ügyaltípus esetekben a kalkuláció az alábbi módokon valósul meg, ’Automatikus’ döntéshozatal esetén - ‘Üi. határidő számításának kezdete’ + 1 nap. ’Sommás’ eljárás esetén - ‘Üi. határidő számításának kezdete’ + 7 nap (ez a mező alapértelmezett értéke). ’Teljes’ eljárás - ‘Üi. határidő számításának kezdete’ + 59 nap.

## 4.3. Ügyintézési határidő számítása egyéb ügyaltípusok esetén

A felsorolt ügyaltípusoknál az „Általános” esettől eltérő, egyedi kalkulációt szükséges alkalmazni. Ezeknek az ügyaltípusú eljárások Appleteinek kinézete is el fog térni a többi általános esettől.

### 4.3.1. Otthonfelújítási támogatás ügyaltípus

Amennyiben az átvétel módja SZÜF, az Ügyintézési határidő letelte mező értéke egyenlő lesz, az ‘Üi. határidő számításának kezdete’ + 30 nappal. Minden más esetben Az ügyintézési határidő 60 nap lesz.

### 4.3.2. KRESZ ügyaltípus

KRESZ ügyaltípus esetekben a kalkuláció az alábbi módokon valósul meg; ‘Üi. határidő számításának kezdete’ + 20 nap.

### 4.3.3. Nyelvvizsga ügyaltípus

Nyelvvizsga ügyaltípus esetekben a kalkuláció az alábbi módokon valósul meg: ‘Üi. határidő számításának kezdete’ + 30 nap.

## 4.4. Kimenő dokumentumok hatása

Nyelvvizsga ügyaltípus esetekben a kalkuláció az alábbi módokon valósul meg: ‘Üi. határidő számításának kezdete’ + 30 nap.

### 4.4.1. Tényállás tisztázása

’Kimenő dokumentum’ rendszerben történő mentésekor a következő irat kategóriák esetében a ‘Tényállás tisztázás alatt’ flag beállításra kerül, manuális állításra nincs lehetőség. ‘Tényállás tisztázása’, ’Jogsegély’, ‘Szakhatósági állásfoglalás’, ‘Elj. Felfüggesztés/Szünetelés’. Ebben az esetben elkezdődik a tényállás tisztázása folyamat, aminek hatására a ’Tényállás tisztázás kezdete’ dátum mező értéke feltöltésre kerül az adott nappal. Ezen felül az eljárás típusa ’Sommás’ értékről ’Teljes eljárássá’ változik, ami a fent említett mértékben növeli, az ’Ügyintézési határidő leteltének’ az értékét.

Amennyiben az ügyintéző úgy ítéli meg, hogy a tisztázás megtörtént, a flag manuálisan kivehető. Abban az esetben, ha az ügyben ’Auto Feldolgozás’ funkció kerül meghívásra, ez automatikusan megtörténik

### 4.4.2. Eljárás felfüggesztés, szünetelés

’Kimenő dokumentum’ rendszerben történő mentésekor az ’Eljárás Felfüggesztés/Szünetelés’ irat kategória kiküldése esetében a ’Eljárás Felfüggesztés/Szünetelés’ flag automatikusan beállításra kerül és elkezdődik az eljárás felfüggesztése vagy szüneteltetése. A flag beállításának pillanatában, a ’Felfüggesztés, szünetelés kezdete’ mező értéke automatikusan kitöltésre kerül, az aznapi dátum értékével, mint a Tényállás tisztázásnál láthattuk.

Amennyiben a 'Eljárás Felfüggesztés/Szünetelés' flag ügyintéző által manuálisan kivételre kerül, vagy az ügyben OPA kerül meghívásra (ebben az esetben automatikus flag kivétel történik) a flag módosítás dátumának értéke kerül beállításra 'Felfüggesztés, szünetelés vége' mezőben.

Ha 'Eljárás Felfüggesztés/Szünetelés' irat kategóriájú dokumentum kerül kiküldésre, akkor az 'Ügyintézési határidő letelte' mező automatikusan null/üres (nincs értéke) lesz. Amikor az ügyintéző kiveszi az 'Eljárás felfüggesztés, szünetelés" flaget vagy OPA hívás történik, és a 'Felfüg-gesztés, szünetelés vége dátum' mező kitöltésre kerül, akkor a 'Felfüggesztés, szünetelés kezdete' és a 'Felfüggesztés, szünetelés vége' dátumok különbségével növeljük az eredeti 'Ügyintézési határidő letelte' értéket.

## 4.5. Eljárás lezárása

Az eljárás lezárása manuálisan és kimenő irat alapján esetek jelölésére az 'Ügy főadatok' appleten 2 darab új mező kerül kialakításra; 'Eljárás lezárva' jelölő' (flag) és 'Eljárás lezárás dátum' elnevezésekkel.

### 4.5.1. Eljárás lezárása kimenő dokumentum alapján

Az eljárás automatikusan lezárásra kerül abban az esetben, ha a Kimenő dokumentum Irat sablonjának 'Eljárás lezáró' flag-je be van állítva. Amennyiben eljárást lezáró dokumentum kerül kiküldésre a rendszerből, automatikusan beállításra kerül az "Eljárás lezárva" flag és az aktuális (tárgy napi) dátum értékét veszi fel "Eljárás lezárás dátum" mező.

Ez alól kizárólagos kivételt jelent Útiköltség ügytípus esetén, ha az Ellátás aktiválásra kerül a rendszerben, akkor ez eljárás automatikusan lezárásra kerül.

### 4.5.2. Eljárás lezárása manuálisan

Az ügyintéző manuálisan lezárhatja az eljárást amennyiben az ügyben szerepel legalább egy Kimenő dokumentum, melynek irat sablonjának az 'Eljárás lezárás' flag értéke be lett állítva. Amennyiben nincs ilyen tulajdonságú sablon, az oldal hibát dob és a program futása megszakad, a flag nem kerül kitöltésre.

## 4.6. Sablonok importálása

Ahogy azt az előző pontokban említettem, egy eljárás lezárásához, kell egy iratsablon, melynek az ’Eljárás lezáró’ tulajdonsága be van állítva, azonban a jelenlegi rendszerben nem található ilyen tulajdonság, hiszen egy új funkcionalitásról beszélünk. Sablonokat a jelenleg is használt verzióban csak kézzel lehet felvinni, az adatokat egyesével kitöltve. Erre fogok megoldást kínálni, hogy az új mező beállítása mellet, az iratsablonok csoportos importálása is megoldott legyen, ezzel könnyítve a folyamatokat.

A feltöltéshez szükségünk lesz .csv (Comma-separated values), illetve .rtf (Rich Text Format) és .xlf (XML Localisation Interchange File Format) kiterjesztésű fájlokra. Ezek közül, talán csak a XLIFF formátum, ami magyarázásra szolgál.

```
<?xml version = '1.0' encoding = 'utf-8'?>
<xliff version="1.0">
   <file source-language="EN-US" target-language="EN-US" datatype="XDO" original="orphan.xlf" product-version="orphan.xlf" product-name="">
      <header>
         <skl>
            <internal-file>H4sIAAAAAAAAAO29e5PbNrYv+r/B14/0oyJ06IFx+TOXuukziPiZ3Jjb1n5uytXadAEuyWmy11RHbb7ZS/+/0t8CFRIiWSUtvtieVyCySBBaz3WgAI/T5bFSmb6cSk
         </skl>
         <prop-group name="ora_reconstruction">
      </header>
      <body>
         <trans-unit id="4b075587" maxbytes="4000" maxwidth="39" size-unit="char" translate="yes">
         <trans-unit id="9ab698a2" maxbytes="4000" maxwidth="47" size-unit="char" translate="yes">
         <trans-unit id="9df2ea90" maxbytes="4000" maxwidth="23" size-unit="char" translate="yes">
         <trans-unit id="79cc0063" maxbytes="4000" maxwidth="411" size-unit="char" translate="yes" xml:space="preserve">
         <trans-unit id="51a2c855" maxbytes="4000" maxwidth="23" size-unit="char" translate="yes">
         <trans-unit id="6ee5c52a" maxbytes="4000" maxwidth="30" size-unit="char" translate="yes">
         <trans-unit id="ab1bc42a" maxbytes="4000" maxwidth="730" size-unit="char" translate="yes">
         <trans-unit id="daff996" maxbytes="4000" maxwidth="203" size-unit="char" translate="yes">
         <trans-unit id="3cbbf7ee" maxbytes="4000" maxwidth="56" size-unit="char" translate="yes">
         <trans-unit id="dbdea62d" maxbytes="4000" maxwidth="29" size-unit="char" translate="yes">
         <trans-unit id="76bde986" maxbytes="4000" maxwidth="59" size-unit="char" translate="yes">
         <trans-unit id="b7667b8e" maxbytes="4000" maxwidth="285" size-unit="char" translate="yes">
         <trans-unit id="961d2e64" maxbytes="4000" maxwidth="26" size-unit="char" translate="yes">
         <trans-unit id="681baaad" maxbytes="4000" maxwidth="47" size-unit="char" translate="yes">
         <trans-unit id="dc8ff908" maxbytes="4000" maxwidth="747" size-unit="char" translate="yes">
         <trans-unit id="9a778a67" maxbytes="4000" maxwidth="51" size-unit="char" translate="yes">
         <trans-unit id="12a98c9d" maxbytes="4000" maxwidth="723" size-unit="char" translate="yes">
         <trans-unit id="53b2d333" maxbytes="4000" maxwidth="46" size-unit="char" translate="yes">
         <trans-unit id="15ca5773" maxbytes="4000" maxwidth="1041" size-unit="char" translate="yes">
         <trans-unit id="a97f29aa" maxbytes="4000" maxwidth="120" size-unit="char" translate="yes">
         <trans-unit id="812817e5" maxbytes="4000" maxwidth="26" size-unit="char" translate="yes">
            <source>A végzést kapják:</source>
            <target>A végzést kapják:</target>
            <note>Text located: body/table</note>
         </trans-unit>
         <trans-unit id="c4a950c6" maxbytes="4000" maxwidth="27" size-unit="char" translate="yes">
         <trans-unit id="ddbcce69" maxbytes="4000" maxwidth="23" size-unit="char" translate="yes">
      </body>
   </file>
</xliff>
```

6. ábra. TÉBA tervezett felülete

### 4.6.1. CSV fájl létrehozása

A feltöltéshez szükséges csv kiterjesztésű file neve ReportsÉÉÉÉHHNN.csv formátumnak kell lennie, ahol az ÉÉÉÉHHNN az aktuális dátumot jelöli.

Az ÉLES-hez a .csv állomány létrehozása: a TEST Siebel oldalon az Adminisztráció – BI Publisher riportoknál a Dokumentumok alatt található szükséges sablonok kijelölése és exportálása. Fontos, hogy az oszlopok sorrendje az előzetes specifikáció szerint legyen.

### 4.6.2. .Rtf és .Xlf file-ok

A sablonok feltöltéséhez szükség van az rtf és xlf kiterjesztésű állományokra. Az egy sablonhoz tartozó állományok nevének azonosnak kell lennie (például VisszavonoVegzesUKTRTFv210811.rtf és VisszavonoVegzesUKTRTFv210811.xlf). A csv állományban ugyanezen a néven a megfelelő sablonnévnél a Template és XLIFF mezőben kell szerepelniük.

Ha nem a sablonok frissítése a cél, hanem a Siebel oldalon az alábbi mezők módosítása, akkor nincs szükség az .rtf és .xlf file-okra

### 4.6.3. Sablonfeltöltés működése

A Business Service vár egy .csv kiterjesztésű fájlt, a mezők fent írt sorrendjével. Ezen felül a csv-ben is szereplő .rtf és .xlf is kell a .csv mellé. Akár új sablon kerül létrehozásra, vagy meglévőt kell frissíteni egy .csv állományban szerepelhetnek.

Egyszerre több .csv file is importálásra kerülhet. Ami fontos, hogy mindennek egy a fent írt helyen kell lenniük.

Amennyiben az adott oszlop Siebel oldalon nem “Required field” típusú, a feldolgozás alatt álló mező üres értékkel is importálásra/ frissítésre kerül! Ez a későbbiekben hibát okozhat. Ez alól kivételt jelent a “Kezdete” mező, ahol üres mező esetén, automatikus kitöltés következik be az aznapi dátummal.

A sikeres feldolgozást követően a fájlok a Backup mappába kerülnek, azon belül meg futtatási dátum értékével megegyező mappába (Példa: 20210923152148115). Ha a program hibára fut, a fel nem dolgozott elemeket nem teszi át a Backup mappába és figyelmeztetést kapunk. Ha a metódus sikeresen lefutott, akkor azt egy felugró ablak is jelzi: ”A Sablon importálás sikeresen lefutott.”.

### 4.6.4. Sablon mezőinek frissítése

Csak akkor használjuk, ha Sablon neve, Primary Integration Object Name, Template, XLIFF és Riport szám mezőket leszámítva akarjuk frissíteni a többi értékeket.

Amennyiben a megadott elérési útnál csak 1 darab .csv file szerepel és semmi más, akkor az adatok feltöltés helyett frissítésre kerülnek. Ha a program sikeresen lefutott, akkor egy felugró ablak fog megjelenni a következő szöveggel: ”Az adatok frissítése befejeződött!”.

### 4.6.5. Technikai leírás a tervezett működésről

Fájlok kigyűjtése: a Business Service (‘MAK Template Import’) elindítja a ‘FileImport’ metódust, ami a megfelelő elérési útvonalon végigmegy a tárolt fájlokon és ellenőrzi annak formátumát. Amennyiben az .csv kiterjesztésű elkezdődik az importálás. Ha nincs ilyen típusú fájl, ezt jelzi a felhasználó felé (Hibaüzenetek pont alatt látható erre a konkrét példa.).

Fájl beolvasása lépésben elkezdjük a fájl beolvasását sorok szerint. Majd ezeket a sorokat bontjuk tovább oszlopokra. A beolvasás során ellenőrizzük, hogy található e már a beérkező adattal megegyező “Sablon neve” érték, ha nem akkor az oszlopok tartalmának beolvasásával feltölthetjük az új fájlt. Azonban, ha találtunk már ilyen értéket, akkor az oszlopok aktualizálása kezdődik el.

Fájlok importálása közben egy globális tömbbe elmentjük az importált .rtf és .xlf fájlokat. Miután az összes sor importálva lett sikeresen, a .csv file-t átrakjuk a Back-up mappába, azon belül pedig futtatási dátum értékével megegyező mappába (Példa: 20210923152148115). Ezek után, ha az egész fájl importálása során nem adódott hiba, végig megyünk a globális tömbön, ahova tároltuk a felhasznált fájlokat, és azokat is áthelyezzük ugyanabba a mappá-ba, ahova a .csv file került.

A program leállását “A folyamat befejeződött!” felugró ablak jelzi. Az OutputArguments-ben olvashatjuk az eredményt. Amennyiben hiba adódik, abban található a részletek.

A hibaüzeneteket a szimulátor Output Arguments alatt lehet megtalálni. Ha a program hiba nélkül futott le, akkor az jelenik meg. Amennyiben több sort töltünk BE, a részletezéshez, nyissuk meg a “Property Name”nél szereplő ikonra.

# 5. Megvalósítás

Ebben a fejezetben érinteni fogom a TÉBA rendszer három fő környezetét, és bemutatásra kerülnek a funkcióik. Ennek részletezése egy keretet ad a fejlesztési folyamatoknak, hogy miként jut el egy teljesen új fejlesztési igény a tervezés után a fejlesztésen át, a tesztelés után, addig, hogy kikerülhessen az éles környezetre.

A fejlesztési struktúra bemutatása után részletezni fogom a TÉBA rendszerben az Ügy-intézési határidőszámítás során érintett, valamint az általam fejlesztett kódrészleteket, illetve User Interface(UI) felületeket. Ezt követően a teljes mértékig, általam megvalósított Tömeges Sablonimportálás működési logikáját részletezem, kódrészletekkel együtt.

## 5.1. TÉBA környezetek

A TÉBA CRM rendszerét három plusz egy önálló környezetre tagoltunk az átláthatóság megőrzése és a fejlesztés során előforduló fejlesztői hibák, illetve az üzleti igények utólagos megváltoztatásának lehetősége miatt. Ezek a DEV, TESZT és PROD környezetek. Mindegyik rendelkezik saját adatbázissal.

### 5.1.1. DEV környezet bemutatása

A DEV környezeten kezdődik egy fejlesztési igény megvalósítása. A komponens lezárása után, az előre megírt fejlesztési jegyzőkönyv szerint el is indulhat a megvalósítás. Siebel rendszerre általánosan jellemző tulajdonság, hogy ha egy user lockolja az adott komponenst, akkor az más fejlesztő számára nem lesz hozzáférhető, a kódütközések elkerülése miatt. Annak céljából, hogy a többi fejlesztőnek ne legyen üresjárati ideje, az adott komponens kódját kiexportáljuk és nem túl elegáns módszerrel, Notepad++ szövegszerkesztő segítségével folytathatja a kódolást. A fejlesztői tesztelések is ezen a környezeten zajlanak.

### 5.1.2. TESZT környezet bemutatása

Ezt környezet, az üzemeltetés használja, a fejlesztő által átadott, és tesztjegyzőkönyvvel ellátott fejlesztések felhasználó oldali tesztelések elvégzésére. Itt rendszerint előjönnek olyan hibák, amit független komponensekkel való együttműködés okozhat, illetve az üzleti igények megváltozása is itt jöhet elő a legnagyobb valószínűséggel. Ezeknek a javítását, illetve fejlesztését, duplán kell lefejleszteni a DEV és TESZT környezeteken.

### 5.1.3. PROD környezet bemutatása

Ezt környezet, az üzemeltetés használja, a fejlesztő által átadott, és tesztjegyzőkönyvvel ellátott fejlesztések felhasználó oldali tesztelések elvégzésére. Itt rendszerint előjönnek olyan hibák, amit független komponensekkel való együttműködés okozhat, illetve az üzleti igények megváltozása is itt jöhet elő a legnagyobb valószínűséggel. Ezeknek a javítását, illetve fejlesztését, duplán kell lefejleszteni a DEV és TESZT környezeteken.


7. ábra. TÉBA Release folyamatábra

## 5.2. Fejlesztési folyamatok

A következő pontban a fejlesztési folyamatok bemutatását fogom részletezni. Első lépésként a Sablonimportálás, és annak fejlesztésének menetét fogom bemutatni, hiszen a későbbiekben a Business Service felhasználásának eredménye elengedhetetlen a határidőszámításhoz. Segítségével részleteiben is bemutatásra kerül az eScript és annak használata.

Ezt követően részletezésre kerül az Ügyintézési határidő számításának kialakítása/refaktorálása. Bemutatásra fog kerülni, az adatbázis oszlopok létrehozás, Business Component field-ek kialakítása. Szkriptek írása, fejlesztési technikák bemutatása. Illetve végül a Frontend környezet bemutatása.

# 6. Sablonimportálás fejlesztési folyamatainak bemutatása

A fejlesztéshez szükséges mezők összegyűjtésével kezdtem. Ezek a kódolás átfogó tervezésekor hasznosak lesznek.

## 6.1. Feldolgozandó adatok rendszerezése

Első lépésként összegyűjtöttem a megkapott adatokat. Ezeknek az adatoknak változó neveket adtam.

| Mező neve | Változó értéke |
|---|---|
| Sablon neve | sRepName |
| Leírás | sRepDesc |
| Kezdete | sActivationDate |
| Vége | sDeactivationDate |
| Primary Integration Object Name | sRepIOB |
| Template | sRepTemp |
| Típus | sOutputType |
| Default Language | - |
| Default Locale | sLocale |
| XLIFF | sRepXlf |
| Riport szám | - |
| Kiválasztott sorok | - |
| Irat kategória | sCategory |
| Ügy altípus | sCaseSubType |
| Ügy típus | sCaseType |
| Doc Flag | bDocFlag |
| Tértivevénnyel kiküldendő | bMailFlag |
| Személyesen kiadható | bMailFlag |
| Személyesen kiadható | bPersFlag |
| Kiadmányozhatja | sIssuerPos |
| Aktivál | - |
| Eljárás lezáró | sProcedureEndingFlag |

## 6.2. Szkript megírása

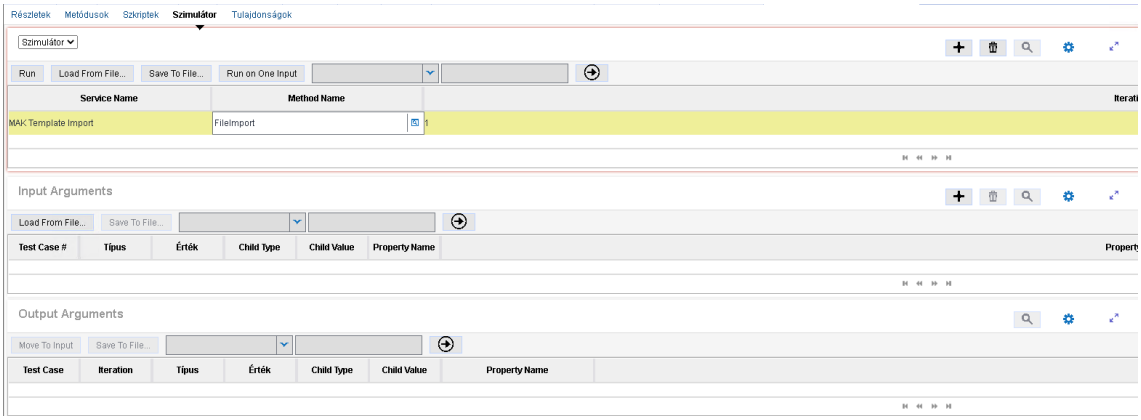
Létrehoztam a fájlok feldolgozásához szükséges Business Service-t (későbbiekben BS)

```
function Service_PreInvokeMethod (MethodName, Inputs,
   Outputs)
{
 switch (MethodName)
 {
   case "FileImport":
   {
     // FileImport method.
     TheApplication().LogDebug(this, 1, "FileImport method:
        Started.");
     FileImport(Inputs, Outputs);
     TheApplication().LogDebug(this, 1, "FileImport method:
        Ended.");
     if(!gbNoUpload)
     {
       TheApplication().RaiseErrorText("A folyamat
          befejezdtt!");
     }
     else
     {
       TheApplication().RaiseErrorText("Az adatok frisstse
          befejezdtt!");
     }
     gbIsLocalEnv = null, gsDelimiter = null,
        gsImportDirPath = null, gsBackupDirPath = null;
     gsToday = null, gsFilePath = null, gsFileName = null,
        gaBackupFiles = new Array(),
     gbIsGlobalError = false, giErrorCounter = 1,
        gbNoUpload = null; //evo_mgog 2022.01.26. #16863
     return (CancelOperation);
   }
 }

 return (ContinueOperation);
}
```

A BS meghívásakor ez a metódus fut le először. Paraméterként ez megkapja a BS-en belül meghívott metódus nevét, az Input és az Output paraméterekkel együtt. Amennyiben több metódus is meghívható egy service-en belül, vagy megvan a lehetőség, hogy a későbbi fejlesztések során lesz ilyem, úgy ezt egy switch-case segítségével dolgozzuk fel. Természetesen csak String-ként kpajuk meg a metódus nevít, így annak felhívása

továbbra is a mi dolgunk marad.




8. ábra. Business Service felhívása

A tesztelhetőség megkönnyítése miatt a fájlok beimportálását 2 részre szedjük szét. Megvizsgáljuk az adott SRF-et, és amennyiben ez lokális környezeten van futtatva, akkor a C meghajtóról fogja keresni az importálásra váró elemeket. Ha szerver környezetről származik az SRF, akkor a megfelelő szerver oldali mappában fog végig menni a script, a beolvasásra váró sablonokon. Ez a mappaszerkezet Érték listákban van eltárolva, így bármikor a kód változtatása nélkül alakítható az elérési útvonal, amennyiben arra igény adódik a későbbiekben.

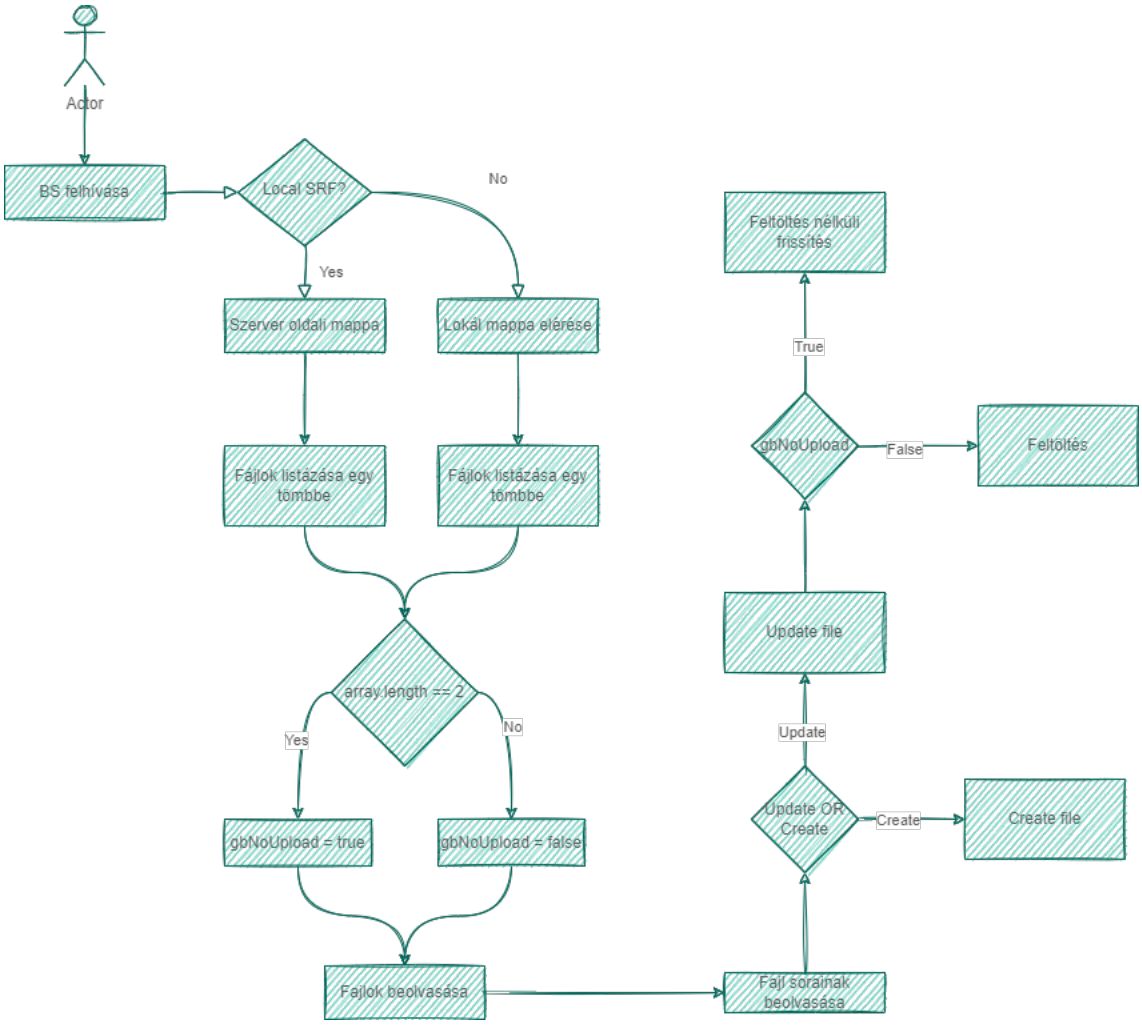
A felhívott mappában, végig megyünk az ott található .csv fájlokon és azokat egy listába rendezzük a feldolgozás megkönnyítése végett. Végig megyünk ezen a listán és megvizsgáljuk, hogy amennyiben két elemű a tömb, az azt jelenti, hogy nincs feltöltendő .rtf és .xlf fájl, így a .csv fájlban szereplő elemeknél nem kell felhívni a feltöltést, csak a Siebelben található értékeket kell frissíteni. Ezt egy globális változó beállításával tesztem lehetővé.

A fájlok beolvasásához végig megyünk a lista elemein, és azokat átadjuk a 'ReadFile' metódusnak. Ez beolvassa a fájl tartalmát stringként és a sortörések mentén egy tömbe rendezi. Majd a sorokat egy for ciklus segítségével továbbadjuk a ' ReadFileLine' metódusnak. Itt példányosítjuk a Business Objectet, és a Business Component elemet. Aktiváljuk a megfelelő mezőket, amiket változtatni akarunk.

Mivel .csv kiterjesztésű elem beolvasásáról van szó, így tudhatjuk, hogy az adatok ';' karakterekkel vannak elválasztva, így az elemeket egy tömbe rendezzük, úgy, hogy a sort szét szabdaljuk a ';'-ők mentén.

A 'Report Name' mező egyedi azonosító szerepkört is belát, így meg kell keresnünk azt a komponenst(BC-t), amivel megfeleltethető az importálni kívánt 'Report Name' oszlop értéke. Amennyiben nem találunk ilyen elemet, be kell szúrnunk egy új sort a

legvégére és oda elmenteni az értékeket. Ezt a két lehetőséget természetesen két külön metódus fogja elvégezni, attól függően, hogy a 'FirstRecord' metódus 'true' vagy 'false' értékkel tér vissza. Fontos kitétel emellett még, hogy amennyiben feltöltés nélküli adatfrissítés történik, az csak abban az esetben következhessen be, ha van frissítendő elem.




9. ábra. Sablonimportálás folyamatábra

Az Update és Create metódusok nagyon hasonlítanak egymásra, két dolgot leszámítva. Az új adat létrehozásánál nem kell vizsgálni, hogy csak frissítünk e feltöltés nélkül, mert teljes bizonyossággal állíthatjuk, hogy itt nem történhet ilyen eshetőség. Amennyiben kell feltöltés, úgy meg kell keresnünk a feltölteni kívánt .rtf és .xlf fájlokat, a nevük alapján, amit a .csv fájlban a megfelelő mezők tartalmaznak. Amennyiben vannak ilyen fájlok, úgy felhívjuk a beépített Business Service-t, ami elvégzi a feltöltés a megfelelő helyre.

Minden hibaüzenetet egy Property Set item-ben tárolunk, így amennyiben hiba adódik a végén, ennek visszatérésével láthatjuk, hogy mely sorban volt a hiba, és mi. Ez jelentősen megkönnyítette a felhasználók hibakeresését, így kevesebb ticketet kaptunk a későbbiekben ismeretlen hibákról.
[7] [8]

# 7. Ügyintézési határidőszámítás

A határidőszámítás refaktorálásához először össze kell szedni azokat a komponenseket, amik a régi logikát tartalmazzák. Ennek összegyűjtésé meglehetősen hosszú feladat volt, mert az eddigi kódok nem voltak egységesítve, és össze vissza helyezkedtek el különböző Business Component-ekben. Ezen felül meg kellett vizsgálni, hogy az adott kódrészleteket használjuk-e esetleg máshol, amennyiben igen, az mennyire lesz helyettesíthető az új kódokkal, illetve azokat, hogy alakítsuk, hogy a későbbiekben is használhatóak legyenek.

## 7.1. Régebbi kódok revíziója

A határidőszámítást tartalmazó komponensesek listája: 'Action', 'HLS Case', 'HLS Releated Case'. Ezekben a scriptekben kellett határidőszámításra utaló függvényeket keresni. Dokumentáció hánya miatt csak a kommentekre, illetve a kódokra kellett hagyatkozni, ami jelentősen megnehezítette a haladást. Miután megtaláltuk azokat az elemeket, amiket érinteni fog a változtatás, elkezdődhetett a szortírozás. Főleg azok a függvények voltak a célkeresztben, amik redundánsak voltak, és több komponensben is implementálva volt ugyanaz a kódrészlet minimális változtatásokkal. Ennek generikussá tétele nagyban segíti a 'Clean Code'-ra való törekvésemet. Fontos, hogy semmit sem volt szabad törölni, mert sosem lehet tudni, hogy melyik kevésbé ismert folyamatot ronthatunk el egy ilyen kódrészlet kivágása miatt, ezért könnyen kereshető módon kikommenteztük a fent említett kódrészleteket, függvényeket.

## 7.2. Üzleti logika implementálása

Az üzleti logika implementálásához meg kell vizsgálnunk, hogy a megrendelő által állított követelmények, és ügyviteli folyamat, hogyan illeszkedik a TÉBA rendszerében kód szinten. Különböző okok miatt az implementálás a TÉBA-ban és az alap logika eltérhet egymástól. Ennek következtében a fejlesztési folyamatábra el fog térni attól, amit a megrendelői logika alapján készítettem, illetve egy folyamatábrában nem is lenne feltétlen célszerű ábrázolni, hiszen annyira szerteágazó, és sok változós, hogy átláthatóság szempontjából nem lenne a legcélravezetőbb.

A fejlesztést többféle sorendben lehetne bemutatni, de úgy döntöttem, hogy azt veszem kezdeti állapotnak, hogy ha egy új bejövő dokumentum érkezik, és ez hogyan befolyásolhatja az adott ügyet. Természetesen nem csak bejövő, hanem kimenő dokumentum hatására is változhat az ügyintézési határidő.

A folyamat az 'Action' BC-ben kezdődik, amikor beérkezik egy bejövő dokumentum. Ilyenkor már több ellenőrzést is végre kell hajtani, hogy a megfelelő kódsorok fussanak le, és a megfelelő flag-ek kerüljenek beállításra. Első és legfontosabb, hogy ellenőrizzük, hogy az adott dokumentum 'Bejövő dokumentum'-e és nem 'Kimenő dokumentum', 'Figyelmeztetés', vagy 'Egyéb' dokumentum. Ezek után meg kell vizsgálnunk, hogy ez az első bejövő dokumentum-e az ügyben. Amennyiben igen, úgy ez lesz az eljárás indító dokumentum, és a dámjaival megegyezően a fent említett üzleti elvárásokkal kitölti az ügy(HLS Case Business Component) dátum mezőit. Ha nem ez az első bejövő dokumentumunk, akkor meg kell vizsgálni, hogy az ügyben a 'Lezárva' flag be van-e pipálva. Abban az esetben, ha igen, akkor ismételten az előző logika szerint új eljárás indul és a dátum mezők frissülnek. Máskülönben nem történik változás, és a dokumentum nem befolyásolja a számításokat. Ezeket a feltételeket, az alábbi kódrészletekkel szemléltetem a logikai feltételeket.

```
//TheApplication().LogDebug(this, 3,"Not a
   RecordedDeliveryIncDoc: " + !RecordedDeliveryIncDoc());
   // evo_mgog #17143 2022.03.01.
    if(bcHLSCase.GetFieldValue("MAK Current Procedure Id") ==
       "" &&
    bcHLSCase.GetFieldValue("MAK Admin Start Date") == "" &&
       gbIsNewRecord &&
    this.GetFieldValue("Type") ==
       TheApplication().InvokeMethod("LookupValue",
       "TODO_TYPE", "Incoming Document")
     && !RecordedDeliveryIncDoc() // evo_mgog #17143
        2022.03.01.
    )// // evo_mgog 2022.02.02. #17040 | Kell a
        !RecordedDeliveryIncDoc() vizsglat a regi ugyek miatt}
     //---------------------------//
    else if (bcHLSCase.GetFieldValue("MAK Procedure End
       Flag") == "Y" && gbIsNewRecord &&
    this.GetFieldValue("Type") ==
       TheApplication().InvokeMethod("LookupValue",
       "TODO_TYPE", "Incoming Document")
    && !RecordedDeliveryIncDoc() // evo_mgog #17143
       2022.03.01.
    ) // evo_mgog 2022.02.02. #17040
```

Amennyiben megfelel a feltételnek, az adott példányosított Action BC tartalma, beállíthatjuk az ügy érintett tulajdonságait. Fontos megjegyezni, hogy a Business Componentek mögött relációs adatbázisok álnak, melyek össze vannak kapcsolva bizonyos oszlopok

mentén. Minden Action BC-nek van egy 'Ügy azonosító Id' mezője, nevezetesen 'Case Id'. Ezen mező segítségével meg tudjuk keresni a nekünk szükséges 'HLS Case' item-et. Példányosítjuk azt, megadunk pár alap paramétert, és aktívájuk a módosítani/lekérdezni kívánt mezőket. Ezt követően a ' SetSearchSpec(x,y)' függvény meghívásával 'Id' alapján megtalálja a kód.

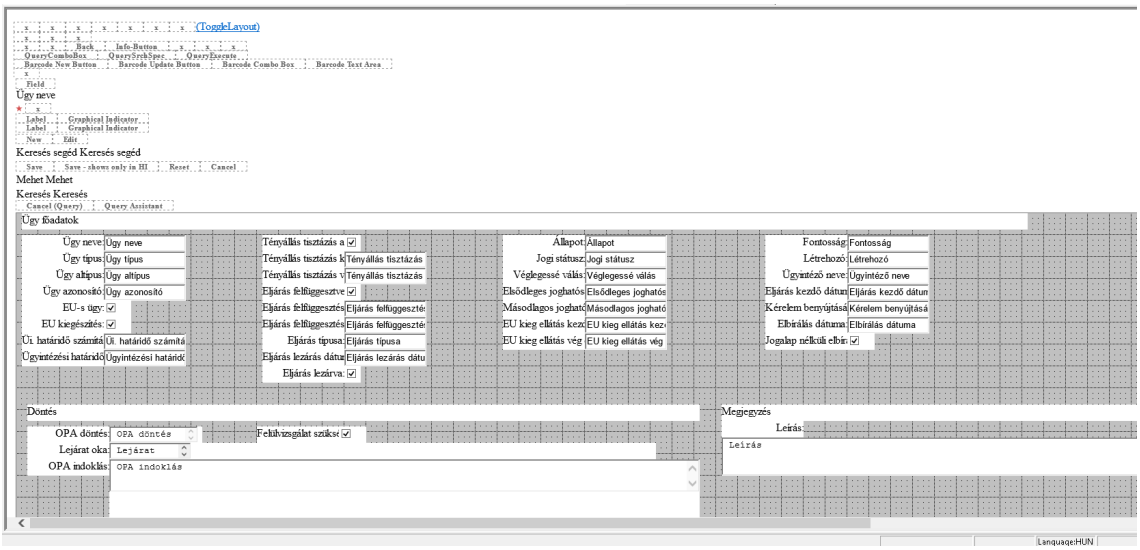
```
    TheApplication().LogDebug(this, 3,"UpdateCaseDates
       started");
  boHLSCase = TheApplication().GetBusObject("HLS Case");
  bcHLSCase = boHLSCase.GetBusComp("HLS Case");
  bcHLSCase.SetViewMode(AllView);
  bcHLSCase.InvokeMethod("SetAdminMode", "TRUE");
  bcHLSCase.ActivateField("MAK Registration Date");
  bcHLSCase.ActivateField("MAK Admin Start Date");
  bcHLSCase.ActivateField("MAK Current Procedure Id");
  bcHLSCase.ActivateField("MAK Procedure End Date");
  bcHLSCase.ActivateField("MAK Process Type");
  bcHLSCase.ClearToQuery();
  bcHLSCase.SetSearchSpec("Id", this.GetFieldValue("Case
     Id"));
  bcHLSCase.ExecuteQuery(ForwardOnly);
```

A fejlesztés során fontos volt figyelni arra, hogy az eddig legyártott több tízmillió ügy nem ezt a logikát használja, és vannak olyan elemek, amik azokban nem léteznek. Példának okáért, eddig nem léteztek eljárások, így azoknak nem volt 'Id' mezőjük(ezek megegyeznek az Action Id mező értékével, hogy a keresés a későbbiek során gyorsabb legyen ). Ezért úgy kellett megoldani, hogy valamilyen úton a régi logika is be legyen építve, de hosszú távon ezek automatikusan át legyenek vezetve az új rendszerbe. Ennek megtervezése elég sok időbe telt, de végül sikerült megoldani a problémát.

Ehhez ismerni kell a régi működést, miszerint au ügyintézési határidők, abban az esetben változtak meg kizárólag, ha az első beérkezett bejövő dokumentum dátum mezőin változtatott az ügyintéző. Ebben az esetben természetesen egy ügyben csak egy folyamat mehet végbe, ami jogszabály szerint nem helyes. Ez alapján létre hoztam egy elővizsgálatot arra, hogy amennyiben az első bejövő dokumentum értékei változnak ÉS nincs eljárás Id az adott ügyben, akkor a régi logika szerint fusson le a logika, viszont, ha van Id értéke, akkor meg kell vizsgálnunk, hogy a változtatott bejövő dokumentum 'Action Id' értéke megegyezik e az ügy 'Eljárás Id' mezőjének az értékével. Amennyiben igen, akkor az új logika szerint fut le a kód, máskülönben nem történik semmi csak a dokumentum mező értékei frissülnek be.

Azonban ezek a mezők, csak abban az esetben láthatóak a felületen, ha azokat kirakjuk

a ”Web Layout” felületen. Nagyon fontos, hogy jó módot válasszunk a nézetnek, azt amelyik megjelenik a felületen. Kézzel létre kell hozni a control mezőket, és azokat össze kell kötni a BC mezőkkel. Így az adatok már meg fognak jelenni a nézeten.



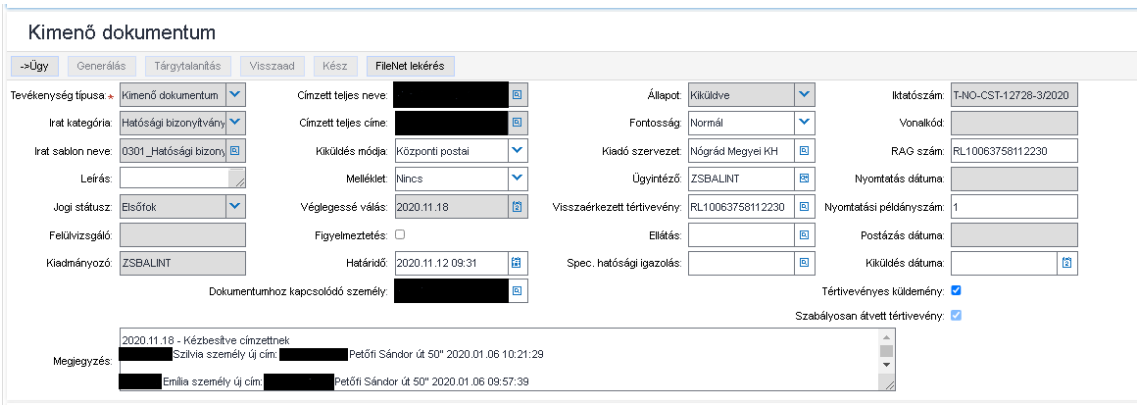

10. ábra. Appletek szerkesztése

Korábbiakban már említettem az eljárásokat. Eljárásnak tekintjük, az ügyben végig vitt ügyintézési folyamatot. Ez tart először az ügy elindulásától az ügy lezárásáig. Sok esetben egy ügyben csak egy eljárás szerepelhet, azonban vannak kivételek. Ilyen például az anyasági támogatás. Ezekben az esetekben az ügy lezárása után egy bejövő dokumentum hatására indulhat új eljárás is. És ez az a pont, ahol, a régi logika szerint működő határidőszámítást migrálni tudjuk az újba. Ugyanis, ha az ügy le van zárva és új bejövő dokumentum érkezik, akkor az ’Eljárás Id’-nek meg tudjuk határozni, a bejövő dokumentum ’Action Id’ mezőjének az értékét, innentől kezdve már, hogy ez az érték kitöltésre került, már az új logika szerint fog végig menni az ellenőrzéseken, és az ügymódosításokon.

A bejövő dokumentumok közvetlenül, az ügyben csak két mező értékét befolyásolják, ezek az ’Eljárás kezdő dátuma’, illetve a ’Kérelem benyújtásának dátuma’. Az ügyekben a ’HLS Case’-ben folyamatosan figyelemmel kíséri a script az ’Eljárás kezdő dátuma’ mező értékét, és amennyiben változás történik, az frissíti az ’Üi. határidő számításának kezdete’ nevű dátum mező értékét, ami egyenlő lesz az ’Eljárás kezdő dátuma’ mező értéke plusz egy nappal. Illetve az ügy típus alapján kiszámolásra kerül az ’Ügyintézési határidő letelte’. Ennek pontos menete a 4.3-as pontba lett kifejtve.

Az ügyintézés folyamán adódhatnak adminisztrációs problémák, és szükség lehet az eljárás-ban Tényállás tisztázásra, illetve az Eljárás felfüggesztésére vagy szüneteltetésére. Ezeket különböző sablonokkal rendelkező kimenő dokumentumok állítják be automatikusan. A sablonok tömeges feltöltéséről ebben a fejezetben feljebb esett szó. És az

ehhez hasonló esetek miatt volt rá szükség.




11. ábra. Kimenő dokumentum

Mind a két esetben nagyon hasonló a mögöttes logikai működés. Amennyiben a megfelelő irat kategóriájú('Felülvizsgálat' vagy 'Felfüggesztés és szüneteltetés') és sablonú kimenő dokumentum kiküldésre került, az irat kategóriájának megfeleltethető flag beállításra kerül automatikusan, aminek hatására a 'Tényállás tisztázás kezdete/'Eljárás felfüggesztés/ szüneteltetés kezdete dátum mező is kitöltésre kerül az adott időpillanattal. Természetesen a Java dátum formátumot át kell alakítani egy Siebellel kompatibilis formátummá is ("YYYY. MM. DD."). A flag beállításának hatására az 'Ügyintézési határidő letelte' dátum mező értéke törlésre kerül, hiszen bizonytalan ideig felfüggesztésre került az. A flag amíg nincs neki érték adva, a felhasználó számára ReadOnly tulajdonsággal rendelkezik. Beállítás után, pedig az ügyintéző által is állítható, viszont a kikapcsolás után megint csak olvasható tulajdonságú lesz.

Amennyiben a korrigálás megtörtént, akár a kérelmező személy, akár az ügyintéző által, úgy az Ügyintéző értesítést kap erről. Abban az esetben, ha minden hiányosság pótlásra került, és nincs más hibás adat, úgy az ügyintéző manuálisan ki tudja kapcsolni a flaget, aminek hatására kitöltésre kerül a 'Tényállás tisztázás vége'/'Eljárás felfüggesztés/ szüneteltetés vége' dátum mező is kitöltésre kerül az aktuális dátummal. Formázás természetesen itt is szükséges.

Amint a flageknél kivételre kerül a pipa, lefut egy script, ami újra vizsgálja a határidőszámítást. Először is megnézi, hogy az eljárás típusa 'Sommás' volt-e. Amennyiben igen, akkor azt átállítja 'Teljes eljárásra' és az 'Üi. határidőszámítás kezdete' dátum mező értékéhez nyolc nap helyett harmincat ad hozzá, abban az esetben, ha az eljárást indító dokumentum átvételi formája SZÜF (online forma), máskülönben hatvan napot adunk a kezdő dátumhoz. Ezt abból a célból tesszük, hogy népszerűsítsük az online dokumentumbeküldést, a hagyományos postaival szemben. Ezután megvizsgálja, hogy a tényállás tisztázás, vagy az eljárás felfüggesztés/szüneteltetés hány napig tartott, és ezt az értéket

hozzáadja az eredeti érékhez.

Az ügyintézési határidőszámítás során megvizsgáltunk minden eshetőséget, az eljárás indulása előtt, az eljárás és a közben adódó problémák során. Következő lépés az eljárás lezárása lenne. Ezt az ügyintéző alapvetően nem tudja manuálisan véghez vinni. Az ügyben szerepelnie kell egy eljárást lezáró tulajdonsággal rendelkező sablonú kimenő dokumentumnak. Itt megint visszatekintünk kicsit, a sablonok importálásának fontosságára, ahol ezt a tulajdonságot is beállítjuk. Az ügyintéző, csak ennek a dokumentumnak a kiküldése után lesz lehetősége lezárni ez éljárást, mert a lezáró flag, egészen idáig amennyiben megpróbálták beállítani a flag-et és utána elmenteni, egy felugró ablak figyelmeztette az ügyintézőt, hogy ”Az eljárás nem zárható le megfelelő kimenő dokumentum hiánya miatt.”. A kiküldés után már állítható manuálisan is. Későbbi eljárások során nem minden esetben kerül kimenő dokumentum kiküldésre, így az üzlettel közösen megegyeztünk, hogy amennyiben nem az első eljárásról van szó, úgy ez a flag ne legyen ReadOnly.

A folyamat végére összegezném, az újonnan létrehozott elemeket és szerepüket az egész folyamatban. Eljárás típusa mező, ami meghatározza a kezdeti ügyintézési határidőnek a végét. Tényállás tisztázása, eljárás felfüggesztés/szüneteltetés vége dátumok, ezeknek az elmentése a logika mellett későbbiekben az Auditálás miatt is fontos lesz. ’MAK Procedure Id’, amire az eddigiekben ’Eljárás Id’-ként beszéltem, ennek köszönhető az új funkció, miszerint egy ügyben több eljárás végig vitele is lehetséges. Azonban fontos megjegyezni, hogy míg egy személynek több ügye is lehet, egy ügyben egyszerre csak egy eljárás indítható. A többi mezőnél, maximum névváltoztatás ált elő, viszont az egész határidőszámítás logika gyökerestől átalakult. Az új sokkal széleskörűbb, több tényezőt vesz figyelembe. Kódjai csoportosítva találhatóak, így a karbantartás is jelentősen egyszerűsödött, és a Clean Code-ra való törekvést is nagyban elősegítette. Az elavult kódrészletek, és metódusok törlésre kerültek, helyettük, új és optimalizáltabbak lettek létrehozva.
[9]

## 7.3. Auditálás

Az új határidőszámítás logikával az átláthatóság és a követhetőség is rendkívül fontos lett, így a folyamatban szereplő elemekre meg kellett oldanunk az auditálást, hogy a jövőben a változtatások visszakereshetek legyenek és az ügyön belüli eljárások listázva legyenek. Ehhez egy új view-t kellet létrehozni, új Business Component-et, Appleteket és adatbázis táblát, ami össze van kapcsolva az ügy táblájával.

Egy ilyen típusú fejlesztésnek mindig belülről haladunk kifele. Először létrehozom Az adatbázis táblát a megfelelő oszlopokkal. Ezek megegyeznek az ügyintézési határidő

számításban használt adatokkal. Majd ezalapján létrehozzuk a Business Component-et, ahol szintén létrehozzuk az általunk létrehozott adatbázis oszlopok alapján az elemeket. Ezek az oszlopok felsorolás szerűen: Eljárás kezdő dátuma, Üi. határidő kezdete, Ügyintézési határidő letelte, Eljárás típusa, Eljárás lezárás dátuma, Tényállás tisztázás kezdete, Tényállás tisztázás vége, Eljárás felfüggesztés/szüneteltetés kezdete, Eljárás felfüggesztés/szüneteltetés vége, Feladat típus. A feladat típus egy olyan link, amire, ha rákattintunk belefúrunk a 'Feladatok' Screen-en belül a dokumentum részleteibe.

Működése úgy lett kialakítva, hogy amint létre jön egy új eljárás a 'List Applet'-re beszúrás-ra kerül egy új sor. Azonban eben az eljárásban még nem szerepel érték, így csak a feladat típusa szerepel. Az üzletnek nem volt szüksége, hogy a felsorolt elemek dinamikusan frissüljenek, így az eljárás lezárásakor kerül felhívásra az a függvény, ami a felsorolt oszlopoknak az értékadását elvégzi. Az üzlet ezt kizárólag utólagosan használja auditálásra, így az eljárás folyamata közben nem volt számukra szükséges a dinamikus frissülés, és így erőforrást takaríthattunk meg.

Fontos pont volt a tervezés meghatározása közben, hogy az új komponens mezői nem lehetnek kapcsolt oszlopok, hiszen bármennyire is megegyezik adott eljáráson belül a két érték, azonban az eljárás lezárása és egy újabb indítása után, a lezárt eljárás sora megegyezne, az újonnan indított értékeivel.

## 7.4. Tértivevények

A tértivevények kezelése az ügyekben könnyű feladatnak indult, de a fejlesztések során rá kellett jönnöm, hogy egy fokkal bonyolultabb, mint azt előre predesztináltam. Talán szokásos fejlesztési komplikációnak nevezhetem azt a problémát, amikor az ügyfél a fejlesztés előrehaladott szakaszában megváltoztatja vagy extra funkcionalitásokkal módosítja az eredeti feladatot. A tértivevények esetében is így történt.

Alapjáraton két esetben jöhet létre tértivevény, kimenő és bejövő dokumentumoknál. Kiküldéskor, olyan információk közlésére szoktak tértivevényt küldeni, mint például az eljárás lezárása például KRESZ és Nyelvvizsga ügyaltípusok esetén. Természetesen ennek kézbesítéséről, átvételéről, vagy éppen annak hiányáról érkezik egy bejövő dokumentum, melynek csatolmánya a tértivevény. Ezek visszajelzése, amik konkrét tartalommal nem rendelkeznek, viszont mégis indítanak új eljárást, hiszen bejövő dokumentum érkezett be. Viszont ezek az eljárások teljes mértékben hibásan jönnek létre, és a havi kimutatásokat elrontja. Erre kellett egy rövidtávú és hosszútávú megoldást is találni.

### 7.4.1. SQL script megírása

A rövidtávú megoldásra két megoldási lehetőséget is figyelembe vettem. Első lehetőségként számításba vettem egy Business Service írását, amit visszaállítja az adatokat az auditálás alapján. Azonban ennek megírása egy elég komplex feladat lett volna, így a másik lehetőség mellett döntöttem, ami a SQL script írása volt. A kód megírása közben nagyon pontosan és óvatosan kellett eljárni, mert az ÉLES környezetben lehet egyedül lefuttatni, mivel mi nem tudunk olyan bejövő dokumentumot gyártani, aminek kezdetektől fogva van csatolmánya, így DEV és TEST környezeten nem tudtuk tesztelni a scriptet.

Select alapján létrehozunk egy új táblát, amiben szerepelnie kell az ügy Id mezőjének, Eljárás kezdő dátumának, kérelem benyújtásának dátumának, az aktuális eljárás Id mezőnek, az ügyet indító feladat id-nak, a feladat érkezés dátumának, a 'Created' dátum érteknek. A selectben meghatározott ROW_NUM() partíció segítségével egy maximális sorszámot az ügy id és a created dátum érték alapján.

Összekapcsoljuk a Ügy, a Feladat, és a feladat Csatolmányának a tábláját egy kapcsoló tábla segítségével. Keresési feltételnek meghatároztuk, hogy milyen ügytípusokban keressen, milyen típúsú feladatoknál, illetve, hogy a csatolmányok típusa mi legyen. Ezt a belső select tartalmát összekapcsoltam a kapcsoló tábla segítségével a Feladatok táblával, hogy meg tudjuk határozni a mostani értéket, és az előzőt. Így megkapjuk azokat az értékeket is amivel felül kell írni a mostaniakat. Mivel egy ügyön belül több feladat is lehet, azonban biztosan meghatározható, hogy számunkra az első értékek fognak kelleni, így a meghatározott partíció értékének egyet adunk.

A kód segítségével ki tudtuk gyűjteni egy segéd táblába azokat az elemeket, amiket változtatnunk kell azokkal az értékekkel együtt, amivel felül kell írni őket. Ezt követően már csak annyi volt a dolgunk, hogy egy ciklus segítségével végig menünk a tábla elemein, és az adatok frissítése megtörténjen. Fontos a script lefutása végén a változtatások commit parancs segítségével feltöltésre kerüljenek. Különben a szerveren nem lesz érzékelhető a változás, csak a lokális adatbázison.

### 7.4.2. Hosszútávú megoldás implementálása Siebelben

Hosszú távon viszont nem célratörő, ha havonta egyszer egy scriptet manuálisan le kell futtatni, így egy olyan megoldáson kezdtem el dolgozni, ami megakadályozza az ilyen esetek létrejöttét. Össze kellett gyűjteni, azokat a bejövő dokumentum csatolmányokat, amik nem indíthatnak eljárást. A három fajta tértivevényen (FileNet Tértivevény, Tértivevény, E-Tértivevény) kívül még Egyéb dokumentumok sem indíthattak eljárást. A szakmán belül ezzel kapcsolatban nem volt teljes az egyetértés, így egy olyan megoldást java-

```
--UPDATE EGÉSZ TÁBLÁRA
DECLARE
    v_ROWCOUNT NUMBER:=0;
BEGIN
    FOR B_CUR IN
            (
                SELECT TERTI_RM_17040_2022_03_09.UGY_ID, FELADAT_ERKEZ_DATUM
                FROM APP_TECH.TERTI_RM_17040_2022_03_09
                WHERE TERTI_RM_17040_2022_03_09.UGY_ID != '1-3CY8QF6'
            )
    LOOP
            UPDATE SIEBEL.S_CASE CAS
            SET
                CAS.X_REGISTRATION_DATE = B_CUR.FELADAT_ERKEZ_DATUM
            WHERE CAS.ROW_ID = B_CUR.UGY_ID;

            v_ROWCOUNT := v_ROWCOUNT + 1;

            IF MOD(V_ROWCOUNT, 1000) = 0 THEN COMMIT; END IF;
    END LOOP;
    COMMIT;
END;
```

12. ábra. SQL frissítés

soltam, aminek következtében, ha ez a lista módosul nem szükséges a kódrészletbe belenyúlni, hanem ezeket a csatolmány típusokat, amik nem indíthatnak eljárást, egy érték listában tároljuk le.

Ehhez a metódusban ki kellett gyűjteni azokat az értékeket, amik az érték listában szerepelnek. Ehhez példányosítanom kellett az érték lista Business Component-et, és aktiválnom a 'Value', hogy az értékek belekerülhessenek a kereséséi kifejezésbe. Egy while ciklus segítségével végig megyünk az összes "X_NOT_PROC_START_INC_DOCTYPE" típusú érték listán, és amig van ilyen elem a kereséséi kifejezés string értékhez hozzáfűzzük a kifejezést. Először megvizsgáljuk, hogy ez a string üres-e. Amennyiben igen, akkor a += operátorral hozzáfűzzük a következő értéket: "[MAK Document Type] != ' " + bcListOfValue.GetFieldValue("Value") + " ' ". Amennyiben már van értéke a változónak, úgy a kifejezés elé kell raknunk egy 'AND' operátort: " AND [MAK Document Type] != '" + bcListOfValue.GetFieldValue("Value") + " ' ".

---

```
        TheApplication().LogDebug(this, 3, "Get List Of
            Values");
    boListOfValue = TheApplication().GetBusObject("List Of
        Values");
    bcListOfValue = boListOfValue.GetBusComp("List Of
```

```
    Values");
bcListOfValue.SetViewMode(AllView);
bcListOfValue.ActivateField("Value");
bcListOfValue.ClearToQuery();
bcListOfValue.SetSearchSpec("Type",
    "X_NOT_PROC_START_INC_DOC_TYPE");
bcListOfValue.ExecuteQuery(ForwardOnly);

bIsLOVRecord = bcListOfValue.FirstRecord();

while(bIsLOVRecord)
{
  if(sSearchExpr == "")
  {
    sSearchExpr += "[MAK Document Type] <> '" +
       bcListOfValue.GetFieldValue("Value") + "'";
  }
  else
  {
    sSearchExpr += " AND [MAK Document Type] <> '" +
       bcListOfValue.GetFieldValue("Value") + "'";
  }

  bIsLOVRecord = bcListOfValue.NextRecord();
}
```

A kifejezés létrehozása után példányosítanunk kell az Action Attachment Business Compo-nent-et, hiszen a bejövő dokumentumok csatolmányai, ezen a komponensen helyezkedik el, illetve a 'this' kulcsszóval megkaphatjuk, az aktuális bejövő dokumentum paramétereit is, mivel a script az Action Business Component-ben íródott. Ennek a bejövő dokumentumnak van egy 'Csatolmány Id' mezője, Így meg tudjuk keresni azt, vagy azokat a csatolmányokat, ami az aktuális bejövő dokumentumhoz kapcsolódik. Emellett meg tudjuk adni neki az általunk megalkotott kifejezést, miszerint csak akkor találjon értéket, amennyiben a csatolmány típusa nem egyenlő egyik értékkel sem az érték lista elmei közül. Amennyiben nem találunk elemet, úgy nincs ilyen csatolmány, és az eljárást el lehet indítani. Máskülönben hamis lesz a visszatérési érték, és ebben az esetben nem indul eljárás.

# 8. Tesztelés

Sajnos nincs automatizált tesztelési lehetőség, így minden fejlesztői tesztet a DEV felületen kellett elvégeznem. Ezek kiterjedtek az összes ügy típusra, amik alkalmazzák az új határidőszámítást. Új ügyeket kellett iktatni, hogy végig nézhessük a folyamatokat az első lépéstől az utolsóig. Ezen felül régebbi, más logikát használó ügyekben is végig kellett vinni a régi, illetve az új folyamatokat, hogy meggyőződhessünk, a felmenő rendszerrel bevezetett integrációról, az új, frissített logiába.

Ezek sajnos nem fedik le a teljes falóságot, így problémák későbbiekben is felmerülhetnek. Ennek esélyét sikerült minimalizálnunk, amennyire lehetséges, illetve a DEV környezetről adott TESZT Release után az üzlet is végig tudta tesztelni felhasználói szemmel az átadott fejlesztéseket és az azon belüli folyamatokat, illetve olyan teszteseteket is el tudtak végezni, amire én fejlesztőként nem voltam képes. Ilyen esetek például a kívülről érkező, más szervezetek által generált bejövő dokumentumok, mely feldolgozása befolyásoló tényező volt a fejlesztések során.

Amennyiben hiba adódik, annak felkutatása elég időigényes tud lenni, főleg ha nem fordít az adott programozó elég időt és energiát a logok elkészítésére. Mivel nincs automatizált tesztelésre lehetőség, így a szerver által készített LogDebug fájlokra vagyunk kénytelenek hagyatkozni. Ott ki tudjuk íratni egyes lépések és részeredmények visszatérési értékeit, így ellenőrizve azt a pontot, amikor a program futása már nem az ideális értékekkel dolgozik.

Több fajta logot is létre tud hozni a rendszer. A leggyakrabban nézett fájl, az adott Business Component, illetve Business Service által generál fájlok. Ezen kívül a kliens és létrehoz egy log fájlt. Ebben főleg az adatbázis lekérdezések szerepelnek. Akkor használjuk, amikor adatokat próbálunk szűrni és amennyiben nem az ideális csoport jelenik meg, itt meg tudjuk tekinteni, hogy mi okozza a hibát a lekérdezésben, és ebből következtetni tudunk, hogy az eScript kódban hol kell változtatni. Harmadik lehetőség az ErrorLog, ez csak hibák esetén fordul elő és a catch ágban szereplő logokat tartalmazza. Ezen felül kiírja a belépett felhasználó nevét, a hiba dátumát, a szervert, ahol a hiba keletkezett, illetve azt az objektumot, illetve metódust, ahol a hiba keletkezett. Ez gyakran félrevezető tud lenni, mert lehetséges, hogy a hiba egy másik metódusban jött létre, azonban csak egy másikban sikerült 'elkapni' azt.

A legtöbb esetben ezt a lehetőséget használjuk, ki a teszteléshez. A kódban elhelyezünk 3-as szintű LogDebug funkciókat, és minden egyes elágazásnál, ciklusnál és funkciónál kilogoljuk a számunkra érdekes értékeket. Azért használunk hármas szintű LogDebug-ot, hogy amennyiben a fejlesztés kikerül élesre, ezek a kódsorok nem futnak, le csak ha egyes szintű debugról van szó, így nem tele szemetelve az éles mappákat

fölöslegesen. Így a futtatás alatt meggyőződhetünk, hogy minden lépés a helyén van kezelve és nincs hiba a program futása során. Természetesen sohasem tudunk minden hibát kiszűrni manuális módszerekkel, azonban jelenleg ennél jobb módszer nincs a jelnlegi applikáció fejlesztése közben.

# 9. Továbbfejlesztési lehetőségek

A következő pontban az általam észrevett fejlesztési lehetőségek kerülnek bemutatásra. Ezek nem minden esetben csak az általam készített fejlesztésekre fókuszálnak, hanem általánosságban is használható, a fejlesztést és tesztelést megkönnyítő lehetőségekről.

## 9.1. Automatizált tesztelés

Talán a legfontosabb és legégetőbb továbbfejlesztési lehetőség a teszt automatizálás lenne. Ez természetesen nem csak a bemutatott fejlesztéseket érintené, hanem az egész TÉBA rendszerre pozitív hatást gyakorolna. Ez a Siebelben nehezen, de megvalósítható lenne. Amennyiben ez a probléma leküzdésre kerülne, úgy akár át lehetne állni Teszt-vezérelt fejlesztésre(Test-driven development, röviden TDD).

## 9.2. További automatizáció

Továbbra is az emberi tényező a legnagyobb hibaforrás egy applikáció fejlesztése során. A jelenlegi feladatom során is nagy hangsúlyt fektettem, hogy ez a tényező minél minimálisabb lehessen, azonban ezen lehetőségek közül nem mindent sikerült kiaknázni üzleti kérésre. A fejlesztések során bemutattaknál sokszor emlegettük az ügyintézőt, hogy még mindig nagy felelősség van rajtuk. Általánosságban kijelenthetjük, emberi faktor különböző okok miatt hibák kialakulásához vezethet a rendszer nem megfelelő használása miatt. A megalkotott funkciókon belül a tényállás tisztázása, illetve az eljárás felfüggesztés/szüneteltetés az a rész, aminél még csökkenteni lehet az ügyintézők feladatkörét. Erre egy lehetséges megoldás, hogy az ebben az esetben beküldött dokumentumnál, ami például hiányt pótol, felveszünk egy gombot, és amennyiben a pótlás teljes és a személynek nincs több teendője megadhatnánk egy gombot, ami elfogadja. Ennek megnyomásának következtében az ügyön belül a beállított flag kivételre kerül és ennek hatására lefutnak azok a kódok, amik a határidőszámítást végzik és javítják a megfelelő dátumokat.

# 10. Összefoglalás

A szakdolgozatom témája, az evoCRM-nél töltött gyakornoki, és alkalmazotti ottlétem során valósulhatott meg. A Magyar Államkincstárnak volt egy igénye a határidőszámítás refektorálására, mivel nem minden működött úgy, ahogy azt ők elvárták, illetve egyéb automatizációs igények is felmerültek a projekt kapcsán. Ennek kidolgozásával és lefejlesztésével én lettem megbízva. Mivel a TÉBA rendszer egy folyamatos fejlesztést és supportot igényló nagyobb volumenű, bonyolultabb rendszer, így a fejlesztési környezett adott volt, így maradt az Oracle Siebel CRM rendszer.

Az első és egyik legfontosabb feladatom az ügyféllel történő megbeszélések voltak a kívánt fejlesztések kapcsán, hogy az jogszabályoknak megfelelő legyen. Készítenem kellett egy rendszertervet, ami a megvalósítandó fejlesztések minél részletesebb leírását tartalmazza. A cél egy olyan dokumentum létrehozása volt, ami nem fejlesztő személynek is értelmezhető, azonban a leírás alapján egy fejlesztő képes legyen lekódolni az igényeket. Ez szerintem a 'Tervezés' szekcióban is megfigyelhető volt. Az igények elég sokrétűek voltak, de végső soron összefüggésben álltak egymással, hiszen sablonok importálása nélkül, nem lehet ügyeket, eljárásokat indítani, kezelni, illetve lezárni.

Ahogy az egész TÉBA rendszert, úgy az általam készített fejlesztéseket is ügyintézők használják, így a fejlesztés során feltételeznem kellett, hogy a rendszert kevésbé ismerő felhasználók hibázhatnak, így széles körű hibakezeléssel el kellett érnem, hogy ezek ne történhessenek meg és a felhasználó egy hibaüzenetben értesüljön, az elrontott lépésekről. Ezen okok miatt, azokat a pontokat, ahol nem szükséges az ügyintéző revíziója, azokat a lépéseket, ahol lehetett automatizáltam, így tovább minimalizálva a hiba lehetőségét.

A fejlesztések 2022. január elején elkészültek és ki is kerültek a PROD környezetre, amit az ügyintézők a mai napon is használnak nap mint nap az eljárások kezeléséhez. Az éles szerverre kiadás után még kiegészítés nem került hozzá, azonban kiegészítési kérelmek bármely esetben érkezhetnek. Rendkívül érdekes és izgalmas volt egy ilyen mélységű projekten dolgozni és rengeteg új tapasztalatra tettem szert akár az ügyféllel való kommunikációban, akár a fejlesztés megtervezésében és megvalósításában. Ugyan a használt technológia és script nyelv amiban rendszer íródott legacy besorolást kapna tőlem, de ezek a tapasztalatok, amiket sikerült megszerezni, minden más fejlesztési területen is alkalmazható lesz a jövőben.

# 11. Summary

The topic of my dissertation may have been realized during my internship and employee stay at evoCRM. The Hungarian State Treasury had a need to reflect on the calculation of the deadline, as not everything worked as expected and other automation needs arose in connection with the project. I was tasked with developing a solution for the issue. As the TÉBA system is a larger, more complex system that requires continuous development and support, the development environment was specific, so the Oracle Siebel CRM system remained.

My first and one of the most important tasks was to discuss with the clients about the desired improvements to comply with the law. I had to create a system plan, which contains as detailed a description as possible of the improvements to be implemented. The goal was to create a document that could be interpreted by a non-developer, but based on the description, a developer would be able to encode the needs. I think this was also observed in the 'Planning' section. The demands were quite diverse, but they were ultimately interrelated, as without importing templates, cases, and procedures could not be initiated, managed, or closed.

Like the whole TÉBA system, the improvements I have made are used by administrators, so during the development, I had to assume that users who were less familiar with the system processes, could make mistakes, so I had to deal with them with a wide range of error handling so that the user would be notified in details about the wrong steps. For these reasons, the points where no administrators envision is required, the steps where I could have automated, thus further minimizing the possibility of error.

The developments were completed in early January 2022 and were released to the PROD environment, and it is still used by administrators today to manage procedures. No additions have been added to the live server after the release, but add-on requests can be received in any case. It was extremely interesting and exciting to work on a project of this depth and I gained a lot of new experience either in communicating with the client or in planning and implementing the development. Although the technology and scripting language used in which a system is written, would be a legacy environment to my mind, but these experiences that I have gained will be applicable to all other areas of development in the future.

# Hivatkozások


[1] Kornél, R. *Biztonságos elektronikus kézbesítés*, Híradástechnika, LXVI. évf. 2011/1. szám, (2011).

[2] Máté B.: *CRM*, (https://matebalazs.hu/crm.html), utoljára megtekintve: 2021.12.06.

[3] Thompson, B. *What is CRM.*, The Customer Relationship Management Primer. What you need to know to get started., (2002).

[4] Siebel, *C. R. M.*, Oracle Engineered Systems., (2013).

[5] Hermann, S. A., *Best-of-breed versus integrated systems*, American Journal of Health-System Pharmacy, 67(17), 67(17), (2010).

[6] Hansal A., Siebel CRM 8 installation and management, Packt Publishing Ltd., (2010).

[7] Hansal A., Oracle Siebel CRM 8 Developer's Handbook, Packt Publishing Ltd., (2011).

[8] Siebel: *GetBusComp Method*, (https://docs.oracle.com/cd/B40099_02/books/OIRef/OIRefInterfaceRef153.html), utoljára megtekintve: 2022.01.26.

[9] Siebel: *InvokeMethod Method*, (https://docs.oracle.com/cd/B40099_02/books/OIRef/OIRefInterfaceRef54.html), utoljára megtekintve: 2022.03.02.

# Ábrák jegyzéke


1. eTértivevény folyamatokhoz kapcsolódó műveletek . . . . . 8
2. Siebel multi-tier architektúrája . . . . . 11
3. Siebel Applikáció felépítése . . . . . 13
4. Siebel Applikáció felépítése . . . . . 14
5. TÉBA tervezett felülete . . . . . 16
6. TÉBA tervezett felülete . . . . . 19
7. TÉBA Release folyamatábra . . . . . 23
8. Business Service felhívása . . . . . 26
9. Sablonimportálás folyamatábra . . . . . 27
10. Appletek szerkesztése . . . . . 32
11. Kimenő dokumentum . . . . . 33
12. SQL frissítés . . . . . 37