Szakmai Gyakorlat Munkanapló

Bauer Brúnó Csaba

2024. 12. 30.

Tartalomjegyzék

Első Hét

1.1. Első nap, projektfeladat ismertetése (09.04.)

Az első nap munkavédelmi, tűzvédelmi, és egyéb céges oktatásokon, valamint cégbemutatón vettem részt. Ez után következett a projektfeladatom ismertetése

Projektfeladat

A két hónapos időszak alatt egy webalkalmazást kell fejlesztenem a cég számára. A fő cél az, hogy egy olyan felületet hozzak létre, ahol az alkalmazottak nyomon követhetik a munkaidejüket, ugyanakkor fontos, hogy lehetőség legyen a későbbi bővítésekre is. További követelmény, hogy az autentikáció és a felhasználói adatok lekérdezése a cégen belül hamarosan bevezetésre kerülő *Active Directory* címtárszolgáltatás integrációjával történjen. A dolgozók beléptetése az *Entra-Pass Special Edition* szoftvercsomag segítségével történik, így ez lesz a forrása a beléptetési adatoknak.

1.2. Active Directory, tech stack, szerver telepítés, tervezés megkezdése (09.05.)

Mivel az Active Directory, illetve az *LDAP* protokoll számomra ismeretlen technológiák, ezért először ezekkel kezdtem el ismerkedni. (Fontosabb forrásokat lásd [2, 5])

Később rá is találtam egy JavaScript (TypeScript) könyvtárra `ldapts` néven (ldapts), amely könyvtár alkalmas lesz az Active Directory szerverrel – továbbiakban *AD szerver* – való kommunikálásra. Elkezdtem ismerkedni a libraryvel, sikerült vele az AD szerverhez kéréseket küldeni. A könyvtár — a webalkalmazás szempontjából – két legfontosabb függvénye:

```
bind(dnOrSaslMechanism, [password], [controls])
```

A `bind` függvény egy *bind* operációt kezdeményez az AD szerver felé. Ezzel az operációval autentikálhatjuk magunkat az AD szerver felé. Sikeres bind után, az azt követő kérések már az adott felhasználó nevével történnek.

```
search(baseDN, options, [controls])
```

A *search* operációval LDAP ,,entrykre" - pl. felhasználókra - kereshetünk. A kereséshez LDAP filtert kell írni. Mivel ezek a filterek – hasonlóan az SQL lekérdezésekhez – sérülékenyek injection támadásra, ezért escape-elni kell majd őket.

Tech Stack

Technológiák terén annyi megkötés volt, hogy lehetőleg olyan eszközökkel dolgozzak, amelyeket az egyetemen is megismerhettünk. Mivel a Web programozás 2 tantárgy keretében lehetőség nyílik a *SvelteKit* keretrendszerrel/meta frameworkkel való megismerkedésre, illetve nekem is a JavaScript alapú webfejlesztésben van a legnagyobb tapasztalatom, így erre esett a választásom.

Backend/Frontend:

- Nyelv: TypeScript
- Keretrendszer: SvelteKit
- CSS: Tailwind, Skeleton – UI Toolkit for Svelte and Tailwind
- ORM: drizzle ORM

Adatbázis:

- PostgreSQL

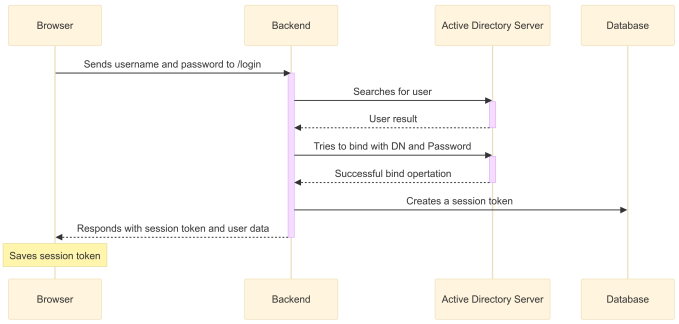
Szerver telepítés

A szerverhez biztosított a cég két 500 GB-os HDD-t, melyeket elsősorban beszereltem. A webszerverre Debian 12 disztribúciót telepítettem a stabilitása miatt. Telepítés során a két HDD-t szoftveres RAID1- be konfiguráltam. A RAID1

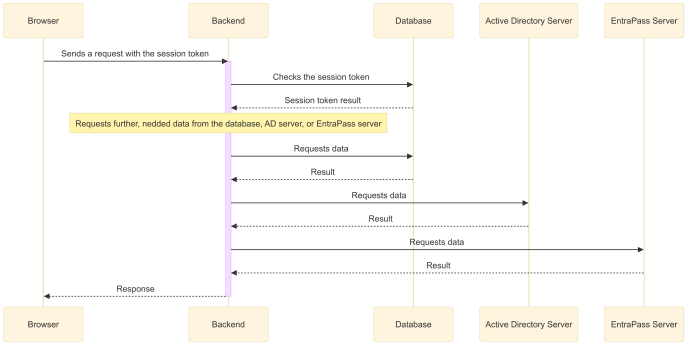
miatt ugyan lassabb lesz az írás a merevlemezre, valamint az 1 TB helyett mindössze 500 GB lesz elérhető, azonban a tükrözés miatt nagyobb lesz a hibatűrés.

Tervezés megkezése

Ezek után a projekt megkezdéséhez szükséges, kezdetleges terveket készítettem el. (Lásd az 1. és a 2. ábrákat)



1. ábra. Bejelentkezés



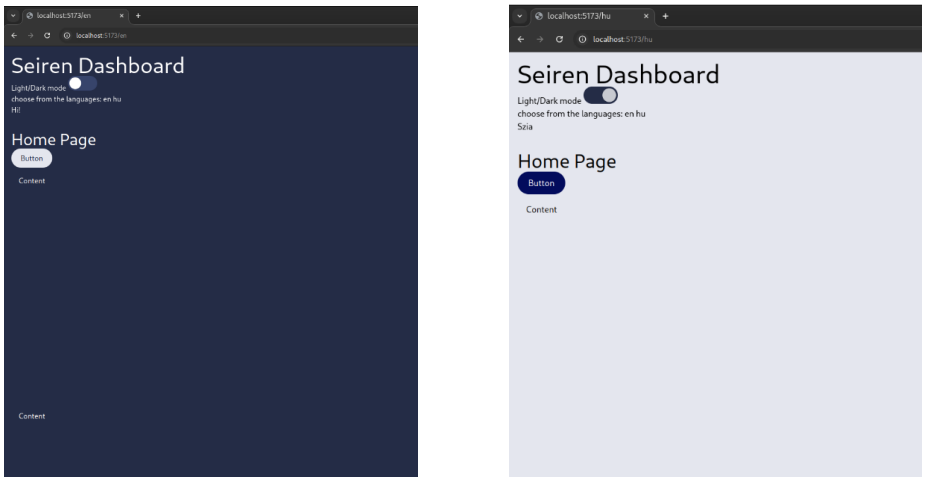
2. ábra. További kérések

1.3. SvelteKit projekt elkezdése (09.06.)

A mai nap a projekt elkezdésével telt. Létrehoztam a SvelteKit projektet, telepítettem, illetve beállítottam a Skeleton UI toolkitet, amely UI komponenseket biztosít a gyorsabb fejlesztés érdekében. Biome.js-t is telepítettem, amely egy fejlesztői eszköztár kód formázásra és ellenőrzésre (linting).

Bár az i18n (több nyelv támogatása) nem volt a projekt követelményeiben, mégis úgy döntöttem, hogy már az elején implementálom, hogy később — amennyiben szükség lesz rá – egyszerűbb legyen bővíteni a nyelvi beállításokat.

A felhasználói élmény növelése érdekében implementáltam a világos/sötét mód támogatást, így a felhasználók váltogathatnak a témák között. Végül létrehoztam az alkalmazás alapvető felépítését: tartalmaz egy fejlécet, egy fő tartalmi részt (body), és egy láblécet. (3. ábra)



(a) Sötét mód, angol nyelv (b) Világos mód, magyar nyelv

3. ábra. Sötét és világos mód

1.4. Active Directory VM, kód futtatása az alkalmazás indulásakor (09.07.)

Szükség volt egy dedikált Active Directory szerverre, hogy ne az éles szervert kelljen használnom a fejlesztéshez. Először egy Docker konténerre gondoltam, azonban mint kiderült, nincsen egy az eredetihez még csak hasonló megoldás sem, lévén az Active Directory egy Windows technológia. A megoldás egy Windows Server VM lett, aminek a telepítését dokumentáltam. Az Active Directory telepítéséhez nagy segítség volt a [3] YouTube videó.

Kód futtatása az alkalmazás indulásakor

Mivel a SvelteKit dokumentációjában nem találtam arra módszert, hogy hogyan kell futtatni kódot az alkalmazás indulásakor, így az interneten keresgéltem. Több módszert is találtam és ki is próbáltam. A végén arra a megoldásra esett a választásom, hogy a `hooks.server.ts` fájlba írom azt a kódot, amit induláskor futtatni kell. Egyetlen hátulütője ennek a megoldásnak az az, hogy dev módban (fejlesztés közben), nem történik meg automatikusan a kód futtatása, csak az első kérés beérkezése során. Erre az lett a megoldásom, hogy írtam egy bash

szkriptet, ami a dev script indulásával együtt lefut és `curl`- el kéréseket küld a szervernek addig, ameddig az nem válaszol.

1.5. Active Directory service, logolás (09.08.)

Az Active Directory service-t kezdem el írni (1. kód), vagyis azt a fájlt, amiben az AD szerverrel való kommunikációhoz szükséges függvények találhatók. Első körben egy `start` és egy `stop` függvényt implementáltam.

```
// lib/services/activeDirectoryService.ts
let ldapClient: LdapClient;
let reconnecting = false;

async function start(activeDirectoryConfig: LdapConfig) { ... }
async function startReconnecting() { ... }
async function stop() { ... }

const activeDirectoryService: ActiveDirectoryService = {
    start,
    stop,
};

export default activeDirectoryService;
```

1. kód. Active Directory service

A `start` függvény lényegében egy LDAP klienst inicializál, majd pedig egy bind operációt hajt végre az AD szerveren, az alkalmazás számára létrehozott admin felhasználó DN és jelszó párosával.

A különböző konfigurációk betöltéséért – mint például az AD szerver hosztja, vagy az előbb említett admin DN és jelszó -– külön modulok felelnek a `lib/configs` mappában. Ezek a modulok a környezeti változókat olvassák be az alkalmazás indulásakor (`hooks.server.ts`)

Fontosnak tartom, hogy az alkalmazás a lehető legjobban hibatűrő legyen, ezért azt is implementáltam a start függvénybe, hogy ha nem sikerül csatlakozni az AD szerverhez, akkor az alkalmazás próbálkozzon újra csatlakozni. Amennyiben ez mégsem sikerül, logolja a hibát. Annak érdekében, hogy ez a működés a későbbiekben is könnyen használható legyen, írtam egy `startReconnecting` függvényt is.

Logolás

Ezt követően a logolással foglalkoztam. A `pino` libraryt adtam hozzá a projekthez.

Második Hét

2.1. Active Directory service, tesztelés előkészítése (09.11.)

A mai napon, első körben, két új függvényt írtam az AD servicehez. Ezek az `authenticateUser`, és a `getUserBysAMAccountName` függvények. Az `authenticateUser` függvény egy DN-t, és egy jelszót vár paraméterként, inicializál egy új LDAP klienst, amellyel egy bind operációval autentikálja a felhasználót az AD szerver felé.

A `getUserBysAMAccountName` függvény pedig egy felhasználónevet (`sAMAccountName`) vár paraméternek és egy search operációval megpróbálja megkeresni az AD szerveren a felhasználót. A válaszként kapott felhasználó parseolásához a typebox library-t használom (lásd a 2. kódot).

```
// lib/types/user.ts
import { Type } from "@sinclair/typebox";
import type { Static } from "@sinclair/typebox";

export const activeDirectoryUserSchema = Type.Object({
        employeeID: Type.String(),
        distinguishedName: Type.String(),
        sAMAccountName: Type.String(),
        employeeNumber: Type.String(),
        memberOf: Type.Union([Type.String(),
        ↪  Type.Array(Type.String())], {
                default: "",
        }),
        sn: Type.String(),
        givenName: Type.String(),
        displayName: Type.String(),
        pwdLastSet: Type.String(),
});

export type ActiveDirectoryUser = Static<typeof
↪  activeDirectoryUserSchema>;
export const activeDirectoryUserValidator:
↪  Validator<ActiveDirectoryUser> =
        createValidator<ActiveDirectoryUser>(activeDirectoryUserSchema
        ↪  );
```

2. kód. user.ts fájl - typebox library használata

Mivel a projekt követelményei közé tartozott az is, hogy az alkalmazás az AD-ben szereplő jelszavak lejárati dátumait vegye figyelembe a session kezelésnél, így a pwdLastSet attribútumot is át kellett konvertáljam dátum objektummá, melyet a 3. kódban látható függvénnyel oldottam meg az [1] segítségével.

```ts
// lib/utils/activeDirectoryUtils.ts
export function parseActiveDirectoryDate(activeDirectoryDate: string):
↪  Date {
        const truncatedPwdLastSet = activeDirectoryDate.substring(
                0,
                activeDirectoryDate.length - 4,
        );

        const windowsEpoch = Date.UTC(1601, 0, 1);
        return new Date(windowsEpoch +
        ↪  Number.parseInt(truncatedPwdLastSet));
}
```

3. kód. parseActiveDirectoryDate függvény

Tesztelés előkészítése

A tesztelés nem szerepelt a projektkövetelmények között, valamint idő szűkében nem is tudnék 100%- os kódlefedettséget elérni. Ennek ellenére az alkalmazás fontosabb részeihez — amennyiben időm engedi -– írok unit teszteket. (Integrációs, illetve e2e teszteket még nem biztos, hogy fogok írni, de ha szükségét érzem, és időm engedi, akkor azokat is fogok.)

Az AD service is egy ilyen fontosabb, illetve kérdéses része az alkalmazásnak. Ahhoz, hogy AD service unit teszteket tudjak írni szükséges egy "mock" LDAP klienst létrehozni. Ehhez írtam először egy `AbstractLdapClient` osztályt, amit majd extendelni fog az `LdapClient` és a `MockLdapClient` osztály. A `MockLdapClient` osztályra már nem volt időm a mai nap.

2.2. MockLdapClient, ActiveDirectoryService osztállyá alakítása, tesztek írása (09.12.)

MockLdapClient

A mai napot a `MockLdapClient` osztály implementálásával kezdtem. Egyelőre csak két függvényt — `search` és `bind` — kellett megírnom, melyek a `tests/mocks/data/` mappában találhatók alapján futnak le, illetve dobnak hibákat.

ActiveDirectoryService osztállyá alakítása

Annak érdekében, hogy az AD service-t megfelelően lehessen tesztelni, ("singleton") osztállyá alakítottam át azt (lásd a 4. kódot), így minden tesztet külön példánnyal lehet végrehajtani.

Tesztek írása

Ezt követően a következő AD service unit teszteket írtam meg. Outputjuk az 5. kódban látható.

```
// lib/services/ActiveDirectoryService.ts
class ActiveDirectoryService {
    protected static _instance: ActiveDirectoryService | undefined =
    ↪  undefined;
    static initialize(
        ldapClient: AbstractLdapClient,
        activeDirectoryConfig: ActiveDirectoryConfig,
    ) {
        if (!ActiveDirectoryService._instance) {
            ActiveDirectoryService._instance = new
            ↪  ActiveDirectoryService(
                ldapClient,
                activeDirectoryConfig,
            );
        }
        return ActiveDirectoryService._instance;
    }

    static get instance(): ActiveDirectoryService {
        if (!ActiveDirectoryService._instance) {
            throw new Error(...);
        }
        return ActiveDirectoryService._instance;
    }

    private ldapClient: AbstractLdapClient;

    ...

    protected constructor(
        ldapClient: AbstractLdapClient,
        activeDirectoryConfig: ActiveDirectoryConfig,
    ) {
        this.ldapClient = ldapClient;
        this.activeDirectoryConfig = activeDirectoryConfig;
    }

    async start() { ... }
    async authenticateUser(bindDN: string, password: string):
    ↪  Promise<boolean> { ... }
    async getUserBysAMAccountName(sAMAccountName: string):
    ↪  Promise<User> { ... }
    ...
}
```

4. kód. activeDirectoryService - singleton osztállyá alakítás

2.3. Header komponens, Login oldal (frontend) (09.13.)

A Header komponenst implementáltam először (4. ábra). Sikerült teljesen reszponzívra megírni (5. ábra). Tartalmaz két — egyelőre placeholder — menüsort, melyekre kattintva egy-egy *dropdown* jelenik meg, a jobb sarokban egy *pro-*

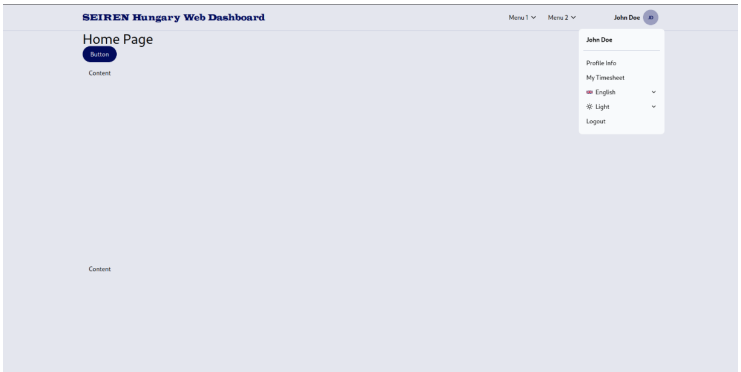
```
// pnpm run test:unit
src/lib/services/ActiveDirectoryService.test.ts (20)
    ActiveDirectoryService Singleton Tests (2)
        should initialize and return the same instance
        should reset the singleton between tests
    start() (5)
        should succeed with correct admin dn and password
        should succeed if service can reconnect after a connection
        ↪  error
        should throw LdapConnectionError if a connection error occures
        should throw UnknownLdapError if an unknown error occures
        should throw LdapAuthenticationError with incorrect admin
        ↪  credentials
    getUserBysAMAccountName() (9)
        should succeed with correct sAMAccountName
        should succeed if user is in multiple groups
        should return user with isAdmin set to true, if user is a
        ↪  member of the
            Admins group
        should throw UserNotFoundError if the user does not exist
        should throw MultipleUsersFoundError when multiple users are
        ↪  found
        should throw an InternalError if one or more active directory
        ↪  attributes are
            missing/incorrect
        should throw an InternalError if the ActiveDirectoryUser cannot
        ↪  be converted
            to a User
        should throw an InternalError if an unknown error occures
        should throw an InternalError if a connection error occures
    authenticateUser (4)
        should return true with correct credentials
        should return false with incorrect credentials
        should throw an InternalError if a connection error occures
        should throw an InternalError if an unknown error occures
```

5. kód. ActiveDirectoryService unit tesztek

fil gombot, melyre kattintva szintén egy dropdown jelenik meg, ahol többek között ki lehet választani a nyelvet (magyar, angol), illetve változtatni lehet a témán (sötét/világos). Ezen kívül, a Header bal sarkában egy *hamburger*, vagyis menü ikon jelenik meg mobileszközökön. Erre kattintva egy oldalsó menüsor (*drawer*) jelenik meg a placeholder menüpontokkal.

Login oldal (frontend)

Ezt követően a bejelentkezési oldal frontend részét csináltam meg (6. ábra). A bejelentkezés gomb megnyomásával a böngésző egy `POST` kérést küld a szervernek. Amennyiben sikeres a bejelentkezés, a szerver átirányít a kezdőoldalra, ahol



4. ábra. Header komponens, asztali méret, világos mód

egy felugró, *toast* üzenetet jelenít meg az alkalmazás. A toast üzenethez tartozó adatot (szöveg, háttérszín) nem szerettem volna az URL-en keresztül átadni, ezért azt egy cookie-ban tárolom, melyet a kiolvasás után egyből törlök is.

2.4. Login backend, session validáció (09.14.)

A bejelentkezés backend részét implementáltam (7. ábra), valamint — a minden kérés előtt lefutó (`hooks.server.ts`) — session validációt, amely a böngészőtől cookie header formájában kapott session token-t ellenőrzi.

A 6. kódban látható függvényeket implementáltam a `lib/auth/session.ts` modulban.

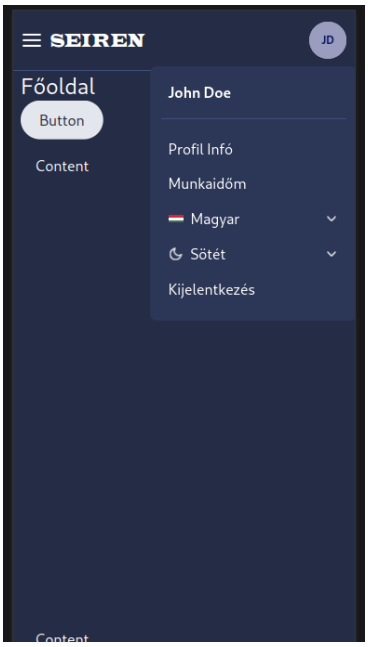
A `createSession()` függvény egy *Postgres* adatbázis `sessions` táblájába tárolja el a létrehozott sessionöket. A létrehozáskor a session lejárati dátumát – a követelményeknek megfelelően – az adott felhasználó `PwdLastSet` attribútuma, valamint a környezeti változóként definiált `passwordLastChangeIntervalInMinutes` határozza meg (`PwdLastSet + passwordLastChangeIntervalInMinutes`). A `validateSessionToken()` függvény a lejárati dátum mellett azt is ellenőrzi, hogy a felhasználó a session létrehozása óta megváltoztatta-e a jelszavát

2.5. Profil infó oldal, EntraPass adatok(09.15.)

A mai napot a profil infó oldal (8. ábra) implementálásával kezdtem. Itt tekinthetik meg a felhasználók az adataikat.

EntraPass adatok

Ezt követően az EntraPass Special Edition szoftverrel kezdtem el ismerkedni, amelyet a cég a beléptetési adatok rögzítéséhez használ. Mint kiderült, ez a szoftver (illetve ez a kiadás) egy beágyazott adatbázissal dolgozik, melyet "kívülről" nem lehet elérni. Ez azt jelenti, hogy az alkalmazás nem fog tudni kéréseket küldeni az EntraPass szervernek. Az egyetlen mód adatok kinyerésére a szoftverből a riport generálás, mely egy adott időintervallumban keletkezett



(a) Mobil méret, sötét mód, magyar nyelv

(b) Oldalsó menüsor

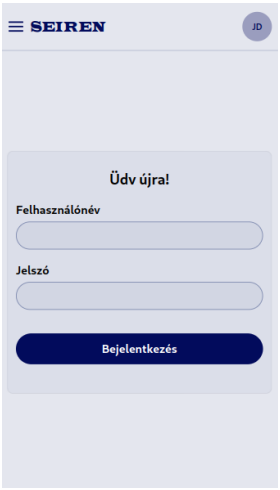
5. ábra. Header komponens, mobil méret

beléptetési adatokat exportálja .csv formátumba. A riport generálást automatizálni lehet. Például be lehet állítani, hogy a hét minden napján, 23:59-kor generáljon egy riportot az adott nap adataiból.

A megoldás tehát, az lesz, hogy az EntraPass szerver automatikusan generálja a riportokat, melyeket valamilyen fájlszinkronizációs programmal eljuttat az alkalmazást futtató szerverre. Az alkalmazás figyelni fogja az adott könyvtárakat, majd a hozzáadott .csv fájlokat feldolgozza és a saját adatbázisába menti.

CSV adatok, adatbázisba mentett adatok

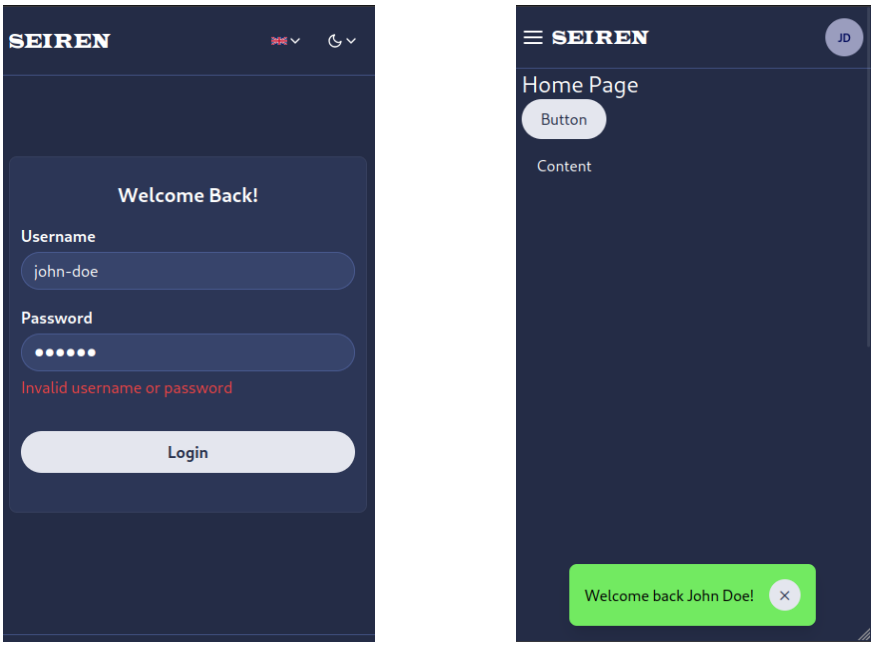
Az EntraPass által generált riportok formája a 9. ábrán látható. Ebből a formátumból kell majd a 10. ábrán megjelölt séma szerint menteni az adatokat az adatbázisba. Idő közben a követelmények bővültek azzal, hogy egy esetleges vészhelyzet esetén az alkalmazással az éppen gyárban tartózkodó személyeket is meg lehessen jeleníteni. Ehhez két további táblát is létre kell majd hozzak. Megfigyelhető, hogy az `emergency_card_usages` tábla nem tartalmaz HR azonosítót, helyette csak a kártyaszámot. Ez több szempontból is indokolt. Egyrészt, HR azonosítója csak a cég által alkalmazott személyeknek van, egy vészhelyzet esetén pedig minden személy helyzetét tudni kell. Másrészt, az AD szerverrel való kommunikáció csak egy plusz hibaforrást jelentene, amely vészhelyzet esetén nem kívánatos.



6. ábra. Login oldal, világos mód

```ts
// lib/auth/session.ts
export function generateSessionToken(): string {}

export async function createSession(
	token: string,
	userEmployeeId: string,
	userPasswordLastSet: Date,
	passwordChangeIntervalInMinutes: number,
): Promise<Session> {}
export async function validateSessionToken(
	token: string,
): Promise<SessionValidationResult> {}

export async function invalidateSession(sessionId: string):
↪  Promise<void> {}

export type SessionValidationResult =
	| { session: Session; user: User }
	| { session: null; user: null };

export function setSessionTokenCookie(
	event: RequestEvent,
	token: string,
	expiresAt: Date,
): void {}

export function deleteSessionTokenCookie(event: RequestEvent): void {}
```

6. kód. session.ts fájl függvények



(a) Hibás jelszó (b) Sikeres bejelentkezés

7. ábra. Bejelentkezés

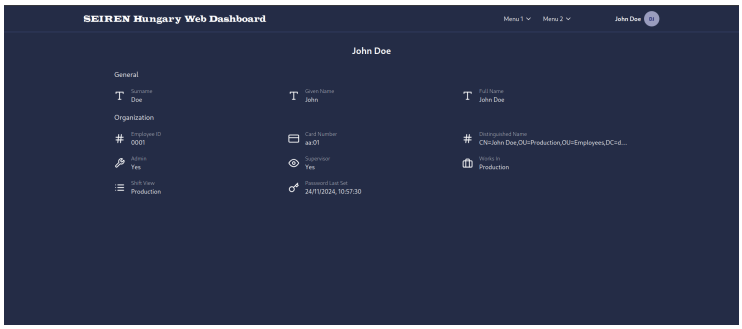
Harmadik Hét

3.1. EntraPass adatok feldolgozása (09.18.)

A tegnapi napon megtervezettek szerint, ma az EntraPass adatok alkalmazás oldali feldolgozását implementáltam. Először megírtam a `reportFileParserService`-t, amely a chokidar könyvtárat felhasználva figyeli a – környezeti változóként megadott — *daily reports* és *emergency reports* mappákat. Amint egy új CSV fájl-t a mappák valamelyikébe másolunk, a service a `csv-parser` könyvtár segítségével parseolja azt, majd az átalakított (`ParsedDailyReportRow`, illetve `ParsedEmergencyReportRow` típusok) adatokkal meghívja a `cardUsageService` `insertParsedDailyReportRows` vagy `insertParsedEmergencyReportRows` metódusát. Ezek a metódusok szintén végeznek némi átalakítást az adatokon majd az adatbázis megfelelő tábláiba tárolják azokat.

3.2. Kártyahasználati adatok átalakítása a munkaidő kimutatás oldal betöltésekor (1) (09.19.)

Egyeztetések alapján, a munkaidő kimutatás (timesheet) oldalnak két nézete lehet az alapján, hogy az alkalmazott milyen beosztásban dolgozik: termelési, illetve irodai nézet. A két nézethez különböző adatokra van szükség. A timesheet oldal betöltésekor az alkalmazás két dátum közötti `cardUsage` rekordokat lekérdezi, majd átalakítja a 7. kódban található két típus valamelyikébe.



8. ábra. /account/profile oldal

Sequence	Date and Time	Event message	Event number	Object #1	Description #1	Object #2	Description #2	Object #3	Description #3	Object #4	Description #4	Card number
75	2023.09.01. 7:43:41	Access granted	203	12	GUARD HOUSE TURNSTILE 1	5	John Doe	0		0		aa:01
215	2023.09.01. 8:00:00	Access granted	203	12	PRODUCTION AREA TURNSTILE 1	5	John Doe	0		0		aa:01
227	2023.09.01. 10:39:16	Access granted	203	12	PRODUCTION AREA TURNSTILE 2 - Exit	5	John Doe	0		0		aa:01
344	2023.09.01. 12:46:32	Access granted	203	12	PRODUCTION AREA TURNSTILE 1	5	John Doe	0		0		aa:01
345	2023.09.01. 16:00:00	Access granted	203	12	PRODUCTION AREA TURNSTILE 2 - Exit	5	John Doe	0		0		aa:01
566	2023.09.01. 16:27:31	Access granted	203	12	GUARD HOUSE TURNSTILE 1 - Exit	5	John Doe	0		0		aa:01
75	2023.09.02. 7:55:41	Access granted	203	12	GUARD HOUSE TURNSTILE 1	5	John Doe	0		0		aa:01

9. ábra. Riport csv

A tervezés után el is elkezdtem implementálni ezt a működést. Az átalakítást két részre bontva, a rekordokat először `Production/OfficeShiftVewDataIntervalDates` típusba alakítom (lásd a 8. kódot). Ezekben a típusokban csak a kezdő és záró dátumok találhatók. A későbbi, számított attribútumokat két másik függvényben számolom ki.

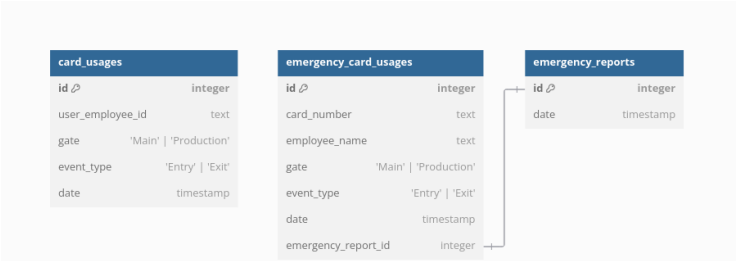
3.3. Kártyahasználati adatok átalakítása a munkaidő kimutatás oldal betöltésekor (2) (09.20.)

Először befejeztem a rekordok `Production/OfficeShiftVewDataIntervalDates` típusba alakítását. Ezt követően a `getProductionShiftViewData`, illetve a `getOfficeShiftViewData` függvényeket írtam meg, melyek a számított adatokkal egészítik ki a `Production/OfficeShiftVewDataIntervalDates` típusú objektumokat `Production/OfficeShiftVewData` típusú objektumokká.

3.4. Timesheet oldal - dátum intervallum beviteli mező, adatok kérése (09.21.)

A timesheet oldallal kezdtem el foglalkozni. Első körben egy dátum intervallum beviteli mezőt szerettem volna implementálni, hogy a munkaidő adatokat a felhasználók dátum szerint tudják szűrni. Mivel, a Skeleton UI Toolkit nem tartalmaz ilyen komponenst, igy a vanilla-calendar-pro könyvtárra esett a választásom. Egyetlen hátulütője az volt, hogy a naptár színeit csak úgy tudtam testreszabni, hogy a könyvtárhoz tartozó CSS fájlokat felülírtam a saját CSS fájljaimmal. A végeredmény a 11. ábrán látható.

Ezt követően létrehoztam egy `/api/shift-view-data/search` API útvonalat, valamint az ennek megfelelő `apiClient.fetchShiftViewData()` függvényt, mely utóbbit a böngésző a „keresés” gombra kattintva meghív (12. ábra). E mellett

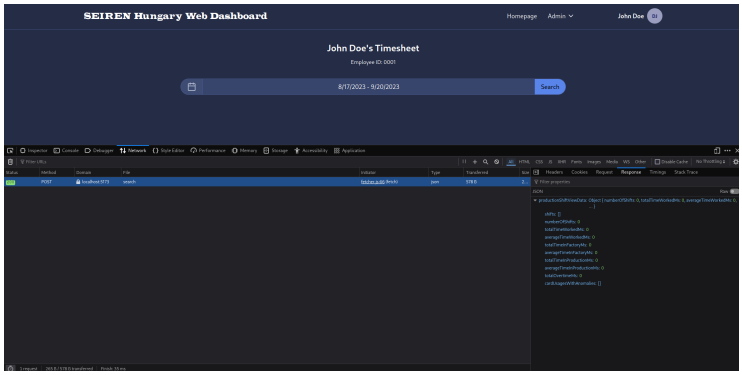


10. ábra. Kártyahasználati adatok, adatbázis séma

(a) Sötét mód (b) Világos mód

11. ábra. Dátum intervallum beviteli mező

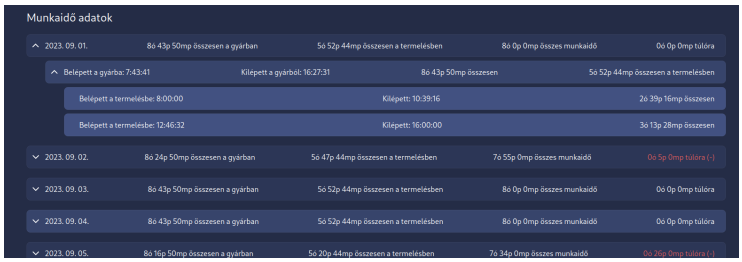
az oldal PageLoad függvényét is megírtam, hogy az első betöltéskor – *SSR*-et alkalmazva – az oldal már tartalmazza az aktuális havi adatokat.



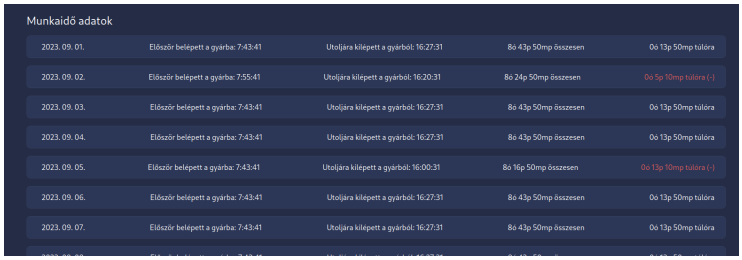
12. ábra. /api/shift-view-data/search API útvonal

3.5. PaginatedShiftTreeView komponens (09.22.)

A timesheet oldalra egy olyan *lapozható* komponenst implementáltam, amellyel a felhasználó a keresett intervallumban szereplő munkaidő adatokat (`Office|Production|ShiftViewData.shifts`) tudja megtekinteni napokra bontva. Termelés nézetben (13a. ábra) a komponens lenyitható füleket tartalmaz, míg irodai nézet esetében (13b. ábra) a komponens csak egyszintes.



(a) Termelés nézet



(b) Irodai nézet

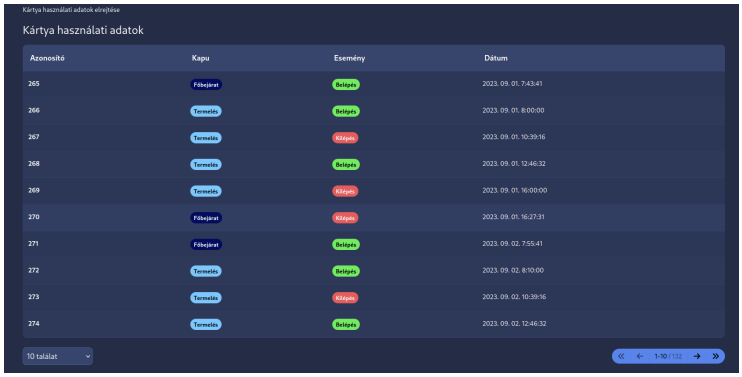
13. ábra. PaginatedShiftTreeView komponens

Negyedik Hét

4.1. Kártyahasználati adatok táblázat (09.25.)

Egy `DataNotFound` komponens implementálásával kezdtem, amely akkor jelenik meg a timesheet oldalon, ha az adott intervallumban nem található adat.

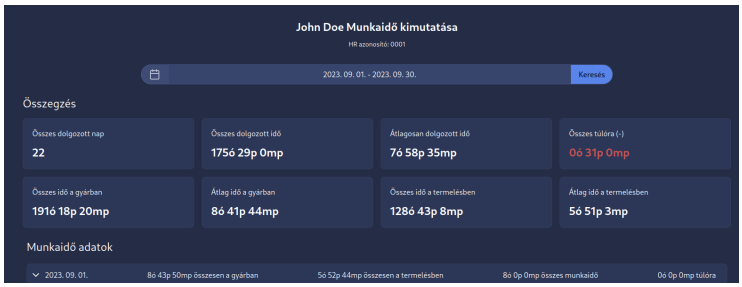
Ez után, egy olyan lapozható táblázatot adtam hozzá az oldalhoz, melyben a kártyahasználati adatokat lehet megtekinteni (14. ábra).



14. ábra. Kártyahasználati adatok táblázat

4.2. StatCards komponens, hiba javítása (09.26.)

A timesheet oldalra írtam egy olyan komponenst, amelyben ”kártyákban” szerepelnek a lekért munkaidő adatokhoz (`Office|ProductionShiftViewData`) tartozó statisztikák (15. ábra). Például: dolgozott napok száma, összes túlóra, összes dolgozott idő, stb.



15. ábra. StatCards komponens

Ezt követően egy hibát javítottam ki, mely a `cardUsageUtils.getOffic`⌋`e|ProductionShiftViewData` függvényekben lépett/léptek fel, valamint írtam néhány tesztet is hozzájuk.

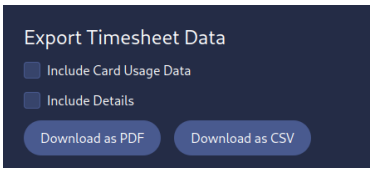
4.3. MultilineChart komponens (09.27.)

A timesheet oldalra implementáltam a layercake library segítségével egy több vonalt tartalmazó diagramot. Ezen a diagramon (16. ábra) napokra bontva szerepel az összes dolgozott idő, a túlóra, valamint – termelési nézet esetén – az összes gyárban, és termelésben töltött idő.

16. ábra. MultiLineChart komponens

4.4. PDF-be exportálás (09.28.)

A munkaidő adatok PDF-ként való exportálását oldottam meg a jspdf könyvtár segítségével. A felhasználó kiválaszthatja egy checkbox segítségével (17. ábra), hogy az exportált pdf (18. ábra) tartalmazza e a kártyahasználati adatokat, valamint termelési nézetben a lenyíló fülek tartalmát.



17. ábra. Export komponens

TIMESHEET REPORT

SEIREN

Name: John Doe
Works in: Production
Report generated by: John Doe (0001)
Total days worked: 22
Total time in factory: 191h 18m 20s
Total overtime: -0h 31m 0s

Employee ID: 0001
Report generated: 12/30/2024, 4:21:52 PM
Report period: 9/1/2023 - 9/30/2023
Total time worked: 175h 29m 0s
Total time in production: 128h 43m 8s

Timesheet data

Date	Time in factory	Time in production	Time worked	Overtime/Undertime
9/1/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/2/2023	8h 24m 50s	5h 47m 44s	7h 55m 0s	-0h 5m 0s
9/3/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/4/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/5/2023	8h 16m 50s	5h 20m 44s	7h 34m 0s	-0h 26m 0s
9/6/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/7/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/8/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/9/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/10/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/11/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/12/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/13/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/14/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/15/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/16/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/17/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/18/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/19/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/20/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/22/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s
9/25/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s

18. ábra. Exportált PDF

4.5. XLSX-be exportálás (09.29.)

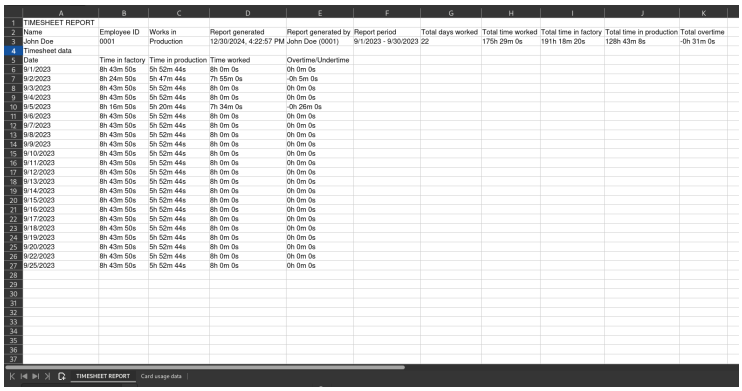
A tegnapi nap mintájára, ma a munkaidő adatok XLSX-ként való exportálását implementáltam az exceljs könyvtár segítségével (19. ábra).

Ötödik Hét

5.1. /admin/employees oldal (backend) (10.16.)

Az adminisztrátorok számára elérhető funkciók implementálását kezdtem meg. Az első funkció a dolgozók keresése és a találatok táblázatos megjelenítése. Elkészítettem a `searchUsers()` függvényt az Active Directory service-hez, amely név, felhasználónév (`sAMAccountName`), HR ID és kártyaszám alapján teszi lehetővé a keresést az Active Directory rekordjai között. Mivel az Active Directory nem támogatja a lapozást (pagination), ezt a kliensoldalon fogom megvalósítani.

Ezt követően létrehoztam a `/api/users/search` végpontot, valamint implementáltam az ehhez tartozó `apiClient.searchUsers()` függvényt.



	A	B	C	D	E	F	G	H	I	J	K
1	TIMESHEET REPORT										
2	Name	Employee ID	Works in	Report generated	Report generated by	Report period	Total days worked	Total time worked	Total time in factory	Total time in production	Total overtime
3	John Doe	0001	Production	12/30/2024, 4:22:57 PM	John Doe (0001)	9/1/2023 - 9/30/2023	22	175h 29m 0s	191h 18m 20s	128h 43m 8s	-0h 31m 0s
4	Timesheet data										
5	Date	Time in factory	Time in production	Time worked	Overtime/Undertime						
6	9/1/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
7	9/2/2023	8h 24m 50s	5h 47m 44s	7h 55m 0s	-0h 5m 0s						
8	9/3/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
9	9/4/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
10	9/5/2023	8h 16m 50s	5h 20m 44s	7h 34m 0s	-0h 26m 0s						
11	9/6/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
12	9/7/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
13	9/8/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
14	9/9/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
15	9/10/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
16	9/11/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
17	9/12/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
18	9/13/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
19	9/14/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
20	9/15/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
21	9/16/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
22	9/17/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
23	9/18/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
24	9/19/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
25	9/20/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
26	9/22/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
27	9/25/2023	8h 43m 50s	5h 52m 44s	8h 0m 0s	0h 0m 0s						
28											
29											
30											
31											
32											
33											
34											
35											
36											
37											

TIMESHEET REPORT | Card usage data

19. ábra. Exportált XLSX

5.2. /admin/employees oldal (10.17.)

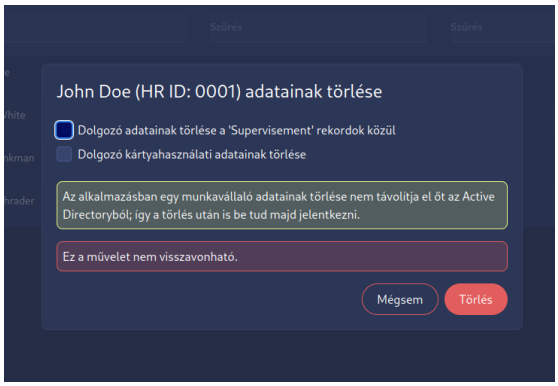
Elkészítettem a dolgozók táblázatát (20. ábra) tartalmazó `/admin/employees` oldalt. A táblázat fejlécében a felhasználó módosíthatja a rendezési beállításokat, és az adott mezőhöz tartozó szűrőt is. Az utolsó oszlopban három gomb található: a bal oldali a dolgozó munkaidő-adatait megjelenítő oldalra, a középső a dolgozó profiloldalára navigál, míg a jobb szélső gomb a dolgozó adatainak törlésére (21. ábra) szolgál. Ez a törlés nem érinti az Active Directory-ban tárolt felhasználói adatokat, csak a dashboard alkalmazás adatbázisából távolítja el a kapcsolódó adatokat.

20. ábra. /admin/employees táblázat

5.3. Kártyahasználati adatok keresése (backend) (10.18.)

A következő adminisztrátori funkció, a kártyahasználati adatok menedzselése került sorra. Első lépésként implementáltam a cardUsage service `getPaginat⌋ edCardUsages()` függvényét, a 9. kódban látható paraméterekkel.

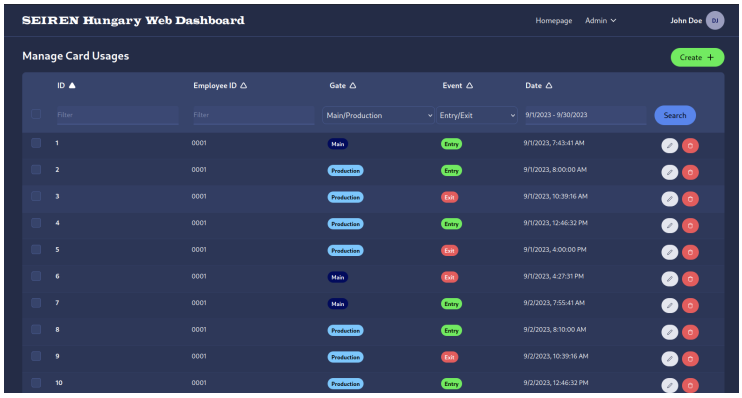
Ezt követően elkészítettem a `/api/admin/card-usages/search` végpontot és a hozzá tartozó `apiClient.searchPaginatedCardUsages` függvényt.



21. ábra. Dolgozó adatainak törlése

5.4. /admin/card-usages oldal (10.19.)

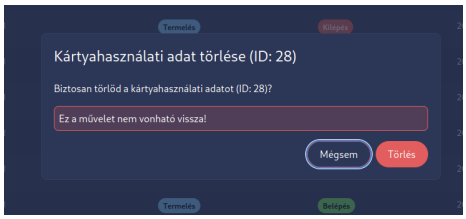
Elkezdtem dolgozni a kártyahasználati adatokat megjelenítő `/admin/card-usages` oldalon. Létrehoztam a kártyahasználati adatokat tartalmazó táblázatot (22. ábra), amely hasonló a dolgozók táblázatához, de az adatok itt szerveroldali pagination-el érhetők el, mivel az alklalmazás az adatokat a saját adatbázisából kéri le. A táblázatban a felhasználó egyesével törölhet, módosíthat, új rekordokat hozhat létre, illetve egyszerre több kijelölt rekordot is törölhet.



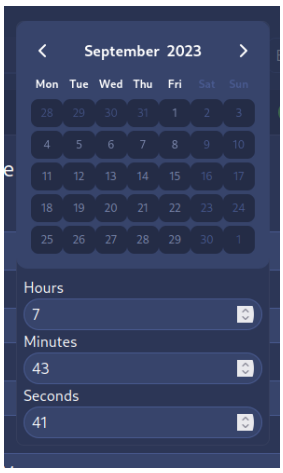
22. ábra. /admin/card-usages oldal

A nap végére sikerült befejezni a törlés funkciót (23. ábra).

A szerkesztési funkcióhoz tartozó modal (modális ablak) implementálása során problémába ütköztem: az általam használt *Vanilla Calendar Pro* könyvtár időbeviteli mezője csak órák és percek megadását támogatja. Ezért létrehoztam egy saját `TimeInput` komponenst (24. ábra), amely lehetővé teszi a másodperc értékének megadását is.



23. ábra. Kártyahasználati adat törlése



24. ábra. Dátum beviteli mező (TimeInput komponenssel)

5.5. /admin/card-usages oldal befejezése (10.20.)

A hátralévő funkciókat, vagyis a kártyahasználati adatok szerkesztését, új rekordok létrehozását, valamint a kijelölt rekordok törlését implementáltam frontend, és backend oldalon. (25. ábra)

Hatodik Hét

6.1. /admin/supervisement-groups oldal (BE) (10.23.)

A következő adminisztrátorok számára elérhető funkción kezdtem el dolgozni, amely lehetővé teszi a Supervisement (Felettesi) csoportok létrehozását és módosítását. Az adminisztrátor képes lesz csoportokat létrehozni, például HR, IT, stb. részlegek számára, majd dolgozókat rendelhet hozzájuk beosztottként vagy feletteként. A csoportokhoz rendelt dolgozók eltávolíthatók, a csoportok átnevezhetők, illetve törölhetők lesznek.

Először létrehoztam a `SupervisementGroups` adatbázis sémát (26. ábra), majd megírtam a supervisement service következő függvényeit: `createGroup`, `getSupervisementGroupById`, `deleteGroup`, `renameGroup`, `getPaginatedSu⌋ pervisementGroups`. Ezt követően az ezekhez a függvényekhez tartozó `/api` végpontokat kezdtem el írni.

6.2. /admin/supervisement-groups oldal (10.24.)

Befejeztem a tegnap megkezdett `/api` végpontok megírását, majd az azoknak megfelelő `apiClient` függvényeket írtam meg. Végül az `/admin/supervisement-groups` oldalt kezdtem el implementálni (27. ábra), amely egy táblázatot tartalmaz a létrehozott supervisement csoportokról.

6.3. /admin/supervisement-groups/[id] oldal (BE) (10.25.)

Befejeztem a tegnap elkezdett `/admin/supervisement-groups` oldal implementálását.

Ezt követően létrehoztam a `SupervisementGroupMemberships` adatbázis sémat (28. ábra), amely a dolgozók csoportokhoz rendelését valósítja meg.

Utána a supervisement service-t egészítettem ki a következő függvényekkel: `addEmployeeToSupervisementGroup`, `removeEmployeeFromSupervisementGroup`, `editEmployeeIsSupervisor`, `getEmployeesByGroupMembership`.

A fenti függvényekhez megírtam az `/api` végpontokat, valamint elkezdtem írni a végpontokhoz tartozó `apiClient` függvényeket.

6.4. /admin/supervisement-groups/[id] oldal (10.26.)

Befejeztem a tegnap elkezdett `apiClient` függvények implementálását, majd az `/admin/supervisement-groups/[id]` oldalt kezdtem el implementálni, amely két táblázatot tartalmaz (29. ábra). A bal oldali táblázat tartalmazza azokat a dolgozókat, akik még nincsenek hozzáadva a csoporthoz. Az utolsó oszlopban található két gomb segítségével lehet hozzáadni őket (feletteként vagy beosztottként). A jobb oldali táblázat pedig a csoporthoz már hozzáadott dolgozókat fogja tartalmazni. tt az utolsó oszlopban lehetőség lesz a dolgozó eltávolítására a csoportból, valamint az "előléptetésre", illetve a "lefokozására".

6.5. /admin/supervisement-groups/[id] oldal (2) (10.27.)

Befejeztem az /admin/supervisement-groups/[id] oldal implementálását (30. ábra).

Hetedik Hét

7.1. isSupervisor függvények, dolgozó adatainak törlése funkció (10.30.)

A supervisement service-t kiegészítettem az `isSupervisor`, `isSupervisorOfGroup` és `isSupervisorOfEmployee` függvényekkel, majd ezeket a függvényeket alkalmaztam, illetve meghívtam minden szükséges helyen.

Ezt követően az /admin/employees oldalon elérhető "Dolgozó adatainak törlése" funkciót fejeztem be, melyhez meg kellett írnom a `supervisementService.deleteEmployeeSupervisementRecords()`, a `cardUsageService.deleteEmployeeCardUsageRecords()` függvényeket, valamint a szükséges `/api` végpontokat.

7.2. /supervisor/groups és /supervisor/groups/[id] oldalak (10.31.)

A felettesek (supervisor) számára elérhető funkció implementálásán kezdtem el dolgozni. Ez a funkció lehetővé teszi, hogy a dolgozó megtekintse a saját maga által felügyelt csoportokat.

Első körben a supervisement service-t egészítettem ki a `getPaginatedSup⌋ ervisedGroupsOfUser()` és a `getSupervisedUsersOfGroup()` függvényekkel, majd az ezeknek megfelelő `/api` végpontokat és az `apiClient` függvényeket írtam meg.

Ez után a `/supervisor/groups`, valamint a `/supervisor/groups/[id]` oldalakat implementáltam (31. ábra). A `/supervisor/groups` oldalon a felettes megtekintheti a saját felügyelt csoportjait. A jobb oldali oszlopban linkek mutatnak az egyes csoportok részletes oldalaira (`/supervisor/groups/[id]`). A részletes oldalon egy táblázat jelenik meg, amely az adott csoport beosztott dolgozóit listázza. Itt a jobb szélső oszlopban a linkek az adott dolgozó munkaidő adatok, valamint profil infó oldalára mutatnak.

7.3. Főoldal és hiba oldalak (11.01.)

A mai napon a főoldalt, valamint a hiba oldalakat implementáltam (Ábra 32. és 33.).

7.4. Dockerfile és docker-stack.yml (11.02.)

Az alkalmazás telepítéséhez Docker-t szerettem volna használni, ezért először elkészítettem a Dockerfile-t.

Első körben docker compose-t szerettem volna használni az alkalmazás, valamint az adatbázis elindításához, azonban a docker compose-t eredetileg egy fejlesztői eszköznek szánták. Így esett a választásom a docker stack-re, ami több - a későbbiekben potenciálisan használható - funkciót is támogat (pl. secrets, rolling updates, rollbacks, orchestráció, stb.). A nap végére megírtam a docker-stack.yml fájlt, amely három service-t tartalmaz: a dashboard alkalmazást, postgres adatbázist, caddy reverse proxyt. Az egyik probléma, amibe ütköztem az az volt, hogy a docker stack nem támogatja a környezeti változók fájlból való beolvasását. A végleges parancs a docker stack elindítására végül ez lett:

```
// lib/types/shiftViewData.ts
export type ProductionShiftViewData = {
        shifts: {
                startDay: Date;
                endDay: Date;
                firstProductionEntry: Date;
                lastProductionExit: Date;
                totalWorkedTimeMs: number;
                totalInFactoryTimeMs: number;
                totalInProductionTimeMs: number;
                overtimeMs: number;
                inFactoryIntervals: {
                        startDate: Date;
                        endDate: Date;
                        timeInFactoryMs: number;
                        totalProductionTimeMs: number;
                        inProductionIntervals: {
                                startDate: Date;
                                endDate: Date;
                                timeInProductionMs: number;
                        }[];
                }[];
        }[];
        numberOfShifts: number;
        totalTimeWorkedMs: number;
        averageTimeWorkedMs: number;
        totalTimeInFactoryMs: number;
        averageTimeInFactoryMs: number;
        totalTimeInProductionMs: number;
        averageTimeInProductionMs: number;
        totalOvertimeMs: number;
        cardUsageAnomalies: {
                cardUsageIndex: number;
                error: string;
        }[];
};

export type OfficeShiftViewData = {
        shifts: {
                day: Date;
                firstFactoryEntry: Date;
                lastFactoryExit: Date;
                timeWorkedMs: number;
                overtimeMs: number;
        }[];
        numberOfShifts: number;
        totalTimeWorkedMs: number;
        averageTimeWorkedMs: number;
        mostTimeWorkedMs: number;
        leastTimeWorkedMs: number;
        totalOvertimeMs: number;
};
```

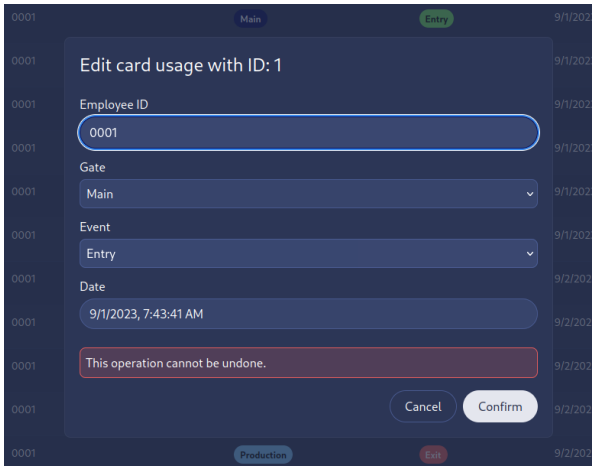
7. kód. Termelési és irodai nézetek sémái

```ts
// lib/types/shiftViewData.ts
export type ProductionShiftViewDataIntervalDates = {
        shifts: {
                startDay: Date;
                endDay: Date;
                inFactoryIntervals: {
                        startDate: Date;
                        endDate: Date;
                        inProductionIntervals: {
                                startDate: Date;
                                endDate: Date;
                        }[];
                }[];
        }[];
        cardUsageAnomalies: {
                cardUsageIndex: number;
                error: string;
        }[];
};

export type OfficeShiftViewDataIntervalDates = {
        shifts: {
                day: Date;
                firstFactoryEntry: Date;
                lastFactoryExit: Date;
        }[];
};
```

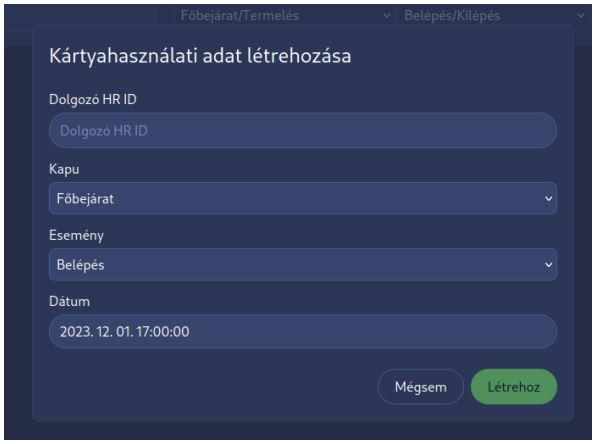
8. kód. Production/OfficeShiftVewDataIntervalDates típusok

```ts
export type CardUsageSearchParams = {
        pageNumber: number;
        pageSize: number;
        orderBy?: "id" | "userEmployeeId" | "gate" | "eventType" |
        ↪  "date";
        order?: "asc" | "desc";
        id?: string;
        userEmployeeId?: string;
        gate?: "Main" | "Production";
        eventType?: "Entry" | "Exit";
        startDate?: Date;
        endDate?: Date;
};
```

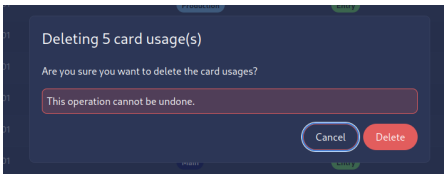
9. kód. getPaginatedCardUsages() paraméterei



(a) Kártyahasználati adat módosítása modal

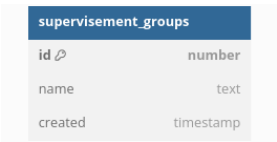


(b) Kártyahasználati adat létrehozása modal

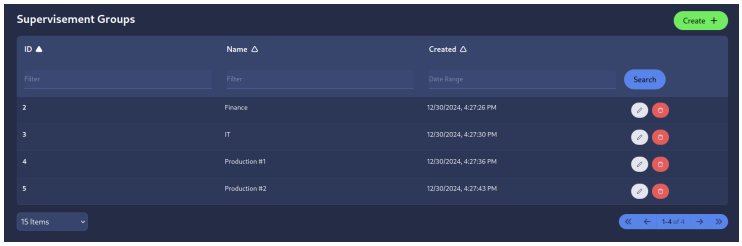


(c) Kiválasztott rekordok törlése modal

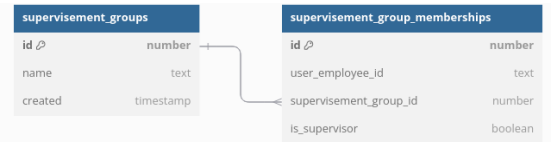
25. ábra. /admin/card-usages oldal funkciói



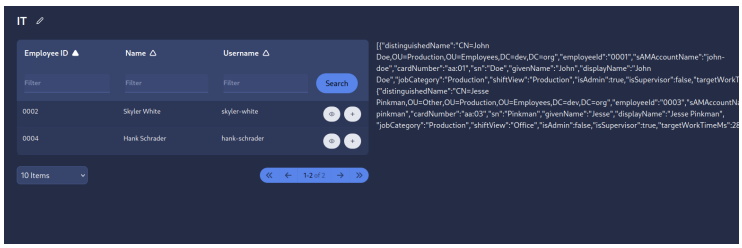
26. ábra. supervisement_groups adatbázis séma



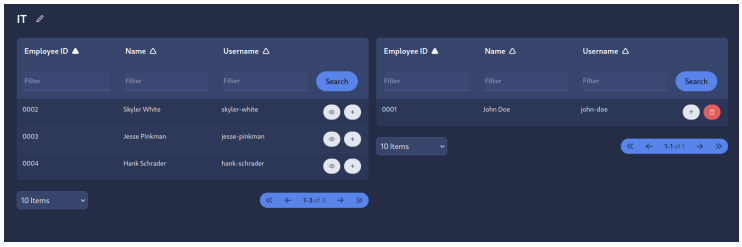
27. ábra. /admin/supervisement-groups oldal



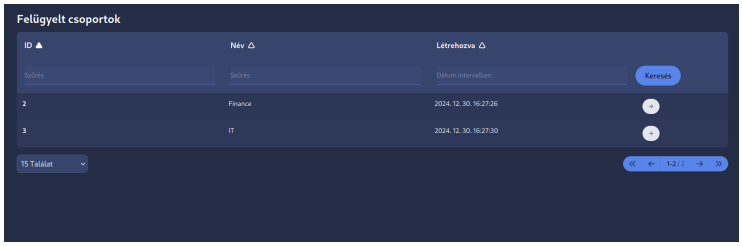
28. ábra. supervisement adatbázis séma



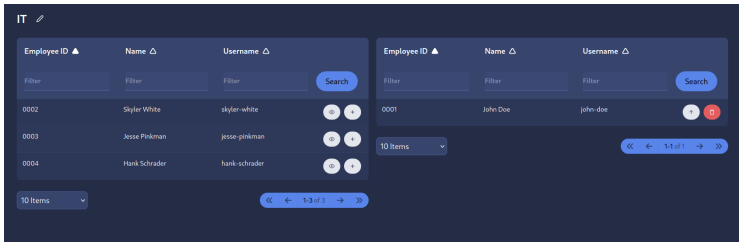
29. ábra. admin/supervisement-groups/[id] olda



30. ábra. admin/supervisement-groups/[id] olda



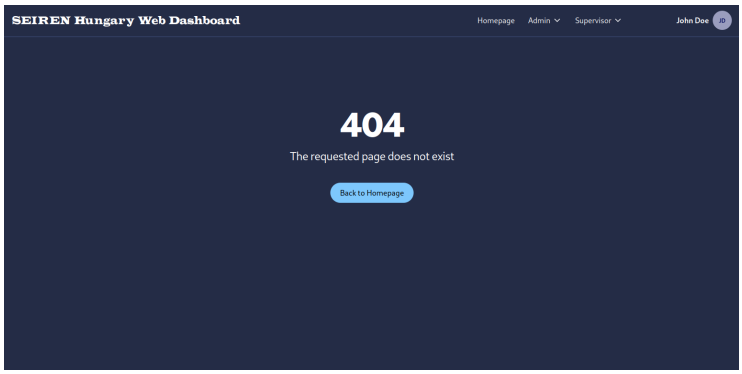
(a) /supervisor/groups oldal



(b) /supervisor/groups/[id] oldal

31. ábra. /supervisor/groups és /supervisor/groups[id] oldalak

32. ábra. Főoldal



33. ábra. hiba oldal

```
env $(cat /srv/dashboard/deploy/.env | grep ^[A-Z] | xargs) docker
↪  stack deploy -c
/srv/dashboard/deploy/docker-stack.yml seiren-dashboard
↪  --with-registry-auth
--detach=true
```

7.5. GitHub Workflow (11.03.)

Sajnos megfelelő mennyiségű teszt megírására nem volt elegendő időm. Ennek ellenére fontosnak tartom, hogy legyen egy CI/CD pipeline, ami automatikusan lefuttatja a teszteket, style check-et, lint- et, stb. Továbbá, az alkalmazás telepítésekor a docker image-et a docker engine egy privát Docker Hub repository-ból tölti le, így a pipeline-nak az image buildelését és feltöltését is meg kell oldania. Ezt a pipeline-t írtam meg a `.github/workflows/pipeline.yaml` fájlban.

Nyolcadik Hét

8.1. Szerver setup és dokumentáció (11.06.)

Ma a szerver telepítését folytattam. Többek között létrehoztam a usereket, a könyvtárakat, azoknak megfelelő jogosultságokat állítottam be, telepítettem a Docker-t, írtam egy szkriptet, ami letölti a repository-t, majd kicsomagolja a deploy mappát, létrehoztam a docker secreteket, stb. Ezt követően pedig elindítottam az alkalmazást, mely így már elérhető volt. Kérésre minden lépésemet dokumentáltam.

8.2. Riport fájlok szinkronizációja (11.07.)

A Kantech (Windows) szerver és az alkalmazás szervere közötti fájlszinkronizációt oldottam meg. Több lehetőség közül végül a unison nevű programra esett a választásom, amely ssh-n keresztül szinkronizálja a riport mappákat a két szerver között. A unison profile fájlok a 10. kódban láthatók. A használatához felhasznált forrást lásd a [4] dokumentumban.

Az a probléma lépett fel, hogy a unison ideiglenes (temp) fájlokkal dolgozik, így az alkalmazás azokat a fájlokat is megpróbálta beolvasni. Egy reguláris kifejezéssel ki tudtam szűrni ezeket a fájlokat.

Ezúttal is minden lépésemet dokumentáltam.

8.3. Két bug kijavítása (11.08.)

Két bug kijavításán kezdtem el dolgozni. Az első mindössze annyi volt, hogy a *Profil gomb*-ban a felhasználó nevének kezdőbetűi hard code-olva voltak, így ezt gyorsan kijavítottam.

A második, a munkaidők, vagyis a túlórák számolását érintő hiba volt. Ez egy komplexebb bug volt. Nem is tudtam a nap végére kijavítani.

```
# daily-report-sync.prf

root = C:\Users\Kantech-PC\Documents\EntraPass\Daily
root = ssh://reportuser@<host>//srv/reports/daily

sshargs = -i C:\\Users\\Kantech-PC\\.ssh\\id_ed25519
repeat = watch
auto = true
batch = true
confirmbigdel = false
confirmmerge = false

# emergency-report-sync.prf

root = C:\Users\Kantech-PC\Documents\EntraPass\Emergency
root = ssh://reportuser@<host>//srv/reports/emergency

sshargs = -i C:\\Users\\Kantech-PC\\.ssh\\id_ed25519
repeat = watch
auto = true
batch = true
confirmbigdel = false
confirmmerge = false
```

10. kód. unison fájlok

8.4. Bug kijavítása, refaktorálás (11.09.)

Először a tegnap megkezdett bug fixálását fejeztem be.

Ezt követően, mivel úgy tűnik, hogy az alkalmazás megfelelően működik, valamint új feature megírására már nincs időm, a kódot refaktoráltam néhány helyen, így javítva kicsit a kódminőségen.

8.5. Refaktorálás, issue-k írása (11.10.)

Az utolsó napon, a tegnap megkezdett kód refaktorálást fejeztem be. Ezt követően a projekt során felmerült ötleteket, kisebb bugokat GitHub issue-k formájában írtam le. Így, ha a későbbiekben más hallgatók érkeznek a céghez, akkor könnyen folytatni tudják a projektet.

Hivatkozások

[1] Microsoft közreműködők. Pwd-last-set attribute. https://learn.microsoft.com/en-us/windows/win32/adschema/a-pwdlastset, 2020. Hozzáférve: 2025. 05. 04.

[2] ldapjs fejlesztők. Ldap guide. https://web.archive.org/web/20240315064825/http://ldapjs.org/guide.html, 2024. Hozzáférve: 2025. 05. 04.

[3] Josh Madakor. How to setup a basic home lab running active directory (oracle virtualbox) — add users w/powershell. https://www.youtube.com/watch?v=MHsI8hJmggI, 2021. YouTube videó, hozzáférve: 2025. 05. 04.

[4] Benjamin C. Pierce. Unison file synchronizer user manual and reference gudie. https://raw.githubusercontent.com/bcpierce00/unison/documentation/unison-manual.pdf, 2025. Version 2.53.3.

[5] Wikipédia közreműködők. Lightweight directory access protocol. https://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol, 2024. Hozzáférve: 2025. 05. 04.