

Baranya Vármegyei Szakképzési Centrum
Simonyi Károly Technikum és Szakképző Iskola

Vizsgaremek

Osztály: 13.C

Készítették: Németh Noel

Lőrincz Noel

Rapcsák János

Konzulens: Fenyvesi János Adrián

Pécs

2025

Baranya Vármegyei Szakképzési Centrum
Simonyi Károly Technikum és Szakképző Iskola

Szakma megnevezése: Szoftverfejlesztő és –tesztelő

A szakma azonosító száma: 5 0613 12 03

Vizsgaremek

Aliasly

Osztály: 13.C

Készítették: Németh Noel

Lőrincz Noel

Rapcsák János

Konzulens: Fenyvesi János Adrián

Pécs

2025

2025

EREDETISÉGI NYILATKOZAT

Alulírottak: Németh Noel, Lőrincz Noel, Rapcsák János

A Baranya Vármegyei SzC Simonyi Károly Technikum és Szakképző Iskola, Szoftverfejlesztő és tesztelő végzős tanulói, büntetőjogi és fegyelmi felelősségünk tudatában nyilatkozunk és aláírásunkkal igazoljuk, hogy a(z)

Aliasly

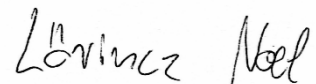
című vizsgaremek saját, önálló munkánk, az abban hivatkozott szakirodalom felhasználása a forráskezelés szabályai szerint történt.

Tudomásul vesszük, hogy vizsgaremek esetén plágiumnak számít:

- szó szerinti idézet közlése idézőjel és hivatkozás megjelölése nélkül;
- tartalmi idézet hivatkozás megjelölése nélkül;
- más publikált gondolatainak saját gondolatként való feltüntetése.

Alulírott kijelentjük, hogy a plágium fogalmát megismertük, és tudomásul vesszük, hogy plágium esetén a vizsgaremekünk visszautasításra kerül.







Tanulók aláírásai

Tartalomjegyzék

I.	Bevezetés.....	5
1.	Projekt bevezető.....	5
2.	Fejlesztési folyamat és Tapasztalatok.....	5
3.	Csapat és Munkamegosztás.....	6
4.	Műszaki megvalósítás.....	6
5.	Következő lépések	6
II.	Felhasználói dokumentáció.....	7
1.	A program általános specifikációja	7
2.	Rendszerkövetelmények	9
3.	A program letöltése és kicsomagolása	9
4.	A program használatának a részletes leírása	13
III.	Fejlesztői dokumentáció	16
1.	Témaválasztás indokolása	16
2.	Az alkalmazott fejlesztői eszközök	16
3.	Tervezési módszer	20
4.	Adatmodell leírása	22
5.	Részletes feladat algoritmusok és forráskód.....	25
6.	Tesztelési dokumentáció.....	33
7.	Továbbfejlesztési lehetőségek	33
IV.	Összegzés	34
	Irodalomjegyzék, forrásmegjelölés	35

I. Bevezetés

A projekt ezen a publikus GitHub repository-n található:

<https://github.com/yAn2049/Aliasly-Password-Manager>

1. Projekt bevezető

1.1 Megoldandó feladat és probléma

A modern vállalati környezetekben gyakori, hogy egy dolgozónak több céges fiókja is van. Ezek kezelése nehézkes, időigényes, és biztonsági kockázatot jelenthet, ha a jelszavak vagy hozzáférési jogok nem megfelelően vannak kezelve. A projekt célja egy olyan centralizált fiókkezelő alkalmazás létrehozása, amely segíti a rendszergazdákat a felhasználói adatok biztonságos és átlátható kezelésében.

1.2 Miért ezt választottuk

A vállalati adatvédelem egyre nagyobb hangsúlyt kap, de a meglévő eszközök gyakran túlbonyolítottak vagy költségesek. Saját fejlesztéssel rugalmasan lehet alkalmazkodni a cégek egyedi igényeihez.

A csapatunk számára lehetőség volt a biztonsági protokollok (pl. titkosítás, szerepkör-alapú hozzáférés) és a skálázható architektúra tervezésének gyakorlati alkalmazására.

1.3 Célunk

Egy egyszerű, intuitív felület biztosítása a rendszergazdáknak. Biztonság: AES-256 CBC titkosítás és mesterkulcsos beléptető rendszer alkalmazása. Felhasználóbarát felület: Intuitív kezelőfelület a rendszergazdák számára. Skálázhatóság: Több ezer felhasználó kezelése kis és közepes vállalatoknál.

2. Fejlesztési folyamat és Tapasztalatok

2.1 Mit tanultunk és új ismeretek

Közös döntéshozatal: Rájöttünk, hogy a különböző szakmai háttérrel rendelkező csapattagok (pl. backend vs. frontend fejlesztők) más szempontokat hangsúlyoznak. Például egy biztonsági funkció implementálása során a backend csapat a teljesítményre, míg a frontend a felhasználói élményre fókuszált. Rendszeres design sprint ülésekkel sikerült közös nevezőre jutni.

Feedback kultúra: A napi minimum 15 perces meetingek segítettek a blokkoló problémák gyors feloldásában.

Konfliktuskezelés: Megtanultuk, hogy a technikai nézeteltérések nem személyesek, hanem a projekt minőségét szolgálják

3. Csapat és Munkamegosztás

3.1 Csapatszerepek

Németh Noel – Backend; Adattitkosítás; Adatbázis: N. Noel a programon belüli adatbázis kezelési funkciókat tervezte és implementálta, emellett az adatbázis finomításán is dolgozott. Ő volt a felelős az adat titkosítás teljes kivitelezésére.

Lőrincz Noel – Frontend; Design; Backend: L. Noel a frontend Design és UI kialakításához készített munkát, az átláthatóság és könnyedén megérthető UI érdekében. A backend részen is hozzáadott a projekthez a funkcionalitás terén.

Rapcsák János – Frontend; Backend; Adatbázis; Dokumentáció: János készítette el az applikáció felhasználói felületét és azoknak a funkcióit a háttérkódban, valamint a kód logikáján is dolgozott. Végül, ő tervezte és készítette el az adatbázis felépítését. A dokumentációt szerkesztette össze és formázta meg.

4. Műszaki megvalósítás

4.1 Architektúra

Backend: A szerveroldali logikát a C# .NET 8.0 keretrendszerrel fejlesztjük.

Frontend: A kliensoldali felület Windows Presentation Foundation (WPF) keretrendszerrel készül. Data Binding és XAML használatával a dinamikus felületek kialakításához.

Adatbázis: Az adatok tárolására MySQL relációs adatbázist használunk, amely megbízható és nagy teljesítményű megoldást nyújt. A phpMyAdmin felület segítségével könnyedén kezelhetők az adatbázisok.

4.2 Biztonsági intézkedések

Adattitkosítás: Az adatok védelmére AES-256 CBC titkosítást használunk.

Hozzáférés-vezérlés: A nem kívánt belépések megelőzéséhez egy mesterkulcsos belépés rendszert hoztunk létre.

Naplózás: Minden fontos műveletet pontosan és átlátható módon naplózunk az adatbázison belül (pl. bejelentkezések, adatmódosítások).

5. Következő lépések

UI és Design továbbfejlesztése: Tisztában vagyunk vele, hogy a design és UI egy kicsit gyenge, ezt szeretnénk sokkal jobban kivitelezni.

Multi-Faktor Hitelesítés (MFA): A rendszergazdák számára kötelezővé tehető MFA bevezetése.

Saját lokális adatbázisra áttérés: A biztonságosabb adattárolás érdekében jobb ötletnek látjuk a lokális és saját adatbázis használatát.

Több platformra térés: Szeretnénk kifejleszteni az applikációt Linux, MacOS és Mobile platformokra is.

II. Felhasználói dokumentáció

1. A program általános specifikációja

1.1 Áttekintés

Az Aliasly egy olyan szoftveralkalmazás, amelyet kifejezetten rendszergazdák számára fejlesztünk ki, hogy biztonságosan kezelhessék a felhasználói fiókokat. A program egy mesterkulcs alapú hitelesítési rendszerrel védi a hozzáférést, lehetővé téve a felhasználói adatok könnyű kezelését, új fiókok tárolását és meglévők törlését. A felület egyszerű és intuitív, így minimális betanulási idővel használható. A rendszer három részből áll:

- Bejelentkezési felület, ahol a rendszergazda mesterkulccsal azonosítja magát.
- Új mesterkulcs létrehozási felület, ahol a rendszergazda létrehozhat egy vagy több különböző kulcsot.
- Főfelület, ahol a felhasználói adatok megjelennek, szerkeszthetők és törölhetők.

A program célja, hogy egy központi helyen biztosítsa a felhasználói adatok biztonságos tárolását és kezelését, miközben a lehető legegyszerűbb működést nyújtja.

1.2 Mesterkulcs kezelése

A program elindításakor a felhasználó egy bejelentkezési képernyővel találkozhat, ahol meg kell adnia a mesterkulcsot. Ha ez az első indítás, és még nincs mentve mesterkulcs, a rendszer felkínálja annak létrehozását. A kulcsot ezután biztonságosan titkosítva tárolja az adatbázisban.

Ha a felhasználó helyes mesterkulcsot ad meg, a program betölti a főfelületet, ahol az összes hozzá tartozó felhasználói fiók látható. Hibás kulcs esetén a rendszer figyelmeztető üzenetet jelenít meg, és lehetőséget ad újra próbálkozásra.

1.3 Felhasználói fiókok kezelése

A főfelületen a rendszergazda áttekintheti az összes tárolt felhasználói fiókot. Az adatok táblázatos formában jelennek meg, lehetővé téve a gyors áttekintést. Minden felhasználóhoz tartozik egy egyedi azonosító, felhasználónév, jelszó, további metaadatok (például. weboldal url, leírás), valamint egy törlés gomb.

Az új felhasználó hozzáadásához a bal oldalon egy űrlap található, ahol meg kell adni a szükséges adatokat (például felhasználónév, jelszó, e-mail cím). A rendszer ellenőrzi, hogy minden kötelező mező ki legyen töltve. Az "Adat felvitele" gombra kattintva az új felhasználó hozzáadódik az adatbázishoz, és azonnal frissül a lista.

Ha egy felhasználót törölni kell, a megfelelő sorban lévő törlés gombra kattintva a rendszer megerősítést. A művelet végleges, így a felhasználó adatai visszavonhatatlanul törlődnek.

1.4 Kilépés

A főfelületen található egy kilépés gomb, amelyre kattintva a rendszer visszairányítja a felhasználót a bejelentkező képernyőre. A kilépéskor a rendszer ellenőrzi, hogy minden változtatás mentésre került-e, és biztonságosan zárja le a munkamenetet.

1.5 Biztonsági intézkedések

A program kiemelt figyelmet fordít az adatvédelemre. A mesterkulcsot és a felhasználó összes adatát titkosítva tárolja a program, így akár adatbázis-szintű illetéktelen hozzáférés esetén is védve maradnak.

1.6 Naplózás

A naplózási rendszer minden jelentős eseményt rögzít, kezdve a belépési és kilépési műveletekkel. Amikor egy rendszergazda sikeresen bejelentkezik a rendszerbe, a napló pontosan rögzíti a belépés időpontját, a használt mesterkulcshoz tartozó azonosítót, valamint a munkamenet egyedi azonosítóját. A kilépéskor a rendszer automatikusan menti a kilépés pontos időpontját.

Az adatmódosítások részletes dokumentálása a naplózási rendszer másik alapvető feladata. Új felhasználói fiók létrehozásakor a rendszer pontosan rögzíti a létrehozás időpontját, az új felhasználóhoz tartozó azonosítót és a műveletet végző rendszergazda azonosítóját. Felhasználói fiók törlésekor a napló rögzíti a törlés időpontját, a fiók azonosítóját és a használatban lévő mesterkulcs azonosítóját.

1.7 Felhasználói felület

A program célja, hogy a lehető legegyszerűbb legyen a használata. A felület egyszerű, a fontos funkciók könnyen elérhetők, és minden művelethez egyértelmű utasítások vagy eszköztípek jelennek meg. A lista nézet lehetővé teszi a gyors szűrést és rendezést, így nagy számú felhasználó kezelése is hatékony marad.

1.9 Összefoglalás

Ez a specifikáció részletesen leírja az Aliasly működését, biztonsági mechanizmusait és felhasználói felületét. A program célja, hogy a rendszergazdák számára hatékony és biztonságos eszközt nyújtson a felhasználói fiókok kezelésére, miközben a lehető legegyszerűbb marad a használata. A továbbfejlesztési lehetőségek garantálják, hogy a rendszer a jövőben is igazodhasson a változó igényekhez.

2. Rendszerkövetelmények

2.1 Hardverkövetelmények

Minimum:

Processzor: Intel Core i3 2.Generációs vagy AMD Ryzen 3 2.Generációs

Memória: 2 GB ddr3 RAM

Videókártya: Integrált

Tárhely: 2 – 6 GB Szabad tárhely

Ajánlott:

Processzor: Intel Core i5 5.Generációs vagy AMD Ryzen 5 3.Generációs

Memória: 4 – 8 GB ddr3 vagy ddr4 RAM

Videókártya: Integrált vagy Dedikált

Tárhely: 4 – 8 GB Szabad tárhely

2.2 Szoftverkövetelmények

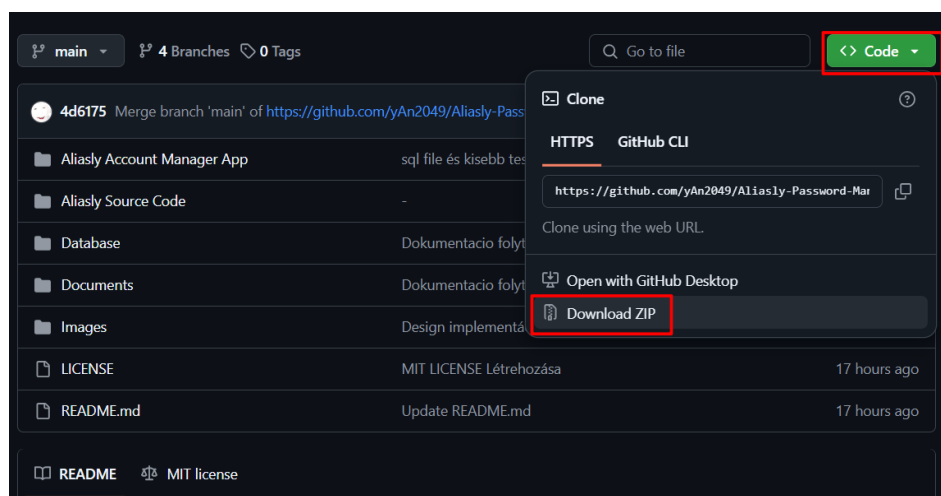
Operációs rendszer: Windows 10 vagy Windows 11

.Net keretrendszer: .NET Runtime 8.0.15 vagy .NET Runtime 9.0.4

Adatbáziskezelés: XAMPP 8.2.12 vagy MAMPP 4

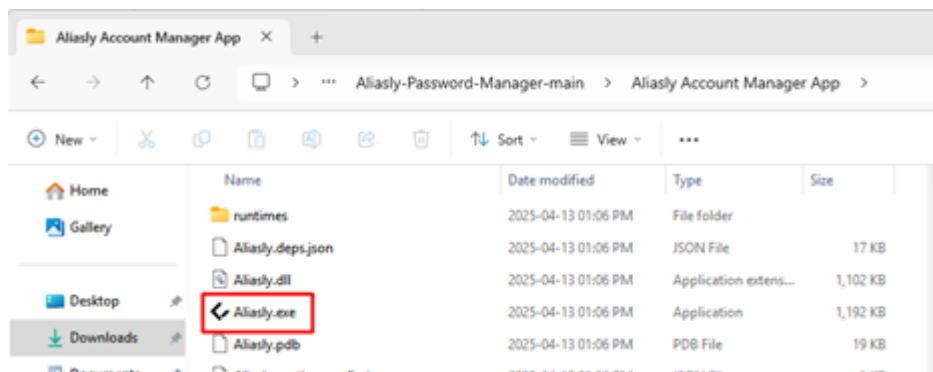
3. A program letöltése és kicsomagolása

Maga a program nem igényel telepítést a jelenlegi állapotában, elég csak letölteni a GitHub repository .zip mappát a következő linken keresztül: <https://github.com/yAn2049/Aliasly-Password-Manager> .



1. ábra: Az Aliasly .zip mappa letöltése a GitHub felületéről

Ha letöltötte a mappát, csomagolja ki egy általa megjelölt meghajtóra. Ehhez WinRAR vagy 7Zip ajánlatos. A program futtató fájlja az Aliasly Account Manager App mappán belül található, 'Aliasly.exe' néven.



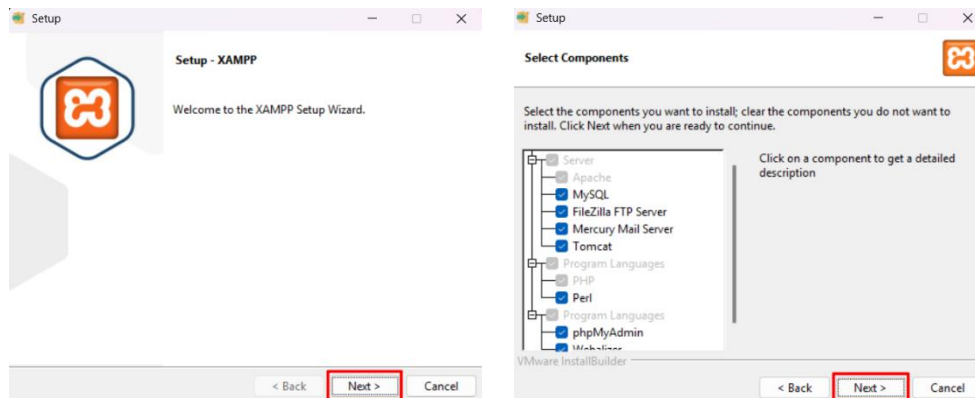
2. ábra: Aliasly.exe az említett mappán belül.

A program működéséhez szükséges egy már említett adatbáziskezelő rendszer. Ehhez a XAMPP vagy MAMPP programot ajánljuk. A következő linkeken tölthetőek le (csak az egyikre van szükség):

XAMPP: <https://www.apachefriends.org/download.html>

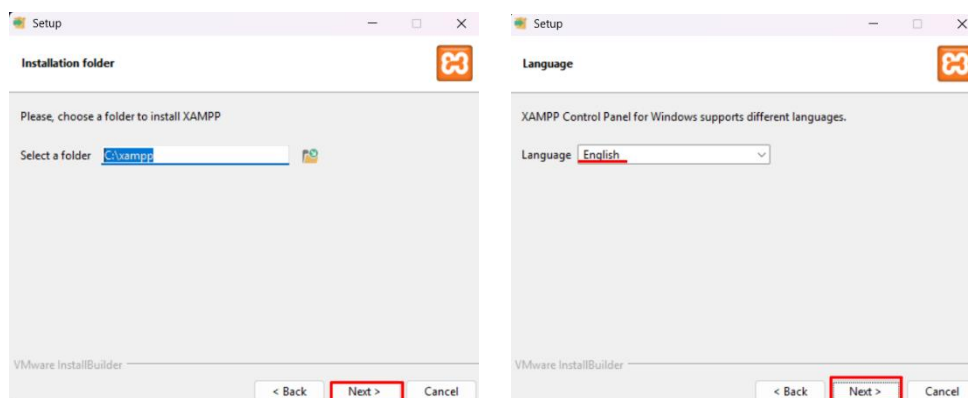
MAMPP: <https://www.mamp.info/en/downloads/>

A telepítés során elég, ha az alapértelmezett telepítést elvégzi.



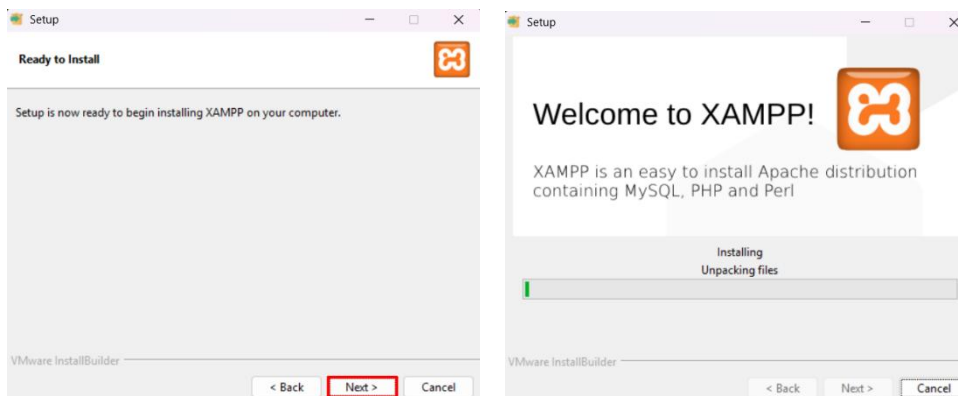
3. és 4. ábra: XAMPP telepítési folyamata. Kezdőlap és komponensek.

A komponensek közül nem ajánlott változtatni kivéve, ha pontosan tudja mit változtat.



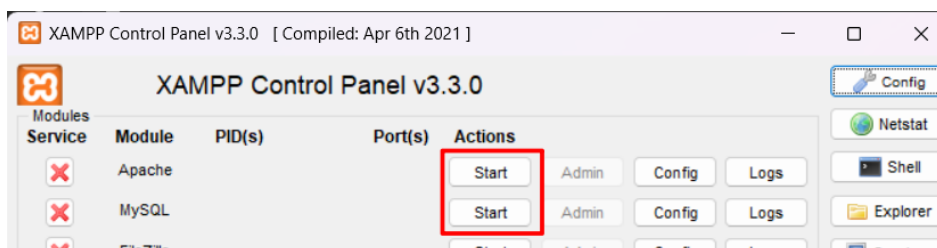
5 és 6. ábra: XAMPP telepítési folyamata. Elérési útvonal és nyelv.

A meghajtó és a mappa elérési útvonalát ne változtassa meg, ez a megfelelő működés érdekét szolgálja. A program megjelenítési nyelvét válassza ki.



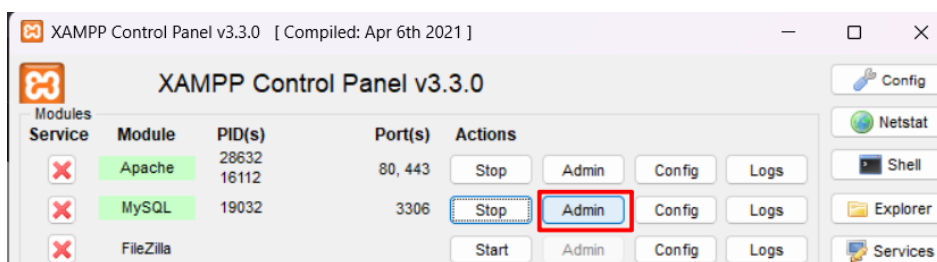
7 és 8. ábra: XAMPP telepítési folyamata. Telepítés véglegesítése.

Miután telepítette az ön által választott adatbáziskezelőt, indítsa el az Apache és MySQL szolgáltatásokat az adatbáziskezelő alkalmazáson belül.



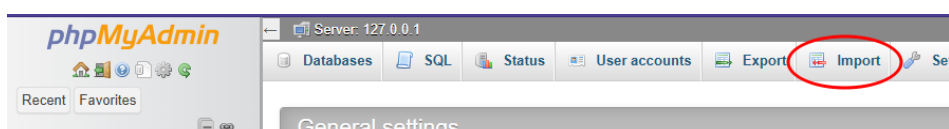
9. ábra: Apache és MySQL szerviz elindítása.

Ezután a MySQL Admin gombra nyomva vagy a böngészője keresőmezőjébe írja be a következőt: ha XAMPP-ot használ: <http://localhost/phpmyadmin/>, ha MAMPP-ot használ: <http://localhost/phpmyadmin5/>. Ezzel megnyitja a phpMyAdmin adatbáziskezelő felületét.



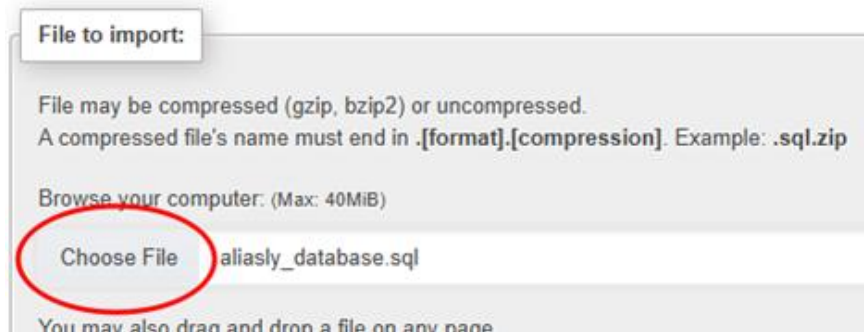
10. ábra: phpMyAdmin felületére belépő gomb.

Ha sikeresen hozzáfért a phpMyAdmin felületéhez, a következőt kell tennie. A GitHub repository-ból töltsse le az 'aliasly_database.sql' fájlt. Ezután a phpMyAdmin felületén lépjen az **Import** felületre.

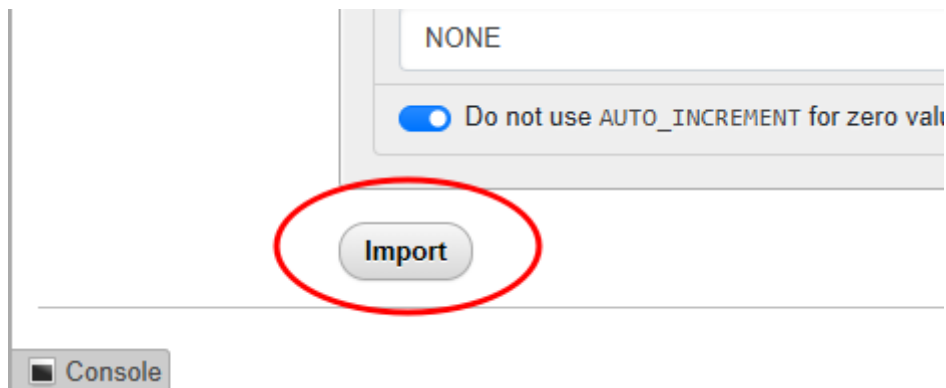


3. ábra: phpMyAdmin felületén hol található az Import felület.

Ezután válassza ki az 'aliasly_database.sql' fájlt, majd ezt követően a felület legalján találja az 'Import' gombot, amivel létrehozhatja az adatbázis struktúrát.



11. ábra: Az Import felületen található fájl kiválasztása gomb.

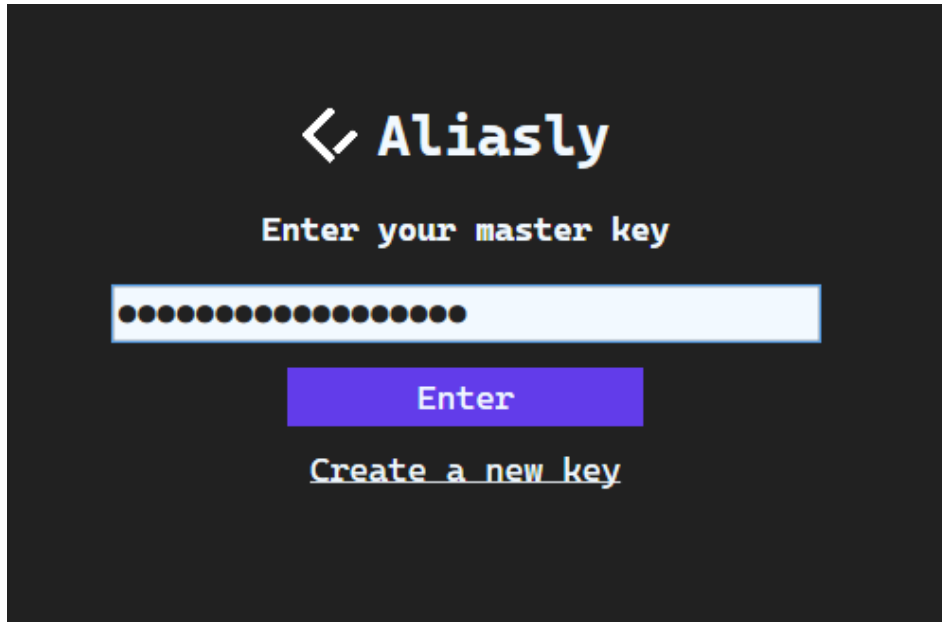


12. ábra: Az Import felület alján található Import gomb.

Az utolsó dolog, ami szükséges a program működéséhez a megfelelő .Net keretrendszer. A programunk a .Net 8.0 keretrendszerrel lett létrehozva. Ehhez a letöltési link a következő: <https://dotnet.microsoft.com/en-us/download/dotnet/8.0> Miután letöltötte a .Net 8.0 keretrendszer telepítőjét, ezt futtassa le és kövesse az alapértelmezett telepítési folyamatot. De ezt nem feltétlen muszáj megtennie, mert ha lefuttatja az applikációt, automatikusan megkéri, hogy töltsse le a .Net keretrendszert.

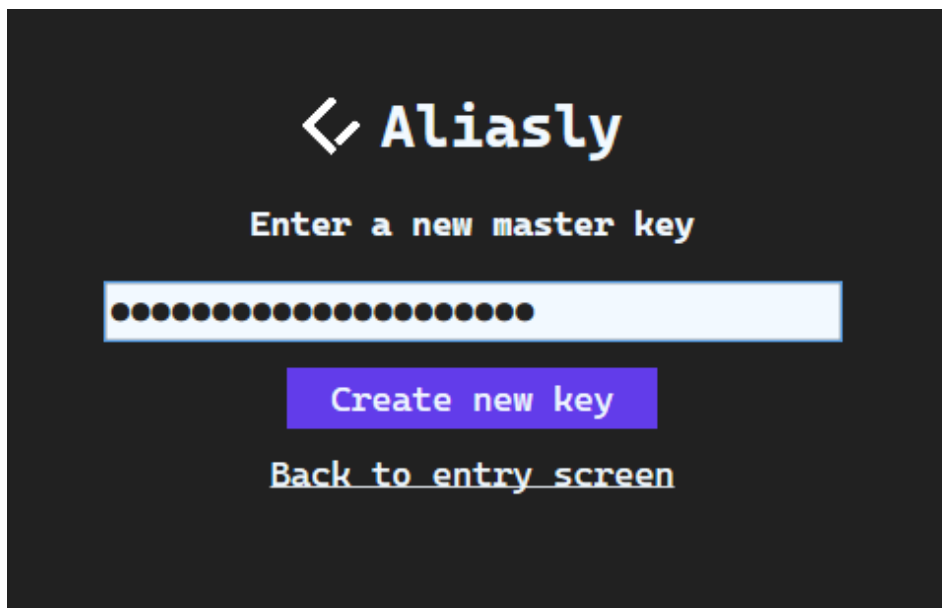
4. A program használatának a részletes leírása

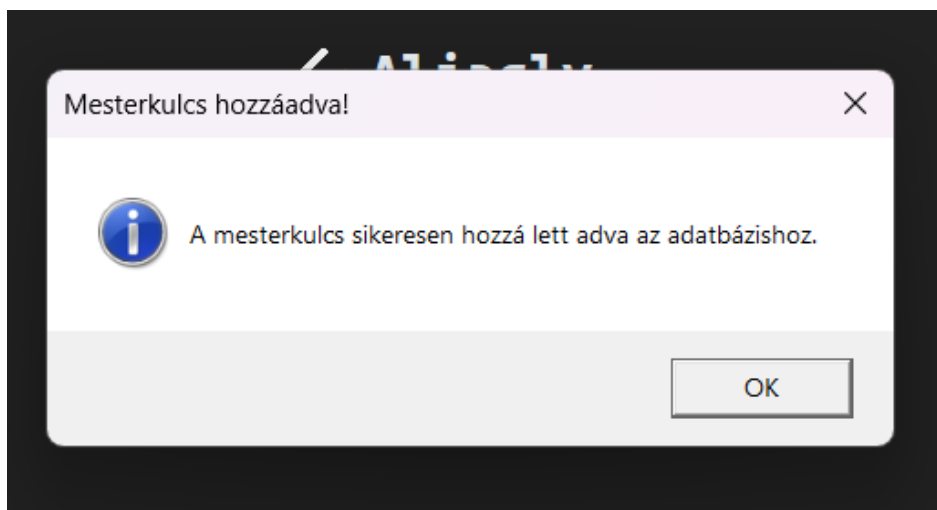
A program indításakor egy mesterkulcsos belépő mező fogad. Ha már használtad a programot akkor egyszerűen be kell ide írnod a már létező mesterkulcsodat. Ha viszont első alkalommal használod az alkalmazást, akkor a 'Create a new key' szövegre nyomva, át léphetsz egy hasonló viszont más mezőre.



13. ábra: mesterkulcs belépő felület

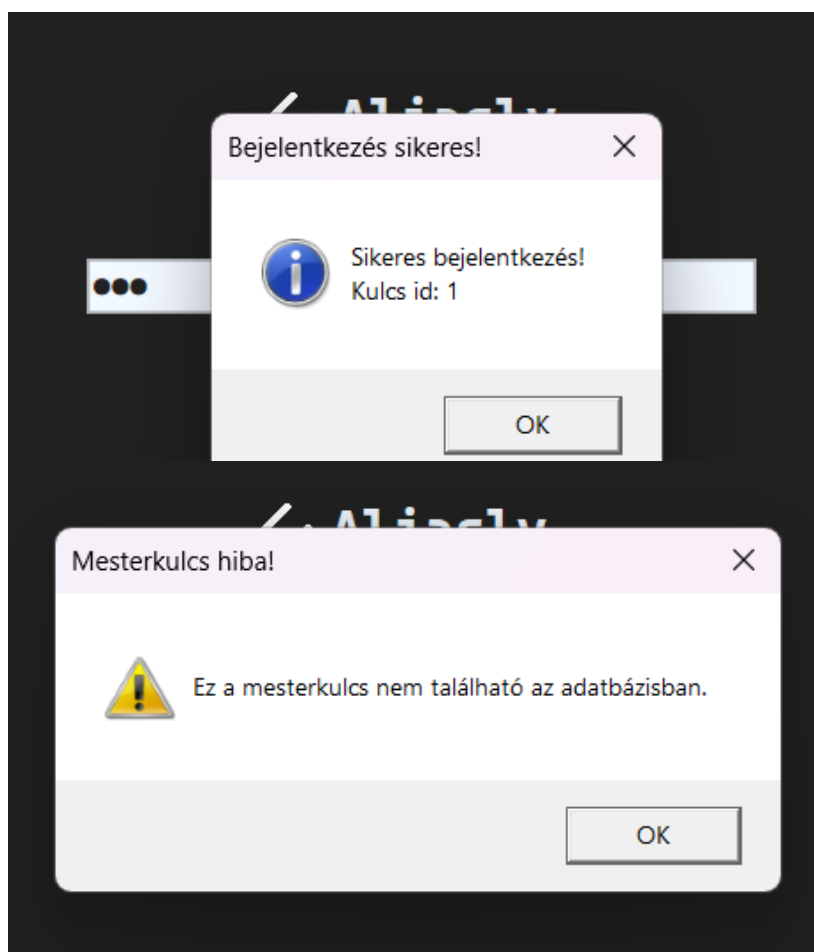
Ezen a felületen létrehozhatod az első mesterkulcsodat, viszont, ha szeretnél, akkor akármennyi mesterkulcsot létrehozatsz. Miután beírtad a kívánt mesterkulcsodat, a 'Create new key' gombra kattintva, ez a kulcs felkerül az adatbázisba és használhatóvá válik.





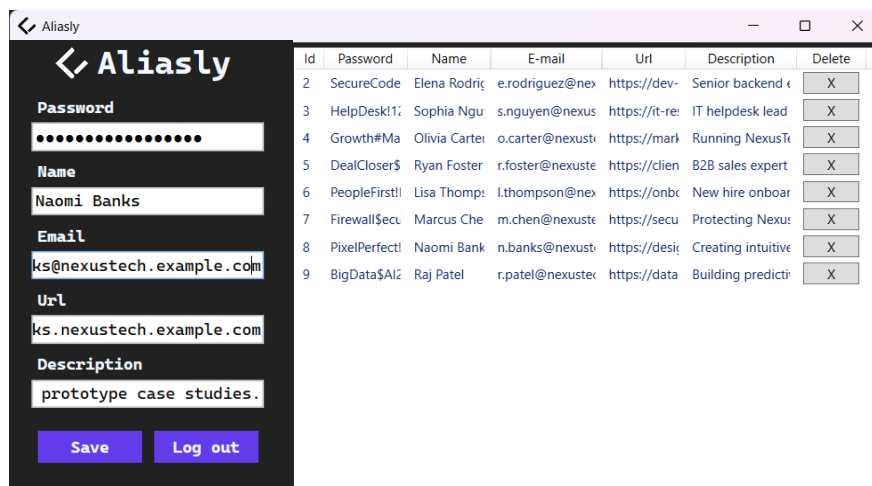
14. és 15. ábra: mesterkulcs létrehozó felület és sikeres mesterkulcs létrehozva üzenetmező.

A '**Back to entry screen**' szövegre kattintva visszaléphet a belépés felületre, ahol, a létrehozott kulccsal beléphet az alkalmazás fő felületére.



16. és 17. ábra: Sikeres és sikertelen bejelentkezés üzenetmező

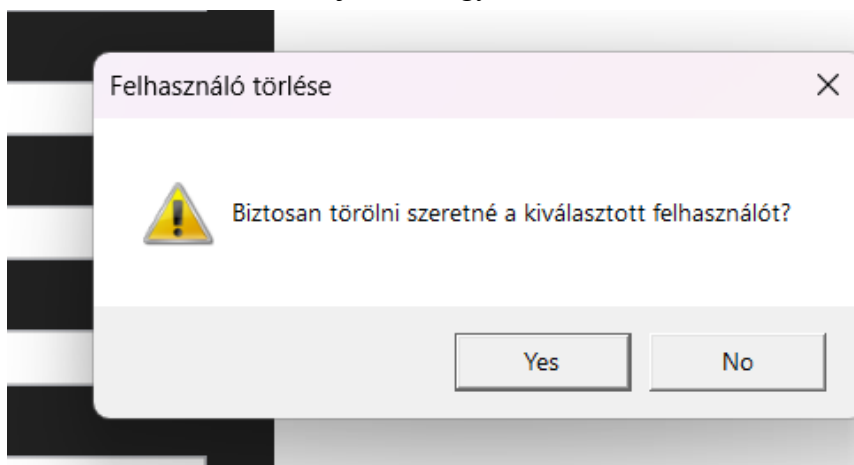
A sikeres belépés után az applikáció fő felülete fog fogadni. Ezen a felületen látható a felhasználói adatok listája, az űrlap, ahova írhatod be a felhasználó adatait, egy rögzítés (**Save**) gomb és egy kilépés (**Log out**) gomb.



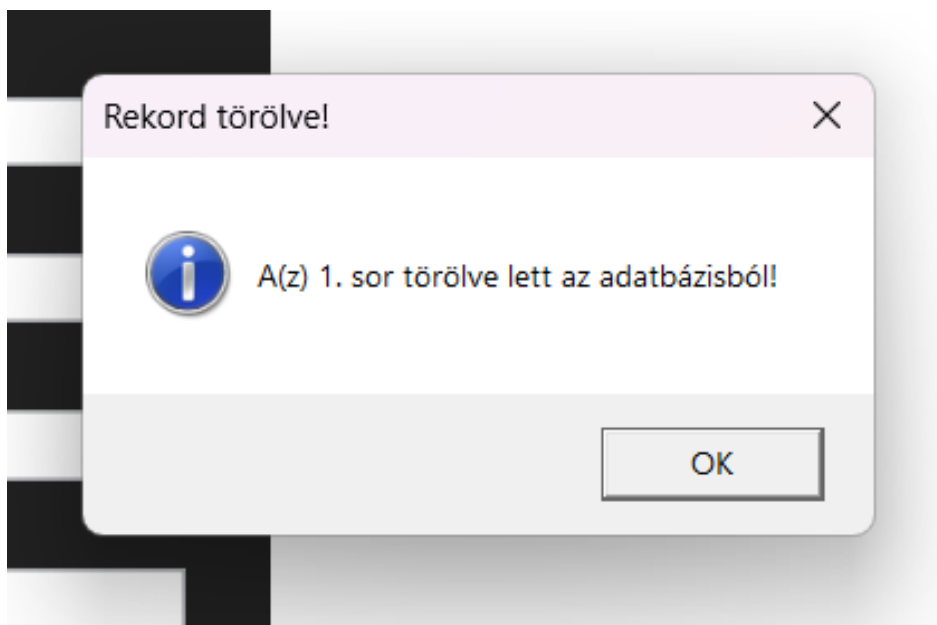
18. ábra: az alkalmazás fő felülete.

A bal oldalon található egy **űrlap**, ahova a felhasználók adatait lehet beírni, ha az adatokat rögzíteni szeretnénk az adatbázisban és a listában, a 'Save' gombra nyomva érhetjük ezt el. Ha törölni szeretnénk adatokat, minden sor végén található egy törlés gomb. Ezt megnyomva megkérdi a program, hogy biztosan törölni szeretné-e az adatokat.

19. ábra: Visszajelzés, hogy biztosan törli-e az adatot.



Az igenre nyomva törli annak a felhasználónak az adatait. Ez a művelet egyelőre végleges, tehát nem lehet visszavonni a törlésre került adatokat.



20. ábra: Üzenetmező a törlésre került sorról.

Végül, ha ki szeretne lépni a fő felületről, a 'Log out' gomb megnyomásával teheti ezt meg.

III. Fejlesztői dokumentáció

1. Témaválasztás indokolása

Mint sok más embernek, számunkra is problémát tud okozni a saját jelszavainknak és azokhoz tartozó adatoknak a megjegyzése, nem beszélve arról, ha néhány vagy több tíz embernek az adatait is nekünk kell valahol tárolnunk. Ezt valaki papíron, valaki a telefonja vagy számítógépének a jegyzetömbjében, valaki ragacsos jegyzeteken a falára ragasztva tárolja.

Ezek a módszerek nem mindig biztonságosak és sokszor nem tisztán követhetőek. Ezért mi szeretnénk egy olyan egyszerűen használható, de ennek ellenére biztonságos alkalmazást létrehozni, amely ezeket az adatokat tárolja el helyettünk. Ez az alkalmazás lenne az Aliasly.

2. Az alkalmazott fejlesztői eszközök

2.1 Programozási nyelv

C#:

A C# egy modern, objektum-orientált programozási nyelv. A nyelv erősen típusos, sokoldalú és könnyen tanulható, miközben kiváló teljesítményt és biztonságot nyújt. A C# különösen alkalmas asztali alkalmazások fejlesztésére, főleg Windows környezetben, ahol a Visual Studio fejlett fejlesztői eszközeivel és a WPF (Windows Presentation Foundation) vagy WinForms keretrendszereivel egyszerűen lehet hatékony, felhasználóbarát felületeket létrehozni.

A nyelv egyik legnagyobb előnye a .NET keretrendszerrel való szoros integrációja, amely gazdag könyvtárat és előre megírt funkciókat kínál, megkönnyítve az adatkezelést, hálózati kommunikációt vagy grafikus megjelenítést. A memóriakezelésben a Garbage Collector automatikusan kezeli a felesleges erőforrásokat, csökkentve a memóriaszivárgás kockázatát. Emellett a C# támogatja az aszinkron programozást, ami lehetővé teszi a zökkenőmentes, blokkolás nélküli felhasználói élményt.

A Windows-alapú asztali alkalmazások terén a C# és a .NET kiemelkedő kompatibilitást és stabilitást nyújt. A Visual Studio támogatása pedig gyors hibakeresést és hatékony fejlesztési folyamatot tesz lehetővé. Mindezek miatt a C# ideális választás olyan asztali szoftverek készítéséhez, amelyeknek robosztusnak, skálázhatónak és karbantarthatónak kell lenniük hosszú távon.

C# WPF:

A WPF (Windows Presentation Foundation) a Microsoft modern UI keretrendszere, amely a C# nyelvvel együtt kiváló választás asztali alkalmazások fejlesztéséhez, főleg akkor, ha komplex, vizuálisan gazdag felhasználói felületekre van szükség. A WinForms egyszerűsége és gyors prototípuskészítési lehetőségei ellenére a WPF számos területen felülmúlja, különösen a modern szoftverek igényeit figyelembe véve.

A WPF legfőbb erőssége az XAML-alapú felületdefiníció, amely lehetővé teszi a felületek elkülönítését az üzleti logikától, így a dizájnerek és a fejlesztők hatékonyabban tudnak együttműködni. A vektoros grafikus renderelés és a felbontásfüggetlen megjelenítés révén a WPF alkalmazások élesek és jól skálázódnak különböző képernyőméreteken, ami különösen fontos a mai nagyfelbontású és érintőképernyős eszközök korában.

A databinding rendszere kifinomultabb, mint a WinForms-ban, lehetővé téve az adatok és a felület közötti dinamikus kapcsolatot minimális kóddal. Emellett a stílusok, animációk és egyéni vezérlőelemek készítése sokkal rugalmasabb, ami lehetővé teszi igényes design megvalósítását. A kompozíciós modell (pl. ControlTemplate, DataTemplate) lehetővé teszi a meglévő vezérlők teljes átalakítását anélkül, hogy újakat kellene írni.

2.2 Adatbáziskezelés

MySQL (phpMyAdmin):

A MySQL egy nyílt forráskódú, megbízható relációs adatbázis-kezelő rendszer (RDBMS), amely széles körben használatban van webfejlesztésben és asztali alkalmazásokban egyaránt. A phpMyAdmin pedig egy webalapú felülete a MySQL adatbázisok kezeléséhez, ami intuitív módon lehetővé teszi táblák létrehozását, adatok módosítását és lekérdezések futtatását anélkül, hogy közvetlen SQL parancsokat kellene írni.

A MySQL egyik legnagyobb előnye a teljesítménye és skálázhatósága, ami lehetővé teszi gyors adatkezelést akár nagy terhelés mellett is. Támogatja a tranzakciókat, tárolt eljárásokat és triggereket, így komplex üzleti logika is könnyen implementálható. Emellett a magas rendelkezésre állás és biztonság jellemzi, különösen nagyvállalati környezetekben, ahol az adatvédelem kulcsfontosságú.

A phpMyAdmin egyszerűsíti az adatbázis-kezelést, mivel grafikus felületen keresztül lehet táblákat szerkeszteni, indexeket létrehozni vagy adatokat exportálni/importálni. Bár nem olyan teljesítményorientált, mint egy natív SQL klienst használni, kiváló választás fejlesztők és rendszergazdák számára, akik gyorsan szeretnének módosításokat végezni vagy adatbázis-struktúrákat áttekinteni.

2.3 Verziókezelés

GitHub:

A GitHub a modern szoftverfejlesztés egyik alapvető eszköze, amely nem csupán verziókövetést biztosít, hanem egy teljes ökoszisztémát teremt a hatékony együttműködéshez. A Git technológiára épülve lehetővé teszi, hogy a fejlesztők pontosan dokumentálják a kód változásait, miközben rugalmasan dolgozhatnak különböző funkciókon párhuzamosan. A platform legfőbb értéke, hogy összeköti az egyéni munkát a csapatban történő fejlesztéssel, így akár tízezrek is részt vehetnek egy projekt fejlesztésében anélkül, hogy zavarnák egymást.

A GitHub előnyeit leginkább a csapatmunkában érvényesülnek. A fejlesztők saját ágakon dolgozhatnak, majd a változtatásaikat pull request formájában beadhatják áttekintésre. Ez a folyamat nemcsak a kódminőséget növeli a kollektív ellenőrzés révén, hanem lehetővé teszi a tudásmegosztást is, hiszen a csapattagok láthatják egymás munkáját és tanulhatnak belőle. A problémák nyomon követése és a feladatok rendszerezése pedig segít abban, hogy mindenki tisztában legyen a projekt aktuális állapotával és a következő lépésekkel.

2.4 Projekt menedzsment szoftver

Discord:

A Discordot hatékonyan használtuk csapatunk kommunikációjára és együttműködésére a projektfeladat során. Dedikált csoportot hoztunk létre a megbeszéléseknek, ahol rendszeres meetingeket tartottunk a haladásról és a feladatokról. Szöveges kommunikáción keresztül is osztottuk meg az ötleteket, dokumentumokat és feedbacket, míg a hanghívások segítettek az azonnali egyeztetésekben. A tervezési fázisban képernyőmegosztással közösen dolgoztunk a projekten, így zökkenőmentesen haladtunk a projekt elkészítésével.

Excel:

Az Excelt hatékonyan használtuk a projektfeladat nyomon követésére és a feladatok koordinálására. Egy közös táblázatot készítettünk, amelyben részleteztük az elvégzendő feladatokat. A táblázatot folyamatosan frissítettük, így mindig átláthattuk, mi készült el és mi van még hátra. A színes jelölésekkel (zöld = kész, sárga = folyamatban, piros = hátralévő) gyorsan orientálódtunk a haladásban. A megosztott fájl lehetővé tette, hogy valós időben lássuk egymás frissítéseit, így mindig szinkronban maradtunk.

2.5 Modellek

Flowchart:

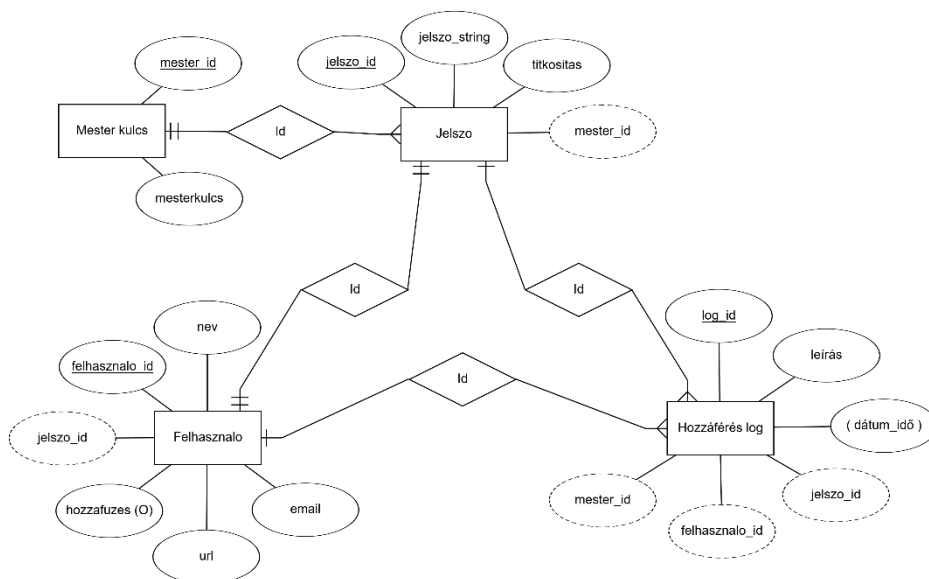
A Flowchart (folyamatábra) elkészítése kiemelten fontos a projektfeladatok tervezési szakaszában, mert vizuálisan ábrázolja a folyamat lépéseit, a döntési pontokat és az adat- vagy vezérlési áramlást. Ez a modell segít áttekinteni a rendszer működését, feltárva a lehetséges hibákat vagy felesleges lépéseket még a fejlesztés megkezdése előtt. A Flowchart egyértelművé teszi, hogy a különböző műveletek hogyan kapcsolódnak egymáshoz, így a fejlesztők és a projekt résztvevői könnyen követhetik a logikai struktúrát.

A folyamatábra különösen hasznos a bonyolultabb algoritmusok vagy folyamatok tervezésekor, mert segít egyszerűsíteni a komplex eljárásokat és azonosítani a kritikus pontokat. Emellett kiváló kommunikációs eszköz a csapat tagjai és az érdekelt felek között, mivel vizuális formában mutatja a munkafolyamatot, így mindenki ugyanazt a modellt veszi alapul. A dokumentáció részeként később is segíthet a rendszer karbantartásában vagy továbbfejlesztésében, hiszen gyors áttekintést nyújt a folyamatról.

ER Diagram:

Az ER Diagram (Entity-Relationship Diagram) létrehozása nélkülözhetetlen lépés az adatbázis tervezése során, mert segít világosan megjeleníteni az entitásokat, azok attribútumait, valamint a köztük lévő kapcsolatokat. Ez a vizuális modell lehetővé teszi, hogy a fejlesztők és a tervezők egyértelműen megértsék az adatstruktúrát, még mielőtt a tényleges implementáció elkezdődne. Az ER Diagram segít elkerülni a redundáns vagy hiányzó adatokat, valamint biztosítja, hogy a táblák logikusan legyenek felépítve, optimalizálva a lekérdezések hatékonyságát.

Emellett az ER Diagram egy hatékony kommunikációs eszköz a csapaton belül, mivel egyszerűen bemutatja, hogyan kapcsolódnak egymáshoz az adatok. Az adatbázis későbbi bővítése vagy módosítása is könnyebbé válik, ha egy jól strukturált diagram áll rendelkezésre, mivel az alapvető logika már megtalálható benne.



20. ábra: ER Diagram ábra

2.6 Visual Studio 2022

A Visual Studio integrált fejlesztői környezetben a C# programozási nyelv és a WPF (Windows Presentation Foundation) keretrendszer kombinációja kiváló alapot nyújtott a projektünk hatékony és strukturált megvalósításához. A C# erősen típusos, objektum-orientált jellegével lehetővé tette a tiszta kódírást és a komplex üzleti logika könnyed implementálását, miközben a WPF segítségével modern, felbontásfüggetlen és vizuálisan vonzó felhasználói felületet építhettünk.

Az objektum-orientált programozás (OOP) elveire történő hangsúlyozott figyelem – az egységbezárás, öröklődés, polimorfizmus és absztrakció alkalmazása – lehetővé tette, hogy a kódunk jól strukturált, könnyen karbantartható és bővíthető legyen.

A Visual Studio fejlett eszközei, például az IntelliSense, a debugger és a XAML dizájnér, pedig jelentősen felgyorsították a fejlesztési folyamatot, lehetővé téve a gyors iterációt és a hibák azonosítását.

3. Tervezési módszer

A szoftverfejlesztés során a tervezési fázis kulcsszerepet játszik a projekt sikerességében. Megfelelő tervezés nélkül a fejlesztés kaotikussá válhat, ami hibákhoz, idővesztéshez és költségtúllépéshez vezethet. A tervezés során meghatározott struktúra és módszertan lehetővé teszi, hogy a csapat egyértelmű irányvonalat kövessen, a kockázatok minimalizálójának, valamint a végtermék megfeleljen a megrendelő elvárásainak. Ráadásul a jól kidolgozott tervek segítenek a kommunikációban is, mivel minden résztvevő tisztában van a feladataival és a projekt céljaival.

Előkészületek és konzultáció

A projekt megkezdése előtt alapos előkészítésre van szükség. Ennek során felmértük a rendelkezésre álló erőforrásokat, a csapat képességeit, valamint a megvalósítandó rendszer környezetét. A konzulens által javasolt módszerek gyakran meghatározóak voltak a tervezési folyamatban. Például agilis módszertan esetén a folyamat iteratív, rugalmasabb, míg a hagyományos vízéses modell esetén a tervezés szigorúbb fázisokra oszlik.

A leggyakrabban alkalmazott tervezési módszerek közé tartozik a prototípus-készítés, amely segítségével korai szakaszban tesztelhető a felhasználói élmény, valamint a moduláris tervezés, amely lehetővé teszi, hogy a rendszer részei egymástól függetlenül fejleszthetők legyenek. Emellett fontos lehet a felhasználóközpontú tervezés, amely során a végfelhasználók igényei kerülnek előtérbe.

Követelmények figyelembevétele

A projekt elkészítésének egyik legfontosabb tényezője, hogy a fejlesztés során betartsuk a kezdetben meghatározott követelményeket. Ehhez szükséges a nyomon követési rendszer kialakítása, amely segítségével ellenőrizhető, hogy a megvalósított funkciók megfelelnek-e az elvárásoknak.

Rendszeres tesztelési fázisok biztosítják, hogy a rendszer hibamentesen működjön, valamint, hogy ne kerüljenek be olyan funkciók, amelyek nem szerepelnek az eredeti specifikációban. Ha változtatásra van szükség, azt mindig dokumentálni kell, és csak a megfelelő jóváhagyás után lehet végrehajtani.

Első lépések a projektben

A projekt indulásakor elsődleges feladat a követelmények gyűjtése és elemzése. Ezt követően kidolgozni a rendszerarchitektúrát, amely meghatározza a komponensek felépítését és kapcsolatát. Fontos továbbá a felhasználói történetek vagy use-case-ek megfogalmazása, amelyek segítenek abban, hogy a fejlesztés során ne vesszünk szem elől a felhasználói igényeket.

Ezek után elkészül a technológiai stack kiválasztása, amely tartalmazza a programozási nyelvet, keretrendszereket, adatbázis-rendszert és egyéb eszközöket. A tervezési dokumentáció ezen szakasza tartalmazza még az időbeosztást, a kockázatelemzést és a minőségbiztosítási tervet.

Fejlesztési ciklusok

A fejlesztési folyamat során különböző életciklus-modellek alkalmazhatók. Az agilis módszertan esetén a munka rövid, ismétlődő sprintekre oszlik, amelyek során folyamatosan értékelik a fejlesztés állapotát. A hagyományos vízéses modell esetén minden fázis (tervezés, implementáció, tesztelés, üzembe helyezés) szigorúan egymás után következik. A választott modell nagyban befolyásolja a tervezés részletességét és a változtatások kezelésének módját.

Implementáció

A tervezési fázis lezárása után megkezdődik a fejlesztés. Ekkor a tervek alapján kialakítottuk a rendszer alapvető komponenseit, majd fokozatosan építettük fel a funkcionalitásokat. Az implementáció során fontos volt, hogy a csapatunk folyamatosan egyeztessen a tervezési dokumentummal, így elkerülve az eltéréseket.

3.1 OOP Megvalósítása

A projekt során az objektumorientált programozás alapelveit követve terveztük és valósítottuk meg a rendszert, hogy biztosítsuk a kód áttekinthetőségét, modularitását és könnyű bővíthetőségét.

Az egységbezárás segítségével elkülönítettük az adatokat és a hozzájuk kapcsolódó műveleteket, így a belső állapot védett maradt, és csak jól definiált interfészekon keresztül módosítható.

Az öröklődés lehetővé tette a közös funkcionalitás kiszervezését alaposztályokba, ami csökkentette a kódismétlést és egységesítette a viselkedést a leszármazott osztályokban.

A polimorfizmus révén különböző osztályok ugyanazon interfész vagy ősosztály metódusait saját igényeikhez igazodóan implementálhatták, rugalmasságot adva a rendszer szerkezetének.

3.2 Projekt Menedzsment

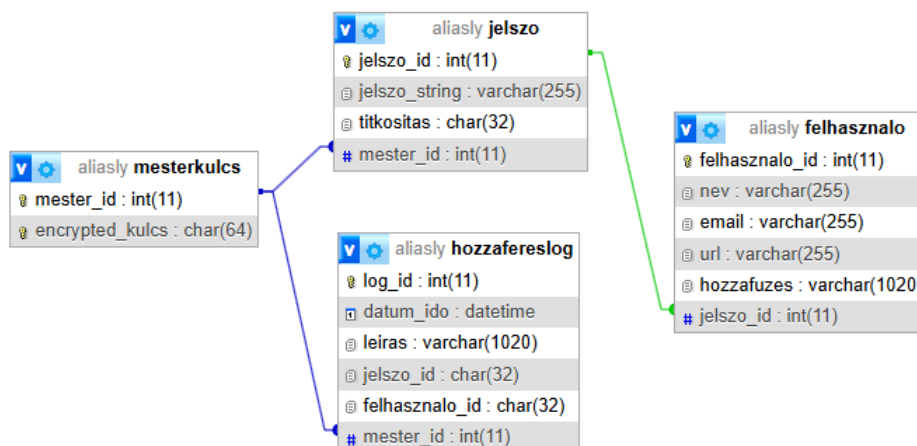
A projekt menedzsment főképp a Discord és Excel alkalmazásokban történt. Excelen belül hoztunk létre egy folyamatábrát és tartottuk tisztában az elkészítendő feladatokat. Discordon létrehoztunk a projektünknek egy csoportot, ahol a fejlesztés folyamán rendszeresen tartottunk meetingeket, ekkor beszéltük át a fejlesztendő és már elkészített feladatokat. A megbeszélések akár egyszerű 20 perces egyeztetések vagy több órás fejlesztési sprintekből álltak.

A munka menetén jegyzeteltünk a chaten és osztottunk meg feladat részleteket, a fontosabb dolgokat kitűztük, hogy később is vissza tudjuk nézni őket.

Annak ellenére, hogy ez egy elfogadható megoldás volt részünkre a jelenlegi projekt elkészítéséhez. Egy nagyobb feladat elkészítésére célszerű egy

4. Adatmodell leírása

Az adatmodell egy projektfeladat alapvető eleme, amely meghatározza, hogy az adatok hogyan strukturálódnak, tárolódnak és kapcsolódnak egymáshoz. Szerepe kiemelkedő, hiszen közvetlen hatással van a rendszer funkcionalitására, a fejlesztés hatékonyságára és a későbbi karbantarthatóságra.



21. ábra: Adatbázis modell a phpMyAdmin designer felületén.

Az adatbázis négy fő táblából áll, amelyek hierarchikus kapcsolatban vannak egymással.

4.1 Táblák és kapcsolatok

1. Mesterkulcs tábla:

Funkció: A rendszer belépéséhez titkosítva tárolja a különböző mesterkulcsokat. Ezek teljesen egyedi kulcsok, nem lehet duplikált entry.

Cellák	Tulajdonságok
mester_id	Elsődleges kulcs, automatikusan növekvő azonosító
encrypted_kulcs	Titkosított mesterkulcs (64 karakteres fix hosszú string)

Jellemzők:

- A mesterkulcs egyedi és kötelező mező
- Az adatok titkosításához szükséges fő kulcsot tárolja

2. Jelszó tábla:

Funkció: A felhasználói jelszavakat titkosítva tárolja.

Cellák	Tulajdonságok
jelszo_id	Elsődleges kulcs, automatikusan növekvő azonosító
jelszo_string	A titkosított jelszó (max 255 karakter)
titkositas	A használt titkosítási módszer (32 karakteres fix hossz)
mester_id	Külső kulcs a Mesterkulcs táblára

Kapcsolatok:

- Egy mesterkulcshoz több jelszó is tartozhat (1:N kapcsolat)
- Ha a mesterkulcs törlődik, a hozzá tartozó jelszavak is törlődnek (ON DELETE CASCADE)

Jellemzők:

- Minden jelszó kötelezően kapcsolódik egy főkulcshoz
- A jelszó tárolása titkosított formában történik

3. Felhasználó tábla

Funkció: A felhasználói fiókok adatait titkosítva tárolja

Cellák	Tulajdonságok
felhasznalo_id	Elsődleges kulcs, automatikusan növekvő azonosító
nev	A felhasználó neve (max 255 karakter)
email	A felhasználó email címe (max 255 karakter)
url	A felhasználó fiókjához tartozó webcím (max 255 karakter)
hozzafuzes	További megjegyzések (max 1020 karakter)
jelszo_id	Külső kulcs a Jelszo táblára

Kapcsolatok:

- Egy jelszóhoz egy felhasználó tartozhat (1:1 kapcsolat)
- Ha a jelszó törlődik, a hozzá tartozó felhasználó is törlődik (ON DELETE CASCADE)

Jellemzők:

- A név, email és url mezők kötelezőek
- A felhasználó mindig kapcsolódik egy jelszóhoz

4. HozzáférésLog tábla

Funkció: A rendszerhez való hozzáférési kísérletek naplózása

Cellák	Tulajdonságok
log_id	Elsődleges kulcs, automatikusan növekvő azonosító
datum_ido	A naplóbejegyzés idő bélyegé (alapértelmezett: aktuális idő)
leiras	A naplózott esemény leírása (max 1020 karakter)
jelszo_id	A naplózott jelszó azonosítója (32 karakteres string)
felhasznalo_id	A naplózott felhasználó azonosítója (32 karakteres string)
mester_id	Külső kulcs a Mesterkulcs táblára

Kapcsolatok:

- Egy mesterkulcshoz több naplóbejegyzés tartozhat (1:N kapcsolat)

Jellemzők:

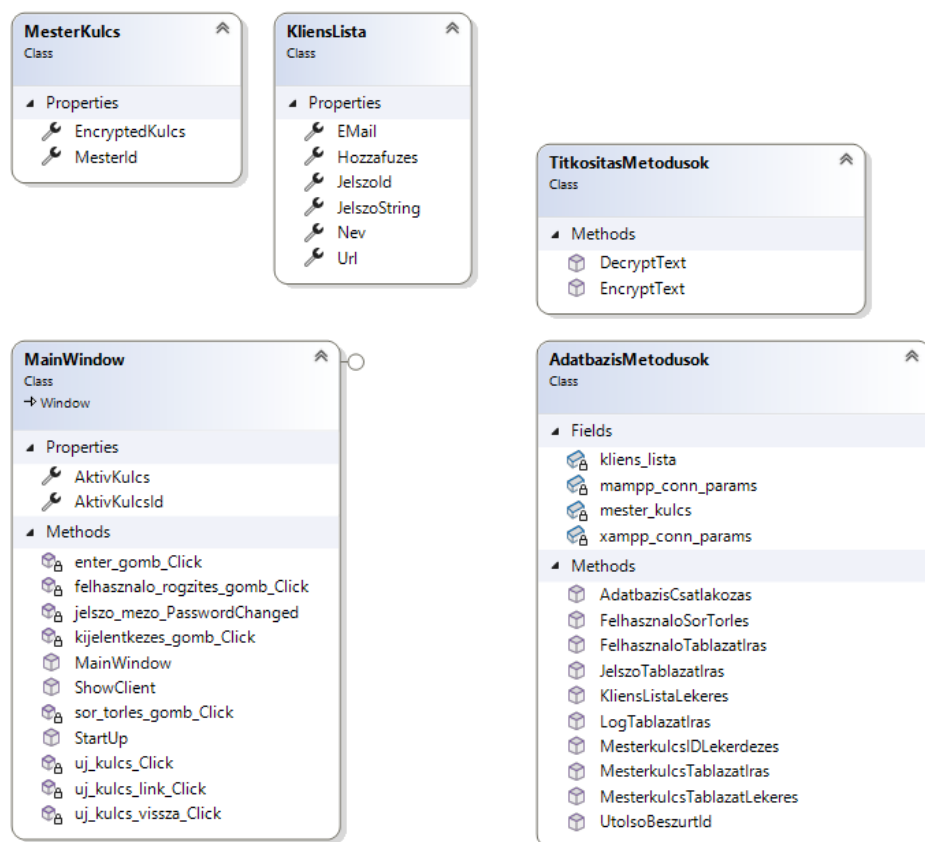
- A dátum és leírás kötelező mezők
- A jelszó és felhasználó azonosítók itt stringként vannak tárolva (nem külső kulcsként)

Adatmodell hierarchia:

A rendszer hierarchikus szerkezetű:

- Mesterkulcs: A legfelső szint, minden más ennek alárendelt.
- Jelszó: Közvetlenül a mesterkulcshoz kapcsolódik.
- Felhasználó: A jelszóhoz kapcsolódik.
- HozzáférésLog: Közvetlenül a mesterkulcshoz kapcsolódik, de referenciát tart a jelszóra és felhasználóra is.

5. Részletes feladat algoritmusok és forráskód



22. ábra: Visual Studio Class Diagram

```

// Sql csatlakozási paraméterek, xampp és mamp //
private string xampp_conn_params = "server=localhost;user=root;database=aliasly;port=3306";
private string mamp_conn_params = "server=localhost;user=root;database=aliasly;port=3306;password=root";

// Adatbázis kapcsolat paraméteradása //
9 references
public MySqlConnection AdatbazisCsatlakozas() // Adatbázis csatlakozás
{
    MySqlConnection db_csatlakozas;
    try
    {
        // XAMPP adatbázis csatlakozás
        db_csatlakozas = new MySqlConnection(xampp_conn_params);
        db_csatlakozas.Open();
    }
    catch
    {
        // MAMP adatbázis csatlakozás
        db_csatlakozas = new MySqlConnection(mamp_conn_params);
        db_csatlakozas.Open();
    }
    return db_csatlakozas;
}

```

23. ábra: Adatbázis kapcsolat létesítő metódus kódrészlet

Az alábbi (23.) ábrán a metódus egy try-catch szerkezet segítségével dönti el, hogy az adatbázishoz XAMPP vagy MAMP kapcsolati paraméterekkel csatlakozzon. Amikor a kódon belül meghívjuk a metódust, ellenőrzi egy try-catch-el hogy a XAMPP vagy MAMP adatbáziskezelőt használja a felhasználó, ezután megnyitja a kapcsolatot az adott paraméterekkel, és visszadobja ezt a kapcsolatot, hogy ahol meg lett hívva, ott használhatóvá váljon.

```

2 references
public List<MesterKulcs> MesterkulcsTablazatLekeres() // Mesterkulcs tábla lekérdezés
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = new AdatbazisMetodusok().AdatbazisCsatlakozas();

    try
    {
        // Tábla SELECT
        string sql_mesterkulcs_select = "SELECT mester_id, encrypted_kulcs FROM mesterkulcs";
        MySqlCommand sql_command_mesterkulcs = new MySqlCommand(sql_mesterkulcs_select, db_csatlakozas);
        MySqlDataReader sql_reader = sql_command_mesterkulcs.ExecuteReader();

        // Sql mesterkulcs beolvasás
        while (sql_reader.Read())
        {
            // Sql Mesterkulcs táblázat betöltése egy konstruktorba
            MesterKulcs temp_mk = new MesterKulcs()
            {
                MesterId = int.Parse(sql_reader["mester_id"].ToString()),
                EncryptedKulcs = sql_reader["encrypted_kulcs"].ToString()
            };
            mester_kulcs.Add(temp_mk);
        }
        sql_reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
    return mester_kulcs;
}

```

24. ábra: Mesterkulcs tábla lekérés metódus a kódban

Az ábrán (24.) látható metódus lekérdezi az összes mesterkulcsot az adatbázisból, majd a kapott eredményt egy MesterKulcs típusú listába rendezi. A lista tartalmazza a kulcsok azonosítóját (mester_id) és titkosított értékét (encrypted_kulcs), amelyet a metódus visszaad.

Funkció: Adatbázisból való lekérdezés + lista visszaadása.

Kulcsadatok: mester_id (azonosító), encrypted_kulcs (titkosított kulcs).

```
3 references
public List<KliensLista> KliensListaLekeres(string mester_id, string mester_kulcs) // Kliens lista lekérdezés
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = new AdatbazisMetodusok().AdatbazisCsatlakozas();

    // vissza titkosítás
    TitkositasMetodusok szuper_titkos = new TitkositasMetodusok();

    try
    {
        // Adatok SELECT
        string sql_kliensek_select = $"SELECT j.jelszo_id, j.jelszo_string, f.nev, f.email, f.url, f.hozzafuzes FROM Jelszo j JOIN Felhasznalo f ON j.jelszo_id = f.jelszo_id WHERE j.mester_id = '{mester_id}'";
        MySqlCommand sql_command_kliensek = new MySqlCommand(sql_kliensek_select, db_csatlakozas);
        MySqlDataReader sql_reader = sql_command_kliensek.ExecuteReader();

        while (sql_reader.Read())
        {
            KliensLista temp_k = new KliensLista()
            {
                JelszoId = int.Parse(sql_reader["jelszo_id"].ToString()),
                JelszoString = szuper_titkos.DecryptText(mester_kulcs, sql_reader["jelszo_string"].ToString()),
                Nev = szuper_titkos.DecryptText(mester_kulcs, sql_reader["nev"].ToString()),
                Email = szuper_titkos.DecryptText(mester_kulcs, sql_reader["email"].ToString()),
                Url = szuper_titkos.DecryptText(mester_kulcs, sql_reader["url"].ToString()),
                Hozzafuzes = szuper_titkos.DecryptText(mester_kulcs, sql_reader["hozzafuzes"].ToString()),
            };
            kliens_lista.Add(temp_k);
        }
        sql_reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }

    return kliens_lista;
}
```

25. ábra: Kliens lista adatainak lekérése metódus a kódban

Az ábrán (25.) látható metódus a 'jelszo' és 'felhasznalo' táblákból kinyeri a kiválasztott adatokat ('jelszo_id', 'jelszo_string', 'nev', 'email', 'url', 'hozzafuzes'), amelyeket a megadott 'mester_id' paraméter alapján szűr. A lekérdezett adatokat a 'TitkositasMetodusok' osztály 'DecryptText' metódusával visszafejti az eredeti formájukba, a visszafejtéshez a 'mester_kulcs' paramétert használva. A metódus a visszafejtett adatokat egy 'KliensLista' típusú listában adja vissza.

Funkció: Adatbázisból való lekérdezés + visszafejtés + lista visszaadása.

Kulcsadatok: jelszo_id, jelszo_string, nev, email, url, hozzafuzes.

Kiemelve: Használt paraméterek (mester_id, mester_kulcs).

```

1 reference
public void MesterkulcsTablazatIras(string encrypted_kulcs) // Mesterkulcs tábla írás
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = new AdatbazisMetodusok().AdatbazisCsatlakozas();
    try
    {
        // Mesterkulcs tábla INSERT
        string sql_kulcs_iras = $"INSERT INTO mesterkulcs (encrypted_kulcs) VALUES ('{encrypted_kulcs}')";
        MySqlCommand sql_command_kulcs_iras = new MySqlCommand(sql_kulcs_iras, db_csatlakozas);
        sql_command_kulcs_iras.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
}

```

26. ábra: Mesterkulcs írás az adatbázisba metódus a kódban

Az ábrán (26.) látható metódus feladata a titkosított mesterkulcs ('encrypted_kulcs') elmentése az adatbázis 'mesterkulcs' táblájába. A metódus paraméterként egy előre titkosított string értéket vár, amelyet közvetlenül beszúr az adatbázisba, így a kulcs titkosított formában kerül tárolásra.

Funkció: Titkosított adat beszúrása az adatbázisba

Bemenet: Előre titkosított string (encrypted_kulcs)

Cél: Biztonságos adattárolás a mesterkulcs táblában

```

1 reference
public void JelszoTablazatIras(string jelszo_string, string mester_id) // Jelszó tábla írás
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = AdatbazisCsatlakozas();
    try
    {
        // Jelszo tábla INSERT
        string sql_jelszo_iras = $"INSERT INTO jelszo (jelszo_string, titkositas, mester_id) VALUES ('{jelszo_string}', 'AES-256', {mester_id})";
        MySqlCommand sql_command_jelszo_iras = new MySqlCommand(sql_jelszo_iras, db_csatlakozas);
        sql_command_jelszo_iras.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
}

```

27. ábra: Jelszó adatok írása az adatbázisba metódus a kódban

Az ábrán (27) látható metódus a 'jelszo' táblába szúr be új rekordot, amely a következő adatokat tartalmazza: a titkosított jelszót ('jelszo_string'), a használt titkosítási módszert ('AES-256 CBC'), valamint a hozzárendelt mesterkulcs azonosítóját ('mester_id'). A metódus paraméterként várja a tárolni kívánt jelszót és a hozzá tartozó mesterkulcs azonosítóját, amely meghatározza, hogy a jelszó melyik mesterkulcshoz lesz társítva az adatbázisban.

Funkció: Új jelszó bejegyzés létrehozása az adatbázisban

Bemenetek:

jelszo_string: titkosított jelszó

mester_id: a hozzárendelendő mesterkulcs azonosítója

Tárolt adatok:

A titkosított jelszó

A titkosítás típusa (AES-256 CBC)

A kapcsolódó mesterkulcs ID // jegyzet

```
1 reference
public void FelhasznaloTablazatIras(string nev, string email, string url, string hozzafuzes, int jelszo_id) // Felhasználó tábla írás
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = AdatbazisCsatlakozas();

    try
    {
        // Felhasználó tábla INSERT
        string sql_felhasznalo_iras = $"INSERT INTO felhasznalo (nev, email, url, hozzafuzes, jelszo_id) VALUES ('{nev}', '{email}', '{url}', '{hozzafuzes}', {jelszo_id})";
        MySqlCommand sql_command_felhasznalo_iras = new MySqlCommand(sql_felhasznalo_iras, db_csatlakozas);
        sql_command_felhasznalo_iras.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
}
```

28. ábra: Felhasználó adatok írása az adatbázisba metódus a kódban

Az ábrán(?) látható metódus a 'felhasznalo' táblába szúr be új rekordot, amely a következő felhasználói adatokat tárolja: név ('nev'), e-mail cím ('email'), URL ('url'), kiegészítő megjegyzés ('hozzafuzes'). A metódus paraméterként várja ezen adatokat, valamint a 'jelszo_id' azonosítót, amely meghatározza, hogy a felhasználói adatok melyik jelszóhoz lesznek társítva az adatbázisban. A kapcsolódó jelszó rekordot a 'jelszo' táblában található megfelelő azonosító alapján lehet elérni.

Funkció: Új felhasználói adatok rögzítése

Bemenetek:

Alapadatok: nev, email, url, hozzafuzes

Kapcsolódó entitás: jelszo_id

Adatbázis logika:

A felhasználó adatai a megadott jelszóhoz kapcsolódnak

A kapcsolat a jelszo_id segítségével valósul meg

```

5 references
public void LogTablaziIras(string leiras, string jelszo_id, string felhasznalo_id, string mester_id) // Hozzáférés log tábla írás
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = new AdatbazisMetodusok().AdatbazisCsatlakozas();

    try
    {
        // Hozzáférés log tábla INSERT
        string sql_log_iras = $"INSERT INTO hozzafereslog ( leiras, jelszo_id, felhasznalo_id, mester_id) VALUES ({leiras}, '{jelszo_
        id}', '{felhasznalo_id}', '{mester_id}')";
        MySqlCommand sql_command_log_iras = new MySqlCommand(sql_log_iras, db_csatlakozas);
        sql_command_log_iras.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
}

```

29. ábra: Naplózás adatok írása az adatbázisba metódus a kódban

Az ábrán (29.) látható metódus a 'hozzafereslog' táblába rögzíti a rendszerbeli tevékenységek naplóbejegyzéseit. A metódus kötelező paraméterként várja a művelet leírását ('leiras') és a mesterkulcs azonosítóját ('mester_id'), amely egyértelműen azonosítja a naplózandó eseményhez tartozó titkosítási kulcsot. Opcionálisan fogadja a felhasználó ('felhasznalo_id') és jelszó ('jelszo_id') azonosítókat is, lehetővé téve a rugalmas naplózást különböző rendszerszintű események esetén. A metódus fő célja a rendszerbeli tevékenységek nyomon követése és auditálása.

Funkció: Rendszerezemények naplózása

Kötelező paraméterek:

leiras: a naplózandó művelet szöveges leírása

mester_id: a kapcsolódó mesterkulcs azonosítója

Opcionális paraméterek:

felhasznalo_id: kapcsolódó felhasználó azonosítója (nullázható)

jelszo_id: kapcsolódó jelszó azonosítója (nullázható)

Adatbázis cél: Audit trail kialakítása, biztonsági nyomkövetés

```

1 reference
public void FelhasznaloSorTorles(int jelszo_id) // Felhasználó sor törlés
{
    // Adatbázis kapcsolat
    MySqlConnection db_csatlakozas = new AdatbazisMetodusok().AdatbazisCsatlakozas();
    try
    {
        // Felhasználó sor törlés parancsok
        string sql_felhasznalo_torles = $"DELETE FROM felhasznalo WHERE jelszo_id = {jelszo_id}";
        string sql_jelszo_torles = $"DELETE FROM jelszo WHERE jelszo_id = {jelszo_id}";
        MySqlCommand sql_command_felhasznalo_torles = new MySqlCommand(sql_felhasznalo_torles, db_csatlakozas);
        MySqlCommand sql_command_jelszo_torles = new MySqlCommand(sql_jelszo_torles, db_csatlakozas);

        // Felhasználó sor törlés végrehajtás
        sql_command_felhasznalo_torles.ExecuteNonQuery();
        sql_command_jelszo_torles.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Adatbázis csatlakozás error!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (db_csatlakozas.State == System.Data.ConnectionState.Open)
        {
            db_csatlakozas.Close();
        }
    }
}

```

30. ábra: Felhasználó adatainak törlése metódus a kódban

Az ábrán(?) látható metódus a 'felhasznalo' és 'jelszo' táblákból törli a megadott 'jelszo_id' -hoz tartozó rekordokat. A metódus először a felhasználói adatokat tároló 'felhasznalo' táblából, majd a jelszó adatokat tartalmazó 'jelszo' táblából végzi a törlést, ezzel biztosítva az adatok teljes és konzisztens eltávolítását.

Funkció: Kapcsolódó rekordok törlése több táblából

Bemenet: jelszo_id - a törlendő rekordok azonosítója

Adatbázis műveletek:

DELETE FROM felhasznalo WHERE jelszo_id = {jelszo_id}

DELETE FROM jelszo WHERE jelszo_id = {jelszo_id}

7 references

```
public string EncryptText(string masterKey, string text) // Szöveg titkosítás
{
    using (var sha256 = SHA256.Create())
    {
        // A mesterkulcsból 256 bites kulcsot generál
        byte[] key = sha256.ComputeHash(Encoding.UTF8.GetBytes(masterKey));

        byte[] iv = new byte[16]; // ennek mindig 16 bytenak kell lennie
        Array.Copy(key, iv, iv.Length); // A kulcs első 16 byteját használjuk IV-nek

        // AES algoritmus elindítása
        using (var aes = Aes.Create())
        {
            // AES beállítások
            aes.Key = key;
            aes.IV = iv;
            aes.Mode = CipherMode.CBC;
            aes.Padding = PaddingMode.PKCS7;

            // Encrypter létrehozása
            using (var encryptor = aes.CreateEncryptor(aes.Key, aes.IV))
            using (var ms = new MemoryStream())
            {
                // titkosított adatok memory streambe írása
                using (var cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))
                using (var sw = new StreamWriter(cs))
                {
                    sw.Write(text); // A titkosítandó szöveg írása.
                }

                // bytek tömbbé alakítja és Base64 stringgé alakítja
                byte[] encryptedBytes = ms.ToArray();
                return Convert.ToBase64String(encryptedBytes);
            }
        }
    }
}
```

5 references

```
public string DecryptText(string masterKey, string encryptedText) // Szöveg visszafejtés
{
    // SHA256 algoritmus elindítása
    using (var sha256 = SHA256.Create())
    {
        // A mesterkulcsból 256 bites kulcsot generál
        byte[] key = sha256.ComputeHash(Encoding.UTF8.GetBytes(masterKey));

        byte[] iv = new byte[16]; // mindig 16 byte hosszú
        Array.Copy(key, iv, iv.Length); // a kulcs első 16 byteját használjuk IV-nek

        // AES algoritmus elindítása
        using (var aes = Aes.Create())
        {
            // AES beállítások
            aes.Key = key;
            aes.Mode = CipherMode.CBC;
            aes.Padding = PaddingMode.PKCS7;

            // Decrypter létrehozása
            using (var decryptor = aes.CreateDecryptor(aes.Key, aes.IV))
            using (var ms = new MemoryStream(Convert.FromBase64String(encryptedText)))
            using (var cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))
            using (var sr = new StreamReader(cs))
            {
                // A visszafejtett visszaadása
                return sr.ReadToEnd();
            }
        }
    }
}
```

31 és 32. ábra: AES-256 CBC Encryption és Decryption metódusok a kódon belül

6. Tesztelési dokumentáció

A tesztelés egy kritikus pontja a fejlesztés folyamatának. A mi projektünkben történt felületes tesztelés viszont mélyebb tesztfolyamatok még nem sikerültek, ez az egyik lépés, amit szeretnénk orvosolni amikor egy következő projektet fejlesztünk ki. A jelenlegi helyzet állapota miatt nem tudunk benyújtani tesztelési dokumentációt és ábrákat azokról mivel, nem elég pontosan lettek elvégezve.

7. Továbbfejlesztési lehetőségek

Mesterkulcs Hashing: A mesterkulcs a jelenlegi egyszerű encryption módszerrel nem a legrosszabb megoldás, viszont szeretnénk a jövőben kifejleszteni egy hashing + salt módszert a mesterkulcsra, ezzel tovább biztosítani, hogy nem jut rossz ember kezébe a kulcs. Erre a PBKDF2 (Password-Based Key Derivation Function 2) kriptográfiai funkciót alkalmazzunk.

MFA és biztonságosabb belépési rendszer: A biztonságosabb adatmegőrzés érdekében szeretnénk egy MFA (Többtényezős hitelesítés) hitelesítés rendszert létrehozni, ami akármelyik platformon és eszközön használható lesz. Emellett szeretnénk, hogy több adatot kelljen megadni a belépési folyamatnál, ezzel is növelni a rendszer biztonságát.

SQL Injection megelőzés: A jelenlegi adatbázisunk nem túlságosan biztonságos, tehát ha valaki hozzáférést szerez a programhoz, esélye van SQL Injection-re, ez az egyik legnagyobb probléma a kódunkkal. Ennek megelőzésére a kód bázisunk nagy részét meg kell változtatnunk és ez lenne az egyik legelső lépés, amit megtennénk a továbbfejlesztés érdekében.

Adatok szűrése és keresése: Az adatokat a jelenlegi állapotukban nem a legegyszerűbb átnézni főleg, ha nagyon sok felhasználói fiók lett feltöltve az adatbázisba. Erre szeretnénk egy keresés és szűrés funkciót.

Design és UI továbbfejlesztése: A felhasználói élmény érdekében szeretnénk továbbá fejleszteni a felhasználói felületünket és a program kinézetét. Szeretnénk sötét és világos szín beállításokat implementálni, több nyelvbeállítási lehetőséget, tisztább és kifinomultabb kinézetet nyújtani a felhasználónak.

Saját adatbáziskezelő rendszer: A jelenlegi XAMPP/MAMPP MySQL és phpMyAdmin adatbázis kezelő rendszert szeretnénk lecserélni egy általunk létrehozott adatbázis rendszerre, amely az adott lokális hálózaton keresztül használható, hogy több helyről is elérhetőek legyenek az adatok a cégen belül. Erre egy olyan rendszert képzelünk el, amelyet az Aliasly program mellé készítenénk és a 2 program jönné egy csomagban.

Az alkalmazás kifejlesztése több platformra: Az alkalmazásunkat szeretnénk, hogy akárki használni tudja, ezért az egyik nagyobb lépés az lenne, hogy kifejlesszük a

Linux és Apple platformokra is, emellett Mobile platformokra is szeretnénk valamikor kiterjedni.

Weboldal az alkalmazás letöltéséhez és dokumentációjához: A kényelmesebb hozzáférés érdekében szeretnénk egy weboldalt készíteni a közeljövőben, ahonnan az alkalmazást lehet majd letölteni és a dokumentációt elolvasni.

Feedback rendszer: A legnagyobb segítség arra, hogy minél több felhasználónak megfeleljen az alkalmazás, a visszajelzés. A jövőbeli weboldalunkon erre lenne egy felület, ahol, ha felhasználó vagy, írhat egy ticket-et ahol visszajelzést adhatsz az alkalmazásról.

IV. Összegzés

A projektfeladat során a csapatunk jelentős fejlődésen ment keresztül, mind szakmai értelemben és emberi értelemben. A közös munka menete során rájöttünk, hogy mennyire fontos egyeztetni és mindenkinek az ötleteit meghallgatni és implementálni a munkába. A bemutatók és konzultációk során jelentős tapasztalatokat sajátítottunk el arról, hogy egy projekt menete hogyan kell, hogy megvalósuljon.

Éreztük a munka menetén, hogy mennyire fontos a hatékony munka és minden lépés átgondolása. Sokszor estünk kisebb nagyobb csapdába, ami miatt stagnált a projekt, visszanézve ezekre, sokat változtunk és sok új tapasztalatot gyűjtöttünk.

A projekt menedzsment eszközök fontosságát jobban átértéktük és itt is már látjuk, hogy min javíthattunk volna, pl. kifejezetten erre kitalált programok és eszközök használata. A Discord nekünk jelenleg bevált módszer volt erre a célra, viszont jobban össze tudtuk volna állítani a projekt tervezését, ha pl. a Jira-t használtuk volna.

Bízunk benne, hogy az újonnan nyert tapasztalatokat tovább vihetjük és még több tapasztalattal gazdagíthatunk meg a következő projektjein során.

Irodalomjegyzék, forrásmegjelölés

Discord chat applikáció: <https://discord.com>

GitHub verziókezelő és felhőtároló: <https://github.com>

GitHub Desktop alkalmazás a GitHub egyszerű használatához:

<https://desktop.github.com/download/>

XAMPP adatbáziskezelő: <https://www.apachefriends.org/index.html>

XAMPP adatbáziskezelő: <https://www.mamp.info/en/windows/>

Visual Studio Integrált Fejlesztési Környezet: <https://visualstudio.microsoft.com>

Excel táblázat tervező, kezelő: <https://www.microsoft.com/en-us/microsoft-365/excel>

ER Diagram elkészítése: <https://erdplus.com/standalone>

Említett Jira tervező program: <https://www.atlassian.com/software/jira>

DeepSeek LLM: <https://chat.deepseek.com>

GitHub Copilot LLM: <https://github.com/features/copilot>

C# Nyelvhez kisegítő hivatalos dokumentációk: <https://learn.microsoft.com/en-us/dotnet/csharp/>

AES-256 CBC Kriptográfia:

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard ;

<https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-9.0>

; [https://learn.microsoft.com/en-us/answers/questions/1291569/how-to-implement-a-c-](https://learn.microsoft.com/en-us/answers/questions/1291569/how-to-implement-a-c-aes-256-encrypt-and-decrypt-f)

[aes-256-encrypt-and-decrypt-f](https://learn.microsoft.com/en-us/answers/questions/1291569/how-to-implement-a-c-aes-256-encrypt-and-decrypt-f) ; [https://www.c-sharpcorner.com/article/encryption-and-](https://www.c-sharpcorner.com/article/encryption-and-decryption-using-aes-in-net-core-and-net-framework/)

[decryption-using-aes-in-net-core-and-net-framework/](https://www.c-sharpcorner.com/article/encryption-and-decryption-using-aes-in-net-core-and-net-framework/)