



FelX Vállalatirányítási Rendszer

Lázár Marcell, Medve Adrián, Varsányi Barnabás
Nógrád Megyei Szakképzési Centrum Szent-Györgyi Albert Technikum
Szoftverfejlesztő és -tesztelő technikus (szakma azonosító: 5-0613-12-03)
2023.04.24.

Tartalom

Bevezetés	7
1. Felhasznált technológiák	8
1.1 PHP	8
1.2 HTML	8
1.3 CSS.....	8
1.4 JavaScript.....	8
1.5 .NET Keretrendszer.....	9
1.6 .NET C#	9
1.7 Visual Studio 2019 / Visual Studio 2022.....	9
1.8 Visual Studio Code	9
1.9 Android Studio	9
1.10 Flutter	10
1.11 XAMPP	10
1.12 Microsoft Word / Google Docs.....	10
1.13 MySQL.....	10
1.14 GitHub.....	11
1.15 Discord	11
1.16 Postman	11
1.17 OpenVPN	11
1.18 WinSCP.....	12
1.19 Xcode	12
1.20 Adobe Illustrator 2023	12
1.21 Adobe Photoshop 2023	12
2. Felhasználói dokumentáció	13
2.1 Webes alkalmazás	13
2.1.1 Kezdőoldal.....	13

2.1.2	Letöltések	14
2.1.3	Kezelőfelület.....	15
2.1.4	Vállalati kiválasztó felület.....	15
2.1.5	Felhasználó feladatai	16
2.1.6	Felhasználó adatai	18
2.1.7	Vállalat adminisztrációja	20
2.2	Mobil alkalmazás	23
2.2.1	Beléptető felület.....	23
2.2.2	Főoldal.....	24
2.2.3	Feladatok menüpont	26
2.2.4	Adminisztráció menüpont	29
2.2.5	Felhasználói adatak menüpont	33
2.2.6	Fiók menüpont.....	34
2.2.7	Beállítások menüpont	38
2.2.8	Kijelentkezés menüpont	40
2.3	Asztali alkalmazás.....	41
2.3.1	Bejelentkezési felület.....	41
2.3.2	Kezelőpult.....	42
2.3.3	Adminisztráció menüpont	44
2.3.4	Saját fiók menüpont.....	52
2.3.5	Feladat menüpont	58
2.3.6	Egyéb menüpont	61
2.3.7	Kijelentkezés menüpont	65
3.	Fejlesztői dokumentáció	66
3.1	API	66
3.1.1	Adatbázishoz történő csatlakozás.....	67
3.1.2	Szerveridő.....	70

3.1.3	Adatvédelmi – jogi nyilatkozat	73
3.1.4	Jogosultság ellenőrzése felhasználónév és jelszó párossal.....	73
3.1.5	Token ellenőrző rendszer	78
3.1.6	Összes ország-adat megjelenítése.....	79
3.1.7	Verziószámok.....	79
3.1.8	QR kód generátor	80
3.1.9	SSO hitelesítés.....	80
3.1.10	Jogosultság ellenőrzése token alapján	81
3.1.11	Vállalati adatkezelés	82
3.1.12	Feladatok kezelése	88
3.1.13	Felhasználói adatok és fiókkezelés	92
3.1.14	Rendszernapló.....	97
3.2	Webes alkalmazás	98
3.2.1	Globális fájlok	98
3.2.2	Kezdőoldal.....	100
3.2.3	Kezelőfelület.....	101
3.2.4	Bejelentkezés	102
3.2.5	Vállalat választás	102
3.2.6	Feladatok kezelése.....	102
3.2.7	Feladatok rögzítése	103
3.2.8	Felhasználói adatok	104
3.2.9	Biztonsági kulcs generálás	105
3.2.10	Jelszómódosítás	105
3.2.11	Aktív munkamenetek.....	105
3.2.12	Vállalat adminisztrációja	106
3.2.13	Felhasználók kezelése.....	106
3.2.14	Munkacsoportok kezelése.....	106

3.2.15	Kijelentkezés.....	106
3.3	Mobil alkalmazás	107
3.3.1	Main.dart állomány.....	107
3.3.2	Models mappa	109
3.3.3	Bejelentkezés ellenőrzése	110
3.3.4	Bejelentkezés	110
3.3.5	Főoldal	111
3.3.6	Feladatok	112
3.3.7	Adminisztráció	113
3.3.8	Felhasználói adatok	113
3.3.9	Fiók	114
3.3.10	Beállítások	114
3.3.11	Kijelentkezés.....	114
3.3.12	Utils mappa.....	115
3.4	Asztali alkalmazás.....	116
3.4.1	Cryptography mappa	117
3.4.2	DataCollection mappa	118
3.4.3	DataStructure mappa	121
3.4.4	Design mappa	122
3.4.5	ErrorHandler mappa	123
3.4.6	Bejelentkezés	124
3.4.7	Kezelőpult.....	125
3.4.8	Vállalat kiválasztása	127
3.4.9	Vállalat adatai	128
3.4.10	Vállalati felhasználók kezelése.....	128
3.4.11	Kiválasztott vállalati felhasználó kezelése	129
3.4.12	Új FMS felhasználó létrehozása	130

3.4.13	Regisztrált FMS felhasználó kezelése	132
3.4.14	Vállalati munkacsoporthoz köthető felhasználó kezelése	133
3.4.15	Saját fiók adatainak megtekintése és kezelése.....	136
3.4.16	Jelszó módosítása.....	137
3.4.17	Jogosultságok.....	138
3.4.18	Aktív munkamenetek.....	138
3.4.19	Biztonság	138
3.4.20	Feladatok.....	139
3.4.21	Kiválasztott feladatok információi	139
3.4.22	Új feladat hozzáadása	140
3.4.23	QR kód megjelenítése.....	141
3.4.24	Változások	141
3.4.25	Diagnosztika	141
3.4.26	Névjegy.....	142
3.4.27	Kijelentkezés a rendszerből	142
4.	Az adatbázis célja	143
4.1	Tervezés megkezdése.....	143
4.2	Tervezési lépések	143
4.3	Egyedtípusok és táblák meghatározása	144
4.3.1	Országok (countries)	144
4.3.2	Vállalatok (companies).....	144
4.3.3	Felhasználók (users)	145
4.3.4	Felhasználói csoportok (usergroups).....	146
4.3.5	Vállalati szerepkörök (roles)	146
4.3.6	Felhasználói szerepkörök hozzáférési (accesses)	148
4.3.7	Felhasználói munkamenetek (sessions).....	148
4.3.8	Tevékenységek (logs).....	149

4.3.9	Felhasználói biztonsági kulcsok (secretkeys).....	150
4.3.10	Feladatok (tasks).....	150
4.3.11	Feladatokhoz tartozó üzenetek, megjegyzések (taskinfos).....	151
4.3.12	Munkacsoporthoz rendelt felhasználók (taskgroupusers)	151
4.3.13	Munkacsoporthoz rendelt felhasználók (taskgroupusers)	152
4.4	Kapcsolatok meghatározása	152
4.5	N:M kapcsolatok felbontása.....	153
4.6	Minta adatok.....	154
4.6.1	Országok (countries)	154
4.6.2	Vállalatok (companies).....	154
4.6.3	Felhasználók (users)	154
4.6.4	Felhasználói csoportok (usergroups)	155
4.6.5	Vállalati szerepkörök (roles)	155
4.6.6	Felhasználói szerepkörök hozzáférései (accesses)	156
4.6.7	Felhasználói munkamenetek (sessions).....	156
4.6.8	Tevékenységek (logs).....	157
4.6.9	Felhasználói biztonsági kulcsok (secretkeys).....	157
4.6.10	Feladatok (tasks).....	157
4.6.11	Feladatokhoz tartozó üzenetek, megjegyzések (taskinfos).....	157
4.6.12	Munkacsoporthoz rendelt felhasználók (taskgroupusers)	158
4.6.13	Munkacsoporthoz rendelt felhasználók (taskgroupusers)	158
4.7	Adatbázis diagram.....	159
4.8	Adatbázis továbbfejlesztési lehetőségek	160
4.9	Megoldott hibák	160
5.	Összefoglalás	161
	Források	162
	Ábrajegyzék.....	165

Bevezetés

A **FELX Management System**, rövidebb nevén „FMS”, egy korszerű technológiákkal felépített, átgondolt vállalatirányítási rendszer, amely a KKV kisvállalati felhasználóktól kezdve a nagy cégek számára is átfogó megoldást biztosít a hétköznapokban.

A keretrendszerhez biztosított modulok további lehetőségeket adnak a vállalatoknak, amelyek a szerződésben meghatározott keretrendszerhez csatolt programcsomagok által alkotnak közösen, egy komplett vállalatirányítási rendszert.

A csapatunkból mindenki dolgozott már valamilyen vállalatnál, amely hozzájárult a vállalatirányítási rendszerek kiismeréséhez és hibáinak felfedezésére. A felületek sokszor nem felhasználóbarátak, leterheltek, helytelen adatokat és üzeneteket jelenítenek meg a felhasználók számára, ami nem lehet egyértelmű egy alkalmazott számára.

Célul tüztük ki, hogy egy olyan egységes platformot hozzunk létre, ahol az általunk más rendszerekben felfedezett hiányosságokra megoldást nyújtsunk, és egy gördülékeny, letisztult felületet teremtsünk meg a vállalatok számára.

A csapatunk 3 főből áll. A csapattagok megválasztásakor figyelembe vettük az eddig megszerzett tudásunk szintjét. Mivel mindenki foglalkoztunk már PHP programozással, és a tudásunkat bizonyítottuk egy komplett CRUD (Create Read Update Delete) feladat megoldása kapcsán, amely képes az adatokat olvasni, létrehozni, frissíteni és törölni, így lehetőséget kaptunk önálló projektmunkára, amelyben egy beléptető rendszert kellett megvalósítanunk. Így mi már decemberben elkezdtünk a záró projektünk tervezését, és kivitelezését.

A feladatokat érdeklődési körünknek megfelelően osztottuk fel. Varsányi Barnabás készítette a Windows alkalmazást, és az API-t (Application Programming Interface), amely összeköttetést biztosít az alkalmazás és az adatbázis között, biztonságos kapcsolattal az interfész kihasználva. Medve Adrián az Android és IOS mobilalkalmazás fejlesztésének felelőse lett. Itt jegyeznénk meg azt is, hogy az ehhez szükséges tudást teljesen önállóan szerezte meg, mivel nem volt a tananyag része és nem került oktatásra. Szabad idejében oktató videókból tanulta meg a szükséges technológiákat. Lázár Marcell a webes megjelenés tervezője, és kivitelezője volt a csapatban.

A projektünk publikus weboldala megtalálható a <https://felx.hu/> hivatalos keresztül, ahol összefoglaltuk a rendszer működését, helyet kapott a rendszer alapvető

bemutatása, illetve az asztali és mobil alkalmazások elérési helye. Rendszerhiba esetén a <https://status.felx.hu/> weboldalon keresztül lehet további információkat olvasni a FELX szerverének állapotával kapcsolatosan.

A csoportmunka megfelelő adattárolásának és verziókövetésének céljából a Githubon létrehozott privát „repository” segítségével folytattuk a fejlesztést az iskolából, illetve otthonról egyaránt. A platformra bejelentkezés után érhető el a „repo” a <https://github.com/varsanyib/fms> hivatkozáson keresztül.

1. Felhasznált technológiák

1.1 PHP

A dinamikus weboldalak elkészítéséhez használt HTML preprocesszor. A PHP segítségével lehetővé válik a weboldalakon történő űrlapküldés, adatbázis-kezelés, bejelentkezési rendszerek, kosárrendszerek, és egyéb felettebb folyamatok implementálása. A PHP nyelv ötvözi a szerveroldali és az adatbázis-kezelő funkciókat, amelyek összetettebb alkalmazások fejlesztéséhez szükségesek, és lehetővé teszi az adatokat valós időben feldolgozni és megjeleníteni az ügyfeleknek.

1.2 HTML

A HTML nagyon egyszerű, XML alapú nyelv, könnyen tanulható, és nagyon széles körben használt, ezáltal a webfejlesztők számára alapvető tudásnak számít. A HTML nyelv és a weboldalak létrehozásához együtt a CSS és JavaScript nyelvekkel alkotja a három alapvető technológiát a dinamikus weboldalak létrehozásához.

1.3 CSS

A CSS egy hasznos eszköz a weboldalak megjelenésének beállítására, és a weboldalak szélesebb körben való használatához szükséges kompatibilitás biztosítására, ezzel javítva a weboldalak használhatóságát és megjelenését.

1.4 JavaScript

A JavaScript segítségével lehetővé válik az oldalakon történő interaktivitás növelése, például animációk, grafikák, és egyéb eszközök beépítése. Emellett a JavaScript segítségével lehetővé válik az oldalakon történő adatgyűjtés és feldolgozás, valamint az oldalak működésének finomhangolása.

1.5 .NET Keretrendszer

A .NET keretrendszer egy Microsoft által fejlesztett platform, amely lehetővé teszi a programozók számára, hogy alkalmazásokat és szolgáltatásokat hozzanak létre a Windows operációs rendszeren és más platformokon is, például a Linuxon és az iOS-en.

A .NET keretrendszer támogatja a többnyelvű programozást, beleértve a C#, a Visual Basic, a F# és más nyelveket is. Emellett számos fejlesztői eszközt is biztosít, például az Visual Studio-t, amely kódkiegészítéssel segíti a fejlesztést.

1.6 .NET C#

A C# eredetileg a Microsoft által fejlesztett, mára nyílt forráskódúvá vált nyelv, jellemzői közé tartozik az objektum-orientált programozás, amely az egységbezárást, öröklés és sokalakúság főtulajdonságokat tartalmazza, illetve a típusbiztosítás, az automatikus memóriakezelés és a könnyű tesztelhetőség is kiemelkedő.

Emellett a C# nyelv jól összeegyeztethető más nyelvekkel, így lehetővé teszi a különböző platformokon futó alkalmazások integrációját is.

1.7 Visual Studio 2019 / Visual Studio 2022

A Visual Studio a Microsoft által fejlesztett, integrált fejlesztői környezet (IDE), amelyet számos programozási nyelvre és platformra használhatunk, például C#, C++, F#, C++, Python, JavaScript, HTML, CSS, és egyéb nyelvek. A Visual Studio segíti a fejlesztőket a kódírás, hibakeresés, tesztelés, profilozás, verziókezelés és egyéb fejlesztési feladatok végrehajtásában.

1.8 Visual Studio Code

Egy nyílt forráskódú, szövegszerkesztő alapú kód szerkesztő, amelyet szintén a Microsoft fejlesztett ki. A Visual Studio Code-ot sokféle programozási nyelv fejlesztésére használhatjuk, például JavaScript, TypeScript, C#, Python, PHP, HTML, CSS, és egyéb nyelvek.

1.9 Android Studio

Az Android Studio a Google által kifejlesztett Android SDK (Software Development Kit) használatával együtt nyújtja a szükséges eszközöket az Android alkalmazások fejlesztéséhez. Az Android Studio jellemzői közé tartozik a

kiterjeszthetőség, amely lehetővé teszi a felhasználó számára, hogy különböző kiegészítőket és plugineket telepítsen, amelyek segítenek a fejlesztésben.

1.10 Flutter

A Flutter egy nyílt forráskódú mobil alkalmazásfejlesztési keretrendszer, amelyet a Google fejlesztett ki. A Flutter segítségével iOS, Android és asztali platformokon is futtatható alkalmazásokat fejleszthetünk. Összességében tehát a Flutter egy nagyszerű eszköz mobil alkalmazások fejlesztésére, amely segíti a gyors fejlesztést, a natív érzékelést biztosító felhasználói felületet, valamint a több platformon való futtathatóságot.

1.11 XAMPP

A XAMPP képes lokálisan futtatni dinamikus webalkalmazásokat anélkül, hogy különösebb beállításokat kellene elvégezni, és segít megtakarítani az időt a projektek tesztelése és hibakeresése során. Az egyik előnye, hogy könnyen telepíthető és használható, ráadásul több platformra is elérhető, beleértve a Windows-ot, a Linux-ot és a macOS-t is. Egy másik előnye, hogy a XAMPP segítségével egyszerűen létrehozható egy adatbázis-kezelő panel, amely lehetővé teszi az adatbázisok könnyű használatát.

1.12 Microsoft Word / Google Docs

A Microsoft Word egy szövegszerkesztő program, amelyet a Microsoft fejlesztett ki. A Word segítségével dokumentumokat írhatunk, szerkeszthetünk, és formázhatunk, valamint gyorsan és egyszerűen oszthatjuk meg őket másokkal. A programot széles körben használják a szövegszerkesztéshez, a cikkíráshoz, a memoárok készítéséhez, a kutatási jelentésekhez, és még sok más felhasználáshoz.

1.13 MySQL

A MySQL egy nagyon elterjedt és népszerű adatbázis-kezelő rendszer, amely jól ismert a stabilitása, használhatósága és a kiszolgáló teljesítménye miatt, így a webfejlesztési projektekben gyakran használják. A MySQL könnyen összekapcsolható más technológiákkal, mint például a PHP, Python, Ruby, Java stb. amelyekkel a webalkalmazásokat fejlesztik.

1.14 GitHub

A GitHub egy web alapú platform, amely lehetővé teszi a fejlesztők számára, hogy tárolják, megosszák és fejlesszék a szoftvereiket. A fejlesztők képesek megfigyelni és rögzíteni a szoftverük módosításait, valamint később visszatérni egy korábbi verzióhoz, ha szükséges. A GitHub egy nagyon népszerű eszköz a fejlesztők körében, mert lehetővé teszi, hogy könnyedén tárolják és megosszák a kódjukat a világ számos pontján, valamint hogy együttműködjenek más fejlesztőkkel.

1.15 Discord

A Discord egy ingyenes VoIP-alkalmazás és digitális kommunikációs platform, amely lehetővé teszi a felhasználók számára, hogy egymással kapcsolatot tartsanak beszédhanggal, szövegesen vagy videóban. A felhasználók különböző szerverekre csatlakozhatnak, ahol csoportos beszélgetéseket, játékokat és egyéb tevékenységeket folytathatnak egymással.

A Discordot gyakran használják közösségek számára, hogy egymással kommunikáljanak játék közben, vagy közös projekteken dolgozzanak. Ezen kívül bármilyen célra használható, ahol fontos a hatékony és egyszerű online kommunikáció.

1.16 Postman

A Postman egy olyan szoftver, amely lehetővé teszi az API-k tesztelését és dokumentálását. Az alkalmazásban egyszerűen lehet HTTP kéréseket létrehozni, illetve tesztelni azok válaszát. A Postman egy nagyon rugalmas és könnyen használható eszköz, amely segítségével személyre lehet szabni az adatokat, fejléc mezőket, query paramétereket és a hitelesítést is. A tesztelő felület lehetővé teszi az eredmények vizuális megjelenítését, illetve az adatok automatikus kezelését. Az ingyenes verzióban maximum 3 fős munkacsoportok létrehozása is lehetséges, így könnyítve a csoportmunkát.

1.17 OpenVPN

Az OpenVPN egy nyílt forráskódú virtuális magánhálózat (VPN) kezelő szoftver, amely biztonságos kapcsolatot biztosít két vagy több számítógép között az interneten keresztül. A projektünkben a FELX szerverének belső hálózatára (intranet) csatlakoztunk távolról, így kezelve az adatbázist, illetve a szerver állományait.

1.18 WinSCP

A WinSCP egy ingyenes, nyílt forráskódú FTP, SFTP, SCP és WebDAV kliens Windows operációs rendszerekhez. Az alkalmazás lehetővé teszi a felhasználók számára a fájlok biztonságos másolását és átvitelét az interneten keresztül, valamint a távoli számítógépeken való fájlkezelést.

1.19 Xcode

Az Xcode egy integrált fejlesztői környezet (IDE) az Apple operációs rendszerhez, a macOS-hez. Az Xcode segítségével fejleszthetünk olyan alkalmazásokat, amelyek futnak az Apple operációs rendszerein, például az iOS-en, az iPadOS-en, a macOS-en és a watchOS-en.

1.20 Adobe Illustrator 2023

Az Adobe Illustrator 2023 egy vektorgrafikus szerkesztőprogram, amely lehetővé teszi a felhasználók számára, hogy vektor alapú képeket készítsenek, szerkesszenek és manipuláljanak.

A vektorgrafika olyan képformátum, amely lehetővé teszi a képek nagyítását és kicsinyítését anélkül, hogy elveszítenék a minőségüket.

A program használatával a felhasználók olyan grafikai elemeket hozhatnak létre, mint például logók, ikonok, infografikák és mások.

1.21 Adobe Photoshop 2023

Az Adobe Photoshop 2023 egy olyan grafikai szerkesztőprogram, amely lehetővé teszi a felhasználók számára a képek szerkesztését, készítését és módosítását.

A program számos különböző funkcióval rendelkezik, melyek megkönnyítik a felhasználó feladatát, például színek és kontrasztok javítása, szűrők és effektek alkalmazása, szöveg hozzáadása, stb...

Az Adobe Photoshop 2023 széles körben használatos digitális fotók szerkesztésére, tervezőgrafikai munkák elkészítésére és webfejlesztési projektekhez is.

2. Felhasználói dokumentáció

Az adatbázist, API interfészt és az asztali alkalmazást Varsányi Barnabás tervezte és készítette, az IOS és Android platformra való fejlesztést Medve Adrián végezte el. A weboldal megjelenéséért és fejlesztéséért Lázár Marcell volt a felelős. Mindhárom app esetében ugyanazokat a sablon színeket használtuk, így biztosítva az egységes megjelenést.

2.1 Webes alkalmazás

Egy olyan webes alapú kezelői felületet alakítottam ki, mely egyaránt használható asztali számítógépen és mobileszközökön is, ez az oldal reszponzivitásának köszönhető.

Az oldalon alkalmaztam minden eddig tudásom, szemantikus HTML elemek használata, megfelelően struktúrált és változókkal optimalizált CSS, továbbá függvényekbe szervezett JavaScript állományok.

2.1.1 Kezdőoldal

A főoldalon először JavaScripttel készített animációk teszik látványossá a weboldal megjelenését, de ugyanakkor nagyon ügyeltem arra, hogy ezek az animációk ne lassítsák meg az oldal betöltését.

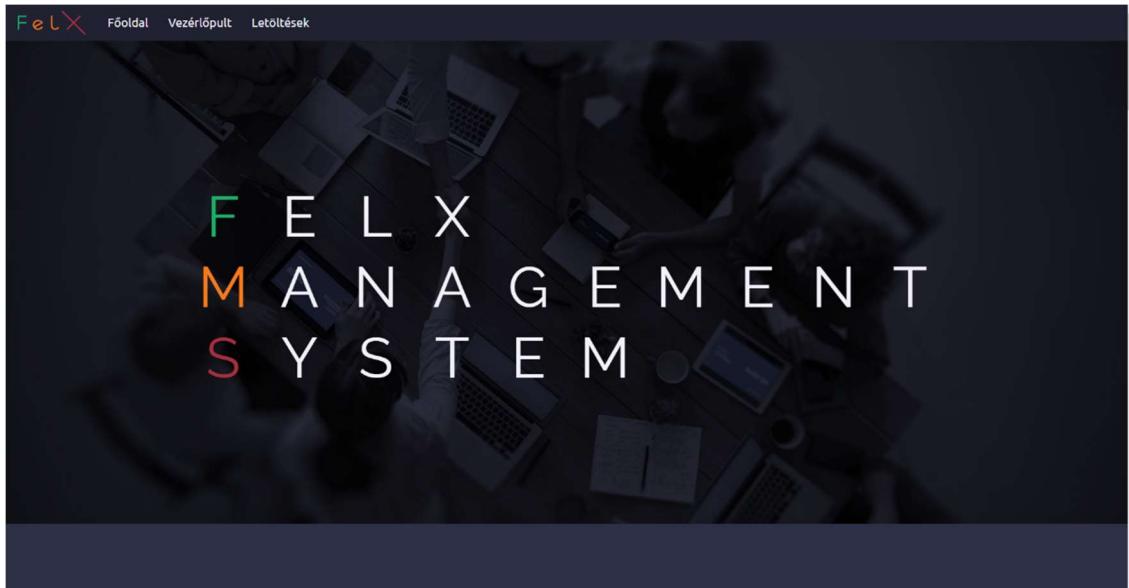
A megjelenés tervezésénél mindvégig szem előtt tartottam a reszponzivitást minden elem hozzáadásánál, így az oldal - bármilyen eszközről tekintjük meg, legyen az táblagép, vagy okostelefon - minden képernyőnélzetben kényelmesen olvasható. A menü hamburger ikonná alakul át, amely lenyílik, ha a felhasználó rákattint. A képek és a szöveg pedig egymás alá rendeződik át. Utánanéztünk a 2022. évi webdizájn trendeknek és láttuk, hogy igen divatos váltak a „parallax” effektek, ezért ezt is beépítettem a weboldalba.

A navigációs sávban megjelenő logó és az SVG képek Adobe Illustrator 2023-ban készültek.

Ahhoz, hogy a színek maximálisan harmonizáljanak, a „color.co” oldalon kerültek kiválasztásra és megtervezésre. A felhasznált színek mindegyikének 3 árnyalata került definiálására. Ezek a színek konzisztensek minden alkalmazásban.

A képeket a „pexels.com” oldalról töltöttük le, amelyek ingyenesen felhasználhatók. Az eredeti képek így is megváltoztatásra kerültek, Adobe Photoshop 2023-ban egy filter került rájuk, ami elsötétíti az eredeti harsány színeket, így egyszínű

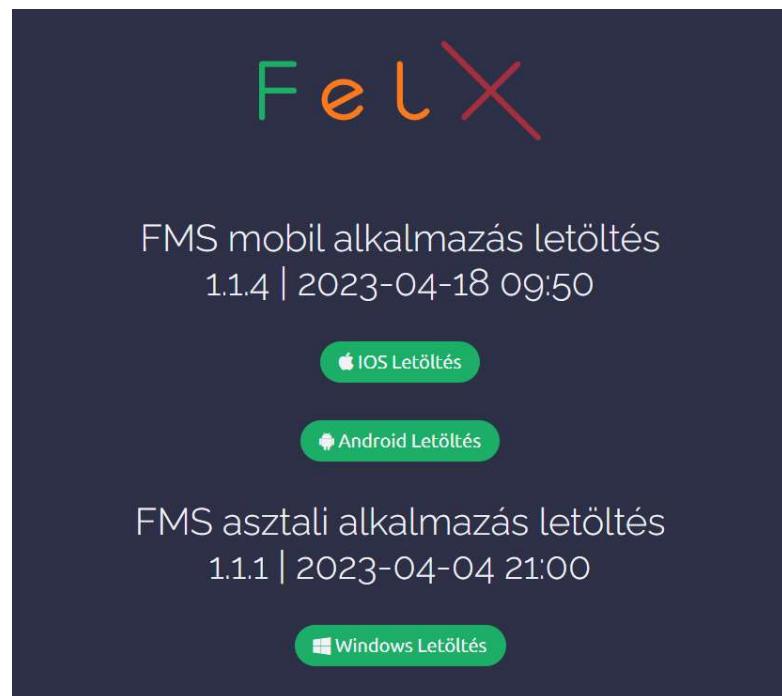
harmonizál az oldalon található színekkel, másrészt pedig nem vonja el a felhasználó figyelmét a tartalomról.



1) ábra: Főoldal

2.1.2 Letöltések

Az asztali- és mobilalkalmazások a „Letöltések” oldalról egységesen letölthetők, ezen az oldalon mindenkor az alkalmazások legfrissebb verzióját lehet megtalálni.



2) ábra: Letöltési oldal

2.1.3 Kezelőfelület

A kezelőfelület a menüből érhető el, amely az oldal közepén megjelenő belépési űrlapból áll, ahol a felhasználónak meg kell adnia az e-mail címét és a jelszavát. A beviteli mezők jobb oldalán egy vékony narancssárga ív jelenik meg, majd amikor a felhasználó elkezdi begépelni az e-mail címét, egészen addig pirosra vált, amíg a benne lévő szöveg nem tesz eleget a bemeneti mező feltételeinek, ennél fogva, amikor a böngésző érzékeli, hogy bekerült a megfelelő karakter, az ív színe átvált zöldre.

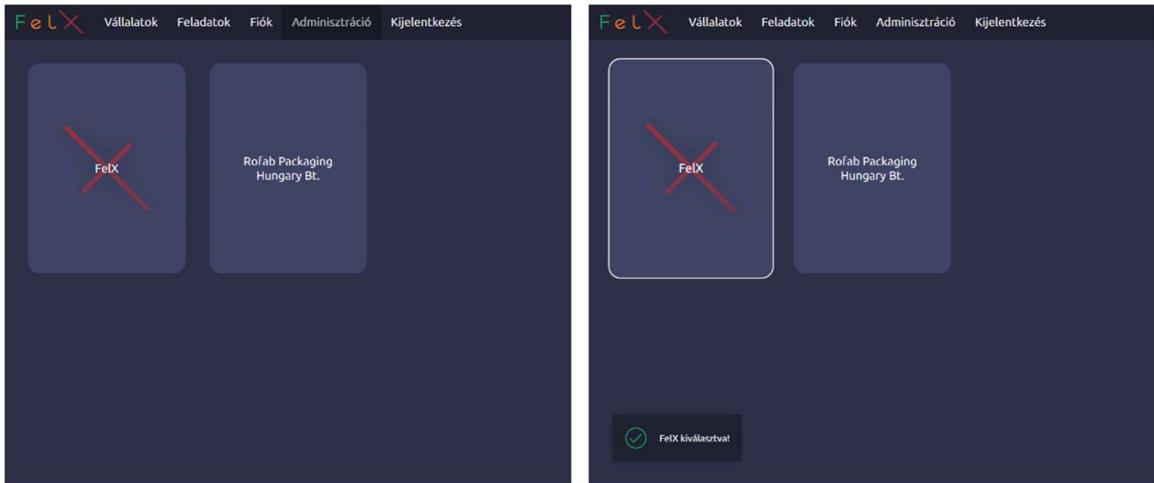
Amikor a felhasználó a bejelentkezés gombra kattint, egy felugró ablakban megjelenik a jogi nyilatkozat, alatta két lehetőséggel, vagy elfogadja a felhasználó vagy elutasítja. Ha a felhasználó elutasítja, akkor a bejelentkezés nem valósul meg. Amennyiben elfogadja, akkor kerül vizsgálatra a rendszerben, hogy érvényesek-e a megadott adatai.

Érvényes adatok megadása esetén egy üzenet jelenik meg, „Sikeress bejelentkezés”, zöld színnel, majd az oldal átirányítja a felhasználót a kezelőfelületre. Érvénytelen adatok megadása esetén a „Hibás felhasználónév vagy jelszó!” szöveg jelenik meg, bordó színnel.

2.1.4 Vállalati kiválasztó felület

A sikeres bejelentkezés után az oldalon megjelennek vállalatok, melyekhez a felhasználó hozzá van rendelve, ezekből több lehet, ha több helyen van egyszerre érdekeltsége. Több vállalat esetén ki tudja választani, hogy melyik vállalkozás ügyviteli rendszerébe szeretne belépni.

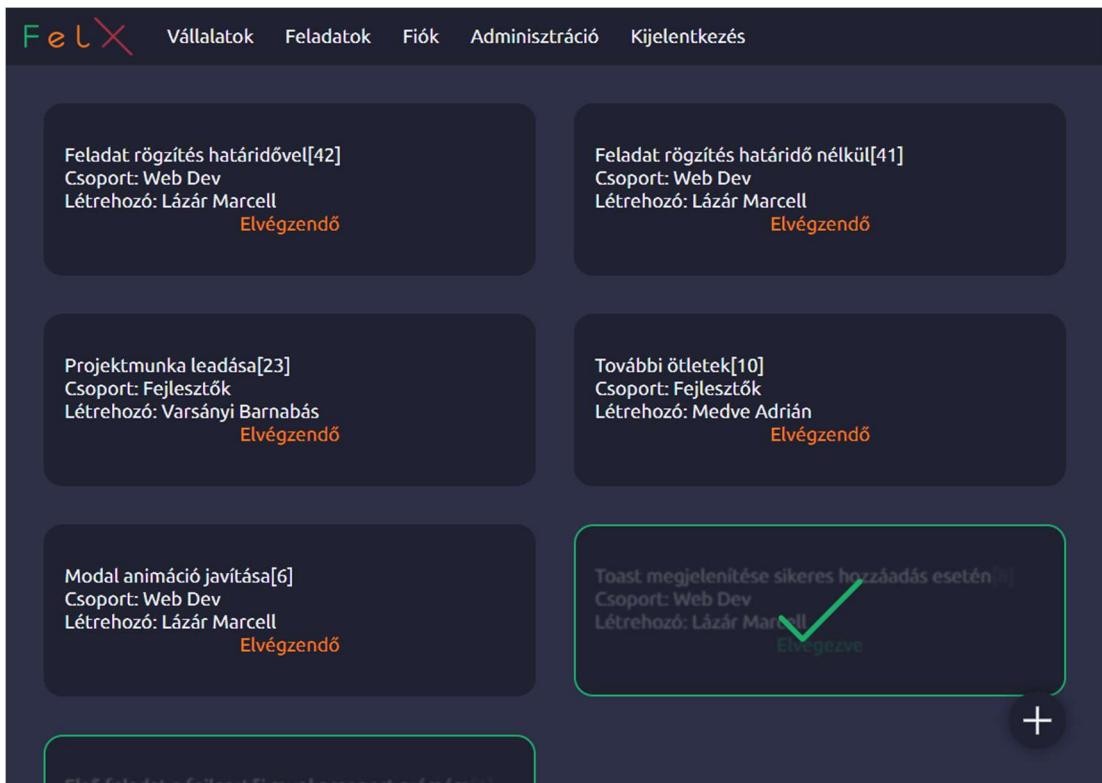
Abban az esetben, ha a felhasználó csupán egy vállalathoz van hozzárendelve, ez a választás automatikusan végbemegy, és az oldal átirányítja a felhasználót a feladatok kezelésére szolgáló felületre.



3) ábra: Vállalatok kiválasztása

2.1.5 Felhasználó feladatai

Egy vállalat kiválasztása után a felhasználó hozzáférést szerez az adott vállalaton belül a számára kiosztott feladatok megtekintésére, továbbá állapotaik módosítására. A már elvégzett feladatok zöld kerettel és bennük egy pipával vannak jelölve. Ha a felhasználó rendelkezik a munkacsoporton belüli megfelelő jogosultsággal, akkor a feladatok állapotának visszaállítására és esetlegesen a feladatok végezetes törlésére is van lehetősége, továbbá a felület jobb alsó sarkában megjelenik számára egy pluszjel, mellyel új feladatot rögzíthet.



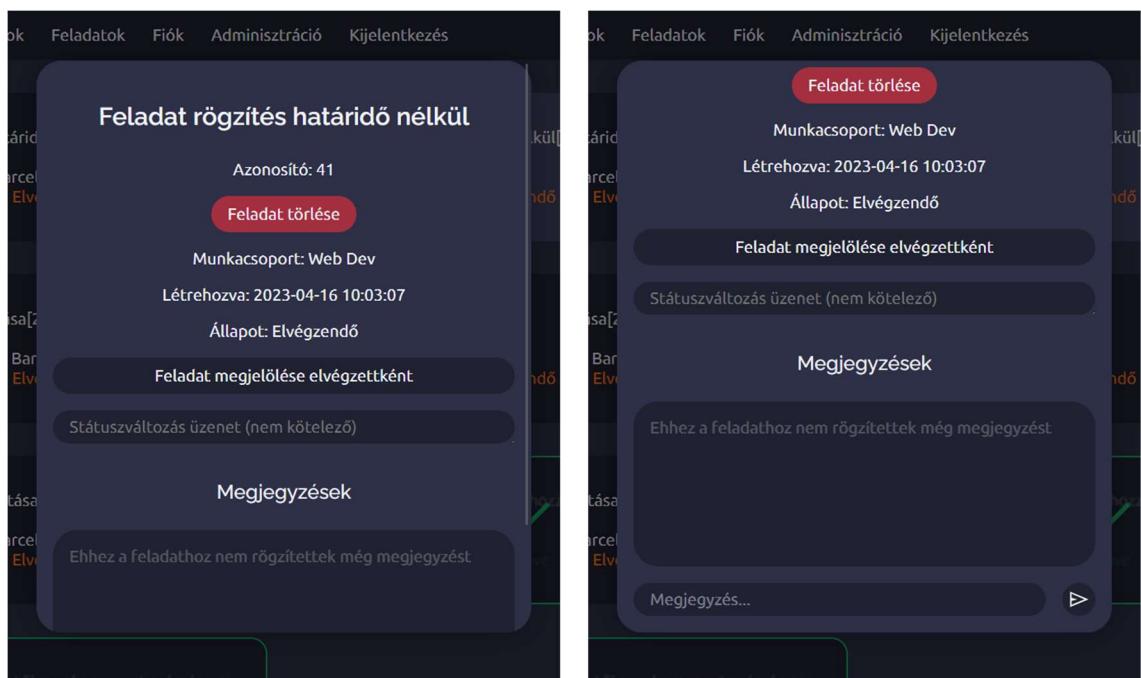
4) ábra: Feladatok megjelenése

Új feladat rögzítésénél kötelező megadnia a feladat címét és kiválasztania azt a munkacsoportot, amelyhez a feladatot kívánja rögzíteni, továbbá opcionálisan határidőt is állíthat be a feladat elvégzésére.



5) ábra: Feladat rögzítése

Egy feladatra kattintáskor felugró ablakban megjelennek annak részletei, többek között a feladat azonosítója, a rögzítő neve, a státusa és a hozzá fűzött megjegyzések. Ezen túl az ablakon belül tud a felhasználó megjegyzést is hozzáfűzni az adott feladathoz.



6) ábra: Feladatok részletei

2.1.6 Felhasználó adatai

A navigációból a belépés után elérhetővé válnak a felhasználói adatok a fiók menüpont alatt. A felhasználó adatai rácsos elrendezésben jelennek meg különböző beviteli mezők és az azokhoz tartozó címkék formájában, ez az elrendezés mobil nézetben egy oszlopra vált.

Azokat az adatokat, amelyeket nem lehet módosítani, mint a felhasználó azonosítója, illetve a felhasználó létrehozásának a dátuma, valamint a módosítások ideje, halványabb betűkkel jeleníti meg a weboldal, továbbá, ha a felhasználó ezek fölé helyezi a kurzort, az megváltozik egy tiltó jelé.

Az ország beállításánál egy lenyíló választó menü jelenik meg, így nem kell az országnevét begépeleine a felhasználónak.

Ha a felhasználó valamilyen adatot módosít, akkor a mentés gombra való kattintást követően az adatai automatikusan mentésre kerülnek, a sikeres mentésről egy felugró üzenet értesíti őket.

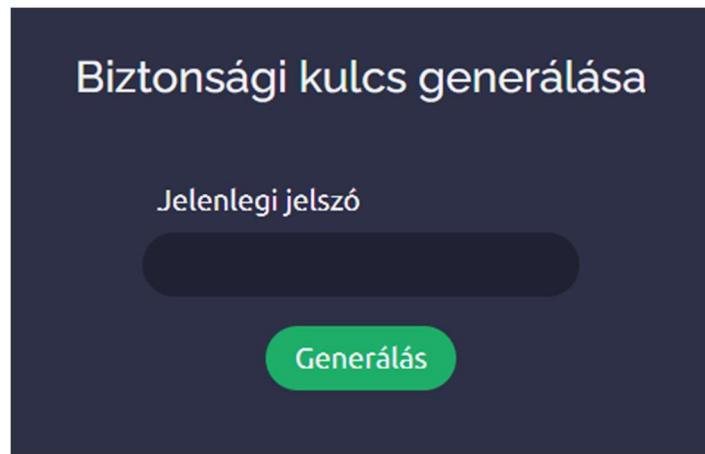
The screenshot shows a mobile application interface for managing user data. The title 'Felhasználó adatai' is at the top. Below it is a section titled 'Adatmódosítás'. The data is presented in a grid:

FMS azonosító	Teljes név	E-mail
1000000002	Lázár Marcell	aldynenn@felx.hu
Másodlagos e-mail	Telefonszám	Másodlagos telefonszám
(redacted)	(redacted)	(redacted)
Születési ország	Születési hely	Születési dátum
Magyarország [HU]	Balassagyarmat	11/02/2003
Nem	Születési név	Anyja leánykorai neve
Férfi	Lázár Marcell	(redacted)
Módosítva	Létrehozva	
04/18/2023 08:45:31 AM	03/16/2023 08:39:36 AM	

A green 'Mentés' (Save) button is located at the bottom right of the grid.

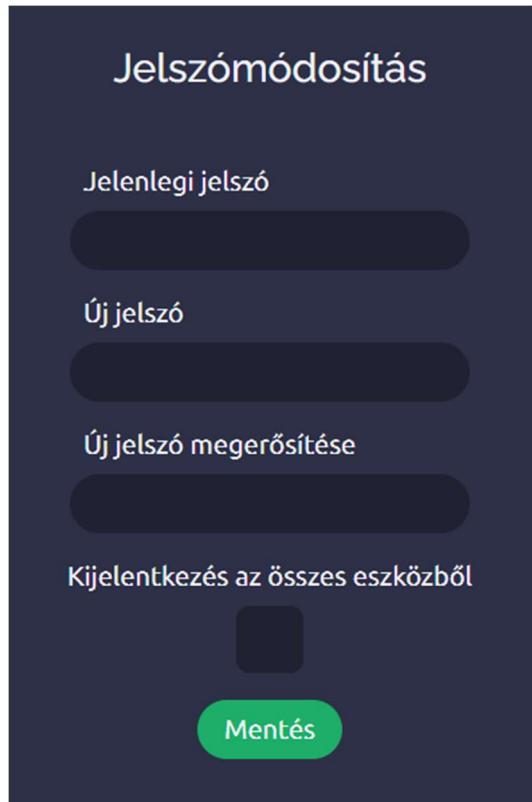
7) ábra: Felhasználó adatai

Abban az esetben, ha a felhasználót egy vállalathoz akarják rendelni, ezen az oldalon van lehetősége az egyszer használatos biztonsági kulcs generálására, amihez kötelezően meg kell adnia a jelszavát. A generálás gomb megnyomását követően egy felugró ablakban megjelenik a biztonsági kulcs és egy QR kód, ami szintén a kulcsot tartalmazza.



8) ábra: Biztonsági kulcs generálása

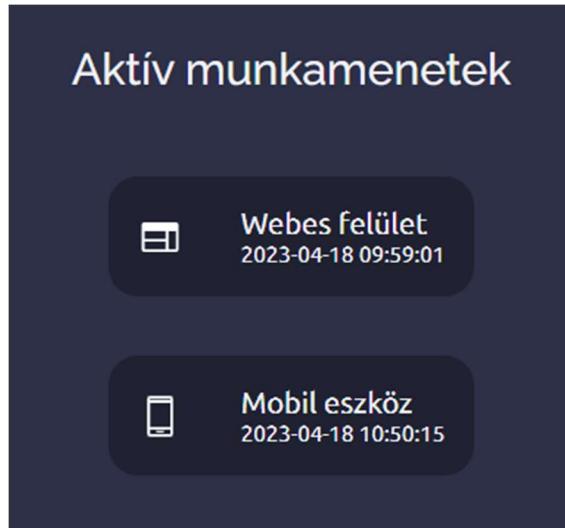
Szintén ezen az oldalon módosíthatja a belépési jelszavát, melyhez a jelenlegi jelszavát, illetve két alkalommal az új jelszavát kell megadnia, továbbá opcionálisan kijelentkeztetheti az összes munkamenetét, ha úgy érzi, hogy valaki hozzáférést nyert a fiókjához.



9) ábra: Jelszómódosítás

A weboldal alsó részén találhatóak az aktív munkamenetek, ahol látható, hogy milyen eszközökről van jelenleg aktív bejelentkezése a felhasználónak. A piktogramok alapján látható, hogy mobilról, weboldalról, vagy asztali alkalmazásból történt-e bejelentkezés. Mellette megjelenik az utolsó aktivitás dátuma és pontos ideje is. Az

eszközök típusa alatt egy-egy kijelentkezés gomb került elhelyezésre, melyek segítségével kiválasztható, mely eszkösről szeretnék kijelentkeztetni a felhasználót.



10) ábra: Aktív munkamenetek

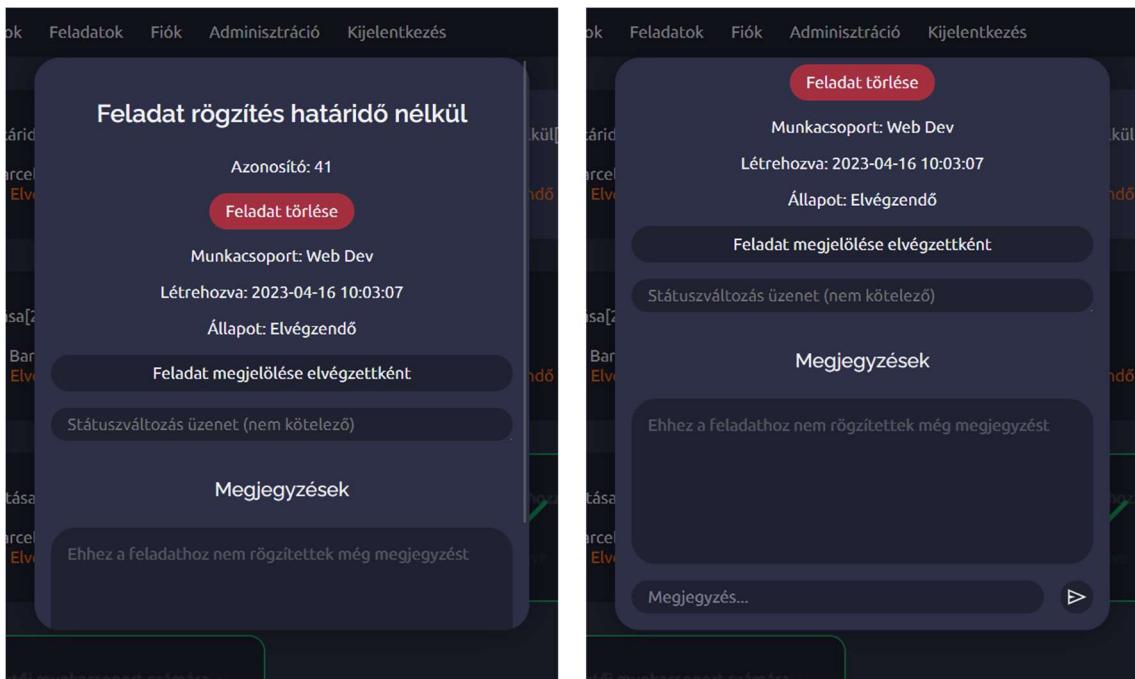
2.1.7 Vállalat adminisztrációja

A webes alkalmazás, utolsó menüpontjával a kiválasztott vállalkozás adminisztrációs felületét tudja elérni a felhasználó. A felület első felében a vállalathoz rendelt felhasználók karbantartására, új felhasználó létrehozására, illetve már az FMS rendszerben lévő felhasználók vállalathoz való hozzárendelésére van lehetőség.



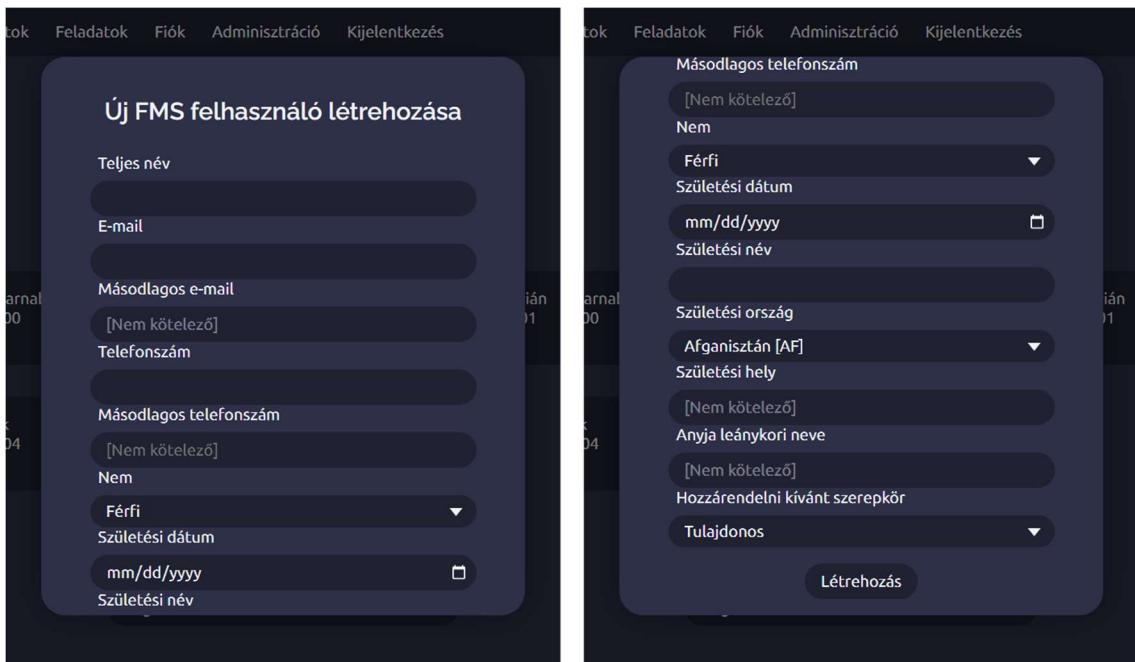
11) ábra: Felhasználók

Egy vállalathoz rendelt felhasználóra való kattintás után, a felhasználó szerepköreinek kezelése válik elérhetővé, itt hozzárendelhetünk vagy megvonhatunk tőle szerepköröket.



12) ábra: Felhasználó részletei

Az „Új FMS felhasználó létrehozása” gomb segítségével létrehozhatunk egy fiókot egy személy számára, aki még nincs jelen az FMS rendszerében, majd egy szerepkört is rendelhetünk hozzá, ezzel integrálva őt a kiválasztott vállalatba.



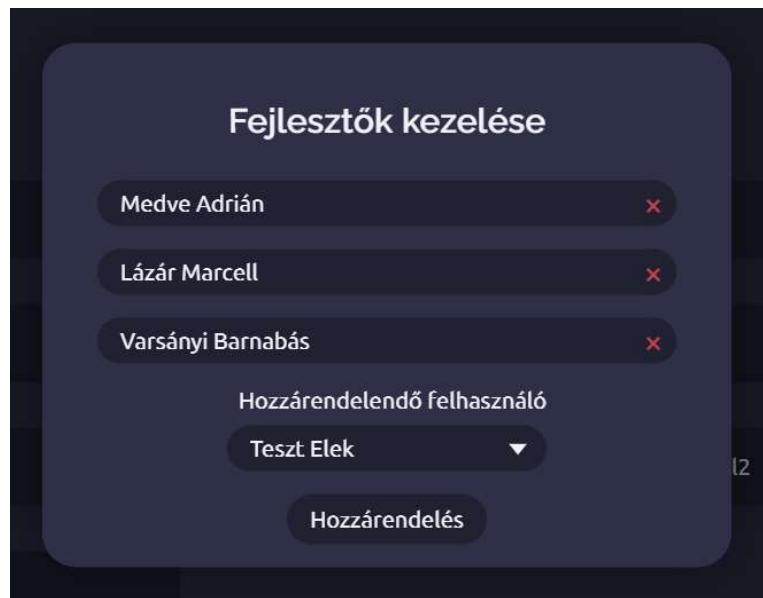
13) ábra: Új felhasználó létrehozása

A „Regisztrált FMS felhasználó hozzárendelése” gombra kattintva lehetőséget kapunk egy felhasználó által generált biztonsági kulcs megadására, mellyel hozzárendelhetjük az adott személyt az általunk kiválasztott vállalathoz.



14) ábra: Regisztrált felhasználó hozzárendelése a vállalathoz

A munkacsoportok hasonlóképp kezelhetők, mint a felhasználók, bármelyikre kattintva láthatjuk a hozzárendelt felhasználókat, és lehetőségünk van további felhasználók hozzárendelésére is.



15) ábra: Munkacsoport részletei, felhasználó hozzárendelése

Új munkacsoportot az „Új munkacsoport létrehozása” gombra kattintva lehet létrehozni, ahol csupán a létrehozandó munkacsoport nevét kell megadni.



16) ábra: *Új munkacsoport létrehozása*

Az összes fentebb említett művelet végrehajtása után, a felhasználót felugró üzenet értesíti, hogy sikeres volt-e a művelet vagy sem.

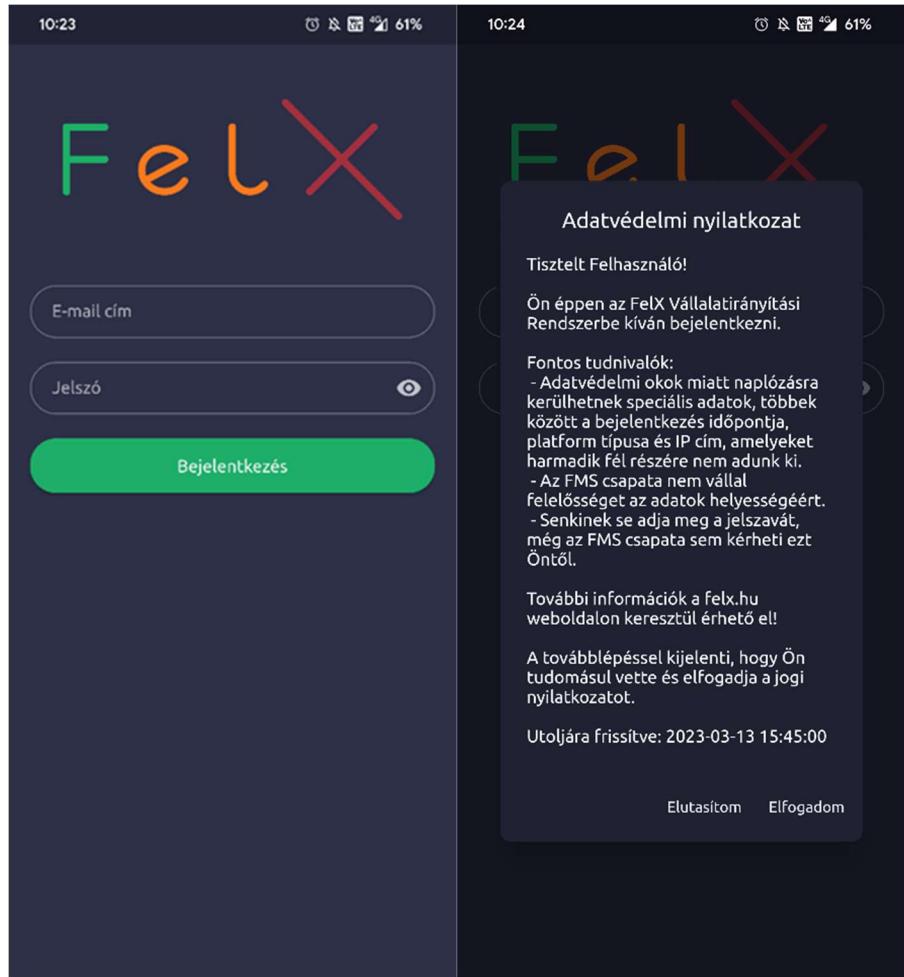
2.2 Mobil alkalmazás

Egy mobiltelefonra letölthető alkalmazást készítettem, hogy az ügyfelek könnyebben tudják kezelní a napi feladataikat. Egyaránt gondoltam az Androidos és Apple felhasználókra, így minden operációs rendszeren elérhető a program. Az applikáció megjelenése, és az összes funkció minden platformon elérhető teljes egészében. Jelenleg a weboldalon a „Letöltések” menüpontra kattintva tölthető le az alkalmazás (<https://felx.hu/downloads.php>).

Az alkalmazás az elindulása után a háttérben ellenőri, hogy elérhető-e újabb verzió. Ha van frissítés, akkor megkérdezi a felhasználót, hogy szeretné-e frissíteni az alkalmazást.

2.2.1 Beléptető felület

A beléptető felület teljes egészében megegyezik a webes megjelenéssel, ugyanazt a dizájnt alkalmaztam, amelyet Marcell megtervezett. Az e-mail cím és a helyes jelszó megadása után megjelenik a jogi nyilatkozat, amelyet el kell fogadni felhasználónak a továbblépéshez.

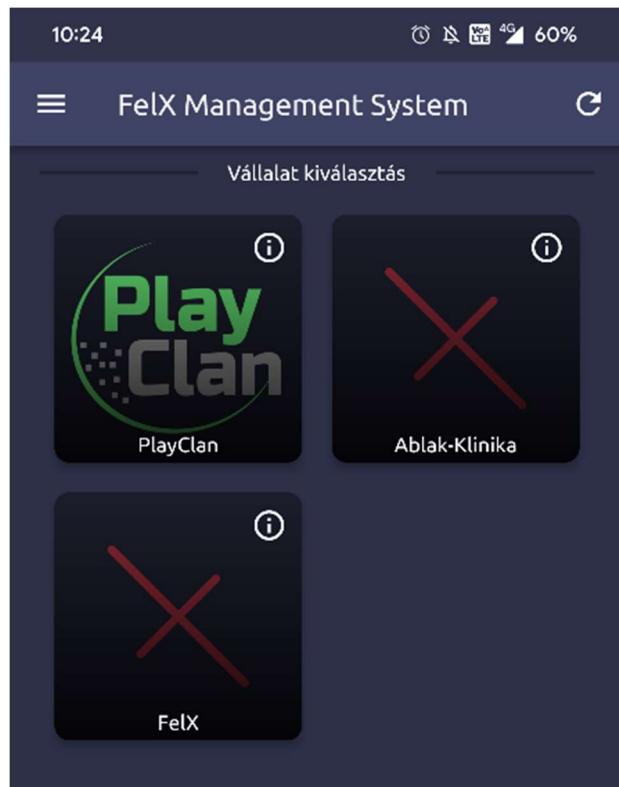


17) ábra: Bejelentkezési képernyő

Amennyiben a felhasználó már korábban bejelentkezett az alkalmazásba, akkor a program ellenőrzi, hogy van-e beállítva az eszközön biometrikus azonosítás (például ujjlenyomat, arcfelismerés), amennyiben igen, a sikeres hitelesítés után automatikusan tovább lesz irányítva az alkalmazás főoldalára.

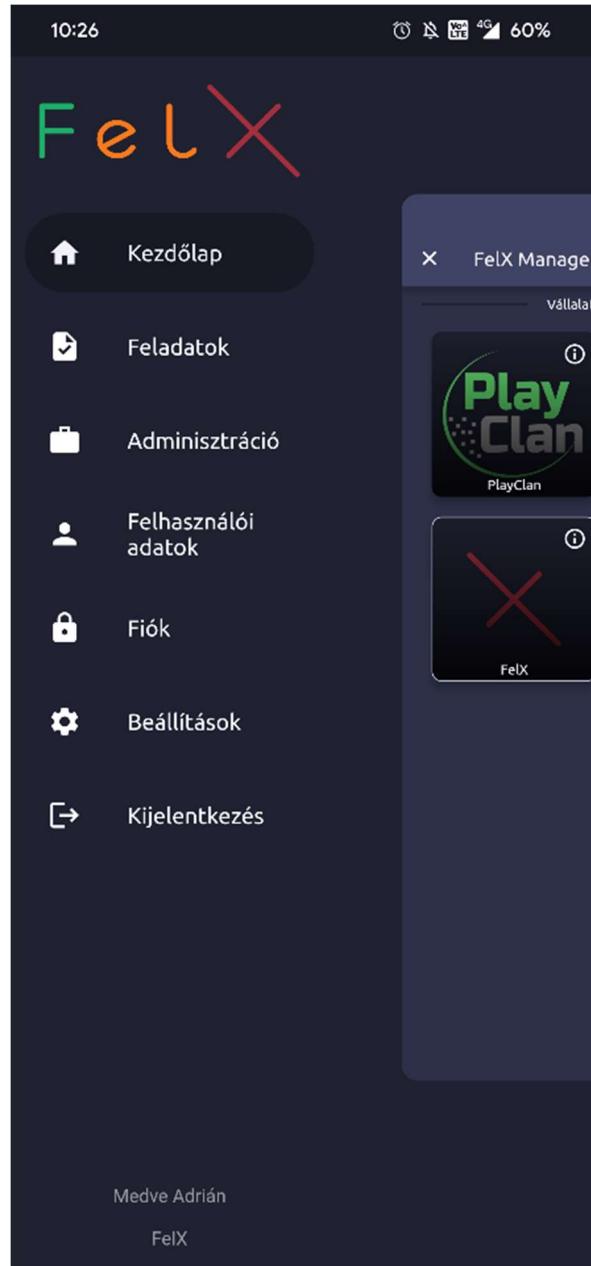
2.2.2 Főoldal

A belépés után megjelenik a főoldal, ahol a felhasználó kiválaszthatja azt a vállalkozást, amelynek az ügyviteli rendszeréhez szeretne hozzá férfi. Ezt a kockákra való koppintással tudja megtenni. A kiválasztott vállalat fehér kerettel jelenik meg. A kockákon megjelenő bekarikázott „i” betűre koppintva az adott vállalat információi jelennek meg egy ablakban.



18) ábra: Főoldal

A bal felső sarokban egy hamburger ikon látható, melyre ha rákoppintunk, megjelennek az alkalmazás további menüpontjai. Itt lehetőség van választani a kezdőlap, feladatok, adminisztráció, felhasználói adatok, fiók, beállítások és kijelentkezés közül. A navigációs oldal alján a felhasználó nevét, és a jelenleg kiválasztott vállalat nevét tekinthetjük meg. A menü a baloldalról csúszik be, de ugyanakkor jobb oldalon még látható az előző oldal egy része, ha oda koppint, akkor bezárhatja a navigációs sávot.



19) ábra: Navigációs menü

2.2.3 Feladatok menüpont

A feladatok menüpont csak akkor érhető el, ha egy vállalat kiválasztásra került. A menüpont kiválasztásával megjelennek a már felvitt feladatok felsorolva. minden feladat előtt megtalálható egy ikon, amely sárga pontokkal az elvégzendő, zöld pipával az elvégzett feladatokat mutatja. Ettől az ikontól jobbra olvasható a cím, a munkacsoporthoz, amihez a feladat tartozik, aki létrehozta a feladatot és a határidő.



20) ábra: Feladatok oldal

Az oldal jobb alsó sarkában a pluszjelre koppintva hozhatunk létre feladatot. Ebben az ablakban kiválaszthatjuk, hogy melyik munkacsoporthoz szeretnénk hozzáadni a feladatot. A munkacsoporthoz található jelölőnégyzet bejelölése után elérhetővé válik a határidő mező. Ezalatt a feladat címét lehet megadni, ami maximum 100 karakteres lehet. A létrehozás gombra kattintva létrehozhatjuk a feladatot.



21) *Feladat hozzáadása*

A feladat kiválasztása után megjelennek a választott feladathoz tartozó üzenetek. A létrehozó alatti téglalapra koppintva megjelennek a feladat információi, még egyszer koppintva ezt eltüntethetjük.

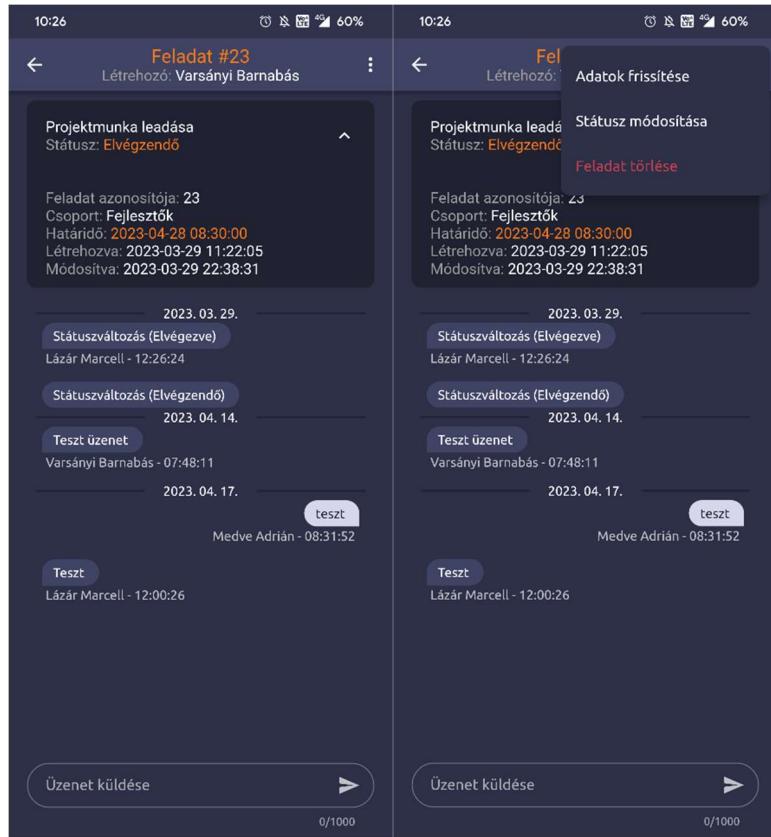
A jobb felső sarokban lévő 3 pontra koppintva az előző menüből választhatunk az adatok frissítése, státusz módosítása, feladat törlése mellett.

Az „Adatok frissítés” pontra koppintva a feladat információi és üzenetei frissítésre kerülnek.

A „Státusz módosítása” lehetőségénél megjelenik egy kérdés, hogy biztosan szeretné-e a felhasználó módosítani a feladat státuszát, és kíván-e üzenetet fűzni hozzá. A mégse gombra koppintva nem történik semmi, az ablak bezáródik. A nem lehetőséget választva a feladat státusza megváltozik és egy előre beállított üzenetet ad hozzá. Az igen gombra koppintva lehetőségünk van a státuszmódosításhoz hozzáadni a saját üzenetünket.

A „Feladat törlése” gombra koppintva megkérdezi az alkalmazás, hogy valóban szeretnénk-e törölni a feladatot.

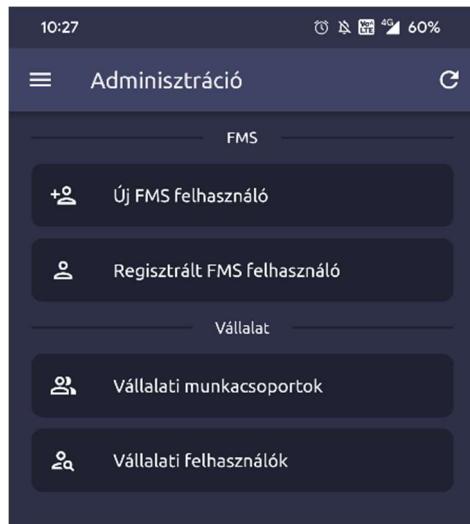
Az oldal alján elhelyezkedő „Üzenet küldése” feliratra koppintva megjelenik az operációs rendszer virtuális billentyűzete, aminek a segítségével beírhatjuk a hozzáadni kívánt üzenetet a feladathoz. A beírt üzenettől jobbra található küldés gombra kattintva hozzáadhatjuk a begépelt szöveget a feladathoz.



22) ábra: Kiválasztott feladat információi, üzenetei

2.2.4 Adminisztráció menüpont

Az adminisztráció menüpont az előző menüponthoz hasonlóan érhető el, emellett csak akkor választható ki, ha a felhasználónak van jog a kiválasztott vállalatot adminisztrálni. Ezen az oldalon felsorolva láthatjuk azokat az almenüpontokat, ami a vállalat kezeléséhez fontos.



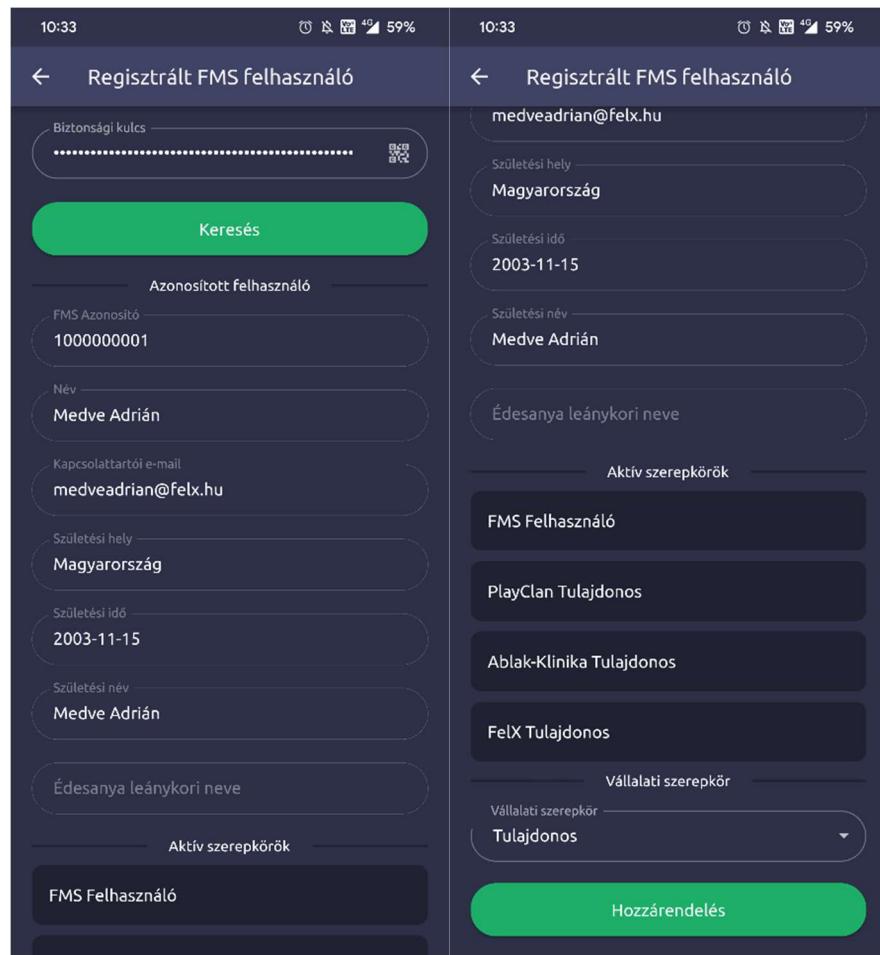
23) ábra: Adminisztráció oldal

Az „Új FMS felhasználó” almenüre koppintva hozhatunk létre új felhasználót az FMS keretrendszerébe, aki eddig még nem volt regisztrálva. A név, kapcsolattartói e-mail, telefonszám, születési hely, születési név és édesanya leánykori neve mezők kitöltése kötelező. A „Vállalati szerepkör” lenyitható menüből kiválaszthatjuk, hogy milyen szerepkörhöz legyen elsősorban hozzárendelve a vállalaton belül. Ha minden kötelező adat kitöltése kész, a sikeres fiók létrehozás után a vágólapra kerül az imént létrehozott fiók jelszava.

24) ábra: Új FMS felhasználó létrehozása

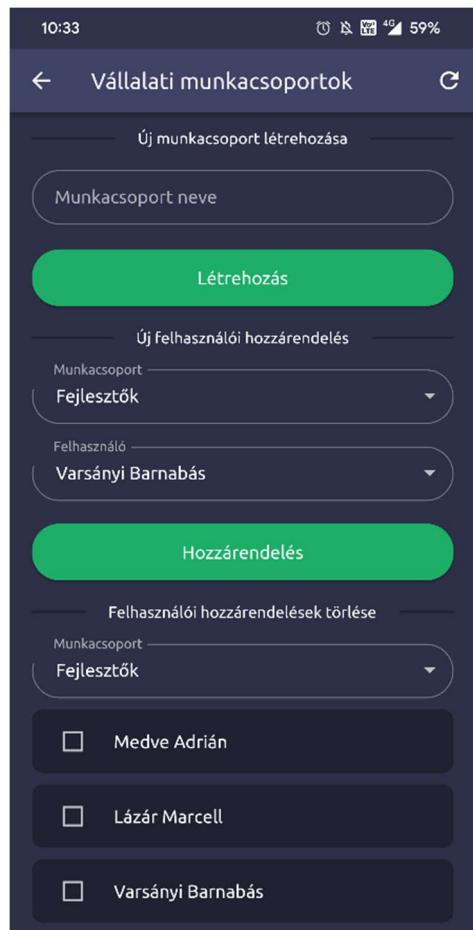
A „Regisztrált FMS felhasználó” almenübe lépve egy már regisztrált felhasználót lehet a kiválasztott vállalathoz társítani egy biztonsági kulcs segítségével. Az oldal tetején látható mezőre koppintva manuálisan beírható vagy bemásolható a vállalathoz hozzáadni kívánt felhasználót. A „Biztonsági kulcs” felirattól jobbra található QR kód ikonra koppintva elindul az eszköz kamerája. Első használatkor az alkalmazás engedélyt kér a kamera használatához, és amennyiben engedélyt adunk, akkor lehetőségünk van az asztali alkalmazás, weboldal vagy a mobil alkalmazás által generált QR kódot beolvasni. A sikeres leolvasás után az alkalmazás visszatér az előző oldalra, ahol manuálisan is van lehetőség a kód beírására, majd egy keresést végez el. Amennyiben a biztonsági kulcs

érvényes, a képernyőn megjelennek a felhasználó egyes adatai és lehetőségünk van hozzárendelni őt egy szerepkörhöz a vállalaton belül.



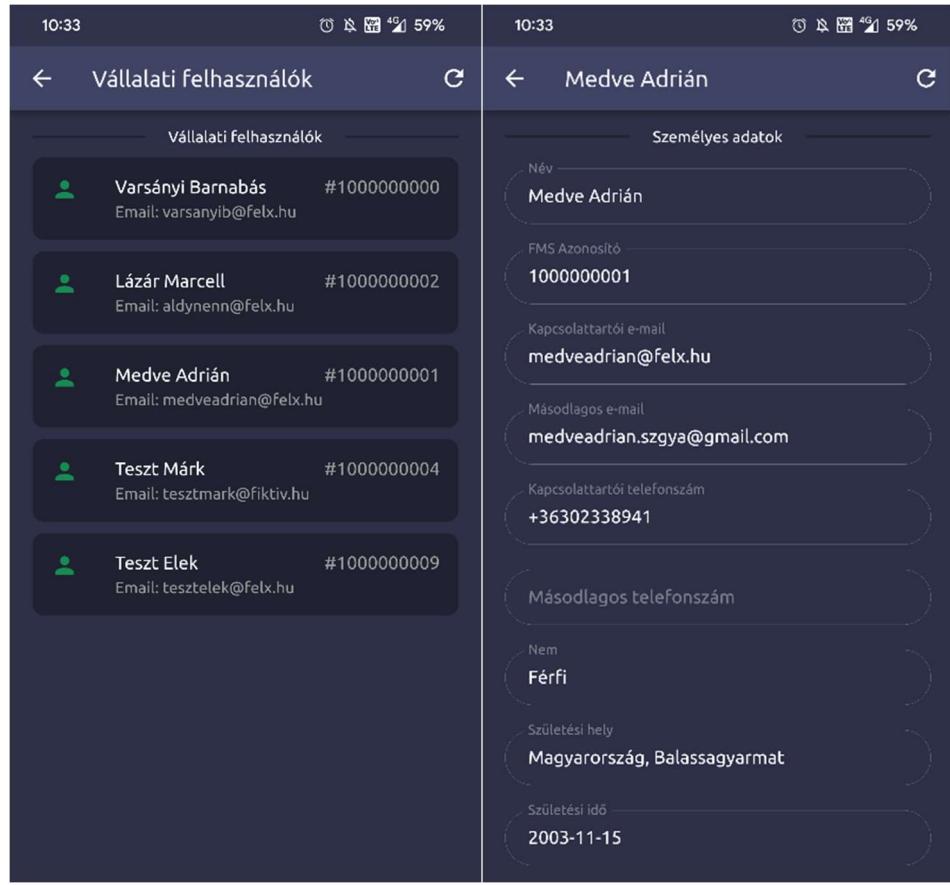
25) ábra: Regisztrált FMS felhasználó oldal

A „Vállalati munkacsoportok” oldalra lépve a felhasználónak lehetősége van új munkacsoport létrehozására, felhasználó munkacsoporthoz rendelésére és törlésére. A munkacsoport nevét az első szövegdobozba írhatjuk be. A „Létrehozás” gombra koppintva hozhatjuk létre a munkacsoportot. Vállalati felhasználót és munkacsoportot a legördülő menüből tudunk kiválasztani. A „Hozzárendelés” gombbal tudjuk hozzárendelni a felhasználót az adott munkacsoporthoz. A kiválasztott munkacsoportból lehetőség van a felhasználók törlésére, ehhez a felsorolásban ki kell jelölnünk a felhasználókat, akiket el szeretnénk távolítani, majd a „Törlés” gombot kell megnyomnunk.



26) ábra: *Vállalati munkacsoporthoz*

A „Vállalati felhasználók” oldalon az összes felhasználó felsorolását láthatjuk. A kiválasztott felhasználóra koppintva megjelennek az adatai, ez alatt a „Szerepkörök hozzárendelése” szöveg alatt választhatunk, mely vállalati szerepkörhöz szeretnénk a személyt hozzáadni. A „Meglévő szerepkörök” listában koppintásra megjelenik egy ablak, ahol a felhasználó hozzáférését tudjuk szabályozni.



27) ábra: Vállalati felhasználók oldal

2.2.5 Felhasználói adatok menüpont

Ebben a menüpontban megjelennek a bejelentkezett felhasználó aktuális adatai, itt ilyenkor még nincs lehetőség módosításra. Ezt azért építettem bele, ne fordulhasson elő olyan eset, hogy a felhasználó véletlen érintés következtében akaratlanul módosítsa az adatait.

Az adatok módosításhoz kell koppintania az „Adatok módosítása” gombra. Ekkor már lehetősége van a felhasználónak adatot módosítani, alul pedig két gomb jelenik meg, az „Elvetés” és a „Mentés”.

Az „Elvetés” gombra koppintva visszatér módosítás nélkül az eredeti adatokhoz. A „Mentés” gombra koppintva pedig megtörténik az adatok módosítása az adatbázisba, és ez esetben is az adatokhoz tér vissza a felhasználó, de már a módosított adatok jelennek meg a kijelzőjén.

The figure consists of two side-by-side screenshots of a mobile application interface. Both screens are titled 'Felhasználói adatok' (User data).
 Left Screen (10:33):
 - Név: Medve Adrián
 - FMS Azonosító: 1000000001
 - Kapsolattartói e-mail: medveadrian@felx.hu
 - Másodlagos e-mail: medveadrian.szgya@gmail.com
 - Kapsolattartói telefonszám: +36302338941
 - Másodlagos telefonszám: (empty)
 - Gender: Férfi (radio button selected)
 - Születési ország: Magyarország [HU]
 - Születési hely: Balassagyarmat
 - Születési idő: 2003-11-15
 - Születési név: Medve Adrián
 - Bottom buttons: Adatok módosítása (orange)

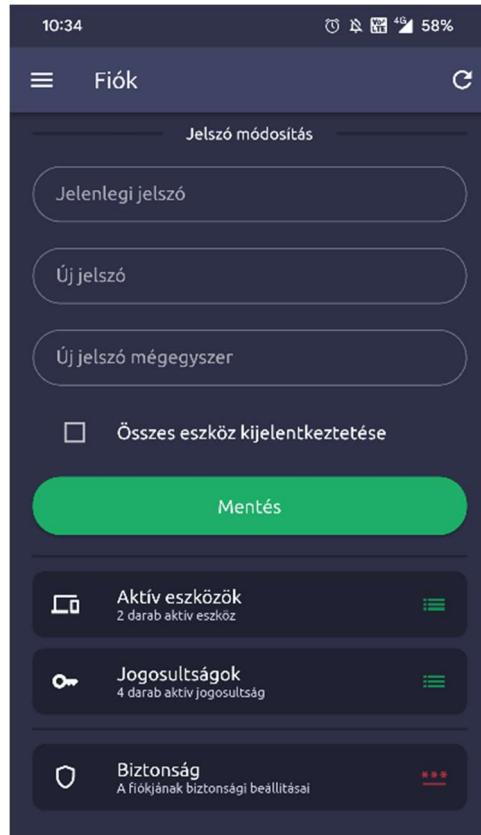
Right Screen (10:34):
 - Név: Medve Adrián
 - FMS Azonosító: 1000000001
 - Kapsolattartói e-mail: medveadrian@felx.hu
 - Másodlagos e-mail: medveadrian.szgya@gmail.com
 - Kapsolattartói telefonszám: +36302338941
 - Másodlagos telefonszám: (empty)
 - Gender: Férfi (radio button selected)
 - Születési ország: Magyarország [HU]
 - Születési hely: Balassagyarmat
 - Születési idő: 2003-11-15
 - Születési név: Medve Adrián
 - Bottom buttons: Elvetés (red) and Mentés (green)

28) ábra: Felhasználói adatok módosítása

Az adatok között az ország kiválasztása a lenyíló mezőből lehetséges, ugyanígy a dátum esetében is, ha új időpontot szeretne választani a felhasználó, akkor egy naptár jelenik meg, ahol könnyedén kijelölheti az új időpontot.

2.2.6 Fiók menüpont

Ebben a menüpontban lehetősége van a felhasználónak a jelszava módosítására. Ehhez be kell írnia a jelenlegi jelszavát, majd ezt követően megadni az újat kétszer. A jelszóval kapcsolatos követelmények itt is ugyanazok, mint a webes felületen, vagyis legalább 8 karakternek kell lennie, tartalmaznia kell számot és speciális karaktert is. A jelszó mezők alatt egy jelölőnégyzet látható, amelyet ha a felhasználó kipipál, akkor az összes eszkösről kijelentkezteti a jelszóváltoztatás után.



29) ábra: Fiók oldal

A „Mentés” gomb alatt látható az „Aktív eszközök”, amire koppintva az aktív eszközök jelennek meg, amelyen bejelentkeztek a fiókba. Egyaránt láthatóak a mobil, a web és az asztali eszközök is. Ha a kiválasztott eszközt balra húzza, akkor választhat az információk megjelenítése vagy az eszköz kijelentkeztetése között.

Az „Infók” gombra koppintva a következő információk jelennek meg minden platform esetén

- IP cím,
- szolgáltató,
- a helyzet meghatározáson belül megjelenik az ország kódja, megye és település, böngésző esetén
- böngésző típusa,
- az operációs rendszer,

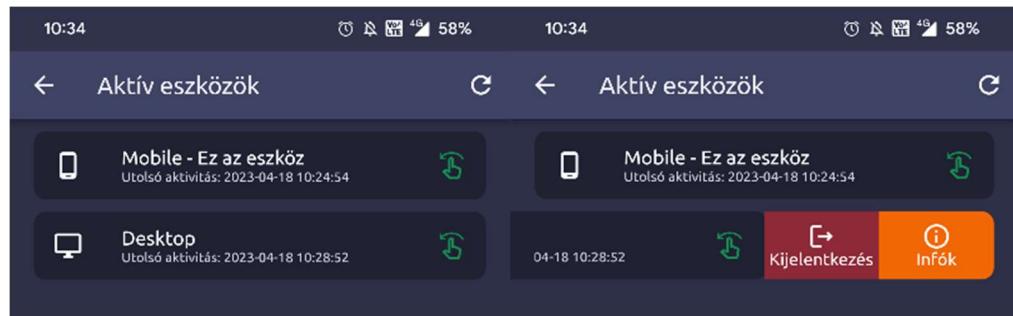
mobil app esetében

- milyen típusú a telefon,
- operációs rendszer verzió száma,
- alkalmazás verziója,

asztali alkalmazásnál

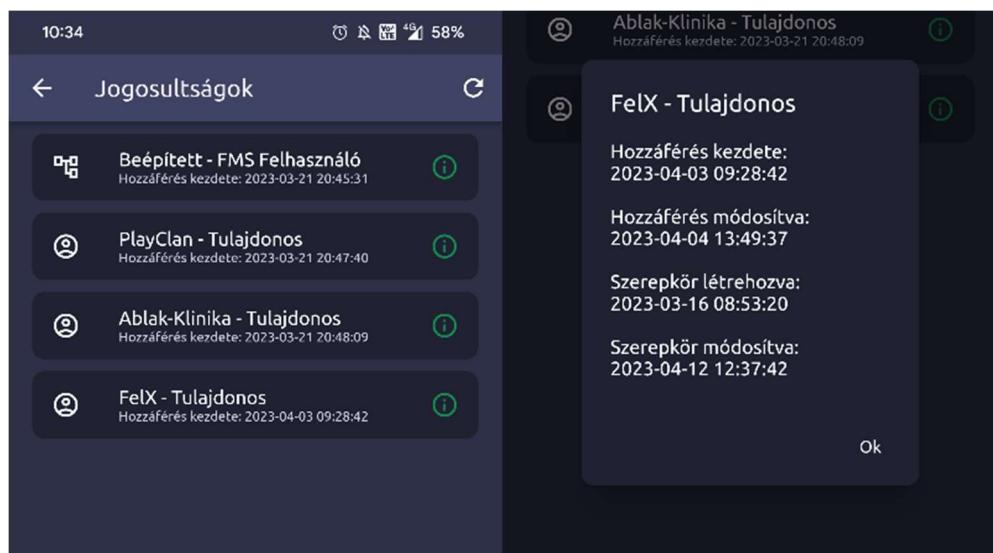
- operációs rendszer,
- számítógép neve,
- alkalmazás verziója.

Ha nem a saját eszköz adatait tekintjük meg, az információk gomb mellett megjelenik még egy gomb, mellyel ki lehet jelentkezni az adott eszkösről.



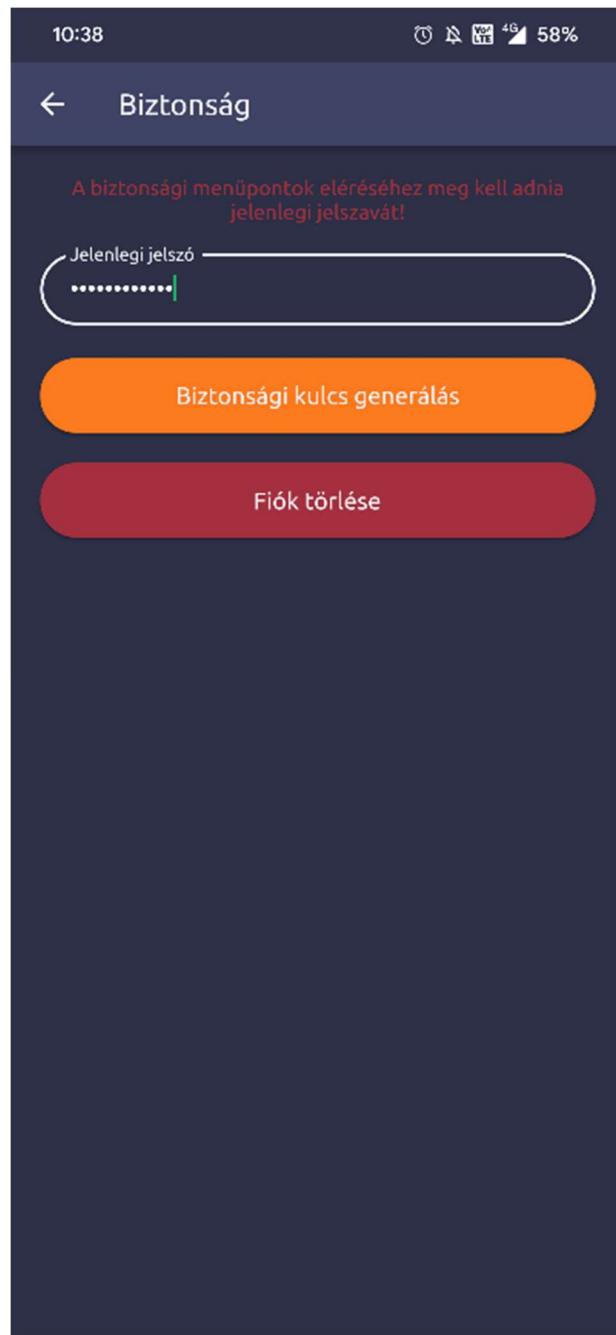
30) ábra: Aktív eszközök oldal

A „Jogosultságok” menübe lépve felsorolva láthatjuk az éppen aktív vállalati hozzáféréseinket. A kiválasztott jogosultságra koppintva egy megjelenő ablakban láthatjuk a hozzáférésünk kezdetét, módosítását, a szerepkör létrehozási és módosítási idejét.



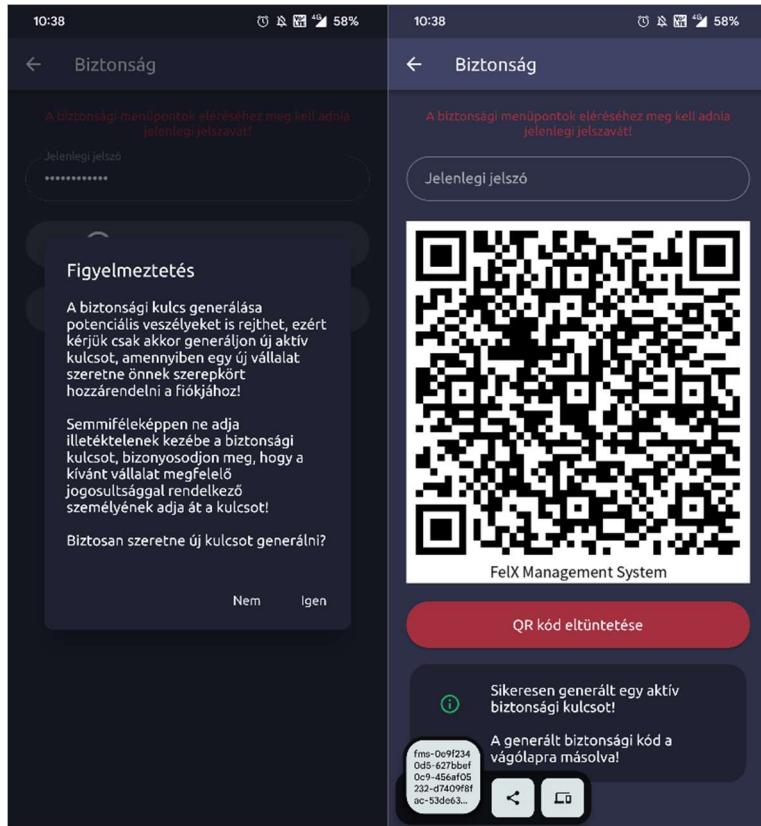
31) ábra: Jogosultságok oldal

A „Biztonság” oldalon biztonsági kulcsot tudunk generáltatni és a fiókunk törlésére is van lehetőség. Ezekhez a menüpontokhoz a jelenlegi jelszavunkat kell megadni. Amit a jelszó mezőbe beírtuk a jelszót, elérhetővé válik az előbb említett 2 menüpont.



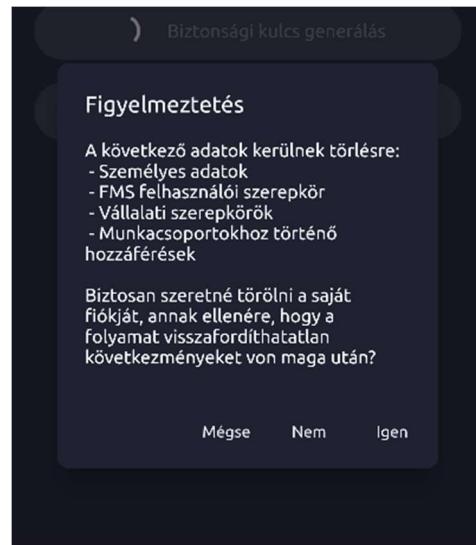
32) ábra: *Biztonság oldal*

A „Biztonsági kulcs generálás” gombra koppintva az alkalmazás még egyszer megkérdezi, hogy biztosan szeretnénk-e generáltatni biztonsági kulcsot. Az „Igen” gombra koppintva az imént generált biztonsági kulcs az eszköz vágólapjára kerül, és megjelenik egy QR kód, amely a biztonsági kulcsot tartalmazza.



33) ábra: *Biztonsági kulcs generálása*

A „Fiók törlése” menüpontot választva az alkalmazás többször is megkérdezi és figyelmezteti a felhasználót, hogy ez visszafordíthatatlan következményeket vonhat maga után. Az összes figyelmeztést elfogadva az FMS felhasználói fiók törlésre kerül.



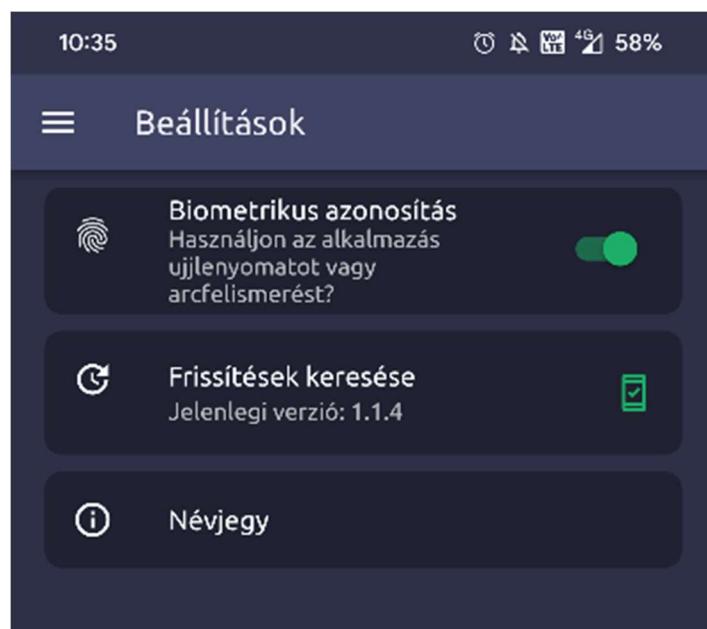
34) ábra: *Fiók törlése figyelmeztetés*

2.2.7 Beállítások menüpont

A beállítások menüpontban a felhasználói fióktól eltérő módosításokat lehet végezni.

A „Biometrikus azonosítás” állapotát tudjuk kapcsolni koppintással. Ha kikapcsoljuk ezt a menüpontot, a következő app-megnyitáskor az alkalmazás nem fogja kérni az eszközön beállított ujjlenyomatot vagy arcfelismerést.

A „Frissítések keresése” menüre koppintva manuálisan tudunk alkalmazás-frissítéseket keresni. Ha elérhető frissítés az alkalmazáshoz, egy felugró ablakban megnézhetjük a változtatásokat. Az „Emlékeztessen később” gombot választva az alkalmazás következő indulásáig nem fog felbukkanni a frissítési ablak. Az „Frissítés” gombra koppintva az alkalmazás-frissítés letöltődik a háttérben, és települ.



35) ábra: Beállítások oldal

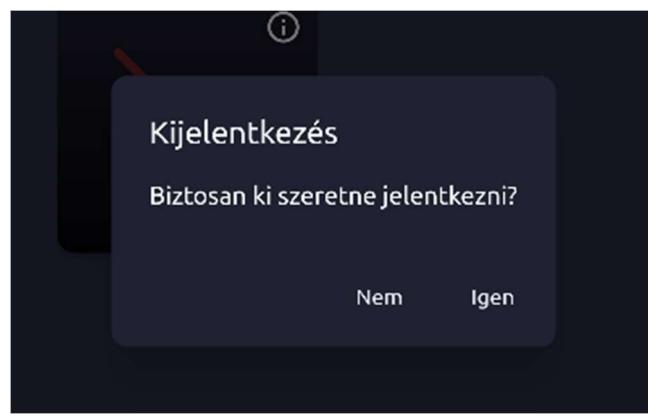
A „Névjegy” oldalra navigálva megtekinthetők azok a változtatások, amik eddig az alkalmazásba bekerültek vagy módosultak. A „Harmadik féltől származó kiegészítők” menüpontra koppintva, felsorolva látható az összes harmadik fél által használt kiegészítő, amelyek segítségével az alkalmazás megíródott. A „Weboldal” és a „Státusz weboldal” opciókra koppintva megnyílik az operációs rendszer alapértelmezett böngészője és betöltenek a választott elemnek megfelelő weboldalak.



36) ábra: Névjegy oldal

2.2.8 Kijelentkezés menüpont

A kijelentkezés menüre koppintva az alkalmazás megkérdezi a felhasználót, hogy biztosan ki szeretne-e jelentkezni az eszkösről. Az igen gombot választva megtörténik a kijelentkeztetés az adott eszkösről, majd a vezérlés visszatér a bejelentkezési oldalra.



37) ábra: Kijelentkezés felhívás

2.3 Asztali alkalmazás

Az alkalmazás indítása után közvetlenül a rendszer ellenőrzi a szoftver verzióját, így új frissítés esetén két kattintással elérhetőek a legfrissebb verziók. A szoftvert úgy alakítottam ki, hogy ha valaki nem szívesen használja az egeret, akkor a billentyűzettel is könnyedén tud navigálni a menüpontok között. Például F2-vel kiválaszthatjuk a vállalatot, F3-al a saját adatainkat tudjuk megnézni, F5-el a jogosultságok tekinthetőek meg, hogy melyik vállalati szerepkörhöz van hozzáférésünk. A felhasználó minden gombon látja a billentyűkombinációt, amely támogatja ezt a funkciót.

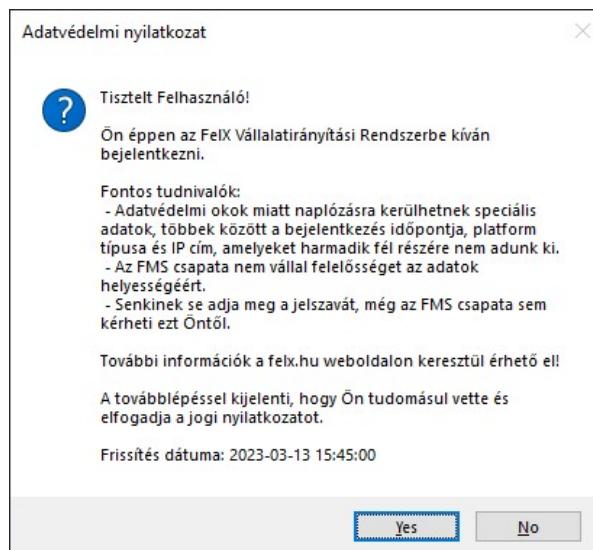
Amit még rendkívül fontos kiemelnem, az a keretrendszerben létrehozott szerepkörök jelentősége. A vállalatokhoz tartoznak szerepkörök, ezek a szerepkörök határozzák meg a jogosultságokat, a felhasználók ezekhez a szerepkörökhöz kerülnek hozzárendelésre. Amennyiben a felhasználó hozzáfér aktívan a szerepkörhöz, és a szerepkör is aktív, akkor a felhasználó hozzá fog féni az adott vállalat specifikus moduljaihoz.

2.3.1 Bejelentkezési felület

A bejelentkezési felület ugyanúgy jelenik meg, mint a webes alkalmazás esetében, az azonosítás a felhasználónév és jelszó kombinációjával, majd az adatvédelmi és jogi nyilatkozat elfogadásával történik. Miután a felhasználó sikeresen bejelentkezett, létrejön a munkamenet, megjelenik a kezelőpult. A munkameneteknek minden esetben meghatározott időtartama van, amely adatvédelmi szempontokat figyelembe véve a használatbavételi platformuktól függ. Asztali alkalmazás esetében a munkamenet 24 órára, weboldal esetében 1 órára, mobil alkalmazás esetében 168 órára, azaz 7 napra jön létre.



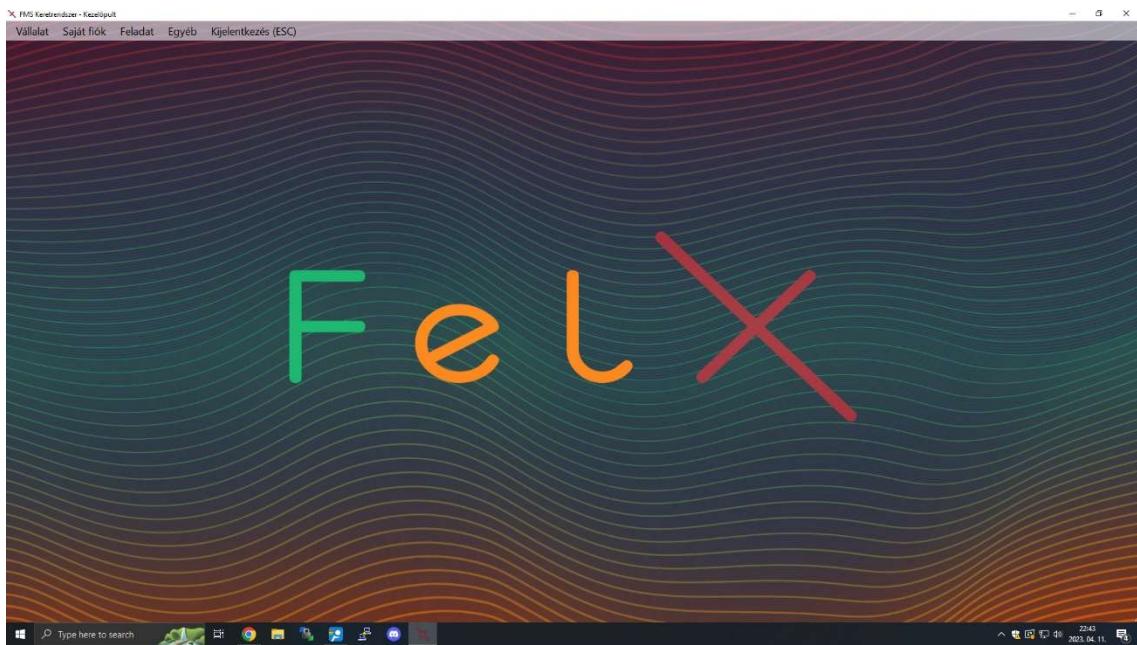
38) ábra: Asztali alkalmazás bejelentkezési felülete



39) ábra: Bejelentkezéskor megjelenő rövidített adatvédelmi nyilatkozat felugró ablakban

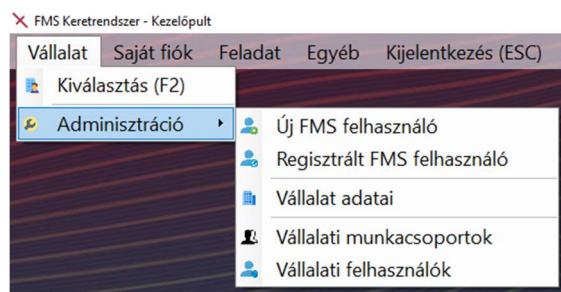
2.3.2 Kezelőpult

A kezelőpult a kiinduló pontja a keretrendszernek, ahol már elérhető a menü szalag, melyeken belül különböző almenüpontok kerültek felsorolásra. A menüpontok között vannak olyanok, amelyek vállalat-specifikusak, vagyis attól függően működnek, hogy melyik vállalatot választja ki a felhasználó.



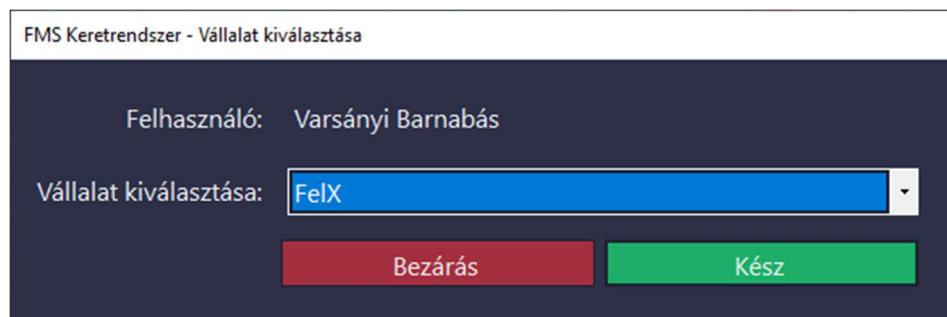
40) ábra: Asztali alkalmazás kezelőpultja

Az első menüpont az a vállalat, ezen belül található a kiválasztás menü, amellyel kiválasztható az adott vállalat.



41) ábra: Vállalat menüpont

A vállalat-specifikus funkciók esetében szükség van a megfelelő vállalat kiválasztására, amelyre figyelmeztető szöveg is felhívja a figyelmet a felhasználó részére, ha elfelejtené elvégezni a szükséges lépéseket.



42) ábra: Vállalat kiválasztása munkaablak



43) ábra: Vállalat-specifikus funkció figyelmeztető üzenete

2.3.3 Adminisztráció menüpont

1) Az „Új FMS felhasználó” pontra kattintva tudunk új felhasználót hozzáadni, ami azért más a mi esetünkben, mint a többi vállalatirányítási rendszer esetében, mert nem egy adott vállalathoz adjuk hozzá, hanem a központi rendszerben regisztráljuk a felhasználó adatait. Ez a funkció azért jó, mert így bármelyik vállalat eléréséhez kaphat jogosultságot, ezzel elkerüljük azt a fajta adatredundanciát, hogy több vállalathoz történő regisztrációkor, újra meg kelljen adni az összes személyes adatot, vagy egy valós személynak több felhasználói fiókja legyen. Ha megbizonyosodtunk arról, hogy a felhasználónak, akit a kiválasztott vállalathoz szeretnénk felvenni, még nem rendelkezik FMS felhasználói fiókkal, akkor ebben a menüpontban tehetjük meg a regisztrációt. Itt van lehetőségünk rögzíteni a felhasználó teljes nevét, e-mail címeit, telefonszámait, nemét, születési helyét és idejét, születési (leánykori) nevét, illetve az édesanya leánykori nevét. Ezen felül ki kell választani, hogy a vállalat melyik szerepköréhez szeretnénk hozzárendelni a felhasználót. Itt csak az első szerepkör felvitelére van lehetőség, a további szerepkörök hozzáadását, engedélyezését és tiltását egy másik menüpontban tehetjük meg, erről figyelmeztet a piros betűszínnel szedett figyelmeztető szöveg, a munkaablak alján. Amennyiben megbizonyosodott róla felhasználó, hogy minden adat helyesen lett kitöltve, akkor az F10 billentyű lenyomásával vagy a létrehozás gombra kattintással indítható a regisztráció. Sikeres regisztráció esetén egy felugró üzenetben kapunk tájékoztatást, hogy milyen FMS azonosítóval történt felvételre a rendszerben, illetve a kiválasztott, hozzáadni kívánt szerepkör hozzárendelési sikerességéről jelenik meg tájékoztatás. Amennyiben valamelyik kötelezően kitöltendő adat hiányzik, a regisztráció félbeszakad, és egy hiánypótlásra felszólító hibaüzenet jelenik meg. Fontos, hogy a

sikeres regisztrációt követően a vágólapra kerül a felhasználó ideiglenes jelszava, amely máshol nem kerül többet megjelenítésre, adatvédelmi okokból.

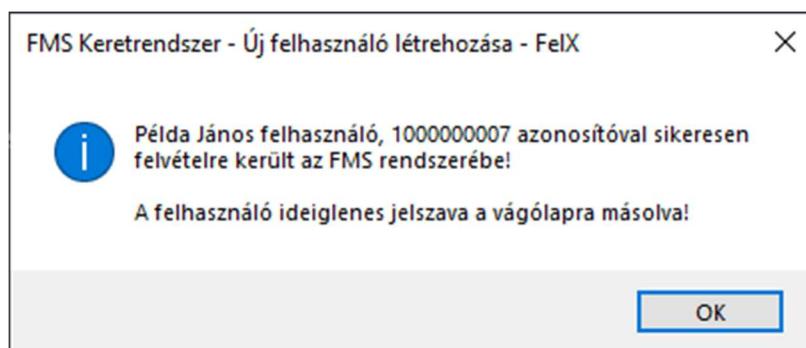
FMS Keretrendszer - Új felhasználó létrehozása - FelX

Név:	Példa János	
Kapcsolattartói e-mail:	peldajanos@fiktiv.hu	
Másodlagos e-mail:	jamesexample@test.net	
Kapcsolattartói telefonszám:	+36301234567	
Másodlagos telefonszám:	+361654321	
Nem:	<input type="radio"/> Férfi <input checked="" type="radio"/> Nő A nem későbbi módosítására nincs lehetőség!	
Születési hely:	Magyarország [HU] <input type="button" value="▼"/>	Balassagyarmat
Születési idő:	1985-07-12 <input type="button" value="▼"/>	
Születési név:	Példa János	
Édesanya leánykori neve:	Minta Emese	
Vállalati szerepkör:	Tesztelek <input type="button" value="▼"/>	

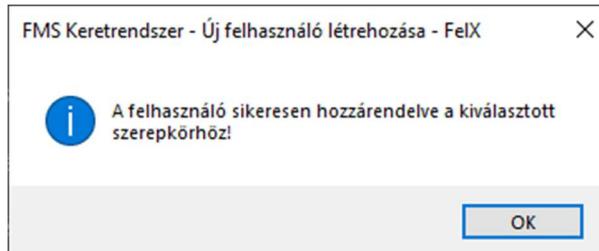
A felhasználó további jogosultságainak kezelését a Vállalat -> Adminisztráció -> Vállalati felhasználók menüpont alatt módosítható a sikeres létrehozást követően!

Vissza (ESC) Létrehozás (F10)

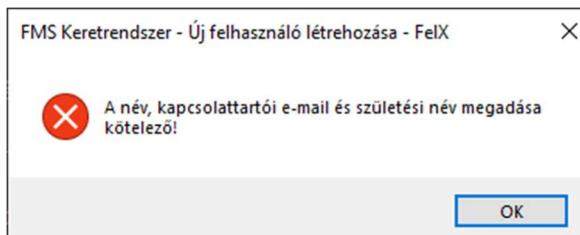
44) ábra: Új FMS felhasználó létrehozása munkaablak



45) ábra: Új FMS felhasználó sikeres létrehozásról szóló tájékoztatás

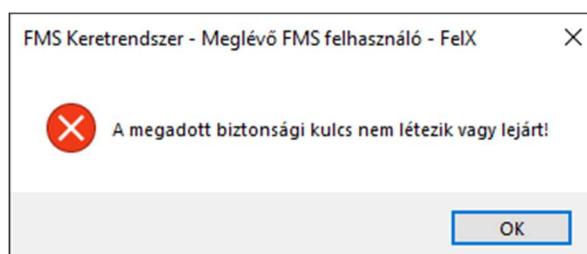


46) ábra: Új FMS felhasználóhoz sikeres szerepkör hozzárendelés üzenete



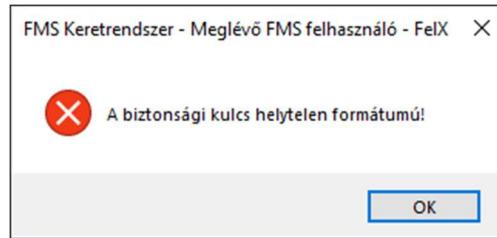
47) ábra: Új FMS felhasználó kötelezően megadandó adatok üzenete

- 2) A következő almenü a (meglévő) „Regisztrált FMS felhasználó”, amelyet abból a célból hoztam létre, ha már létezik a felhasználó az FMS rendszerben, de más vállalat is szeretne neki hozzáférést, szerepköröket, és jogokat biztosítani, akkor a meglévő felhasználói fiókjához történő hozzáférés zavartalan legyen. A menüpont megnyitásakor megjelenik a vállalat neve, amelyhez a meglévő felhasználó hozzárendelésre fog kerülni, és kéri a felhasználó által generált biztonsági kulcsot. A biztonsági kulcs az asztali alkalmazásban a saját fiók menüpont alatt a biztonság munkaablakban generálható. A kulcsot annak a felhasználónak kell generálnia, aki szeretné, hogy hozzárendelést nyerjen egy új vállalathoz. Ebben az esetben a generált kulcsot annak a személynek kell rendelkezésére bocsátani, aki a vállalatban „adminisztrációs” feladatakat is elláthat, vagyis van jogosultsága adminisztrálni a vállalat felhasználóit. A kulcs csak egyszer használható, a bevitelt követően a Keresés gombra kattintva, vagy az F9 billentyűt lenyomva indítható el a kulcs ellenőrzésének folyamata. Ha a kulcs felhasználásra került vagy nem létezik, akkor egy felugró hibaüzenet jelenik meg.



48) ábra: Meglévő FMS felhasználó helytelen kulcs hibaüzenete

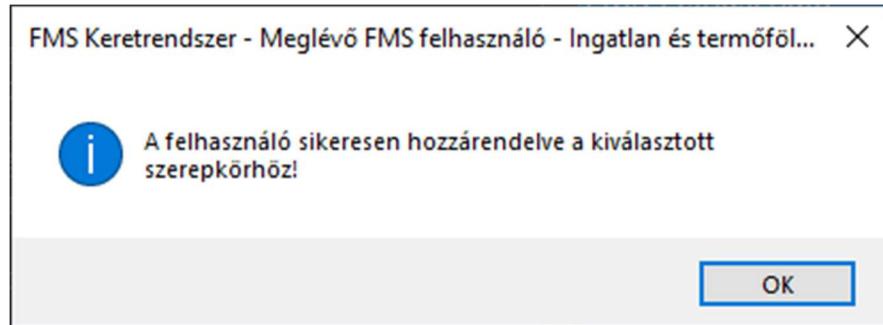
Ha a kulcs nem megfelelő formátumú, akkor figyelmeztet a rendszer, hogy nem megfelelő adatot vittünk be a beviteli mezőbe, vagy a felhasználó rossz adatot adott át a „vállalat adminisztrátorának”.



49) ábra: Meglévő FMS felhasználó helytelen formájú kulcs hibaüzenete

Amennyiben az azonosítás sikeres, érvénytelenítésre kerül a felhasznált kulcs és megjelennek a felhasználó adatai, aktív szerepkörei. Adatvédelmi szempontokat követve csak annak a felhasználónak az adatai láthatóak megjelenítve, aki a biztonsági kulcsot generálta, így akinek átadja a kulcsot, hozzájárul a személyes adatainak kezeléséhez. Itt van lehetőség kiválasztani azt a vállalati szerepkört, amelyhez szeretnénk a felhasználót hozzárendelni. Amennyiben az adat megfelelő és kiválasztásra került a megfelelő szerepkör, a hozzárendelés vagy az F10 billentyű lenyomásakor felugró üzenetben olvashatjuk a hozzárendelés sikerességét vagy a sikertelenség okát.

50) ábra: Meglévő FMS felhasználó munkaablak



51) ábra: Meglévő felhasználóhoz sikeres szerepköri hozzárendelés

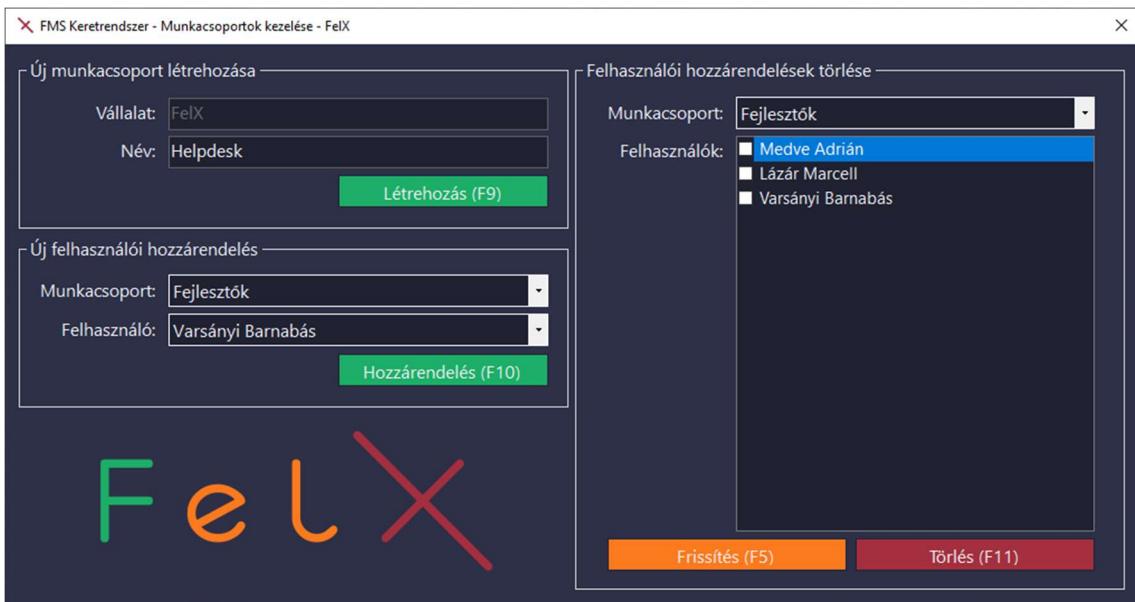
- 3) A „Vállalat adatai” menüpont alatt a kiválasztott vállalat adatai tekinthetőek meg, mint a vállalat logója, neve, adószáma, NyT száma, azaz nyilvántartási száma, létrehozás dátuma és az esetleges módosítás dátuma.

The screenshot shows a form with the following data:

Név:	FelX
Adószám:	0000000-0-00
NYT szám:	00000000
Létrehozva:	2023-03-16 08:49:59
Módosítva:	2023-03-20 13:14:30

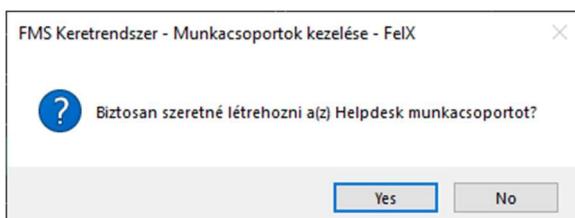
52) ábra: Vállalat adatai munkaablak

- 4) A „Vállalati munkacsoporthoz” menüpont összetett, mivel itt lehetőség van új munkacsoporthoz létrehozására, dolgozók hozzárendelésére a munkacsoporthoz, valamint törlni is lehet a munkacsoporthoz a felhasználókat. A munkacsoporthoz törlésére itt nincs lehetőség, mivel a munkacsoporthoz vannak a feladatok hozzárendelve, így a törlés lehetősége nem okozhat a későbbiekben adatvesztéssel kapcsolatos hibákat.



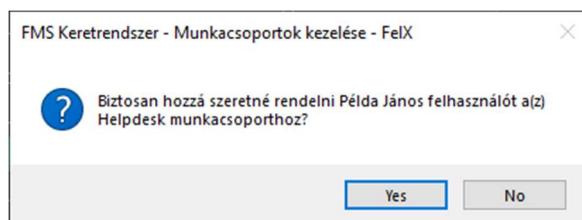
53) ábra: Vállalati munkacsoportok kezelése munkaablak

Új munkacsoport létrehozásához először meg kell adni a csoportnak egy fantázianevet, majd a megjelenő létrehozás gombra kell kattintani, vagy az F9 billentyűkombinációval jelenik meg egy felugró üzenet, hogy valóban szeretné-e a felhasználó létrehozni a beírt névvel ellátott csoportot.



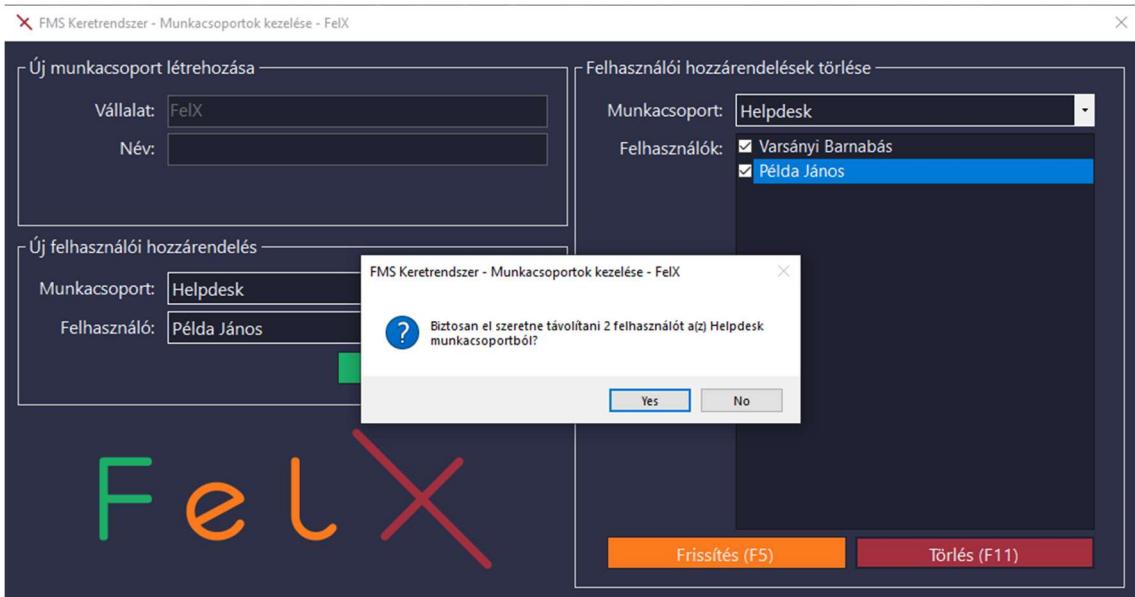
54) ábra: Vállalati munkacsoport létrehozása

Az „Új felhasználói hozzárendelés” résznél a munkacsoportot kiválasztva, hozzárendelhetünk a vállalathoz tartozó felhasználók közül új személyeket, amelyet a hozzárendelés gombra kattintva vagy az F10 billentyűt lenyomva egy felugró ablakban dönthetjük el, hogy valóban a kiválasztott felhasználót szeretnénk-e hozzárendelni a munkacsoporthoz. Amennyiben helyesen döntöttünk, nyugtázhatjuk az üzenetet, és megtörténik a hozzárendelés a rendszerben.



55) ábra: Vállalati munkacsoporthoz felhasználó hozzárendelése

Amennyiben el szeretnénk távolítani egy vagy akár több embert a kiválasztott munkacsoportból, elegendő pipát tenni azoknak a felhasználóknak a neve mellé, akiket nem szeretnénk a csoportban látni. A törlés gombra kattintva, vagy az F11 billentyűt lenyomva egy felugró üzenetben rákérdez a szoftver, hogy biztosan szeretnénk-e az összes kijelölt felhasználót eltávolítani.



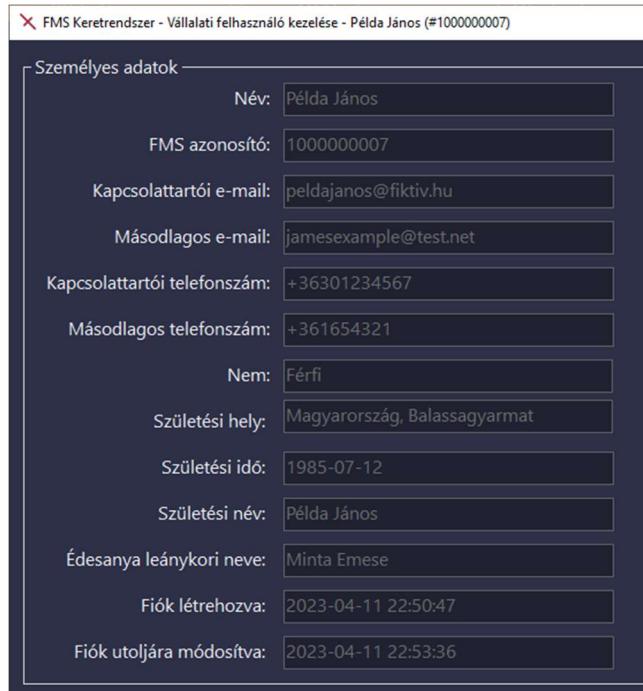
56) ábra: Vállalati munkacsoportból felhasználó eltávolítása

- 5) A „Vállalati felhasználók” menüpont alatt, az adott felhasználó kiválasztása után az információk gombra kattintva, vagy az F10 billentyűkombinációt lenyomva, az összes vállalathoz hozzárendelt felhasználó adatai láthatóak.

Aktív vállalati felhasználók									
FMS azonosító	Név	Email	Másodlagos email	Tel. szám	Másodlagos tel. szám	Születési idő	Születési ország	Születési hely	S.
1000000000	Varsányi Barnabás	varsanyib@felx.hu							
1000000002	Lázár Marcell	aldynenn@felx.hu							
1000000001	Medve Adrián	medveadrian@felx.hu							
1000000004	Teszt Márk	teszmarko@fiktiv.hu	teszmarko@test.net	+36301234567	+36201234567	1990-01-01	Ausztria	Wien	Te
1000000007	Példa János	peldajanos@fiktiv.hu	jamesexample@test.net	+36301234567	+361654321	1985-07-12	Magyarország	Balassagyarmat	Pé

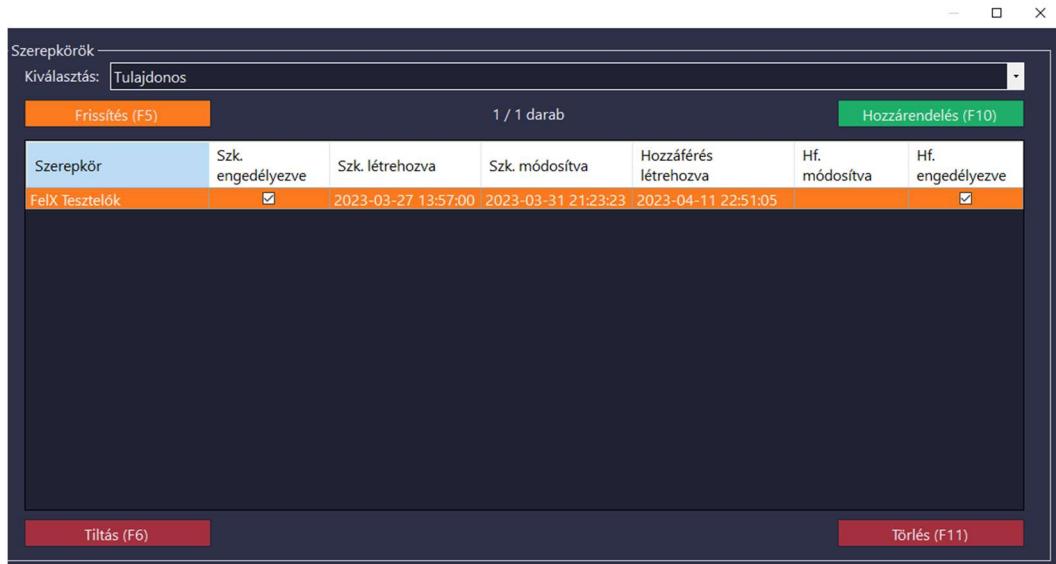
57) ábra: Vállalati felhasználók munkablak

A megjelenő ablak bal felében a felhasználó személyes adatai láthatóak, a munkablak jobb oldalában láthatjuk az összes vállalati szerepkört, amelyet kiválasztva hozzárendelhetjük a felhasználóhoz. A táblázatban leolvashatóak a felhasználóhoz rendelt szerepkörök nevei, állapota (engedélyezett vagy letiltott állapot), létrehozás ideje, utolsó módosítás ideje, illetve a szerepkörhöz való hozzáférés létrehozásának ideje (amikor a szerepkörhöz hozzárendelésre került), módosításának ideje (amikor a státusza módosításra került), hozzáférésének státusza.

 A screenshot of the FMS Kerrendszer software interface showing the management of a corporate user. The title bar indicates it's for managing corporate users (Vállalati felhasználó kezelése) for user Példa János (ID #1000000007). The main area is titled 'Személyes adatok' (Personal data) and contains the following fields:

Név:	Példa János
FMS azonosító:	1000000007
Kapcsolattartói e-mail:	peldajanos@fiktiv.hu
Másodlagos e-mail:	jamesexample@test.net
Kapcsolattartói telefonszám:	+36301234567
Másodlagos telefonszám:	+361654321
Nem:	Férfi
Születési hely:	Magyarország, Balassagyarmat
Születési idő:	1985-07-12
Születési név:	Példa János
Édesanya leánykori neve:	Minta Emese
Fiók létrehozva:	2023-04-11 22:50:47
Fiók utoljára módosítva:	2023-04-11 22:53:36

58) ábra: Vállalati felhasználók kezelése I. rész

 A screenshot of the 'Szerepkörök' (Roles) section of the FMS Kerrendszer. The title bar says 'Szerepkörök - Tulajdonos'. The table shows a single row of data:

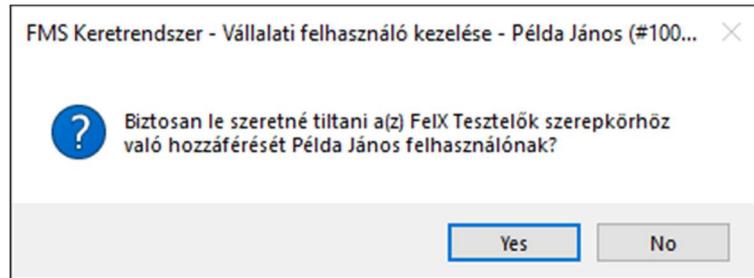
Szerepkör	Szk. engedélyezve	Szk. létrehozva	Szk. módosítva	Hozzáférés létrehozva	Hf. módosítva	Hf. engedélyezve
FelX Tesztelők	<input checked="" type="checkbox"/>	2023-03-27 13:57:00	2023-03-31 21:23:23	2023-04-11 22:51:05		<input checked="" type="checkbox"/>

Buttons at the bottom: 'Frissítés (F5)' (Refresh), 'Tiltás (F6)' (Ban), 'Hozzárendelés (F10)' (Assign), and 'Törlés (F11)' (Delete).

59) ábra: Vállalati felhasználók kezelése II. rész

A szerepkört kijelölve lehetőség van a szerepkörhöz való hozzáférésének törlésére a törlés gombra kattintva, vagy az F11 billentyű lenyomásával megjelenő ablakban történő jóváhagyásával.

Amennyiben, ha nem szeretnénk a szerepkörhöz való hozzáférést törlni, lehetőség van ideiglenes letiltására is. A szerepkör kiválasztása után lévő tiltás gombra kattintáskor, vagy az F6 billentyű lenyomása után, felugró ablakban való ellenőrzésével és elfogadásával kezdeményezhető a művelet.



60) ábra: Felhasználó hozzáférésének letiltását tartalmazó üzenet

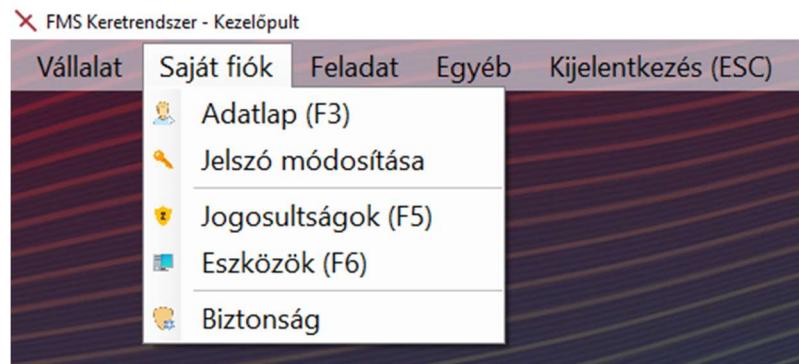
Az ideiglenesen letiltott szerepköri hozzáférést kiválasztva a megjelenő engedélyezés gombra kattintásakor, vagy az F7 billentyűkombinációt lenyomása után felugró ablakban való ellenőrzéssel és elfogadásával állítható vissza.

Szerepkörök						
Kiválasztás: Tulajdonos						
Frissítés (F5)		1 / 1 darab			Hozzárendelés (F10)	
Szerepkör	Szk. engedélyezve	Szk. létrehozva	Szk. módosítva	Hozzáférés létrehozva	Hf. módosítva	Hf. engedélyezve
FelX Tesztelők	<input checked="" type="checkbox"/>	2023-03-27 13:57:00	2023-03-31 21:23:23	2023-04-11 22:51:05	2023-04-11 23:11:33	<input type="checkbox"/>

[Engedélyezés \(F7\)](#) [Törlés \(F11\)](#)

61) ábra: Vállalati felhasználó letiltott szerepköreinek engedélyezése

2.3.4 Saját fiók menüpont



62) ábra: Saját fiók menüpont

- 1) Adatlap (F3): itt láthatóak a bejelentkezett felhasználó személyes adatai, mint például kapcsolattartó e-mail címe, születési helye, születési ideje, fiók létrehozásának dátuma stb.

Név: Példa János

FMS azonosító: 1000000007

Kapcsolattartói e-mail: peldajanatos@fiktiv.hu

Másodlagos e-mail: jamesexample@test.net

Kapcsolattartói telefonszám: +36301234567

Másodlagos telefonszám: +361654321

Nem: Férfi

Születési hely: Magyarország, Balassagyarmat

Születési idő: 1985-07-12

Születési név: Példa János

Édesanya leánykori neve: Minta Emese

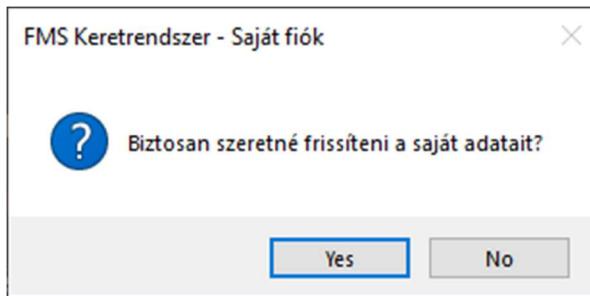
Fiók létrehozva: 2023-04-11 22:50:47

Fiók utoljára módosítva: 2023-04-11 22:53:36

Vissza (ESC) **Adatok módosítása (F9)**

63) ábra: Saját fiók (adatlap) munkaablak

Az adatok módosítása gombra kattintáskor, vagy az F9 billentyű megnyomásával elérhetővé válnak azok a menüpontok, amelyek szerkeszthetők a felhasználó által. Az adatok mentése gomb megnyomásával, vagy az F10 billentyű megnyomásával a felugró üzenetet jóváhagyva frissíthetők az adatmódosítások.



64) ábra: Saját adatok módosításának jóváhagyó felugró üzenete

- 2) Jelszó módosítása: A felhasználó saját jelszavát bármikor módosíthatja, a régi jelszó megadásával, illetve a kiválasztott új jelszó komplexitásának ellenőrzését követően. Az új jelszónak minimum 8 karaktert kell tartalmaznia, amelyből minimum 1 darab nagybetű, 1 darab kisbetű és 1 darab szám. Amennyiben a jelszó módosítása gombra kattintáskor, vagy az F10 billentyű lenyomásával az „Összes eszköz kijelentkeztetése” lehetőség előtt be van pipálva a jelölőnégyzet, akkor a jelszómódosítással egy huzamban a felhasználó összes munkamenetét inaktiválja a

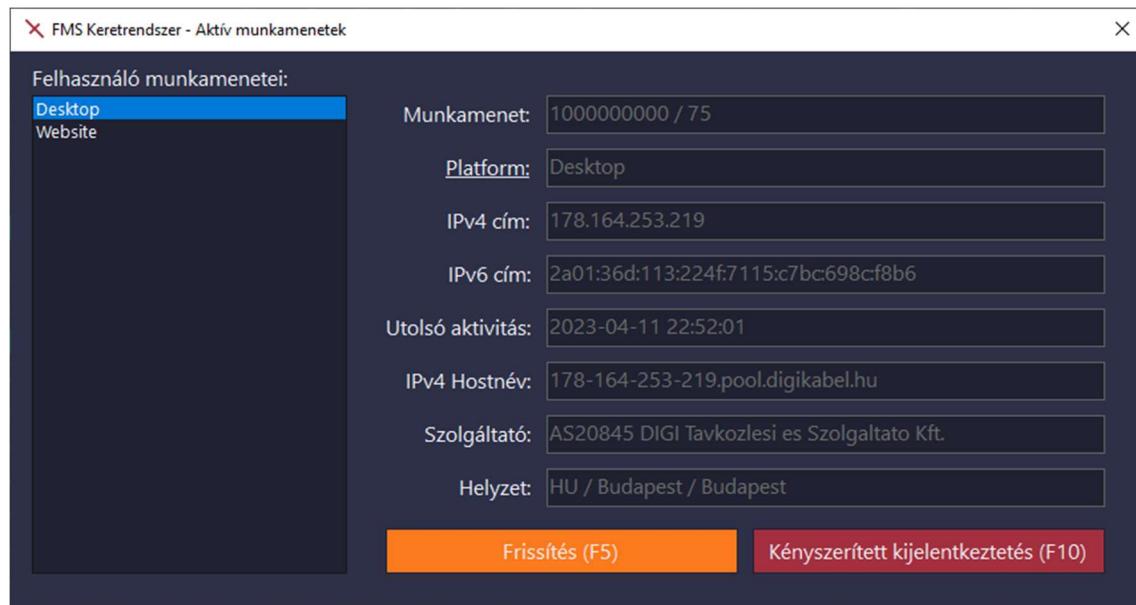
rendszer, így az összes platformon újra be kell jelentkezni az új jelszó használatával. Egy felugró ablakban kaphatunk értesítést a művelet sikerességéről, vagy pedig az adatok megfelelő hiánypótolásáról kapunk felszólítást egy hibaüzenetben.

65) ábra: Saját jelszó módosítása munkaablak

- 3) Jogosultságok (F5): Itt megtekintheti a felhasználó a saját aktív szerepköreit. Az inaktív szerepkörök a normál felhasználó számára nem kerülnek megjelenítésre. Az ablak bal oldalában kiválaszthatóak az aktív szerepkörök, amelyről további információk láthatóak az ablak jobb oldalában, többek között a felhasználó FMS azonosítója, a kiválasztott szerepkör neve, vállalat neve, létrehozás és utolsó módosítás ideje. Itt láthatjuk még, hogy az adott szerepkörhöz mikor rendeltük hozzá a felhasználót (szerepköri hozzáférés létrejöttének ideje), illetve a szerepköri hozzáférés utolsó módosításának idejét, (ideiglenes letiltás és engedélyezés keretében).

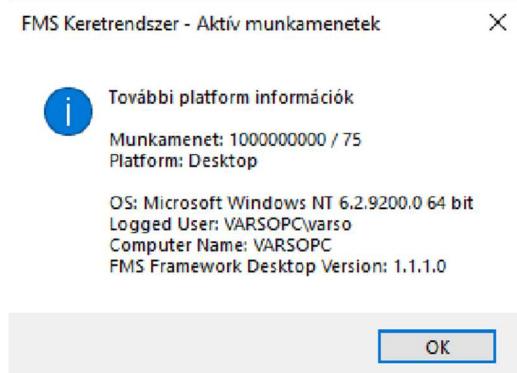
66) ábra: Jogosultságok munkaablak

- 4) Eszközök (F6): Itt láthatóak a felhasználó munkamenetei, amely kap egy sorszámot, amely a felhasználó azonosítójából és a munkamenet számából áll egy perjellel elválasztva. Ha ebben az esetben mobiltelefonról is bejelentkezik a felhasználó, az asztali alkalmazás használatával egyidejűleg, akkor annak is megjelenik a munkamenete, és adatai.



67) ábra: Felhasználó aktív munkameneti munkaablak

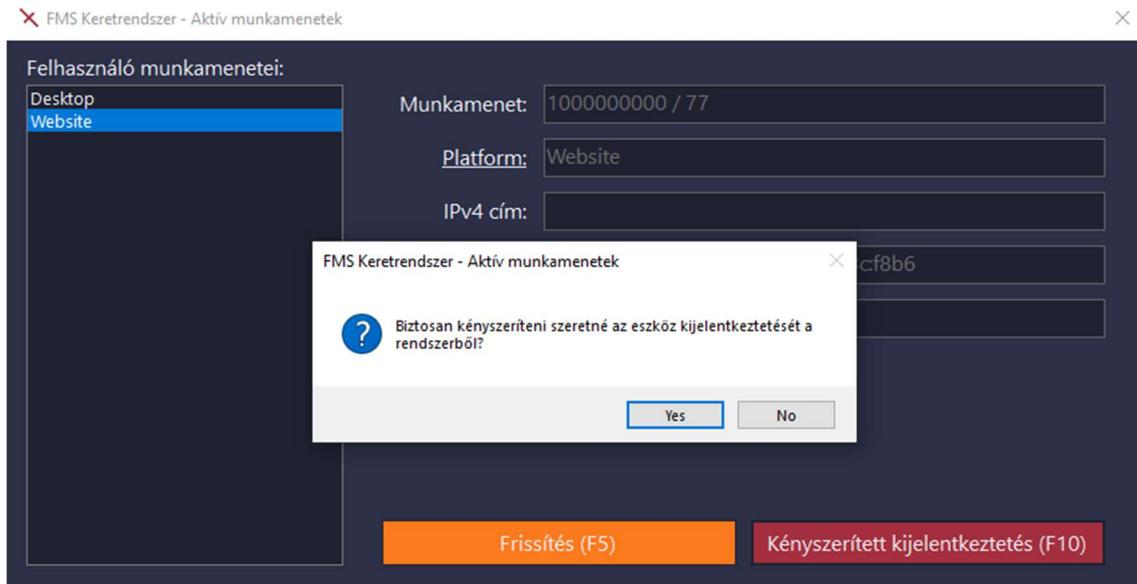
A platform mutatja, hogy milyen eszkösről történt a bejelentkezés, ha pedig a platform feliratra kattintunk, akkor egy felugró ablakban részletezi a további információkat, mint pl. telefon típusa, vagy az asztali gép operációs rendszerének típusa.



68) ábra: Munkamenet további információi felugró ablakban

Alatta láthatóak a publikus IPv4 és IPv6-os címek, amelyen keresztül megtörtént a csatlakozás. Az utolsó aktivitás megmutatja, mikor történt az adott munkamenetben legutóbb változás. Az IPv4 hostnév (publikus név), az adott IP címhez tartozó host nevet adja vissza. A szolgáltató az IPv4 IP cím alapján az autonóm rendszerben bejegyzett AS szám (Autonomous System Number) alapján a szolgáltató nevét is kirészletezi. A Helyzet pedig az IP cím körülbelüli helyzetét adja vissza. A frissítés gomb, vagy az F5-ös billentyű lenyomásával frissíthetjük az adatokat, valamint adott a lehetőség az eszközök kényszerített kijelentkeztetésére, melyet a kényszerített kijelentkezés gombra kattintva, vagy az F10 billentyű megnyomásával is megtehet a

felhasználó. A figyelmeztető ablakban rákérdez a rendszer, hogy biztosan szeretné-e a felhasználó elindítani az eszközről történő kijelentkezetét. A kijelentkeztetés sikereségéről egy felugró ablakban jelenik meg tájékoztatás.



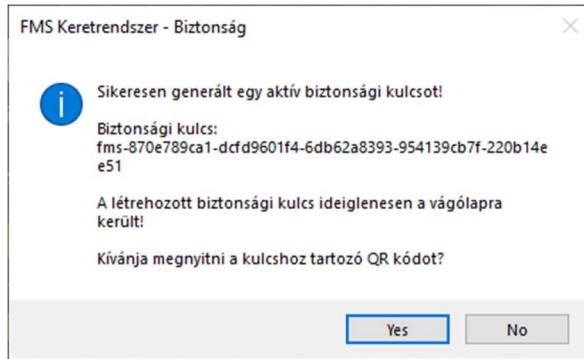
69) ábra: Munkamenet kényszerített kijelentkeztetése

- 5) Biztonság: Itt kap lehetőséget a felhasználó, hogy biztonsági kulcsot generáljon, illetve a felhasználói fiókját véglegesen törölje az FMS rendszeréből. A saját jelszó megadását követően elérhetővé válik a két lehetséges eljárás elindítása.



70) ábra: Saját fiók kezelése a biztonsági munkaablakban

A biztonsági kulcs generálásánál elsősorban egy figyelmeztető üzenet jelenik meg, amely felhívja a figyelmet a kulcs generálásával adódó veszélyekre. Az üzenet jóváhagyásával legenerálódik a biztonsági kulcs, amely megjelenítésre, illetve vágólapra kerül.



71) ábra: *Sikeres biztonsági kulcs generálásról szóló felugró üzenet*

Amennyiben a biztonsági kulcsot meg szeretné a felhasználó jeleníteni QR kód formátumban, akkor az üzenetet jóváhagyva megjelenik a kulcsot tartalmazó kód, amely lehetővé teszi a mobil alkalmazásban az egyszerű beolvasást.



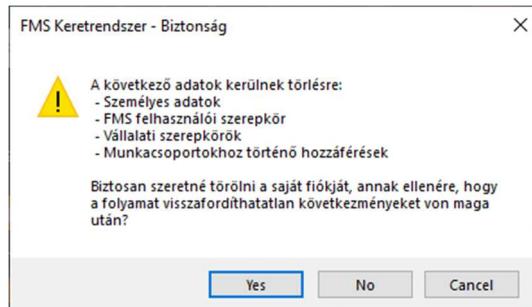
72) ábra: *Biztonsági kulcs megjelenítve QR kódos formátumban*



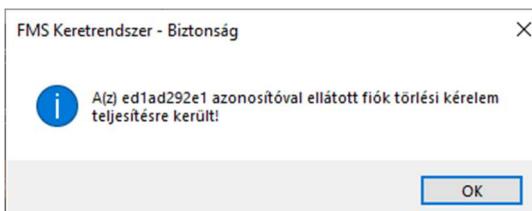
73) ábra: *Megjelenített QR kód beolvasásának értéke*

Amennyiben a felhasználói fiók adatait szeretné törlni a bejelentkezett személy az FMS rendszerből, akkor a fiók törlése gombra kattintva kérelmezheti a személyes adatok eltávolítását a biztonsági kérdések elfogadását követően. A fiók törlése nem

visszafordítható, visszaállítása nem lehetséges. Ez esetben az összes személyes adat, alap FMS felhasználói és vállalati szerepkörök, munkacsoportokhoz való hozzáférés - minden törlésre kerülnek. A törlést követően az összes munkamenet lezárásra kerül, bejelentkezésre nem kerülhet sor.



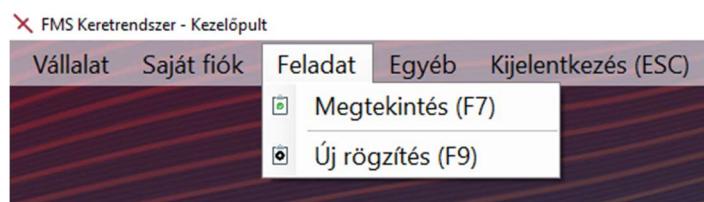
74) ábra: Felhasználói fiók törlésre felszólító üzenet



75) ábra: Sikeres felhasználói fiók törlés üzenete

A törlési kérelem azonosítója ezek után hozzárendelésre kerül a személyes adatok helyére, így ezek után a felhasználó beazonosítása nem lehetséges.

2.3.5 Feladat menüpont



76) ábra: Feladat menüpont

- 1) Megtekintés (F7): Amennyiben a bejelentkezett felhasználónak van jogosultsága a feladatok megtekintésére, akkor megjelenik egy táblázat, amelyben az összes feladat megjelenítésre kerül, az elvégzendő feladatok előresoroval és létrehozás dátuma szerinti csökkenő sorrendbe rendezve. A táblázatban megtaláljuk a feladat azonosítóját, munkacsoportját, leírását, „státuszát” vagyis állapotát (zöld pipa esetén elvégzett, piros x esetén elvégzendő), határidejét, létrehozási idejét és utolsó módosítás dátumát. Az ablak bal alsó sarkában található frissítés gombra kattintva, vagy az F5 billentyű segítségével lehet aktualizálni (frissíteni) a feladatok listáját. Az

ablak jobb alsó sarkában lévő új feladat gombra kattintva, vagy az F9 billentyűt megnyomva (amennyiben van megfelelő jogosultsága a felhasználónak) megjelenik az új feladat létrehozása munkablak, ahol az adatok kitöltésével új feladatot hozhatunk létre, meglévő munkacsoport részére. A táblázatban kiválasztott feladat információit az F10 gomb megnyomásával, vagy az információk gombra kattintással nyithatjuk meg.

#	Munkacsoport	Feladat	Státusz	Létrehozó	Határidő	Létrehozva	Módosítva
23	Fejlesztők	Projektmunka leadása	✗	Varsányi Barnabás	2023-04-28 08:30:00	2023-03-29 11:22:05	2023-03-29 22:38:31
10	Fejlesztők	További ötletek	✗	Medve Adrián	-	2023-03-24 18:02:54	-
11	Desktop Dev	DataGridview többsoros megjelenítés	✓	Varsányi Barnabás	-	2023-03-27 09:07:41	2023-03-27 12:02:49
9	Desktop Dev	Biztonsági kulcsok rendszer elkészítése	✓	Varsányi Barnabás	-	2023-03-24 12:02:32	2023-03-29 11:13:04
4	Desktop Dev	Válalati munkacsoportok javítás	✓	Varsányi Barnabás	-	2023-03-22 08:27:02	2023-03-22 13:56:23
1	Fejlesztők	Első feladat a fejlesztői munkacsoport számára	✓	Varsányi Barnabás	-	2023-03-22 08:16:38	2023-03-27 08:35:02

Frissítés (F5) 6 / 6 darab Új feladat (F9) Információk (F10)

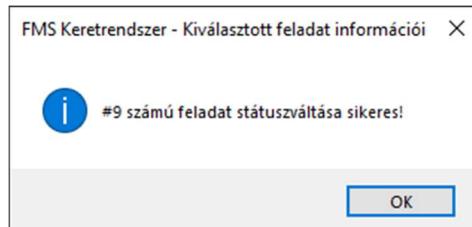
77) ábra: Feladatok munkablak

A „Kiválasztott feladat információi” munkablak bal oldalán találhatók a feladat részletei, többek között a feladat azonosítója, státusza szövegesen, létrehozás ideje, utolsó módosítás ideje, határideje, feladat létrehozójának a neve, illetve a feladat rövid leírása. Az ablak jobb oldalán találhatóak a feladathoz fűzött megjegyzések, üzenetek.

Információk		Üzenetek	
Feladat azonosító:	Desktop Dev - #9	Üzenet	Rögzítő
Státusz:	Elvégzett	Szöveg (Elvégezve)	Varsányi Barnabás
Létrehozva:	2023-03-24 12:02:32	Majdnem kész, már csak a szerepkörök hozzáadását kell megoldani	Varsányi Barnabás
Módosítva:	2023-03-29 11:13:04	Hozzáadva Winapphoz, tesztelés szükséges, illetve API jelszóval való	Varsányi Barnabás
Határidő:	-	UI kész, lekérdezés winapphoz hozzáadás szükséges	Varsányi Barnabás
Létrehozó:	Varsányi Barnabás	API sso/user/secretkey hivatkozás elérhető!	Varsányi Barnabás
Feladat:	Biztonsági kulcsok rendszer elkészítése		
<input type="button" value="Szövegváltás (F10)"/> <input type="button" value="Törölés (F11)"/>		<input type="button" value="Frissítés (F5)"/> 5 / 5 darab <input type="button" value="Új (F9)"/>	

78) ábra: Kiválasztott feladat információi munkablak

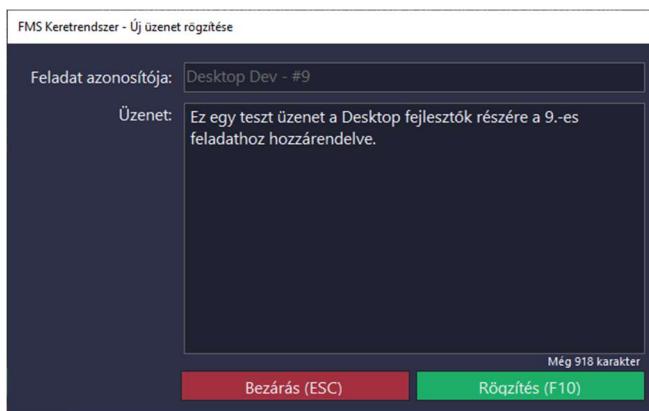
A feladatok „státuszát” a státuszváltás gombra kattintva, vagy az F10 billentyűt lenyomva lehet megtenni. Két féle állapot lehetséges a rendszerben, amely a váltás alkalmával az ellentett állapotára változik. Például: Elvégzendő feladatnál a váltás után elvégzett lesz. Az elvégzett feladatokat újra elvégzendőre állítani csak az a felhasználó teheti meg, akinek a szerepköre rendelkezik a feladatok teljes irányítási jogosultságával. A váltás eredményéről egy felugró ablakban jelenik meg értesítés.



79) ábra: Feladat állapotváltozásának sikerességeiről tájékoztatás

A feladatok törlése a törlés gombra kattintva, vagy az F11 billentyű lenyomásával tehető meg, amennyiben a felhasználó rendelkezik a feladatok teljes irányítási jogosultságával. A folyamat törli a feladathoz fűzött összes megjegyzést és a feladatot.

A feladatokhoz új üzenetet az új gombra való kattintással, vagy az F9 billentyű lenyomásával tehető meg. Ekkor a megjelenő új üzenet munkablakban gépelhetjük be az üzenetünket maximum 1000 karakter terjedelemben. A rögzítés gombra kattintásakor, vagy az F10 billentyű lenyomásával feljegyzésre kerül az új üzenet, ezek után a bezárás, vagy az ESC billentyű lenyomásával kiléphetünk az új üzenet rögzítése munkablakból.



80) ábra: Feladathoz új üzenet rögzítése

Amennyiben a felhasználó szeretné látni az újonnan hozzáfűzött üzeneteket, akkor a frissítés, vagy az F5 billentyű segítségével lefrissíthetők az adatok. Fontos, hogy az FMS rendszeréből törölt felhasználóknak a rögzített feladatai és üzenetei nem

kerülnek törlésre, az adatvesztés elkerülése érdekében. A törölt felhasználó neve helyett a törlési kérelem azonosítója jelenik meg a rögzítő nevénél.

Üzenetek		
Üzenet	Rögzítő	Létrehozás
Státszváltozás (Elvégezve)	Törölt felhasználó ed1ad292e1	2023-04-11 23:29:01
Hatóridő előtt készre jelentem a feladatot!	Törölt felhasználó ed1ad292e1	2023-04-11 23:28:56
Hatóridő betartása fontos!	Varsányi Barnabás	2023-04-11 23:28:24
Teszt meqeqyzés	Törölt felhasználó ed1ad292e1	2023-04-11 23:27:57

81) ábra: Törölt felhasználó üzenetei a kiválasztott feladat információi munkaablakban

- 2) Új rögzítés (F9): Megfelelő jogosultságok mellett a megjelenő ablakban tudunk rögzíteni új feladatot a kiválasztott munkacsoporthoz. Az első mezőben a vállalat neve jelenik meg, a második lenyíló objektumban a vállalat munkacsoporthoz kötött választhatjuk ki a megfelelőt. Amennyiben szeretnénk a feladathoz határidőt rendelni, akkor a dátumválasztó mellett a jelölőnégyzet bepipálásával aktívvá válik a dátumválasztás lehetősége. Itt a nap kiválasztása után az időt kézzel is felvihetjük a beviteli mezőbe. A feladat címét maximum 100 karakterben kötelező megadni. A rögzítés gombra kattintva, vagy az F9 billentyűt lenyomva a feladat létrehozásra kerül a rendszerben. Az ablakot a bezárás gombra kattintva, vagy az ESC billentyű lenyomásával zárhatjuk be. Ez a munkaablak jelenik meg a feladatok munkaablak új feladat gombra kattintva, vagy az F9 billentyűt lenyomva.

FMS Kerrendszer - Új feladat - FelX

Vállalat: FelX

Munkacsoporthoz kötött: Desktop Dev

Hatóridő: 2023-05-01 12:00:00

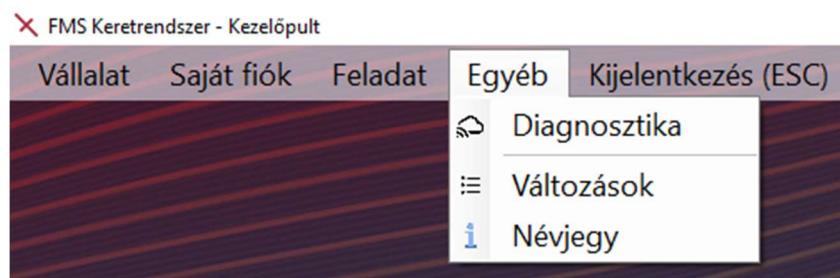
Cím: Teszt határidős feladat

Még 77 karakter

Bezáras (ESC) Rögzítés (F9)

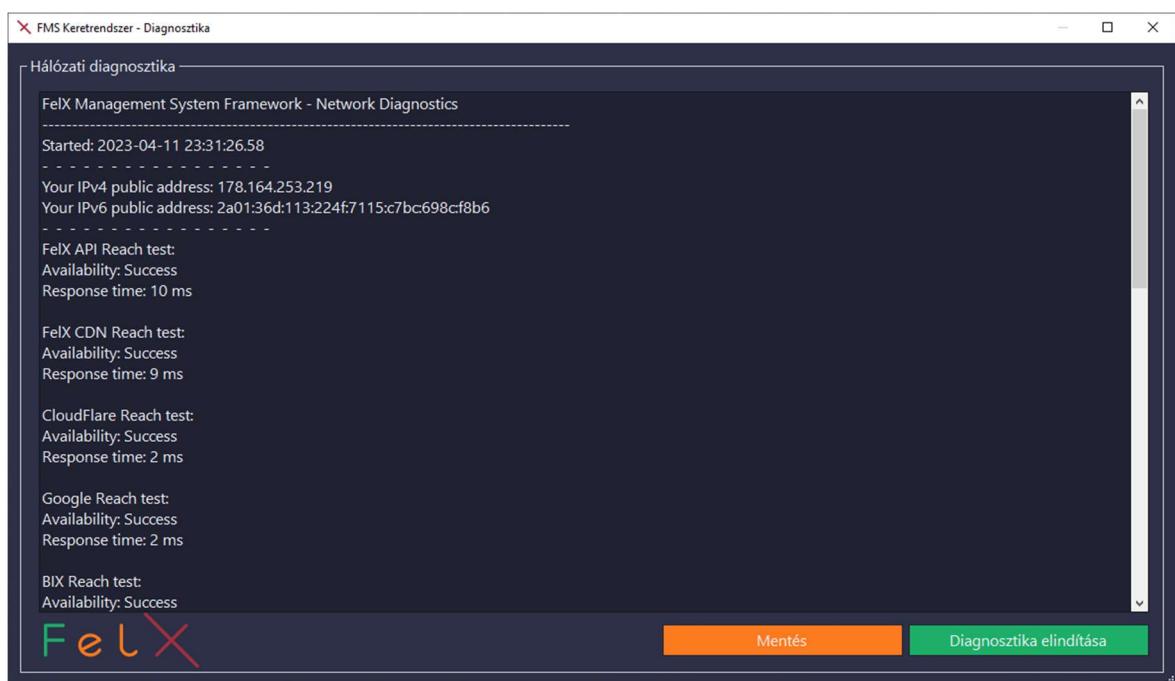
82) ábra: Új feladat rögzítése munkacsoporthoz számára munkaablak

2.3.6 Egyéb menüpont



83) ábra: Egyéb menüpont

- 1) Diagnosztika: Hiba esetén a felhasználók számára lehetőség nyílik egy hálózati öntesztet elindítani, amely tartalmazza a következő adatokat: Publikus IPv4 és IPv6 cím, FELX API és FELX CDN (Content Delivery Network) tartalommegosztó hálózat, Cloudflare, Google DNS szerver és BIX (Budapest Internet Exchange) elérését (availability) és válaszidejét (response time) ezredmásodpercben (ms), aktív hálózati adapterek információit, alapértelmezett átjárók elérését és válaszidejét, valamint a teszt lefutásának idejét. A diagnosztikai adatokkal megfelelően kinyerhetők a lehetséges hibaforrások. A teszt a jobb alsó sarokban található diagnosztika elindítása gombra való kattintással elindítható, majd a lefutást követően a mentés gombra kattintva megjelenik egy fájlkezelő ablak, ahol kiválaszthatjuk a tesztről készült jelentés (riport) helyét, itt tetszőleges fájlnevet adhat meg a felhasználó. A diagnosztika nem gyűjt semmilyen egyéb információt a felhasználó számítógépéiről és hálózatáról, illetve nem is továbbít semmit sem az FELX szerverei felé.

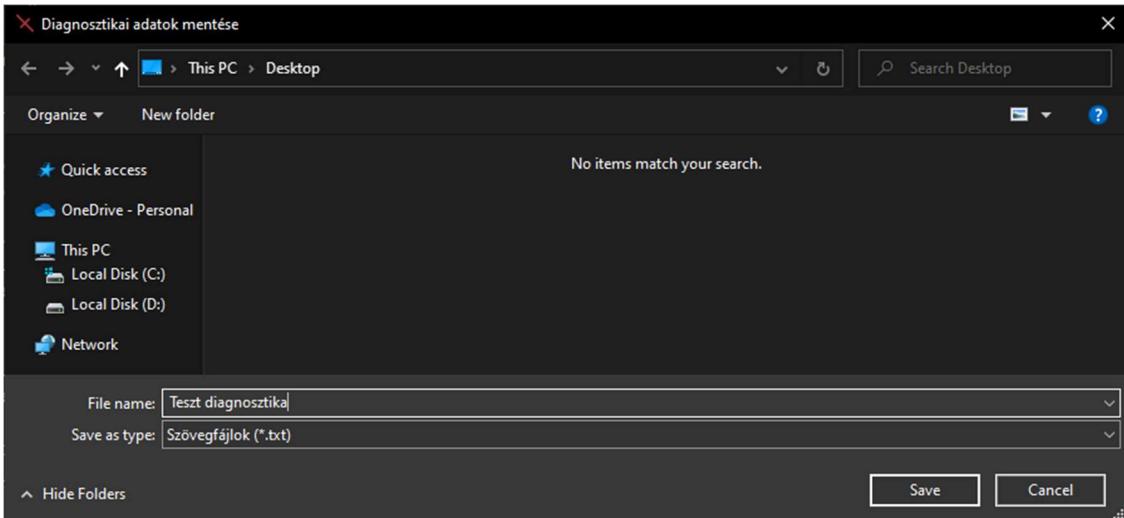


84) ábra: Lefuttatott sikeres diagnosztika munkaablak

jelenség	FELX API	FELX CDN	Cloudflare	Google	BIX	alapért. átjárók			
a.	<50 ms								
b.	>100 ms	<50 ms							
c.	>100 ms				<50 ms				
d.	>100 ms								
e.	TimedOut	Success							
f.	TimedOut				Success				
g.	TimedOut								

Jelenségek:

- a. Normál működés: minél kevesebb a hálózati válaszidő, annál dinamikusabb, gyorsabb lesz az alkalmazás kommunikációja a szerverrel.
- b. Szerver oldali lassulás: a szerver hálózata leterhelt, az alkalmazás működésében lassulás tapasztalható.
- c. Felhasználó oldali lassulás: az alapértelmezett átjáró (router vagy gateway) és a szerver felé terhelt, vagy hibás a hálózat. Ez esetben érdemes a felhasználónak felkeresnie az internetszolgáltatóját a hálózati eszközök újraindítását követően.
- d. Felhasználó oldali lassulás: az alapértelmezett átjáró (router vagy gateway) és a felhasználó eszköze közötti kommunikáció lassú, vagy hibás. Ez esetben érdemes ellenőrizni a felhasználónak az eszközét, vezetékes kapcsolat esetében a vezeték lehetséges sérüléseit, vezeték nélküli kapcsolat esetén javítani az eszközök közötti jelszintet, és csökkenteni a kommunikációt zavaró tényezőket (nagy elektromágneses területek kerülése, mikrohullámú sütő kikapcsolása, légköri tényezők ellenőrzése).
- e. Szerver oldali hiba: A távoli szerver jelenleg nem elérhető, érdemes ellenőrizni a <https://status.felx.hu/> weboldalt.
- f. Felhasználó oldali hiba: az alapértelmezett átjáró (router vagy gateway) nem csatlakozik az internethöz. Ez esetben érdemes a felhasználónak felkeresnie az internetszolgáltatóját a hálózati eszközök újraindítását követően.
- g. Felhasználó oldali hiba: az alapértelmezett átjáró és a felhasználó eszköze között nincs kommunikáció. Ez esetben érdemes ellenőrizni a felhasználónak a vezetékes, vagy vezeték nélküli kapcsolatot.



85) ábra: Diagnosztikai adatok mentése

```

Teszt diagnosztika.txt - Notepad
File Edit Format View Help
Type: Ethernet
MAC Address: 00155D7F6985
Speed: 1000000000
Multicast: Supported
Received Data: 0
Sent Data: 4986381

Name: Ethernet (Intel(R) I211 Gigabit Network Connection)
Type: Ethernet
MAC Address: B06EBFBFEEEA
Speed: 1000000000
Multicast: Supported
Received Data: 4648021241
Sent Data: 102610167

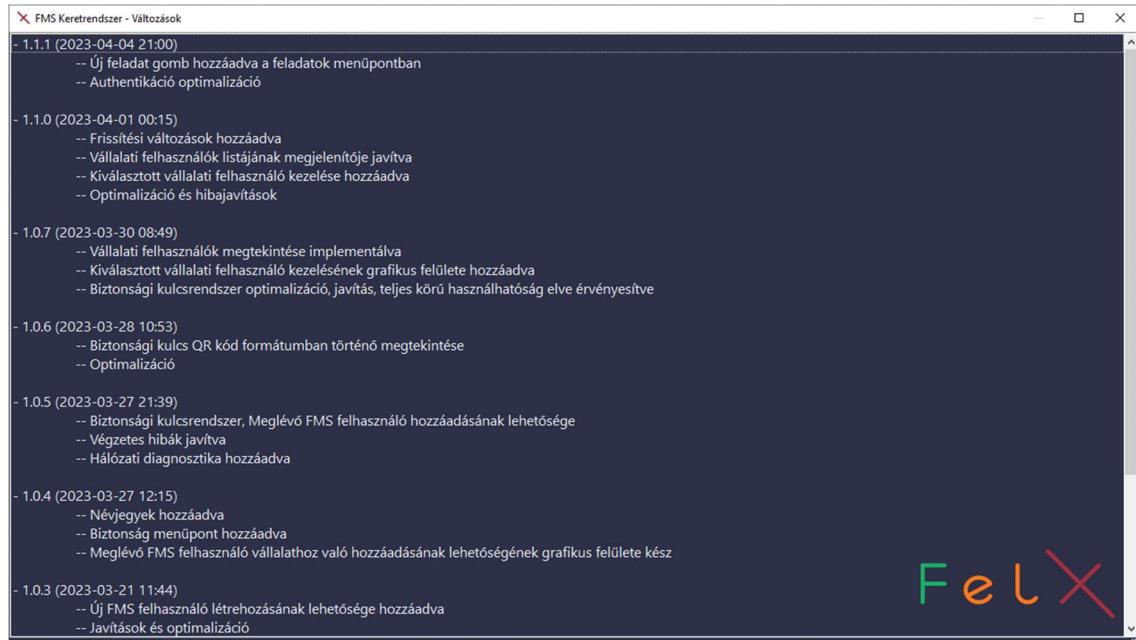
Network Gateway (fe80::6d4:c4ff:fedb:a5a4%2) test:
Availability: Success
Response time: 0 ms

Network Gateway (192.168.1.254) test:
Availability: Success
Response time: 0 ms

```

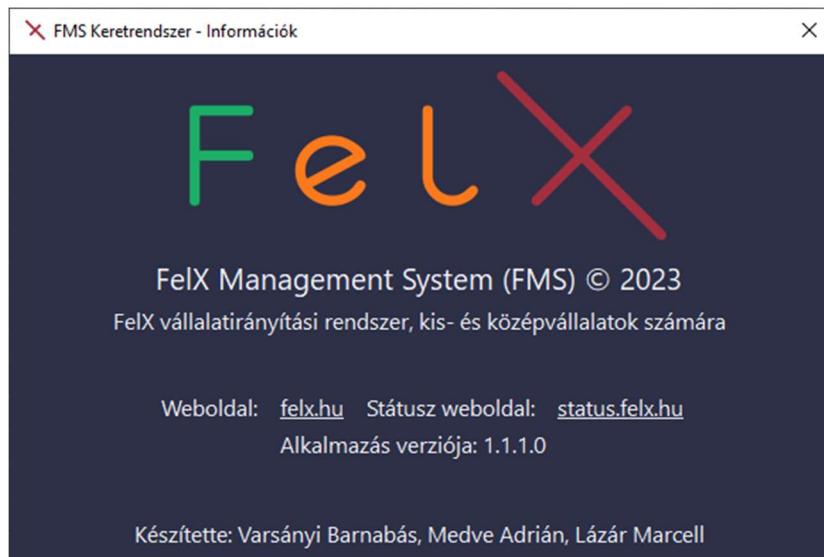
86) ábra: Lementett diagnosztikai adatok szövegszerkesztő szoftverben

- 2) Változások: A megjelenő ablakban megtekinthetjük az alkalmazás frissítésével és verziószám változásával bekövetkezett újításokat és változásokat, csökkenő időrendi sorrendben.



87) ábra: Asztali alkalmazás változásai ablak

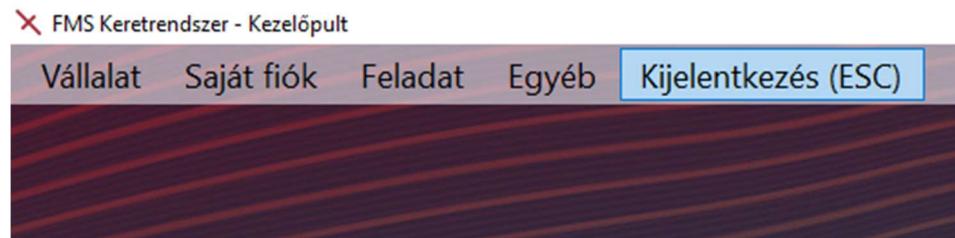
- 3) Névjegy: Itt megtekinthető az alkalmazás névjegye, megnyitható az alapértelmezett böngészőben a <https://felx.hu/> és a <https://status.felx.hu/> weboldal a linkre kattintással. Látható még az alkalmazás aktuális verziója, illetve a FelX fejlesztők nevei is.



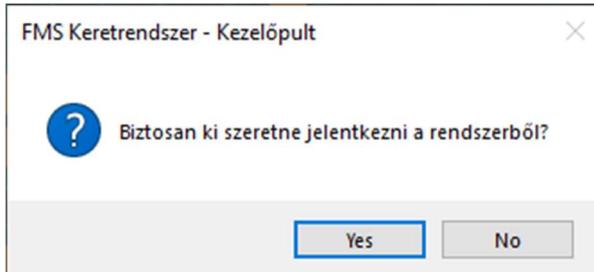
88) ábra: Névjegy ablak

2.3.7 Kijelentkezés menüpont

Az utolsó menüpont a kijelentkezés, amely segítségével lehetőség nyílik a munkamenet lezárására a kijelentkezésre kattintva, vagy az ESC billentyű lenyomásával felugró ablak jóváhagyásával.



89) ábra: Kijelentkezés menüpont



90) ábra: Kijelentkezés megerősítése felugró ablak

3. Fejlesztői dokumentáció

3.1 API

Az API (Application Programming Interface) egy olyan közös felület a rendszerünkben, amely összeköttetést biztosít az alkalmazások és az adatbázis között. Az API objektumorientált PHP nyelven íródott. 43 lekérdezés járul hozzá a megfelelő adatok adatbázisban történő hozzáadásához, módosításához, törléséhez és lekérdezéséhez. A csatlakozás helyi hálózaton belül történik, így az adatbázishoz való csatlakozás távolról a tüzfalszabályok megfelelő beállításával letiltásra került.

Az API kérések GET és POST módszerekkel kommunikálnak, az eredménye 200-as HTTP státuszkódnál, JSON formátum. A kliens által küldött lekérdezés HTTPS protokollon, Cloudflare által biztosított védelmen (proxy) keresztül ad választ a FELX szervere. A Cloudflare szerverei és a FELX szervere között teljes hitelesítés van, HTTP (80-as) porton az érdemleges információk közlése nem engedélyezett. A <http://api.felx.hu:80/> automatikusan átirányít a <https://api.felx.hu:443/> hivatkozásra.

A PHP fájlok funkcionalitásukat figyelembe véve lettek kategorizálva, amely a későbbiek folyamán is egyszerűen bővíthető:

- 1) Auth: Olyan hitelesítéssel kapcsolatos fájlok vannak eltárolva, amely a hitelesítéssel és a bejelentkezással kapcsolatos főbb műveleteket tartalmazza.
- 2) Info: Itt érhetőek el a közérdekű információk, amelyek nem igényelnek semmiféle hitelesítést.

- 3) Mail: E-mail rendszerrel kapcsolatos fájlok.
- 4) QR: Kétdimenziós vonalkód generálásával kapcsolatos fájlok.
- 5) SSO: Egyszeri hitelesítés által generált kóddal (token) történő fájlok.
- 6) Syslog: Rendszernaplóval kapcsolatos fájlok.
- 7) Test: API tesztelésére használt fájlok.

3.1.1 Adatbázishoz történő csatlakozás

A csatlakozási adatok konstansként kerültek rögzítésre a connection.php fájlban, ezen kívül a projektben sehol sem található eltérő adatok, a káros redundancia elkerülése érdekében.

Elsősorban definiálásra került, hogy a továbbiakban megadott szerver elérési címe pontosan milyen adatot tartalmaz. Amennyiben a szerver címe IP címet tartalmaz, akkor a „HOSTCONN” konstans értékét hamisra kell állítani, amennyiben DNS névfeloldás szükséges az eléréshez.

```
define("HOSTCONN", TRUE);
```

A következő „SERVER” konstans értéke lesz az adatbázis szerverének elérési címe. A FELX esetében a webszolgáltatás (Apache2) mellett fut azonos szerveren a MySQL adatbázis, így a PHP a saját IP címén csatlakozik. A „localhost” DNS által feloldott címe a 127.0.0.1, de használható még jelenleg a csatlakozáshoz az „s2.varsonet.intra” domain vagy a 192.168.1.220 IP cím is.

```
define("SERVER", "localhost");
```

A csatlakozás során szükséges megadni olyan hitelesítő adatokat, amelyek az utolsóként hozzáadott „DATABASE” konstans értékhez hozzáfér. A felhasználónév a „USER” értékeként kell megadni, a jelszót pedig a „PASSWORD” konstansként definiáljuk.

```
define("USER", "fms_api1");
define("PASSWORD", "W.JrpFXfrknwJY8u");
define("DATABASE", "fms_main");
```

Ezek után szükségessé vált egy olyan modell létrehozása, amely során a megadott hitelesítő adatokat ne legyen szükséges megadni, de felesleges adatbázis csatlakozás ne maradjon a rendszerben, csak azok, amelyek aktív lekérdezést folytatnak. Ennek érdekében egy osztály létrehozása mellett döntöttem.

```
class MainDatabase { }
```

A létrehozott osztályban létrehozásra került négy darab adattag, amelyek közül az első tárolja az adatbázis csatlakozást, második a csatlakozás jelenlegi állapotát, az utolsó állapotváltozás idejét, és a csatlakozáshoz köthető üzeneteket, amelyek hiba esetén

tartalmazzák a MySQL szabványos hibaüzeneteit, normál működés mellett pedig a csatlakozás sikerességéről kapunk szöveges üzenetet. A „conn” változó itt még nem került példányosításra, a másik három esetében alap adatokkal kerültek feltöltésre.

```
public $conn;  
private $connStatus = FALSE;  
private $connStatusTime = "1990-01-01 01:00:00";  
private $connStatusMessage = "Default";
```

Ezek után az osztályon belül kerülnek létrehozásra az adatbázis csatlakozásának kezelésére szolgáló metódusok.

Az első metódus a „Connect”, amelynek első sorában a csatlakozás változásainak időpontja, információgyűjtés céljából aktualizálásra (beállításra) kerül.

```
function Connect() {  
    $this->connStatusTime = date("Y-m-d H:i:s");}
```

Ezek után egy elsőleges hibakeresési folyamat fut le a csatlakozás előtt. Amennyiben a „HOSTCONN” értéke igaz és a „SERVER” értéke megegyezik a „gethostbyname” függvény értékével, akkor a „connStatus” értéke hamisra kerül beállításra, illetve a „connStatusMessage” értéke tárolja el a hibának a forrását. A „gethostbyname” függvény paramétereként átadott „SERVER” konstans érték, DNS névfeloldást követően visszaadja a kiszolgáló IP címét, amennyiben nem sikerül a névfeloldás, a „SERVER” értékét adja vissza változatlanul. Így ha „HOSTCONN” értéke igaz, azaz név alapján szeretnénk csatlakozni, akkor egyszerűen lehet kezelní a névfeloldási problémát úgy, hogy az API nem kerül felesleges lassításra, helytelen „mysqli” osztály létrehozásával, mivel a return megszakítja a folyamatot.

```
if (HOSTCONN == TRUE && gethostbyname(SERVER) == SERVER) {  
    $this->connStatus = FALSE;  
    $this->connStatusMessage = "Nem sikerült feloldani az adatbázis szerverének a hostnevét!";  
    return;}
```

Az ellenőrzést követően egy kivételkezelésen belül példányosításra kerül a „conn” változó, amennyiben hiba lép fel, akkor a „connStatus” értéke hamis lesz, és a csatlakozási üzenet, amely a „connStatusMessage” egyenlő lesz a keletkezendő hibaüzenettel, és a „Connect” metódus megszakításra kerül.

```
try {  
    $this->conn = new mysqli(gethostbyname(SERVER), USER, PASSWORD,  
    DATABASE);  
} catch (Exception $e) {  
    $this->connStatus = FALSE;  
    $this->connStatusMessage = $e->getMessage();  
    return;}
```

Az utolsó hibakezelésben olyan hibák kerülnek felkutatásra, amelyet a MySQL kapcsolat ad vissza az API számára a csatlakozást követően. Ilyenek lehetnek például a hibás felhasználónév vagy jelszó, vagy akár nem létező adatbázis név a „DATABASE” konstansban. Amennyiben található csatlakozási hiba, a „connStatus” hamis, „connStatusMessage” értéke pedig a MySQL szerver által adott hibaüzenet. Ellenkező esetben a „connStatus” igaz lesz, és a csatlakozási üzenet megkapja a „Connected!” kifejezést.

```
if ($this->conn->connect_error) {
    $this->connStatus = FALSE;
    $this->connStatusMessage = $conn->connect_error;
} else {
    $this->connStatus = TRUE;
    $this->connStatusMessage = "Connected!";
}
```

A csatlakozás bezárására is készült metódus, amelyeket a későbbiek folyamán a lekérdezések után hajtunk végre. Ez esetben, ha van aktív kapcsolat, akkor bontja elsősorban a kapcsolatot, a „connStatus” értéke hamis lesz, a „connStatusTime” egyenlő lesz az aktuális idővel, majd az üzenet értéke „Manually Disconnected!” lesz.

```
function Disconnect() {
    if ($this->conn) {
        mysqli_close($this->conn);
        $this->connStatus = FALSE;
        $this->connStatusTime = date("Y-m-d H:i:s");
        $this->connStatusMessage = "Manually Disconnected!";
    }
}
```

A csatlakozás aktuális állapota lekérdezhető a „Status” függvény meghívásával, amelynek visszatérési értéke JSON formátumban kerül kódolásra. Fontos, hogy itt a manuális szándék kérdezhető le a csatlakozások alatt, tehát a csatlakozást követően fellépő hibák (például hálózati hiba) esetén az értékek nem kerülnek módosításra, mivel csak az osztályban található változók kerülnek kiíratásra.

```
function Status() {
    $answer = [
        "success" => TRUE,
        "result" => [
            "connStatus" => $this->connStatus,
            "connStatusTime" => $this->connStatusTime,
            "connStatusMessage" => $this->connStatusMessage,
        ],
        "message" => "Sikeres státuszlekerdezés!",
    ];
    return json_encode($answer, JSON_UNESCAPED_UNICODE);}
```

Az egyszerűsített csatlakozás ellenőrzéseként a fájlban utolsó függvényként létrehozásra került az „`isConnected`”, amelynek visszatérési értéke olyan logikai érték, amely „`connStatus`” értékét adja vissza.

```
function isConnected() { return $this->connStatus; }
```

3.1.2 Szerveridő

Az adatbázis által visszaadott jelenlegi időt értjük a szerveridő kifejezése alatt, amely bemutatja azt az alap modellt, hogy az adatbázis csatlakozásból hogyan kerülnek az adatok kiíratásra JSON formátumban. A szerveridő-lekérdezés az alap mintája a többi lekérdezésnek.

A fejlécben (header) elsődleges beállításra kerül a tartalom típusa, amelyeket a böngészők felismerve különböző beállítások futnak le. Például: json formátum esetén a Google Chrome böngésző másik betűtípushat használ, felveszi az operációs rendszer színösszeállítását, stb...

```
header("Content-type: application/json");
```

A fejlécben még beállításra kerül, hogy a CORS (cross-origin resource sharing) előírja a webkiszolgálók és a böngészők közötti cserélt adatforgalmat és ellenőrzi, hogy a megfelelő helyről érkezik-e a kérés. Ahhoz, hogy weboldalon keresztül is felhasználhatók legyenek azok az adatok, amelyeket válaszként kap meg a böngésző, engedélyezni kell a webhelyek közötti erőforrást, amely a „`Access-Control-Allow-Origin`” segítségével lehet beállítani. minden webhelynek engedélyeztük az elérését és az erőforrás-cserét, a CORS szabályok miatt.

```
header("Access-Control-Allow-Origin: *");
```

A lekérdezés engedélyezett típusait is érdemes beállítani, így tudatva a böngészővel, illetve a különböző platformokkal, hogy melyek azok az engedélyezett típusok. Jelen esetben a szerveridőnél a GET lekérdezés a mérvadó.

```
header("Access-Control-Allow-Methods: GET");
```

Az adatbázishoz való csatlakozás felhasználásával egyszer be kell „importálni” a `connection.php` fájlt, ami tartalmazza a „`MainDatabase`” osztályt, illetve a beállított csatlakozási adatokat, csatlakozást és hibakezeléseket. A fájl elérési útja a mi esetünkben a <https://api.felx.hu/> weboldal, az operációs rendszerben a „/var/www/fms/api.felx.hu” weboldal könyvtárának a helye, de mivel a mappaszerkezet változása miatt el akartuk kerülni a hibákat, így az összes fájl dinamikusan kapja meg a weboldal könyvtárának a helyét az operációs rendszer gyökér-mappájától nézve, amelyet a „\$_SERVER['DOCUMENT_ROOT']” határoz meg.

```
require once($_SERVER['DOCUMENT_ROOT']."/connection.php");
```

Így, hogy már biztosítva van a hozzáférés a MainDatabase osztályhoz, példányosításra kerül egy objektum, amelyet folyamatosan használendő a lekérdezéseknel, illetve a csatlakozásnál.

```
$db = new MainDatabase();
```

Az adatbázisból való lekérdezés előtt csatlakoznunk kell az adatbázishoz, ha sikerült a csatlakozás (amely az „*isConnected*” függvényel ellenőrizhető), akkor készíthető elő az SQL utasítás és a hozzá kapcsolódó opcionális paraméterek.

```
$db->Connect(); // Csatlakozás az adatbázishoz  
if ($db->isConnected()) { }
```

A sikeres csatlakozás után előkészíthetjük a lekérdezést a „*prepared statements*” használatával, ami biztonsági szempontból jelentős tulajdonságokkal bír, amikor különböző paraméterek hozzáadása szükséges, mivel a paraméterként elküldött adatok teljes mértékben szövegként kerül továbbításra, így az „SQL injection” ellen is egy jó védelem a beépített PHP függvényeken kívül. Jelen esetben az SQL utasítás a dátumot, időt és az egyben lefűzött dátum-idő párost tartalmazza.

```
$stmt = $db->conn->prepare("SELECT CURRENT_DATE AS 'serverdate',  
CURRENT_TIME AS 'servertime', CURRENT_TIMESTAMP AS  
'servertimestamp';");
```

Jelen esetben nincs szükség különböző paraméterek megadására, így az SQL utasítás rögtön végrehajtható az „*execute*” utasítással.

```
$stmt->execute();
```

A lekérdezés eredménye eltárolásra kerül egy „*requestResult*” változóban a „*get_result*” metódus segítségével, amelyből kinyerhetők soronként az adatok a „*fetch_assoc*” metódus segítségével. Jelen esetben csak egy sor lekérdezésére van szükség. Amennyiben több rekordot tartalmazna a lekérdezés, akkor „*while*” ciklus segítségével, végig lehet egyesével lépni a sorokon.

```
$requestResult = $stmt->get_result();  
$result = $requestResult->fetch_assoc();
```

Egy sor egy darab objektumnak felel meg, ezért egy létrehozott tömbhöz hozzárendelhetők az objektumok, a későbbi megjelenítés vagy adatfeldolgozás érdekében. A szerveridő lekérdezése során egy rekord jön létre, és ez fog megjelenítésre kerülni egy sablon szerint. minden API lekérdezés általában egy egyéni szabványt fogalmaz meg, mivel 3 darab kulcs jelenik meg, az első a „*success*”, ami a lekérdezés sikerességét foglalja magában egy logikai értékkal, a második a „*result*”, amely általában további objektumokat, listákat tartalmaz, itt kerülnek megjelenítésre érdemleges információk, lekérdezések eredményei, sikertelenség esetén részletező üzenetek. Az

utolsó a „message”, amely a felhasználók számára is rendelkezésre bocsátható üzenetet tartalmaz, amely teljesen felhasználóbarát. Az API nagy részében ez a végkimenete egy GET vagy POST lekérdezésnek az alkalmazáson belül, legtöbbször az „msg” változónévvel kerül elnevezésre.

```
$msg = [
    "success" => TRUE,
    "result" => $result,
    "message" => "Az aktuális szerveridő: " . $result["servertimestamp"],
];
```

A PHP régebbi verzióiban a „prepared statements” utasítás bezárásra került a lekérdezések után, amely megfelelően működött, több lekérdezés összefűzése (lekérdezés a lekérdezésben) esetében is.

```
$stmt->close();
```

A fejlesztés közben a PHP 7-es verziójáról áttértünk a PHP 8.1-es verzióra, amely az összefűzött lekérdezések esetében a bezárásra 500-as szerveroldali hibát adott válasznak bizonyos lekérdezések, így hosszabb kutatómunka után a „statements” bezárása helyett a visszaállítás (reset), mellett döntöttem, attól függetlenül, hogy a lekérdezés legvégén az adatbázissal való kapcsolat manuális megszakítására kerül sor.

```
$stmt->reset();
```

Az utolsó előtti lépés az adatbázissal való kapcsolat manuális bontása volt, hiszen hosszabb kísérletezések után megállapítottam, hogy a manuális megszakítás kevésbé erőforrásigényes, mint a lekérdezések utáni kapcsolattartás. Erre a célra a „MainDatabase” osztály „Disconnect” metódusa kerül meghívásra.

```
$db->Disconnect();
```

A lekérdezés ezennel lezárársa került, viszont az API még semmi választ nem adott, ezért utolsó lépésként a „die” metódussal lezárársa kerül a szerveroldali kód futtatása. Paraméterként a „json_encode” függvény kerül átadásra, amelynek első paramétere tartalmazza a JSON formátumba átalakítani kívánt objektumot, amely megjelenítésre kerül, illetve második paraméterként a „JSON_UNESCAPED_UNICODE” kerül átadásra, amely kódolás megakadályozza, hogy a magyar ábécében található ékezetes betűk, illetve speciális karakterek nem a megfelelő formátumban jelenjenek meg. Például: á betű helyett kérdőjel.

```
die(json_encode($msg, JSON_UNESCAPED_UNICODE));
```

A lekérdezés eredménye a következő JSON formátumban:

```
{
    "success": true,
    "result": {
        "serverdate": "2023-04-18",
```

```
        "servertime": "17:09:33",
        "servertimestamp": "2023-04-18 17:09:33"
    },
    "message": "Az aktuális szerveridő: 2023-04-18 17:09:33"
}
```

3.1.3 Adatvédelmi – jogi nyilatkozat

Az alkalmazásokba történő belépéskor egy lerövidített adatvédelmi nyilatkozat jelenik meg a felhasználók részére, ami úgyszintén API-ból kerül lekérdezésre. A noticeinfo.php fájl érdekessége, hogy nem található benne adatbázissal történő kommunikáció, a rövidített szöveg a „details” változóban van megírva, a frissítés időpontja az „mkttime” függvény segítségével, a „date” változóba kerül, amely az előzetesen bemutatott 3 kulcsérték-páros egyéni szabványban a „result” értékeként egy objektumban kerül megadásra a címmel, vagyis „title” változóval, az adatvédelmi nyilatkozat szövegével, a lekérdezés időpontjával, illetve a nyilatkozat időpontjával együtt. Az egyéni szabványunkban szereplő „message” értéke a következő: „Üzenet a fejlesztőktől”. A szerveridő mintájában már megismert „die” metódussal zárjuk a php fájlt, és így már az alkalmazásokban a JSON formátumból könnyen kiolvasható az aktuális nyilatkozat. Az API-ban való eltárolás mellett azért döntöttünk, mert így az alkalmazások frissítése nélkül, elengedő ebben a fájlban módosítani a szöveget.

3.1.4 Jogosultság ellenőrzése felhasználónév és jelszó párossal

A szerveridőnél bemutatott modell felépítését használjuk, kibővítve a POST értékek vizsgálatával, rendszernaplózással, több lekérdezéssel. A fejlécben beállításra kerültek az alap értékek, ezek után a POST kulcsérték-párjainak ellenőrzése következik. Először ellenőrzésre kerülnek a kötelezően megadandó kulcsok, amelyek nélkül a lekérdezés biztosan sikertelen lenne. Az „isset” segítségével vizsgálhatjuk meg, hogy létezik-e a POST kulcsok között a keresendő érték, amely logikai választ ad vissza a számunkra. Ha létezik, akkor igaz értékkel tér vissza, ellenkező esetben pedig hamissal. Ennél a lekérdezésnél kötelező megadni a felhasználónevet, jelszót, vizsgálandó jogosultságot, vállalat azonosítóját és a platform nevét.

```
if (isset($_POST['username']) && isset($_POST['password']) &&
isset($_POST['permission']) && isset($_POST['company']) &&
isset($_POST['platform']))
```

Amennyiben, ha az összetett feltétel hamis, akkor az egyéni szabványunknak megfelelően a „success” értéke hamis, a „result” értéke „Field Error”, a „message” értéke pedig „Egy vagy több kötelező paraméter megadása kötelező!” üzenet kerül válaszadásra

JSON formátumban, amely a „die” metódussal lezárásra is kerül, tehát a szerveroldali kód nem fut tovább.

Az összetett feltétel teljesülése esetén újabb ellenőrzés következik, amelyben az előzetesen említett kulcsok értékeinek vizsgálata következik. mindenépp az kerül megvizsgálásra, hogy ne kerüljön üres érték elküldésre, vagyis az érték hossza nem lehet egyenlő nullával.

Kettő módszerrel vizsgáljuk az összes API fájlokban a kötelező kulcsérték-pároknál az úgynevezett „Zero Error” hibát, amelynek logikája megegyezik, a szintaktikája kissé eltér:

- 1) Ha valamelyik érték hossza nem nulla ellentéte.

```
if (!(strlen($_POST['username']) == 0 || strlen($_POST['password']) == 0 ||  
      strlen($_POST['permission']) == 0 || strlen($_POST['company']) == 0 ||  
      strlen($_POST['platform']) == 0))
```

- 2) Ha valamelyik érték nem nulla.

```
if ((strlen($_POST['username']) != 0 && strlen($_POST['password']) != 0 &&  
    strlen($_POST['permission']) != 0 && strlen($_POST['company']) != 0 &&  
    strlen($_POST['platform']) != 0 ))
```

Ha valamelyik értéke nulla, akkor az egyéni szabványunknak megfelelően hibaüzenetet írunk ki, a „Zero Error” eredménnyel.

Sikeress ellenőrzés esetében eltárolásra kerülhetnek külön változókban az értékek, egy olyan tisztító eljárás után, amelyek védelmi célokat szolgálnak. Az úgynevezett tisztító eljárás során levágásra kerülnek a „trim” segítségével az érték elejéről és végéről feleslegesen bevitt szóközök, illetve a „strip_tags” segítségével eltávolításra kerülnek a HTML, XML és PHP tag-ek.

```
$username = strip_tags(trim($_POST['username']));
```

Léteznek olyan értékek, amelyek azonnali feldolgozás alá kerülnek a vizsgálatot követően. Ilyenek lehetnek a jelszavak, amelyek azonnali titkosítására kerül sor. A jelszavak az API-hoz érkezve már átesnek egyfajta kriptográfiai műveleten, amely titkosítja a jelszót, de a védelem érdekében az API SHA256 típusú titkosítást is elvégez a már letitkosított jelszón.

```
$password = hash('sha256',strip_tags(trim($_POST['password'])) .  
'AXfGrSSEkW7cRP9D');
```

A POST értékek ezennel eltárolásra kerültek a rendszerben, ha az alap követelmények teljesítésre kerültek. A fájl további része folytatódik a szerveridő fájlban bemutatott modell szerint, ugyanis „importálásra” kerül a szükséges csatlakozási fájl, illetve a rendszernaplóhoz hozzárendelő fájl, amely további osztályokat tartalmaz.

Ezen túl megvalósul az adatbázishoz való csatlakozás, amelynél a csatlakozás vizsgálatát követően lefut a „clear_sessions” metódus, amelynek a paramétere a „MainDatabase” osztályból példányosított objektum. Itt elsősorban azoknak a munkameneteknek az érvénytelenítése történik meg, amelyek nem kerültek meghosszabbításra, és a lejárati idő kisebb, mint az aktuális szerveridő.

```
$stmt = $db->conn->prepare("UPDATE `sessions` SET `active` = '0' WHERE `sessions`.`expiry` <= CURRENT_TIMESTAMP();");
$stmt->execute();
$stmt->reset();
```

A következő metódus a „permission_checker”, amely ellenőrzi, hogy a beírt jogosultság, amire szeretnénk vizsgálatot indítani, egyáltalán létezik-e az adatbázisban. Itt egy speciális adatbázis-lekérdezés kerül végrehajtásra, ahol is kilistázásra kerül annak az egyed neve, amelyre szűrni szeretnénk az „fms_main” adatbázisban.

```
$msg = permission_checker($db, $permission);
```

Előkészítésre kerül az SQL utasítás, majd a „bind_param” segítségével egy paraméter kerül hozzáadása szövegként. A változó paramétereket kérdőjelekkel jelöljük minden esetben az előkészített utasításban. Ezek elvégzése után az utasítás végrehajtásra kerül, majd az eredmények első sorának lekérdezése történik meg a „result” változóban.

```
$stmt = $db->conn->prepare("SELECT DISTINCT TABLE_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE COLUMN_NAME IN (?) AND TABLE_SCHEMA='fms_main';");
$stmt->bind_param("s", $permission);
$stmt->execute();
$result = $stmt->get_result()->fetch_assoc();
```

Amennyiben a lekérdezett „TABLE_NAME” értéke egyenlő nullával, az egyedi szabványunkban belső hibára felhívó üzenetet jelenítünk meg üzenetként, ellenkező esetben a „success” értéke igaz. Az előkészített utasítást visszaállítjuk a „reset” metódus segítségével, majd az objektumot visszaadjuk a metódust meghívó értékeként.

```
$stmt->reset();
return $msg;
```

Ezek után a visszatért objektumban a „success” igaz értékének esetében egy újabb feltétel kerül vizsgálatra. Ha a „company” POST értéke egyenlő a „null” nagybetűsre alakított értékével, akkor olyan utasítás kerül előkészítésre, amelyben szűrésre kerül a szerepkörökönél lévő vállalat értéke.

```
if (strtoupper($company) == "NULL")
```

Bizonyos jogosultságok benne lehetnek olyan szerepkörökben is, amelyek nem köthetőek vállalathoz, például a bejelentkezésnél még nincs lehetőségünk vállalatot választani, így a beépített FMS felhasználói szerepkör segítségével jelentkeztetjük be a

felhasználót az alkalmazásba, ha tartalmazza a „fms_framework_login” jogosultságot. Az előkészített utasításban csak értéknek lehet paramétert hozzáadni a „bind_param” segítségével, így a „permission” változó az utasításban kerül elhelyezésre. Ez így „SQL injection” veszélyének lenne kitéve, viszont ezért ellenőrizzük a jogosultság meglétét a „permission_checker” metódussal, így itt már kizárolag csak olyan lekérdezés futhat le, amely jog ténylegesen létezik a rendszerben. Amennyiben van megadva vállalat, az is hozzáadásra kerül szövegként a paraméterek között, viszont ha nem került megadásra, akkor az SQL utasításban „`roles`.`company_id` is NULL” helyettesíti az értéket.

```
$stmt = $db->conn->prepare("SELECT `accesses`.`user_id` AS 'user_id',
`accesses`.`allowed` AS 'allowedUserAccess', `roles`.`allowed` AS
'allowedRoleAccess', `roles`.`name` AS 'roleName' FROM `accesses` INNER JOIN
`roles` ON `roles`.`id` = `accesses`.`role_id` WHERE `user_id` = (SELECT `users`.`id`
FROM `users` WHERE `users`.`email` = ? AND `users`.`password` = ? LIMIT 1) AND
`roles`.`company_id` is NULL AND `roles`.`".$permission."` = TRUE ;");
$stmt->bind_param("ss", $username, $password);
```

A feltétlen kívül az utasítás mindenkorábban végrehajtásra és ellenőrzésre kerül, hogy van-e jogosultsága az adott felhasználónak. Hiányzó jogosultság esetén az egyéni szabványunknak megfelelő „Hibás felhasználónév vagy jelszó!” üzenettel ellátott eredmény kerül JSON formátumban kiírásra. Ha található a rendszerben fiók, aikhez hozzá van rendelve a vizsgálandó jogosultság, akkor további lekérdezésekre van szükség. Ekkor meghívásra kerül a „validate_user” függvény, amelyben átadásra kerül az azonosított felhasználó adatai, az adatbázis változója, a megadott platform típusa, illetve a vizsgált jogosultság.

A felhasználó ellenőrzésben egy ciklus került elhelyezésre, amely a felhasználónál talált hozzáféréseken megy végig, vagy találat esetén kerül megszakításra. Először ellenőrzésre kerül, hogy maga a szerepkör engedélyezve van-e, illetve a szerepkörhöz való hozzáférés aktív-e, amennyiben mindenkorábban igaz, a felhasználó sikeresen azonosításra került. Ezek után ellenőrzésre kerül, hogy a felhasználónak van-e a platformtól függően aktív munkamenete. Az ellenőrzést az „exist_session_checker” függvény végzi el, amelynek paramétere a felhasználó egyedi FMS azonosítója, az adatbázis kapcsolata, illetve a platform típusa. A visszatérési érték abban az esetben lesz a token, ha már létezik a platformhoz létrehozott érvényes munkamenete, ellenkező esetben „NULL” értékkel tér vissza.

```
$stmt = $db->conn->prepare("SELECT `user_id`, `active`, `token`, `platform` FROM
`sessions` WHERE `user_id` = ? AND `active` = 1 AND `platform` = ? ORDER BY
`created` DESC LIMIT 1;");
$stmt->bind_param('ss', $id, $platform);
```

```

$stmt->execute();
$result = $stmt->get_result()->fetch_assoc();
$stmt->reset();
if ($result == NULL) { return NULL; }
else { return $result['token']; }

```

Amennyiben nem található a megadott feltételeknek megfelelő token, akkor egy új generálására lesz szükség. Ilyenkor ellenőrzésre kerül, hogy a „webinfos” nevű POST kulcs létezik-e, mert akkor a „create_session” metódus utolsó értéke is átadásra kerül, ellenkező esetben üres lesz.

Új munkamenet generálásánál először ellenőrzésre kerül, hogy az IPv4 vagy IPv6 cím hozzáadásra került a POST kulcsok között.

```

$ipv4 = (empty($_POST['ipv4'])) ? NULL : strip_tags(trim($_POST['ipv4']));
$ipv6 = (empty($_POST['ipv6'])) ? NULL : strip_tags(trim($_POST['ipv6']));

```

Ezek után létrehozásra kerül 20 darab random byte, amely átkonvertálásra kerül egy szöveges karakterláncra, így létrejön a token kulcs, amely felvitelre kerül a munkamenetekhez.

```

getBytes = random_bytes(20);
$token = bin2hex($bytes);

```

Ezek után az adatbázisban kerül letárolásra az újonnan generált kulccsal a munkamenet. Többágú szelekcióban történt meg a „platform” változó vizsgálata. A dokumentációban már korábban említésre került, hogy a weboldalnál 1 óra, mobilnál 1 hét, asztali alkalmazásnál 1 nap, egyéb esetben 10 perc a munkamenet lejáratának határideje, így ezek alapján kerültek előkészítésre az utasítások. Majd a végrehajtást követően visszaadásra kerül az újonnan generált token a metódust meghívó értéknek.

Példa a weboldalról történő platform bejelentkezésekor létrejövő SQL utasításról:

```

switch ($platform) {
    case 'Website':
        $stmt = $db->conn->prepare("INSERT INTO `sessions` (`user_id`, `token`,
`platform`, `infos`, `expiry`,
`ipv4`, `ipv6`) VALUES (?, ?, ?, ?, (DATE_ADD(CURRENT_TIMESTAMP(),
INTERVAL 1 HOUR)), ?, ?)");
        break;
}
$stmt->bind_param('ssssss', $id, $token, $platform, $infos, $ipv4, $ipv6);
$stmt->execute();
$stmt->reset();

```

A létrejövő munkamenetről kerül egy bejegyzés a rendszernaplóba.

```

$log = new Log($token, "Jogosultság ellenőrzése új token létrehozásával");
$log->Execute();

```

Amennyiben már van létező aktív tokenje a platformhoz a felhasználónak, akkor a meglévő frissítésre kerül ugyanazon elv alapján, mint ahogy az új került létrehozásra. A különbség az SQL utasításban rejlik, illetve itt nem történik új token generálása.

```
switch ($platform) {
    case 'Website':
        $stmt = $db->conn->prepare("UPDATE `sessions` SET `expiry` =
(DATE_ADD(CURRENT_TIMESTAMP(), INTERVAL 1 HOUR)), `infos` = ?, `ipv4` =
?, `ipv6` = ? WHERE `token` = ?;");
        break;
}
$stmt->bind_param('ssss', $infos, $ipv4, $ipv6, $token);
$stmt->execute();
$stmt->reset();
```

Ezek után itt is megtörténik a rendszernaplóba való bejegyzés, illetve az adatbázisról lecsatlakozás és a JSON formátumban kiírt adatokkal a „die” metódussal zárul a fájl.

3.1.5 Token ellenőrző rendszer

A felhasználóhoz társított token érvényességét és meglétének ellenőrzését teszi lehetővé a következő rendszer. Az előbb említett fejléc beállításával kezdődik a „tokencheck.php” fájl. Egy darab POST érték kerül vizsgálatra, amely tartalmazza az ellenőrizni kívánt tokent. Amennyiben nem található hiba, akkor „importálásra” kerül a rendszernapló kezelését és az adatbázis csatlakozását tartalmazó osztályt. Az adatbázishoz való csatlakozás ugyanúgy működik, mint az előző lekérdezéseknel, a példányosítás után valósul meg. Amennyiben a csatlakozás megfelelő, akkor lefut az a kérdés, ahol a rendszer megnézi, hogy az aktív munkamenetek között megtalálható-e a vizsgálandó adat. Az utasítás előkészítésre kerül, majd az egy hiányzó paraméter hozzáadásra kerül.

```
$stmt = $db->conn->prepare("SELECT `token` FROM `sessions` WHERE `token` = ?
AND `active` = 1;");
$stmt->bind_param("s", $token);
```

Az utasítás végrehajtásra kerül, majd a lekérdezés eredményének első sora eltárolásra kerül a „result” változóban.

```
$result = $stmt->get_result()->fetch_assoc();
```

Az utasítás a paraméterekkel együtt visszaállításra kerül, majd ellenőrizzük, hogy a „result” változó értéke egyenlő-e nullával. Ha nulla, akkor az egyéni szabványunknak megfelelő üzenet értéke „A token nem létezik vagy lejárt!” lesz, ellenkező esetben az üzenet „A token aktív!”, és a „result” értéke egyenlő lesz az ellenőrzött tokennel.

```
if ($result == null) { $msg = ["success" => FALSE, "result" => "Token not found!",  
"message" => "A token nem létezik vagy lejárt!"];}  
else {$msg = ["success" => TRUE, "result" => $token, "message" => "A token  
aktív!"]};
```

Ahogy az összes lekérdezsnél, itt is lezárásra kerül az adatbázis csatlakozása, és az „msg” változó kiírásra kerül. Néhány lekérdezsnél, amennyiben nem sikerül az adatbázis csatlakozás, akkor egy hibaüzenet jelenik meg a felhasználó számára, amelynek a szövege belső hibát tartalmaz.

```
$msg = ["success" => FALSE, "result" => "API Connection Error!", "message" =>  
"Belső hiba!"];
```

3.1.6 Összes ország-adat megjelenítése

A publikusan elérhető országok listájának forrását a hozzá tartozó adatokkal CSV állományban sikerült megszerezni, így létrehozásra került az adatbázisban egy olyan tábla, amelyek ezekkel az adatokkal vannak feltöltve. A „countries.php” fájlban beállításra került a fejléc, importálásra került az adatbázis osztályt tartalmazó fájl, és a csatlakozást követően az előkészített utasítások végrehajtása után a rekordokat egy „while” ciklus segítségével olvassuk ki az eredményekből, majd adjuk hozzá a „result” nevű tömbhöz.

```
$stmt->execute();  
$requestResult = $stmt->get_result();  
$result = [];  
while ($row = $requestResult->fetch_assoc()) {  
    array_push($result, $row);  
}
```

Ezek után, ha az egyedi szabványunkban meghatározott üzenetben írjuk ki a felhasználó részére a „result” tömb elemeit.

```
$msg = [ "success" => TRUE, "result" => $result, "message" => "Országok információi  
kapcsolt adatokhoz."];
```

Amennyiben nem sikerült az adatbázishoz csatlakozni, az előző pont végén bemutatott hasonló üzenet jelenik meg eredményként.

3.1.7 Verziószámok

A „versions.php” állományban tároljuk el az alkalmazások verziószámait, letöltések elérési helyeit, frissítések nélküli kiadott változások adatait. Először beállításra kerül a fejléc, azaz a „header” utasításai. Ezek után az „msg” változóban új objektumok létrehozásával kerül fel egy új frissítés. Jelen esetben az API, mobil alkalmazás, asztali alkalmazás verziószámát, Android, IOS, asztali alkalmazás elérési hivatkozását találjuk. Ezen kívül a mobil alkalmazásban az utolsó változásokról olvashatóak információk,

illetve az asztali alkalmazásban található változásnapló is itt kapott helyet. A változó a beállítása után JSON formátumban kiíratásra kerül, amely feldolgozható az alkalmazásokban a GET lekérdezés végrehajtása követően.

3.1.8 QR kód generátor

A QR kód generálása a „Composer” segítségével került megoldásra. A „gen.php” első lépése az összetevők listájának betöltését tartalmazó php fájl importálása.

```
require_once($_SERVER['DOCUMENT_ROOT']."/qr/vendor/autoload.php");
```

Ezek után megvizsgálásra kerül a GET kulcsérték-párok közül a „value”, amelyben megadható a QR kód értéke. Amennyiben létezik, akkor a „value” változó értéke lesz a megtisztított GET értéke, amely a „htmlspecialchars”, „strip_tags” és a „trim” segítségével történik. Ha nem létezik a „value” kulcsérték-pár, akkor a felx.hu-ra fog mutatni a generált QR kód.

```
if ( isset($_GET["value"]) ) {  
    $value = htmlspecialchars(strip_tags(trim($_GET["value"])));  
} else { $value = "felx.hu"; }
```

Ezek után az Endroid-féle dokumentációban található értékek segítségével megoldásra került a generálás, a szükséges adatok beállítása után. A sorozat végén található „build” generálja le a „result” változóba a szükséges kódot.

```
$result = Builder::create()  
    ->writer(new PngWriter())  
    ->writerOptions([])  
    ->data($value)  
    ->encoding(new Encoding('UTF-8'))  
    ->errorCorrectionLevel(new ErrorCorrectionLevelHigh())  
    ->size(600)  
    ->margin(10)  
    ->roundBlockSizeMode(new RoundBlockSizeModeMargin())  
    ->labelText('FelX Management System')  
    ->labelFont(new NotoSans(20))  
    ->labelAlignment(new LabelAlignmentCenter())  
    ->validateResult(false)  
    ->build();
```

A fejlécben beállításra kerül ezután a generált QR kód MIME típusa, majd megjelenítésre kerül az oldalon.

```
header('Content-Type: '.$result->getMimeType());  
echo $result->getString();
```

3.1.9 SSO hitelesítés

A rendszerhez köthető adatok lekérdezése, hozzáadása, módosítása és törlésnél a szerepkörök, azok hozzáférhetősége, és a munkamenetek érvényessége szabályozza a

hozzáférést. A hitelesítés közben csak egyszer történik felhasználónév és jelszó párossal azonosítás, amely megfelelő autentikáció után visszaad egy olyan generált kód sorozatot (token), amely a későbbi lekérdezések folyamán hitelesíti a felhasználót. A többi lekérdezésben ezek szerint történik a hitelesítés, ezért is hívjuk SSO, azaz Single Sign-On módszernek, mivel a belépés után, minden szolgáltatás elérhető az adott munkamenetben.

3.1.10 Jogosultság ellenőrzése token alapján

A korábbiakban már említettem a felhasználónév és jelszóval történő bejelentkezés részletes működését, amely nagy százalékban megegyezik a token alapján történő jogosultság ellenőrzésével. Szinte minden lekérdezés előtt szükséges a token érvényességének, és a jogosultsághoz való hozzáférésnek az ellenőrzése.

Létrehozásra került egy „permcheckobject.php” nevű fájl, amely több olyan függvényt tartalmaz, amely a jogosultság-ellenőrzések során szükségesek. Bárhol importálásra kerülhet, és az „SSOPermCheckTool” függvényt a megfelelő paraméterekkel meghívva egyszerűen ellenőrizhető token alapján a jogosultság.

```
require_once($_SERVER['DOCUMENT_ROOT']."/sso/permcheckobject.php");
```

A függvénynek a paraméterei között tartalmaznia kell a tokent, a jogosultság nevét, illetve a vállalat azonosítóját. A visszatérési érték minden esetben JSON.

```
$permcheck = SSOPermCheckTool($token, "fms_framework_company_manager_full", $company);
```

Az összes lekérdezésnél ugyanezt a modellt alkalmaztuk, ha a visszatérési érték hamis, vagy a „success” hamis, akkor a visszaadott JSON megjelenítésre kerül, és a „die” metódus zárja a szerveroldali kód futását, így a további adatok lekérdezése, végrehajtása nem kerül feldolgozásra. A „permcheckobject.php”-ban létrehozott ellenőrző függvény alapértelmezett működése a „permcheck.php”-ban került tesztelésre.

Beállításra kerültek a kötelezően megadandó fejlécadatok. Ezek után a POST kulcsérték-párok ellenőrzése következik, jelen esetben kötelező megadni a tokent, a jogosultságot és a vállalat azonosítóját, amely ha megfelel a követelményeknek, akkor importálásra kerül a fentebb említett állomány, majd meghívásra kerül a függvény, amelynek visszatérési értéke az „msg” változóban kerül eltárolásra, majd a JSON üzenettel zárul a lekérdezés.

```
require_once($_SERVER['DOCUMENT_ROOT']."/sso/permcheckobject.php");
$msg = SSOPermCheckTool($token, $permission, $company);
die(json_encode($msg, JSON_UNESCAPED_UNICODE));
```

3.1.11 Vállalati adatkezelés

A vállalatokkal kapcsolatos lekérdezések állományainak helye az „sso”-n belül a „company” mappában található meg. Mindegyik lekérdezés alapjai megegyeznek a korábban megemlített mintákkal, így csak az eltérésekre, illetve a funkciókra térnék ki. Kötelező megadni POST módszerrel történő lekérdezésnél a tokent, és a kiválasztott vállalat azonosítóját.

- 1) Kiválasztott vállalat adatainak lekérését az „info.php” állomány kezeli, amelyben a vállalathoz tartozó szerepkörök közül az „fms_framework_login” jogosultság kerül vizsgálatra az SSO ellenőrzés kapcsán. A feltételek megfelelése esetén, ha a megadott vállalat azonosítója alapján a lekérdezett sorok száma nem egyenlő nullával, akkor a „SuccessfulAction” függvényt meghívva ellenőrzi, hogy a vállalatnak van-e beállított logója. Amennyiben a logó értéke nulla, a saját FELX tárhelyről kerül egy logó beállításra.

```
if ($result[$i]["logo"] == NULL)
{ $result[$i]["logo"] = "https://cdn.felx.hu/assets/img/logo_x.png"; }
```

Ha nem található egy rekord sem a lekérdezésben, akkor válaszként visszaadásra kerül, hogy a vállalat nem létezik. A hibaüzenet megjelenése elég valószínűtlen, mivel ha a vállalat táblában nem létezik a keresett azonosítóval hozzá kapcsolódó érték, akkor már a jogosultság ellenőrző vizsgálaton „megbukik” a lekérdezés. Ettől függetlenül hozzáadásra került az üzenet, dupla védelmet szolgáló célként.

- 2) Aktív vállalati szerepkörök lekérdezését az „activeroles.php” állomány kezeli, amelyben a „fms_framework_company_manager_full” szerepkör kerül vizsgálatra, megegyező vizsgálati módszerrel, amelyet az „info.php”-ban is használtam. Amennyiben a jogosultságok megfelelőek, a kért vállalathoz tartozó szerepkörök kerülnek kilistázásra. Ha nem található egy szerepkör sem, akkor az egyedi szabványunkban az üzenet értéke egyenlő lesz a „Nincs aktív szerepkör hozzárendelve a vállalathoz!” kifejezéssel.
- 3) Vállalati adatok, szerepkörök, és a szerepkörökhöz rendelt felhasználók lekérdezését az „admininfo.php” fájlban található utasításhalmaz kezel, amelyben elsősorban ugyanarra a szerepkörre kerül a jogosultság ellenőrzése, amely az előző pontban is felhasználásra került. Itt a JSON-ben kiadott válasz 3 darab lekérdezést tartalmaz. Adatbázishoz való csatlakozás után létrehozásra kerül egy üres „result” névvel ellátott tömb. Ezek után hozzáadásra kerül a vállalati adatok lekérdezése, amely a „CompanyRequest” függvény meghívásával és visszatérési értékének

hozzáadásával történik, ezek után az „AccessRequest” és a „RolesRequest” függvény is lefutásra kerül, amely után egy feltételvizsgálat dönti el a megjelenített válasz eredményét.

```
$result = [];
$result["company"] = CompanyRequest($db, $company);
$result["accesses"] = AccessRequest($db, $company);
$result["roles"] = RolesRequest($db, $company);
```

A vizsgálatban a „count” használatával ellenőrizzük a tömb elemeinek számát. Ha a tömb elemeinek száma nulla, akkor hibaüzenet kerül megjelenítésre, ellenkező esetben a válaszban a „result” értéke egyenlő lesz a „result” tömbben eltárolt adatokkal.

```
$msg = [ "success" => TRUE, "result" => $result, "message" => "Kiválasztott vállalatinformációk lekérdezése sikeres!"];
```

- 4) Vállalati felhasználók, szerepkörök csoportosítása alapján történő lekérdezése a „users.php”-ban található meg. Itt is szükséges a teljes vállalatkezelési jogosultság, amelynek a sikeres ellenőrzését követően kiírásra kerül a vállalat azonosítója és neve, hozzáférés-azonosítója és állapota, a szerepkör neve és állapota, illetve a felhasználó azonosítója és neve. Itt fontos megemlítenem, hogy annyiszor kerül kilistázásra a felhasználó, ahány vállalati szerepkör van hozzárendelve a felhasználóhoz, így előfordulhat, hogy 1 alkalomnál többször találkozhatunk ugyanazzal a felhasználóval, ez a szerepköri hozzáférések elsődleges kilistázása miatt került így kialakításra.
- 5) Vállalati felhasználók lekérdezése az „activeusers.php” fájlban található meg, ahol az SQL utasításban található „DISTINCT” utasítás biztosítja, hogy a felhasználók csak egyszer jelenjenek meg, attól függetlenül, hogy több szerepkör is hozzá van-e rendelve a fiókjukhoz. Itt fontos megjegyezni, hogy már csak az aktív hozzáféréssel rendelkező felhasználók kerülnek listázásra, tehát a szerepkörök aktívnak (engedélyezettnek) és a hozzáférésnek is aktívnak kell lennie ahoz, hogy a felhasználó megjelenjen a listán. Ha nincs egy felhasználó sem, aki megfelelne a feltételnek, akkor hibaüzenet nyújt tájékoztatást az eredményről.
- 6) Vállalati felhasználók teljes adatainak lekérdezése a „usersdata.php” állományban található, az előzőtől annyi eltéréssel, hogy itt megjelenhetnek azok a felhasználók is, akik nem rendelkeznek aktív szerepköri hozzáféréssel, illetve a felhasználók összes személyes adata is megjelenítésre kerül.
- 7) Felhasználóhoz tartozó szerepkörök lekérdezését a „useraccesses.php” fájl biztosítja. Itt az alap POST kulcsérték-párok megadása mellett szükség van a

felhasználó azonosítójának megadásra is. A megfelelő jogosultságok ellenőrzése után megjelenítésre kerül a megadott felhasználói azonosítóhoz tartozó szerepköri hozzáférések lekérdezése.

- 8) Vállalatokhoz tartozó munkacsoportok lekérdezését a „group.php” teszi lehetővé, amelynek eredményeként megjelenítésre kerül a „result” értékeként a vállalati munkacsoportok azonosítója, neve, létrehozásának és utolsó módosításának ideje, illetve a vállalat azonosítója és neve. Ha nem található egyetlen munkacsoport sem a vállalatnál, akkor a hibaüzenet, „A vállalathoz nem tartozik egyetlen munkacsoport sem!” kerül kiíratásra.
- 9) Vállalati munkacsoporthoz tartozó felhasználók a „taskusers.php” fájl segítségével kérdezhető le az adatbázisból, ahol az alap kulcsérték-párok megadása mellett a munkacsoport azonosítójára is szükség van. A vállalat adminisztrálását tartalmazó jogosultság ellenőrzése után megjelenítésre kerül a munkacsoporthoz való hozzárendelés azonosítója, létrehozási és utolsó módosítási ideje, illetve a felhasználó azonosítója és neve is megtalálható a lekérdezésben. A JSON formátumban történő sikeres megjelenítés a következő:

```
{  
    "success": true,  
    "result": [  
        { "assign_id": 3, "assign_create": "2023-03-16 09:10:59", "assign_modified":  
null, "user_id": 1000000000, "user_name": "Varsányi Barnabás"},  
        { "assign_id": 4, "assign_create": "2023-03-16 09:11:27", "assign_modified":  
null, "user_id": 1000000001, "user_name": "Medve Adrián"},  
        { "assign_id": 5, "assign_create": "2023-03-16 09:11:33", "assign_modified":  
null, "user_id": 1000000002, "user_name": "Lázár Marcell"}  
    ],  
    "message": "3 felhasználó tartozik a(z) 5. számú vállalat 5. számú  
munkacsoporthoz!"  
}
```

- 10) Új vállalati munkacsoport létrehozása az „addgroup.php” állomány segítségével használható, amely eltér az eddigi lekérdezésektől. Itt nem adatok megjelenítése történik, hanem egy új rekord létrehozása az adatbázisban. A POST kulcs-érték pároknál az alapokon kívül szükséges megadni a munkacsoport nevét. A „Field” és a „Zero” ellenőrzési módszerek mellett az „Overflow” is alkalmazásra kerül, amelyben a változó értékének a hossza kerül megvizsgálásra. Ha 100 karakternél hosszabb a munkacsoport neve, akkor az „Overflow Error” kerül a „result” értéknél megjelenítésre, majd a szerveroldali kód futtatásának leállításával nem futnak le az adatbázisban történő műveletek.

```
$msg = [ "success" => FALSE, "result" => "Overflow Error", "message" => "Egy vagy több limitált hosszú paraméter nagyobb mint a megendededett érték!"];  
die(json_encode($msg, JSON_UNESCAPED_UNICODE));
```

Az ellenőrzések és adatbázishoz való csatlakozás után előkészítésre kerül az „INSERT” utasítás, amely két változtatható paramétert fog tartalmazni. A paraméterek hozzárendelése után az utasítás végrehajtásra kerül, és az érintett sorok száma alapján történik az eredmény megjelenítése az API lekérdezésben.

```
$stmt = $db->conn->prepare("INSERT INTO `taskgroups` (`name`, `company_id`)  
VALUES (?, ?)");  
$stmt->bind_param("ss", $name, $company);  
$stmt->execute();
```

Az érintett sorok számánál a -1 számérték hibás utasítást tartalmaz. Ekkor előfordulhat, hogy nem megfelelő paraméterek kerülnek hozzáadásra az utasításhoz, pl. logikai értéket szeretnénk dátum mezőhöz rendelni és hasonlók.

Amennyiben a lekérdezés 0 darab sort érint, akkor nem történt hozzáadás, de ez általában rekordok törlésénél és módosításánál használandó.

Ezeken túl pedig a számérték jelöli a módosított, törlött, hozzáadott rekordok számát.

```
if ($stmt->affected_rows == -1)  
else if ($stmt->affected_rows == 0 || $stmt->affected_rows == NULL)  
Általánosságban a lekérdezésekben, amikor valamiféle adatmanipuláció történik, amely nem lekérdezést tartalmaz, rendszernaplóba való automatikus bejegyzését alkalmaztam, már az ismertetett módszerrel.
```

```
$log = new Log($token, "SSO - Új munkacsoport");  
$log->Execute();
```

A sikeres módosítás során a „message” változóban megjelenítésre kerül a létrehozott munkacsoport neve, sikertelenség esetén a „result” tartalmaz olyan értéket, amely alapján a hiba besorolható kétféle csoportba. Az egyik az „Internal Error” hiba, amely a -1 számérték esetén történik kiírásra, amikor meg nem érint sort a lekérdezés, akkor általában az üzenet egyszerűsített angolra fordítással történik megjelenítésre.

```
$msg = [ "success" => TRUE, "result" => "Created group successfully!", "message" =>  
$name . " munkacsoport sikeresen rögzítve!"];
```

11) Felhasználó hozzárendelése munkacsoporthoz az „assigngroupuser.php” segítségével az „fms_framework_company_management_full” jogosultság ellenőrzésével történik. Az alap kulcsérték-párokon kívül itt a munkacsoport azonosítójának és a felhasználó FMS azonosítójának megadása is kötelező. Elsőként megvizsgálásra kerül, hogy a felhasználó már tagja-e a megadott

munkacsoportnak. Az utasítás végrehajtása után, ha a lekérdezett sorok száma nagyobb, mint 0, akkor már tagja a munkacsoportnak, mivel a „taskgroupusers” táblában már van ezzel a két értékkel megegyező rekord.

```
$stmt->execute();
$requestResult = $stmt->get_result();
if ($requestResult->num_rows > 0) {
```

Ellenkező esetben meghívásra kerül az „Assign” függvény, amelynek visszatérési értéke egy JSON üzenet, amely tartalmazza a hozzárendelés sikerességének eredményét. A függvényen belül hozzárendelésre kerül a „taskgroupuser” táblához a munkacsoport és a felhasználó FMS azonosítója, amely a megfelelő eredménytől függően a visszaadandó érték beállítása megtörténik. Sikeres rögzítésnél új bejegyzés történik a rendszernaplóba.

```
$msg = Assign($db, $token, $group, $user);
```

- 12) Felhasználó eltávolítása egy kiválasztott munkacsoportból a „removegroupuser.php” használatával elvégezhető. A fájl megegyezik az előző pontban említett hozzárendeléssel, annyi különbséggel, hogy itt csak egy SQL utasítás kerül végrehajtásra, amely a rekord törlését tartalmazza.
- 13) Meglévő felhasználó adatainak lekérdezése az „existuser.php” állomány segítségével végrehajtható, ahol szükség van a generált biztonsági kulcs megadására. A fájl 5 darab lekérdezést tartalmaz összefűzve, amelyben a „GetUserData” a kulcs érvényességtől függően lekéri a felhasználó adatait, majd ha az előző lekérés sikeres volt, akkor a „GetActiveRoles” függvény lekéri a felhasználó szerepköreinek hozzáférését, végül a „GetCompanyActiveRoles” pedig lekéri a vállalat szerepköreit, amelyeket hozzárendelhetjük majd a biztonsági kulccsal generált felhasználóhoz. Utolsó lépésként a „DeactivateKeys” deaktiválja a biztonsági kulcsot, mivel az csak egyszer felhasználható a rendszerben. A lekérdezett adatok hozzáadásra kerülnek a „result” tömbhöz, majd megjelenítésre kerülnek a felhasználó részére.

```
$result = [];
$userAccount = GetUserData($db, $secretkey);
if ($userAccount != NULL) {
    $result["user_data"] = $userAccount;
    $result["user_roles"] = GetActiveRoles($db, $userAccount["id"]);
    $result["company_roles"] = GetCompanyActiveRoles($db, $company);
    $result["deactivated_secretkey"] = DeactivateKeys($db, $secretkey);
}
```

Ha nem sikerül a felhasználó azonosítása a kulcs segítségével, a „GetUserData” által, akkor „A megadott biztonsági kulcs nem létezik vagy lejárt!” hibaüzenet kerül megjelenítésre.

- 14) Szerepköri hozzáférést az „assignuseraccess.php” fájl segítségével rendelhetünk hozzá a felhasználó számára. Kötelező megadni a hozzárendelni kívánt szerepkör és a felhasználó FMS azonosítóját. A hozzárendelés előtt ellenőrzi a rendszer egy lekérdezéssel, hogy létezik-e hozzárendelés a szerepkör és a felhasználó között. Amennyiben létezik, akkor „A felhasználó már korábban hozzárendelésre került a kiválasztott szerepkörhöz!” szöveggel zárul az API válaszadása, ha pedig még nem született hozzárendelés, akkor ellenőrzésre kerül, hogy a megadott szerepköri azonosító létezik-e, illetve a szerepkör ahhoz a vállalathoz van-e rendelve, amely adminisztrálása éppen folyamatban van. Hiba esetén „A hozzárendelni kívánt szerepkör nem létezik!” üzenetet jelenítjük meg a válaszban. Ellenkező esetben meghívásra kerül az „AssignUserToRole” függvény, amely hozzárendeli a felhasználót a kívánt szerepkörhöz, illetve készít egy bejegyzést a rendszernaplóba.
- 15) Szerepkörhöz való felhasználói hozzáférésének módosítása az „accesschange.php” fájlból történik, ahol kötelező megadni kulcsérték-páronként a felhasználó FMS azonosítóját, szerepköri hozzáférés azonosítóját, illetve a feladat állapotát, amely 0 esetén letiltja az ideiglenesen a hozzáférést, 1 esetén pedig engedélyezi. Ha a „status” értéke nem 0 vagy 1, akkor egy új ellenőrzési rendszer állítja meg a szerveroldali kód futtatását, amelynek válaszában a „result” értéke a „Value Error” lesz.

```
if ($status != 0 && $status != 1) {  
    $msg = [ "success" => FALSE, "result" => "Value Error", "message" => "Egy vagy  
    több paraméter értéke nem megfelelő!"];  
    die(json_encode($msg, JSON_UNESCAPED_UNICODE)); }
```

Az értékek, jogosultságok vizsgálata, és az adatbázishoz való csatlakozás után, frissítésre kerül az a rekord a kívánt értékkel, amely szerepkör és felhasználó együttes meglétének ellenőrzésével megtörténik. Ilyenkor az SQL utasítás tartalmaz egy olyan allekérdezést, amely megnézi, hogy a vállalathoz tartozik-e az adott szerepkör, amelynél a módosítás meg fog történni. Sikeres végrehajtást követően a rendszernaplóba újabb bejegyzés kerül „SSO - Felhasználói hozzáférés módosítás” megnevezéssel.

```
$stmt = $db->conn->prepare("UPDATE `accesses` SET `allowed` = ? WHERE `id` = ?  
AND `user_id` = ? AND `role_id` IN (SELECT `id` FROM `roles` WHERE  
`company_id` = ?);");
```

```
$stmt->bind_param("ssss", $status, $access, $user, $company);
$stmt->execute();
```

- 16) Szerepkörhöz való felhasználói hozzáférés törlése a „removeuseraccess.php” állomány használatával lehetséges. Az előző lekérdezéssel megegyezően a „status” POST kulcsérték-párt kihagyva, a többi érték és a jogosultság ellenőrzésével az adatbázishoz való kapcsolódás után törlésre kerül a rendszerből. Sikeres törlés esetén a „SSO - Felhasználói hozzáférés törlése” címmel kerül bejegyzés a rendszernaplóba.
- 17) Új felhasználó regisztrálása az FMS rendszerébe az eddigiek től eltérve a „user” könyvtáron belül a „new.php” fájl meghívásával történik. Az „fms_framework_company_manager_full” jogosultság szükséges a használatához, illetve a POST kulcsok közül az összes megadása kötelező, viszont a tokenen, vállalat azonosítóján, teljes nevén, kapcsolattartói e-mail címén és születési nevén kívül minden más értéknek a hossza lehet 0. Az ellenőrzés és az adatbázishoz való csatlakozás után a felhasználó létrehozása történik. A regisztráció alkalmával a felhasználó egy random generált jelszót kap, amelyet a későbbiek folyamán bármikor megváltoztathat. A sikeres létrehozást követően lekérdezésre kerül az újonnan létrejött felhasználó FMS azonosítója, a „ GetUserFMSID ” meghívásával, majd az alap FMS felhasználói szerepkörhöz történő hozzárendelés kerül megvalósításra, hogy a regisztrációt követően a belépés lehetséges legyen a rendszerbe, az összes platform használatával. Sikeres létrehozás és hozzárendelés után az API a következőt adja válaszul:

```
{
  "success": true,
  "result": {
    "user_id": 1000000037,
    "password": "39f927041cab750a9b8c"
  },
  "message": "Kiss Péter felhasználó, 1000000037 azonosítóval sikeresen felvételre került az FMS rendszerébe!"
}
```

3.1.12 Feladatok kezelése

A feladatok kezelésének állományai az „sso”-n belül a „task” könyvtárban találhatóak meg. A különböző SQL utasításokat tartalmazó fájlok felépítése megegyező az előbb említettekkel. Kétféle jogosultság ellenőrzése jelenik meg a feladatoknál, amelynél az első szint az „fms_framework_task”, a második szint pedig az

„fms_framework_task_full”. Az utóbbi jogosultsággal rendelkező felhasználók több hozzáféréssel rendelkeznek, így biztosítva az adatok védelmét.

- 1) Bejelentkezett felhasználóhoz rendelt munkacsoportok lekérdezését a „group.php” állomány biztosítja. A token és a kiválasztott vállalat azonosítóját megadva, az értékek ellenőrzését követően az előbb említett első jogosultsági szint vizsgálata során az SSO jogosultság-ellenőrző rendszer „access” objektumában lévő, munkamenethez tartozó felhasználó FMS azonosítója eltárolásra kerül az SQL utasítás egyik paramétereként felhasználható „user_id” változóba.

```
$user_id = $perm["result"]["access"]["user_id"];
```

Ezek után előkészítésre kerül az SQL utasítás, majd a vállalat azonosítója, és az előbb említett FMS azonosító paraméter hozzáadásra kerül a lekérdezéshez, majd a lekérdezés végrehajtásában, a rekordok a „result” tömbben eltárolt értékeinek megvizsgálásával döntünk a kiírásban. Amennyiben a tömb elemeinek száma 0, akkor a tokenhez tartozó felhasználóhoz nem tartozik egyetlen munkacsoport sem hozzárendelve, ellenkező esetben kiírásra kerül a munkacsoport azonosítója, neve, létrehozási és utolsó módosítási ideje, a vállalat azonosítója és neve.

- 2) A feladatok listáját a „list.php” lekérdezésével lehet megtekinteni. Nem szükséges a token és a vállalat azonosítóján kívül más megadni, az első jogosultsági szint („fms_framework_task”) elegendő a felhasználóhoz tartozó munkacsoport feladatainak megtekintésére. A feladatok sikeres lekérdezésénél előre kerülnek az elvégzendő feladatok, a sorrend pedig a létrehozás ideje szerinti csökkenő sorrend, amely szerint az újabb feladatok kerülnek előre. A lekérdezés eredményét befolyásolja a „result” tömb elemeinek száma, amelyben a rekordok kerülnek eltárolásra. Ha a lekérdezett adatok száma 0, akkor a „Jelenleg nincs rögzítve egyetlen egy feladat sem!” hibaüzenet kerül megjelenítésre. Sikeresség esetén kiírásra kerül a feladat azonosítója, munkacsoport azonosítója és neve, vállalat azonosítója és neve, a feladat létrehozását elrendelő személy FMS azonosítója és neve, feladat címe, állapota (0 esetén elvégzendő, 1 esetén elvégzett), határideje, létrehozásának és utolsó módosításának ideje.
- 3) Új feladat létrehozása az „addtask.php” használatával lehetséges, amely POST kulcsérték-pároknál az alapokon kívül, kötelező megadni annak a munkacsoportnak az azonosítóját, amelynek a feladatot szeretné a felhasználó kiosztani, illetve maximum 100 karakter terjedelemben szükséges megadni a feladat címét. Ezen kívül nem kötelező megadni a feladat határidejét, amely a

„deadline” POST kulcs megadásával lehetséges. A feladat hozzáadásához a második, azaz a „task_full” jogosultság szükséges. A hitelesítés és az adatbázishoz való csatlakozása után elsősorban lefuttatásra kerül egy olyan SQL utasítás, amelyben a vállalat azonosítóját és a munkacsoport azonosítójának ellenőrzése történik meg. Ha a lekérdezés rekordjainak száma nagyobb, mint 0, akkor a „companysearch” változó értéke igaz lesz, ellenkező esetben hamis. A feladat rögzítése csak akkor történik meg, ha a munkacsoport és a vállalat összerendelése megegyezik, tehát az előbb említett logikai változó értéke igaz. A rögzítés sikeres végrehajtása esetén a „Feladat sikeresen rögzítve!” üzenet jelenik meg válaszként, illetve a rendszernaplóba az „SSO - Új feladatrögzítés” címmel kerül rögzítés. Ha a „companysearch” értéke hamis, akkor az új feladat létrehozása helyett, a következő hibaüzenet kerül JSON formátumban megjelenítésre.

```
$msg = [ "success" => FALSE, "result" => "Pairing Error", "message" => "A kért cégszöport nem a kiválasztott vállalathoz tartozik!"];
```

- 4) Kiválasztott feladat törlése a „delete.php” segítségével lehetséges, az alap kulcsérték-párok megadásán kívül a „task” kulcsnak tartalmaznia kell a feladat azonosítóját. A törléshez a második jogosultsági szint, azaz a „task_full” szükséges. Az ellenőrzések után az első lépésben meghívásra kerül a „UserAccess” függvény, amely megnézi, hogy a tokenhez tartozó felhasználó benne van-e abban a munkacsoportban, amely a feladathoz van kiadva. Ha a feltétel igaz, akkor először törlésre kerülnek a feladathoz hozzáfűzött megjegyzések, amely a „DeleteTaskInfos” meghívásával lehetséges. Ha az érintett sorok száma 0, akkor nem volt olyan üzenet, amely a feladathoz tartozott volna, így ez nem hibaként kezelendő. Ha volt törlendő üzenet, akkor a rendszernaplóba is kerül bejegyzés az üzenetek törléséről. A feladat törlése előtt azért van szükség az üzenetek törlésére, mert az adatbázisban beállított megszorítások nem engednék addig törölni a „tasks” táblából a rekordot, ameddig van hozzátartozó üzenet a „taskinfos” táblában. A függvény „success” értéke az esetben lesz hamis a lekérdezést követően, amennyiben az érintett sorok száma -1. Ha a „success” értéke igaz, akkor törlésre kerül a feladat is, amelyről szintén történik bejegyzés a naplóban. Ha az üzenetek törlése sikertelen, akkor az „msg” értéke egyenlő lesz a feladat üzeneteinek törlésénél keletkező hibaüzenettel, azaz az „dti”-vel. Ha pedig a felhasználó nem tartozik a megadott feladat munkacsoportjába, akkor pedig az „msg” változó értéke egyenlő lesz az „ua” változóval. minden esetben az „msg” kerül megjelenítésre.

```

$ua = UserAccess($db, $task, $user_id);
if ($ua["success"] == TRUE) {
    $dti = DeleteTaskInfos($db, $token, $task);
    if ($dti["success"] == TRUE) { $msg = DeleteTask($db, $token, $task); }
    else { $msg = $dti; }
} else { $msg = $ua; }

```

- 5) Kiválasztott feladat üzeneteinek megtekintése a „message.php” állomány lekérdezésével lehetséges. Az első jogosultsági szint, vagyis az „fms_framework_task” elegendő az üzenetek megtekintéséhez. Plusz értékként kötelező megadni a feladat azonosítóját, amely alapján a sikeres lekérdezésben megjelenik a munkacsoport azonosítója, feladat azonosítója és címe, a teljes üzenet, az üzenet rögzítését elvégző felhasználó FMS azonosítója és neve, a létrehozásának és utolsó módosításának idejét. Az adatbázisból való lekérésnél „alselect” használatával kerül ellenőrzésre, hogy a munkacsoport tagja-e a tokenhez tartozó bejelentkezett felhasználó. Az üzenetek időrendi sorrendben kerülnek kiírásra.
- 6) Kiválasztott feladathoz üzenet létrehozása az „addmessage.php” segítségével valósítható meg. Az alap kulcsérték-párokon kívül az üzenet tartalmának és a feladat azonosítójának a megadásával lehetséges a rögzítés. Az üzenet hossza nem lehet nagyobb, mint 1000 karakter. A feladat létrehozásának második jogosultságával ellentében elegendő csak a „fms_framework_task”. Az adatbázisban az üzenetet létrehozó felhasználó is eltárolásra kerül, így a jogosultság ellenőrzésénél a tokenhez tartozó felhasználó FMS azonosítója kerül hozzáadásra. Az üzenet sikeres hozzáadását az „Üzenet sikeresen rögzítve!” válasz nyugtázza, a rendszernaplóba „SSO - Új üzenet rögzítve, meglévő feladathoz.” címmel bejegyezve.
- 7) Kiválasztott feladat állapotának átváltása a „statuschange.php” segítségével elvégezhető. Az állapot vagy „státusz” segítségével lehet az elvégzendő feladatot átállítani elvégzetre. Az alap kulcsérték-párok megadása mellett szükség van a „status” logikai értékének megadására, amely beállítja a megfelelő értéket. 0 esetén a feladat elvégzendő lesz, 1 esetén elvégzett állapotra kerül beállításra. Egyéni üzenet rögzítése is lehetséges, a „message” megadásával. Ezen kívül kötelező megadni a feladat azonosítóját, amivel a műveletet szeretné a felhasználó elvégezni. A megjegyzés hossza itt sem lehet hosszabb, mint 1000 karakter, így ez is ellenőrzésre kerül. Ha nem kerül üzenet a POST lekérdezéshez hozzáadásra, akkor a feladathoz hozzárendelt üzenet szövege a „Státuszváltás (Elvégzett)” vagy „Státuszváltás (Elvégzendő)” lesz. A feladat elvégzetre való átállítása, alap

„fms_framework_task” jogosultság használatával történhet. Amennyiben elvégzett feladatot szeretne a felhasználó elvégzendőre állítani, akkor már szükséges a magasabb „task_full” jogosultság. Az értékek vizsgálását követően az alap ellenőrzésre kerül sor, majd a magasabb jogosultság is ellenőrzésre kerül. Az utóbbi esetében viszont nem állítja meg a szerveroldali kód futtatását. Ha a magasabb jogosultság is megtalálható, akkor az „AdminStatusChange”, ellenkező esetben a „NormalStatusChange” kerül meghívásra. Ha a státuszváltás sikeresnek bizonyul, akkor a feladathoz rögzítésre kerül a felhasználó által megadott üzenet, vagy annak helyettesítő szövege.

```
if ($permAdmin["success"] == TRUE) {
    $msg = AdminStatusChange($db, $status, $task);
}
else {
    $msg = NormalStatusChange($db, $status, $task);
}
if ($msg["success"] == TRUE) {
    $msg = PlusMessage($db, $task, $message, $user_id);
}
```

3.1.13 Felhasználói adatok és fiókkezelés

A felhasználók személyes adatait, illetve a fiókkezeléssel kapcsolatos lekérdezéseket az „sso” könyvtáron belül a „user”-ben kerülnek eltárolásra. A legtöbb felhasználói lekérdezésnél nincs szükség a vállalat azonosítójára, mivel nem vállalatspecifikusak az adott jogosultságok, viszont a beépített szerepkörnek tartalmaznia kell az „fms_framework_personal” jogot. A token értékét kötelező megadni minden lekérdezésnél, változatlanul.

- 1) Bejelentkezett felhasználó FMS azonosítójának és nevének lekérése a „minimal.php” állomány használatával lehetséges. A felhasználói jogosultság ellenőrzése után, a munkamenethez tartozó felhasználó neve és FMS azonosítója megjeleníthető. Az SQL utasításban található allekérdezés biztosítja, hogy a megadott tokenhez tartozó aktív munkamenetben lévő azonosító alapján történjen meg a lekérdezés.

```
$stmt = $db->conn->prepare("SELECT `users`.`id`, `users`.`name` FROM `users` WHERE `users`.`id` =( SELECT `user_id` AS 'logged_user_id' FROM `sessions` WHERE `sessions`.`token` = ? AND `sessions`.`active` is TRUE LIMIT 1 );");
```

- 2) Az előző lekérdezés kibővítéseként a felhasználó összes adatainak lekérdezése a „full.php” használatával lehetséges. Az állományban csak az SQL utasítás tér el, ahol a „users” táblában lévő további egyed-előfordulások tulajdonságértéke kerül

megjelenítésre, többek között a teljes név, kapcsolattartói és másodlagos e-mail cím, telefonszámok, születési hely és idő, születési név, édesanya leánykori név, nem és a továbbiak.

- 3) Aktív munkamenetek lekérdezése a „devices.php”-ban megadott érvényes token megadása alapján történik. Az SQL utasításban a felhasználóhoz tartozó aktív munkamenetek kerülnek lekérdezésre, amelyben a felhasználó FMS azonosítója és neve, platform neve és információi, utolsó frissítés dátuma, IPv4 és IPv6 címek lekérdezése kerül megjelenítésre. Az IP címekhez tartozó IP információk megjelenítése egy külső API segítségével kerül integrálásra a FELX API-ra. Az IP cím lekérése a „file_get_contents” függvény használatával lekérve, a JSON válasz hozzáadásra kerül a munkamenetet tartalmazó objektumhoz.

```
function ipgeoData($ip) {  
    $api = file_get_contents("https://ipinfo.io/".$ip."/json?token=XXXXXXXXXX");  
    if ($api == FALSE) { $api = ["result" => "No IP informations"];}  
    return $api;  
}
```

A külső API használatához az „IPInfo” szolgáltatást használva, az ingyenes lekérdezést tartalmazó token kerül paraméterként megadásra a GET lekérésben. A kérés visszaadja válaszként, az IP címhez tartozó „hostnevet”, ország kódját, megye és város nevét, irányítószámát, körülbelüli koordinátákat, szolgáltató AS számát és bejegyzett nevét, illetve az időzönét. A válasz hozzáadásra kerül a „result”, „ipinfo”, és „ipinfo_ipv6” értékekhez.

```
$ipgeo = ipgeoData($row["ipv4"]);  
if ($ipgeo != FALSE) { $row["ipinfo"] = json_decode($ipgeo,  
JSON_UNESCAPED_UNICODE); }
```

Ha a munkamenetek száma egyenlő 0-val, akkor a „Lejárt a felhasználó munkamenete! Kérjük, jelentkezzen be újra!” hibaüzenet kerül megjelenítésre.

- 4) Munkamenet kényszerített kijelentkeztetése (deaktiválása) a „stopsession.php” meghívásával történik. A POST kéréshez szükséges megadni a munkamenet azonosítóját, amelyet az előző lekérdezés ad meg a felhasználó számára. Az előkészített SQL lekérdezés két „alselect” utasítással rendelkezik, amely lehetővé teszi, hogy olyan rekordokat frissítsünk, amely a felhasználóhoz tartozik, de csak azzal a lehetőséggel, ha aktív a felhasználó jelenlegi munkamenete. Erre a szűrésre azért van szükség, mert a fejlesztés alkalmával még nem állt rendelkezésre az SSO jogosultság ellenőrzését biztosító rendszer. Ha a kiválasztott rekord „active” mezőjének értékét sikerült hamisra állítani, akkor „A munkamenet sikeresen

érvénytelenítésre került!” üzenet kerül megjelenítésre, illetve a rendszernaplóba „SSO - Token session kényszerített deaktiválása” címmel kerül bejegyzés.

```
$stmt = $db->conn->prepare("UPDATE `sessions` SET `sessions`.`active` = 0 WHERE `sessions`.`id` = ? AND `sessions`.`user_id` = (SELECT `user_id` FROM (SELECT `user_id` FROM `sessions` WHERE `token` = ? AND `active` is TRUE) AS x);");
$stmt->bind_param("ss", $session_id, $token);
```

- 5) Felhasználó saját szerepköreinek lekérdezése az „accesslist.php” segítségével történik. A token az SQL utasításban kerül megvizsgálásra, sikeres lekérdezés esetén megjelenítésre kerül a hozzáférés és a felhasználó FMS azonosítója, a hozzáférés létrejöttének és utolsó módosításának időpontja, a vállalat azonosítója és neve, a szerepkör azonosítója és neve, illetve a szerepkör létrejöttének és utolsó módosításának időpontja. Ha nem tartalmaz aktív szerepkört a felhasználó, vagy a token alapján nem sikerült a hitelesítés, akkor a „Nincsen felhasználóhoz rendelve aktív szerepkör!” kerül megjelenítésre.
- 6) Felhasználó vállalati hozzáférései a „companyaccess.php” segítségével kérhetők le az adatbázisból. A felhasználói alapjogosultságot ellenőrző rendszer hitelesítése után az SQL utasításban az allekérdezés a tokenhez tartozó felhasználó FMS azonosítója alapján az „accesses”, vagyis a hozzáféréseket tartalmazó táblában keres egyezést azokkal, amelyek szerepköre a „roles” táblában hozzá van rendelve egy vállalathoz. A bővített keresés azért hasznos, mivel az alapjogosultságok (beépített szerepkörök) nem rendelkeznek vállalattal, így ezek kiválasztása nem lehetséges. Előfordulhat, hogy egy vállalat több szerepköréhez egy felhasználó több hozzárendeléssel rendelkezik, így a „DISTINCT” utasítással minden vállalati szerepkör csak egyszer kerül megjelenítésre.

A lekérdezett sorok a „result” elnevezésű tömbbe kerülnek elmentésre, amin ezután egy ciklus végigmegy.

```
$result = [];
while ($row = $res->fetch_assoc()) {
    array_push($result, $row);
}
```

A ciklusban a vállalat logójának ellenőrzése történik, ugyanis amelyik vállalat nem rendelkezik logóval, annak az egyik FELX logó kerül beállításra.

```
for ($i=0; $i < count($result) ; $i++) {
    if ($result[$i]["logo"] == NULL) {
        $result[$i]["logo"] = "https://cdn.felx.hu/assets/img/logo_x.png";
    }
}
```

A sikeres lekérdezésben a vállalat neve, logója és azonosítója jelenik meg. Az utóbbi adat az, amely kiemelten fontos szerepet játszik az API működésében, mivel a vállalat-specifikus lekérdezések használatához a vállalat azonosítóját kötelező megadni a POST kulcsérték-párok között.

- 7) Saját felhasználói adatok módosítása az „update.php” meghívásával történik, amelyben az értékek ellenőrzése más elven működik az eddigiekhez képest. Itt is kötelező megadni a POST kulcsait, viszont a kulcsokhoz tartozó értékek hossza egyenlő lehet nullával. Kivétel a token, név, születési név és kapcsolattartói e-mail cím megadása. A saját adatok módosításához elegendő az SSO által vizsgált alap felhasználói jogosultság. A frissítés során nem elegendő csak a frissítendő adatokat megadni, hanem a korábbiak megadására is szükség van. Sikeres módosítás esetén a rendszernaplóba bejegyzés mellett a „Sikeresen módosította a felhasználói adatokat!” üzenet jelenik meg a felhasználó számára. Amennyiben egyik adat sem módosult, vagy hibás a lekérdezés, akkor „A felhasználói adatok nem változtak, vagy nem sikerült a módosítás!” hibaüzenet tájékoztatja a felhasználót a sikertelenségről.
- 8) Saját jelszó módosítása a „passwordchange.php” segítségével történik. POST kulcsok közül kötelező megadni a régi jelszót, új jelszót. Ezen kívül, ha a felhasználó ki szeretné léptetni az összes eszközről saját magát, akkor a „logout” érték megadása is kötelező. Az utóbbi értéke csak 0 vagy 1 lehet, amelynél az 1 érték jelentkezteti ki az összes eszközt.

```
if($logout != 0 && $logout != 1) {  
    $msg = [ "success" => FALSE, "result" => "Value Error", "message" => "Egy  
vagy több paraméter értéke nem megfelelő!"];  
    die(json_encode($msg, JSON_UNESCAPED_UNICODE));  
}
```

Ellenőrzésre kerül, hogy a régi és az új jelszó megegyezik-e. A jelszavak az API felé már titkosított állapotban érkeznek meg, adatvédelmi szempontok szerint a titkosított jelszóra egy másfajta további titkosítás is hozzáadásra kerül, amelyben az oldal SHA256 titkosítással további védelmet tesz a jelszavakra.

```
$newpassword = hash('sha256',strip_tags(trim($_POST['newpassword'])) . '  
AXfGrSSEkW7cRP9D'));
```

Amennyiben minden megfelelő, akkor az SQL utasítás ellenőrzi a jelszavak módosításának lehetőségét. Az allekérdezés ellenőrzi, hogy a felhasználó rendelkezik-e aktív tokennel. Az utasításban a frissítés csak akkor kerül végrehajtásra, ha a régi jelszó megegyezik az adatbázisban nyilvántartottal. Sikeres

jelszómódosítás esetén, ha a „logout” értéke 1 vagyis igaz, akkor meghívásra kerül a „DeactiveAllSessions” függvény, amelynek visszatérési értéke az összes olyan token érvénytelenítése, amely a felhasználóhoz tartozik. Ha ez sikeres, akkor a „deactivesessions” kulcs értéke igaz lesz a lekérdezésben.

```
if ($stmt->affected_rows <= 0 || $stmt->affected_rows == NULL)
{ $msg = FailedAction($token); }
else {
    $msg = SuccessfulAction($token);
    if ($logout == TRUE)
        { $msg["result"] = ["deactivesessions" => DeactiveAllSessions($db, $token)];}
}
```

- 9) Biztonsági kulcs generálása a „secretkey.php” segítségével lehetséges, ahol a token elküldésén kívül az egyik legerősebb védelmi ellenőrzés, azaz a jelszó ellenőrzése is megtörténik. Erre azért van szükség, mivel a kulcs generálása olyan potenciális veszélyeket is tartalmazhat, amely sértheti a saját személyes adatok védelmét. A megadott jelszó és a tokenhez tartozó felhasználó FMS azonosító együttesével hitelesítésre kerül a kérés, majd egy random generált kulcs kerül elkészítésre, amely első három karaktere minden esetben az „fms”, ezek után kötőjelekkel elválasztva egy 5 tagú karakterlánc generálása következik, amely feltöltésre kerül az adatbázisba aktív státusszal. A biztonsági kulcs csak a generálásakor visszaadott válaszkor látszódik, mivel az adatbázisba a generált kulcs „sha256” titkosítás kerül fel. A kulcs sikeres létrehozását a következő üzenet nyugtázza:

```
{
    "success": true,
    "result": {
        "user_id": 1000000000,
        "secretkey": "fms-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX"
    },
    "message": "Sikeresen generált egy aktív biztonsági kulcsot!"
}
```

Fontos, hogy a kulcs csak egyszer használható fel a későbbiek folyamán, mivel a személyes adatok lekéréssével automatikusan deaktiválásra kerül. Természetesen a felhasználásról és a generálásról is születik bejegyzés a rendszernaplóban.

- 10) Kijelentkezés a rendszerből, vagyis a munkamenet határidő előtti érvénytelenítése a „logout.php” használatával lehetséges. A POST kulcsérték-párként megadott token érvénytelenítésre kerül, amennyiben a "Sikeresen kijelentkezett a munkamenetből!" üzenetet kapja a felhasználó válaszul. Ha a kulcs nem létezik

vagy a munkamenet már érvénytelenítésre került, akkor ez hibaüzenet részletezésre kerül.

11) Saját fiók törlését a „deldata.php” állomány segítségével lehet végrehajtani. Ehhez a token megadásán túl a saját jelszó megadására is szükség van, mivel potenciálisan veszélyes művelet elvégzésére készülhet a felhasználó. Ha a megadott jelszó megegyezik a rendszerben lévővel, akkor a személyes adatok eltávolítása esetén a név helyére a „Törölt felhasználó (XXXXXXXXXX)” kerül. A fiktív kapcsolattartói e-mail címben található @ előtt, illetve a névnél zárójelben lévő adat a kérelem azonosítóját tartalmazza, amely a fiók törlésekor kerül megjelenítésre. Fontos, hogy a kérelem nem kerül eltárolásra a személyes adatokkal összekapcsolva. A fiók típusa ilyenkor „FMS Deleted User”-re változik, illetve a jelszó helyére egy olyan random generált kulcs kerül eltárolásra, amellyel a bejelentkezés megakadályozásra kerül. Ezeken kívül törlésre kerül a felhasználónak az összes szerepkörhöz való hozzáférése, többek között az FMS alapjogosultságai is, így a bejelentkezést a rendszer lehetetlenné teszi. Eltávolításra kerül az összes vállalati munkacsoportból a felhasználó, majd az összes munkamenete érvénytelenítésre kerül, a generált biztonsági kulcsokkal együtt. A folyamat végeztével a fiók visszaállítására nincs lehetőség, erről a következő üzenet tájékoztatja a felhasználót.

```
{  
    "success": true,  
    "result": {  
        "user_id": 1000000004,  
        "request_id": "e8654925ab"  
    },  
    "message": "A(z) e8654925ab azonosítóval ellátott fiók törlési kérelem teljesítésre került!"  
}
```

A feladatok, illetve ahhoz rögzített üzenetek nem kerülnek törlésre a rendszerből, mivel konkrét vállalatok munkáit bénítaná meg, ha a felhasználó a törlés mellett dönt. Ezeken túl a biztonsági kulcsok és a munkamenetek nem kerülnek visszamenőleg törlésre a kérelem teljesítése alatt, ezek törlését a hatályos adatvédelmi és jogi nyilatkozatok írják elő.

3.1.14 Rendszernapló

A napló megalkotásának fő célja a rendszer folyamatos monitorozása, statisztikai és hibakeresési szempontokat figyelembe véve. Ezeken túl a munkamenet létrejöttétől, a manuális lezáráсáig vagy lejártáig részletesen nyomon követhető a felhasználó által

elvégzett műveletek. Visszaélés esetén a felhasználó számára konkrét információkkal tudunk szolgálni a munkamenetében végzett cselekményeiről. Például: ha a felhasználónak megváltozott a jelszava, amelyet nem a tényleges személy végzett el, akkor a munkamenet információi segítségével visszakereshető, szükség esetén eljárás indítható azzal a személyel szemben, aki más adataival él vissza.

Jelen állás szerint a rendszernaplóból a felhasználók részére nincs lehetőség adatokat kiolvasni, ezért csak a bejegyzés része van használatban, amely az „add.php” fájlban létrehozott „Log” osztállyal kezelhető.

Az osztály létrehozásakor, a konstruktörben megadott token és esemény alapján eltárolásra kerül a megfelelő változókban, az API elérési linkjével együttesen. Az „Execute” metódus végrehajtásakor a rendszer csatlakozik az adatbázishoz, majd a „logs” táblába feltölti az aktuális adatokat. Az „msg” mező tartalmazza a feltöltés sikerességéről szóló információkat. Ezek után a kapcsolat bezárásra kerül.

```
function __construct($token, $event) {  
    $this->token = $token;  
    $this->event = $event;  
    $this->actual_link = "{$_SERVER[HTTP_HOST]}{$_SERVER[REQUEST_URI]}";  
}
```

3.2 Webes alkalmazás

A webes felület kialakítása során felhasznált főbb programnyelvek a PHP, HTML, CSS és JavaScript, ezek közül a JavaScriptet emelném ki főként, ugyanis ez felel az oldal legfőbb funkcióinak működéséért, továbbá ennek a fejlesztésével töltöttem a legtöbb időt.

3.2.1 Globális fájlok

Két globálisan felhasznált állományt hoztam létre, ezek szinte minden oldalon szerepet kaptak, így a „cdn.felx.hu/assets/” elérési út alatt lehet hozzájuk férni.

A „main.css” fájl a böngésző általi alapformázások eltávolítását, illetve az általam írt, oldalakon átívelő alapformázásokat tartalmazza. Ezek közé tartozik a bemeneti mezők kinézete, a színek változókba szervezése, betűméret beállítása, a navigáció kinézete, az alap elrendezés, továbbá az oldal reszponzivitásához elengedhetetlen „media query”-k.

A „:root” szelektorban deklaráltam az összes globális változót, amire szükség lehet az oldalon keresztül, például az 5 fő színnek színenként 3 árnyalata, amik konzisztensek a 3 platformon, a belső margó mérete, illetve egy alternatív margó, amely az előzőnek a fele. Ezen kívül az alap betűméretet és betűtípust is beállítottam.

Ezen kívül az alap betűméretet és betűtípust is beállítottam, a méretet az alap 16 képpontos értékről 20 képpontra, így javítva az olvashatóságot, továbbra a betűtípust a Google Fonts oldalról beimportált „Ubuntu” nevű betűtípusra állítottam.

```
:root {  
    font-size: 20px;  
    font-family: 'Ubuntu', Arial, Helvetica, sans-serif;  
    --light: 242, 242, 248;  
    --light-2: 250, 250, 255;  
    --light-3: 215, 215, 234;  
    --dark: 45, 48, 71;  
    --dark-2: 64, 68, 100;  
    --dark-3: 32, 34, 50;  
    --burgundy: 164, 48, 63;  
    --burgundy-2: 200, 65, 83;  
    --burgundy-3: 142, 41, 55;  
    --orange: 252, 122, 30;  
    --orange-2: 252, 137, 54;  
    --orange-3: 241, 103, 4;  
    --green: 29, 175, 104;  
    --green-2: 34, 211, 125;  
    --green-3: 23, 140, 84;  
    --gap: 2rem;  
    --half-gap: calc(var(--gap) / 2);  
}
```

A „main.js” tartalmazza azon függvényeket, melyek minden oldalon elengedhetetlenek a megfelelő működés érdekében. Az API domain-je egy konstansként van eltárolva, így ha az API címe megváltozna, akkor elég lenne ezt az egy értéket módosítani, nem szükséges az összes állományban átirni az új elérési útra. Létezik továbbá két függvény, amelyek a sütik olvasására és írására szolgálnak. Írtam egy alprogramot a felugró üzenetek megjelenítésére, amit több alkalommal is meghívtam a többi JavaScript fájlban, API lekérdezést követő visszatérési státuszüzenetek megjelenítésének érdekében.

Készítettem egy általános használatra való kódrészleteket, melyek elősegítik a háttérben történő „AJAX” (Asynchronous JavaScript and XML) lekérdezések „POST” módszerrel.

```
let formData = new FormData();  
fetch(``, {  
    method: "POST",  
    body: formData  
})  
.then(response => response.json())  
.then(data => {  
    console.log(data); }).catch(error => console.log(error));
```

Ennek a „GET” módszeren alapuló verziója is elérhető, ez többek között az országok listájának lekérdezésénél kapott szerepet.

A „formData” változóban tárolódnak azon adatok, melyek továbbküldésre kerülnek az API számára, legtöbb esetben a bejelentkezéi munkamenet megtartásáért szolgáló véletlenszerű karakterlánc.

```
fetch(``)
.then(response => response.json())
.then(data => {
    console.log(data);

})
.catch(error => console.error(error));
```

Ezutóbi használatakor nincs szükség „FormData” objektumra, ugyanis nem küldünk adatot az API felé, csupán lekérdezzük a válaszban tárolt adatokat.

3.2.2 Kezdőoldal

A webes felület kialakítása JavaScript, PHP, HTML, illetve CSS kódok egybehangolásával történt. A főoldal HTML leírónyelven készült, ezt leszámítva minden más oldal PHP nyelven, ugyanis ezekben a megjelenést megelőzően ellenőrizni kell, hogy a felhasználó be van-e jelentkezve vagy sem, ez a művelet pedig túl lassú lenne JavaScript használatával végrehajtott, háttérben történő lekérdezések használatával.

Mivel törekedni kellett az oldal reszponzivitására, a navigációs sávot mobileszközön egy hamburger menü váltja le, amelyre ha rákattint a felhasználó, akkor jobbról beúsznak a navigációs gombok, mindig az előzőhöz képest 50 milliszekundumos késleltetéssel, amit a CSS kezel.

A „FELX MANAGEMENT SYSTEM” felirat animációja CSS-el, illetve JavaScripttel valósult meg, a karakterek külön elemekként vannak deklarálva, így egyesével animálhatók.

```
let bannerTitle = document.getElementById('banner-title');
let bannerDescription = document.getElementById('banner-description');
let words = document.querySelectorAll('.word');
let chars = document.querySelectorAll('.char');
setTimeout(() => {
    let index = 0;
    const word = setInterval(() => {
        chars[index].classList.add("show");
        index++;
        if (chars.length <= index) {
            clearInterval(word);
        }
    }, 75);}, 10);
```

A szekciókban megjelenő képek SVG fájlok, amelyeket én készítettem az Adobe Illustrator 2023 programmal, az animálásukat pedig JavaScript kezeli, ami mindenkor játszódik le, mikor a felhasználó számára láthatóvá válnak.

3.2.3 Kezelőfelület

A kezelőfelületen a lekérdezéseket és a különböző oldalak közötti navigációt teljes mértékben a JavaScript kezeli, nem történik másik oldalra való átirányítás. A lekérdezések a „Fetch API” használatával valósulnak meg, ez teszi lehetővé, hogy a folyamatok a háttérben menjek végbe, ahelyett, hogy a felhasználót minden alkalommal átirányítanánk egy másik kezelőfelületre.

A JavaScript kód külön fájlokra van bontva a menüpontok szerint, ezeken belül is minden függvényekbe van szervezve, így akár egy függvény meghívásával megjeleníthető egy menüponthoz tartozó teljes oldal, például a „getUserData” alprogram a „Fiók” menüpont alatt lévő oldalt tölti be, majd jeleníti meg. A navigáció is ezeknek a függvények a meghívásával működik.

```
function getUserData() {
    fetch(`$apiURL}/sso/user/full`, {
        method: "POST",
        body: token
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            user = data.result[0];
            renderUserData();
        }
        else {
            showToast(data.message, 3);
        }
    })
    .catch(error => {
        console.log(error);
        showToast("Nem lehetett kapcsolatot létesíteni a kiszolgálóval", 3);
    });
}
```

Azért döntöttem „egyoldalas” kialakítás mellett, mert a különböző lapok közötti váltás gyorsabban történik meg, ha azt a JavaScript kezeli, mintha tényleges átirányítás történne PHP fájlok között.

3.2.4 Bejelentkezés

A felhasználó bejelentkezési adatainak beírását, majd a bejelentkezés gombra való kattintást követően az oldal a háttérben lekérdezi a legfrissebb adatvédelmi nyilatkozatot az API-ról, amelyet a felhasználónak kötelezően el kell fogadni ahhoz, hogy bejelentkezzen. Az elfogadást követően a bejelentkezési űrlap alatt a JavaScript megjelenít egy üzenetet, amely sikeres bejelentkezés esetén zöld, sikertelennél pedig bordó.

A „login.php” állománynak köszönhetően a felhasználó jelszava már titkosított formában kerül továbbításra az API számára, így teljesen biztonságossá téve a folyamatot, továbbá lekérdezi a kliens bizonyos adatait, például a böngésző nevét és mobil eszközről történő bejelentkezés esetén az eszköz azonosítóját is. Ha ennek a fájlnak a visszatérési értékében szereplő „success” mező igaz értékkel rendelkezik, akkor a „login.js” fájl beállítja a munkamenet megtartásához szükséges sütit, majd egy másodperc múlva átirányítja a felhasználót a kezelőfelületre.

3.2.5 Vállalat választás

Egészen addig, amíg a felhasználó nem választ vállalatot, nem fér hozzá sem a feladatok menüponthoz, sem pedig az adminisztráció menüponthoz. A kiválasztott vállalatot az oldal egy fehér kerettel, illetve enyhén nagyobb mérettel jelzi.

Abban az esetben, ha a felhasználó csak egy vállalathoz van hozzárendelve, azonnal, a vállalatok adatainak lekérését követően a választást automatikusan elvégzi a „company.js”, majd átirányítja a felhasználót a feladatok kezelésére szolgáló felületre.

Választást követően a felhasználó bármikor újra megnyithatja ezt az oldalt, és kiválaszthat egy másik vállalatot.

3.2.6 Feladatok kezelése

A feladatok csempékként jelennek meg az oldalon, egy dinamikus rácsos elrendezésben, ami az ablak szélességéhez igazítja az oszlopok számát.

```
#task-container {  
    width: 100%;  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(min(25rem, 100%), 1fr));  
    padding: var(--gap);  
    gap: var(--gap);  
}
```

Egy feladatra való kattintáskor a „task.js” lekérdezi a feladat részleteit, így minden esetben a legfrissebb adatokat látjuk a felugró ablakban.

A felugró ablakban lehetőség van a feladat státuszának változtatására, továbbá megjegyzés írására, minden esetben egy beúszó üzenet értesít, hogy sikeresen végbement-e a tranzakció.

3.2.7 Feladatok rögzítése

Ha a felhasználó rendelkezik a kiválasztott vállalaton belül megfelelő jogosultsággal, akkor megjelenik a jobb alsó sarokban egy pluszjel, ezt az ellenőrzést a „task.js” fájl hajtja végre. A pluszjelre kattintva megjelenik a képernyő közepén egy űrlap, melynek kitöltésével új feladatot rögzíthetünk.

Új feladat rögzítésénél kötelező megadni a feladat címét, illetve kiválasztani a kívánt munkacsoportot. A határidő megadása nem kötelező, abban az esetben, ha a felhasználó nem akar hozzárendelni a feladathoz, elég üres állapotban hagynia. A sikeres létrehozásról beúszó üzenetben értesül a felhasználó.

A feladatok létrehozását a „createTask” alprogram végzi, ehhez hasonló módon működik a legtöbb függvény, amely valamilyen összetett adathalmaznak az FMS rendszerbe való rögzítésére szolgál.

```
function createTask(title, deadline, group) {
    let taskParams = new FormData();
    taskParams.append("token", session);
    taskParams.append("taskgroup", group);
    taskParams.append("title", title);
    if (deadline.length != 0) {
        taskParams.append("deadline", deadline);
    }
    taskParams.append("company", selectedCompany);
    fetch(`${apiURL}/sso/task/adddtask`, {
        method: "POST",
        body: taskParams
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            showToast(data.message, 1);
        }
        else {
            showToast(data.message, 3);
        }
        getTasks();
    })
    .catch(error => console.log(error));}
```

3.2.8 Felhasználói adatok

A felhasználói adatok megtekintését és módosítását a „Fiók” menüpont alatt lehet elérni. Az adatok megjelenítése beviteli mezők segítségével történik, ezeknek a létrehozására külön alprogramokat írtam, melyek a paramétereknek megfelelően létrehozzák és hozzáadják a mezőt az oldalhoz. Ezen alprogramok közé tartoznak a „createInput”, a „createDropdown”, illetve a „createCheckbox” elnevezésű függvények, melyek mindegyike megköveteli legalább a mező szülő elemét, a hozzárendelendő címke szövegét, a mező azonosítóját, továbbá azt, hogy engedélyezett-e a felhasználónak a benne lévő adatot módosítani, mindeneket paraméterek formájában.

Az oldalon történő eseményekért a „user.js” állományban található függvények felelősek.

```
function createInput(parent, labelText, id, type, value, editable, additionalAttributes) {  
    let inputWrapper = document.createElement("div");  
    inputWrapper.classList.add("input-wrapper");  
    if (0 < labelText.length) {  
        let label = document.createElement("label");  
        label.setAttribute("for", id);  
        label.innerText = labelText;  
        inputWrapper.appendChild(label);  
    }  
    let input = document.createElement("input");  
    input.setAttribute("id", id);  
    input.setAttribute("type", type);  
    if (additionalAttributes.length) {  
        for (const attribute of additionalAttributes) {  
            input.setAttribute(attribute.key, attribute.value);  
        }  
    }  
    input.value = value;  
    if (editable) {input.setAttribute("data-editable", "");}  
    else {input.setAttribute("disabled", "")};  
    inputWrapper.appendChild(input);  
    parent.appendChild(inputWrapper);  
}
```

A fenti kódrészlet felelős a felhasználói adatok nagy részének megjelenítéséért, többek között az „FMS azonosító”, „Teljes név”, „E-mail” mezőkért is.

Ha a felhasználó megváltoztatta a kívánt adatokat, akkor a módosításait a „Mentés” gombra kattintva küldheti el az API-nak, ami ellenőrzi, majd ha nem talál problémát, akkor elmenti ezeket az adatokat.

3.2.9 Biztonsági kulcs generálás

Szintén a „Fiók” menüpont alatt érhető el, a felhasználói adatok megváltoztatására szolgáló űrlap alatt. A felhasználói jelszó beírását követően a „Generálás” gombra kattintva a „createSecretkey” alprogram leellenőrzi a jelszót a „secretkey.php” háttérben való elérésével, és ha a megfelelő jelszó lett begépelve, akkor a felhasználónak felugró ablakban megjelenik az egyszer használatos biztonsági kulcs, továbbá alatta egy QR kód, amely szintén azt tartalmazza. A QR kód generálását az API végzi.

3.2.10 Jelszómódosítás

A jelszómódosítás a biztonsági kulcs generálása szekció alatt foglal helyet, a felhasználónak kötelezően 3 mezőt kell kitöltenie, ezek a „Jelenlegi jelszó”, az „Új jelszó” és az „Új jelszó megerősítése” címkekkel vannak ellátva. A jelszavak átfutnak egy ellenőrzésen, melyet a „change_password.php” fájl végez el. A mezők, melyek az új jelszót tartalmazzák, meg kell, hogy egyezzenek, különben hibával tér vissza a fájl. Itt az új jelszónak eleget kell tenni az általunk megszabott feltételeknek is, legalább 8 karakterből kell állnia, kell tartalmaznia kis- és nagybetűket, illetve legalább egy számot.

```
(  
    strlen($newpass) < 8 || ctype_lower($newpass) ||  
    ctype_upper($newpass) || ctype_alpha($newpass) ||  
    ctype_digit($newpass) || strtolower($newpass) == $newpass ||  
    strtoupper($newpass) == $newpass || ($newpass != $newpass_again)  
)
```

Ha a fenti összetett feltétel hamis értékkel tér vissza, akkor a felhasználó jelszava sikeresen megváltozik, erről beúszó üzenet formájában is értesül az oldalon.

Az egyetlen opcionális mező a „Kijelentkezés az összes eszközből” címkelvel rendelkező jelölőnégyzet, amelyet ha megjelöli a felhasználó, akkor a jelszó módosítását követően minden munkamenetét kijelentkezteti az API, a mobil- és asztali alkalmazásban lévőket is.

3.2.11 Aktív munkamenetek

Ez a „Fiók” menüpont utolsó szekciója, itt láthatók és kezelhetők a felhasználó jelenleg aktív munkamenetei. A munkamenetek a feladatokhoz hasonlóan dinamikus rácsos elrendezésben jelennek meg, rájuk kattintva megjelennek a részleteik, és lehetőségünk van deaktiválni őket.

3.2.12 Vállalat adminisztrációja

Az „Adminisztráció” menüpont alatt foglalnak helyet a vállalat kezelésével kapcsolatos szekciók, ez az oldal két fő részre osztható, az első a vállalathoz rendelt felhasználók kezelésére szolgál, az ezt követő pedig a vállalati munkacsoportok adminisztrálására.

3.2.13 Felhasználók kezelése

A „Vállalathoz hozzárendelt felhasználók” címsor alatt jelennek meg azok a felhasználók, akik jelenleg hozzá vannak rendelve a választott vállalathoz. A felhasználókra kattintva megtekinthetjük adataikat, viszont azok módosítására nincs lehetőség, mindegyikük csak saját maga képes erre. Lehetőségünk van szerepkör hozzárendelésére a személyekhez, továbbá ezeknek a szerepköröknek a törlésére is.

A felsorolt felhasználók alatt két gomb jelenik meg, amelyek segítségével új felhasználókat hozhatunk létre, vagy a már FMS rendszerében rögzítetteket hozzárendelhetjük.

Létrehozás során kötelező megadni az új felhasználó nevét, e-mail címét, születési nevét, illetve a nemét.

Regisztrált felhasználó hozzárendelésekor a felhasználó által generált biztonsági kulcsra és a megfelelő munkacsoport kiválasztására van szükség. A weboldalon nem lett implementálva a QR kód beolvasásának lehetősége, mivel ezt a funkciót inkább csak mobil alkalmazások esetén lehet megfigyelni.

3.2.14 Munkacsoportok kezelése

A munkacsoportok ugyan olyan megjelenéssel rendelkeznek, mint a felhasználók. Egy munkacsoportra kattintva megjelennek az adott csoporthoz rendelt felhasználók, továbbá lehetőség nyílik új felhasználó hozzárendelésére.

A felsorolt munkacsoportok alatt található gomb megnyomásával hozható létre új munkacsoport, ahol csupán a csoport nevét kell megadni a létrehozáshoz.

3.2.15 Kijelentkezés

A kijelentkezési folyamatot a „logout.php” fájl kezeli, amely elküldi a kijelentkezési kérelmet a felhasználó tokenjével az API felé, a böngészőből a bejelentkezésért szolgáló sütit érvényteleníti, majd átirányítja a felhasználót a bejelentkezési felületre.

```

<?php
$cookie_key = "session";
if(isset($_COOKIE[$cookie_key])) {
    $token = $_COOKIE[$cookie_key];
    $form_data = ['token' => $token];
    $ch = curl_init('https://api.felix.hu/sso/user/logout');
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($form_data));
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $response = curl_exec($ch);
    curl_close($ch);
}
if (isset($_COOKIE["session"])) {
    unset($_COOKIE["session"]);
    setcookie("session", "", time()-3600, "/");
}
header("Location: index.php");
exit();
?>

```

3.3 Mobil alkalmazás

A mobil alkalmazás a Flutter keretrendszer segítségével került megírásra. A Flutter a Dart objektumorientált programozási (OOP) nyelvet használja, amelyet a Google fejlesztett ki.

A keretrendszer „lib” nevű mappája tartalmazza a működéshez és a megjelenítéshez szükséges mappákat és fájlokat. Ebben a mappában a következő mappákat hoztam létre: models, views, utils.

Fejlesztői környezetnek az Android Studio-t választottam, mivel egyrészt ez a legismertebb mobilos fejlesztői környezet, másrészt maximálisan támogatja és megkönnyíti a programozás folyamatát.

3.3.1 Main.dart állomány

A „main.dart” fájl tartalmazza az app azon beállításait, amit még az alkalmazás betöltése előtt kell megadni. A main() metóduson belül adtam meg főbb beállításokat, amiket csak ezen metóduson belül lehet meghívni.

```
void main() async {
```

Az „ensureInitialized()” metódus biztosítja, hogy az alkalmazás inicializálási folyamata befejeződjön, még mielőtt a többi kód meghívásra kerül. Az inicializálási feladatok sikeres befejezése elengedhetetlen ahhoz, hogy az alkalmazás megfelelően működjön.

```
WidgetsFlutterBinding.ensureInitialized();
```

A „SystemChrome.setEnabledSystemUIMode()” metódus segítségével beállítható a rendszer felhasználói felületének módja. A „SystemUiOverlay.top” beállításával az operációs rendszer értesítési sávja megjelenítve marad. A „SystemUiOverlay.bottom” a navigációs gombokat (vissza, kezdőképernyő, alkalmazásválasztó) hagyja a képernyő alján.

```
SystemChrome.setEnabledSystemUIMode(SystemUiMode.manual, overlays:  
[SystemUiOverlay.bottom, SystemUiOverlay.top]);
```

A „SystemChrome.setPreferredOrientations()” metódus meghívása az alkalmazás képernyőorientációjának beállítására szolgál. A „portraitUp” és „portraitDown” beállítással csak állított módban fog üzemelni az alkalmazás, nem fordul el fektetett nézetbe, ha az eszközt elfordítják.

```
SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp,  
DeviceOrientation.portraitDown]);
```

A „GetStorage.init()” inicializálja az alkalmazás adattárolóját, a GetStorage segítségével felhasználói adatokat tudunk tárolni és olvasni.

```
await GetStorage.init();  
await GetStorage.init('loginInfo');  
await GetStorage.init('settingsInfo');
```

A „runApp()” utasítás meghívása elindítja az alkalmazást, és megjeleníti a „FelX” widgetet (A widget egy olyan építőelem, amely a felhasználói felület egy részének viselkedését és megjelenését határozza meg) a képernyőn.

```
runApp(const FelX());}
```

A „FelX” osztály egy olyan widget, amely az „StatelessWidget” osztályból származik. Az „StatelessWidget” osztály azoknak a widgeteknek a főosztálya, amelyek állapotmentesek, a widget nem tartalmaz változó állapotot, amely befolyásolná a felhasználói felület megjelenését.

```
class FelX extends StatelessWidget {  
const FelX({super.key});
```

Az „@override” annotáció használatával felülírjuk az ősosztály egy meglévő metódusát.

```
@override
```

A „build” metódus feladata a widget fa felépítése és visszaadása, amely meghatározza a felhasználói felület megjelenését. A metódus egy „BuildContext” objektumot vár paraméterként, amely reprezentálja a widget fa aktuális pontját.

```
Widget build(BuildContext context) {
```

A „return AdaptiveTheme()” visszatér az „AdaptiveTheme” widgettel, amellyel az alkalmazás színének és témájának beállítására van lehetőség.

```
return AdaptiveTheme(  
    dark: ThemeData(  
        fontFamily: 'Ubuntu'),
```

Az „initial” tulajdonság az alkalmazás indulásakor megjelenő alapértelmezett téma beállítását teszi lehetővé. Az „AdaptiveThemeMode.dark” értékkel a sötét témát állítjuk be.

```
initial: AdaptiveThemeMode.dark,
```

A „MaterialApp” az egyik legfontosabb widget az alkalmazásban. Ez a widget a felhasználói felület biztosítása érdekében szükséges.

```
builder: (theme, darkTheme) => MaterialApp(
```

Az „Overlay” egy olyan widget, amely lehetővé teszi más widgetek elhelyezését egymáson. Ennek a widgetnek a segítségével az alkalmazás bármelyik pontján meg lehet jeleníteni felugró üzeneteket.

```
builder: (context, child) => Overlay(
```

```
    initialEntries: [  
        if (child != null) ...[  
            OverlayEntry(  
                builder: (context) => child, ), ], ], ),
```

A „debugShowCheckedModeBanner” tulajdonság „false”-ra való állításával a fejlesztési környezetben eltünteti a „Debug” feliratot az alkalmazás futtatása közben.

```
debugShowCheckedModeBanner: false,
```

A „title” tulajdonsággal az alapértelmezett címsort tudjuk beállítani, ami az alkalmazás tetején jelenhet meg.

```
title: 'FelX Management System',
```

A „home” beállításával a kezdő oldalt állítjuk be, hogy melyik legyen a következő oldal, amit az alkalmazás elindítson.

```
home: const AuthPage(),
```

A „navigatorKey” tulajdonság az alkalmazás által kezelt különböző oldalak közötti navigációt kezeli.

```
navigatorKey: navigatorKey, ), ); } }
```

3.3.2 Models mappa

A „models” mappában hoztam létre azokat a „.dart” kiterjesztésű fájlokat, amelyekben meghatároztam az osztályokat és azon belül az adatok típusait.

Ezek közül a „company.dart” fájlt emelném ki, amelyben létrehoztam egy Company nevű osztályt.

```
class Company { }
```

Ezen belül létrehoztam egy konstruktort, melyben felsoroltam a szükséges tulajdonságokat.

```
const Company(this.id, this.name, this.logo, this.visible);
```

Ezt követően az API-ból kapott JSON fájlból fromJson függvény segítségével, egyesével beolvastam a tulajdonságokat és eltároltam egy-egy változóban.

```
Company.fromJson(Map<String, dynamic> json)
  : id = json['id'],
    name = json['name'],
    logo = json['logo'],
    visible = true;
```

Végül meghatároztam egyesével a változók típusát (szám, logikai vagy karakterlánc).

```
final int id;
final String name;
final String logo;
final bool visible;
```

3.3.3 Bejelentkezés ellenőrzése

Az „auth.dart” fájlban ellenőrizzük, hogy a felhasználó az alkalmazás elindításakor be van-e jelentkezve, vagy nincs.

Az „initState()” metódusban, ami csak az oldal betöltésekor fut le egyszer, a „GetStorage” objektummal egyes beállításokat állítunk be, mint például, hogy jelenleg frissít-e az alkalmazás, vagy a felhasználó választotta-e a „Emlékeztessen később” gombot a frissítésnél.

```
@override
void initState() { }
```

A „load()” metódusban ellenőrizzük, hogy a felhasználó jelenleg be van-e jelentkezve az app-ba.

```
if (box.read("token") != null) { }
```

Amennyiben a felhasználó tokene null, vagyis nincs értéke, átirányítjuk a bejelentkezési oldalra. Ha a token értéke nem üres, akkor meghívjuk a „FMS().authUser()” metódust, ami ellenőrzi, hogy az appban engedélyezve van-e a biometrikus azonosítás. A sikeres ellenőrzés után az „authUser()” metódus visszatér egy „true” logikai értékkel, és a felhasználót átvisszük a főoldalra. Ha a felhasználónak nem sikerült az azonosítás, akkor a „retry” változót „true” értékre állítjuk, és megjelenítjük a „Belépés újrapróbálása” gombot, ami újra meghívja a „load()” metódust.

3.3.4 Bejelentkezés

A „login.dart” fájlban a felhasználó bejelentkezése történik meg. Az oldal megjelenítésekor ellenőrizzük, hogy az alkalmazás csatlakozik-e internetre.

A „NetChecker().check()” használatával egy logikai értéket kapunk vissza, a jelenlegi internetelérésünkkel kapcsolatban. Első paraméterként a jelenlegi kontextust adjuk át, hogy jelenleg melyik részben vagyunk az appban. Második paraméternek, meg a logikai érték, ha szeretnénk-e felugró üzenetet, hogy nincs internet elérésünk.

```
if (await NetChecker().check(context, true)) { }
```

Amennyiben rendelkezünk internettel, frissítéseket kereshetünk az FMS osztályban létrehozott „checkForUpdates()” metódussal, itt is 2 paramétert kell megadnunk, ahogy feljebb, az internet ellenőrzésnél lett említve.

```
FMS().checkForUpdates(context, false);
```

Az „FMS().getNotice()” metódus használatával lekérjük az API-tól az adatvédelmi nyilatkozatot, és amint a felhasználó beírja az e-mail címét és jelszavát, majd a Bejelentkezés gombra koppint, a lekért nyilatkozatot megjelenítjük a „agreementAlert()” metódussal, amiben paraméterként a nyilatkozatot és a bejelentkezési információkat adjuk át.

```
BuildAlertDialog().agreementAlert(notice["result"]["title"], notice["result"]["details"] +  
"\n\nUtoljára frissítve: " + notice["result"]["date"], "Elfogadom", "Elutasítom",  
emailAddress.text, password.text, permission.id, loadingButton, context);
```

A nyilatkozat elfogadása és a helyes adatok megadása után átirányítjuk a felhasználót a főoldalra a „Navigator.pushAndRemoveUntil()” metódussal. A „type” tulajdonsággal állíthatjuk be, hogy milyen animációval térjen át a másik oldalra, amit a „child” tulajdonságban adhatunk meg,

```
Navigator.pushAndRemoveUntil(context, PageTransition(  
    type: PageTransitionType.rightToLeftWithFade,  
    alignment: Alignment.center,  
    child: const DrawerPage(selected: 0)  
(, (e) => false);
```

3.3.5 Főoldal

A „home.dart” fájlban a felhasználó a vállalat kiválasztását végezheti el. Az oldal megnyitásakor létrehozunk egy listát, amiben a vállalatokat tároljuk el. Ezt a listát a „load()” metódusban töltjük fel, miután lekértük a vállalatokat az API-ról.

```
static List<Company> items = <Company>[  
    const Company(-1,"Nincs vállalat", "https://cdn.felix.hu/logo.png", false),  
];
```

A vállalatokon kívül lekérjük még a felhasználó személyes adatait és aktív eszközeit is.

```
await FMS().getPersonalData();  
await FMS().getActiveDevices();
```

A vállalat kiválasztásakor leellenőrizzük a „checkPermission()” metódussal, hogy az adminisztráció oldalhoz van-e jog a felhasználónak, ha van, akkor letároljuk a „fullAccess” kulcs segítségével.

```
bool access = await FMS().checkPermission(items.where((element) =>
  element.visible).toList()[index].id, "fms_framework_company_manager_full");
box.write("fullAccess", access);
```

3.3.6 Feladatok

A „tasks.dart” fájlban a felhasználó a hozzárendelt feladatokat tudja kezelní. Az oldal megjelenítésekor létrehozunk egy listát, amiben a feladatokat fogjuk tárolni. Ezt a listát feltöltjük az API-ból lekért feladatokkal, majd kilistázzuk a képernyőre. Az egyik feladatra való koppintáskor meghívjuk a „getTaskMessages(setState, index)” metódust, melynek paraméterként átadjuk az aktuális státusz módosítót, amivel a metódusból tudjuk frissíteni az üzeneteket, második paraméternek meg a feladat azonosítóját. A „FMS().getTaskMessages()” metódussal lekérjük a feladat üzeneteit, ami visszatér egy JSON vállazzal, amit a „TaskMessage.fromJson()” konstruktőrrel feldolgozunk.

```
await FMS().getTaskMessages(taskList[index].company_id, taskList[index].task_id);
```

A kiválasztott feladat oldalon az üzenet küldése gombra koppintva ellenőrizzük, hogy a felhasználó írt-e be szöveget.

```
if (message.text.isNotEmpty) { }
```

Amennyiben írt be szöveget, meghívjuk a „taskAddMessage()” metódust, aminek átadjuk a vállalat, feladat azonosítóját és az üzenetet.

```
await FMS().taskAddMessage(task.company_id, task.task_id, message.text);
```

Amint elküldtük az üzenetet, kitöröljük a szövegdobozból a szöveget, és lekérjük újra az üzeneteket a „getTaskMessages()” metódus használatával.

```
message.text = "";
```

```
await getTaskMessages();
```

Státusz módosításnál a „BuildAlertDialog().questionAlert()” metódussal megjelenítünk felugró ablakban egy kérdést, a metódus visszatérési értékének egy szöveget várunk, ha a felhasználó az „Igen”-re koppintott, akkor „true” karakterláncú értékkel tér vissza a metódus, miután a visszatérési érték ellenőrizve lett, megjelenítünk egy szövegdobozt felugró menüben, ahová a felhasználó a státusz módosítási üzenetet írhatja. Az üzenet hozzáadása után a „Rendben” gombra koppintva meghívjuk az „FMS().changeTaskStatus()” metódust, amivel megtörténik a státuszváltás.

```
FMS().changeTaskStatus(task.company_id, task.task_id, task.task_completed == 1 ? 0 :
  1, message: message);
```

3.3.7 Adminisztráció

Az „administration.dart” fájl a kiválasztott vállalat adminisztrációs részét kezeli.

Az „Új FMS felhasználó” oldalon a „Létrehozás” gombra koppintva ellenőrizzük, hogy minden kötelező szövegdoboz ki lett-e töltve a „form.currentState!.validate()” metódussal. Ha ennek a visszatérési értéke „true”, akkor meghívjuk az „FMS().createUser()”-t. Sikeres felhasználó-létrehozás esetén az API-tól visszakapott jelszót a vágólapra tudjuk illeszteni a „Clipboard” osztály „setData()” metódusával.

```
Clipboard.setData(ClipboardData(text: data["result"]["password"]));
```

A „Meglévő FMS felhasználó” menüben a QR kód ikonra koppintva meghívjuk a „MobileScannerController()” osztályt. Ezzel az osztály segítségével a „MobileScanner” widget segítségével megjelenítjük az eszköz kamerájának a képét. Amint az olvasó talál egy QR kódot, ami „fms” karakterlánc kezdetű, visszatérünk a talált QR kód adatával az előző képernyőre.

```
if (barcodes[0].rawValue!.startsWith("fms")) {  
    cameraController.stop();  
    Vibration.vibrate();  
    Navigator.of(context).pop(barcodes[0].rawValue);  
}
```

A képernyőre való visszatérés után meghívjuk a „searchUser()” metódust, amiben ellenőrizzük a biztonsági kulcs létezését a „FMS().existingUser()” metódussal, amiben paraméterként a vállalat azonosítóját, és a biztonsági kulcsot adjuk át.

A „Vállati munkacsoportok” oldalon a felhasználó a munkacsoportokat tudja kezelni. Az oldal megjelenésekor egy listába betöljük az eddig létrehozott munkacsoportokat, a vállalathoz hozzárendelt aktív felhasználókat. A munkacsoport választásánál lekérjük a munkacsoporthoz hozzáadott felhasználókat, és megjelenítjük egy listában az oldal alján, a törléshez ki kell jelölni a felhasználókat, akiket el szeretnénk távolítani, majd a „Törés” gombra koppintva végigmegyünk a listán, akik törlésre fognak kerülni.

```
Future.wait(taskGroupUsers.where((element) => element.selected).map((input) async {  
    await FMS().removeUserFromGroup(taskGroup.company_id, taskGroup.taskgroup_id,  
    input.user_id)}));
```

3.3.8 Felhasználói adatok

A „user.dart” oldal használatával a felhasználó módosítani tudja a személyes adatait. Az oldal betöltésekor egy listába letároljuk az országokat, majd lekérjük a felhasználó adatait. Az „Adatok módosítása” gombra koppintva megváltoztatjuk az

„isEditing” logikai változó értékét „true”-ra, ilyenkor a megnyomott gomb helyett 2 gomb jelenik meg, amelyiknél az „Elvetés”-re koppintunk, akkor visszaállítjuk az „isEditing”-et „false”-ra. A „Mentés” gombra koppintva meghívjuk a „saveChanges()” metódust.

3.3.9 Fiók

Az „account.dart” oldalon a fiókhöz kapcsolódó beállításokat és adatokat tudjuk megnézni és módosítani.

A jelszó módosításnál a „Mentés” gombra koppintáskor az app ellenőrzi, hogy van-e engedélyezve biometrikus azonosítás, ha igen, akkor csak sikeres azonosítás után indul el a jelszómódosítás.

A „Biztonság” menüben a jelszó mező ellenőrzését a widget fában ellenőrizve történik meg, amíg a jelszó mező üres, addig nem jelennek meg a „Biztonsági kulcs generálás” és „Fiók törlése” gombok.

```
passwordControl.text.trim().isEmpty ? ElevatedButton(...) : const SizedBox(),
```

3.3.10 Beállítások

A „settings.dart” fájl segítségével a felhasználó a biometrikus azonosítás beállítását tudja változtatni, és frissítések manuális keresésére is van lehetőség. A „Frissítések keresése” gombra koppintva ellenőrizzük a jelenlegi alkalmazás verziószámát.

```
PackageInfo packageInfo = await PackageInfo.fromPlatform();  
buildNumber = packageInfo.buildNumber;
```

A „buildNumber”-t összevetjük az API-ról lekért verziószámmal, és ha a verziószám nem egyezik, akkor megjelenítünk egy felugró ablakot a frissítéssel kapcsolatban.

```
await FMS().checkForUpdates(context, true);
```

3.3.11 Kijelentkezés

A menüoszlopban megjelenő „Kijelentkezés” gombra koppintva egy felugró ablakban megkérdezzük a felhasználót, hogy biztosan ki szeretne-e jelentkezni, ezt a „BuildAlertDialog().questionAlert()” metódussal tesszük meg. Ha a felhasználó az „Igen” gombot válassza, meghívjuk az „FMS().logout()” metódust, amiben a „box.erase()” parancssal tudjuk törölni a tárolót, amiben a felhasználói adatok voltak. Ezután átirányítjuk a felhasználót a bejelentkezési oldalra.

3.3.12 Utils mappa

Ebben a mappában a lekérések kerültek programozásra. Ezek a programkódok az újból felhasználhatóság miatt kerültek ide.

Az „FMS.dart” fájl a program működtetése szempontjából legfontosabb programkódokat tartalmazza. Ugyanis ez a fájl végzi az összes lekérdezést a API felé, dolgozza fel a frissítések letöltését, ellenőrzi a biometrikus azonosítást.

A „login()” metódus felelős a bejelentkezés lebonyolításáért. Először lekéri az eszköz információit, amelyről a bejelentkezési kísérlet történik.

```
String device = await getDeviceInfo();
final ipv4 = await Ipyfy.ipv4();
final ipv6 = await Ipyfy.ipv6();
```

Majd pedig egy „dio.post()” metódussal elküldi a bejelentkezési adatokat az API-nak.

```
var dio = Dio();
var formData = FormData.fromMap({
    'username': username,
    'password': EncryptText.encrypt(password.toString().isEmpty ? password : "nincs"),
    'permission': permission,
    'company': 'null',
    'platform': "Mobile",
    'ipv4': ipv4,
    'ipv6': ipv6 == ipv4 ? null : ipv6,
    'webinfos': device
});
```

Az API-ról érkező választ feldolgozza, majd ha sikeres a bejelentkezés, akkor visszaadja a kapott választ, oda, ahonnan meg lett hívva a metódus.

```
var response = await dio.post('${Variables.apiURL}/auth/permcheck', data:
formData).timeout(const Duration(seconds: 20));
Map<String, dynamic> data = json.decode(json.encode(response.data));
return data;
```

Az „encrypt_text.dart” fájl végzi a jelszavak titkosítását. Az „encrypt” függvénynek egy szöveges paramétert kell átadni, melyet titkosítani szeretnénk.

```
static String encrypt(String args) {
```

Létrehozunk egy Key típusú változót, amelyet a late kulcsszóval jelölünk meg, mivel az értékét csak később fogjuk beállítani.

```
late Key key;
```

Ellenőrzük, hogy a paraméter hossza legalább 16 karakter-e és ha nem, akkor kiegészítjük az „X” karakterekkel, hogy elérje a 16 karaktert.

```
if (args.length < 16) {
    String encodedKey = args;
```

```
for (int i = 0; i < 16 - args.length; i++) {
    encodedKey += "X";
}
key = Key.fromUtf8(encodedKey);
} else {
    key = Key.fromUtf8(args.substring(0, 16));
}
```

Létrehozunk egy inicializáló vektort az IV konstruktor segítségével, amelynek az „encodedIv” változó az értéket adjuk. Az IV egy olyan osztály, amely az AES titkosítás során használt inicializáló vektorokat reprezentálja.

```
final iv = IV(encodedIv);
```

Az „Encrypter” osztály egy példányát létrehozzuk, amely az AES titkosítást végzi el az előzőleg létrehozott kulccsal és inicializáló vektorral. Az „AESMode.ecb” paraméter a titkosítás módját határozza meg. A „PKCS7” padding paraméter pedig a titkosítás során alkalmazott típusát határozza meg.

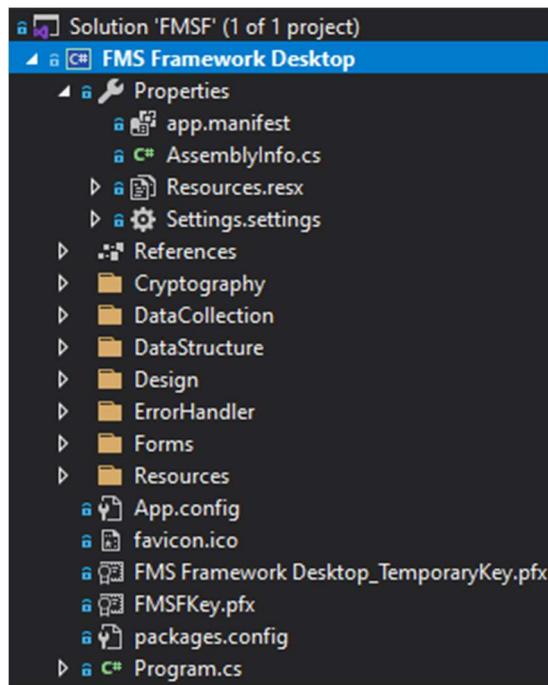
```
final encrypter = Encrypter(AES(key, mode: AESMode.ecb, padding: 'PKCS7'));
final encrypted = encrypter.encrypt(args, iv: iv);
```

A függvény visszatérési értéke az eredeti paraméter titkosított változata, amelyet a base64 metódus segítségével kódolunk.

```
return encrypted.base64; }
```

3.4 Asztali alkalmazás

Az asztali alkalmazás C# nyelven íródott, a .NET Framework 4.7.2-es keretrendszer felhasználásával, „Windows Forms” használatával. Az alkalmazás futtatásához a keretrendszer telepítése szükséges, amely az alkalmazás telepítésekor, első lépésként ellenőrzésre és telepítésre kerül. A projekt neve az „FMS Framework Desktop” nevet kapta, az FMSF „solution”-ben. A legfőbb cél a programkód egyszerű átláthatósága és az objektumorientált programozás követése. A rendszerezés miatt létrehozásra került több könyvtár, amely különböző olyan objektumokat és osztályokat tartalmaz, amely a működés közben elengedhetetlen. A névterek létrehozása könyvtárakként került létrehozásra.



91) ábra: Asztali alkalmazás könyvtárszerkezete Visual Studio 2019-ben.

3.4.1 Cryptography mappa

Az adatok titkosításáért felelős könyvtárszerkezet, amelyben jelenleg egy darab olyan fájl található, amely a kriptográfiai folyamatokért felel.

Az „Encrypt.cs” állományban létrehozott „Encrypt” osztály felelős a jelszavak megfelelő titkosításáért, amely jelen esetben a nyers szöveget AES-128 ECB titkosítással lát el a „RijndaelManaged” használatával. Hárrom „property” található meg az osztályban, amely tartalmazza a szükséges adatokat. Az első a „Text”, amely a titkosítandó szöveget tartalmazza, a második a „SecretKey”, amely a titkosításban használandó biztonsági kulcsot tartalmazza, amely a visszafejtéshez szükséges, illetve a titkosított szöveg, amely az „EncryptedText” tulajdonság tartalmazza a szöveg titkosított változatát.

Itt a jelszó titkosítása történik meg első lépében, amelynél a biztonsági kulcs egy dinamikusan előállított szöveg. A konstruktorban az eredeti szöveg kerül megvizsgálásra, amely során, ha 16 karakter alatt van, akkor a „StringBuilder” használatával kiegészítésre kerül 16 karakter hosszban. Ellenkező esetben a titkosítandó szöveg első 16 karaktere kerül mentésre a „SecretKey”-ben, majd a „Crypter” függvény segítségével, titkosításra, majd eltárolásra kerül az utolsó „property”-ben.

```
public Encrypt(string text)
{
    Text = text;
    if (Text.Length < 16)
    {
```

```

StringBuilder build = new StringBuilder(Text);
for (int i = 0; i < 16 - Text.Length; i++)
{ build.Append("X");}
SecretKey = build.ToString();
}
else { SecretKey = Text.Substring(0, 16); }
EncryptedText = Crypter(text, SecretKey);
}

```

A dinamikusan változó biztonsági kulcs mellett azért döntöttünk, mert így minden jelszó más kulccsal kerül titkosításra, a visszafejtéshez viszont minden esetben szükség van a kulcsra. Tehát csak az tudja visszafejteni a jelszót, aki tudja is a saját jelszavát, viszont más jelszavát nem tudja visszafejteni, mivel az összes jelszó ugyanazon elv alapján került titkosításra.. Ettől függetlenül a jelszavak további titkosítási eljárásban is átesnek, amelyet már nem az alkalmazás végez, hanem az API.

3.4.2 DataCollection mappa

A különböző adatok gyűjtésére használt osztályok kaptak ebben a könyvtárban helyet. Az alkalmazás használata közben legtöbbször használt osztály az „APIRequest”, amely az API-val történő POST és GET módszerek kommunikációját segíti elő. Jelenleg 88 alkalommal került példányosításra ez az osztály. A „client” olvasható tulajdonság fog kommunikálni a közös interfésszel. A „PostMethod” igaz értéke esetén POST kérést fog tartalmazni a kérés, hamis esetén GET kérés kerül előkészítésre. A „BaseUrl” tartalmazza az API elérési hivatkozását, amelyet a „ModuleURL” egészít ki egy teljes URL hivatkozássá. A „Values” nevű „Dictionary” (szótár) adatszerkezet tartalmazza a kulcsérték-párokat, amelyek a kéréshez szükségesek. Az „EventLog” szótár tartalmazza a lekérdezés úgynevezett életútját. Amennyiben a HTTP kérés során a státuszkód 200-as értéket kap POST módszernél, akkor a „SuccessRequest” értéke igaz. Az egyéni szabványt, mint API válaszokat tekintve, mindegyik tartalmazza a „success” kulcsot, amelynek értékét az API határozza meg. A „Result” JSON objektumban a „JObject” alapján kerül eltárolásra. A fejlesztés közben a későbbiekben hozzáadásra került a „StatusCode”, amely a kérés http státuszát adja vissza. Sikeres kérés esetén „HTTP OK” 200, „HTTP Not Found” 404, „HTTP Internal Server Error” esetén 500 hibakódra utal.

Az osztálynak két konstruktora van, amelyből az egyiknél a paraméterek között kötelező megadni az API elérési útvonalat, és modult a lekérdezés típusával együtt. Mivel az összes lekérdezés az „api.felx.hu” hivatkozásban kommunikál, ezért eltérő útvonalra nem volt szükségem, így a második konstruktort használtam a lekérdezésekben, ahol elegendő megadni a lekérdezés típusát és a modul elérését. Ilyenkor az alkalmazás

beállításaiból olvasható ki az API elérési hivatkozása, amely a projekt indításakor beállításra került, és könnyedén módosítható, akár programon belül is.

A „StartRequest” aszinkron metódus meghívásával történik, amelynél, ha a „PostMethod” igaz, akkor a „PostRequestAsync” aszinkron metódus meghívása történik, ellenkező esetben „GetRequestAsync”. A lekérdezett HTTP tartalom JSON formátumba való átalakítása után eltárolásra kerül a „Result” tulajdonságban. Itt kerül beállításra a két sikereséget jelző logikai változó is. Mindkét lekérdezés tartalmaz kivételkezelést (hibakezelést), amelynél a logikai értékek automatikusan hamisra kerülnek beállításra és az „EventLog” segítségével a keletkező hiba mentésre kerül.

A „ComputerData.cs” tartalmazza a számítógépes hardver és szoftverinformációkat gyűjtő osztályt. Itt kerül eltárolásra a felhasználó operációs rendszere, architektúrája, számítógép neve, NetBIOS neve, bejelentkezett Windows felhasználó neve, amely az „Enviroment” segítségével kerül meghatározásra.

```
OS = Convert.ToString(Environment.OSVersion);
OSx64 = Environment.Is64BitOperatingSystem;
UserDomain = Environment.UserDomainName;
UserName = Environment.UserName;
ComputerName = Environment.MachineName;
AppVersion = Assembly.GetExecutingAssembly().GetName().Version.ToString();
```

A „ComputerData” tartalmaz egy „ToString” metódus felülírást, amelyben az adatok szöveges megjelenítése kerül beállításra egy szöveges karakterlánc formátumban. A beállított szöveg fog a későbbiekben az eszközök platform információinak megjelenítése közben megjelenni.

```
public override string ToString()
{
    return $"OS: {OS} {(OSx64 ? "64 bit" : "32 bit")}" +
        $"{Environment.NewLine}Logged User: {UserDomain}\\"{UserName}" +
        $"{Environment.NewLine}Computer Name: {ComputerName}" +
        $"{Environment.NewLine}FMS Framework Desktop Version: {AppVersion}";}
```

Alapvető hálózati adatok és információk lekérdezése a „NetworkData.cs” tartalmaz, amelyben a publikus IPv4, IPv6 és a számítógép hálózati adapterjének információs listáját lehet aktualizálni a konstruktorban megadott logikai értékek segítségével.

Az IP címek lekérdezését az „icanhazip.com” segítségével a „ WebClient” segítségével kerül lekérdezésre. Hibakezelés a lekérdezéseknel is található, amely esetében az IP cím tulajdonságértékét nullára állítja be.

```
string externalIpString = new WebClient()
```

```

    .DownloadString("http://ipv4.icahnazip.com/")
    .Replace("\r\n", "").Replace("\n", "").Trim();
var externalIp = IPAddress.Parse(externalIpString);
PublicIPv4 = externalIp.ToString();

```

A hálózati adapterek információi az „OperationalStatus” vagyis a működési állapot alapján kerülnek szűrésre, mivel csak az aktív adapterek adatait adjuk hozzá a „NetAdapters” nevű listához. A szűrés a ciklusban elhelyezett lambda kifejezéssel került megvalósításra.

```

NetworkInterface[] adapters = NetworkInterface.GetAllNetworkInterfaces();
foreach (NetworkInterface adapter in adapters.Where(a => a.OperationalStatus ==
OperationalStatus.Up))
{
    NetAdapters.Add(adapter);
}

```

A felhasználók jogosultságainak ellenőrzését a „PermissionCheck” biztosítja. Az SSO által hitelesített felhasználó aktuális jogosultságához való hozzáféréseit ellenőrző API lekérdezést tartalmazza. Az osztályban eltárolásra került a felhasználó munkamenetéhez tartozó token, a vizsgálni kívánt szerepkör, illetve a vállalat azonosítója, amennyiben vállalatspecifikus jogosultság vizsgálatára van szükség. Az utóbbi a két konstruktur meghívásától függ, amennyiben tartalmazza a vállalat azonosítóját paraméterként átadva, akkor a „Spec” logikai érték igaz lesz, amennyiben viszont nem vállalathoz köthető szerepkör keresésére van szükség, akkor a vállalat azonosítóját kihagyva a „Spec” értéke hamis. A lekérdezés eredményének üzenete a „Message” változóban kerül eltárolásra. Az SSO aszinkron függvény meghívásakor az „APIRequest” osztályban található API lekérdezés kerül végrehajtásra. A függvény visszatérési értéke a jogosultságot tartalmazó logikai változó. Ha igaz, akkor van valamelyik szerepkör által biztosított jogosultsága. A letárolt üzenet kerül még beállításra, mivel a későbbiekben a megjelenő, felugró ablakokban ezek kerülnek megjelenítésre.

A token érvényességét ellenőrző POST lekérdezés a „TokenCheck” osztályban található meg, amelyben a beállítások közé lementett bejelentkezés utáni token kerül vizsgálatra. A konstruktor betölti a tokent a beállításokból, az ellenőrzés értéke hamis, az utolsó vizsgálat időpontja nulla, érvényessége hamis. Az ellenőrzést a „Check” aszinkron metódus segítségével tesztelhetjük. Sikeres lekérdezést követően a „Valid”, vagyis az érvényesség értéke beállításra kerül az API válaszadása által.

Egy API lekérdezés a következőképpen történik:

```

APIRequest checkToken = new APIRequest(true, "auth/tokencheck");

```

```

checkToken.Values = new Dictionary<string, string>
{
    { "token", Token }
};
await checkToken.StartRequest();
LastCheckTime = DateTime.Now;
if (checkToken.SuccessRequest) { Valid = checkToken.SuccessRequestByAPI; }
else { Valid = false; }

```

3.4.3 DataStructure mappa

Az adatok optimális tárolásához szükség volt több osztály létrehozására. Az osztálypéldányokból összeállított, egyszerűen kezelhető listák tárolják a futó lekérdezések eredményeit.

A „Company” osztály tartalmazza a vállalat azonosítóját, nevét, adószámát, regisztrációs vagy nyilvántartási számát, logójának hivatkozását, létrehozásának idejét és utolsó módosításának dátumát. Kétféle konstruktur került létrehozásra, az egyik csak a vállalat azonosítóját és nevét dolgozza fel, a másik az összes adat kezelését elvégzi.

A „Country” tartalmazza a világ országainak legfontosabb adatainak tárolását, többek között létrehozásra került az ország azonosítója, neve, teljes neve, kódja, fővárosa, melléknevei, pénzneme, váltópénze. Ezen kívül a „NameCountryCode” nevű tulajdonság az ország nevét és kódját adja vissza értékül egy meghatározott formában. Például: Magyarország [HU].

```
public string NameCountryCode { get { return $"{{Name}} [{CountryCode}]"; } }
```

Az „IPInfo” osztály a felhasználói munkameneteknél felhasznált és lekérdezett IP információkat tárolja, többek között több tulajdonság is létrehozásra került az egyszerűbb adatfeldolgozás érdekében. A konstruktur segítségével megadható az IP cím, hostnév, ország, megye, város, irányítószám, időzóna, koordináták, internetszolgáltató adatai. A „Location” tulajdonság értéke a három helyet tartalmazó „property”-ből áll.

```
Location = $"{{Country}} / {{Region}} / {{City}}";
```

A „Role” segítségével tárolhatóak el egy adott hozzáférés és szerepkör adatai. Hárrom különböző konstruktur segítségével tölthetjük fel adatokkal a tulajdonságokat. A „CompanyRoleName” a vállalat nevétől függően kerül beállításra. Amennyiben a vállalat neve üres (hossza 0), akkor csak a szerepkör neve kerül beállításra, ellenkező esetben a vállalat neve és a szerepkör neve egy szóközzel kerül elválasztásra az olvasható megjelenítés érdekében. A különböző paraméterekkel ellátott konstruktorkok általában az API válaszokhoz igazodva kerültek kialakításra.

```
if (companyName.Length == 0) { CompanyRoleName = roleName; }
else { CompanyRoleName = $"{{companyName}} {{roleName}}"; }
```

A „Session” teszi lehetővé a munkamenetek információinak tárolását. A munkamenet és a felhasználó azonosítója mellett számos információ tárolása oldható meg. A korábban megírt „IPInfo” osztály felhasználásra került egy tulajdonságként megadva, ugyanis a munkamenetek tartalmazhatnak olyan információkat, amely rendelkezik IP információval. Ha van létező kiegészítő információ, akkor a „hasIPInfo” értéke igaz, és az egyik konstruktur utolsó paramétereként kötelező átadni az „IPInfo” osztályból példányosított objektumot.

A „TaskGroup” osztály segítségével készíthető olyan lista, amely tartalmazza a feladatokhoz rendelt munkacsoportokat, a lekérdezés típusától függően a munkacsoporthoz tartozó vállalat adatait, amely a „Company” osztály felhasználásával kerülhet definiálásra.

A „TaskList” osztály tartalmazza a feladatokat, mivel a „Task” osztály foglalt az aszinkron és egyéb műveletek kezelése miatt. Ilyen például a feladat azonosítója, munkacsoportja, vállalata, rögzítő felhasználója, határideje. Az egyik konstruktorban található ternális operátor segítségével a határidő, létrehozás és utolsó módosítás ideje üres megadása esetén a tulajdonságok értéke egy kötőjel lesz. A „CompletedImage” egy „Picture” típusú tulajdonság, amiben a „Resources”, másnéven források mappában található kép kerül beállításra értékként, a „Completed” logikai változótól függően. Ha az érték igaz, akkor egy zöld pipa kerül beállításra, ellenkező esetben a piros x jelenik meg a tulajdonság megjelenítésekor.

```
Deadline = deadline == "" ? "-" : deadline;
Created = created == "" ? "-" : created;
Modified = modified == "" ? "-" : modified;
CompletedImage = (Completed ? Resources.check : Resources.x);
```

A „TaskMessage” osztály segítségével a feladatokhoz rendelt üzeneteket lehet eltárolni. Több olyan „property” kerül használatba, amely meglévő publikus osztályok példányosításával használható.

A „User” osztály segítségével egyszerűen tárolhatjuk el a felhasználó azonosítóját és nevét, de akár az összes személyes adatát. A háromfélé konstruktur a lekérdezések eredményeihez igazítva használható.

3.4.4 Design mappa

A „Design” mappa tartalmazza az „AppColor” osztályt, amely az összes „Form” alapvető kelléke. Az osztály olyan tulajdonságokat tartalmaz, amely azokat a

sablonszíneket adja vissza, amelyeket a web és mobil alkalmazásokban is használatban vannak.

```
public Color DefaultBlack { get { return Color.FromArgb(45, 48, 71); } }
public Color Accept { get { return Color.FromArgb(29, 175, 104); } }
public Color Cancel { get { return Color.FromArgb(164, 48, 63); } }
public Color Warning { get { return Color.FromArgb(252, 122, 30); } }
```

Az „AppSettings” tartalmazza azoknak a képeknek a forrását, amelyek felhasználásra és megjelenítésre kerülnek. A fejlesztés során csak pár helyen kapott helyet ez a lehetőség.

```
public Bitmap LogoLarge { get { return Properties.Resources.logo; } }
```

3.4.5 ErrorHandler mappa

Az „ErrorLog” osztály jelenleg 93 alkalommal került felhasználásra a program kódrészleteiben, hiszen a különböző tulajdonságok beállításával és a „CreateErrorLog” metódus használatával készíthető olyan hibaüzenet, amely a későbbiekben szükséges lehet akár fejlesztések folyamán, vagy csak a szoftver hibára futásakor történő naplázás céljára. A „SaveErrorLog” metódus a felhasználó ideiglenes könyvtárába („temp”) létrehozásra kerül az FMSF mappa, amelybe a hibaüzenetek mentése kerül szöveges fájl formátumban.

```
Directory.CreateDirectory($"@'{LogPath}\FMSF");
using (StreamWriter sw = new StreamWriter(new
FileStream($"@'{LogPath}\FMSF\FMSF_ErrorLog_{DefDate.ToString("yyyyMMdd_
HHmmss")}.txt", FileMode.Create), Encoding.UTF8))
{
    sw.WriteLine(HeaderText);
}
```

A hibanaplót tartalmazó kiírásából a „Title”, a hibában keletkezett alkalmazásablak címét, a „Message” hibaüzenetet, a „Source” a hiba forrását adja meg. A hiba adatait és a számítógép paramétereit összegyűjtő „ComputerData” osztály segítségével összeállításra kerül az állomány tartalma. A felhasználó ideiglenes könyvtárának meghatározása mellett a „CreateErrorLog” metódus beállítja az „HeaderText” tulajdonságának értékeként az összeállított fájl.

```
public ErrorLog(string title, string message, string source) {
    Title = title;
    Message = message;
    Source = source;
    DefDate = DateTime.Now;
    LogPath = Path.GetTempPath();
    PCInfo = new ComputerData();
    CreateErrorLog();
}
```

3.4.6 Bejelentkezés

A „Program.cs” fájlban meghatározott alkalmazás a „FormLogin” megjelenítésével indul, ahol a felhasználónév és jelszó megadása után lehet továbblépni a rendszerben.

A munkaablak keretének stílusa üresre lett állítva, a színek és a logó megjelenítését a „ColorSettings” és a „LogoSettings” metódus végzi el. Az első metódus megtalálható az összes munkaablakban, a megfelelő színek beállítása miatt.

Az ablak kezdőpozíciója a képernyő közepe, tehát az ablak az indítást követően középen jelenik meg. Az ablak minimum és maximum mérete mindenhol egyformára lett beállítva, így az elrejtett vezérlőgombok mellett a billentyűkombinációk segítségével sem lehet teljes képernyőssé állítani a bejelentkező felületet.

A jelszó mezőn a „UseSystemPasswordChar” tulajdonságnak igaz értéket adva a jelszó beviteli mezőben a karakterek helyett fekete telített körök jelennek meg az olvashatóság megakadályozása céljából.

A bezárás gombra kattintva az alkalmazás futtatása megáll, a bejelentkező gomb megnyomására elsősorban egy egyéni hibakezelési folyamat fut le, amelyben elsőként ellenőrzésre kerül, hogy a felhasználónév és jelszó páros megadásra került-e. A jogi nyilatkozat lekérése a „GetNoticeInfoAsync” aszinkron metódus segítségével történik.

Amennyiben a felugró ablakban az igen lehetőséget választja a felhasználó, akkor folytatódik a felhasználó azonosítása, a „UserAuthenticationAsync” metódus segítségével. Ha a nyilatkozatot a felhasználó elutasította, akkor a jelszó beviteli mező kiürítésre kerül, és az kerül újra fókuszba.

A hibakezelésben előfordulhat, hogy szemantikai hiba található a rendszerben, illesmi lehet valamelyik beviteli mező kitöltésének hiánya, vagy egyéb olyan hiba, amely nem gátolja a program futtatását. Amennyiben, ha a hiba típusa nem „Exception”, akkor a hiba elmentésre kerül, és egy hibaüzenet jelenik meg a felhasználó képernyőjén.

```
if (ex.GetType() != Type.GetType("System.Exception"))
{
    ErrorLog loginError = new ErrorLog(this.Text, ex.Message, "Bejelentkezés
tevékenység");
    loginError.LogError();
    MessageBox.Show("Hiba történt az alkalmazás futtatása közben!", this.Text,
MessageBoxButtons.OK, MessageBoxIcon.Error);
}
else { MessageBox.Show(ex.Message, this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Error); }
```

Az azonosítás a hálózati IPv4 és IPv6 címek lekérdezésével történik a „NetworkData” használatával, majd az „auth/permcheck” API lekérdezésének

összeállítása következik. Az értékek megadása után a POST lekérdezést elindítva a „StartRequest” aszinkron metódussal vár az alkalmazás az API válaszára. Amennyiben a lekérdezés és a hitelesítés is sikeres, akkor a token elmentése kerül ideiglenesen a beállításokba és a bejelentkező ablak eltüntetésre kerül, végül megjelenik a „FormMain”, azaz a kezelőpult.

```
Settings.Default.token = Convert.ToString(userAuth.Result["result"]["token"]);
Settings.Default.Save();
FormMain frmMain = new FormMain();
this.Hide();
frmMain.ShowDialog();
```

Sikertelen hitelesítés esetén a jelszó beviteli mező kiürítésre kerül, majd egy hibaüzenet dobása következik, amelyben az API eseménykezelő naplójában időrendi sorrendbe rendezett utolsó érték lesz a hibaüzenet az egyéni hibakezelésnél.

```
tbPassword.Text = string.Empty;
tbPassword.Focus();
throw new Exception(userAuth.EventLog.OrderBy(x => x.Key).Last().Value);
```

Amennyiben a „SuccessRequest” értéke is hamis, akkor nem megfelelő a kommunikáció az interfésszel, ekkor újabb hiba dobása következik a „Hiba történt a szerverrel történő kommunikáció közben!” üzenettel.

3.4.7 Kezelőpult

A bejelentkezést követően beállításra kerül az alkalmazás közepén lévő háttérkép az „ImageSettings” metódussal, illetve a színek a „ColorSettings” metódus segítségével. A token beolvasásra kerül a beállításokból, majd eltárolódik a Token nevű „property”-be.

A funkcióbillentyűk a „Form” eseményei között lévő „KeyDown” esemény használatával kerül összerendelésre. Fontos, hogy a támogatás csak akkor működik, ha a „KeyPreview” értéke igaz. Többágú szelekcióban a „KeyEventArgs” billentyűkódjai tartalmazzák a gyorsgombokat. Amennyiben a billentyűlenyomásnál a többágú szelekció egyik ága tartalmazza a lenyomott gombot, akkor a művelet végrehajtásra kerül, általában a „Click” események kerülnek meghívásra.

```
private void FormMain_KeyDown(object sender, KeyEventArgs e)
{
    switch (e.KeyCode)
    {
        case Keys.Escape:
            //KIJELENTKEZÉS
            msiLogout_Click(sender, e);
            break;
        case Keys.F2:
            //VÁLLALATVÁLASZTÁS
```

```

        msiCompanySelector_Click(sender, e);
        break;
    case Keys.F3:
        //SAJÁT ADATOK
        msiPersonalUser_Click(sender, e);
        break;
    }
}

```

Az alkalmazás bezárása előtt a beállításokban eltárolt token értéke az alkalmazás bezárása előtt törlésre kerül.

```

private void FormMain_FormClosing(object sender, FormClosingEventArgs e)
{
    Settings.Default.token = "X";
    Settings.Default.Save();
    Application.Exit();
}

```

A kiválasztott munkaablak megnyitása előtt szükség van a jogosultság ellenőrzésére, amely az SSO jogosultság ellenőrző rendszer segítségével történik. Példányosításra kerül a „PermissionCheck” osztályból, illetve paraméterként átadásra kerül a token, illetve a jogosultság neve. Mivel nem vállalatspecifikus szerepkör jogosultságát szeretnénk ellenőrizni, így a vállalat azonosítójának megadása nem szükséges. Jelen esetben a felhasználó a saját jogosultságait szeretné megtekinteni, amelyhez elegendő a személyes jogosultsági szint. Ha a „permCheck.SSO” függvény visszatérési értéke igaz, akkor megnyitásra kerül a jogosultságok munkaablak, a token paraméter átadásával.

```

private async void msiPermissions_Click(object sender, EventArgs e)
{
    PermissionCheck permCheck = new PermissionCheck(Token,
"fms_framework_personal");
    if (await permCheck.SSO())
    {
        FormUserPermissions frmUserPermissions = new
FormUserPermissions(Token);
        frmUserPermissions.ShowDialog();
    }
    else
    {
        MessageBox.Show(permCheck.Message, this.Text, MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

```

A vállalat kiválasztásánál a „selectedCompany” értéke lesz a kiválasztott vállalat példánya. A vállalatspecifikus jogosultságok ellenőrzése után átadásra kerülnek a kiválasztott vállalat adatai.

Ha a felhasználó olyan funkciót szeretne használni, amely vállalatspecifikus, de nincs kiválasztva még az indítás óta vállalat, akkor a gombra kattintáskor lefutó művelet első lépései között ellenőrzésre kerül, hogy az előbb említett tulajdonság értéke nulla-e. A feltétel teljesülése esetén egy hibaüzenet hívja fel a figyelmet a megfelelő vállalat kiválasztására, majd megnyitja a vállalat-választó ablakot és ezáltal megszakítja a további műveletet a „return” használatával.

```
if (selectedCompany == null)
{
    MessageBox.Show("A továbblépéshez szükség van a megfelelő vállalat
kiválasztására!", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Warning);
    msiCompanySelector_Click(sender, e);
    return;
}
```

3.4.8 Vállalat kiválasztása

A megnyíló vállalat kiválasztását biztosító ablakban megjelenik a felhasználó neve, illetve a választható, elérhető vállalatok. Amennyiben nincs aktív választható vállalat, akkor a „Kész” gomb nem érhető el. Ha a felhasználó neve helyén az „N/A” kiírás látható, akkor nem sikerült lekérdezni a munkamenethez tartozó felhasználó adatait.

Először az alkalmazásban beállításra kerülnek a színek, majd a „Reset” metódussal a felhasználó nevek törlésre kerülnek, illetve a vállalatok listáját tartalmazó doboz kiürítésre kerül, a kész gomb pedig eltűnik. Ezek után a „LoggedNameRequest” lekéri a bejelentkezett felhasználó azonosítóját és nevét az „sso/user/minimal” API lekérdezésével, majd az „sso/user/companyaccess” kéréssel hozzáadásra kerül az „allowedCompanies” listához. Amennyiben a lekérdezést az API sikeresnek minősíti, akkor engedélyezi a kész gomb megjelenését. Ezek után feltöltésre kerül a vállalatok nevét tartalmazó doboz, amelynek értéke a vállalat azonosítója.

```
cbCompanies.DataSource = allowedCompanies;
cbCompanies.DisplayMember = "name";
cbCompanies.ValueMember = "id";
```

A kész gombra kattintva ellenőrzésre kerül, hogy a kiválasztott vállalat nem nulla. Az irányítópult „selectedCompany” értéke beállításra kerül a kiválasztott vállalattal. Ezek után az irányítópult munkaablak címe végére hozzáadásra kerül a vállalat neve, majd bezáródik az ablak.

```
frmMain.selectedCompany = (Company)cbCompanies.SelectedItem;
if (frmMain.selectedCompany != null)
{
    frmMain.Text = $"FMS Keretrendszer – Kezelőpult -
{frmMain.selectedCompany.Name}";}
```

```
else { frmMain.Text = $"FMS Keretrendszer - Kezelőpult"; }
this.Close();
```

3.4.9 Vállalat adatai

A vállalat adatainak megnyitása előtt az SSO jogosultság ellenőrzésére volt szükség, sikeres hitelesítést követően az alkalmazás megnyitásra került, ahol a kiválasztott vállalat-azonosítóval és a token alapján az „sso/company/info” API meghívást tartalmazó „DataRequest” aszinkron metódust a „FillData” metódus hívta meg, amely a lekérdezés lefutását követően leellenőrzi, hogy a „fullSelectedCompany” változó értéke nem egyenlő-e nullával. Sikeres ellenőrzés keretében kitölti a megjelenő mezőket, illetve a logót.

```
APIRequest dataReq = new APIRequest(true, "sso/company/info");
dataReq.Values = new Dictionary<string, string>()
{ { "token", token }, { "company", Convert.ToString(selectedCompany.Id) } };
await dataReq.StartRequest();
```

A sikeres lekérdezések nél az objektumok feltöltésére törekszünk az értékekkel történő azonnali mezők kitöltése helyett, mert a későbbiekben szükség lehet, akár bővítés esetén a lekérdezést tartalmazó eredmény adataira.

3.4.10 Vállalati felhasználók kezelése

A menüpont és az SSO ellenőrzést követően a „UserRequest” metódust meghívva lekérdezi a felhasználók adatait az „sso/company/userdata” API-ról, amelynél sikeres lekérdezésnél először ürítésre kerül az ablak megnyitásakor létrehozott „users” lista, majd a lekérdezésben szereplő összes felhasználó adatai hozzáadásra kerülnek a listához. A feltöltést követően meghívásra kerül a „FillGridView” metódus, amely csak akkor állítja be a táblázat adatforrásának a listát, ha a lista elemeinek száma nem 0. Az adatforrásként való alapértelmezett beállítás esetén nem megfelelően jelennek meg az adatok a táblában, ezért az adatforrásnál egy új lista kerül létrehozásra, amelynek értékei egyenlőek lesznek a „users” lista tartalmával. Az adatszámlálót beállítja a megjelenített sorok száma és a lista elemeinek számával perjellel elválasztva.

```
if (users.Count != 0) {
    dgData.DataSource = new List<User>(users);
    lblDataCounter.Text = $"{dgData.Rows.Count} / {users.Count} darab";}
```

A frissítés gombra kattintva, vagy az F5 gyorsgombot használva először elrejtésre kerülnek a gombok, majd meghívásra kerül a „UsersRequest” metódus, ami után újra megjelenítésre kerül a frissítés gomb. Az információk gomb a táblázatban való kiválasztás után jelenik meg újra.

```

private void btnUpdate_Click(object sender, EventArgs e)
{
    btnInfo.Visible = false;
    btnUpdate.Visible = false;
    UsersRequest();
    btnUpdate.Visible = true;
    if (dgData.CurrentCell.Selected)
    {
        btnInfo.Visible = true;
    }
}

```

Az információk gombra kattintva, megnyitásra kerül a „FormSelectedCompanyUser” munkaablak, amelyben a felhasználó személyes adatait, illetve szerepköreit láthatjuk, és kezelhetjük. A munkaablak csak akkor nyílik meg, ha a gomb látható állapotban van. Mivel az elrejtett gomb eseményét meghívta a billentyűkombináció, ezért minden eseménynél megvizsgálásra kerül, hogy a gomb láthatósága igaz-e, mert csak abban az esetben kerülnek végrehajtásra a feltételen belüli lépések.

```

if (btnInfo.Visible)
{
    FormSelectedCompanyUser frmSelectedCompanyUser = new
    FormSelectedCompanyUser(token, selectedCompany,
    users[dgData.CurrentCell.RowIndex]);
    frmSelectedCompanyUser.ShowDialog();
}

```

3.4.11 Kiválasztott vállalati felhasználó kezelése

A paraméterként átadott token, vállalat és kiválasztott felhasználó elmentésre és kiírásra kerül az ablak címsorába. A logó és a színek beállítása után a „FillPersonalData” feltölti a személyes adatok mezőket a megkapott felhasználói objektum értékeivel. Ezek után meghívásra kerül az „UpdateAllRolesInfos” metódus, amely először eltünteti az összes gombot az ablakból, majd a „RequestCompanyRoles” lekéri a vállalati aktív szerepköröket az „sso/company/activeroles” használatával az API-ról, majd lekéri a felhasználó jogosultságait is, az sso/company/useraccesses segítségével. Az adatok a lekérések lefutása végén feltöltik a megfelelő helyekre az adatokat az ablakban, majd engedélyezi a hozzárendelés, törlés és frissítés gombokat.

A frissítés gomb meghívja az „UpdateAllRolesInfos” metódust, ha a gomb nincs eltüntetve az ablakból.

A hozzárendelés gombra kattintva egy felugró ablakban megkérdezi a felhasználótól a rendszer, hogy biztosan szeretné-e a felhasználóhoz a kiválasztott szerepkört. Ha a felugró ablakban az igen lehetőséget választja, akkor meghívásra kerül az „AssugnUserToAccessAPI” metódus, amely lefuttatja az „sso/company/assignuseraccess” API lekérdezést a megfelelő paraméterekkel. Sikeres lekérdezés esetén lefrissíti az ablakban található adatokat.

A törlés gombra kattintva megkérdezi a rendszer a felhasználótól, hogy a táblázatban kijelölt szerepkört szeretné-e eltávolítani a felhasználótól. Amennyiben az igen lehetőséget választja és csak egy szerepkörrel rendelkezik a felhasználó, akkor a rendszer újabb kérdést tesz fel, amiben felhívja a figyelmet, hogyha törli a felhasználótól az utolsó vállalati szerepkört is, akkor az ablak bezárását követően, eltűnik az adminisztrációból a felhasználó és csak biztonsági kulcs generálásával és újra adminisztrálásával lehetséges a visszaállítás. Ha itt is az igen lehetőséget választja a felhasználó akkor meghívásra kerül a „RemoveAccessAPI”, amely lefuttatja az „sso/company/removeuseraccess” lekérdezést. A lekérdezés eredményének üzenetét megjeleníti egy felugró ablakban, majd lefrissíti a felhasználó adatait.

A tiltás vagy engedélyezés gombra kattintva a táblázatban kiválasztott szerepkör hozzáférése módosítható. Tiltás esetén a felugró figyelmeztető ablakot jóváhagyva meghívásra kerül a „ChangeAccessAPI” metódus, amelynek paramétere hamis. Engedélyezés esetében a metódus paramétere igaz. A paraméter a „sso/company/accesschange” API lekérdezésben a POST „status” értéke igaz esetén 1, hamis esetén 0. Az eredményről felugró üzenet ad tájékoztatást, majd sikeres lekérdezést követően lefrissíti az alkalmazásban található hozzáférési adatokat.

```
MessageBox.Show(Convert.ToString(changeAccessReq.Result["message"]), this.Text,
MessageBoxButtons.OK, (changeAccessReq.SuccessRequestByAPI ?
MessageBoxIcon.Information : MessageBoxIcon.Error));
if (changeAccessReq.SuccessRequestByAPI)
{
    UpdateAllRolesInfos();
}
```

3.4.12 Új FMS felhasználó létrehozása

Az SSO jogosultság ellenőrzése után, az ablak megnyílását követően a színek beállításra kerülnek, majd lekérésre kerülnek a választható országok, amelyek a „RequestCountries” meghívásával történik.

A kérés GET formában lesz teljesítve az „info/countries” elérésre. GET értékek megadása nem szükséges, sem hitelesítés, így sikeres lekérdezést követően a „countries” lista felöltésre kerül, majd a „FillCountries” metódus meghívásával beállításra kerül az országválasztó, a listában megjeleníteni kívánt tulajdonságok értékei, az azonosító, illetve kiválasztásra kerül a 138-as elem, amely az adatbázisban 139 azonosítóval Magyarországot jelöli. Az országválasztó csak abban az esetben kerül beállításra, ha az országok lista értéke nem egyenlő nullával.

Lekérdezésre kerülnek a vállalathoz tartozó aktív szerepkörök, mivel új FMS felhasználó létrehozásánál szükség van az első vállalati szerepkörhöz való hozzáférést is biztosítani. Az aktív szerepkörök meghatározása az „sso/company/activeroles” által kerül meghatározásra. Sikeres lekérdezést követően az „activeRoles” lista kiürítésre kerül, majd aktualizálódik az adatbázisból lekért adatokkal. A lekérést követően meghívásra kerül a „FillActiveRoles”, amely csak abban az esetben tölti fel a szerepkört választót, ha a lista elemeinek száma nem egyenlő nullával.

```
private void FillActiveRoles()
{
    if (activeRoles.Count != 0)
    {
        cbFirstRole.DataSource = activeRoles;
        cbFirstRole.DisplayMember = "RoleName";
        cbFirstRole.ValueMember = "Id";
    }
}
```

Az adatok kitöltését követően a létrehozás vagy az F10 billentyű lenyomásával ellenőrzésre kerül, hogy a név, kapcsolattartói e-mail és a szerepkör értéke nem üres-e. Amennyiben megfelel minden a követelményeknek, akkor sorrendben meghívásra kerülnek a „PostCreateNewUser” és a „PostAssignUserToRoles” metódusok.

Az első létrehozza a felhasználót az „sso/user/new” API lekérdezésével. Sikeres válasz esetén az API egy ideiglenes jelszót ad vissza, amely a „Clipboard.SetText” segítségével a vágólapra kerül, felugró üzenet jelenik meg, elmentésre kerül a felhasználó FMS azonosítója a „createdId” változóba, majd kitörli az ablakban található beviteli mezőket és a név kerül fókuszba.

A második metódus az „sso/company/assignuseraccess” segítségével a létrehozott felhasználóhoz hozzárendeli az első vállalati szerepkört, aminek sikerességéről felugró üzenet tájékoztatja a felhasználót.

A vissza gomb vagy az ESC billentyű megnyomására lefut a „this.close” metódus, amely bezárja az ablakot.

A születési dátum kiválasztásánál és megjelenítésénél egyéni dátum formátum kerül beállításra, amely kötőjellel tagolja az év, hónap, napot egymástól.

```
dtBirthDate.Format = DateTimePickerFormat.Custom;  
dtBirthDate.CustomFormat = "yyyy-MM-dd";
```

3.4.13 Regisztrált FMS felhasználó kezelése

Az SSO jogosultság ellenőrzése után az ablak megnyitásra kerül, amelyben a „DefaultSettings” metódust meghívva beállításra kerülnek alap adatok. Az azonosított felhasználó „groupbox” láthatósága hamisra lett állítva, a vállalat mező kitöltésre került a kezelőpultból áthozott vállalat adataival. A biztonsági kulcs „PasswordChar” értéke a „\u2022” lett, amely egy halványabb, kisebb ponttal tölti ki a beviteli mezőt. Hasonló a bejelentkezési rendszernél használittal, csak itt egyéni kitöltő került beállításra, a bejelentkezésnél pedig a rendszer által használt kitöltőt alkalmaztam. Az alapbeállításokon kívül még betöltésre kerültek a színek.

A keresés gombra kattintva levágásra kerülnek a beviteli mezőbe fölöslegesen bevitt szóközök, majd megvizsgálom, hogy a keresés gomb látható-e, illetve a beviteli mező értéke egyenlő-e nullával. Ha a feltétel teljesül, akkor a program egy felugró ablakban tájékoztatja a felhasználót, hogy nem lett megadva biztonsági kulcs, és a fókusz a biztonsági kulcs beviteli mezőre kerül. Ellenkező esetben megvizsgálásra kerül, ha három karakternél hosszabb a bevitt adat, hogy az első három betű az „fms” értéket tartalmazza-e. Amennyiben megfelel a feltétel, akkor meghívásra kerül a „SearchSecretKeyAPI” metódus. Ha a feltételnek nem felel meg az érték, akkor hibaüzenet kerül megjelenítésre a helytelen formátumról, a bevitt adat törlésre kerül, majd a fókusz a beviteli mezőre kerül.

```
if (tbKey.Text.Length >= 3)  
{  
    if (tbKey.Text[0] == 'f' && tbKey.Text[1] == 'm' && tbKey.Text[2] == 's')  
    { SearchSecretKeyAPI(); }  
    else{  
        MessageBox.Show("A biztonsági kulcs helytelen formátumú!", this.Text,  
        MessageBoxButtons.OK, MessageBoxIcon.Error);  
        tbKey.Text = string.Empty;  
        tbKey.Focus();  
    }  
}
```

A meghívás után az „sso/company/existuser” lekérdezés kerül elindításra. Amennyiben az API szerint az aktív biztonsági kulcshoz található felhasználó, akkor az „identifiedUser” értéke először nullára állítódik, majd a felhasználó adataival kerül

feltöltésre. Kiürítésre kerülnek az „activeRoles” és a „companyRoles” listák, és feltöltésre kerülnek az API lekérdezés által, majd a sikeres feltöltést követően meghívásra kerül a „FillIdentifiedFields” és megjelenítésre kerül az API válaszüzenete.

A metódus beállítja a szöveges mezőkben a felhasználó adatait, a „listbox”-ban megjelenítésre kerül a felhasználó összes szerepköre, illetve beállításra kerül a vállalati szerepkörválasztó a vállalat aktív szerepköreivel. Ezek után megjelenítésre kerül a „groupbox”, amivel láthatóvá válnak a személyes adatok.

A hozzárendelés gombra kattintva az „sso/company/assignuseraccess” API lekérdezés fut le, az „AssignUserToAccessAPI” metódus által. A válasz megjelenítésre kerül egy felugró ablakban, majd sikeres lekérdezés esetén az ablak automatikus bezárásra kerül a „this.close” segítségével.

3.4.14 Vállalati munkacsoportok kezelése

A jogosultság ellenőrzését követően megnyílik a munkaablak, amelyben a token és a kiválasztott vállalat kerül elmentésre, majd az ablak címe beállításra kerül. A cím kiegészítésre kerül a kiválasztott vállalat nevével.

```
this.Text = $"FMS Kerrendszer - Munkacsoportok kezelése -  
{selectedCompany.Name}";
```

A csoportok, felhasználók és a kiválasztott csoport felhasználói tárolásához használt listák példányosításra kerülnek, majd a színek és a megjelenő logó beállításra kerül a „ColorSettings” és a „LogoSettings” metódusok segítségével.

A „DisableFields” tartalmazza az összes olyan beviteli mezők engedélyezésének és megjelenésének a letiltását, amely frissítés, adatlekérés közben meghívásra kerül.

A „FillDefaultFields” feltölti a vállalat nevét megjelenítő mezőt az új munkacsoport létrehozásában, illetve az „UpdateAllFields” lekéri és aktualizálja az adatokat.

A metódusban a frissítés idejére letiltásra kerülnek azok a vezérlők, amellyel különböző eseményekben tartalmazó API lekérdezések találhatóak. Erre azért van szükség, mert egy lassabb interneteléréssel rendelkező eszközön a lekérdezések több időt vehetnek igénybe az átlagosnál. A módszer segítségével megakadályozásra kerül az egyidőben megnövekvő aszinkron műveletek elindítása, így nem kerül az API nagyobb terhelés alá.

A munkacsoportokat tartalmazó lista kiürítésre kerül, majd a „GetGroups” segítségével lekérdezésre kerül az „sso/company/group” által az adatok, amelyekkel az előbb kiürített „groups” lista feltöltésre kerül. Ha a lekérdezést követően a lista elemeinek száma nem egyenlő nullával, akkor a munkacsoportokat tartalmazó választók

(„combobox”) adatforrása a frissen aktualizált lista lesz. A „selectedGroupUsers” lista kiürítésre kerül, majd a „GetGroups” által lekérdezésre kerülnek az adminisztrálás alá vont vállalat felhasználói, az „sso/company/activeusers” API kérésével.

A felhasználók adatai az „allusers” listában kerül eltárolásra, majd lekérdezésre kerülnek a kiválasztott munkacsoport felhasználói a „GetTaskUsers” metódusban található „sso/company/taskusers” kérés segítségével, amely a „selectedGroupUsers” listában kerül elmentésre.

Az „UpdateAllFields” metódus még tartalmazza az „allusers” elemeinek számának megvizsgálását, amelyben, ha nem üres a lista, akkor a kiválasztott munkacsoporthoz rendelhető kiválasztható felhasználók választó adatforrására kerül beállítására.

```
groups.Clear();
GetGroups();
if (groups.Count != 0)
{
    cbGroups.DataSource = groups;
    cbGroups.DisplayMember = "Name";
    cbNewGroup.DataSource = groups;
    cbNewGroup.DisplayMember = "Name";
    selectedGroupUsers.Clear();
    GetGroupUsers();
    GetTaskUsers();
}
if (allusers.Count != 0)
{
    cbNewUser.DataSource = allusers;
    cbNewUser.DisplayMember = "Name";
}
```

A frissítő lekérdezések lefutásával újra engedélyezésre kerülnek azok a vezérlők, amelyek letiltásra, illetve elrejtésre kerültek.

Ezek után a kiválasztott munkacsoport tagjait kilistázó megjelenítő eszköz („CheckedListBox”) adatforrásának nullára való beállítása kerül beállításra, majd ellenőrzésre kerül, hogy a kiválasztott munkacsoporthoz hozzárendelt felhasználók értéke nem nulla. Ha a feltétel teljesül, akkor engedélyezésre kerül a felhasználók neveit megjelenítő doboz, majd adatforrásként kerül beállításra.

```
clbUsers.DisplayMember = null;
clbUsers.DataSource = null;
clbUsers.Items.Clear();
if (selectedGroupUsers != null) {
    clbUsers.Enabled = true;
    clbUsers.DataSource = selectedGroupUsers;
    clbUsers.DisplayMember = "Name";
}
```

A metódusban található kiürítések és az adatforrások nullára való beállítására azért van szükség, mert több olyan műveletnél kerülhet meghívásra, amelynél az adatok frissítése a cél, és a helytelen, elavult adatok megjelenítésének elkerülésére az üresre állított adatok API általi frissítése esetén biztosan a legfrissebb és a legújabb adatok kerülnek feldolgozásra.

Az új munkacsoport létrehozásánál található gomb vagy az F9 billentyű lenyomásával a felugró ablakban feltett ellenőrző kérdés, igen lehetőségét választva meghívásra kerül a „PostCreateGroup” metódus, amelyben az „sso/company/addgroup” lekérdezés eredményéről egy felugró ablakban kerül a felhasználó tájékoztatásra. Ezek után az „UpdateAllFields” metódus aktualizálja az adatokat és a beviteli mezők neve üres lesz és rákerül a fókusz. A felugró üzenetek a „MessageBox” használatával kerül megjelenésre, és a visszatérési értékének típusa „DialogResult”, amely a felhasználó által adott választ tartalmazza. Többféle módszerrel lehet megadni a paramétereket, az alkalmazás fejlesztése közben a következő formátumot használtam:

```
MessageBox.Show($"Biztosan szeretné létrehozni a(z) {tbGroupName.Text}  
munkacsoportot?", this.Text, MessageBoxButtons.YesNo, MessageBoxIcon.Question);
```

Az első paraméter az ablak törzsében megjelenített üzenet, majd következik az ablak címe, választható gombok párosítása, illetve az üzenet típusa, amely legfőbb ismertetőjele a rendszer által adott hangok és a megjelenő ikon a bal felső sarokban.

A kiválasztott munkacsoporthoz, a felhasználó hozzárendelését a gombra kattintva vagy az F10 billentyű segítségével hajtható végre. A megjelenő ablak kérdésénél igen lehetőségét választva, meghívásra kerül a „PostAssignUserToGroup” metódus, amely az „sso/company/assigngroupuser” POST kérését hajtja végre, amelynek eredménye felugró ablakban kerül megjelenítésre. Ezek után az „UpdateAllFields” metódus frissíti az összes adatot.

A kiválasztott munkacsoporthoz a felhasználók eltávolítását a jobb oldalt található listából lehet megtenni. A jelölést követően a törlés gombra kattintva a korábban említett módszerrel a „PostRemoveUserFromGroup” metódus kerül meghívásra az adatok frissítését megelőzően.

A metódusban található „foreach” ciklus segítségével az összes kijelölt felhasználón kerül meghívásra az „sso/company/removegroupuser”, amelynek a „user” értékének megkeresése a felhasználók listájából kerülnek megkeresésre lambda használatával. A kiválasztott név alapján a kiválasztott felhasználó FMS azonosítója kerül kilistázásra.

```
allusers.Where(x => x.Name.Equals(Convert.ToString(item))).Select(x => x.Id).First()
```

A frissítés gomb az „UpdateAllFields” metódust hívja meg, amely aktualizálja az adatokat.

3.4.15 Saját fiók adatainak megtekintése és kezelése

A színek beállítását követően a „RequestUserData” metódus az „sso/user/full” lekérését tartalmazza, amelynek sikeres lekérdezését követően a „loggedUser” nevű változóban kerül eltárolásra. Az eltárolást követően meghívásra kerül a „FillFields”, ami kitölti a beviteli mezőket, a bejelentkezett felhasználó adataival. A születési idő csak akkor kerül kitöltésre, ha van megadott dátum az adatbázisban.

A lekérdezést követően a „DisableFields” letiltja a szerkeszthető mezők módosítását, illetve elrejti az országválasztó dobozt, amely az adatok módosítása esetén jelennek meg. Vannak olyan beviteli mezők, amely olyan adatokat tartalmaznak, amelyek módosítása nem lehetséges rendszerszinten sem. Ilyen adatok lehetnek például a felhasználó neme, FMS azonosítója, fiók létrehozásának és utolsó módosításának időpontja. Ezekben túl a felhasználó születési hely egy darab szöveges mezőben kerül megjelenítésre, adatok módosítása viszont egy országválasztóból és a települést tartalmazó szöveges beviteli mezőből áll, így ezek megfelelő elrejtése és módosítása itt történik.

Az adatok módosítása gombra kattintva amennyiben az FMS azonosító értéke nem üres (vagyis a lekérdezés sikeresnek nyilvánítható), akkor meghívásra kerül a „ModifyBtnSettings”, amely beállítja a mezők szerkeszthetőségét és a gombok megjelenítését. A születési idő módosításához külön formátum deklarálására volt szükség, amely a „Format” és a „CustomFormat” tulajdonságok módosításával történik.

```
dtBirthDate.Format = DateTimePickerFormat.Custom;  
dtBirthDate.CustomFormat = "yyyy-MM-dd";
```

A „RequestCountries” lekéri a születési helyként választható országokat az „info/countries” GET lekérdezéssel, amelyek a „countries” listában kerül eltárolásra, majd a „FillCountries” meghívásával beállításra kerül a választó adatforrása, illetve kiválasztásra kerül az adatbázisban szereplő országgal. A külön megjelenítő település is itt kerül kitöltésre.

Az adatok mentésénél lévő üzenet jóváhagyásával az „UpdatePersonalData” frissíti a felhasználói adatokat az „sso/user/update” segítségével. A sikeres végrehajtást követően a frissített adatok lekérdezésre kerülnek a „RequestUserData” segítségével, majd a

„DisableFields” visszaállítja az adatok megjelenését, újra csak olvasható módba kerülnek a beviteli mezők.

A „FormClosing” eseménynél, ha a mentés gomb látható állapotban van, akkor elköpzelhető, hogy a felhasználó által módosított adatok mentése nem valósult még meg, és a felhasználó mentés nélkül szeretné bezárni az ablakot. Ez esetben egy felugró ablak tájékoztatja, hogy biztosan ki szeretne lépni a munkaablakból, mentés nélkül. A nem lehetőség esetében a bezárási folyamat megszakításra (visszavonásra) kerül az „e.Cancel” igaz értékének beállításával.

3.4.16 Jelszó módosítása

A munkaablakban a beviteli mezőket kitöltve és a gombra vagy az F10 billentyű lenyomásával a jelszó követelmények kerülnek ellenőrzésre elsősorban. A mezők nem lehetnek üresek, az új jelszavaknak meg kell egyeznie, az új jelszónak minimum 8 karakterből kell állnia és tartalmaznia kell a jelszónak egy kis és nagy betűt.

Ha az „Összes eszköz kijelentkeztetés” előtti jelölönégyzet kiválasztásra került, akkor egy felugró ablakban tájékoztatásra kerül a felhasználó, hogy mindenazon kijelentkeztetésre kerül és csak az új jelszó használatával tud belépni ezek után az FMS rendszerébe. A felugró ablak mégse lehetőségét választva megszakításra kerül a jelszómódosítási művelet, a nem lehetőség kiveszi a jelölést a négyzetből, igen lehetőség esetében pedig megy tovább a módosítási folyamat.

A „PasswordChange” metódus elvégzi POST módszerrel az „sso/user/passwordchange” segítségével a módosítást. A régi és az új jelszó megadásának titkosítása szükséges, ezért az „Encrypt” osztályt felhasználva az „EncryptedText” tulajdonságnak az értéke kerül elküldésre.

```
{ "oldpassword", new Encrypt(tbOldPassword.Text).EncryptedText },  
{ "newpassword", new Encrypt(tbNewPassword.Text).EncryptedText },
```

Ha az összes eszköz kijelentkeztetése engedélyezésre került, akkor a „logout” kulcs értéke 1 lesz, ellenkező esetben 0, amely a ternális operátor segítségével kerül beállításra.

A lekérdezés eredményét a felugró ablak jeleníti meg, majd törlésre kerülnek a beviteli mezőkbe begépelt adatok, illetve a jelölés eltávolításra kerül a négyzetből és a régi jelszó beviteli szöveges mező kerül fókuszba az ablakon belül.

```
{ "logout", (chkAllDevices.Checked ? "1" : "0") }
```

3.4.17 Jogosultságok

A színek és a beviteli mezők szerkeszthetőségének letiltása után a „RolesRequest” aszinkron művelet tartalmazza az "sso/user/accesslist" kérését, amely sikereség esetén a szerepköröket hozzáadja a „roles” nevű listához, majd a választható szerepkörök listájának adatforrásaként a „roles” kerül beállításra.

Amennyiben a vállalat azonosítójának értéke nulla, amely a „JTokenType.Null” vizsgálat alapján kerül megvizsgálására, akkor az azonosító értéke 0 lesz.

```
role["company_id"].Type == JTokenType.Null ? 0 : role["company_id"]
```

A megjelenített aktív szerepkörök listájában való választás változásakor aktualizálásra kerülnek a szöveges mezők.

3.4.18 Aktív munkamenetek

Az ablak megnyitását követően a „DisableControllers” letiltja a beviteli mezők szerkeszthetőségét és hozzáférését, majd elrejti az IP információkat tartalmazó mezőket és a színek beállítását követően lekérésre kerül a „SessionRequest” metódus segítségével az „sso/user/devices” modul által biztosított adatok.

Sikeres lekérdezsnél a „sessions” listához hozzáadásra kerülnek az adatok, majd az aktív munkamenetek adatforrása lesz a munkamenetek listája, amennyiben a lekérdezések tartalmaznak legalább 1 darab sort.

Frissítés esetében a „RefreshData” metódus meghívásával törli az összes adatot a listából, majd egy új lekérdezés segítségével aktualizálásra kerülnek az adatok.

A platform szövegre kattintva, megjelenítésre kerül egy felugró ablakban a munkamenet azonosítója, a bejelentkezett platform típusa, illetve a további információk is megjelenítésre kerülnek.

A kiválasztott munkamenetek kényszerített kijelentkeztetését vagy az F10 billentyű lenyomásával és a felugró ablakot tartalmazó kérdést jóváhagyva meghívásra kerül a „StopSessionRequest”, amely az „sso/user/stopsession” lekérdezés segítségével deaktiválható a kiválasztott munkamenet. A kijelentkezet sikeréről felugró üzenet tájékoztatja a felhasználót, majd az adatok aktualizálását a „RefreshData” biztosítja.

3.4.19 Biztonság

A megjelenő ablakban a felhasználó saját jelszavát megadva a biztonsági kulcs generálására kattintva a felugró ablakot jóváhagyva a „GenerateSecretKeyAPI” metódus meghívására kerül sor, amely során a beírt jelszót az „Encrypt” osztály segítségével

titkosítjuk, és az „sso/user/secretkey” API POST kérés „password” kulcs értékének beállítása történik.

Sikerességen követően a biztonsági kulcs a vágólapra kerül másolásra, majd a felugró ablakban a QR kód megjelenítésére rákérdező dialógus, igen lehetőségét választva, megnyitásra kerül a megjelenítő, a biztonsági kulcsot átadva paraméterként.

```
FormQRViewer frmQRViewer = new  
FormQRViewer(Convert.ToString(getKeyReq.Result["result"]["secretkey"]));  
frmQRViewer.ShowDialog();
```

A fiók törlésére kattintva három darab figyelmezhető kérdés kerül megjelenítésre, minden három igen lehetőséget kiválasztva meghívásra kerül a „DeleteAccountAPI”, amely az „sso/user/deldata” végrehajtásáról szóló eredmény kerül megjelenítésre egy felugró ablakban. Sikerességen követően, az összes funkció inaktívvá válik, mivel az API a törlést követően érvényteleníti az összes munkamenetet is.

3.4.20 Feladatok

A munkaablak megnyitását követően a „DisablePlusInfo” elrejti a kiválasztható feladat információit, majd a színek beállítása után az adatok lekérése következik, amelyben az „sso/task/list” segítségével a sikeres lekérdezést követően törlésre kerül a „tasklists” elnevezésű lista, amely után az API által visszaadott feladatokkal történik feltöltésre a lista, majd a FillDataGridView beállítja a táblázat adatforrásaként a már korábban említett módszerrel, illetve beállítja az adatszámlálót is.

3.4.21 Kiválasztott feladatok információi

Az információkat megnyitva, amennyiben a paraméterként megkapott kiválasztott feladat vizsgálatára kerül sor, ami során a „FillInfos” kitölți az ablak bal oldalában található mezőket.

A „RequestMessages” az „sso/task/message” segítségével feltölti az üzenetek listát a feladatokhoz tartozó üzenetekkel, majd a „FillMessagesDataGridView” beállítja az üzenetek táblázat adatforrásaként az üzenetek létrehozási időpontja szerinti csökkenő sorrendben, amennyiben az üzenetek száma nem egyenlő nullával.

```
messages = new List<TaskMessage>(messages.OrderByDescending(x => x.Created));  
dgMessages.DataSource = messages;  
lblCounter.Text = $"{dgMessages.Rows.Count} / {messages.Count} darab";
```

A státuszváltás gombra kattintva a „HideButtons” elrejti a frissítés idejére a gombokat, majd a „StatusChange” metódusban lévő „sso/task/statuschange” lekérés hajtja végre. A lekérés sikereségtől függően felugró ablakban kerül megjelenítésre az

API válasza, illetve sikeresség esetén a „Completed” tulajdonság értékének ellentétjére kerül megváltoztatásra. Ezek után a „FillInfos” aktualizálja az adatokat, majd újra láthatóvá teszi a feladatok és üzenetek kezelését irányító gombokat.

```
selectedTask.Completed = statusChangeReq.SuccessRequestByAPI ?  
!selectedTask.Completed : selectedTask.Completed;
```

Törlés gombra kattintva a felugró ablakban lévő kérdés elfogadásával elrejtésre kerülnek a gombok, majd a „DelTaskAPI”, az „sso/task/delete” segítségével kerül törlésre, majd az ablak automatikusan bezárásra kerül.

Új gombra kattintva megnyitásra kerül az új üzenet rögzítéséhez kitöltendő munkaablak, amelyben maximum 1000 karakter mennyiségen áll lehetőség a feladathoz üzenetet hozzáadni a rögzítés vagy az F10 gomb megnyomásával. A kattintáskor a vizsgálatokat követően a „PostMessages” segítségével az „sso/task/addmessage” API POST kérés kerül lefuttatásra.

```
FormNewMessage frmNewMessage = new FormNewMessage(token, selectedTask);  
frmNewMessage.ShowDialog();
```

3.4.22 Új feladat hozzáadása

Az új feladat megnyitását követően a „GetGroups” metódus lekéri a „sso/task/group” segítségével azokat a vállalati munkacsoportokat, amelyekben a bejelentkezett felhasználó is benne van. A „groups” lista ürítését és feltöltését követően a „FillGroupComboBox” beállítja a „ComboBox” adatforrásaként, majd a feladat címének szerkeszthetősége engedélyezésre kerül abban az esetben, ha a választható munkacsoportok száma nem egyenlő nullával.

```
if (groups.Count != 0) {  
    cbGroup.DataSource = groups;  
    tbTitle.Enabled = true;  
    cbGroup.DisplayMember = "Name";  
    cbGroup.ValueMember = "ID";  
}
```

Amennyiben, ha a felhasználó szeretne a feladathoz határidőt hozzárendelni, akkor a dátumválasztó melletti jelölőnégyzet kijelölése szükséges, ezáltal szerkeszthetővé válik a határidő. A rögzítés gombra vagy az F9 gombot megnyomva a „PostNewTask” metódus az „sso/task/adddtask” segítségével kerül létrehozásra. A POST kulcsérték-párok előkészítése a „chkDeadline” jelölőnégyzet kijelölésétől függően kerülnek összeállításra. Ha a négyzetet a felhasználó bejelölte, akkor a „deadline” kulcs hozzáadásra kerül.

A cím maximum 100 karakter terjedelmű lehet, a számláló alapján nyomon követhető, hogy még hány karakter beírása lehetséges.

```
lblCharCounter.Text = $"Még {Convert.ToInt32(tbTitle.MaxLength)} -  
tbTitle.Text.Length} karakter";
```

Ha a szöveg értéke egyenlő nullával, akkor a rögzítés gomb elrejtésre kerül, ellenkező esetben megjelenik a jobb alsó sarokban, amivel elvégezhető a feladat rögzítése.

3.4.23 QR kód megjelenítése

A QR kód ablak egy képet megjelenítő vezérlőt tartalmaz, amely a megnyitáskor a QR kódot tartalmazó paraméter megadásával a **FELX API** generálásával történik, amely képnek a forrása a következő:

```
pbQR.ImageLocation = Settings.Default.apiurl + $"/qr/gen?value={value}";  
Az ablak bezárása az „ESC” gomb megnyomásával is lehetséges.
```

3.4.24 Változások

A változtatások lekérdezése a színek és a logó beállítása után a munkaablak megnyitásával automatikusan megtörténik, amely a „RequestInfo” metódust tartalmazó „info/versions” GET lekérdezéssel az asztali alkalmazásra való szűréssel megjelenítésre kerül.

3.4.25 Diagnosztika

A hálózati diagnosztika a gombra kattintva kerül elindításra, amelyben először a sorokat tartalmazó lista törlésre kerül, majd a gombok elrejtése történik és a „Start” metódus elindításával indul a diagnosztika.

Először az aktuális idő kerül elmentésre a „started” változóba. Lekérdezésre kerül a publikus IPv4 és IPv6 cím a „NetworkData” osztály használatával, majd az elérési teszt a „Ping” használatával lehetséges, amelyben a „PingReply” tartalmazza a teszt eredményét. Ha az „answer” eredménye nem nulla, akkor a listához hozzáadásra kerül az „Availability” azaz a válasz elérhetősége, illetve a „Response time”, amely a válaszidőt tartalmazza ezredmásodpercben kifejezve.

```
Ping ping = new Ping();  
PingReply answer = ping.Send("api.felx.hu", 1000);  
if (answer != null)  
{  
    lbData.Items.Add($"Availability: {answer.Status}");  
    lbData.Items.Add($"Response time: {answer.RoundtripTime} ms");  
}
```

Kiíratásra kerülnek az aktív hálózati adapterek, illetve az adaptereket tartalmazó alapértelmezett átjáró címe is „Ping” teszteléssel történik.

A teszt zárása esetén a jelenlegi idő „ended” változóba való mentése történik meg először, amely után a „TimeSpan” segítségével az eltelt idő kiírásra kerül.

A mentés gomb ekkor láthatóvá válik, amely segítségével szöveges állományba lehet menteni a hálózati teszt eredményét.

A „SaveFileDialog” segítségével kiválasztásra kerül az állomány mentési helye, majd a „StreamWriter” létrehozza, és beleírja a fájlba soronként a tartalmat, UTF-8 kódolásban.

```
using (StreamWriter sw = new StreamWriter(new FileStream(path,  
FileMode.CreateNew), Encoding.UTF8)) {  
  
    foreach (var line in lbData.Items)  
    {  
        sw.WriteLine(line);  
    }  
    MessageBox.Show("Diagnosztikai adatok sikeresen elmentve a kiválasztott  
helyre!", this.Text, MessageBoxButtons.OK, MessageBoxIcon.Information);  
}
```

3.4.26 Névjegy

A névjegy egy „statikus” megjelenésű információkat tartalmazó munkaablak, amelyben a színek és a logó beállításán kívül az alkalmazás verziósáma a „VersionNumber” metódus segítségével kerül aktualizálásra, amely az „Application.ProductVersion” értékét adja vissza.

```
private void VersionNumber()  
{ lblVersion.Text = $"Alkalmazás verziója: {Application.ProductVersion}"; }
```

A „felx.hu” vagy a „status.felx.hu” weboldalak hivatkozására kattintva az alapértelmezett böngészőben kerül megnyitásra az elérési link a „Process.Start” segítségével.

3.4.27 Kijelentkezés a rendszerből

A kijelentkezés a kezelőpult menüjében található gombra vagy az „ESC” gombra kattintással kezdeményezhető. A felugró ablakot nyugtázva a „LogoutSession” metódus érvényteleníti a tokent az „sso/user/logout” API kérés segítségével, amely eredménye egy felugró ablakban kerül megjelenítésre. Ezek után újra megnyitásra kerül a bejelentkező felület, és a kezelőpult bezáródik a token beállításérték törlését és mentését követően.

```
LogoutSession();  
Settings.Default.token = "X";
```

```
Settings.Default.Save();
FormLogin frmLogin = new FormLogin();
this.Hide();
frmLogin.ShowDialog();
```

4. Az adatbázis célja

A vállalati rendszerünk fő alkotóeleme egy központi adatbázis, amely rengeteg felhasználói és vállalati adat eltárolására alkalmas, valós idejű elérést biztosítva. A tervezésnél a legfőbb szempontok között szerepel, hogy konzisztens adatokkal dolgozzunk, a lekérdezések gyorsan végrehajthatóak legyenek, és ne tartalmazzon felesleges redundanciát, amely későbbi komoly hibákat idézhet elő a rendszer működésében.

4.1 Tervezés megkezdése

A tervezést az alapok lefektetésével indítottuk. A fejlesztői csapat egyhangú döntése alapján a kiindulópont a rendszer felhasználói és az adminisztrációs eljárás alá vont vállalatainak adatai. Döntő tényező volt, hogy egy felhasználó ne csak egy vállalathoz férjen hozzá, hanem akár többhöz is, különböző jogosultsági szintekkel, amelyek szabályozhatják, hogy miket adminisztrálhatnak a rendszerben. A projektmunkánkban elsősorban a vállalati felhasználók kezelése és a különböző feladatok létrehozása, módosítása és törlése a cél.

4.2 Tervezési lépések

Elsőként a vállalatok és a felhasználók létrehozása történt meg, majd az API és az alkalmazások összehangolása miatt szükséges vált a szerepkörös rendszer megtervezése és fejlesztése, így időrendben létrejöttek a „szerepkörök” és a „huzzáférések” táblák, amellyel már megkezdődhettek a fejlesztés további lépései. Szükséges vált egy olyan tábla létrehozása, amely a Föld összes hivatalosan nyilvántartott országát tartalmazza. Ezek létrehozása és adatokkal történő feltöltése után már könnyebben tudtunk tesztelni bizonyos lekérdezésekkel, így a megfelelő ellenőrzések után bővítettük az adatbázisunkat a feladatokkal.

A csapatszintű egyeztetés után rögzítettük a rendszerben a változtatásokat. Az alkalmazások és az API fejlesztésének vége felé érve rájöttünk, hogy szükség van egy olyan egyedi biztonsági rendszer kifejlesztésére, amely védi a személyes felhasználói adatokat.

4.3 Egyedtípusok és táblák meghatározása

4.3.1 Országok (countries)

egyed neve	egyed típusa	egyed leírása	megjegyzés
ctr_id	int	ország azonosítója	elsődleges kulcs
ctr_name	varchar(128)	ország rövidített neve	
ctr_longname	varchar(128)	ország teljes neve	
ctr_country_code	varchar(2)	ország kódja	
ctr_capital	varchar(64)	ország fővárosa	
ctr_adjective	varchar(256)	ország neve melléknevesítve	
ctr_currency	varchar(64)	ország pénzneme	
ctr_curr_iso	varchar(4)	ország pénznemének ISO kódja	
ctr_sub_currency	varchar(32)	ország váltópénze	
ctr_name_en	varchar(128)	ország angol teljes neve	

4.3.2 Vállalatok (companies)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	vállalat azonosítója	elsődleges kulcs
name	varchar(200)	vállalat neve	
tax_number	varchar(200)	vállalat adószáma	
company_registration_number	varchar(200)	vállalat nyilvántartási száma (NyT)	
logo	varchar(300)	vállalat logójának web	

		elérési hivatkozása	
created	datetime	létrehozás dátuma	
modified	datetime	utolsó módosítás dátuma	

4.3.3 Felhasználók (users)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	felhasználó azonosítója	elsődleges kulcs
name	varchar(200)	felhasználó teljes neve	
email	varchar(200)	felhasználó kapcsolattartói e-mail címe	
password	varchar(200)	felhasználó titkosított jelszava	
email_secondary	varchar(200)	felhasználó másodlagos e-mail címe	
contact_tel	varchar(200)	felhasználó kapcsolattartói telefonszáma	
contact_tel2	varchar(200)	felhasználó másodlagos telefonszáma	
birth_date	date	felhasználó születési ideje	
birth_country	int	felhasználó születési országának azonosítója	idegen kulcs

birth_location	varchar(200)	felhasználó születési települése	
birth_name	varchar(200)	felhasználó születési neve	
mother_birth_name	varchar(200)	felhasználó édesanyjának leánykorú neve	
user_group	int	felhasználó csoportjának azonosítója	idegen kulcs
gender	tinyint(1) – boolean	felhasználó neme	0 – nő; 1 – férfi
created	datetime	felhasználó fiókjának létrehozási dátuma	
modified	datetime	felhasználó adatainak utolsó módosítási dátuma	

4.3.4 Felhasználói csoportok (usergroups)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	csoport azonosítója	elsődleges kulcs
name	varchar(200)	csoport megnevezése	

4.3.5 Vállalati szerepkörök (roles)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	szerepkör azonosítója	elsődleges kulcs
company_id	int	vállalat azonosítója	idegen kulcs
name	varchar(100)	szerepkör fantázianeve	

allowed	tinyint(1) – boolean	szerepkör státusza	0 – letiltva; 1 – engedélyezve
created	datetime	szerepkör létrehozási dátuma	
modified	datetime	szerepkör utolsó módosítási dátuma	
fms_framework_login	tinyint(1) – boolean	keretrendszeri bejelentkezés jogosultság	0 – nincs hozzárendelve; 1 – hozzárendelve
fms_framework_personal	tinyint(1) – boolean	keretrendszeri személyes adatok megtekintése és módosítása jogosultság	0 – nincs hozzárendelve; 1 – hozzárendelve
fms_framework_full	tinyint(1) – boolean	keretrendszeri ún. szuperadmin jogosultság	0 – nincs hozzárendelve; 1 – hozzárendelve
fms_framework_company_manager_full	tinyint(1) – boolean	keretrendszerben vállalat adminisztrációs jogosultság	0 – nincs hozzárendelve; 1 – hozzárendelve
fms_framework_task	tinyint(1) – boolean	keretrendszeri feladatok megtekintése és alapkezelés jogosultság	0 – nincs hozzárendelve; 1 – hozzárendelve
fms_framework_task_full	tinyint(1) – boolean	keretrendszeri feladatok hozzáadása és	0 – nincs hozzárendelve; 1 – hozzárendelve

		teljes jogosultság	
--	--	--------------------	--

4.3.6 Felhasználói szerepkörök hozzáférései (accesses)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	hozzáférés azonosítója	elsődleges kulcs
user_id	int	felhasználó azonosítója	idegen kulcs
role_id	int	szerepkör azonosítója	idegen kulcs
allowed	tinyint(1) – boolean	hozzáférés státusza (állapota)	0 – letiltva; 1 – engedélyezve
created	datetime	hozzáférés létrehozásának dátuma	
modified	datetime	hozzáférés utolsó módosításának dátuma	

4.3.7 Felhasználói munkamenetek (sessions)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	munkamenet azonosítója	elsődleges kulcs
user_id	int	felhasználó azonosítója	idegen kulcs
token	varchar(200)	rendszer által random karakterekből generált érték	
active	tinyint(1) – boolean	munkamenet érvényességi státusza	0 – inaktív; 1 – aktív

platform	varchar(200)	bejelentkezési felület típusa	
infos	varchar(500)	platformhoz tartozó szükséges információk	
ipv4	varchar(100)	munkamenethez társított publikus IPv4 cím	
ipv6	varchar(100)	munkamenethez társított publikus IPv6 cím	
created	datetime	munkamenet létrejöttének dátuma	
expiry	datetime	munkamenet lejáratának dátuma	
last_update	datetime	munkamenet utolsó módosítása (frissítése)	

4.3.8 Tevékenységek (logs)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	tevékenység azonosítója	elsődleges kulcs
session_id	int	munkamenet azonosítója	idegen kulcs
url	varchar(200)	API hivatkozás, amely tevékenység naplózásra kerül	
action	varchar(200)	tevékenység műveletének neve	
created	datetime	tevékenység naplózásának dátuma	

4.3.9 Felhasználói biztonsági kulcsok (secretkeys)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	biztonsági kulcs azonosítója	elsődleges kulcs
user_id	int	felhasználó azonosítója	idegen kulcs
secretkey	varchar(150)	rendszer által generált biztonsági kulcs	
active	tinyint(1) – boolean	biztonsági kulcs állapota	0 – aktív kulcs; 1 – felhasznált kulcs
created	datetime	biztonsági kulcs létrehozási dátuma	
modified	datetime	biztonsági kulcs utolsó módosításának dátuma	

4.3.10 Feladatok (tasks)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	feladat azonosítója	elsődleges kulcs
taskgroup_id	int	munkacsoport azonosítója	idegen kulcs
created_user_id	int	feladat létrehozójának felhasználói azonosítója	idegen kulcs
title	varchar(200)	feladat címe	
completed	tinyint(1) – boolean	feladat státusza	0 – elvégzendő; 1 – elvégzett
deadline	datetime	feladat határideje	

created	datetime	feladat létrehozásának dátuma	
modified	datetime	feladat utolsó módosításának dátuma	

4.3.11 Feladatokhoz tartozó üzenetek, megjegyzések (taskinfos)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	üzenet azonosítója	elsődleges kulcs
task_id	int	feladat azonosítója	idegen kulcs
message	varchar(1000)	üzenet	
created_user_id	int	üzenetet létrehozó felhasználó azonosítója	idegen kulcs
created	datetime	üzenet létrehozásának dátuma	
modified	datetime	üzenet utolsó módosításának dátuma	

4.3.12 Munkacsoportok (taskgroups)

egyed neve	egyed típusa	egyed leírása	megjegyzés
id	int	munkacsoport azonosítója	elsődleges kulcs
name	varchar(100)	munkacsoport neve	
company_id	int	vállalat azonosítója	idegen kulcs
created	datetime	munkacsoport létrehozási dátuma	
modified	datetime	munkacsoport utolsó	

		módosításának dátuma	
--	--	----------------------	--

4.3.13 Munkacsoporthoz rendelt felhasználók (taskgroupusers)

id	int	hozzárendelés azonosítója	elsődleges kulcs
taskgroup_id	int	munkacsoporthoz rendelni kívánt felhasználó azonosítója	idegen kulcs
user_id	int	munkacsoporthoz rendelni kívánt felhasználó azonosítója	idegen kulcs
created	datetime	hozzárendelés létrehozási dátuma	
modified	datetime	hozzárendelés utolsó módosításának dátuma	

4.4 Kapcsolatok meghatározása

- A `companies` 1:N-es kapcsolatban áll a `roles` táblával, a `companies` tábla egy egyed-előfordulásához a `roles` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `roles` N:M-es kapcsolatban áll a `users` táblával, a `roles` tábla több egyed-előfordulásához az `users` tábla több egyed-előfordulása tartozik, ez megfordítható.
- A `companies` 1:N-es kapcsolatban áll a `taskgroups` táblával, a `companies` tábla egy egyed-előfordulásához a `taskgroups` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `taskgroups` 1:N-es kapcsolatban áll a `tasks` táblával, a `taskgroups` tábla egy egyed-előfordulásához a `tasks` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `taskgroups` N:M-es kapcsolatban áll a `users` táblával, a `taskgroups` tábla több egyed-előfordulásához a `users` tábla több egyed-előfordulása tartozik, ez megfordítható.

- A `taskinfos` 1:N-es kapcsolatban áll a `tasks` táblával, a `taskinfos` tábla egy egyed-előfordulásához a `tasks` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `taskinfos` 1:N-es kapcsolatban áll a `users` táblával, a `taskinfos` tábla egy egyed-előfordulásához a `users` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `tasks` 1:N-es kapcsolatban áll a `users` táblával, a `taskgroups` tábla egy egyed-előfordulásához a `tasks` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `usergroups` 1:N-es kapcsolatban áll a `users` táblával, a `usergroups` tábla egy egyed-előfordulásához a `users` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `countries` 1:N-es kapcsolatban áll a `users` táblával, a `countries` tábla egy egyed-előfordulásához a `users` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `users` 1:N-es kapcsolatban áll a `sessions` táblával, a `user` tábla egy egyed-előfordulásához a `sessions` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- A `sessions` 1:N-es kapcsolatban áll a `logs` táblával, a `sessions` tábla egy egyed-előfordulásához a `logs` tábla több egyed előfordulása tartozik, ez fordítva nem igaz.

4.5 N:M kapcsolatok felbontása

- Mivel a `roles` N:M-es kapcsolatban áll a `users` táblával, ezért felbontjuk kettő darab 1:N-es kapcsolatra, így létrejön az `accesses` tábla
 - és a `roles` 1:N-es kapcsolatban áll az `accesses` táblával, a `roles` tábla egy egyed-előfordulásához az `accesses` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz,
 - és a `users` 1:N-es kapcsolatban áll az `accesses` táblával, a `users` tábla egy egyed-előfordulásához az `accesses` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz.
- Mivel a `taskgroups` N:M-es kapcsolatban áll a `users` táblával, ezért felbontjuk kettő darab 1:N-es kapcsolatra, így létrejön az `taskgroupusers` tábla

- és a `taskgroups` 1:N-es kapcsolatban áll a `taskgroups` táblával, a `taskgroups` tábla egy egyed-előfordulásához a `taskgroups` tábla több egyed-előfordulása tartozik, ez fordítva nem igaz,
- és a `taskgroups` 1:N-es kapcsolatban áll a `users` táblával, a `taskgroups` tábla egy egyed-előfordulásához a `users` tábla több-egyed előfordulása tartozik, ez fordítva nem igaz.

4.6 Minta adatok

4.6.1 Országok (countries)

ctr_id	ctr_name	ctr_longname	ctr_country_code	ctr_capital	ctr_adjective	ctr_currency	ctr_curr_iso	ctr_sub_currency	ctr_name_en
16	Ausztria	Osztrák Köztársaság	AT	Bécs	osztrák, ausztriai, osztrák köztársasági	euró	EUR	cent	Austria
58	Egyesült Államok	Amerikai Egyesült Államok	US	Washington	egyesült államokbeli, amerikai egységű államokbeli...	USA-dollár	USD	cent	United States
117	Kína	Kínai Népköztársaság	CN	Peking	kínai, kínai népköztársasági	renminbi jüan	CNY	jiao; fen	China
139	Magyarország	Magyar Köztársaság	HU	Budapest	magyar, magyarországi, magyar köztársasági	forint	HUF	fillér	Hungary
174	Oroszország	Orosz Föderáció	RU	Moszkva	orosz, oroszországi, orosz föderációbeli	orosz rubel	RUB	kopejka	Russian Federation
211	Szlovákia	Szlovák Köztársaság	SK	Pozsony (Bratislava)	szlovák, szlovákiai, szlovák köztársasági	euró	EUR	cent	Slovakia

92) ábra: Országok tábla

4.6.2 Vállalatok (companies)

id	name	tax_number	company_registration_number	logo	created	modified
1	Fresh FM	0000000-0-00	00000000	https://freshfm.hu/static/img/freshmaszk.png	2023-03-16 08:45:17	2023-04-01 21:58:47
2	PlayClan	0000000-0-00	00000000	https://play-lh.googleusercontent.com/9g5QG85dJc_n...	2023-03-16 08:46:09	NULL
5	FelX	0000000-0-00	00000000	https://cdn.felx.hu/assets/img/logo_x.png	2023-03-16 08:49:59	2023-03-20 13:14:30
7	DarkSystems	0000000-0-00	00000000	https://cdn.felx.hu/assets/img/company/darksystems...	2023-03-20 18:50:01	2023-03-20 18:54:41

93) ábra: Vállalatok tábla

4.6.3 Felhasználók (users)

id	name	email	password	email_secondary	contact_tel	contact_tel2
1000000000	Varsányi Barnabás	varsanyib@felx.hu	d72a22a2f3d989b9db9da21dfa8d77505ac6adaf2c17774a68...	varsanyib0125@gmail.com	+36303760712	NULL
1000000001	Medve Adrián	medveadrian@felx.hu	60aeeef10e6dd3802fef201a5cf377c5e3ea6b6c065b4d84ef...	medveadrian03@gmail.com	+36302338941	NULL
1000000002	Lázár Marcell	aldynenn@felx.hu	60aeeef10e6dd3802fef201a5cf377c5e3ea6b6c065b4d84ef...	NULL	NULL	NULL
1000000003	Torolt felhasználó	a8f9cda023@fms.intra	6335db2e36ba7c57c30f a8f9cda023	NULL	NULL	NULL

94) ábra: Felhasználók tábla I. rész

birth_date	birth_country	birth_location	birth_name	mother_birth_name	user_group	gender	created	modified
2004-01-25	139	Balassagyarmat	Varsányi Barnabás	Kalcsó Boglárka	1	1	2023-03-16 08:35:49	2023-04-01 22:00:49
2003-11-15	139	Balassagyarmat	Medve Adrián	Takács Marianna	1	1	2023-03-16 08:37:19	2023-03-30 13:53:53
2003-11-02	139	Balassagyarmat	Lázár Marcell	Kozma Anita	1	1	2023-03-16 08:39:36	2023-03-30 13:54:00
NULL	NULL	NULL	Törölt felhasználó	NULL a8f9cda023	2	1	2023-03-27 13:51:37	2023-03-27 13:52:03

95) ábra: Felhasználók tábla II. rész

4.6.4 Felhasználói csoportok (usergroups)

id	name
1	FMS User
2	FMS Deleted User

96) ábra: Felhasználói csoportok tábla

4.6.5 Vállalati szerepkörök (roles)

id	company_id	name	allowed	created	modified	fms_framework_login	fms_framework_personal
1	NULL	FMS Felhasználó	1	2023-01-15 21:22:11	2023-02-21 20:21:58	1	1
2	NULL	FMS Admin	0	2023-01-15 21:22:38	2023-04-01 22:05:35	1	1
3	1	Rendszergazda	1	2023-03-16 08:51:03	NULL	1	1
4	1	Műsorvezetők	1	2023-03-16 08:51:32	NULL	1	1
8	5	Tulajdonos	1	2023-03-16 08:53:20	2023-03-31 21:03:45	1	1
11	5	Tesztelők	1	2023-03-27 13:57:00	2023-03-31 21:23:23	1	1

97) ábra: Vállalati szerepkörök tábla I. rész

fms_framework_full	fms_framework_company_manager_full	fms_framework_task	fms_framework_task_full
0	0	0	0
1	1	1	1
0	1	1	1
0	0	1	0
0	1	1	1
0	0	1	0

98) ábra: Vállalati szerepkörök tábla II. rész

4.6.6 Felhasználói szerepkörök hozzáférései (accesses)

id	user_id	role_id	allowed	created	modified
1	1000000000	1	1	2023-03-21 20:45:25	NULL
2	1000000001	1	1	2023-03-21 20:45:31	NULL
3	1000000002	1	1	2023-03-21 20:45:37	NULL
5	1000000001	5	1	2023-03-21 20:47:40	2023-03-22 08:59:02
45	1000000004	11	0	2023-03-31 23:19:13	2023-04-01 00:23:02

99) ábra: Felhasználói szerepkörök hozzáférései tábla

4.6.7 Felhasználói munkamenetek (sessions)

id	user_id	token	active	platform	infos
3	1000000000	b0e3479e0ec423cd9a662bd4b67b6b88389766e3	1	Desktop	OS: Microsoft Windows NT 6.2.9200.0 64 bit Logged User: VARSOPC\varso Computer Name: VARSOPC FMS Framework Desktop Version: 1.1.0.0
29	1000000002	ae0cd7e01dcf9b17063010191f21e64efac9a497	0	Website	Browser: Safari OS: iPhone
38	1000000001	39d8445ec15c61b2812a5ee48189ba964a9039e4	0	Mobile	Brand: iPhone 13 Model: iPhone Device: iPhone14,5 Software: iOS 16.2 App version: 1.0.7
40	1000000006	5e2a2ca7da286d1e1fa41e2a553676aeb8eec1c9	0	Desktop	OS: Microsoft Windows NT 6.2.9200.0 64 bit Logged User: MEDVEADRINBFDC\adrian Computer Name: MEDVEADRINBFDC FMS Framework Desktop Version: 1.0.6.0
57	1000000002	f1938a6326f6429358194f401de5bd4af256d357	0	Desktop	OS: Microsoft Windows NT 6.2.9200.0 64 bit Logged User: DESKTOP-6U5RBNKA\dynenn Computer Name: DESKTOP-6U5RBNK FMS Framework Desktop Version: 1.0.7.1

100) ábra: Felhasználói munkamenetek tábla I. rész

ipv4	ipv6	created	expiry	last_update
178.164.136.183	2a01:36d:113:279f:80d4:bb9d:a3a0:1bcb	2023-03-27 13:50:24	2023-04-02 10:42:30	2023-04-01 10:42:30
NULL	2a09:bac2:395a:478::72:106	2023-03-29 12:29:00	2023-03-29 13:43:54	2023-03-29 12:44:50
5.38.143.17	NULL	2023-03-29 20:28:28	2023-04-05 20:28:28	2023-03-29 20:32:35
5.38.143.17	2001:4c4e:12d7:3c00:3506:e6fe:c0ae:aa7b	2023-03-29 21:50:32	2023-03-30 21:50:32	2023-03-29 22:14:14
195.199.232.1	NULL	2023-03-31 11:35:49	2023-04-01 11:35:49	2023-04-01 14:27:54

101) ábra: Felhasználói munkamenetek tábla II. rész

4.6.8 Tevékenységek (logs)

id	session_id	url	action	created
83	3	api.felx.hu/sso/user/secretkey	SSO - Biztonsági kulcs generálás	2023-03-28 10:34:56
112	15	api.felx.hu/sso/user/deldata	SSO - Felhasználói fiók törlés	2023-03-28 11:42:36
276	30	api.felx.hu/auth/permcheck	Jogosultság ellenőrzése új token létrehozásával	2023-03-29 12:45:06
279	30	api.felx.hu/sso/task/addtask	SSO - Új feladatrögzítés	2023-03-29 12:59:28
453	52	api.felx.hu/sso/company/assigngroupuser	SSO - Felhasználó hozzárendelése munkacsoporthoz	2023-03-31 20:28:01
476	3	api.felx.hu/sso/company/accesschange	SSO - Felhasználói hozzáférés módosítás	2023-03-31 22:24:46
477	3	api.felx.hu/sso/company/removeuseraccess	SSO - Felhasználói hozzáférés törlése	2023-03-31 22:32:11

102) ábra: Tevékenységek tábla

4.6.9 Felhasználói biztonsági kulcsok (secretkeys)

id	user_id	secretkey	active	created	modified
1	1000000005	8b0847c3fe3088199731ba9f44cd91437c31696d02fc9221b7...	0	2023-03-28 11:36:36	2023-03-28 11:42:35
9	1000000006	c8331d0246de7cdf8d9c3df68027de6498e64a49c2af472a89...	0	2023-03-29 21:50:45	2023-03-29 22:14:14
18	1000000001	650b1746e280d45cd9bd00c63badd2892a0391018b2e1479ca...	0	2023-03-30 09:44:54	2023-03-30 09:47:58
22	1000000001	41d32cff6441f42e7de5a6b629452beb2ead9fb20e52afeac0...	1	2023-03-30 10:14:44	NULL
26	1000000001	9587f9bb5d6fd06c66b431a4abd8f817f90fe22d3338d9e968...	0	2023-03-30 10:51:35	2023-03-30 10:51:43
31	1000000001	09ad6c46e50fe5134c8a8be57683b50bfceeb1ba2d91bae597...	1	2023-03-30 13:55:04	NULL

103) ábra: Felhasználói biztonsági kulcsok tábla

4.6.10 Feladatok (tasks)

id	taskgroup_id	created_user_id	title	completed	deadline	created	modified
1	1	1000000000	Első feladat a fejlesztői munkacsoport számára	1	NULL	2023-03-22 08:16:38	2023-03-27 08:35:02
7	4	1000000002	Feladatok módosításának implementálása	1	NULL	2023-03-23 08:36:25	2023-03-24 09:04:08
11	5	1000000000	DataGridView többsoros megjelenítés	1	NULL	2023-03-27 09:07:41	2023-03-27 12:02:49
23	1	1000000000	Projektmunka leadása	0	2023-04-28 08:30:00	2023-03-29 11:22:05	2023-03-29 22:38:31
26	8	1000000002	Teszt feladat	0	NULL	2023-03-29 13:02:13	NULL

104) ábra: Feladatok tábla

4.6.11 Feladatokhoz tartozó üzenetek, megjegyzések (taskinfos)

id	task_id	message	created_user_id	created	modified
1	1	Első üzenet az első feladathoz	1000000000	2023-03-22 08:18:03	NULL
5	5	Többi vezetőségi tag felvitele szükséges, a műsorvezetőkkel együtt	1000000000	2023-03-22 09:42:25	NULL
7	4	Javításra került, V1.0.5.0 updateben lesz megtalálható.	1000000000	2023-03-22 09:44:59	NULL
26	1	Státuszváltás (Elvégezve)	1000000000	2023-03-25 23:51:29	NULL
35	9	UI kész, lekérdezés winapphoz hozzáadás szükséges	1000000000	2023-03-27 09:03:18	NULL

105) ábra: Feladatokhoz tartozó üzenetek, megjegyzések tábla

4.6.12 Munkacsoportok (taskgroups)

id	name	company_id	created	modified
1	Fejlesztők	5	2023-03-22 08:02:30	2023-03-22 08:02:53
2	Android Dev	5	2023-03-22 08:03:09	2023-03-22 08:03:25
3	IOS Dev	5	2023-03-22 08:04:04	2023-03-22 08:05:07
4	Web Dev	5	2023-03-22 08:04:15	2023-03-22 08:05:10
5	Desktop Dev	5	2023-03-22 08:04:25	2023-03-22 08:05:14

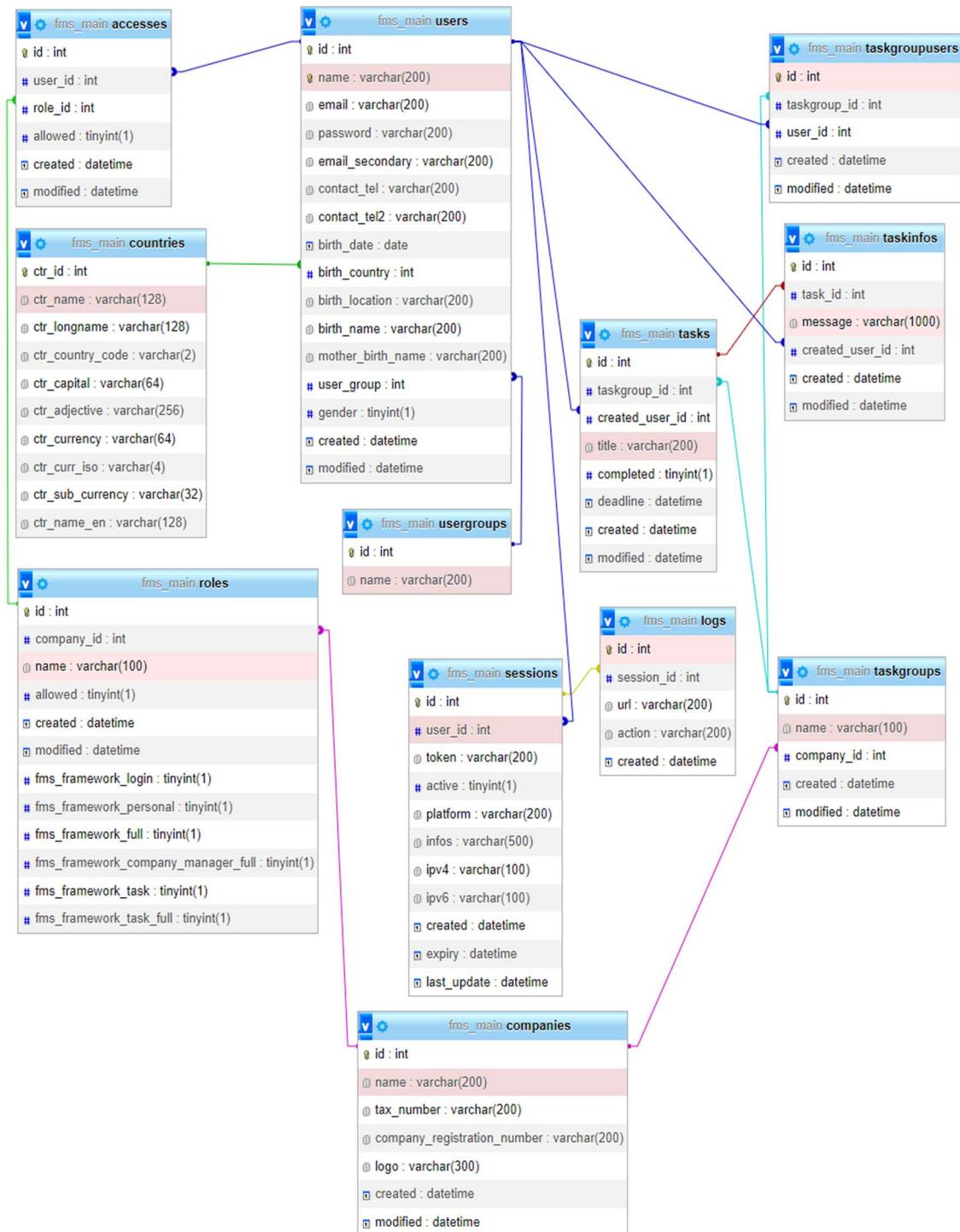
106) ábra: Munkacsoportok tábla

4.6.13 Munkacsoporthoz rendelt felhasználók (taskgroupusers)

id	taskgroup_id	user_id	created	modified
2	1	1000000001	2023-03-22 08:10:50	NULL
3	1	1000000002	2023-03-22 08:11:02	NULL
9	5	1000000000	2023-03-22 08:15:47	NULL
11	6	1000000000	2023-03-25 23:53:18	NULL
20	2	1000000001	2023-03-29 11:47:42	NULL
42	10	1000000001	2023-03-31 20:28:01	NULL

107) ábra: Munkacsoporthoz rendelt felhasználók tábla

4.7 Adatbázis diagram



108) ábra: Adatbázis diagram

4.8 Adatbázis továbbfejlesztési lehetőségek

- Feladat létrehozása egy felhasználó számára, munkacsoport létrehozása nélkül (új egyedtípus).
- Feladatok munkacsoportok számára történő megjelenítési dátum hozzáadása (új egyedtípus).
- Egy feladat kiosztása több munkacsoport részére (új kapcsolótábla, létrejövő N:M kapcsolat felbontása).
- Vállalatok további adatainak bővítése (új egyedtípusok).
- Felhasználó lakcímének eltárolása (új egyedtípusok és a címtár tárolásához szükséges táblák létrehozása).

4.9 Megoldott hibák

- Felhasználók személyes adataiban a születési országnál felléphetett olyan hiba, hogy nem mindenki egységesen viszi fel az ország nevét, amelyből akár káros adat-redundancia is keletkezhetett volna, vagy megváltozik az ország neve, amely elsődleges elhárítási prioritással kezeltem. (Megoldás: létrehoztam egy új táblát.)
- Lehetőség lett volna a munkamenet lejáratí idő megkerülésére, ha a felhasználói oldalon ellenőrizzük a lejáratí és az aktuális idő differenciáját. (Megoldás: Az összes létező létrehozási, utolsó módosítási és lejáratí dátum az aktuális szerveridő által történik mentésre.)

5. Összefoglalás

A fejlesztés közben a háromtagú csapatunk folyamatos együttműködése és kapcsolattartása folyamán sikerült párhuzamosan haladni a munkálatokkal.

Továbbfejlesztési céljaink közé tartozik az API által használt technológia dinamikussá fejlesztése és a Node.js, illetve Python által biztosított lehetőségek kihasználása.

Az API és az adatbázis Varsányi Barnabás otthoni szerverén fut, amelynek rendszerkövetelménye rendkívül alacsony, így szeretnénk felköltözteni a felhőbe a rendszerünket, a rendelkezésre állási idő maximalizálására törekedve.

Az asztali alkalmazásban a feladatok és a vállalati felhasználók kilistázásánál lévő szűrés hozzáadása lenne a cél.

Mindhárom alkalmazás esetén a felhasználó saját jelszavának visszaállítása a levelezőrendszeren kereszttüli visszaállító link hozzáadásával lenne elsődleges prioritásként a továbbfejlesztés célja. Az ehhez kapcsolódó levél elküldéséhez tartozó API folyamatok előkészítésre kerültek az FMS rendszerében.

A mobil alkalmazást szeretnénk az alkalmazások áruházaiba feltölteni és rendelkezésre bocsátani, amelynek alapvető feltételeinek nagy része már most rendelkezésünkre áll.

A modern technológiákhoz igazodva, a webes alkalmazást szeretnénk átültetni vagy a Google által fejlesztett és karbantartott Angular keretrendszerbe vagy pedig a Facebooktól származó React könyvtáron alapuló Next.js keretrendszerbe, mivel ez könnyebbé teszi a különböző bővítmények hozzáadását és implementálását az alkalmazásba.

Források

Adatbázis

Országok: http://webdraft.eu/orszagok_varosok/

C# (asztali alkalmazás)

Await Async Taskok logikájának felhasználása másik programnyelvből:

<https://learn.microsoft.com/hu-hu/dotnet/visual-basic/programming-guide/concepts/async/>

DateTimePicker Format módosítása: <https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.datetimepicker.customformat?view=windowsdesktop-7.0>

Error Log mentéséhez Temp mappa: <https://stackoverflow.com/questions/944483/how-to-get-temporary-folder-for-current-user>

FormClosing event visszavonása, figyelmeztető ablak bezárás előtt:

<https://stackoverflow.com/questions/47788288/how-to-cancel-closing-winform>

GET és POST method: <https://stackoverflow.com/questions/4015324/send-http-post-request-in-net>

Hálózati adapterinformációk: <https://gist.github.com/BrunoCaimar/8399190>

JObject és JToken üres ellenőrzés:

<https://stackoverflow.com/questions/24066400/checking-for-empty-or-null-jtoken-in-a-jobject>

Loading indicator:

https://www.google.com/url?sa=i&url=https%3A%2F%2Ftenor.com%2Fsearch%2Floading-gifs&psig=AOvVaw0lF5Ti3bzaS_fHU31B3z2N&ust=1680090002762000&source=images&cd=vfe&ved=0CA8QjRxqFwoTCOjD8o_F_v0CFQAAAAAdAAAAABAE

MenuStrip ikonok: <https://icons8.com/icons/set/users> és <https://iconscout.com/>

Nem létező mappa létrehozása vagy figyelmen kívül hagyása IO műveletnél:

<https://stackoverflow.com/questions/9065598/if-a-folder-does-not-exist-create-it>

Projektfájlok gitignore hozzáadás: <https://gist.github.com/kmorcinek/2710267>

Vágólapra másolás: <https://stackoverflow.com/questions/3546016/how-to-copy-data-to-clipboard-in-c-sharp>

Flutter (mobil alkalmazás)

Adatátadás AlertDialog-nak: <https://stackoverflow.com/a/66075284>

Android 11+ tárhely írási jog javítása: <https://stackoverflow.com/a/73630987>

Android felhasználói adatok törlésre kényszerítés app törlés után:

<https://stackoverflow.com/questions/33169618/an-android-app-remembers-its-data-after-uninstall-and-reinstall>

Betöltés ikon megjelenítése gombon belül: <https://stackoverflow.com/a/66711802>

Betűtípus beállítása: <https://docs.flutter.dev/cookbook/design/fonts>

Billentyűzet overflow fix: <https://stackoverflow.com/a/67571609>

CheckBox szöveg elé helyezése: <https://stackoverflow.com/a/53842698>

DataTable overflow fix: <https://stackoverflow.com/a/66002643>

Dátumválasztó színmódosítás: <https://www.flutterbeads.com/change-date-picker-color-in-flutter/>

Dropdown kulcs-érték pá�: <https://stackoverflow.com/a/49722255>

Dropdown szélesség fix: <https://stackoverflow.com/a/70275741>

DropdownButton border radius fix: <https://github.com/flutter/flutter/pull/106657>

Elválasztó sáv szöveggel: <https://stackoverflow.com/a/54059454>

Encrypt IV nélkül: <https://stackoverflow.com/a/70845352>

Fektetett mód letiltása: <https://www.kindacode.com/article/how-to-disable-landscape-mode-in-flutter/>

Fektetett nézet drawer fix:

https://github.com/medyas/flutter_zoom_drawer/issues/54#issuecomment-915880574

FormData küldése bool-ként: <https://stackoverflow.com/a/51222349>

JSON átalakítás osztállyá: <https://docs.flutter.dev/development/data-and-backend/json>

JSON feldolgozás: <https://stackoverflow.com/a/67620007>

JSON pretty print: <https://gist.github.com/kasperpeulen/d61029fc0bc6cd104602>

Képernyő alján elhelyezkedő gombok kitakarnak tartalmat fix:

<https://github.com/flutter/flutter/issues/50314#issuecomment-805688949>

Lista await fix: <https://stackoverflow.com/a/51112799>

Lista border radius: <https://stackoverflow.com/a/70771589>

Lista elemek közötti hely hozzáadása: <https://stackoverflow.com/a/57034446>

Lista renderelési hiba fix: <https://stackoverflow.com/a/66476465>

Más oldalra navigálás: <https://docs.flutter.dev/cookbook/navigation/named-routes>

POST method API lekérés: <https://pub.dev/packages/dio#sending-formdata>

Splash szín letiltása CheckBox-on: <https://stackoverflow.com/a/58673392>

StatefulBuilder StatelessWidget-ben: <https://stackoverflow.com/a/69492443>

Text overflow fix DropDownMenu:

<https://github.com/flutter/flutter/issues/42328#issuecomment-825985560>

Toast overlay error fix:

<https://github.com/ponnamkarthik/FlutterToast/issues/393#issuecomment-1423684027>

Widget frissítés AlertDialog-ban: <https://stackoverflow.com/a/74152900>

Titkosítás

SHA256: <https://www.codeproject.com/Tips/1156169/Encrypt-Strings-with-Passwords-AES-SHA>

PHP API

IP Infó API: <https://ipinfo.io/>

MySQL Prepared Statements: Szabadságtervező SzGyA és

https://www.w3schools.com/php/php_mysql_prepared_statements.asp

OOP PHP: https://www.w3schools.com/php/php_oop_classes_objects.asp

PHP GET lekérdezés: <https://stackoverflow.com/questions/959063/how-to-send-a-get-request-from-php>

PHP POST lekérdezés: <https://stackoverflow.com/questions/5647461/how-do-i-send-a-post-request-with-php>

PHP Request link visszakérés: <https://stackoverflow.com/questions/6768793/get-the-full-url-in-php>

QR kód generátor: <https://github.com/endroid/qr-code>

Update MySQL Subquery: <https://stackoverflow.com/questions/45494/mysql-error-1093-cant-specify-target-table-for-update-in-from-clause>

Web

CSS Reset (átalakítva): <https://meyerweb.com/eric/tools/css/reset/>

Folyamatjelző sáv (még nincs felhasználva): <https://www.npmjs.com/package/elastic-progress-lib>

Lassabb leállás: <https://easings.net/>

Betűtípus: <https://fonts.google.com/specimen/Ubuntu>

Süti olvasó függvény: <https://stackoverflow.com/questions/10730362/get-cookie-by-name>

SVG animáció: https://www.youtube.com/watch?v=H_7Ld5Psgg0

Ábrajegyzék

1)	ábra: Főoldal	14
2)	ábra: Letöltési oldal	14
3)	ábra: Vállalatok kiválasztása	16
4)	ábra: Feladatok megjelenése	16
5)	ábra: Feladat rögzítése	17
6)	ábra: Feladatok részletei	17
7)	ábra: Felhasználó adatai	18
8)	ábra: Biztonsági kulcs generálása	19
9)	ábra: Jelszómódosítás	19
10)	ábra: Aktív munkamenetek	20
11)	ábra: Felhasználók	20
12)	ábra: Felhasználó részletei	21
13)	ábra: Új felhasználó létrehozása	21
14)	ábra: Regisztrált felhasználó hozzárendelése a vállalathoz	22
15)	ábra: Munkacsoport részletei, felhasználó hozzárendelése	22
16)	ábra: Új munkacsoport létrehozása	23
17)	ábra: Bejelentkezési képernyő	24
18)	ábra: Főoldal	25
19)	ábra: Navigációs menü	26
20)	ábra: Feladatok oldal	27
21)	Feladat hozzáadása	28
22)	ábra: Kiválasztott feladat információi, üzenetei	29
23)	ábra: Adminisztráció oldal	29
24)	ábra: Új FMS felhasználó létrehozása	30
25)	ábra: Regisztrált FMS felhasználó oldal	31
26)	ábra: Vállalati munkacsoportok	32
27)	ábra: Vállalati felhasználók oldal	33
28)	ábra: Felhasználói adatak módosítása	34
29)	ábra: Fiók oldal	35
30)	ábra: Aktív eszközök oldal	36
31)	ábra: Jogosultságok oldal	36
32)	ábra: Biztonság oldal	37

33)	ábra: Biztonsági kulcs generálása	38
34)	ábra: Fiók törlése figyelmeztetés	38
35)	ábra: Beállítások oldal	39
36)	ábra: Névjegy oldal	40
37)	ábra: Kijelentkezés felhívás	40
38)	ábra: Asztali alkalmazás bejelentkezési felülete.....	42
39)	ábra: Bejelentkezéskor megjelenő rövidített adatvédelmi nyilatkozat felugró ablakban	42
40)	ábra: Asztali alkalmazás kezelőpultja.....	43
41)	ábra: Vállalat menüpont.....	43
42)	ábra: Vállalat kiválasztása munkaablak	43
43)	ábra: Vállalat-specifikus funkció figyelmeztető üzenete	44
44)	ábra: Új FMS felhasználó létrehozása munkaablak.....	45
45)	ábra: Új FMS felhasználó sikeres létrehozásról szóló tájékoztatás	45
46)	ábra: Új FMS felhasználóhoz sikeres szerepkör hozzárendelés üzenete ..	46
47)	ábra: Új FMS felhasználó kötelezően megadandó adatok üzenete.....	46
48)	ábra: Meglévő FMS felhasználó helytelen kulcs hibaüzenete.....	46
49)	ábra: Meglévő FMS felhasználó helytelen formázú kulcs hibaüzenete	47
50)	ábra: Meglévő FMS felhasználó munkaablak	47
51)	ábra: Meglévő felhasználóhoz sikeres szerepköri hozzárendelés	48
52)	ábra: Vállalat adatai munkaablak.....	48
53)	ábra: Vállalati munkacsoportok kezelése munkaablak	49
54)	ábra: Vállalati munkacsoport létrehozása	49
55)	ábra: Vállalati munkacsoporthoz felhasználó hozzárendelése.....	49
56)	ábra: Vállalati munkacsoporthoz felhasználó eltávolítása	50
57)	ábra: Vállalati felhasználók munkaablak	50
58)	ábra: Vállalati felhasználók kezelése I. rész	51
59)	ábra: Vállalati felhasználók kezelése II. rész.....	51
60)	ábra: Felhasználó hozzáférésének letiltását tartalmazó üzenet.....	52
61)	ábra: Vállalati felhasználó letiltott szerepköreinek engedélyezése.....	52
62)	ábra: Saját fiók menüpont	52
63)	ábra: Saját fiók (adatlap) munkaablak	53
64)	ábra: Saját adatok módosításának jóváhagyó felugró üzenete.....	53
65)	ábra: Saját jelszó módosítása munkaablak.....	54

66)	ábra: Jogosultságok munkaablak	54
67)	ábra: Felhasználó aktív munkamenetei munkaablak	55
68)	ábra: Munkamenet további információi felugró ablakban.....	55
69)	ábra: Munkamenet kényszerített kijelentkeztetése	56
70)	ábra: Saját fiók kezelése a biztonsági munkaablakban.....	56
71)	ábra: Sikeres biztonsági kulcs generálásról szóló felugró üzenet.....	57
72)	ábra: Biztonsági kulcs megjelenítve QR kódos formátumban.....	57
73)	ábra: Megjelenített QR kód beolvasásának értéke.....	57
74)	ábra: Felhasználói fiók törlésre felszólító üzenet	58
75)	ábra: Sikeres felhasználói fiók törlés üzenete	58
76)	ábra: Feladat menüpont.....	58
77)	ábra: Feladatak munkaablak	59
78)	ábra: Kiválasztott feladat információi munkaablak	59
79)	ábra: Feladat állapotváltozásának sikerességéről tájékoztatás.....	60
80)	ábra: Feladathoz új üzenet rögzítése	60
81)	ábra: Törölt felhasználó üzenetei a kiválasztott feladat információi munkaablakban	61
82)	ábra: Új feladat rögzítése munkacsoporthoz számára munkaablak	61
83)	ábra: Egyéb menüpont	61
84)	ábra: Lefuttatott sikeres diagnosztika munkaablak.....	62
85)	ábra: Diagnosztikai adatok mentése	64
86)	ábra: Lementett diagnosztikai adatok szövegszerkesztő szoftverben.....	64
87)	ábra: Asztali alkalmazás változásai ablak.....	65
88)	ábra: Névjegy ablak	65
89)	ábra: Kijelentkezés menüpont.....	66
90)	ábra: Kijelentkezés megerősítése felugró ablak.....	66
91)	ábra: Asztali alkalmazás könyvtárszerkezete Visual Studio 2019-ben. .	117
92)	ábra: Országok tábla	154
93)	ábra: Vállalatok tábla	154
94)	ábra: Felhasználók tábla I. rész.....	154
95)	ábra: Felhasználók tábla II. rész	155
96)	ábra: Felhasználói csoportok tábla.....	155
97)	ábra: Vállalati szerepkörök tábla I. rész.....	155
98)	ábra: Vállalati szerepkörök tábla II. rész	155

99)	ábra: Felhasználói szerepkörök hozzáférései tábla	156
100)	ábra: Felhasználói munkamenetek tábla I. rész	156
101)	ábra: Felhasználói munkamenetek tábla II. rész	156
102)	ábra: Tevékenységek tábla.....	157
103)	ábra: Felhasználói biztonsági kulcsok tábla.....	157
104)	ábra: Feladatok tábla.....	157
105)	ábra: Feladatokhoz tartozó üzenetek, megjegyzések tábla	157
106)	ábra: Munkacsoportok tábla.....	158
107)	ábra: Munkacsoportokhoz rendelt felhasználók tábla	158
108)	ábra: Adatbázis diagram	159