Tartalom

[Témaválasztás és Projekt Célja 2](#_Toc195299034)

[Kommunikáció és Dokumentáció 2](#_Toc195299035)

[Verziókezelés és Fejlesztői Környezet 3](#_Toc195299036)

[Módszertan 4](#_Toc195299037)

[Használt Technológiák 6](#_Toc195299038)

[A SyncroHub fejlesztése során a következő technológiákat alkalmaztuk: 6](#_Toc195299039)

[Alkalmazás Front-end Elemei 8](#_Toc195299040)

[Bejelentkezési felület: 11](#_Toc195299041)

[Fő vezérlőgombok: 13](#_Toc195299042)

[Főoldal: 13](#_Toc195299043)

[Finance oldal: 14](#_Toc195299044)

[Employees oldal: 15](#_Toc195299045)

[Vállalati eszköz nyilvántartás oldal: 17](#_Toc195299046)

[Szolgáltatások oldal: 19](#_Toc195299047)

[Vállalati Dokumentumok nyilvántartás oldal: 20](#_Toc195299048)

[Felhasználói Profil oldal: 22](#_Toc195299049)

[Adatbázis Elemzés 24](#_Toc195299050)

[Fő adattáblák és funkcióik: 24](#_Toc195299051)

[Táblák Közötti Kapcsolatok: 31](#_Toc195299052)

[ER Diagram: 32](#_Toc195299053)

[BACK-END 33](#_Toc195299054)

[Felhasználói Profilkezelés Modell: 33](#_Toc195299055)

[Eszközök és Dokumentumok Kezelése Back-End: 35](#_Toc195299056)

[Szolgáltatások és Pénzügyi Adatok Back-End: 37](#_Toc195299057)

[Cég Adatok Nyilvántartása Back-End Modell: 39](#_Toc195299058)

[Tesztelési Folyamat 42](#_Toc195299059)

[1. Egységtesztek (NUnit Framework) 42](#_Toc195299060)

[2. Automatizált Tesztek (Selenium WebDriver) 43](#_Toc195299061)

[3. Kódfelülvizsgálatok 44](#_Toc195299062)

[4. Felhasználói Elfogadási Tesztek (UAT) 44](#_Toc195299063)

[5. Kódok Működése 44](#_Toc195299064)

[Jövőbeli Fejlesztések 45](#_Toc195299065)

[Felhasználói Visszajelzések 46](#_Toc195299066)

# Témaválasztás és Projekt Célja

A SyncroHub projektet ketten valósítottuk meg: Kamarás Ádám és Szegedi Imre Márk. Projektünk célja egy olyan átfogó vállalati menedzsment rendszer létrehozása volt, amely segíti a különböző vállalatok irányítását és átláthatóságot biztosít. A témaválasztást személyes vállalkozásaink igényei inspirálták, felismerve a modern vállalatok számára elengedhetetlen hatékony projektmenedzsment és erőforrás-optimalizálás szükségességét.

A SyncroHub jelentősége abban rejlett, hogy integrálta a projektmenedzsment, az alkalmazottak kezelése, a pénzügyi kimutatások készítése, a szerződések kezelése és az eszközök nyomon követése funkciókat egy egységes platformba. Ez lehetővé tette a vállalatok számára, hogy könnyebben vezessék és átlássák munkamenetüket. Csoportunk mindkét tagja egyenlően osztozott a megvalósítandó feladatokon, mivel így tudtuk egymást segíteni különböző területeken.

# Kommunikáció és Dokumentáció

A közös munka során az információ zökkenőmentes megosztásának érdekében a következő eszközöket használtuk:

Discord: Valós idejű kommunikációra, videóhívásokra és képernyőmegosztásra használtuk. Ez a platform segített a tanulási anyagok, dokumentumok és erőforrások tárolásában, valamint a feladatok nyomon követésében.

Google Drive: A projekt dokumentációjának és fájljainak kezelésére, valós idejű szerkesztéssel és verziókövetéssel. Ez a felhőalapú tárhely lehetővé tette számunkra, hogy könnyen megosszuk és szinkronizáljuk a projekthez kapcsolódó összes dokumentumot, prezentációt és egyéb fájlt. A Google Drive együttműködési funkciói különösen hasznosnak bizonyultak a közös munkánk során. Emellett a Drive biztonságos tárolást és hozzáférés-kezelést is biztosított, ami fontos volt a projekt adatainak védelme szempontjából.

A white lined paper with blue lines

AI-generated content may be incorrect.

# Verziókezelés és Fejlesztői Környezet

Git és GitHub: A projekt verzióinak kezeléséhez a Git verziókezelőt alkalmaztuk. A Git segítségével nyomon követhettük a kód változtatásokat, lehetővé téve a csapat számára, hogy párhuzamosan dolgozzunk anélkül, hogy felborítanánk a projekt állapotát. A GitHub platform használata elősegítette a kódok tárolását és a közös munkát.

A projekt fejlesztése során két fő fejlesztőkörnyezetet használtunk. A frontend tervezéséhez és előzetes megjelenítéséhez a Visual Studio Code-ot alkalmaztuk. Ez lehetővé tette számunkra, hogy gyorsan és hatékonyan dolgozzunk a felhasználói felület kialakításán, és azonnal láthassuk a változtatások eredményét. A Visual Studio Code könnyű kezelhetősége és gazdag bővítmény-ökoszisztémája segített a frontend elemek gyors prototipizálásában.

A háttérrendszer kialakításához és a végleges implementációhoz a Visual Studio fejlesztőkörnyezetet választottuk, amely kiemelkedő eszközöket biztosított a C# nyelven történő programozáshoz. Ez a szoftver lehetővé tette számunkra, hogy hatékonyan dolgozzunk a komplex logika megírásán, miközben minden szükséges eszköz egy helyen volt elérhető. A Visual Studio beépített hibakeresője lehetővé tette, hogy valós időben kövessük a kód működését, és azonnal javítsuk az esetleges hibákat, ami jelentősen lerövidítette a fejlesztési időt.

Emellett a Visual Studio integrált támogatást nyújtott az ASP.NET Core keretrendszerhez, ami lehetővé tette a modern webes szolgáltatások kényelmes fejlesztését. A könnyen testreszabható felületnek köszönhetően mindkettőnk számára kényelmes volt a munkafolyamat, és akár egyszerre is dolgozhattunk ugyanazon a kódon a GitHub integráció segítségével.

A szoftver előnyei közé tartozott még a részletes dokumentáció és a közösségi támogatás, így bármilyen kérdés vagy technikai kihívás esetén gyorsan találtunk megoldást. A Visual Studio nem csak egy egyszerű kódszerkesztő volt számunkra, hanem egy átfogó eszközkészlet, amely hozzájárult a stabil és jól strukturált háttérrendszer létrehozásához, miközben a Visual Studio Code-dal tervezett frontend elemeket könnyedén tudtuk integrálni a teljes projektbe.

A projekt fejlesztése során a Docker konténerizációs technológiát is alkalmaztuk, amely lehetővé tette az alkalmazás és annak összes függőségének egységes környezetben történő futtatását. A Docker segítségével biztosítottuk, hogy a fejlesztési, tesztelési és éles környezetek között ne legyenek eltérések, így minimalizáltuk az inkompatibilitásból adódó problémákat. A konténerek gyors indítása és leállítása jelentősen felgyorsította a munkafolyamatot.

A projekt egy kisebb részfeladatának elvégzéséhez támogatásként mesterséges intelligencia alapú eszközt, a GitHub Copilotot is igénybe vettük.

# Módszertan

A projekt során a fejlesztési folyamatot agilis módszertan szerint végeztük, amely lehetővé tette a folyamatos visszajelzést és a gyors alkalmazkodást a változó igényekhez. Csapatunk rendszeresen tartott megbeszéléseket, ahol áttekintettük a projekt előrehaladását és a felmerülő problémákat.

A projekt során a fejlesztési folyamatot az Extrém Programozás (XP) módszertana szerint végeztük, amely lehetővé tette számunkra a folyamatos visszajelzést és a gyors alkalmazkodást a változó igényekhez. Az XP módszertan alapelveit követve csapatunk rendszeresen tartott rövid, de intenzív megbeszéléseket, ahol áttekintettük a projekt előrehaladását és a felmerülő problémákat. Az XP módszertan kulcselemeit alkalmaztuk, mint például a páros programozást, amely során a kód minőségének javítása és a tudásmegosztás érdekében gyakran dolgoztunk párban a fejlesztés során. Emellett folyamatos integrációt is alkalmaztunk, amelynek keretében a kódot naponta többször integráltuk, hogy korán felismerjük és kijavítsuk az esetleges konfliktusokat. A rövid fejlesztési ciklusok révén kis, kezelhető egységekben fejlesztettünk, ami lehetővé tette a gyors visszajelzést és a változásokra való rugalmas reagálást. Az egyszerű tervezés elve alapján a lehető legegyszerűbb megoldásokat kerestük, kerülve a túlbonyolítást, míg a folyamatos refaktorálás során rendszeresen felülvizsgáltuk és optimalizáltuk a kódot, hogy javítsuk annak minőségét és karbantarthatóságát. Ez a megközelítés segített abban, hogy gyorsan és hatékonyan tudjunk reagálni a felmerülő kihívásokra, miközben folyamatosan javítottuk a szoftver minőségét és biztosítottuk annak megbízhatóságát.

# Használt Technológiák

## A SyncroHub fejlesztése során a következő technológiákat alkalmaztuk:

**C#**: A projekt fő programozási nyelve volt, amely stabilitást és bővíthetőséget biztosított. A C# objektumorientált természete lehetővé tette a komplex üzleti logika hatékony implementálását. A C# remek választásnak bizonyult, mert nagyon stabil és jól bővíthető volt, így tökéletes volt a webszolgáltatások készítéséhez.

**ASP.NET Core**: A webalkalmazás fejlesztéséhez használt keretrendszer volt, amely lehetővé tette a cross-platform fejlesztést. Az ASP.NET Core MVC architektúrája segített a kód tiszta strukturálásában és a könnyű karbantarthatóság biztosításában.

**JavaScript és React**: A dinamikus front-end elemek létrehozásához JavaScript nyelvet és a React könyvtárat használtuk. A React segítségével komponens-alapú, interaktív felhasználói felületeket hoztunk létre, amelyek zökkenőmentesen kommunikáltak a back-end szolgáltatásokkal. A React rugalmassága és hatékonysága lehetővé tette számunkra, hogy könnyen kezelhető, újrafelhasználható komponenseket építsünk, ami jelentősen javította a fejlesztés hatékonyságát és az alkalmazás teljesítményét. A JavaScript és React kombinációja biztosította, hogy felhasználóink dinamikus és reszponzív élményt kapjanak, miközben az alkalmazás gyorsan reagált a felhasználói interakciókra és adatváltozásokra.

**SQL**: Az adatok tárolásához és kezeléséhez használt adatbázis-kezelő nyelv volt. Az SQL Server adatbázist használtuk, kihasználva annak fejlett lekérdezési és adatkezelési képességeit.

**Axios**: A front-end és back-end közötti HTTP kérések kezelésére az Axios könyvtárat alkalmaztuk. Az Axios segítségével könnyedén tudtunk aszinkron API hívásokat intézni a szerverhez, ami javította az alkalmazás reakcióidejét és felhasználói élményét.

**Mailgun**: Az e-mail kommunikáció kezelésére a Mailgun szolgáltatást használtuk, amely lehetővé tette az automatikus e-mail küldést, fogadást és nyomon követést. A Mailgun API-jának segítségével könnyedén implementáltunk tranzakciós e-maileket, például regisztrációs visszaigazolásokat és jelszó-helyreállítási üzeneteket. A szolgáltatás skálázhatósága és megbízhatósága biztosította, hogy az e-mailek időben és pontosan érkezzenek meg a címzettekhez.

**Swagger**: Az API dokumentáció létrehozásához és teszteléséhez a Swagger keretrendszert alkalmaztuk. A Swagger lehetővé tette az API struktúrájának automatikus leírását YAML vagy JSON formátumban, ami interaktív dokumentációt eredményezett. Ez nemcsak a fejlesztők munkáját könnyítette meg, hanem segítette az API-k gyorsabb tesztelését és integrációját is.

**Postman**: Az API-k tesztelésére és hibakeresésére a Postman eszközt használtuk. A Postman mock szerverei lehetővé tették az API végpontok szimulációját még az élesítés előtt, így biztosítva, hogy minden funkció megfelelően működjön. Az eszköz segítségével gyorsan írhattunk tesztcsomagokat, amelyek javították az API-k minőségét és megbízhatóságát.

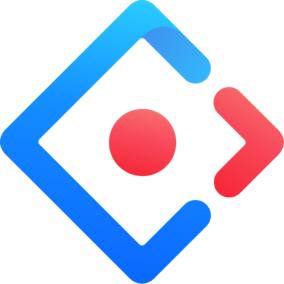
**Recharts**: Az adatok vizualizálásához a Recharts könyvtárat használtuk, amely egyszerűen használható és hatékony eszköz React fejlesztők számára. A Recharts komponens-alapú felépítése lehetővé tette interaktív grafikonok gyors létrehozását, például vonal-, oszlop- és kördiagramokat. Ez jelentősen javította az alkalmazás adatvizualizációs képességeit.

**JWT**: A felhasználói hitelesítéshez és jogosultságkezeléshez JWT-t alkalmaztunk. A JWT stateless természete lehetővé tette a skálázható autentikációs rendszer kialakítását, miközben biztosította az adatok biztonságos továbbítását. A tokenek digitális aláírása garantálta azok sértetlenségét és hitelességét.

# Alkalmazás Front-end Elemei

Az alkalmazás front-end dizájnját teljes mértékben a saját elképzeléseink alapján alakítottuk ki. Nem használtunk hozzá konkrét tervezőeszközöket, hanem folyamatosan formáltuk a kód írása közben, ahogy a fejlesztés haladt előre. Célunk az volt, hogy egy teljesen felhasználóbarát felületet (UI/UX) alakítsunk ki. Letisztult, harmonikus színvilágú felületet terveztünk, világos háttérrel és kontrasztos fő elemekkel. A betűtípusokat a könnyű olvashatóság és a különböző méretekhez való alkalmazkodás szempontjából választottuk ki. Olyan harmonikus színeket választottunk, amik jól mutattak együtt. A háttér világos lett, míg a fő elemek (gombok, címkék) sötétebbek, hogy kiemelkedjenek.

A fejlesztés során a React könyvtárat használtuk, amely lehetővé tette számunkra, hogy komponens-alapú megközelítéssel építsük fel az alkalmazást, így növelve annak újrafelhasználhatóságát és karbantarthatóságát. A React rugalmassága és hatékonysága segített abban, hogy gyorsan tudjunk iterálni és finomítani a felhasználói felületen.

Az Ant Design (antd) komponens könyvtárat is integráltuk a projektbe, amely egy átfogó design rendszert biztosított számunkra. Az antd előre elkészített, testre szabható komponensei - mint például gombok, űrlapok, táblázatok - segítségével konzisztens és professzionális megjelenést tudtunk kialakítani, miközben jelentősen csökkentettük a fejlesztési időt. Az antd rugalmas témázási lehetőségei révén könnyen tudtuk a komponenseket a saját színvilágunkhoz és stílusunkhoz igazítani, megőrizve az egységes megjelenést az egész alkalmazásban.

A blue and white logo

AI-generated content may be incorrect.A Google Translate API-t használtuk az alkalmazás szövegeinek fordítására, amely lehetővé tette, hogy többnyelvű felhasználói élményt biztosítsunk.

A graph with different colored lines

AI-generated content may be incorrect.Az adatok vizualizálására különböző diagramokat építettünk be az alkalmazásba a React Chart.js és React-chartjs-2 könyvtárak segítségével. Ezekkel a könyvtárakkal interaktív és reszponzív grafikonokat hoztunk létre, például oszlopdiagramokat, kördiagramokat és vonaldiagramokat. Az adatok REST API-ból való betöltésével dinamikus grafikonokat készítettünk, amelyek valós idejű frissítést is támogatnak. A Chart.js rugalmassága lehetővé tette számunkra, hogy testre szabjuk a diagramok megjelenését és funkcionalitását.

**Ikonográfia**

Az ikonográfia kialakításánál egyszerű, modern ikonokat választottunk, amelyek segítették a navigációt és javították a felhasználói élményt. Az ikonok egységes stílusúak és könnyen érthetőek lettek. Ebben nagy segítségünkre volt az Ant Design (antd) ikonkészlete, amelyet integráltunk a projektbe. Az antd széles választékú, következetes stílusú ikonjai lehetővé tették számunkra, hogy minden funkcióhoz megfelelő vizuális jelölést társítsunk. Az antd ikonok skálázhatósága és testreszabhatósága révén könnyen tudtuk az ikonokat az alkalmazásunk általános dizájnjához igazítani, miközben megtartottuk azok felismerhetőségét és érthetőségét. Ez a megközelítés nemcsak a fejlesztési időt csökkentette, de biztosította azt is, hogy az ikonográfiánk professzionális megjelenésű és konzisztens legyen az egész alkalmazásban.

**Felhasználói Élmény (UX)**

Fontos volt számunkra, hogy a felhasználók könnyen feltalálják magukat a felületen. A műveletek sorrendjét logikusan alakítottuk ki, és igyekeztünk minimalizálni a lépések számát a feladatok elvégzéséhez. Figyelembe vettük, hogy a felhasználók hogyan használják az alkalmazást, hogy a lehető legjobb élményt nyújtsuk.

A felhasználói élmény optimalizálása érdekében számos tesztet és iterációt végeztünk. Hőtérképeket és kattintási elemzéseket használtunk, hogy megértsük, mely területekre összpontosítanak leginkább a felhasználók. Ennek alapján finom hangoltuk az elrendezést, hogy a leggyakrabban használt funkciók könnyen elérhetőek legyenek.

Különös figyelmet fordítottunk az űrlapok és adatbeviteli mezők kialakítására. Valós idejű validációt implementáltunk, amely azonnal visszajelzést ad a felhasználónak az esetleges hibákról, így csökkentve a frusztrációt és gyorsítva a folyamatokat. Az automatikus kitöltés és intelligens alapértelmezett értékek használata tovább egyszerűsítette a felhasználói interakciókat.

A reszponzív dizájn kialakításakor gondosan ügyeltünk arra, hogy az alkalmazás minden eszközön - legyen az asztali számítógép, tablet vagy okostelefon - egyformán jól használható legyen. Az érintőképernyős eszközökre optimalizált nagyobb gombok és egyértelműbb navigációs elemek beépítésével biztosítottuk a zökkenőmentes mobilhasználatot is.

A screenshot of a phone

AI-generated content may be incorrect.A navigációs sáv elhelyezésének kérdése komoly megbeszélés tárgyát képezte a fejlesztés során. Végül úgy döntöttünk, hogy a bal oldali elrendezés mellett maradunk, mivel ez jobban illeszkedett az általunk elképzelt felhasználói élményhez és a SyncroHub dizájn koncepciójához.

A felső header sávot úgy alakítottuk ki, hogy a felhasználók számára azonnal elérhetőek legyenek az alapvető információk és funkciók, miközben vizuálisan is letisztult és modern megjelenést biztosít. A sáv központi eleme az üdvözlő üzenet, amely dinamikusan jelenik meg a bejelentkezett felhasználó nevével, így személyesebb élményt nyújt. A jobb oldalon található egy profil gomb. Ennek segítségével a felhasználók gyorsan hozzáférhetnek a profiljukhoz kapcsolódó adatokhoz és beállításokhoz.

A white circle with black arrows and a black arrow

AI-generated content may be incorrect.A jobb alsó sarokban egy lebegő gombcsoportot helyeztünk el, amely gyors és kényelmes hozzáférést biztosít néhány alapvető funkcióhoz. Ez a megoldás nemcsak helytakarékos, hanem modern és intuitív is, lehetővé téve a felhasználók számára, hogy könnyen elérjék a legfontosabb beállításokat anélkül, hogy elhagynák az aktuális oldalt.

Ez a döntés tükrözi azt a törekvésünket, hogy minél inkább felhasználóbarát és könnyen kezelhető felületet hozzunk létre, amely igazodik a különböző felhasználói preferenciákhoz.

## A screenshot of a login form AI-generated content may be incorrect.Bejelentkezési felület:

A bejelentkezési felület (Sign In Page) az alkalmazás egyik kulcsfontosságú része, amely lehetővé teszi a felhasználók számára, hogy hitelesítést követően hozzáférjenek a rendszer funkcióihoz. A felület kialakítása során nagy hangsúlyt kapott az egyszerűség, a biztonság és a felhasználói élmény maximalizálása. Az alábbiakban részletesen bemutatjuk a bejelentkezési oldal főbb elemeit és azok működését.

**Felhasználói hitelesítés folyamata**

**A bejelentkezési folyamat két fő mezőt tartalmaz**

**Email or Username mező**: Ez a mező lehetőséget ad arra, hogy a felhasználók e-mail címükkel vagy felhasználónevükkel jelentkezzenek be. Ez a rugalmasság növeli a használhatóságot.

**Password mező**: A jelszómező fogadja el a felhasználói jelszót. A jelszó helyességét az alkalmazás az adatbázisban tárolt adatokkal ellenőrzi.

A hitelesítési folyamatot egy **"Sign In"** gomb indítja el, amely csak akkor aktív, ha minden kötelező mező megfelelően ki van töltve.

**Jelszó megjelenítési funkció**

A jelszómező mellett található ikon lehetőséget biztosít a jelszó megjelenítésére vagy elrejtésére. Ez a funkció különösen hasznos, ha a felhasználók szeretnék ellenőrizni, hogy helyesen írták-e be jelszavukat. Az ikon interaktív, és egyszerű kattintással aktiválható.

**Elfelejtett jelszó kezelése**

Az oldal tartalmaz egy **"Can't log in?"** linket, amely lehetőséget nyújt az elfelejtett jelszó kezelésére. A folyamat több lépésből áll:

**FogPassLog komponens**: Lehetővé teszi, hogy a felhasználók e-mail címük megadásával helyreállítási linket kérjenek.

**RecCodeInp komponens**: Az e-mailben kapott helyreállítási kód megadására szolgáló felület.

**ResPass komponens**: Az új jelszó beállítását biztosítja.

Ezek a komponensek egymásra épülnek, és zökkenőmentes élményt nyújtanak a felhasználóknak.

**Sikeres bejelentkezés visszajelzése**

Sikeres hitelesítés esetén egy modális ablak jelenik meg (**SuccessLogin**), amely vizuális visszajelzést ad a felhasználónak, majd automatikusan átirányítja őt az alkalmazás főoldalára (/home).

**Vizuális megjelenés és reszponzivitás**

**Az oldal dizájnja modern és letisztult:**

**Háttérkép dinamikus változtatása**: Az oldal háttérképe automatikusan igazodik a felhasználó által választott témához (világos vagy sötét mód). Ezt az isDarkMode állapot alapján határozza meg.

**Reszponzív kialakítás**: Az oldal elemei rugalmasan alkalmazkodnak különböző eszközök kijelzőméretéhez.

A háttérképek (DarkBg és LightBg) vizuálisan vonzóvá teszik az oldalt, miközben professzionális megjelenést biztosítanak.

**Technikai részletek**

Az oldal React keretrendszerrel készült, és több külső könyvtárat használ:

**Ant Design (antd)**: A felhasználói interfész elemeinek gyors és esztétikus kialakításához.

**React Router Dom**: Az oldalak közötti navigáció kezeléséhez.

Az állapotkezeléshez React hookokat használnak, például useState és useEffect. Ezek biztosítják az interaktív elemek megfelelő működését.

## Fő vezérlőgombok:

Ez a felület négy fő vezérlőgombot tartalmaz, amelyek különböző funkciókat látnak el az alkalmazásban:

A white circle with black arrows and a black arrow

AI-generated content may be incorrect.**Main Control Buttons**: Ez a komponens egy központi vezérlőpanelként működik, amely tartalmazza a témaváltó gombot, nyelvváltó gombot és kijelentkezési gombot. Az FloatButton elemek biztosítják a lebegő gombok megjelenését, amelyek könnyen elérhetők és vizuálisan kiemelkednek.

**Language Change Button**: Ez a gomb lehetővé teszi a felhasználók számára, hogy a Google Translate integrációval dinamikusan változtassák az oldal nyelvét. A komponens betölt egy Google Translate szkriptet, amely automatikusan inicializálja a fordítási funkciót. A widget láthatósága egy kapcsolóval állítható, amely megjeleníti vagy elrejti a fordítási elemet.

**Theme Toggle Button**: Ez a gomb az alkalmazás világos és sötét módja között vált. A témakezelést kontextus segítségével valósítja meg, és tooltipet jelenít meg, amely jelzi az aktuális módot (például "Switch to Light Mode" vagy "Switch to Dark Mode"). A felhasználói élményt növeli az egyszerű váltási lehetőség.

**LogOut Button**: Ez a komponens biztonságos kijelentkezést biztosít. Törli az autentikációs tokeneket a helyi és munkameneti tárból (localStorage és sessionStorage), majd átirányítja a felhasználót a bejelentkezési oldalra (/sign-in). Ez garantálja, hogy az adatok védettek maradnak a kijelentkezés után.

Az összes vezérlőgomb egyszerűen használható, reszponzív kialakítású, és biztosítja az alkalmazás alapvető funkcióinak gyors elérését.

## Főoldal:

Ez a főoldal tartalma különböző komponensekből áll, amelyek az alkalmazás kulcsfontosságú funkcióit jelenítik meg és segítik a gyors áttekintést:

**ChIncExpCard**: Ez a kártya egy vizuális diagramot jelenít meg, amely az alkalmazás bevételeit és kiadásait ábrázolja. A diagram segítségével a felhasználók könnyen nyomon követhetik pénzügyi helyzetüket. A sötét és világos mód támogatása biztosítja a felhasználói élmény testreszabhatóságát.

**CashFlowBnr**: Ez a szalagcím részletes pénzügyi összefoglalót nyújt. Kiszámolja és megjeleníti az összesített bevételeket, kiadásokat, valamint azok havi változásait százalékos formában. Emellett kiszámítja az egyenleget, amely pozitív vagy negatív pénzáramlást jelez. A színek (zöld vagy piros) vizuálisan kiemelik a pénzügyi helyzetet.

**StaffListCard**: Ez a kártya a felhasználó által felügyelt munkatársak listáját jeleníti meg, legfeljebb három főig. Az aktuális felhasználó e-mailje alapján szűri az adatokat, hogy kizárja saját magát a listából. A "View All" gomb lehetőséget biztosít az összes munkatárs megtekintésére egy dedikált oldalon.

**EquipmentListCard**: Ez a kártya az eszközök listáját jeleníti meg, amelyeket az alkalmazásban kezelnek. Bár a pontos működés nincs részletezve, valószínűleg lehetőséget ad az eszközök gyors áttekintésére és kezelésére.

A screenshot of a graph

AI-generated content may be incorrect.Az oldal reszponzív kialakítású, modern dizájnnal rendelkezik, és biztosítja, hogy a felhasználók gyorsan hozzáférjenek a legfontosabb információkhoz. Az egyes komponensek közötti szoros integráció segíti az adatok konzisztens kezelését és megjelenítését.

## Finance oldal:

Ez az oldal a pénzügyi adatok kezelésére szolgál, és három fő komponensből áll, amelyek különböző funkciókat biztosítanak:

**1. Bevétel és Kiadás Kártyák (IncCard és ExpCard)**

Az IncCard és ExpCard komponensek a bevételek és kiadások részleteit jelenítik meg. Mindkét kártya listázza az adott tranzakciókat, beleértve a leírást, összeget, fizetési státuszt (pl. "Pending", "Paid", "Overdue"), valamint lehetőséget biztosít új bevételek vagy kiadások hozzáadására. Az adatok dinamikusan frissülnek az API hívások alapján, így mindig naprakészek. Az új tranzakciók hozzáadására dedikált modális ablakok szolgálnak (AddNewInc és AddNewExp), amelyek részletes adatokat kérnek be, mint például dátum, összeg, kategória, fizetési mód, kedvezményezett neve, és adószám.

**2. Diagram (ChartIncExp)**

A ChartIncExp komponens egy interaktív diagramot jelenít meg, amely vizuálisan ábrázolja a bevételek és kiadások alakulását havi vagy éves bontásban. A diagram adatainak feldolgozása során az API-ból származó bevétel- és kiadásadatok kerülnek összesítésre. A felhasználók választhatnak, hogy csak a bevételeket, csak a kiadásokat vagy mindkettőt szeretnék megtekinteni. Az időszak váltása (havi/éves nézet) dinamikusan frissíti a megjelenített adatokat.

**3. Adatok Kezelése**

Mind a bevételek (AddNewInc) mind a kiadások (AddNewExp) hozzáadására szolgáló modális ablakok részletes adatbevitelt tesznek lehetővé. A felhasználók kiválaszthatják a szolgáltatást, céget, fizetési módot és státuszt előre definiált listákból, amelyek az API-ból töltődnek be. Ha az API nem érhető el, fallback opciók biztosítják az alapvető működést. Az adatok validálása garantálja a helyes formátumot minden mezőben.

**Felhasználói Élmény**

Az oldal támogatja a sötét és világos módot a felhasználói preferenciák alapján (isDarkMode). Az egyszerű navigációt biztosító gombok és interaktív elemek garantálják az intuitív használatot. Az adatok biztonságos kezelése érdekében minden tranzakció API-n keresztül történik, hibakezeléssel kiegészítve.

Ez az oldal átfogó eszközt nyújt a pénzügyi adatok kezelésére, vizualizációjára és nyomon **követésére egy modern és reszponzív környezetben.**

## Employees oldal:

Ez az oldal a munkavállalók adatait kezeli, és több funkciót biztosít az alkalmazásban:

**1. Munkavállalók Listázása (ListedEmployee)**

Az ListedEmployee komponens megjeleníti az összes munkavállalót egy táblázatban, amely tartalmazza az alapvető információkat, például nevet, e-mail címet és pozíciót. A táblázat támogatja a keresést és szűrést, így a felhasználók könnyen megtalálhatják a kívánt adatokat. Az adatok az API-ból töltődnek be, és dinamikusan frissülnek. A felhasználók számára elérhetők interaktív gombok, amelyek lehetőséget nyújtanak új munkavállaló hozzáadására (UserAddOutlined), szerkesztésre (EditOutlined), vagy törlésre (DeleteOutlined). A törlési művelet megerősítést igényel (Popconfirm), hogy elkerülje a véletlen adatvesztést.

**2. Munkavállalói Profil Megtekintése (EmpDetails)**

Az EmpDetails komponens egy modális ablakban jeleníti meg a kiválasztott munkavállaló részletes adatait, például személyes adatokat (név, e-mail), címadatokat (város, irányítószám), és pénzügyi információkat (adószám, bankszámlaszám). Az adatok szerkeszthetők, de bizonyos mezők (például adószám) módosítása korlátozott, és csak adminisztrátorok által végezhető el. A módosítások mentése után az adatok automatikusan frissülnek.

**3. Új Munkavállaló Hozzáadása (AddNewEmployee)**

Az AddNewEmployee komponens egy lépésről lépésre haladó folyamatot biztosít új munkavállaló regisztrációjához. Az űrlap négy szakaszból áll:

**Címadatok** (HomeAddress): Az irányítószámot, címet, várost és országot rögzíti. Az irányítószám és város validálása API segítségével történik.

**Születési Adatok** (DataOfBirth): Születési helyet és dátumot kér be. A születési dátum validálása biztosítja, hogy a munkavállaló legalább 18 éves legyen.

**Személyes Adatok** (PersonalData): E-mail címet, jelszót, telefonszámot, személyi igazolványszámot, adószámot és társadalombiztosítási számot tartalmaz.

**Szerződéses Adatok** (ContractData): Munkaidőt, órabért, szabadságok számát és munkakezdési dátumot rögzíti. A pozíciót és felügyelőt előre definiált listákból lehet kiválasztani.

**4. Navigáció és Interakciók**

Az oldal reszponzív kialakítású, támogatja a sötét és világos módot (isDarkMode). A navigáció egyszerűen kezelhető az útvonalakon keresztül (useNavigate), például új munkavállaló hozzáadásakor vagy profil megtekintésekor. Az adatokat helyileg tárolja (localStorage) a folyamat közben, hogy elkerülje az adatvesztést.

**Biztonság és Validáció**

Az űrlapok minden mezője validálva van a megfelelő formátum biztosítása érdekében (például irányítószám mintázat vagy jelszó komplexitás). Az API-hívások hibakezeléssel vannak ellátva, hogy értesítsék a felhasználót bármilyen probléma esetén.

Ez az oldal átfogó eszközt nyújt a munkavállalók kezelésére, beleértve azok hozzáadását, szerkesztését és részletes adatainak megtekintését egy modern és intuitív környezetben.

## Vállalati eszköz nyilvántartás oldal:

Ez az oldal az eszközök kezelésére szolgál, és több interaktív funkciót biztosít, amelyek lehetővé teszik az eszközök listázását, hozzáadását, szerkesztését és részleteinek megtekintését.

**1. Eszközök Listázása (EqTable)**

Az EqTable komponens egy táblázatos nézetben jeleníti meg az eszközöket, beleértve az olyan adatokat, mint a sorozatszám (serialNumber), az eszköz neve (name), a felelős munkavállaló (employee), és az állapot (status). A táblázat támogatja:

* *Keresés*: Egy beépített keresőmező segítségével a felhasználók gyorsan megtalálhatják a kívánt eszközt.
* *Szerkesztés és Törlés*: Az egyes sorokhoz tartozó gombok lehetőséget nyújtanak az eszköz adatainak módosítására vagy törlésére. A törlési művelet megerősítést igényel (Popconfirm).
* *Részletek Megtekintése*: Az EqDetails modális ablakban jeleníti meg a kiválasztott eszköz részletes adatait.

**2. Új Eszköz Hozzáadása (EqAddNew)**

Az EqAddNew komponens egy modális ablakot biztosít új eszközök hozzáadására. Az űrlap mezői:

* *Sorozatszám és Név*: Az eszköz egyedi azonosítója és neve.
* *Felelős Munkavállaló:* Az eszközt használó személy neve.
* *Állapot:* Előre definiált értékekből választható (pl. "Issued", "Under Maintenance", "Worn Out").
* *Dátumok:* Vásárlás dátuma, utolsó szervizelés dátuma, garancia lejárata.
* *Megjegyzések*: Opcionális szöveges mező további információkhoz.

Az adatok validálása biztosítja, hogy minden mező megfelelő formátumban legyen kitöltve. A sikeres mentés után az új eszköz automatikusan megjelenik a táblázatban.

**3. Eszközök Részletei (EqDetails)**

Az EqDetails komponens egy modális ablakban jeleníti meg az adott eszköz teljes adatkészletét. A felhasználók itt:

* *Szerkeszthetnek Adatokat:* Például frissíthetik az állapotot vagy dátumokat.
* *Menthetik a Módosításokat:* Az adatok validálása után a frissített információk mentésre kerülnek.
* *Visszavonhatják a Módosításokat:* Ha nem kívánják véglegesíteni a változtatásokat.
* Az állapot vizuálisan is kiemelésre kerül színkódokkal (pl. sárga karbantartás alatt, zöld kiadott állapot).

**4. Oldal Felépítése és Témakezelés**

Az oldal reszponzív kialakítású, amely különböző képernyőméreteken is jól működik. A sötét és világos mód támogatása (isDarkMode) biztosítja a felhasználói élmény testreszabhatóságát. Az Add New Equipment gomb segítségével egyszerűen hozzáadhatók új elemek, míg a táblázatban található interaktív elemek gyors adatkezelést tesznek lehetővé.

Ez az oldal átlátható és hatékony eszközt nyújt az eszközök nyilvántartásához, kezeléséhez és nyomon követéséhez egy modern felhasználói felületen keresztül.

## Szolgáltatások oldal:

Ez az oldal a szolgáltatások kezelésére és megjelenítésére szolgál, több interaktív funkcióval, amelyek segítik a szolgáltatások kezelését, hozzáadását, módosítását és részleteinek megtekintését.

1. **Szolgáltatások Listázása (*ServiceWidget* és *Pagination*)**

Az oldal a *ServiceWidget* komponens segítségével jeleníti meg a szolgáltatásokat kártyák formájában. Minden kártya tartalmazza a szolgáltatás nevét, árát, pénznemét (*serviceFiatType*), rövid leírását és egy képet. A felhasználók interaktív módon kattinthatnak egy kártyára, hogy megtekintsék a szolgáltatás részletes adatait. A lista lapozható (*Pagination*), így nagyobb mennyiségű adat is könnyen kezelhető.

1. **Szolgáltatás Részletei (*ServiceDatas*)**

A *ServiceDatas* komponens egy modális ablakban jeleníti meg a kiválasztott szolgáltatás részletes adatait, például:

* Szolgáltatás neve (*serviceName*)
* Ár és pénznem (*servicePrice*, *serviceFiatType*)
* Leírás (*serviceDescription*)
* Kép (*servicePhoto*)

A felhasználók szerkeszthetik az adatokat, például módosíthatják az árat vagy leírást. Az adatok validálása biztosítja, hogy minden mező helyesen legyen kitöltve. A mentés után az új információk automatikusan frissülnek.

1. **Új Szolgáltatás Hozzáadása (*AddNewService*)**

Az *AddNewService* komponens lehetőséget biztosít új szolgáltatások hozzáadására egy modális ablakon keresztül. Az űrlap mezői:

* Szolgáltatás neve (*serviceName*)
* Ár és pénznem (*servicePrice*, *serviceFiatType*)
* Leírás (*serviceDescription*)
* Kép feltöltése (*servicePhoto*): Csak *SVG*, *PNG*, *JPG*, *JPEG* formátumú képek engedélyezettek, maximum 300x300 pixel méretben.

Az űrlap kitöltöttsége dinamikusan követhető egy előrehaladási sávon (*Progress*). Az adatok validálása garantálja a helyes formátumot minden mezőben.

1. **Szolgáltatások Feltöltési Folyamata (*ProcessingPageService*)**

A *ProcessingPageService* komponens egy folyamatjelző oldalt biztosít a szolgáltatások feltöltése közben. Az adatok feldolgozása során egy töltési animáció jelenik meg (Spin), amely 10 másodperc után automatikusan eltűnik. A felhasználók visszatérhetnek az előző oldalra vagy folytathatják a következő lépésre.

1. **Sikeres Hozzáadás Visszajelzése (*SuccessNewService*)**

A sikeres hozzáadás után egy visszajelző modális ablak jelenik meg (*SuccessNewService*), amely értesíti a felhasználót az új szolgáltatás sikeres létrehozásáról.

1. **Felhasználói Élmény és Témakezelés**

Az oldal támogatja a sötét és világos módot (isDarkMode), amely vizuálisan testre szabhatóvá teszi a felületet. Az interaktív gombok és vizuális elemek intuitív használatot biztosítanak.

Ez az oldal átfogó eszközt nyújt a szolgáltatások kezelésére és nyomon követésére egy modern és reszponzív környezetben.

## Vállalati Dokumentumok nyilvántartás oldal:

Ez az oldal dokumentumok kezelésére szolgál, és több funkciót biztosít a dokumentumok listázására, hozzáadására, szerkesztésére és részleteinek megtekintésére.

1. **Dokumentumok Listázása (*DocsTable*)**

A *DocsTable* komponens egy táblázatos nézetben jeleníti meg a dokumentumokat. A táblázat tartalmazza a következő oszlopokat:

* Document ID: A dokumentum egyedi azonosítója.
* Név: A dokumentum neve.
* Típus: A fájl típusa (pl. PDF, Word, Excel).
* Létrehozó: Az a felhasználó, aki feltöltötte a dokumentumot.

A felhasználók kereshetnek a dokumentumok között, szűrhetik az adatokat, valamint interaktív gombok segítségével szerkeszthetik vagy törölhetik azokat. A törlés megerősítést igényel (*Popconfirm*), hogy elkerüljék a véletlen adatvesztést. A részletek megtekintéséhez egy külön modális ablak nyílik meg (*DocsDetails*).

1. **Dokumentum Részletei (*DocsDetails*)**

A *DocsDetails* komponens egy modális ablakban jeleníti meg a kiválasztott dokumentum részletes adatait:

* Név: A dokumentum neve.
* Típus: A fájl típusa.
* Létrehozó: Az a személy, aki feltöltötte.
* Megjegyzések: Opcionális szöveges mező további információkhoz.

A felhasználók szerkeszthetik a dokumentum adatait (pl. megjegyzések hozzáadása vagy módosítása), feltölthetnek új fájlokat, és menthetik a változtatásokat. Csak bizonyos fájltípusok engedélyezettek (PDF, Word, Excel), és ezek feltöltése validálva van.

1. **Új Dokumentum Hozzáadása (*DocsAddNew*)**

Az *DocsAddNew* komponens lehetőséget biztosít új dokumentumok hozzáadására egy modális ablakon keresztül. Az űrlap mezői:

* Document ID: Egyedi azonosító.
* Név: A dokumentum neve.
* Típus: A fájl típusa (pl. PDF, Word, Excel).
* Létrehozó: Az a személy, aki feltölti.
* Megjegyzések: Opcionális leírás.

A fájl feltöltésekor csak bizonyos típusok engedélyezettek (PDF, Word, Excel), és a validáció biztosítja a helyes formátumot. A sikeres mentés után az új dokumentum automatikusan megjelenik a táblázatban.

1. **Dokumentumok Kezelése**

Az oldal támogatja a sötét és világos módot (*isDarkMode*), amelyet a felhasználói preferenciák alapján vált. Az adatok dinamikusan frissülnek az API-hívások alapján, így mindig naprakészek. Az új dokumentumok hozzáadása és szerkesztése során minden adat validálva van, hogy biztosítsa a helyes formátumot.

1. **Felhasználói Élmény**

Az oldal reszponzív kialakítású, így különböző eszközökön is könnyen használható. Az interaktív elemek (pl. keresés, szűrés, modális ablakok) intuitív használatot biztosítanak, miközben az adatkezelés biztonságos és hatékony marad.

## Felhasználói Profil oldal:

Ez az oldal a felhasználói profilok kezelésére szolgál, és több interaktív funkciót biztosít, amelyek lehetővé teszik az adatok megtekintését, szerkesztését, valamint kapcsolódó műveletek végrehajtását.

**1. Profil Adatok Megjelenítése (*DataCard*)**

A *DataCard* komponens a felhasználó alapvető adatait jeleníti meg kártyák formájában. Az adatok között szerepel:

A kártyák interaktív elemeket tartalmaznak, amelyek lehetőséget nyújtanak a részletes adatok megtekintésére vagy módosítására.

**2. Profil Módosítása (*ProfileEdit*)**

A *ProfileEdit* modális ablak lehetővé teszi a felhasználók számára, hogy módosítsák saját profiljukat. Az űrlap mezői validálva vannak, hogy biztosítsák a helyes adatbevitelt. A mentés után az új információk automatikusan frissülnek a profilban.

**3. Jelszó Módosítása (*NewPassword*)**

A *NewPassword* komponens egy külön modális ablakban biztosítja a jelszó megváltoztatásának lehetőségét. A jelszó mező tartalmaz egy ikon gombot, amely lehetővé teszi a jelszó megjelenítését vagy elrejtését. A mentés után egy visszajelző ablak (*SuccessNewPass*) értesíti a felhasználót a sikeres műveletről.

**4. Saját Cég Adatainak Kezelése (*OwnComp*)**

Az *OwnComp* komponens lépésről lépésre haladó folyamatot biztosít a cégadatok kezelésére. Két fő szakaszból áll:

* Általános Adatok: A cég alapvető információi.
* Cégspecifikus Adatok: Regisztrációs szám, bankszámlaszám és más pénzügyi információk.

Az adatok kitöltése közben egy előrehaladási sáv (*Progress*) mutatja az aktuális állapotot. A sikeres mentés után az új cégadatok elérhetők lesznek a profilban.

5. Kapcsolódó Információk (*ProjectEmployeCard*)

A *ProjectEmployeCard* komponens további adatokat jelenít meg három fő kategóriában:

* Munkatársak (*StaffTable*): A felhasználó által felügyelt munkatársak listája.
* Dokumentumok (*DocumentsTable*): A felhasználóhoz kapcsolódó dokumentumok kezelése és letöltése.
* Eszközök (*EquipmentsTable*): Az eszközök nyilvántartása és állapotának nyomon követése.

**6. Felhasználói Élmény**

Az oldal reszponzív kialakítású, amely minden eszközön jól működik. Támogatja a sötét és világos módot (*isDarkMode*), amely vizuálisan testre szabhatóvá teszi a felületet. Az interaktív elemek intuitív használatot biztosítanak, miközben az adatkezelés biztonságos és hatékony marad.

A screenshot of a computer

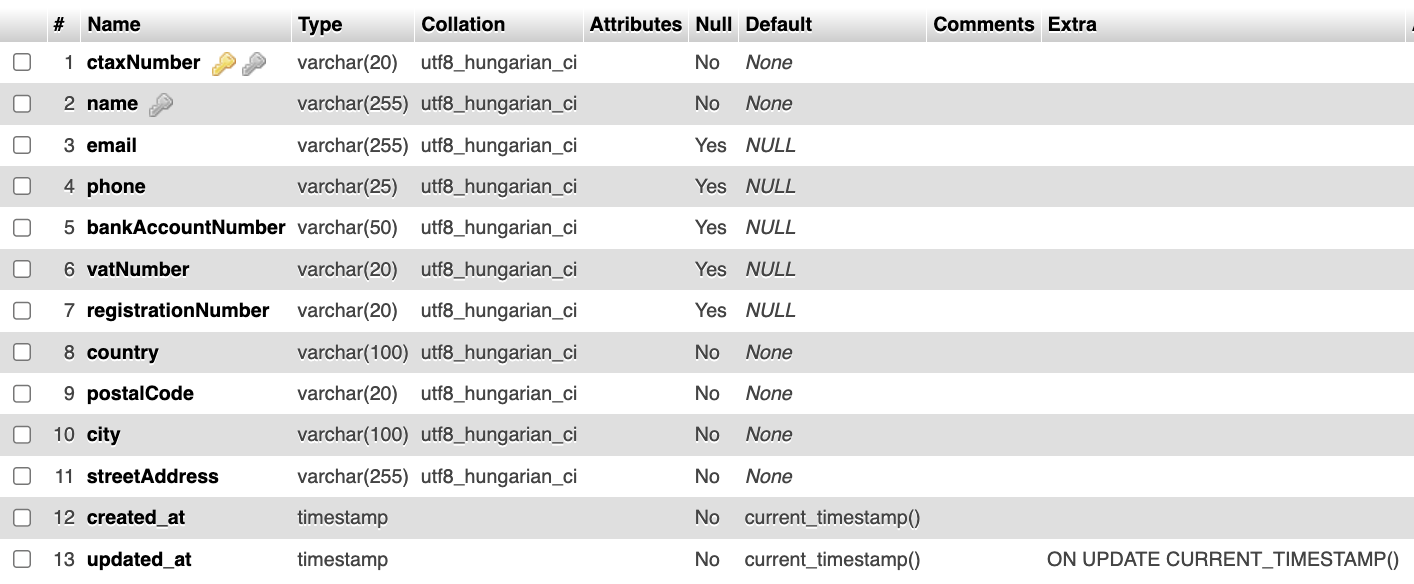
AI-generated content may be incorrect.Ez az oldal átfogó eszközt nyújt a felhasználói profilok kezelésére, beleértve az adatok módosítását, kapcsolódó információk megjelenítését és műveletek végrehajtását egy modern és professzionális környezetben.

# Adatbázis Elemzés

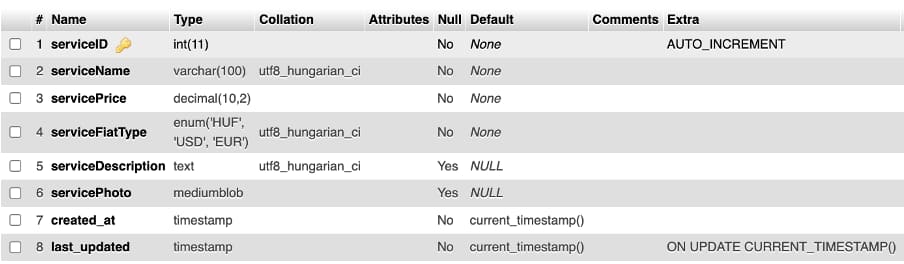
A Syncrohub rendszer adatbázis-struktúrája komplex entitáskapcsolati modellre épül, amely hatékonyan támogatja a vállalati folyamatok és erőforrások kezelését. Az alábbiakban részletezem az adatbázis főbb komponenseit és azok kapcsolatait. A karakterkódolás utf8mb4\_hungarian\_ci, ami a magyar nyelv helyes kezelését teszi lehetővé.

## Fő adattáblák és funkcióik:

**company (Cég) tábla:**

Ez az adatbázistábla egy vállalkozásokkal kapcsolatos adatokat tároló struktúrát mutat be, amelyben a legfontosabb azonosító a ctaxNumber (adószám), ami egyedi kulcsként szolgál. A name oszlop a vállalkozás nevét tartalmazza, míg az elérhetőségeket a email, phone, és a streetAddress, valamint a city, postalCode, és country mezők írják le. Pénzügyi adatok, mint például a bankszámlaszám (bankAccountNumber) és az adószám (vatNumber), szintén szerepelnek. A registrationNumber a cég hivatalos regisztrációs számát tárolja. Az időbélyegzők, created\_at és updated\_at, automatikusan rögzítik az adatok létrehozásának és frissítésének idejét.

**services (Szolgáltatások) tábla:**

Ez a táblázat egy szolgáltatásokkal kapcsolatos adatokat tároló struktúrát mutat be. A serviceID mező az elsődleges kulcs, amely automatikusan növekvő értéket vesz fel. A serviceName a szolgáltatás nevét tartalmazza, míg a servicePrice mező a szolgáltatás árát rögzíti, két tizedesjegy pontossággal. A serviceFiatType mező előre meghatározott valuták (HUF, USD, EUR) közül választja ki az árat kifejező pénznemet. A serviceDescription szöveges formában részletezi a szolgáltatást, míg a servicePhoto mező bináris adatokat tárol, például képeket. Az időbélyegzők, created\_at és last\_updated, automatikusan rögzítik az adatok létrehozásának és utolsó módosításának idejét.

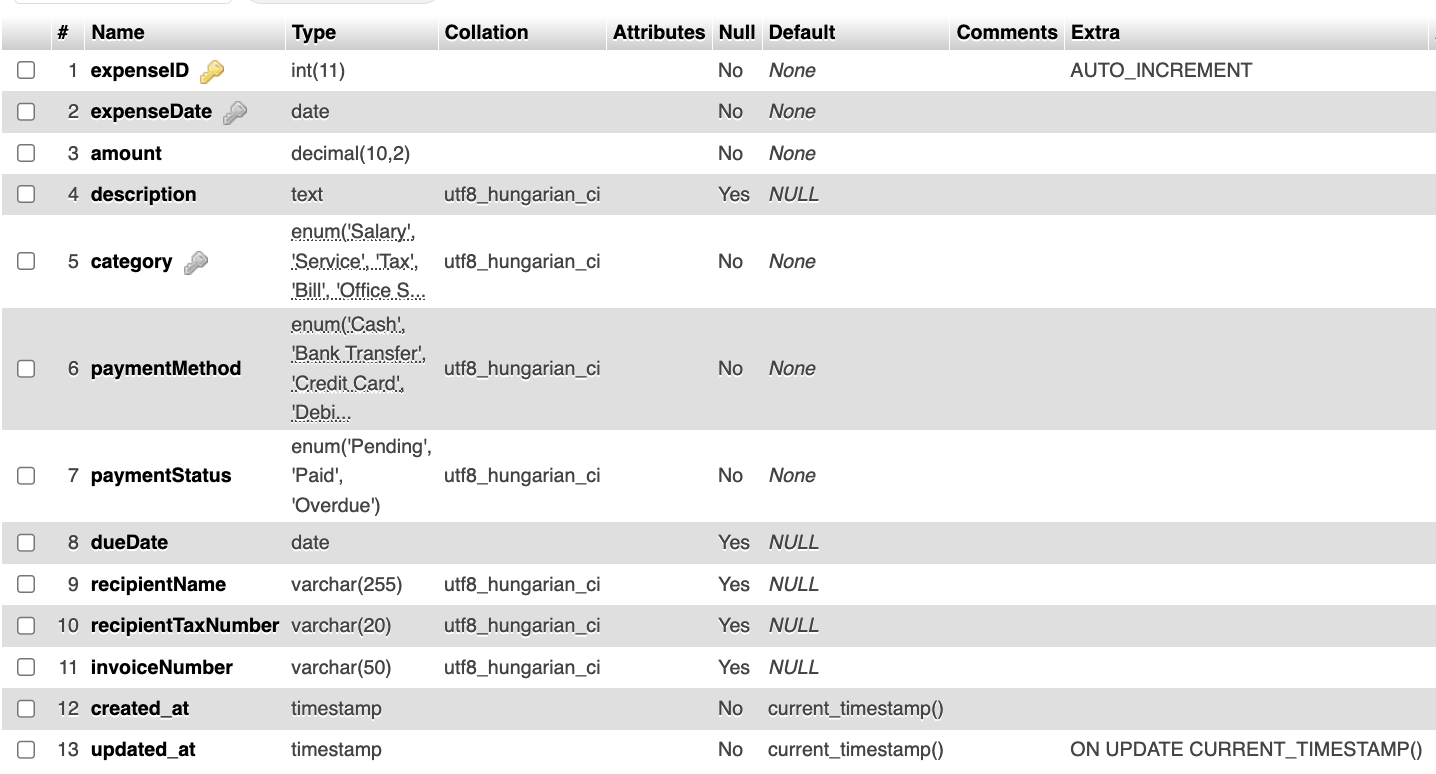
**income (Bevétel) tábla:**

Ez az adatbázistábla számlázási adatokat tárol. Az incomeID az elsődleges kulcs, amely automatikusan növekvő értékeket vesz fel. Az invoiceNumber a számla egyedi azonosítója, míg a serviceID a kapcsolódó szolgáltatás azonosítóját tartalmazza. A price mező a szolgáltatás árát rögzíti, az invoiceDate és a dueDate pedig a számla kiállításának és esedékességének dátumát tárolja. A paymentStatus mező a fizetési állapotot jelzi (például „Pending”, „Paid”, „Overdue”), míg a paymentMethod a fizetési módot rögzíti (például „Cash”, „Bank Transfer”, „Credit Card”). A quantity, taxRate, taxAmount, és totalAmount mezők az összegzésekhez szükséges adatokat tartalmazzák, beleértve a mennyiséget, az adókulcsot, az adó összegét és a teljes összeget. A description és a notes mezők opcionálisak, és további leíró információkhoz használhatók. Az időbélyegzők, a created\_at és az updated\_at, automatikusan naplózzák az adatok létrehozásának és utolsó módosításának időpontját, utóbbi automatikusan frissül minden változtatáskor (ON UPDATE A screenshot of a computer

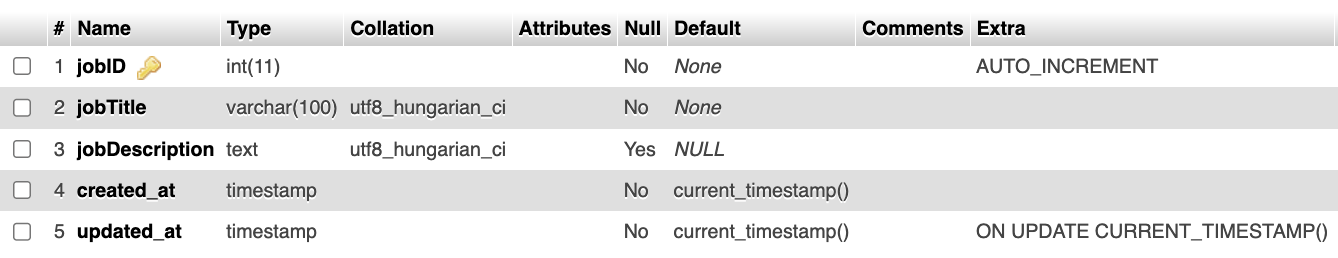
AI-generated content may be incorrect.CURRENT\_TIMESTAMP).

**expenses (Kiadások) tábla:**

Ez a tábla kiadásokkal kapcsolatos adatokat tárol. Az expenseID az elsődleges kulcs, amely automatikusan növekvő értéket vesz fel. Az expenseDate a kiadás dátumát rögzíti, míg az amount mező a kiadás összegét tárolja két tizedesjegy pontossággal. A description szöveges leírást tartalmaz a kiadásról. A category mező előre meghatározott kategóriák (pl. fizetés, szolgáltatás, adó) közül választja ki a kiadás típusát. A paymentMethod a fizetési módot (pl. készpénz, banki átutalás) jelzi, míg a paymentStatus a fizetés állapotát (függőben, kifizetve, lejárt) rögzíti. Az opcionális dueDate mező a fizetési határidőt tartalmazza. A recipientName és a recipientTaxNumber mezők a kedvezményezett nevét és adószámát tárolják, az InvoiceNumber pedig az esetleges számla azonosítóját tartalmazza. Az időbélyegzők, created\_at és updated\_at, automatikusan naplózzák az adatok létrehozását és frissítését.

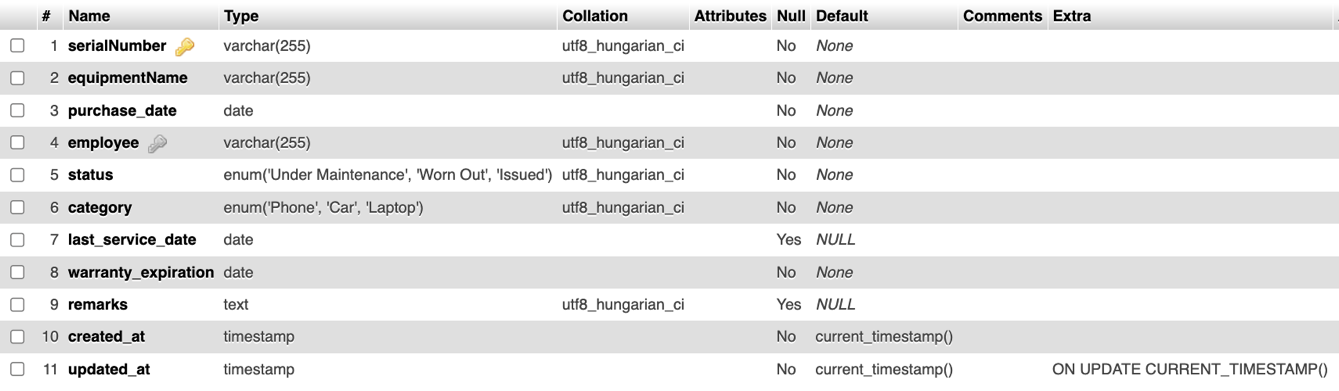


**jobs (Munkakörök) tábla:**

Ez a tábla álláshirdetésekkel kapcsolatos adatokat tárol. Az jobID az elsődleges kulcs, amely automatikusan növekvő értéket vesz fel (AUTO\_INCREMENT). A jobTitle mező az állás megnevezését tartalmazza, legfeljebb 100 karakter hosszúságban. A jobDescription szöveges formában részletezi az állást, és lehetővé teszi, hogy NULL értéket vegyen fel, ha nincs megadva leírás. Az időbélyegzők közül a created\_at mező automatikusan rögzíti az állás létrehozásának időpontját, míg az updated\_at mező a legutóbbi módosítás idejét naplózza, és automatikusan frissül minden változtatáskor (ON UPDATE CURRENT\_TIMESTAMP).

**equipments (Eszközök) Tábla:**

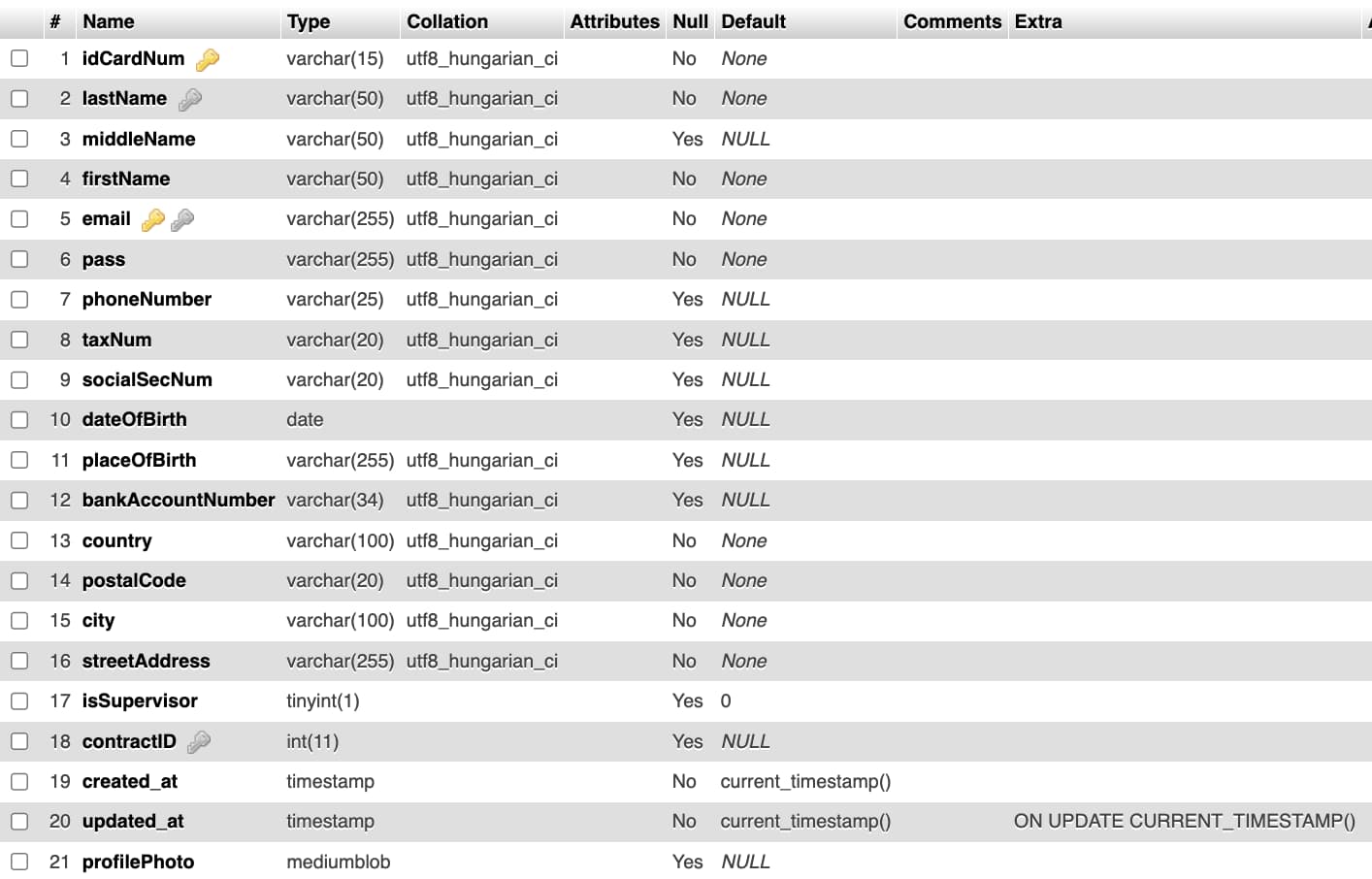
Ez a tábla eszközök nyilvántartására szolgál. A serialNumber az elsődleges kulcs, amely minden eszközt egyedileg azonosít. Az equipmentName mező az eszköz nevét tárolja, míg a purchase\_date az eszköz beszerzésének dátumát rögzíti. Az employee mező az eszközt használó munkavállaló nevét tartalmazza. A status mező előre meghatározott állapotokat rögzít (például "Karbantartás alatt", "Elhasználódott", "Kiadva"). A category mező az eszköz típusát jelzi (például "Telefon", "Autó", "Laptop").

A last\_service\_date az utolsó szervizelés dátumát, míg a warranty\_expiration a garancia lejáratának dátumát tárolja. A remarks mező szöveges megjegyzésekhez használható, és opcionálisan lehet kitöltve. Az időbélyegzők, a created\_at és az updated\_at, automatikusan rögzítik az adatok létrehozásának és frissítésének időpontját, utóbbi automatikusan frissül minden módosításkor (ON UPDATE CURRENT\_TIMESTAMP).

**employees (Alkalmazottak) Tábla:**

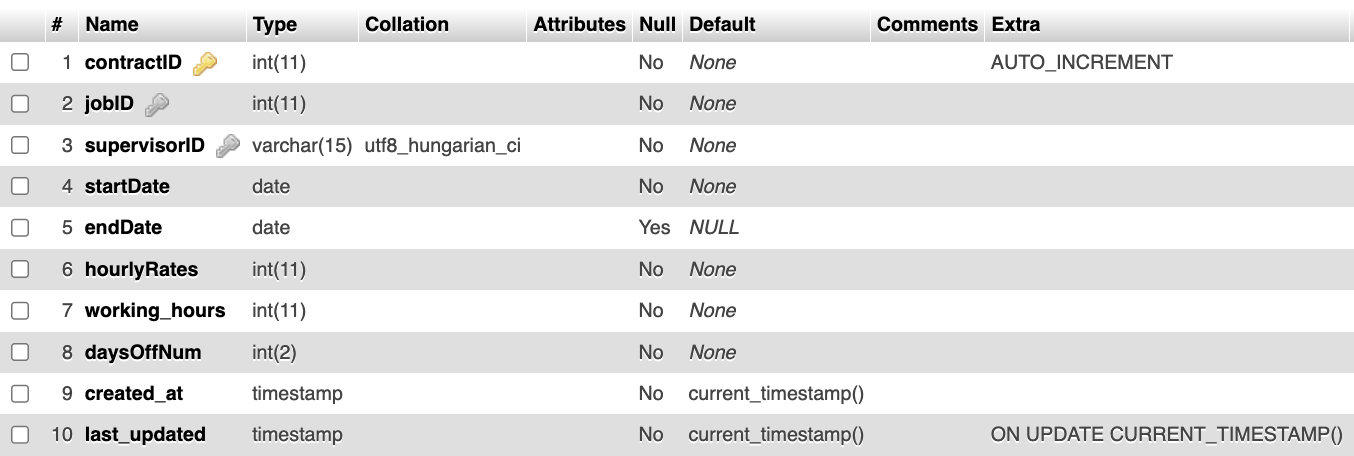
Ez a tábla személyes és kapcsolódó adatokat tárol. Az idCardNum az elsődleges kulcs, amely az egyedi azonosítást biztosítja. A lastName, middleName, és firstName mezők a személy nevét rögzítik, míg az email mező az elektronikus elérhetőséget tartalmazza, szintén egyedi kulcsként. A pass mező a jelszót tárolja, titkosításra alkalmas formátumban. A kapcsolati adatok között szerepel a phoneNumber, valamint a pénzügyi adatokhoz tartozik a bankAccountNumber és az adószám (taxNum).

A személyazonosító információkat a socialSecNum, a születési dátum (dateOfBirth), és a születési hely (placeOfBirth) mezők tartalmazzák. A címadatokat a country, postalCode, city, és streetAddress mezők rögzítik. Az isSupervisor mező logikai értéket tárol, amely azt jelzi, hogy az adott személy vezető-e. Az contractID mező egy külső hivatkozásként szolgál egy másik táblára.

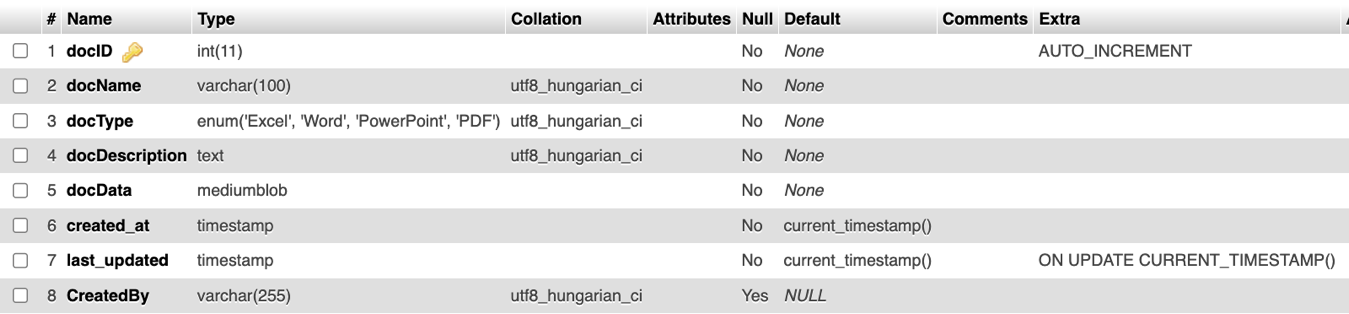
Az időbélyegzők, created\_at és updated\_at, automatikusan rögzítik az adatok létrehozásának és módosításának idejét, utóbbi automatikusan frissül minden változtatáskor. A profilePhoto mező bináris formátumban tárolja a személy fényképét, amely lehetővé teszi képek kezelését is.

**employmentContracts (Munkaszerződések) Tábla:**

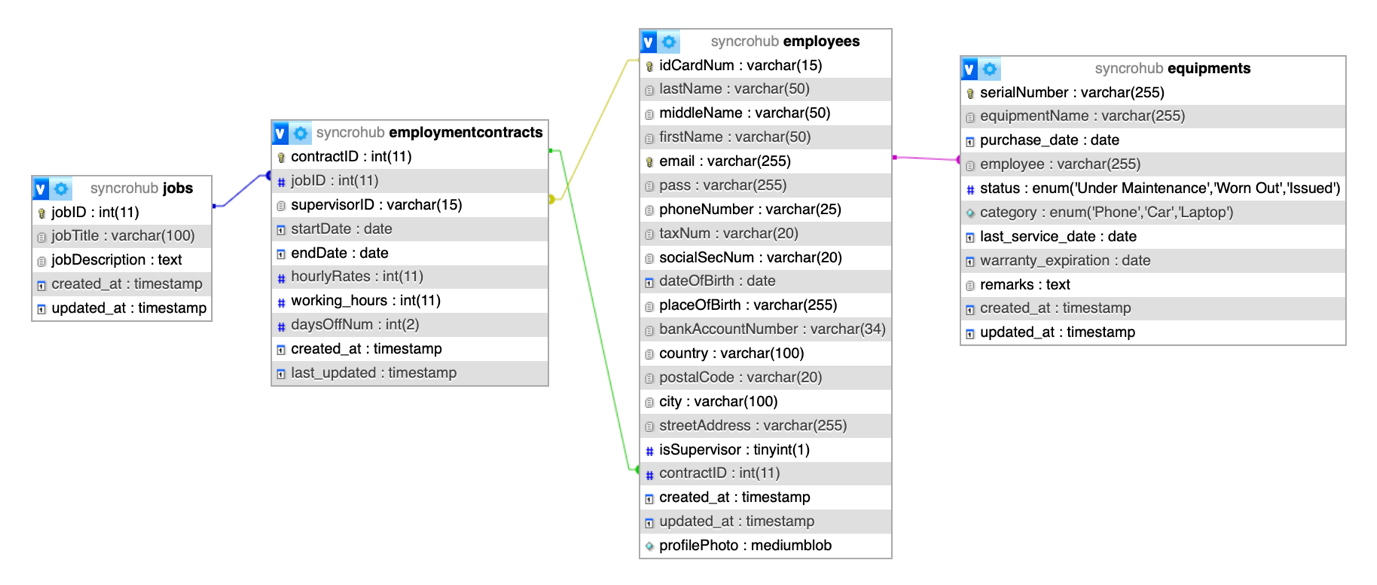
Ez a tábla munkaszerződésekkel kapcsolatos adatokat tárol. A contractID az elsődleges kulcs, amely automatikusan növekvő értéket vesz fel. A jobID mező külső hivatkozásként szolgál az állásokat tartalmazó táblára, míg a supervisorID a felügyelő azonosítóját tárolja. A startDate és az opcionális endDate mezők a szerződés kezdetének és végének dátumát rögzítik.

A hourlyRates mező az órabért, a working\_hours a heti munkaórák számát, míg a daysOffNum az éves szabadságok számát tartalmazza. Az időbélyegzők, a created\_at és a last\_updated, automatikusan naplózzák a szerződés létrehozásának és utolsó módosításának idejét, utóbbi automatikusan frissül minden változtatáskor.

**documents (Dokumentumok) Tábla:**

Ez az adatbázistábla dokumentumok kezelésére szolgál. A docID az elsődleges kulcs, amely automatikusan növekvő értéket vesz fel (AUTO\_INCREMENT). A docName mező a dokumentum nevét tárolja, legfeljebb 100 karakter hosszúságban. A docType mező előre meghatározott típusokat tartalmaz (például „Excel”, „Word”, „PowerPoint”, „PDF”), amelyek azonosítják a dokumentum formátumát. A docDescription mező szöveges leírást biztosít a dokumentumhoz. A docData mező bináris adatokat tárol, például a dokumentum tartalmát vagy fájlját. Az időbélyegzők közül a created\_at mező automatikusan rögzíti a dokumentum létrehozásának időpontját, míg a last\_updated mező az utolsó módosítás idejét naplózza, és automatikusan frissül minden változtatáskor (ON UPDATE CURRENT\_TIMESTAMP). A CreatedBy mező opcionális, és azt tárolja, hogy ki hozta létre a dokumentumot.

## Táblák Közötti Kapcsolatok:

Ez az adatbázis több táblát tartalmaz, amelyek egymással kapcsolatban állnak, és különböző típusú adatokat tárolnak. Az employees tábla a munkavállalók személyes és kapcsolati adatait tartalmazza, amely a employmentcontracts táblával van összekapcsolva az contractID mezőn keresztül. Az employmentcontracts tábla a munkaszerződések adatait tárolja, és a jobs táblával is kapcsolatban áll a jobID mezőn keresztül, amely az állások részleteit rögzíti. Az equipments tábla az eszközöket tartalmazza, és kapcsolódik az employees táblához az employee mezőn keresztül, amely azt jelzi, hogy melyik munkavállaló használja az adott eszközt. A documents tábla dokumentumok kezelésére szolgál, de jelenleg nincs közvetlen kapcsolatban más táblákkal.

A screenshot of a computer

AI-generated content may be incorrect.Az income tábla a számlázási adatokat tartalmazza, és kapcsolódik a services táblához a serviceID mezőn keresztül, amely a nyújtott szolgáltatások adatait tárolja.

## ER Diagram:

# BACK-END

## Felhasználói Profilkezelés Modell:

1. **Jelszó-visszaállítási folyamat modellek**

A jelszó-visszaállítás három lépését kezelő DTO-k (*Data Transfer Objects*):

* PasswordRecoveryRequestDto: E-mail cím validációja a visszaállítás kezdeményezéséhez ([Required], [EmailAddress]).
* VerifyCodeDto: 6 karakteres ellenőrzőkód és e-mail cím validációja.
* ResetPasswordDto: Új jelszó beállítása (minimum 8, maximum 100 karakter) a kóddal és e-mail címmel együtt.

1. **Profilkép kezelés**

* ProfilePhotoUploadDto: Profilkép feltöltéséhez szükséges modell, amely egy fájlt (*IFormFile*) kötelezően megkövetel.

1. **Személyes adatok frissítése**

* PersonalDataUpdateDto: E-mail, telefonszám, bankszámlaszám, adószám és társadalombiztosítási szám módosítására szolgál.
* AddressUpdateDto: Címadatok frissítése (ország, irányítószám, város, utca).
* BirthDataUpdateDto: Születési adatok (név, születési hely, dátum) kezelése opcionális középső névvel.

1. **Jelszó módosítás**

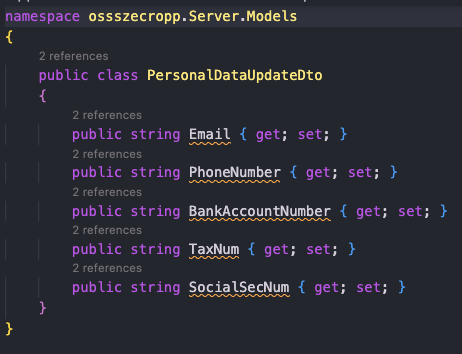
* ChangePasswordDto: Régi és új jelszó megadását követeli meg, mindkettőt kötelezően validálva.

1. **Alkalmazotti adatok frissítése**

* EmployeeUpdateDto: Átfogó modell, amely egyesíti a személyes adatokat (név, e-mail), cím adatokat, születési adatokat, pénzügyi információkat (bankszámla, adószám) és egyéb opcionális mezőket.

A screenshot of a computer program

AI-generated content may be incorrect.



**Közös jellemzők**

* **Névtér**: Minden modell az osszecropp.Server.Models névtérben található [1-7].
* **Validáció**: A kötelező mezőket [Required], az e-mail formátumot [EmailAddress], a jelszó hosszt [StringLength] attribútumokkal ellenőrzik.
* **Opcionális mezők:** A MiddleName (középső név) és DateOfBirth (születési dátum) például nullázhatóak (?), ha nem kötelezőek.

A screenshot of a computer program

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.A modellek reszponzív adatkezelést tesznek lehetővé, és integrálhatók ASP.NET Core alapú backend rendszerekbe. A validációs szabályok biztosítják az adatok integritását, míg a modularitás egyszerű bővítést tesz lehetővé.

## Eszközök és Dokumentumok Kezelése Back-End:

1. **Eszközkezelés Modellek (*EquipmentDto*.*cs*)**

Az EquipmentDto osztály az eszközök adatainak átvitelére szolgál:

**A screenshot of a computer program

AI-generated content may be incorrect.Kötelező mezők:**

* SerialNumber: Egyedi sorozatszám.
* EquipmentName: Eszköz neve.
* Status: Állapot (pl. "Kiadva", "Karbantartás alatt").

**Opcionális mezők:**

* Employee: Felelős munkavállaló neve.
* Category: Eszköz kategóriája (pl. "Laptop").
* PurchaseDate, WarrantyExpiration, LastServiceDate: Dátumok string formátumban.
* Remarks: Megjegyzések.

1. **Dokumentumkezelés Modellek (*DocumentDto*.*cs*)**

A dokumentumkezelés több DTO-t tartalmaz:

1. **Alapmodell** (*DocumentDto*)

* DocID: Egyedi azonosító (opcionális feltöltéskor).
* DocName: Dokumentum neve (max. 255 karakter).
* DocType: Típus (pl. "PDF", max. 50 karakter).
* DocDescription: Leírás (opcionális).
* DocFile: Fájl feltöltéshez (*IFormFile*).

1. **Feltöltési modell** (*DocumentUploadDto*)

* Validációs hibák kezelése: kötelező név, típus és fájl.

1. **Frissítési modell** (*DocumentUpdateDto*)

* Opcionális mezők: Lehetőség részleges adatmódosításra (pl. csak megjegyzés hozzáadása).

1. **Adatbázis entitás** (Document)

* Bináris adattárolás: DocFile mező byte[] típusban.
* Automatikus időbélyegek: CreatedAt.

**Közös jellemzők**

* **Névtér**: Mindkét modell az ossszecropp.Server.Models névtérben található.
* **Validáció**: Kötelező mezők ([Required]), hosszkorlátok ([StringLength]).
* **Dátumkezelés**: Dátumok string ként kerülnek tárolásra az egyszerűség kedvéért.

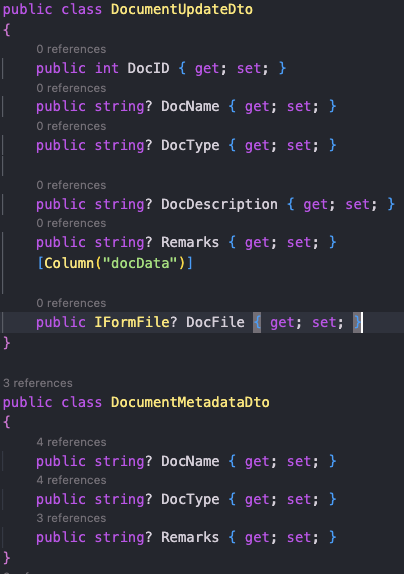
**Működési elv:**

* **Eszközök**: Az *EquipmentDto* segítségével lehet létrehozni, frissíteni vagy lekérdezni eszközöket. A *SerialNumber* garantálja az egyediségét.
* **Dokumentumok**: A *DocumentUploadDto* kezeli a fájlfeltöltést, míg a *Document* entitás tárolja az adatbázisban bináris formátumban. A *DocType* mező szabványos típusokat kényszerít ki.



A screen shot of a computer program

AI-generated content may be incorrect.



## Szolgáltatások és Pénzügyi Adatok Back-End:

**1. Szolgáltatáskezelés (Service.cs)**

**a) Service osztály**

Ez az osztály a szolgáltatások alapvető adatait tárolja:

* **ServiceID**: Egyedi azonosító (nullable, opcionális).
* **ServiceName**: A szolgáltatás neve (nullable).
* **ServicePrice**: A szolgáltatás ára, két tizedesjegy pontossággal (nullable).
* **ServiceFiatType**: A pénznem típusa (pl. HUF, USD, EUR; nullable).
* **ServiceDescription**: Szöveges leírás (nullable).
* **Created\_at** és **Last\_updated:** Időbélyegek a létrehozás és módosítás rögzítésére.
* **ServiceIcon**: Új mező, amely a szolgáltatáshoz tartozó ikon URL-jét tárolja.

**b) DTO-k (Data Transfer Objects)**

A *ServiceCreateDto* és *ServiceUpdateDto* osztályok lehetővé teszik az új szolgáltatások létrehozását és meglévők frissítését:

* Tartalmazzák a *ServiceName*, *ServicePrice*, *ServiceFiatType*, *ServiceDescription*, és az új *ServiceIcon* mezőt.
* Az adatok validálása opcionális (nullable).

**2. Bevételkezelés (Income.cs)**

**a) Income osztály**

Ez az osztály a bevételek részletes nyilvántartására szolgál:

* **Kulcsmezők**:
  + incomeID: Egyedi azonosító, automatikusan generált.
  + invoiceNumber: Számla egyedi azonosítója (max. 20 karakter).
  + companyID: Kapcsolódó cég azonosítója (max. 20 karakter).
  + serviceID: Kapcsolódó szolgáltatás azonosítója.
* **Pénzügyi adatok:** 
  + price: Szolgáltatás ára (két tizedesjegy pontosságú decimális érték).
  + taxRate, taxAmount, és totalAmount: Adókulcs, adó összege, és végösszeg.
* Dátumok:
  + invoiceDate: Számla kiállításának dátuma.
  + dueDate: Fizetési határidő.
* Egyéb mezők:
  + paymentStatus: Fizetési állapot (pl. Pending, Paid).
  + paymentMethod: Fizetési mód (pl. Bank Transfer, Cash).
  + Opcionális mezők: Leírás (description) és megjegyzések (notes).

**b) Időbélyegek**

A created\_at és updated\_at mezők automatikusan rögzítik a bevételek létrehozásának és módosításának időpontját.

**3. Kiadáskezelés (Expense.cs)**

**a) Expense osztály**

Ez az osztály a kiadások részletes kezelésére szolgál:

* **Kulcsmezők:**
  + expenseID: Egyedi azonosító, automatikusan generált.
* **Pénzügyi adatok:**
  + amount: Kiadás összege (két tizedesjegy pontosságú decimális érték).
  + Opcionális mezők: Számlaazonosító (invoiceNumber) és kedvezményezett neve/adószáma (recipientName, recipientTaxNumber).
* **Dátumok:** 
  + expenseDate: Kiadás dátuma.
  + Opcionális mező: Fizetési határidő (dueDate).
* **Kategóriák és állapotok:** 
  + category: Kiadás típusa (pl. fizetés, szolgáltatás).
  + paymentStatus: Fizetési állapot (pl. Pending, Paid).

**b) Időbélyegek**

A kiadások létrehozásának és frissítésének időpontját a created\_at és az updated\_at mezők rögzítik.

**Közös Jellemzők**

1. **Validációs attribútumok**: Az összes kötelező mezőt [Required], míg a szövegmezőket [StringLength] attribútumokkal validálják.
2. **Adattípusok pontosítása**: A pénzügyi értékek pontosan definiáltak (decimal(10,2)), míg a dátumokat szabványos formátumban tárolják (date).
3. **Modularitás**: A DTO-k külön kezelik a létrehozási és frissítési műveleteket.
4. **Integráció lehetőségei**: Ezek a modellek könnyen integrálhatók API-alapú rendszerekbe ASP.NET Core környezetben.

## Cég Adatok Nyilvántartása Back-End Modell:

Az alábbiakban a cégadatok kezelésére szolgáló back-end modell részletei találhatók, a CompanyDto osztály alapján:

1. **Modell Áttekintése (CompanyDto):**

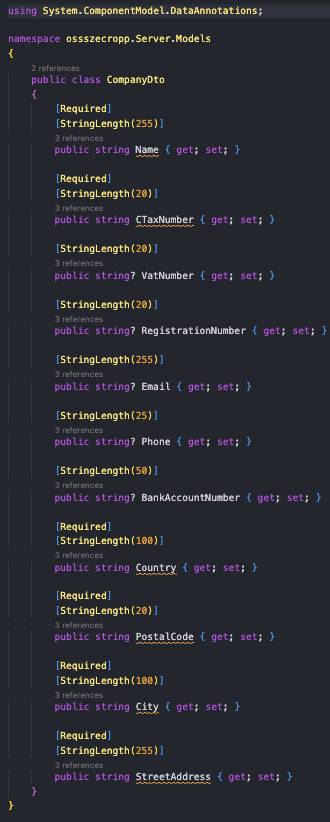
A CompanyDto osztály a cégadatok nyilvántartására és átvitelére szolgál. Az osztály a következő mezőket tartalmazza:

**Kötelező mezők:**

* Name: A cég neve, maximum 255 karakter hosszúságú.
* CTaxNumber: A cég adószáma, maximum 20 karakter.
* Country: Az ország neve, maximum 100 karakter.
* PostalCode: Az irányítószám, maximum 20 karakter.
* City: A város neve, maximum 100 karakter.
* StreetAddress: Az utca és házszám, maximum 255 karakter.

**Opcionális mezők:**

* VatNumber: Az áfa (VAT) azonosító, maximum 20 karakter.
* RegistrationNumber: A cég hivatalos regisztrációs száma, maximum 20 karakter.
* Email: Elektronikus levelezési cím, maximum 255 karakter.



* Phone: Telefonszám, maximum 25 karakter.
* BankAccountNumber: Bankszámlaszám, maximum 50 karakter.

1. **Validáció**

Az osztály validációs attribútumokat használ az adatok integritásának biztosítására:

* [Required]: Kötelező mezők megjelölése (pl. Name, CTaxNumber, Country).
* [StringLength]: A mezők maximális hosszának korlátozása a túlméretezett adatok elkerülésére (pl. Name: max. 255 karakter).

1. **Felhasználási Szenáriók**

A CompanyDto modell az alábbi műveletekhez használható:

1. **Új cég létrehozása:**

* Kötelező mezők kitöltése (pl. név, adószám, címadatok).
* Opcionális mezők megadása (pl. e-mail, telefonszám).

1. **Cégadatok frissítése:**

* Egyedi azonosító alapján (CTaxNumber) meglévő adatok módosítása.
* Csak a szükséges mezők frissítése (pl. új bankszámlaszám hozzáadása).

1. **Cégek listázása és keresése:**

* Szűrés opcionális mezők alapján (pl. város vagy ország szerint).

1. **Adatellenőrzés API-n keresztül:**

* Validációs hibák visszajelzése a front-end felé (pl. túl hosszú név vagy hiányzó kötelező mező).

1. **Integrációs Lehetőségek**

A CompanyDto könnyen integrálható ASP.NET Core API-kba:

* Adatbázis-kezeléshez: Használható entitásokhoz kapcsolódó DTO-ként az adatbázisból érkező vagy oda mentett adatok kezelésére.
* REST API-khoz: Alkalmas bejövő és kimenő HTTP kérések adatainak kezelésére.
* Validációs hibák kezelése: Az attribútumok automatikusan generálnak hibajelzéseket nem megfelelő bemenet esetén.

1. **Előnyök**

* Egyszerű és jól strukturált modell.
* Könnyen bővíthető további mezőkkel (pl. logó URL-je vagy kapcsolattartó személy neve).
* Biztosítja az adatok konzisztenciáját és integritását.

# Tesztelési Folyamat

A tesztelési folyamat során különböző tesztelési módszereket alkalmaztunk, amelyek biztosították az alkalmazás megbízhatóságát és funkcionalitását. Az alábbiakban részletesen bemutatjuk a használt technológiákat és azok működését:

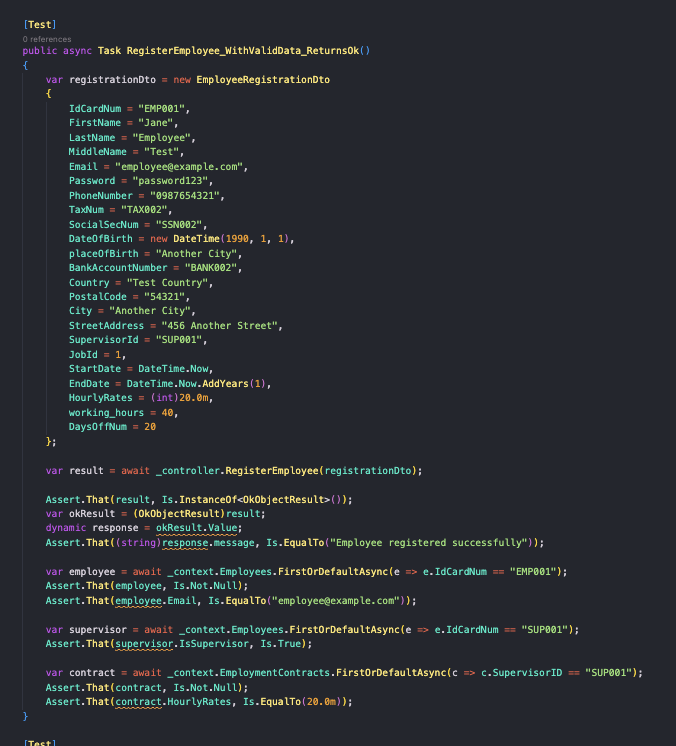
## Egységtesztek (NUnit Framework)

Az egységtesztekhez az NUnit keretrendszert használtuk, amely a .NET ökoszisztéma egyik legnépszerűbb tesztelő eszköze. Az NUnit lehetővé tette:

* Attribútum-alapú tesztelést: Az olyan attribútumok, mint a [Test], [SetUp], és [TestCase], megkönnyítették a tesztek definiálását és konfigurálását.
* Párhuzamos futtatást: Az NUnit támogatja a párhuzamos tesztvégrehajtást, amely jelentősen csökkentette a tesztelési időt.
* Adatvezérelt tesztelést: A [TestCase] attribútum segítségével különböző bemeneti értékekkel futtathattuk ugyanazt a tesztet, így biztosítva a kód robusztusságát.

Például az EmployeeRegistrationControllerTests fájlban az egységtesztek ellenőrizték az alkalmazott regisztrációjának helyes működését, beleértve az érvényes adatokkal történő sikeres regisztrációt, a duplikált adatok kezelését, valamint az érvénytelen bemenetek hibakezelését.

A screenshot of a computer program

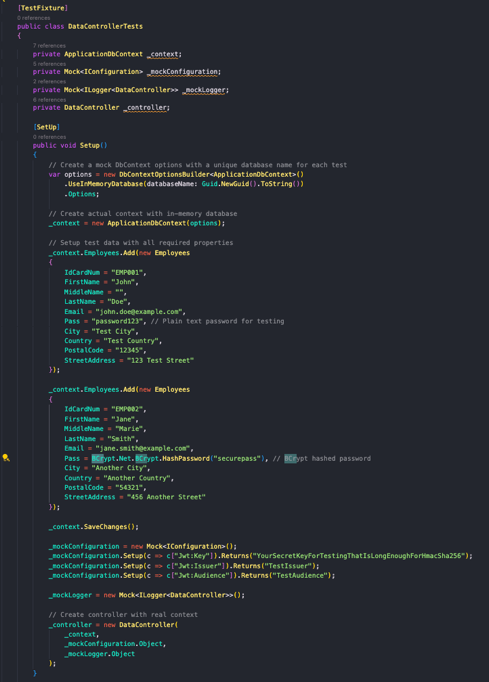
AI-generated content may be incorrect.

## Automatizált Tesztek (Selenium WebDriver)

Az automatizált teszteléshez a Selenium WebDriver-t használtuk, amely lehetővé tette:

* **Böngészőfüggetlen tesztelést**: A Selenium támogatja több böngészőt, például Chrome, Firefox, Edge stb., így biztosítva az alkalmazás kompatibilitását különböző platformokon.
* **Felhasználói interakciók szimulálását**: A Selenium segítségével automatizáltuk az olyan műveleteket, mint űrlapok kitöltése, gombok kattintása, és adatbeviteli mezők ellenőrzése.
* **CI/CD integrációt**: A Selenium könnyen integrálható CI/CD folyamatokba, amely lehetővé tette a folyamatos fejlesztés során végzett automatikus tesztelést.

Például az DataControllerTests fájlban a Selenium segítségével validáltuk a felhasználói bejelentkezési funkciókat, beleértve az érvényes hitelesítést és hibás jelszó esetén történő megfelelő visszajelzést.



## Kódfelülvizsgálatok

Rendszeres kódfelülvizsgálatokat végeztünk annak érdekében, hogy biztosítsuk a kód minőségét és megfelelőségét. A felülvizsgálatok során ellenőriztük a kód logikai helyességét, olvashatóságát és optimalizáltságát.

## Felhasználói Elfogadási Tesztek (UAT)

A fejlesztési ciklus végén felhasználói elfogadási teszteket végeztünk, amelyek során kulcsfontosságú érintettek validálták az alkalmazás funkcionalitását valós üzleti forgatókönyvek alapján.

## Kódok Működése

* A screenshot of a computer

  AI-generated content may be incorrect.Az EquipmentControllerTests fájl például ellenőrizte az eszközök kezelését:
* Az eszközök listázása (GetEquipment) biztosította, hogy minden eszköz helyesen jelenjen meg.
* Az új eszköz hozzáadása (CreateEquipment) validálta az adatbevitel helyességét.
* Az eszközök frissítése (UpdateEquipmentBySerial) és törlése (DeleteEquipment) garantálta az adatok pontos kezelését.

A DocumentsControllerTests fájlban pedig dokumentumkezelési funkciókat teszteltünk:

* Dokumentumok feltöltése (UploadDocumentFile) biztosította a fájlok megfelelő mentését.
* A dokumentum metaadatainak frissítése (UpdateDocumentMetadata) validálta a módosításokat

# Jövőbeli Fejlesztések

A jövőbeli fejlesztések során tervezzük a mesterséges intelligencia és a gépi tanulás integrálását a rendszerbe, hogy még hatékonyabbá tegyük a vállalatok irányítását és a döntéshozatali folyamatokat. Konkrét tervek között szerepel:

A prediktív analitikai eszközök lehetővé teszik a múltbeli és aktuális adatok elemzését, hogy azonosítsák a potenciális kockázatokat, például késéseket vagy erőforrás-problémákat. Ezáltal a projektmenedzserek proaktívan kezelhetik a problémákat, megelőző intézkedéseket hozva a hatékonyabb projektvégrehajtás érdekében.

A chatbot asszisztens fejlesztése célja az ügyfélszolgálati folyamatok automatizálása és javítása. Az MI-alapú chatbot képes lesz valós idejű válaszokat adni, személyre szabott ügyfélélményt nyújtva. A rendszer integrálva lesz a CRM platformmal, hogy az ügyféladatok alapján releváns információkat szolgáltasson. Az AI asszisztens helyettesíti a Google Translate API-t, biztosítva az oldal automatikus fordítását több nyelvre. Ez javítja a felhasználói élményt és támogatja a globális kommunikációt.

Az automatizált jelentéskészítő modul személyre szabott üzleti elemzéseket készít a vállalati adatok alapján. Ez lehetővé teszi az adatok gyors feldolgozását és vizualizációját, támogatva az üzleti döntéshozatalt.

A Blockchain technológia bevezetése növeli a szerződéskezelés biztonságát és átláthatóságát. Az okosszerződések használata biztosítja az automatikus végrehajtást és az adatok integritását, csökkentve az adatmanipuláció kockázatát.

A dolgozók adatainak kiegészítése crypto tárcacímekkel lehetővé teszi fizetésük közvetlen utalását kriptovaluta formájában. Ez modern alternatívát kínál a hagyományos fizetési rendszerek mellett.

A projektek vizualizációját naptár nézetben tesszük elérhetővé, amely megkönnyíti a határidők és feladatok áttekintését. A naptár integrálva lesz más időkezelő eszközökkel.

# Felhasználói Visszajelzések

A projekt során kiemelt figyelmet fordítottunk a felhasználói visszajelzések gyűjtésére és azok beépítésére a fejlesztési folyamatba. A célunk az volt, hogy az alkalmazás ne csak funkcionálisan legyen megfelelő, hanem a felhasználók számára intuitív és könnyen használható felületet biztosítson. A visszajelzések strukturált gyűjtése és elemzése lehetővé tette számunkra, hogy folyamatosan javítsuk az alkalmazás minőségét és megfeleljünk a felhasználói igényeknek.