



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

ALGORITMUSOK ÉS ALKALMAZÁSAIK

TANSZÉK

Asztali naptár alkalmazás

Témavezető:

Kovácsné Dr. Pusztai Kinga

adjunktus, Ph.D

Szerző:

Sólyomvári Dávid

programtervező informatikus BSc

Budapest, 2023

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Figyelem! Nyári záróvizsga esetén a módosítás határideje február 1., őszi záróvizsga esetén augusztus 31.

Hallgató adatai:

Név: Sólyomvári Dávid

Neptun kód: U4XU7I

Képzési adatok:

Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)

Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: Kovácsné Dr. Pusztai Kinga Emese

munkahelyének neve, tanszéke: ELTE IK, Algoritmusok és Alkalmazásai Tanszék

munkahelyének címe: 1117, Budapest, Pázmány Péter sétány 1/C.

beosztás és iskolai végzettsége: egyetemi adjunktus, PhD

A szakdolgozat címe: Asztali naptár alkalmazás

A szakdolgozat témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

A program szerver-kliens alkalmazásként C++ nyelven, Qt GUI keretrendszerben fog elkészülni. A szerver több klienst is ki fog tudni szolgálni. Az alkalmazás egy API-n keresztül fog kommunikálni a adatbázissal. A megvalósításhoz a háromrétegű modell/nézet/perzisztencia architektúrát használnám, melyben elkülönül a megjelenítés, az üzleti logika, valamint az adatelérés.

Az alkalmazásban lehet adott napra, vagy akár több napra is eseményt megadni. Ha az eseményünk nem egész napos, beállíthatunk neki időpontot vagy időszávot is.

Az eseményekhez adhatunk címkéket (például családi, munka, egyéb) is. Be tudjuk állítani, hogy az adott esemény csak egyszer forduljon elő, vagy rendszerszerű előfordulás esetén ismétlődő eseményünk legyen. Az eseményeket szabadon tudjuk szerkeszteni és kitörölni. Emellett közös programok tervezéséhez egy közös megfelelő időpont kereső adna segítséget.

Az alkalmazás kinézetét szabadon személyre szabhatjuk a háttérszín, események színének stb. megváltoztatásával.

Az alkalmazásban az adatok betöltését és mentését egy adatbázis valósítaná meg.

Budapest, 2022. 12. 01.

SZAKDOLGOZAT / DIPLOMAMUNKA

EREDETISÉG NYILATKOZAT

Alulírott Sólyomvári Dávid Neptun-kód: U4XU7I

ezennel kijelentem és aláírással megerősítem, hogy az Eötvös Loránd Tudományegyetem
Informatikai Karának, Algoritmusok és Alkalmazásaik Tanszékén írt,
Asztali naptár alkalmazás

.....
című szakdolgozatom/diplomamunkám saját, önálló szellemi termékem; az abban hivatkozott
szakirodalom felhasználása a szerzői jogok általános szabályainak megfelelően történt.

Tudomásul veszem, hogy szakdolgozat/diplomamunka esetén plágiumnak számít:

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

Budapest, 2023.05.24.

.....
Sólyomvári Dávid
hallgató aláírása

Tartalomjegyzék

1. Bevezetés	3
1.1. Feladat ismertetése	3
2. Felhasználói dokumentáció	4
2.1. Célközönség	4
2.2. Rendszerkövetelmény	4
2.3. Az alkalmazás használata	5
2.4. Bejelentkezés előtti állapot	5
2.4.1. Login dialógus ablak	5
2.4.2. CreateAccount dialógus ablak	7
2.5. Bejelentkezés utáni állapot	10
2.5.1. Admin felület	10
2.5.2. Clnr felület	11
2.5.3. Settings dialógus ablak	12
2.5.4. Mutual spare time finder dialógus ablak	14
2.5.5. AllEvent dialógus ablak	18
2.5.6. Help dialógus ablak	18
2.5.7. Account dialógus	19
2.6. Események	20
2.6.1. Esemény hozzáadása	20
2.6.2. Esemény szerkesztése	21
2.6.3. Esemény törlése	22
3. Fejlesztői dokumentáció	24
3.1. Tervezés	24
3.1.1. Architektúra	24
3.1.2. Kliens	24
3.1.3. Szerver	24

3.1.4.	Adatbázisterv	25
3.1.5.	Követelményelemzés	25
3.2.	Megvalósítás	26
3.2.1.	Felépítés	26
3.2.2.	Adatbázis megvalósítása	26
3.2.3.	Model osztályok	27
3.2.4.	Perzisztencia osztályok	31
3.2.5.	Nézet	34
3.3.	Tesztelés	35
3.3.1.	Egységtesztek	35
3.3.2.	Végfelhasználói tesztek	36
3.4.	Fejlesztési lehetőségek	38
Irodalomjegyzék		39
Ábrajegyzék		40
Táblázatjegyzék		42

1. fejezet

Bevezetés

1.1. Feladat ismertetése

A szakdolgozatom témája egy asztali naptár alkalmazás mely C++ ([1]) nyelven íródott. Az alkalmazást fiókkal rendelkező felhasználók használhatják bejelentkezés után. A bejelentkezéshez egy felhasználónév és egy jelszó szükséges.

Az alkalmazás hasonlóan viselkedik mint egy szokványos naptár: eseményeket tudunk létrehozni, szerkeszteni, törölni. Események létrehozásánál megadhatjuk mi legyen a neve az eseménynek, mettől meddig tartson, milyen címke tartozzon hozzá, milyen előfordulása legyen és mely felhasználókat szeretnénk meghívni az eseményre.

Az alkalmazás egyik egyedi funkciója a közös megfelelő időpont kereső. Ha felhasználó más felhasználókkal közös programot szeretne szervezni, de nem biztos benne hogy vajon melyik barátja mikor ér rá, a közös megfelelő időpont kereső megkeresi az időpontot amely minden résztvevőnek alkalmas.

Az alkalmazás megjelenését a felhasználók személyre tudják szabni. Két előre definiált megjelenés téma mellett lehetőségük van az alkalmazás főbb alkotóelemeinek színének beállítására.

Az alkalmazás tartalmaz egy segítség menüpontot, melyben a felhasználó tájékozódhat a program funkcióinak használatáról.

2. fejezet

Felhasználói dokumentáció

2.1. Célközönség

Mivel az alkalmazás egy naptár alkalmazás, így bármilyen társaság (például család, baráti kör) célközönség, amely tagjai eseményeket szeretnének létrehozni akár maguknak, akár közösen. A felhasználói felület felépítése egyszerű, felhasználóbarát, illetve személyre szabható.

2.2. Rendszerkövetelmény

Operációs rendszer

Windows 10-en fejlesztettem és teszteltem az alkalmazást.

Memória

8 GB memória ajánlott.

Processzor

Bármelyik modern processzor megfelelő.

Internetkapcsolat

Az alkalmazás használatához szükséges az internetkapcsolat a HTTP szerverrel való kommunikáció miatt.

2.3. Az alkalmazás használata

Az alkalmazás indításához a szerver és a kliens `.exe` fájljaira van szükség. Ezen két fájlra (először a szerverre, majd a kliensre) duplán kattintva tudjuk elindítani a két programot.

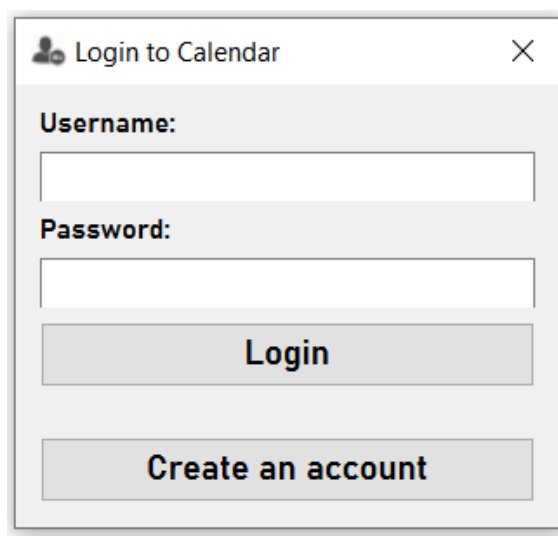
2.4. Bejelentkezés előtti állapot

2.4.1. Login dialógus ablak

Az alkalmazást megnyitva a Login dialógus ablak (2.1) fogad minket.

A Login dialógus biztosítja a számunkra az alkalmazásba való belépést. A dialógus a következő lehetőségeket tárja elénk:

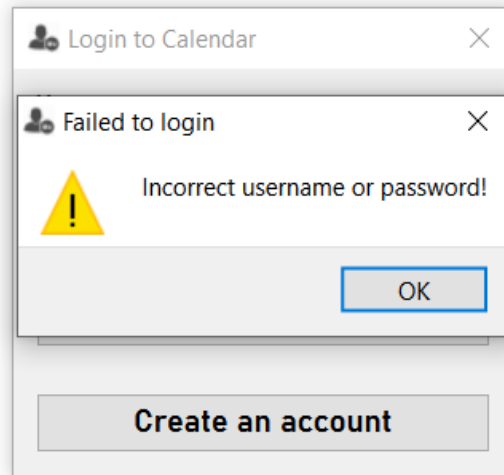
- Ha már van felhasználónk, a felhasználónevünkkel és jelszavunkkal be tudunk lépni az alkalmazásba a `LOGIN` gombra kattintva.
- Ha még nincs felhasználónk, a `CREATE AN ACCOUNT` gombra kattintva a `CreateAccount` dialógus jelenik meg (2.5).
- Ha úgy döntünk, elállunk bejelentkezési szándékunktól, és be szeretnénk zárni az alkalmazásunkat, az `X` gombra kattintva be tudjuk zárni a Login dialógus ablakot, ezzel együtt az alkalmazást is.



2.1. ábra. Login dialógus

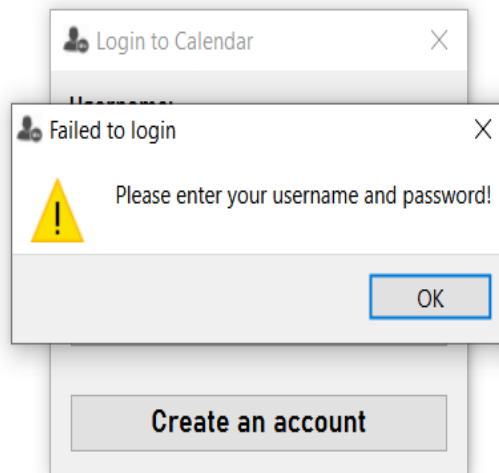
Ha a belépés sikertelen, a program hibaüzeneteken keresztül jelzi a felhasználónak. Az alábbi hibaüzeneteket kaphatjuk:

- Ha rosszul adjuk meg a felhasználónevünket vagy a jelszavunkat (vagy mindkettőt):



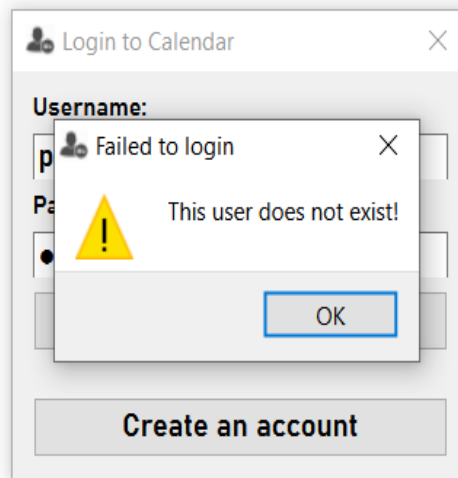
2.2. ábra. Hibás felhasználónév vagy jelszó

- Ha bármelyik szövegdobozt üresen hagyjuk:



2.3. ábra. Hiányos felhasználónév vagy jelszó

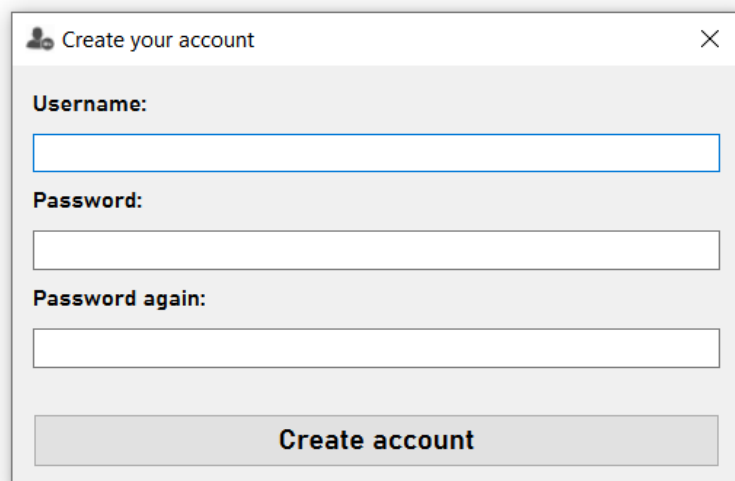
- Ha olyan felhasználónevet adunk meg, amivel még nem regisztrált senki felhasználót:



2.4. ábra. Ez a felhasználó nem létezik

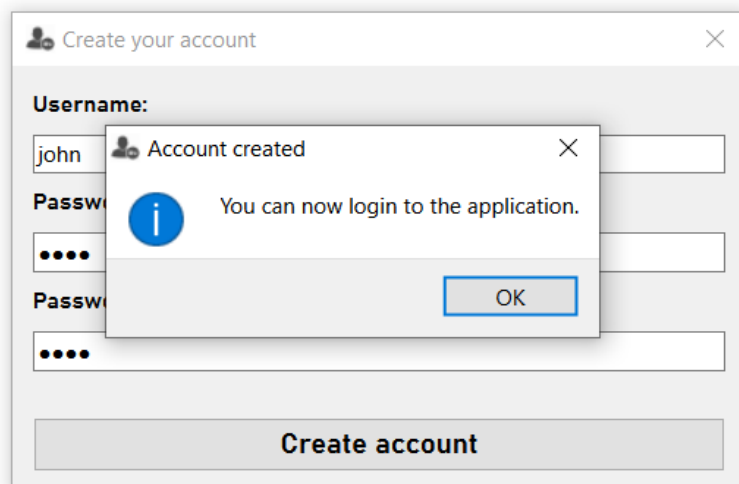
2.4.2. CreateAccount dialógus ablak

Az `CreateAccount` dialógus ablak segítségével tudunk új felhasználót létrehozni az alkalmazáshoz. A dialógusban az adott szövegdobozokba megadjuk az általunk preferált felhasználónevet (feltéve ha az még nem foglalt) és jelszót (amit kétszer kell megadnunk, ezzel eliminálva azt a eshetőséget, hogy egy elírt jelszót adjunk meg), így tudunk felhasználót készíteni magunknak. Ha ez sikeres volt, jelzi ezt nekünk az alkalmazás (2.6) és visszakerülünk a `Login` dialógus ablakhoz, ahol ezután az elkészített felhasználónkkal a `LOGIN` gombra kattintva be tudunk lépni az alkalmazásba.



A dialog box titled "Create your account" with a close button (X) in the top right corner. It contains three input fields: "Username:", "Password:", and "Password again:". Below the fields is a "Create account" button.

2.5. ábra. CreateAccount dialógus

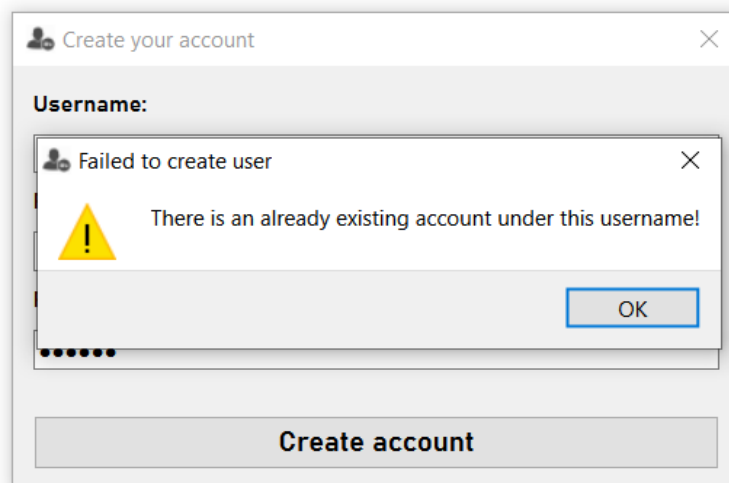


The "Create your account" dialog box is shown with the "Username:" field containing "john" and the "Password:" and "Password again:" fields masked with dots. A smaller dialog box titled "Account created" is overlaid on top, containing an information icon (i) and the text "You can now login to the application." with an "OK" button.

2.6. ábra. Sikeres új felhasználó készítés

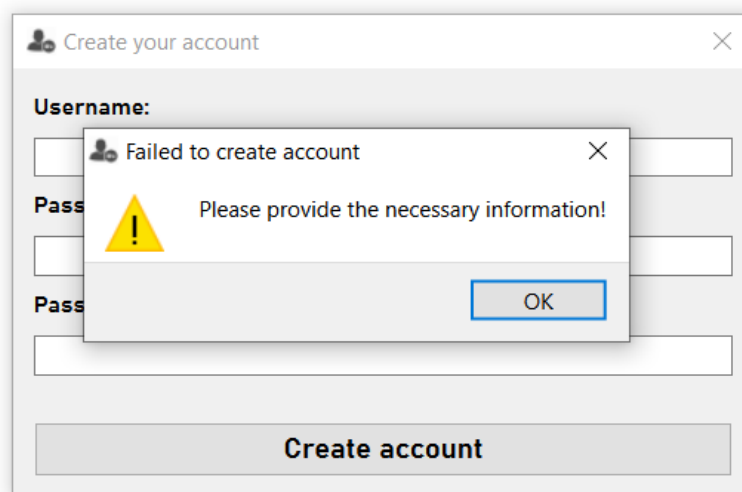
Hibás input esetén a CreateAccount dialógus is hibaüzenetekkel jelzi számunkra, hogy mit rontottunk el.

- Ha olyan felhasználónevet adunk meg, amelyen már létezik felhasználó:



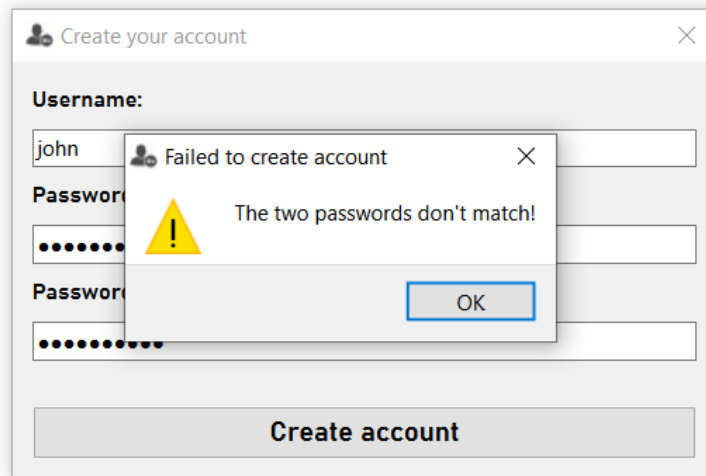
2.7. ábra. A felhasználónév már foglalt

- Ha bármelyik szövegdobozt üresen hagyjuk:



2.8. ábra. Hiányos felhasználónév vagy jelszó

- Ha úgy adjuk meg a kívánt jelszót másodjára, hogy az nem egyezik az elsővel:



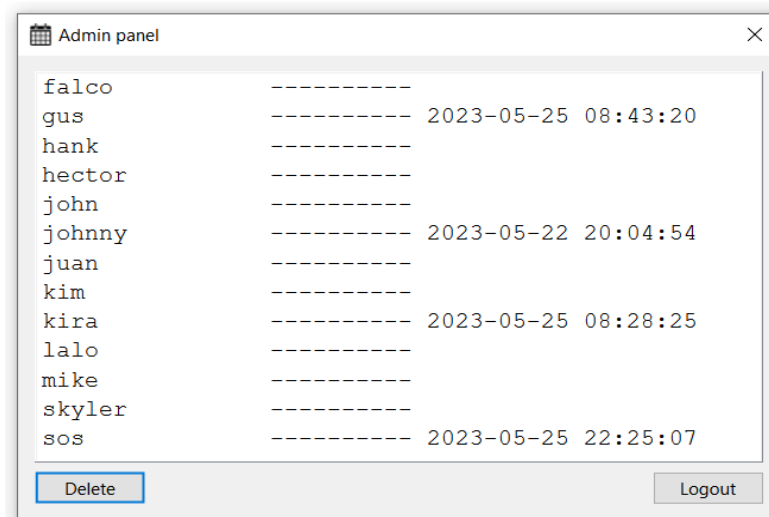
2.9. ábra. A jelszavak nem egyeznek

2.5. Bejelentkezés utáni állapot

2.5.1. Admin felület

Az alkalmazásban kétféle felhasználói szerepkör van: van egy darab admin felhasználó, a többi mind általános felhasználó. Ha az admin felhasználóval lépünk be az alkalmazásba, teljesen más felület fogad minket, mint az általános felhasználóknál.

Az admin felhasználó fő szerepe az alkalmazás általános felhasználóinak kezelése. Az **Admin** dialógus (2.10) megjeleníti az összes felhasználót akik az alkalmazást használják és hogy az egyes felhasználó mikor lépett be utoljára az alkalmazásba (ha még nem lépett be, akkor nem szerepel időpont a neve mellett). Az admin felhasználóként lehetőségünk van általános felhasználó törlésére, ezt a következőképpen tehetjük meg: a felsorolásban kijelölünk egy felhasználót (egyszerre csak egyet), majd a felület bal alsó sarkában elhelyezkedő **DELETE** gomb megnyomásával kitöröljük a felhasználót az adatbázisból, és ezzel együtt az összes eseményt, amely valamilyen formában kötődött a törölt felhasználóhoz. Az admin felületről kilépni a felület jobb felső sarkában található **X** gomb, vagy a jobb alsó sarkában lévő **LOGOUT** gomb megnyomásával lehet, mindkét esetben a **Login** dialógus ablak fogad minket.



2.10. ábra. Admin felület

2.5.2. Clndr felület

A Clndr az alkalmazás fő felülete, mely általános felhasználóval történő bejelentkezés után válik elérhetővé. A felület bal oldalán található a naptár, jobb oldalán pedig az adott napon lévő események listája, ez a két fő komponense(2.11).

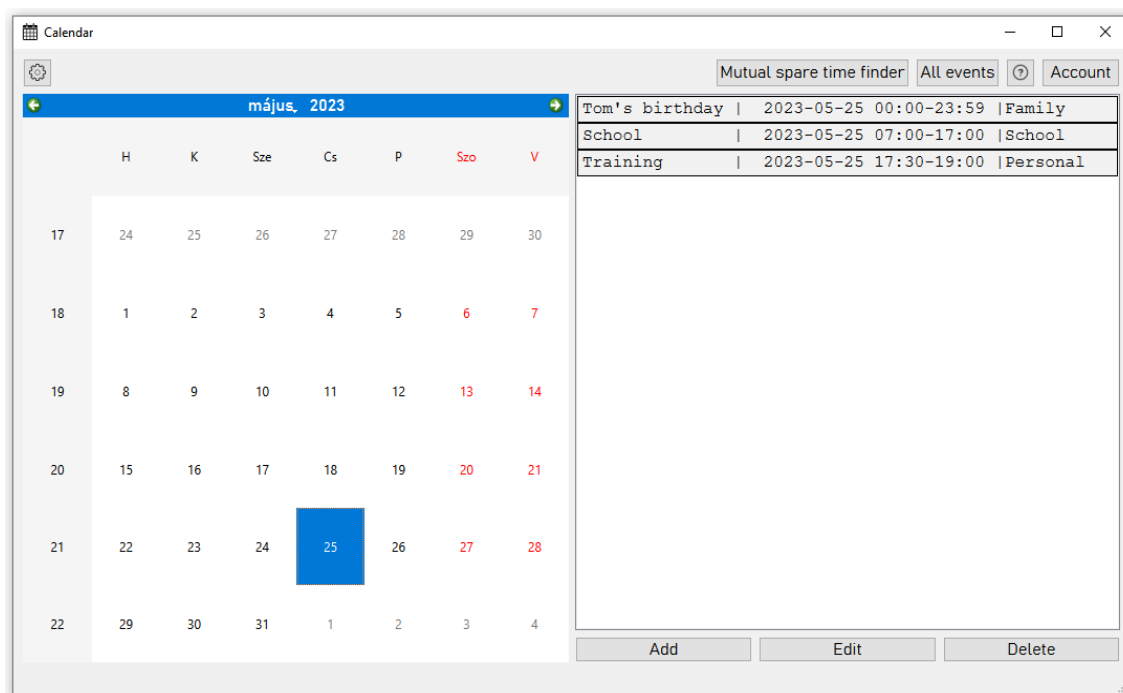
Használatuk a következőképpen történik:

Naptár komponens:

- a naptár tetején találjuk a jelen hónapot illetve évet, melyeket szabadon mi is beállíthatunk tetszőleges évre és hónapra
- a naptár bal oldali oszlopában az év heteinek számozását találjuk
- a naptár legfelső sorában a hét napjait láthatjuk
- a naptár napjaira rákattintva a jobb oldalon található listában meg tudjuk tekinteni, hogy arra a napra milyen események vannak betervezve

Lista komponens:

- ez a komponens listázza ki az adott napon lévő eseményeket
- e komponens segítségével tudunk eseményeket hozzáadni naptárunkhoz
- tetszőleges esemény kijelölése az eseményre való kattintással történik, ez után lehetőségünk van szerkesztetni, vagy törölni a meglévő eseményt



2.11. ábra. Clndr felület

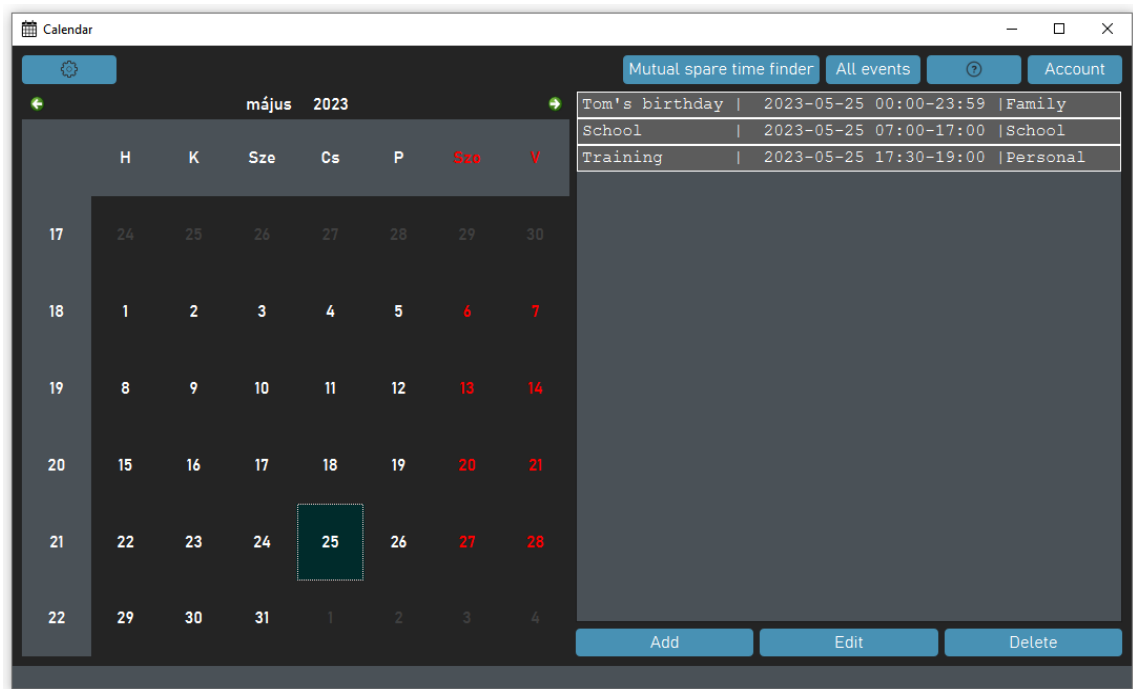
A felületen mindezek mellett beállítások, segítség, közös időpont kereső, összes esemény, fiók dialógus ablakok gombjai, valamint az eseményekhez kapcsolódó hozzáadás, szerkesztés, és törlés gombok találhatók.

2.5.3. Settings dialógus ablak

Az alkalmazás megjelenése személyre szabható, ehhez a **Settings** dialógus ablakot (2.14) kell megnyitnunk, melyet az alkalmazás bal felső sarkában elhelyezkedő fogaskerék ikonra való kattintással érhetünk el.

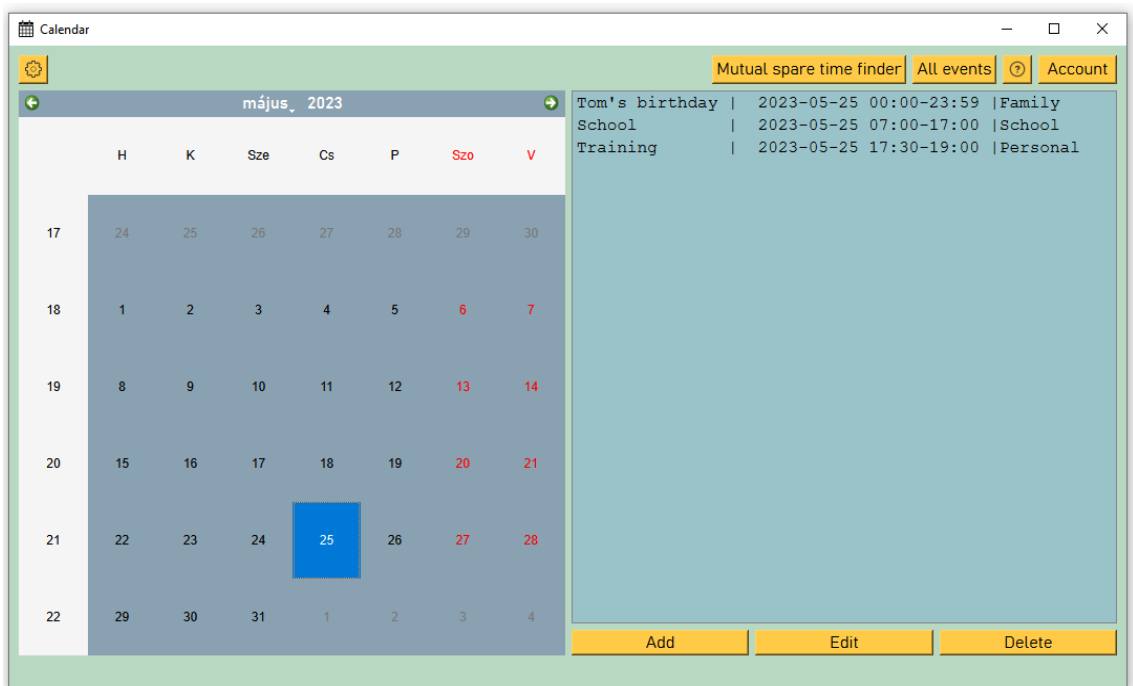
A **Settings** dialógusban kétféle opciónk van az alkalmazás személyre szabására:

- választunk egy előre definiált témát, vagyis a világos (2.11) vagy a sötét (2.12) témát



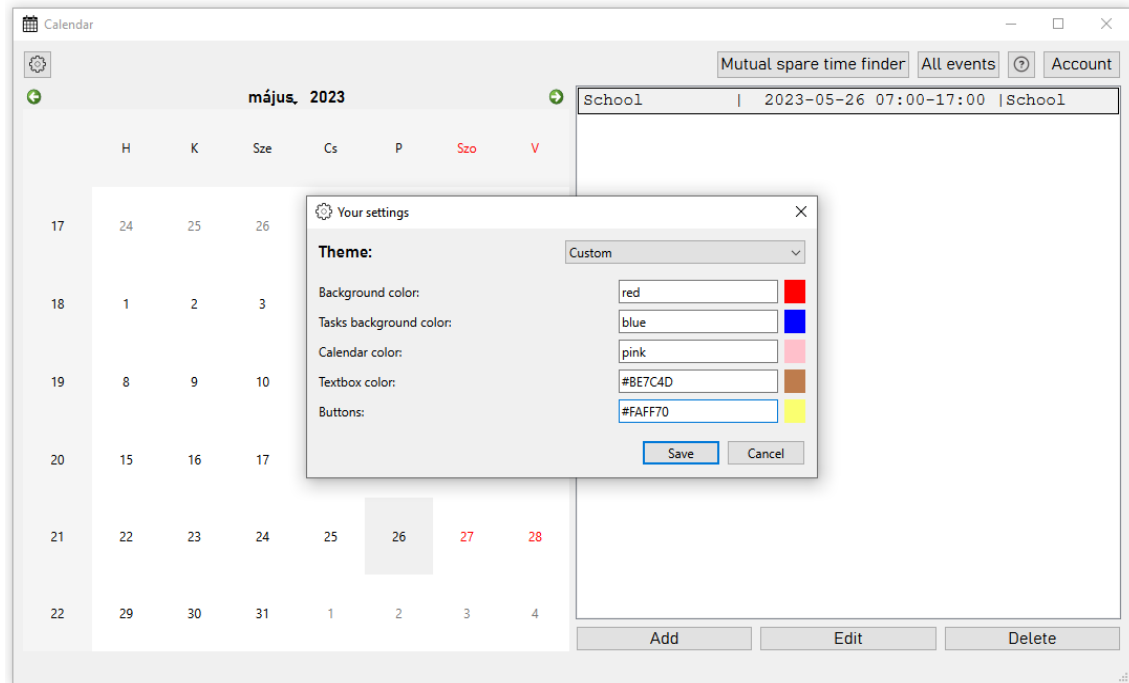
2.12. ábra. Sötét téma

- saját magunk adjuk meg az alkalmazás főbb alkotóelemeinek színét (2.13). Ehhez a lenyíló listában az **CUSTOM** opciót kell kiválasztanunk, majd ezután megadni a szövegdobokba a kívánt színeket. A színeket megadhatjuk **hex** színkódban, illetve ha angolul beírjuk a szín nevét, ugyanúgy el fogja fogadni az alkalmazás.



2.13. ábra. Custom téma

A szövegdobozokba beírt színeket (ha érvényesek) a szövegdobozok mellett megjelenő négyzetekben (2.14) láthatjuk vizuálisan, így pontosan tudni fogjuk hogy az adott komponens színe milyen lesz. Ez főként a hex kódoknál lehet hasznos.

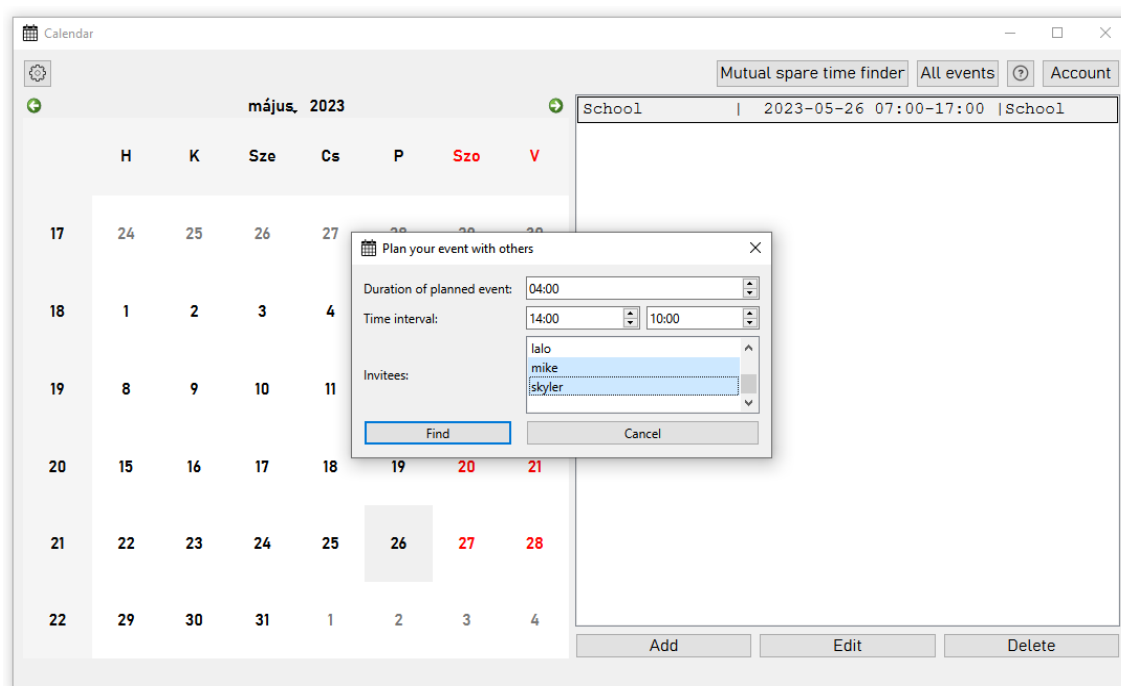


2.14. ábra. Settings dialógus

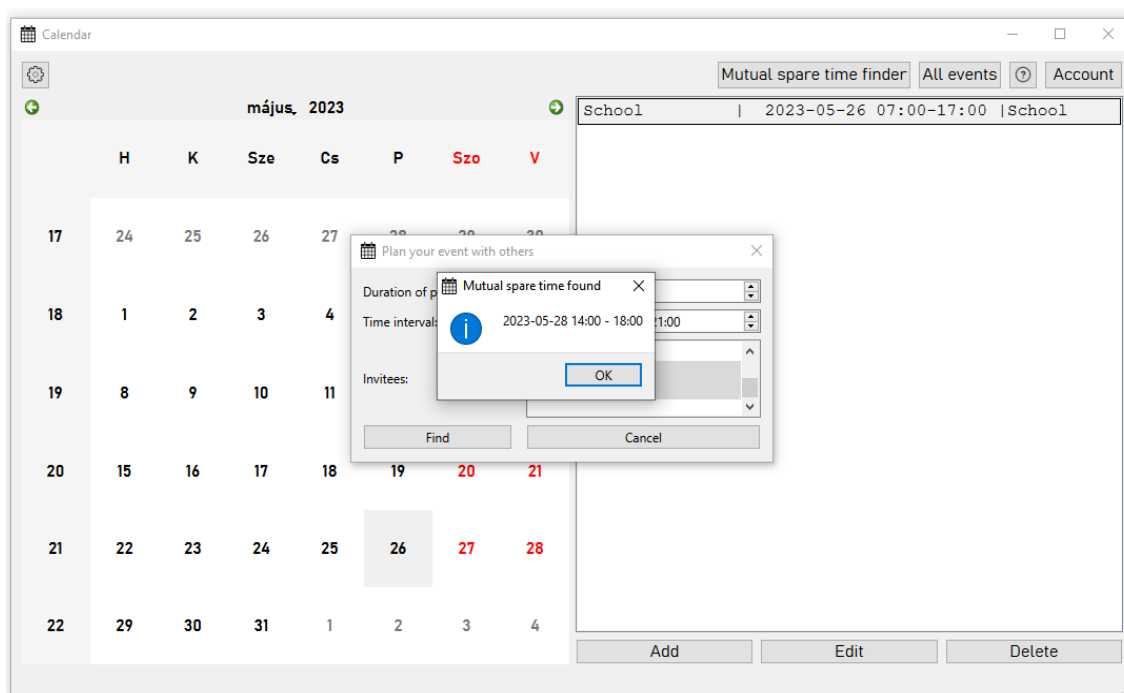
2.5.4. Mutual spare time finder dialógus ablak

Ezt a dialógus ablakot közös események szervezésekor tudjuk segítségül hívni a közös megfelelő időpont kereséséhez. Ha a **ClnDr** felületen rákattintunk a **MUTUAL SPARE TIME FINDER** gombra, az alábbi dialógus ablak fogad minket (2.15). Miután megadtuk milyen hosszú eseményt szeretnénk, és milyen időszámban (mettől meddig), a közös megfelelő időpont kereső az elkövetkező két hétben próbál a megadott kritériumoknak megfelelő időpontot találni. Ha sikerrel járt, informálja a felhasználót a legkorábbi megfelelő időpontról. (2.16) Ha nem talált, azt is jelzi számunkra (2.19).

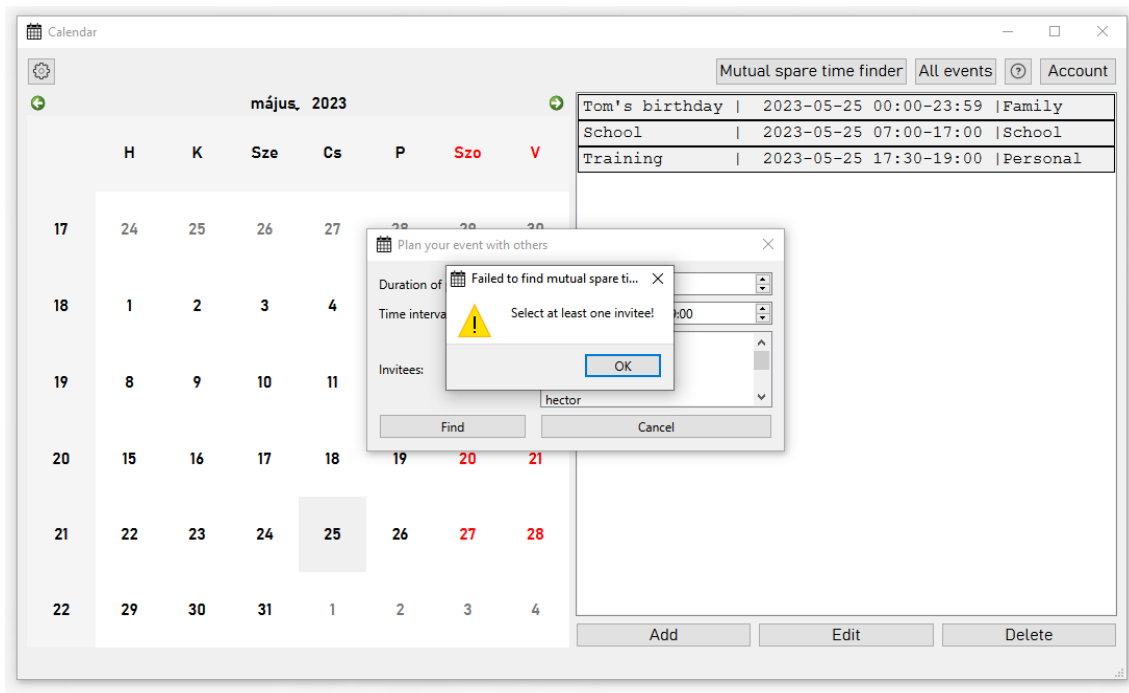
Hibás bemenet esetén a program tájékoztat minket a hibáról.(2.17)(2.18)



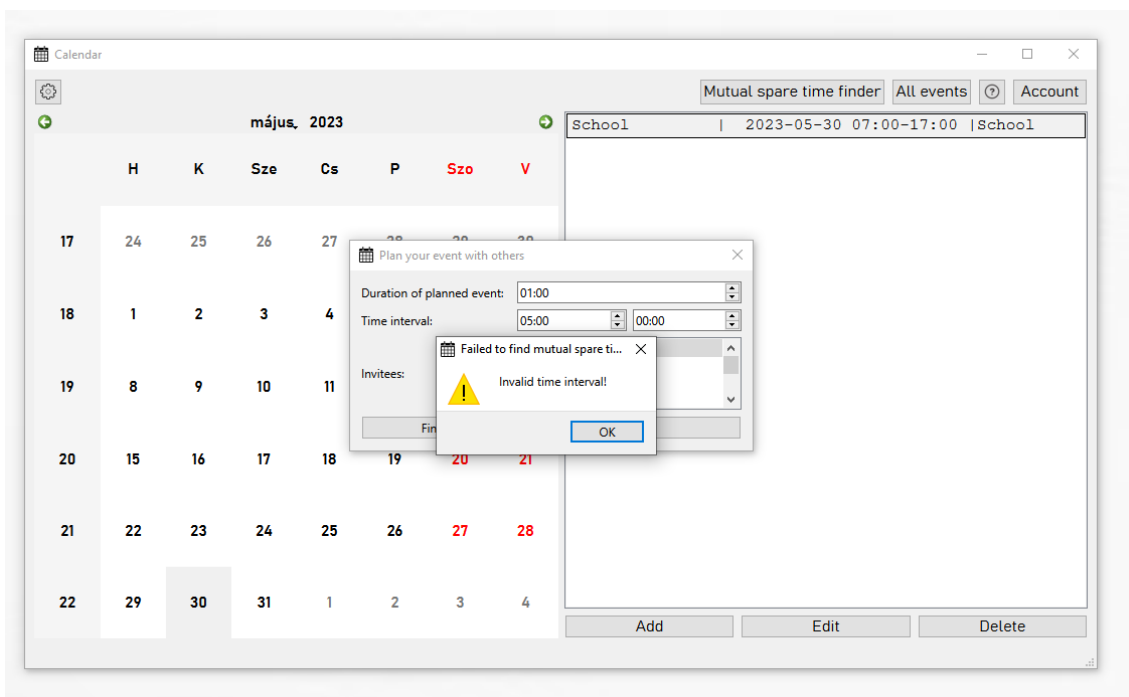
2.15. ábra. Mutual spare time finder dialógus



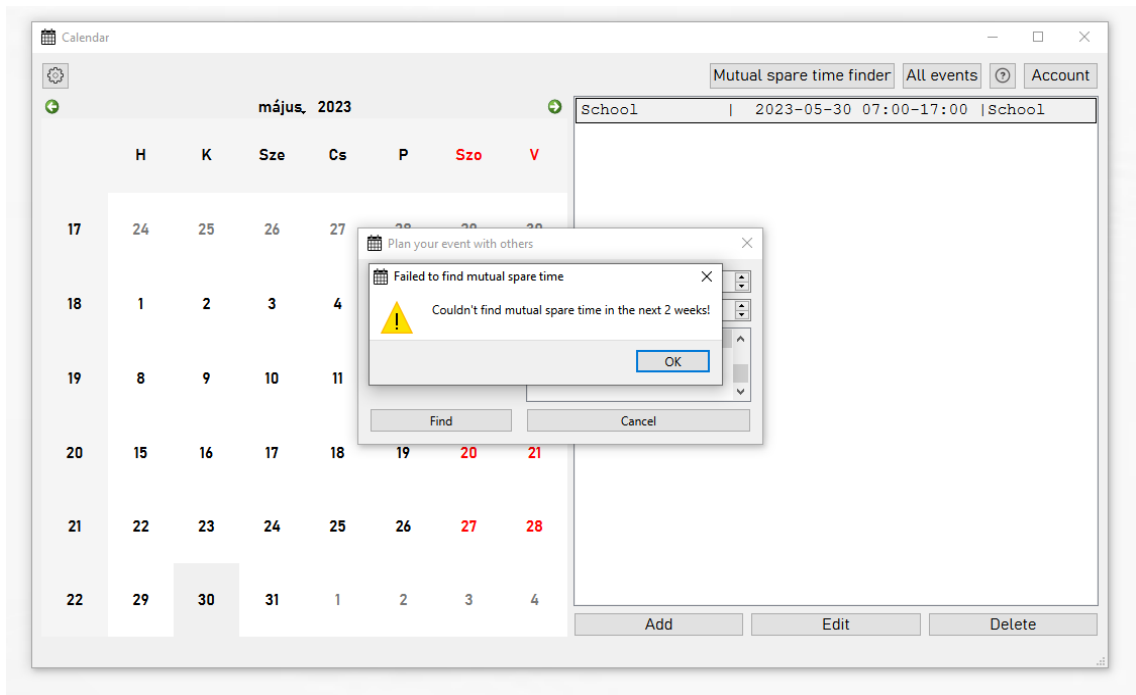
2.16. ábra. Sikeres közös megfelelő időpont keresés



2.17. ábra. Hiányos közös megfelelő időpont keresés



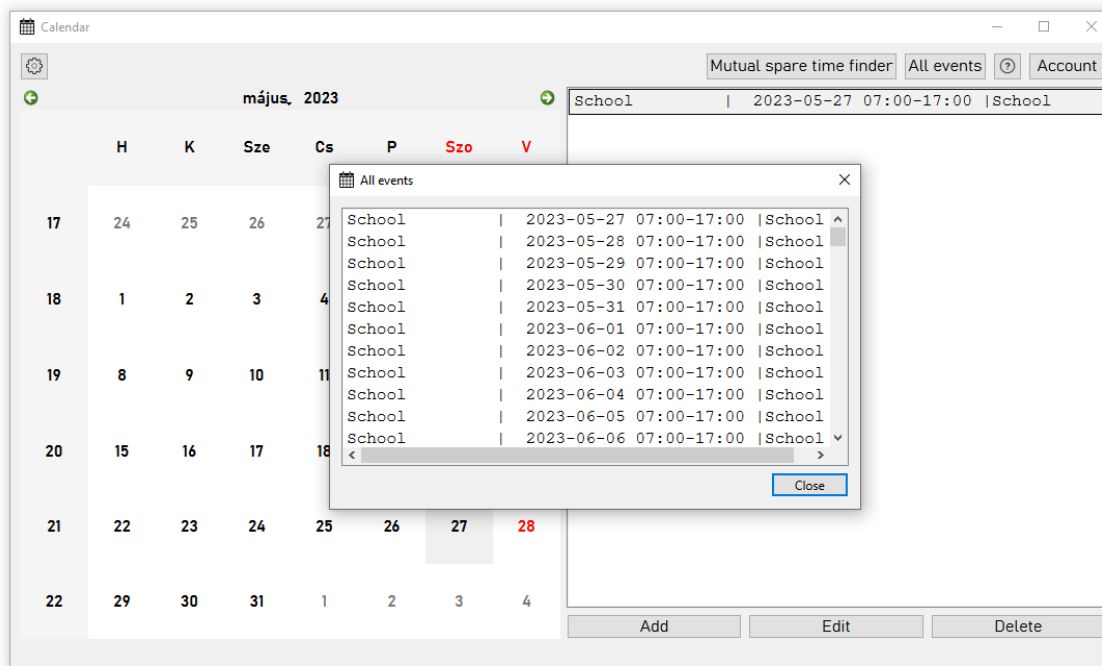
2.18. ábra. Hibás közös megfelelő időpont keresés



2.19. ábra. Sikertelen közös megfelelő időpont keresés

2.5.5. AllEvent dialógus ablak

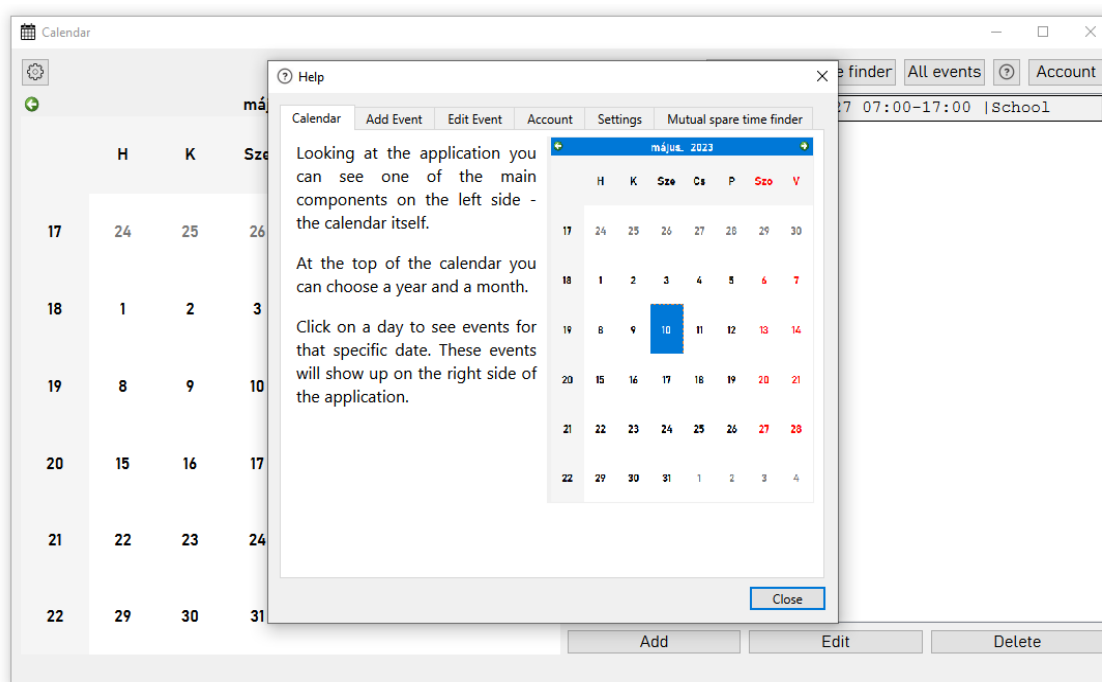
Az AllEvent dialógus ablakban (2.20) megtekinthetjük az összes olyan, eseményt amely esedékessége a jelenlegi dátumom, vagy utána van. Ez azért hasznos, mert így egy helyen meg tudjuk névni az eseményeinket, nem csak naponként.



2.20. ábra. AllEvent dialógus

2.5.6. Help dialógus ablak

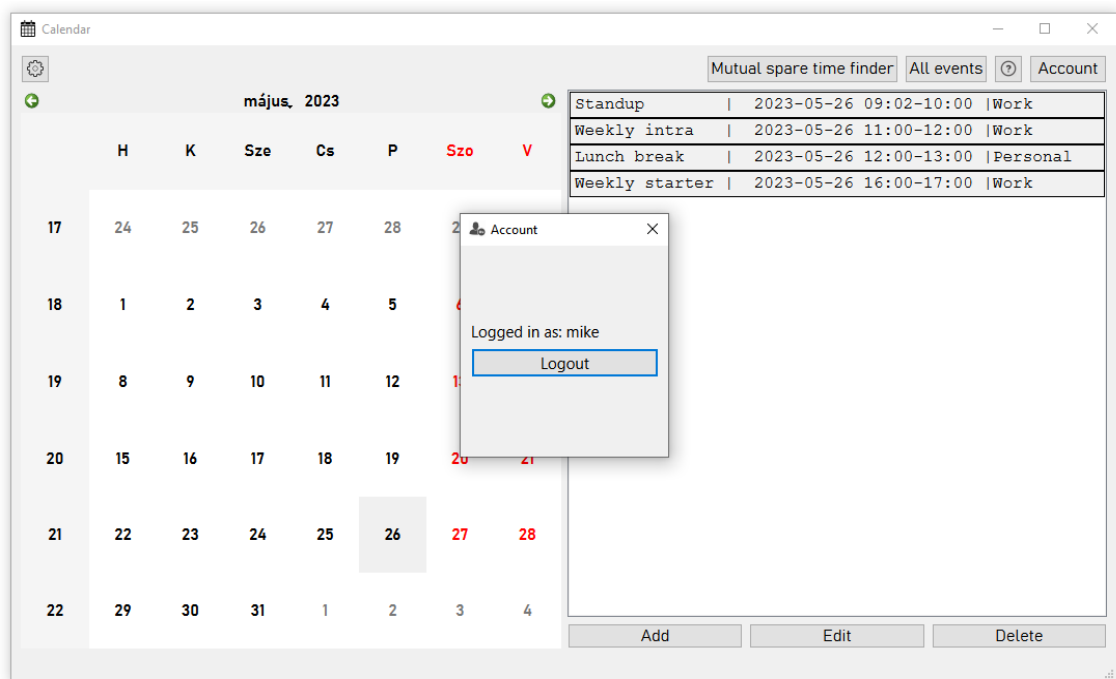
Ha úgy érezzük, segítségre van szükségünk az alkalmazás használatához, az alkalmazás felületén bal felül fellelhető ? gombra kattintva a Help dialógus ablak (súgó) képekkel és útmutatóval magyarázza el számunkra az alkalmazás főbb funkcióinak és elemeinek működését. (2.21)



2.21. ábra. Help dialógus

2.5.7. Account dialógus

Az Account dialógus ablak (2.22) az alkalmazás bal felső sarkában található ACCOUNT gomb megnyomásával idézhető elő. A dialógus megjeleníti hogy épp melyik felhasználó van bejelentkezve, valamint az alkalmazás fő felületéről való kilépést is itt tudjuk végrehajtani, ehhez a LOGOUT gombra kell kattintanunk. Ezután ismételten a Login dialógus ablak fogad minket.



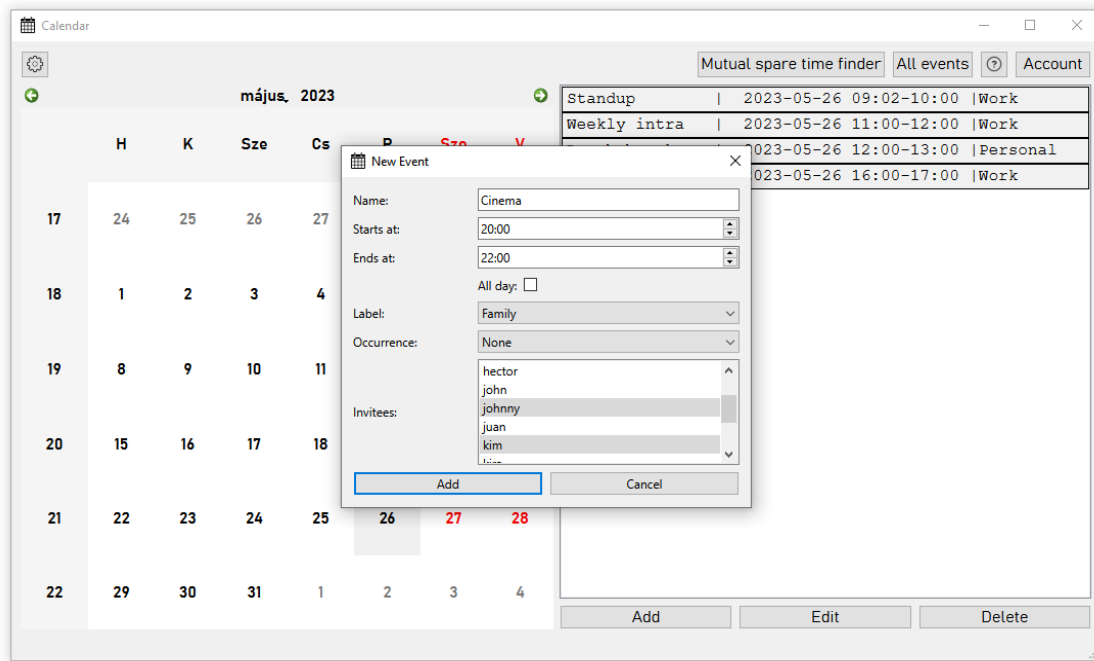
2.22. ábra. Account dialógus

2.6. Események

Naptár alkalmazás lévén alkalmazásom legfontosabb funkciója az események kezelése. Az eseményeken végezhető műveletek a hozzáadás (2.6.1), a szerkesztés (2.6.2), illetve a törlés (2.6.3).

2.6.1. Esemény hozzáadása

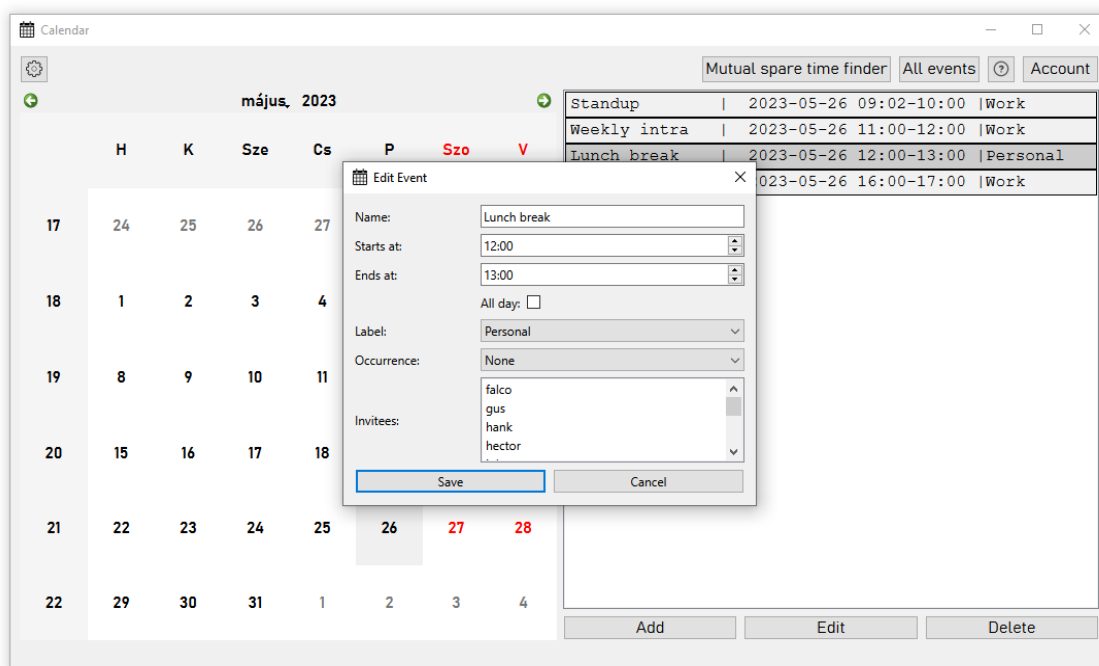
Esemény hozzáadásához először egy tetszőleges napot kell kijelölnünk a naptárban, majd az **ADD** gombra kattintanunk. Ekkor megjelenik az **AddEvent** dialógus ablak (2.23), ahol be tudjuk állítani az esemény tulajdonságait. Meg tudjuk adni az esemény nevét, mikor kezdődjön és mikor legyen vége, vagy ha egész napos eseményt tervezünk, a jelölőnégyzetre való kattintással tudjunk ezt beállítani. Az eseményhez tudunk címkét hozzáadni, valamint kiválaszthatjuk, hogy milyen előfordulása legyen az eseménynek. Ha szeretnénk meghívni más/másokat is az eseményre, egy lenyíló listából tudunk több felhasználót is hozzáadni eseményünkhöz. Az esemény létrehozása a dialógus alján található **ADD** gombra való kattintással vihető végbe. Ekkor a jelenlegi dialógus bezáródik, és visszakerülünk az alkalmazás fő felületére.



2.23. ábra. AddEvent dialógus

2.6.2. Esemény szerkesztése

Egy meglévő esemény szerkesztéséhez egy létező eseményt kell kijelölnünk a listából, majd az **EDIT** gombra kattintani. Ezután megjelenik az **EditEvent** dialógus ablak (2.24) amely felépítésben megegyezik az **AddEvent** dialógus ablakkal, viszont a szerkesztendő esemény adatai be vannak töltve az egyes mezőkbe. Az esemény szerkesztése után a **SAVE** gombra kattintva elmentődik a szerkesztett esemény, a dialógus bezárul, és a bal oldali esemény listában már a frissített esemény jelenik meg. Ismétlődő esemény szerkesztésekor a gyöker esemény szerkesztése esetén, és sima eseménynél szerkesztésénél is csak az adott eseményt szerkesztjük.

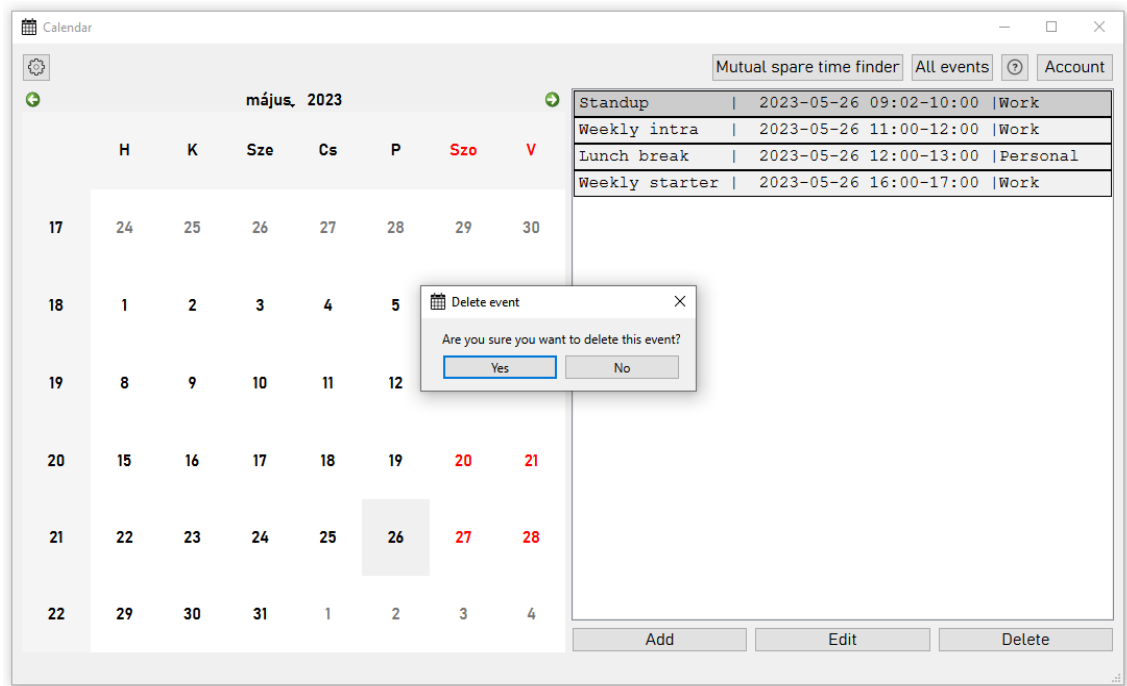


2.24. ábra. EditEvent dialógus

2.6.3. Esemény törlése

Egy meglévő esemény törléséhez egy létező eseményt kell kijelölnünk a listából, majd az DELETE gombra kattintani. Ekkor egy dialógus ablak jelenik meg (2.25), mely megkérdezi, biztosan törölni szeretnénk-e az eseményt. Ha igen, a YES gombra kattintva a kijelölt esemény törlődik az adatbázisból, ezzel együtt a listából is. Ha meggondolnánk magunkat, a NO gombra, vagy a jobb felső sarokban lévő X gombra kattintva vonhatjuk vissza a kiválasztott esemény törlését.

Ismétlődő esemény szerkesztésekor a gyökér esemény törlése esetén a teljes eseménysorozat törlődik, míg sima eseménynél törlésénél csak az adott esemény törlődik.



2.25. ábra. DeleteEvent dialógus

3. fejezet

Fejlesztői dokumentáció

3.1. Tervezés

3.1.1. Architektúra

Az szakdolgozat programjának architektúrájának tervezésekor a háromrétegű Model-View-Persistence (model-nézet-perzisztencia) architektúra elveit követtem: a kliens felhasználói felülete mint nézet, a nézet mögött lévő alkalmazáslogika mint model, és a webszerver, mely az adatbázissal áll összeköttetésben, mint perzisztencia.

3.1.2. Kliens

A kliensnél a grafikus felhasználói felületet egy fő `QMainWindow`-ból, a többi felületet `QDialog`-ból leszármazó osztályok valósítanak meg. Mindezek mögött lenne a model, mely a webszerverrel való kommunikációt bonyolítaná le.

3.1.3. Szerver

Az adatelérést a szerver bonyolítaná le, közvetlenül csatlakozna az SQLite adatbázishoz. A szervert a külön komponensként kell megvalósítani.

3.1.4. Adatbázisisterv

Alkalmazásom funkcionalitását nagy részt események és a rajtuk végzett műveletek teszik ki. Emiatt erősen indokoltnak éreztem, hogy az adatbázisban dedikált táblát kapjon. Továbbá szükségem volt egy táblára, melyben a felhasználókat tárolom, illetve egy másikra, ahol az előbb említett két táblát kötöm össze egymással.

3.1.5. Követelményelemzés

Minden alkalmazás tervezéséhez szervesen hozzátartozik a követelmények elemzése. Fontos átgondolni, hogy az alkalmazás milyen funkciókkal rendelkezzen (3.1), vagyis az alkalmazás a felhasználónak milyen eszközöket biztosítson egy adott cél eléréséhez.

Szakedolgozatom esetében események hozzáadása, szerkesztése, törlése, illetve a közös megfelelő időpont kereső jelentenék a legfőbb funkcionalitást.

Az alkalmazáshoz használata bejelentkezéshez lenne kötve. Bárki tud magának új felhasználót létrehozni. Az általános felhasználó és az admin külön felületet kapnak.

Az alkalmazás legyen egyszerű használatú, felhasználóbarát, személyre szabható, Legyen benne súgó, illetve kijelentkezés funkció.

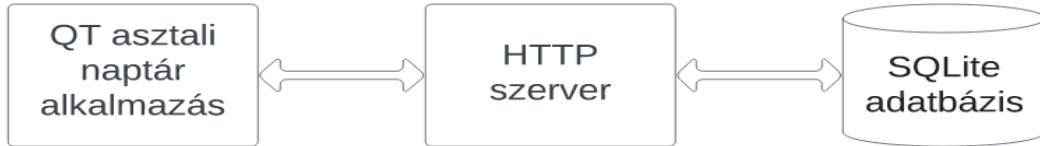


3.1. ábra. Usecase diagram

3.2. Megvalósítás

3.2.1. Felépítés

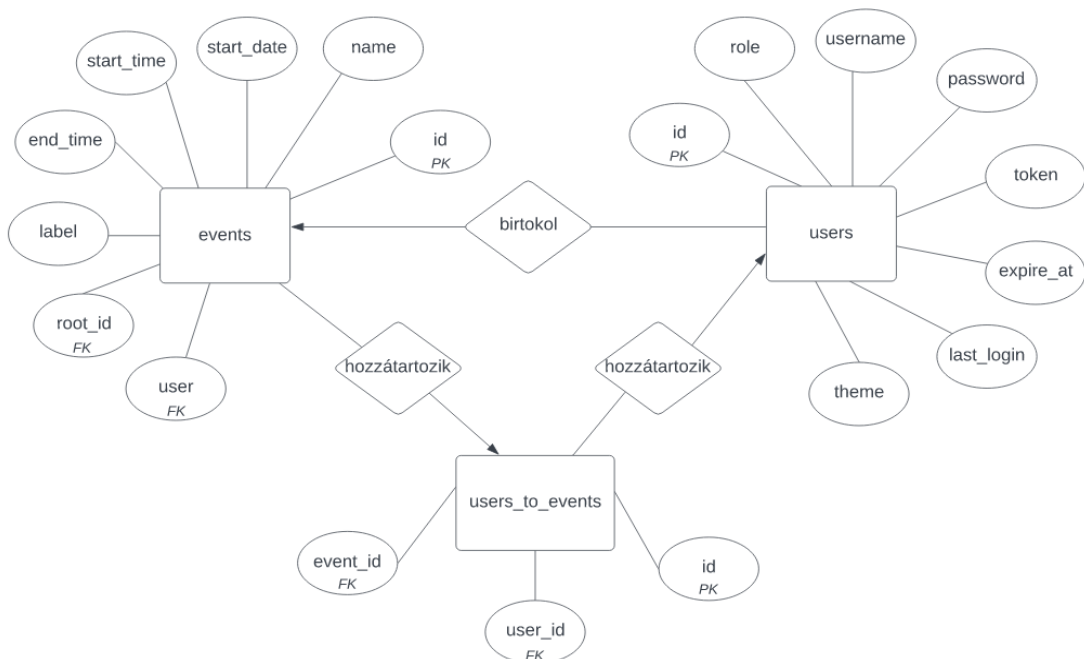
A szakdolgozatot három komponensből tevődik össze:



3.2. ábra. Szakdolgozat komponensei

A naptár alkalmazással a felhasználó interaktál, kizárólag ennek a komponensnek van grafikus felhasználói felülete, mely a Qt ([2]) ui elemeiből áll. A HTTP szerver ([3]) köti össze a klienst és az adatbázist. A kliens modelje HTTP request-eket küld a szervernek, amelyekre az válaszol a kérésnek megfelelő SQL lekérdezés vagy utasítás eredményével. Az SQLite adatbázis ([4]) felel az adatok tárolásáért.

3.2.2. Adatbázis megvalósítása



3.3. ábra. Adatbázis megvalósítása

Az adatbázisnak három táblája van:

- **users** tábla:

- Ez a tábla tárolja a felhasználók adatait.
- A regisztrációnál megadott adatokon kívül tartalmazza a belépéskor generált token-t és annak lejárási időpontját, a felhasználó utolsó belépési időpontját, illetve a személyes preferencia szerint beállított alkalmazás téma értékét.
- Elsődleges kulcsa az `id` mező.

- **events** tábla:

- Ez a tábla tárolja az események adatait.
- Elsődleges kulcsa az `id` mező.
- Idegen kulcsai a `user` és a `root_id`.
- `root_id`-ra ismétlődő eseményeknél van szükség.

- **users_to_events** tábla:

- Elsődleges kulcsa az `id` mező.
- Idegen kulcsai a `user_id` és a `event_id`.
- A `user_id` mező adja meg, hogy az egyes eseményekhez mely felhasználókat rendeljük hozzá pluszban.

3.2.3. Model osztályok

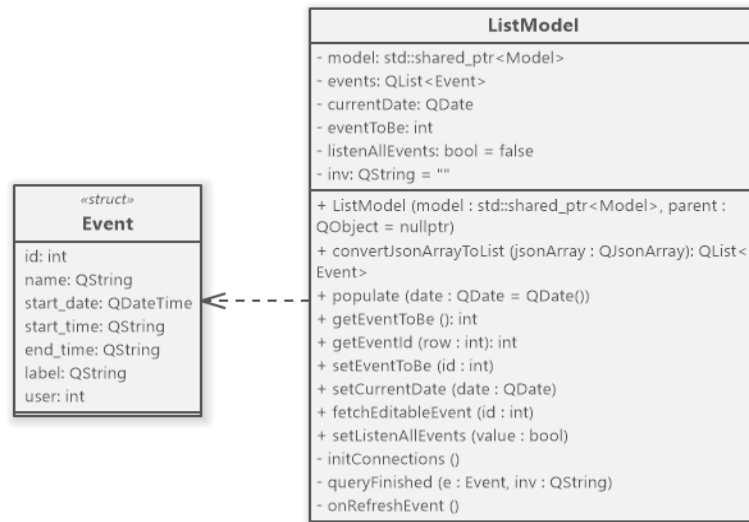
Az model osztályok felelnek az alkalmazás üzleti logikájáért. A szerver felé HTTP kéréseket küldenek, és HTTP válaszokat kapnak vissza, majd ezeket továbbítják a nézet felé.

ListModel osztály

A `ListModel` osztály (3.4) a nézetben megjelenő `QListView` ui komponensek mögötti model, az események megjelenítéséért felel. `QAbstractListModel`-ből származik le, melynek `rowCount` és `data` függvényeit implementálja.

Összes függvénye a `Model` osztály függvényeit hívja, majd a `connect` függvényeken keresztül a kapott eredményt lekezeli. A `convertJsonArrayToList` alakítja át a `Json`-ben kapott választ események listájává.

A `populate` függvény felel a model adatokkal való feltöltéséért, a `getEventId` egy adott esemény szerkesztésekor a modelből adja vissza az elem indexét, a `setCurrentDate` a naptárban történt minden dátum változáskor lefut.



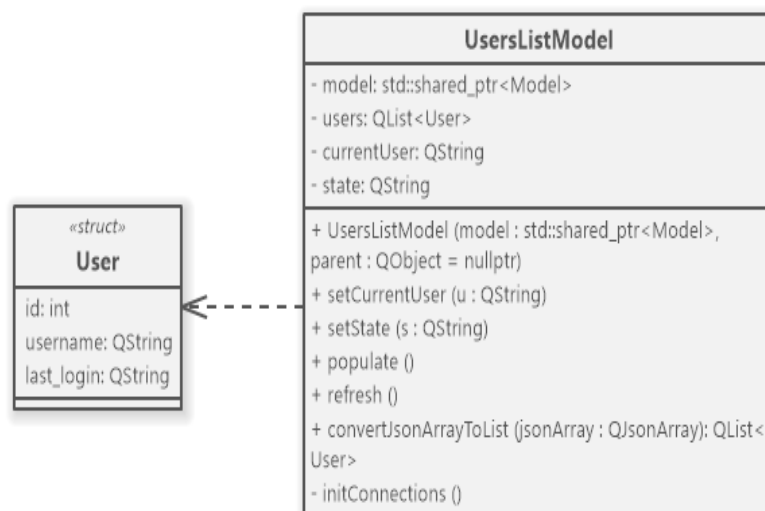
3.4. ábra. ListModel osztálydiagramja

UsersListModel osztály

A `UsersListModel` osztály (3.5) a felhasználók megjelenítéséért felelő model. `QAbstractListModel`-ből származik le, melynek `rowCount` és `data` függvényeit implementálja.

Összes függvénye a `Model` osztály függvényeit hívja, majd a `connect` függvényeken keresztül a kapott eredményt lekezeli. A `convertJsonArrayToList` alakítja át a `Json`-ben kapott választ felhasználók listájává.

Az osztályban a `populate` függvény felel a model adatokkal való feltöltéséért.



3.5. ábra. UsersListModel osztálydiagramja

Model osztály

A `Model` osztály végzi a kommunikációt a HTTP szerverrel, amit a `QNetworkAccessManager` segítségével tesz meg. Ezt az osztály szinte mindegyik nézet osztály `include`-olja.

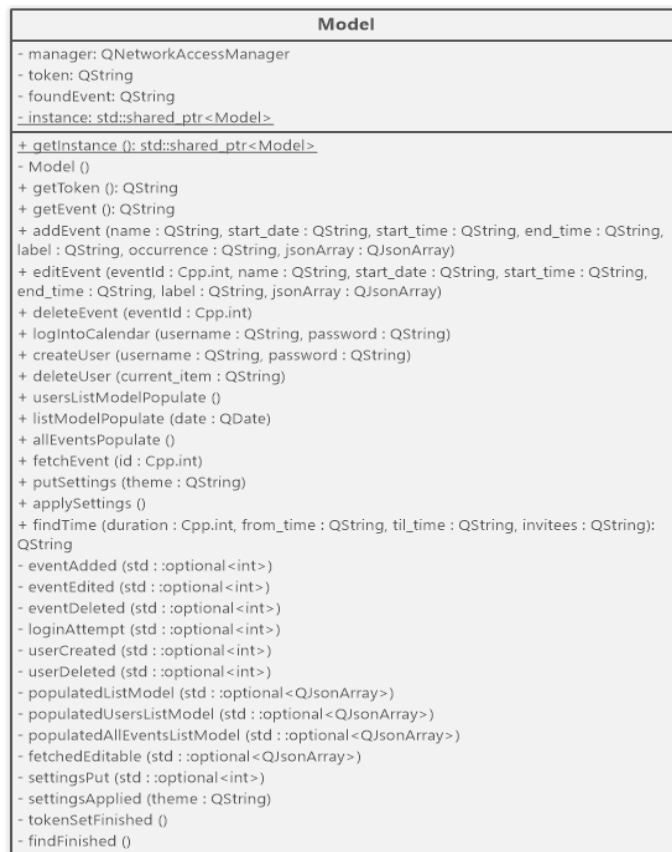
Publikus konstruktor helyett a singleton tervezési mintát követve egy publikus instance `std::shared_ptr<Model>` hozódik létre, amit minden `Model` osztályt használó osztály a konstruktorában megkap paraméterként. Fontos szerepe van a token adattagnak, ami a sikeres belépés után kap értéket, majd utána az összes hálózati kommunikációt használó művelet csakis kizárólag ennek segítségével lehetséges.

A függvények egy adott végpontra küldik a kérésüket amit a `QNetworkAccessManager` egyik metódusa (`get`, `post`, `put`, `deleteResource`) kap paraméterül. A `connect` függvényben feliratkozunk a `QNetworkReply finished` signal-jára, és ha befejeződik a szerver oldali tevékenység, lekezeljük a kapott választ amit egy `QNetworkReply` típusú pointer kap meg. Sikeres és sikertelen esetben is emittál egy `signal`-t, amire az egyik nézet osztály iratkozik fel `connect` függvénnyel.

Fontosabb metódusok:

- `addEvent`: Esemény hozzáadása.
- `editEvent`: Esemény szerkesztése.
- `fetchEvent`: A szerkeszteni kívánt esemény adatainak lekérése.

- deleteEvent: Esemény törlése.
- logIntoCalendar: Bejelentkezés az alkalmazásba.
- createUser: Felhasználó létrehozása.
- deleteUser: Felhasználó törlése.
- putSettings: Beállított téma mentése.
- applySettings: Téma beállítása.
- findTime: Közös megfelelő időpont keresés.



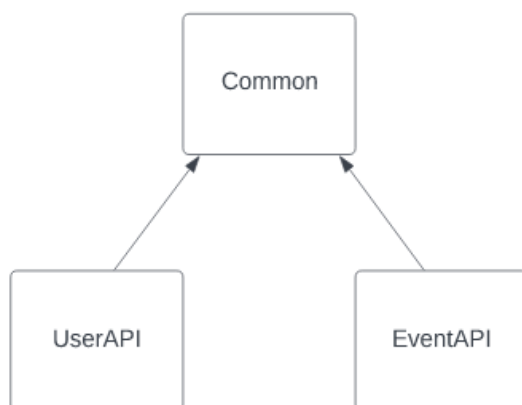
3.6. ábra. Model osztálydiagramja

3.2.4. Perzisztencia osztályok

Az alkalmazásban a HTTP szerver végzi az adatelérést, mivel a szerver közvetlenül csatlakozik az adatbázishoz. A szerver esetében csak backendről beszélhetünk.

QSqlQuery-k készítik elő, majd futtatják az sql parancsokat, lekérdezéseket. A kapott adatokat az API továbbítja a kliens modeljének.

A main függvényben definiálom, hogy adott endpoint-nál milyen függvény hívódjon meg.



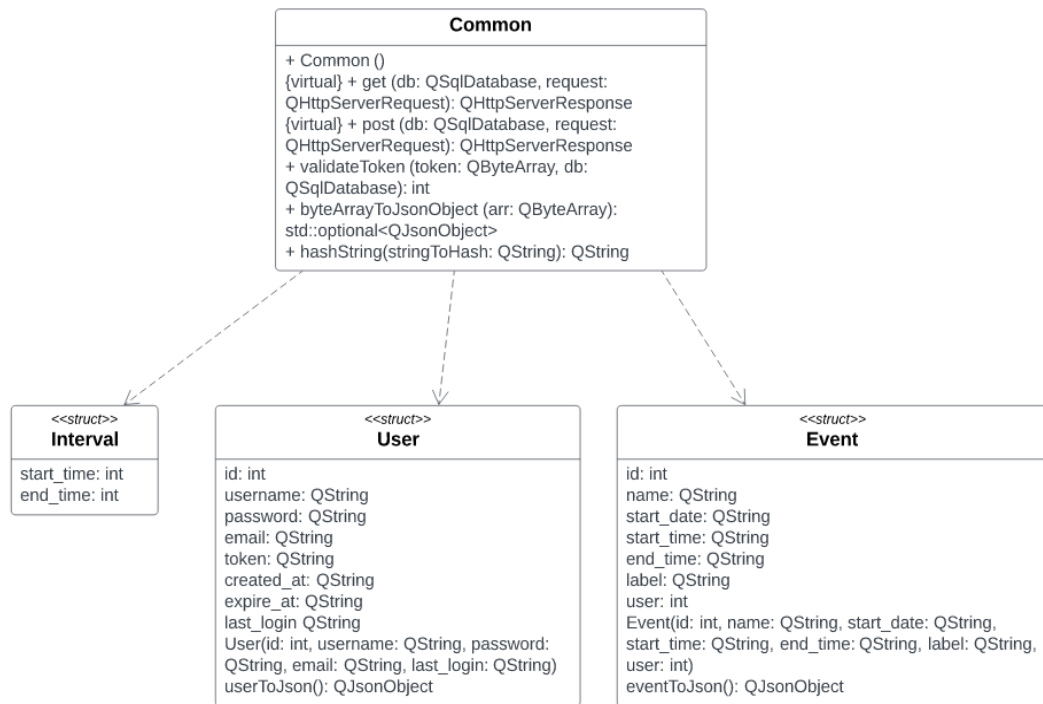
3.7. ábra. Perzisztencia réteg hierarchiája

Common osztály

A **Common** osztály (3.8) egy absztrakt osztály, amely tartalmazza az API által használt két struktúrát: a **User**-t és az **Event**-et. Emellett a két másik perzisztencia osztály által közösen használt függvények implementációját találjuk ebben az osztályban. Ezen függvények és feladatuk:

- **validateToken**: Ez a függvény felel a token validációért, ellenőrzi, hogy egyezik-e a token, és hogy még érvényes-e.
- **byteArrayToJsonObject**: Az API post request-ek body-ját alakítja át QJsonObject-té.
- **hashString**: Jelszavak hashselése és a token létrehozása a feladata.
- **get**: Tisztán virtuális függvény.

- **post**: Tisztán virtuális függvény.



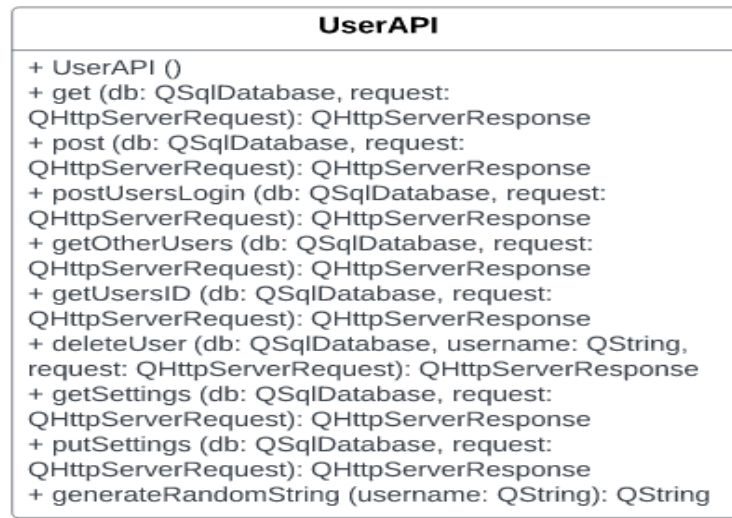
3.8. ábra. Common osztálydiagramja

UserAPI osztály

A UserAPI osztály a felhasználókon végzett sql műveletek végrehajtásáért felel. A Common osztályból származik és felüldefiniálja `get` és `post` metódusát.

- **generateRandomString**: Ez a függvény generálja a token, ami létfontosságú az alkalmazásban végezhető tevékenységekhez.
- **get**: Az adatbázisban szereplő összes felhasználót vissza adja.
- **post**: Új felhasználó létrehozása a feladata.
- **postUsersLogin**: Felhasználó bejelentkezésekor fut le a függvény, itt kapjuk meg a token.
- **getOtherUsers**: `UsersListModel`-nél az adatok model-be való betöltéséért felel.
- **getUsersID**: Esemény törlését végzi azonosító alapján.

- **deleteUser**: Felhasználó törlését végzi a metódus.
- **getSettings**: Az egyes felhasználóhoz tartozó téma beállítások lekérését végzi.
- **putSettings**: A téma itt állítódik át.



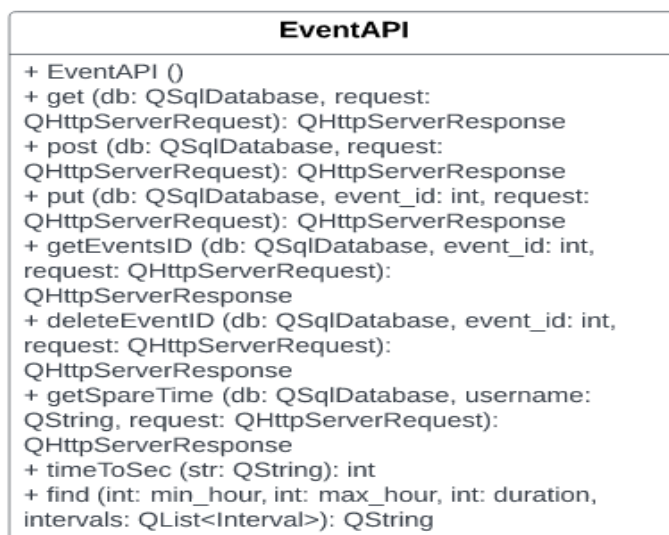
3.9. ábra. UserAPI osztálydiagramja

EventAPI osztály

A EventAPI osztály az eseményeken végzett sql műveletek végrehajtásáért felel. A Common osztályból származik és felüldefiniálja **get** és **post** metódusát. Főbb metódusok és feladatuk:

- **find**: Egy adott napon belül ellenőrzi, van-e közös megfelelő időpont.
- **get**: A request tartalmától függően az egy **User**-hez tartozó összes eseményt vagy az egy napon lévő összes eseményt küldi a kliensnek.
- **post**: Esemény létrehozását bonyolítja le a kapott adatokkal.
- **put**: Esemény szerkesztését bonyolítja le a kapott adatokkal.

- `getEventID`: Esemény szerkesztése előtt tudnunk kell, hogy melyik eseményt szerkesztjük, és annak mik a tulajdonságai. Ez a függvény az előbb említett adatokat küldi a kliensnek.
- `deleteEventID`: Esemény törlését végzi azonosító alapján.
- `getSpareTime`: Visszatér a közös megfelelő időpont keresés eredményével, ami sikeres keresés esetén egy időintervallum.



3.10. ábra. EventAPI osztálydiagramja

3.2.5. Nézet

A fő felületen (CInDr) kívül az összes ablak `QDialog` ami az alkalmazáson belül megjelenik. A különböző dialógus ablakok sokféle Qt-s ui elemet tartalmaznak, ezek közül a legfontosabbak:

- `QListView`: A `UsersListModel` és `ListModel` objektumok elemeit jeleníthetjük meg benne.
- `QCalendarWidget`: Az alkalmazás fő felületén megjelenő naptár.

- `QComboBox`: A beállításoknál, események létrehozásakor és szerkesztésekor le-nyíló lista.
- `QLineEdit`: A felhasználó az inputot ezekbe a mezőkbe írhatja be.
- `QPushButton`: Az alkalmazásban használt összes nyomógomb.
- `QTimeEdit`: A beállításoknál, események létrehozásakor és szerkesztésekor az időpontokat itt tudjuk beállítani.

Az alkalmazás sötét témája `qss` ([5]) fájl, a többi téma pedig `stylesheet`-ként állítódik be.

3.3. Tesztelés

A tesztelés minden szoftver fejlesztéséhez hozzátartozik. Szakdolgozatomnál a unit tesztelést, és a manuális végfelhasználói teszteket alkalmaztam. A végpontok tesztelését a Postman ([6]) alkalmazás segítségével teszteltem.

3.3.1. Egységtesztek

Az alkalmazás tesztelése a Qt `QTest` tesztelő framework-jével történt, amit unit tesztelésre szoktak használni. A tesztesetek a következők:

<i>Teszteset</i>	<i>Működés</i>
<code>testUsersConvertJsonArrayToList</code>	A <code>UsersListModel</code> osztály a szervertől kapott adatokat konvertálja át <code>User</code> típusúvá, amely már használható formátum a model számára.
<code>testEventsConvertJsonArrayToList</code>	A <code>ListModel</code> osztály a szervertől kapott adatokat konvertálja át <code>Event</code> típusúvá, amely már használható formátum a model számára.
<code>testFindTime</code>	A közös megfelelő időpont kereső talált üres időintervallumot.
<code>testFindNoTime</code>	A közös megfelelő időpont kereső nem talált üres időintervallumot.

3.1. táblázat. Egységtesztek

3.3.2. Végfelhasználói tesztek

A kliens nézetének (view) tesztelést manuálisan, végfelhasználói tesztek segítségével végeztem. Minden egyes végfelhasználói tesztesetnél volt egy elvárt működés, ez alapján dőlt el hogy a teszt sikeres volt-e.

Login dialog			
<i>Folyamat</i>	<i>Teszteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Belépés</i>	Létező felhasználóval és helyes jelszóval próbálunk belépni.	A belépés sikeres, megjelenik a naptár alkalmazás.	Sikeres
<i>Belépés</i>	Létező felhasználóval és helytelen jelszóval próbálunk belépni.	A belépés sikertelen, maradunk a belépés dialógusában.	Sikeres
<i>Belépés</i>	Nem létező felhasználóval próbálunk belépni.	A belépés sikertelen, maradunk a belépés dialógusában.	Sikeres
<i>Belépés</i>	Felhasználónév vagy jelszó (vagy mindkettő) nélkül próbálunk belépni.	A belépés sikertelen, maradunk a belépés dialógusában.	Sikeres

3.2. táblázat. Login dialog végfelhasználói tesztesetei

Create account dialog			
<i>Folyamat</i>	<i>Teszteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Új felhasználó létrehozása</i>	Létező felhasználóval próbálunk új felhasználót létrehozni.	Hibaüzenet, maradunk az új felhasználó létrehozó dialógusban.	Sikeres
<i>Új felhasználó létrehozása</i>	Nem létező felhasználóval és nem egyező jelszavakkal próbálunk új felhasználót létrehozni.	Hibaüzenet, maradunk az új felhasználó létrehozó dialógusban.	Sikeres
<i>Új felhasználó létrehozása</i>	Nem létező felhasználóval és egyező jelszavakkal próbálunk új felhasználót létrehozni.	Üzenet kapunk a létrehozás sikerességéről, ami után bezárul a dialógus, visszatérünk a belépés dialógusához.	Sikeres
<i>Új felhasználó létrehozása</i>	Felhasználónév vagy jelszó (vagy mindkettő) nélkül próbálunk felhasználót létrehozni.	Hibaüzenet, maradunk az új felhasználó létrehozó dialógusban.	Sikeres

3.3. táblázat. Create account dialog végfelhasználói tesztesetei

Settings dialog			
<i>Folyamat</i>	<i>Teszteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Téma változtatása</i>	Más témát (Dark vagy Light) választunk ki a listából, elmentjük a változtatást.	Következő belépéskor az alkalmazás témája a beállított téma lesz.	Sikeres
<i>Téma változtatása</i>	Más témát (Custom) választunk ki a listából, hiányosan vagy helytelenül adjuk meg a színeket, majd elmentjük a változtatást.	Hibaüzenet, maradunk a beállítások dialógusban.	Sikeres
<i>Téma változtatása</i>	Más témát (Custom) választunk ki a listából, hiánytalanul és helyesen adjuk meg a színeket, majd elmentjük a változtatást.	Következő belépéskor az alkalmazás témája a beállított téma lesz.	Sikeres

3.4. táblázat. Settings dialog végfelhasználói tesztesetei

Account dialog			
<i>Folyamat</i>	<i>Teszteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Kijelentkezés az alkalmazásból</i>	Rákattintunk a LOGOUT gombra, ezzel jelezve kijelentkezési szándékunkat.	A naptár alkalmazás bezárul, visszatérünk a belépés dialógusához.	Sikeres

3.5. táblázat. Account dialog végfelhasználói tesztesetei

Mutual spare time finder dialog			
<i>Folyamat</i>	<i>Teszteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Közös megfelelő időpont keresés</i>	Helytelenül adjuk meg az esemény időtartamát, az időintervallumot, vagy a meghívott felhasználókat majd közös megfelelő időpontot keresünk.	Hibaüzenet, maradunk a közös időpont kereső dialógusban.	Sikeres
<i>Közös megfelelő időpont keresés</i>	Helyesen adjuk meg az esemény időtartamát, az időintervallumot, és a meghívott felhasználókat majd közös megfelelő időpontot keresünk.	Az alkalmazás megadja nekünk a legkorábbi közös megfelelő időpontot, ha van ilyen.	Sikeres

3.6. táblázat. Mutual spare time finder dialog végfelhasználói tesztesetei

Admin dialog			
<i>Folyamat</i>	<i>Teszteteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Felhasználó törlése</i>	Kiválasztunk egy felhasználót, majd rákattintunk a DELETE gombra.	Az előbb kiválasztott felhasználó törlődik a listából.	Sikeres

3.7. táblázat. Admin dialog végfelhasználói tesztesei

Események			
<i>Folyamat</i>	<i>Teszteteset</i>	<i>Elvárt működés</i>	<i>Eredmény</i>
<i>Hozzáadás</i>	Új esemény hozunk létre.	Az esemény létrehozása sikeres, megjelenik a listában.	Sikeres
<i>Szerkesztés</i>	Létező eseményt szerkesztünk, elmentjük a változtatásokat.	A szerkesztés sikeres, frissül az esemény a listában.	Sikeres
<i>Szerkesztés</i>	Létező eseményt szerkesztünk, nem mentjük a változtatásokat.	A szerkesztés sikertelen, az eseményben nem történik változás.	Sikeres
<i>Szerkesztés</i>	Létező eseményt szerkesztünk, nem mi vagyunk az esemény tulajdonosa.	A szerkesztés sikertelen, az eseményben nem történik változás.	Sikeres
<i>Törlés</i>	Létező eseményt törlünk.	A törlés sikeres, az esemény törlődik a listából.	Sikeres
<i>Törlés</i>	Létező eseményt törlünk, nem mi vagyunk az esemény tulajdonosa.	A törlés sikertelen, az esemény nem törlődik a listából.	Sikeres

3.8. táblázat. Eseményeken végezhető műveletek végfelhasználói tesztesei

3.4. Fejlesztési lehetőségek

Fejlesztői lehetőségek közé sorolnám a felhasználók egymás közti üzenetek integrációját, vagyis hogy ha valakit meghívunk egy eseményre, el tudja dönteni hogy elfogadja az invitációt, vagy sem.

Email értesítések is emelnének a program szintjén. Például, minden reggel kapnánk egy összefoglalót arról, milyen eseményeink lesznek aznap.

Irodalomjegyzék

- [1] „C++ Documentation”. URL: <https://en.cppreference.com/w/>.
- [2] „Qt Documentation”. URL: <https://doc.qt.io/qt-6/classes.html>.
- [3] „Qt HTTP server example”. URL: <https://doc.qt.io/qt-6/qthttpserver-simple-example.html>.
- [4] „SQLite Documentation”. URL: <https://www.sqlite.org/docs.html>.
- [5] „Qss Templates”. URL: <https://qss-stock.devsecstudio.com/templates.php>.
- [6] „Postman Documentation”. URL: <https://learning.postman.com/docs/getting-started/overview/>.

Ábrák jegyzéke

2.1. Login dialógus	5
2.2. Hibás felhasználónév vagy jelszó	6
2.3. Hiányos felhasználónév vagy jelszó	6
2.4. Ez a felhasználó nem létezik	7
2.5. CreateAccount dialógus	8
2.6. Sikeres új felhasználó készítés	8
2.7. A felhasználónév már foglalt	9
2.8. Hiányos felhasználónév vagy jelszó	9
2.9. A jelszavak nem egyeznek	10
2.10. Admin felület	11
2.11. Clnr felület	12
2.12. Sötét téma	13
2.13. Custom téma	13
2.14. Settings dialógus	14
2.15. Mutual spare time finder dialógus	15
2.16. Sikeres közös megfelelő időpont keresés	15
2.17. Hiányos közös megfelelő időpont keresés	16
2.18. Hibás közös megfelelő időpont keresés	16
2.19. Sikertelen közös megfelelő időpont keresés	17
2.20. AllEvent dialógus	18
2.21. Help dialógus	19
2.22. Account dialógus	20
2.23. AddEvent dialógus	21
2.24. EditEvent dialógus	22
2.25. DeleteEvent dialógus	23
3.1. Usecase diagram	25
3.2. Szakdolgozat komponensei	26

3.3. Adatbázis megvalósítása	26
3.4. ListModel osztálydiagramja	28
3.5. UsersListModel osztálydiagramja	29
3.6. Model osztálydiagramja	30
3.7. Perzisztencia réteg hierarchiája	31
3.8. Common osztálydiagramja	32
3.9. UserAPI osztálydiagramja	33
3.10. EventAPI osztálydiagramja	34

Táblázatok jegyzéke

3.1. Egységtesztek	35
3.2. Login dialog végfelhasználói tesztesetei	36
3.3. Create account dialog végfelhasználói tesztesetei	36
3.4. Settings dialog végfelhasználói tesztesetei	37
3.5. Account dialog végfelhasználói tesztesetei	37
3.6. Mutual spare time finder dialog végfelhasználói tesztesetei	37
3.7. Admin dialog végfelhasználói tesztesetei	38
3.8. Eseményeken végezhető műveletek végfelhasználói tesztesetei	38