



EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

PROGRAMOZÁSELMÉLET ÉS SZOFTVERTECHNOLÓGIAI  
TANSZÉK

## Shalendar

*Témavezető:*

Pintér Balázs

egyetemi adjunktus, PhD

*Szerző:*

Kertész János

programtervező informatikus BSc

*Budapest, 2025*

## SZAKDOLGOZAT TÉMABEJELENTŐ

### Hallgató adatai:

Név: Kertész János

Neptun kód: AM2VZ8

### Képzési adatok:

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

Tagozat : Nappali

Belső témavezetővel rendelkezem

### Témavezető neve: Pintér Balázs

munkahelyének neve, tanszéke: ELTE IK, Programozásmélet és Szoftvertechnológia Tanszék

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

beosztás és iskolai végzettsége: egyetemi adjunktus, PhD

A szakdolgozat címe: Shalendar

### A szakdolgozat témája:

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

A dolgozat témája egy mindennapi életben használható time management szoftver megvalósítása. A projekt felépítése három fő felületre és egy bejelentkezési oldalra oszlik, az utóbbi lehetővé teszi több felhasználó számára a profil kezelését és az egymás közötti naptár megosztást.

#### Főoldal:

Az oldalon egy naptár található, mellette pedig minimum egy oszlop, amely a feladatkezelő szoftverekből ismert lista formátumot követi. A felhasználó igényei szerint több, saját tematikáinak megfelelő oszlopot is felvehet. Az oszlopban kártyák helyezhetők el, amelyeket a felhasználó szabadon hozzáadhat vagy törölhet. A kártyákon kötelezően megadható cím, valamint opcionálisan kezdeti dátumok, határidők és prioritások. Ezek a kártyák a naptár megfelelő napjaira húzhatók, így segítve a feladatok ütemezését.

#### Napi nézet:

A naptár adott napjára kattintva megjelenik a nap részletezése. Itt két lista található: Az egyik lista egy időjelző sávval rendelkező feladatlista, amelyben a naphoz tartozó, időponthoz kötött feladatok jelennek meg. A másik lista olyan teendőket tartalmaz, amelyek nem kötöttek időponthoz. Mindkét listában a feladatok „elvégeztnek” jelölhetők.

#### Több naptár kezelése:

A naptár mellett található egy plusz gomb, amely lehetővé teszi több naptár létrehozását és kezelését. Ezekhez a naptárakhoz további felhasználók is hozzáadhatók, valamint lehetőség nyílik a naptárakból egyes kártyák vagy teljes naptárak importálására a saját naptárba.

Budapest, 2024. 10. 05.

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>3</b>
<b>2. Felhasználói dokumentáció</b>	<b>4</b>
<b>3. Fejlesztői dokumentáció - Megoldási terv</b>	<b>5</b>
3.1. Rendszer architektúrája . . . . .	5
3.1.1. Alrendszerek és rétegek szerepei és felelősségei . . . . .	5
3.1.2. Alkalmazott technológiák és eszközök . . . . .	6
3.1.3. Fejlesztési módszertan . . . . .	8
3.1.4. Csomag diagramok (UML) . . . . .	9
3.2. A rendszer működése . . . . .	10
3.2.1. Kommunikációs diagram (UML) . . . . .	11
3.2.2. Tevékenységdiagram (UML) . . . . .	12
3.3. Adatbázis modell . . . . .	13
3.3.1. Az adatmodell áttekintése . . . . .	14
3.3.2. Tábla-szintű leírás . . . . .	14
3.4. Modul- és osztályszerkezet . . . . .	17
3.4.1. Backend modulok és rétegek . . . . .	17
3.4.2. Osztálydiagram (UML) . . . . .	17
3.4.3. Főbb metódusok leírása . . . . .	17
3.5. A felhasználói felület terve . . . . .	17
3.5.1. Képernyők listája és leírása . . . . .	17
3.5.2. Navigációs térkép . . . . .	17
3.5.3. Felhasználói események kezelése . . . . .	17
<b>4. Fejlesztői dokumentáció -Megvalósítás</b>	<b>18</b>
4.1. Fejlesztés közbeni döntések . . . . .	18
4.2. Kiemelt kódrészletek . . . . .	18

4.3. Komponens terv és telepítés . . . . .	18
<b>5. Fejlesztői dokumentáció - Tesztelés</b>	<b>19</b>
5.1. Tesztelési stratégia . . . . .	19
5.2. Tesztesetek . . . . .	20
5.3. Tapasztalatok és módosítások . . . . .	20
5.4. Nagy adatmennyiség melletti viselkedés . . . . .	20
5.5. Eredmények és hatékonyság elemzése . . . . .	20
5.6. Manuális frontend teszt . . . . .	20
<b>6. Összegzés</b>	<b>26</b>

# 1. fejezet

## Bevezetés

ide le lehet írni  
hogyan vannak a képek szövegei,  
hol vannak,  
valamint egy szómagyarázat  
(esetleg jelölés)  
ezt majd akkor kezdem  
ha végeztem  
mindennel.  
egyszerűsíttem a szöveget  
módvalamint személyszempontjából,  
kiemeléseket,  
dőlt karakterjelöléseket  
jelenít meg bizonyos konvenciók szerint

Harmadik személyű, tárgyilagosságot megfogalmazás

Mondatok szerkezete: Tény megállapítása + következtetés

## 2. fejezet

### Felhasználói dokumentáció

## 3. fejezet

# Fejlesztői dokumentáció - Megoldási terv

### 3.1. Rendszer architektúrája

Az alkalmazás a kliens-szerver modellt követve 3 fő komponensből áll. A Microsoft SQL Server adatbázis, az ASP.Net WebAPI alapú backend és a Vue.JS-alapú frontend. Az első két komponens a szerveret, míg a harmadik a klienst képviseli.

A 3 komponenes tisztán szétválasztható, ezzel biztosítva a moduláris fejlesztést. A frontend és a backend API hívások segítségével kommunikál. A backend és az adatbázis pedig közvetlen kapcsolatban állnak az Entity Framework-nek köszönhetően.

#### 3.1.1. Alrendszerek és rétegek szerepei és felelősségei

##### Frontend

Feladata a felhasználói interakciók kezelése valamint az üzleti logika megjelenítése. Az üzleti logika elérésére axios HTTP kéréseken keresztül történik. A kérés headerjébe automatikusan integrálja az autentikációhoz szükséges adatokat (JWT token) valamint a naptár azonosítóját amikor az releváns. Ezzel segítve, hogy a felhasználó csak számára elérhető adatokhoz férhessen hozzá.

## Backend API

Felelősségei közé tartozik, hogy a JWT token megfelelően generálva legyen a bejelentkezés során, tartalmazza a felhasználó azonosítóját, email címét, JWT ID-ját (egyedi, véletlenszerű GUID) valamint az user naptárakhoz való engedélyeit a token megszokott jellemzői mellett. (Issuer, Audience, Expiration...)

Feladata továbbá, hogy kezelje a kliens által küldött HTTP kéréseket. Ahol szükséges ellenőriznie, hogy a felhasználó rendelkezik-e érvényes tokennel, valamint a naptárakkal kapcsolatos tevékenységek esetén figyelje az írási, olvasási vagy tulajdonosi engedélyek meglétét. A token ellenőrzését a beépített [Authorize] attribútummal végzi. Az engedély ellenőrzés pedig a tokenben szereplő engedélyek és a fejlécben kapott naptár azonosító összehasonlításával történik.

Amikor megtörtént az adatok validálása és az üzleti logika végrehajtása a backend feladata, hogy értesítse az összes klienst a változásról amely az adott naptár valamelyik nézetén tartózkodik .

## Adatbázis

Az adatbázis szerepe, hogy hosszútávon, jól struktúráltan tárolják az adatokat a felhasználókról valamint azok naptáiról, engedélyeiről, kártyáiról.

Indexek, kulcsok és idegen kulcsok valamint megszorítások segítségével biztosítja a következetességet és teljesítményt. Ezek tényleges kapcsolatát a *3.3. Adatbázis modell* című pont alatt tárgyaljuk.

### 3.1.2. Alkalmazott technológiák és eszközök

#### Programozási nyelvek, keretrendszerek

- **.NET 8.0 SDK** – A backend teljes projektje .NET 8.0 épül.
- **C#** – backend logika és API implementáció.
- **ASP.NET Core** – REST API keretrendszer.
- **Entity Framework Core** – ORM a relációs adatbázis kezelésére.
- **JavaScript** – frontend logika.
- **Vue.js** – JavaScript keretrendszer.



- **HTML, CSS** – struktúra és stílus.
- **Vite** – frontend build és hot-reload.

### Fejlesztői eszközök

- **Visual Studio 2022** – backend fejlesztés, debug, teszt.
- **Visual Studio Code** – frontend fejlesztés, Vue komponensek.
- **Node.js** – frontend futtatási környezet.
- **Postman** – API-k kipróbálásához és manuális teszteléshez.
- **Git + GitHub** – verziókövetés.
- **SSMS** – SQL szerver kezelése, tesztelés, queryk írása.

### Külső csomagok (NuGet és npm)

A projekt során több külső könyvtárat használtam, melyeket NuGet illetve npm segítségével kezeltem.

#### Backend (NuGet csomagok):

- **Microsoft.AspNetCore.Authentication.JwtBearer** (v8.0.0) – JWT tokenek feldolgozásához és hitelesítéshez.
- **Microsoft.AspNetCore.SignalR** (v1.2.0) – kliens oldali értesítések a backendből SignalR hubok felhasználásával
- **Microsoft.EntityFrameworkCore** (v9.0.2) – ORM réteg relációs adatbázisokhoz.
- **Microsoft.EntityFrameworkCore.SqlServer** (v9.0.2) – SQL Server-specifikus EF Core provider.
- **Microsoft.EntityFrameworkCore.Tools** (v8.0.11) – EF Core migrációs és scaffold eszközök; csak fejlesztési célra használva.
- **Microsoft.VisualStudio.Web.CodeGeneration.Design** (v8.0.7) – scaffold eszközök a WebAPI fejlesztéshez.

- **Swashbuckle.AspNetCore** (v6.4.0) – Swagger generálása és dokumentáció REST API-hoz.
- **System.IdentityModel.Tokens.Jwt** (v8.5.0) – JWT tokenek létrehozása és kezelése.

#### Frontend (npm csomagok):

- **vue** (v3.5.13) – A Vue.js 3 keretrendszer magja.
- **vue-router** (v4.5.0) – Oldalak közötti navigáció Vue-ban.
- **axios** (v1.7.9) – HTTP kliens API hívásokhoz.
- **@microsoft/signalr** (v8.0.7) – SignalR kliens valós idejű frissítésekhez. A backendelből érkező értesítések feldolgozásához.
- **jwt-decode** (v4.0.0) – JWT tokenek tartalmának frontend oldali dekódolása.
- **lucide-vue-next** (v0.479.0) – Ikonkészlet Vue 3-hoz.
- **vuedraggable** (v4.1.0) – Drag-and-drop funkcionalitás Vue komponensekhez.

#### Fejlesztői függőségek:

- **vite** (v6.0.11) – Build és hot-reload rendszer Vue-hoz.
- **@vitejs/plugin-vue** (v5.2.1) – Vue támogatás Vite-hez.
- **vite-plugin-vue-devtools** (v7.7.1) – Vue fejlesztői eszközök bővítménye.

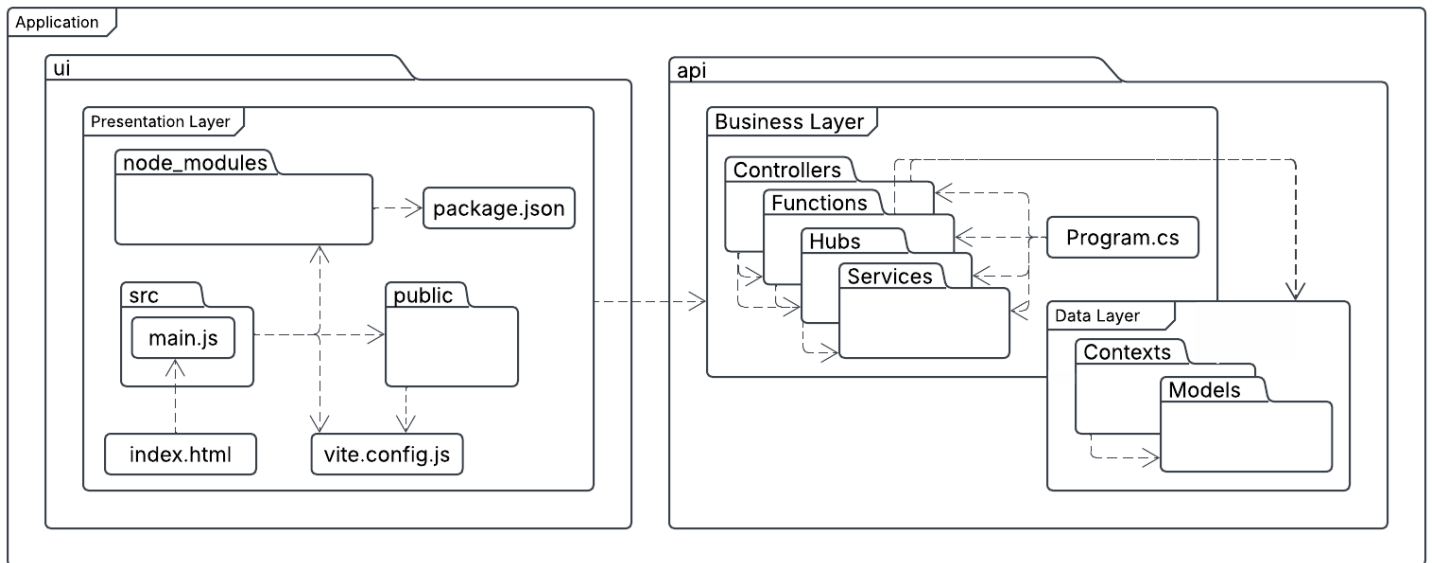
### 3.1.3. Fejlesztési módszertan

A fejlesztés során az első kitűzött cél egy működő MVP volt, csak funkcionálitást figyelve. Amint ez elkészült fokozatosan adtam hozzá funkciókat nézetek szerint csoportosítva. Amint a funkció elkészült manuális tesztelés, majd szükség esetén debugolás / refaktorálás után kezdtem a következő feladat implementálásába. A UI/UX dizájn megvalósításába a funkcionalitás 100%-os működése után kezdtem bele. Ezeket követte a unit majd integrációs tesztek megírása.

Ide még leírom  
hatom hogy  
vettem fel iss-  
ukat gitHubon

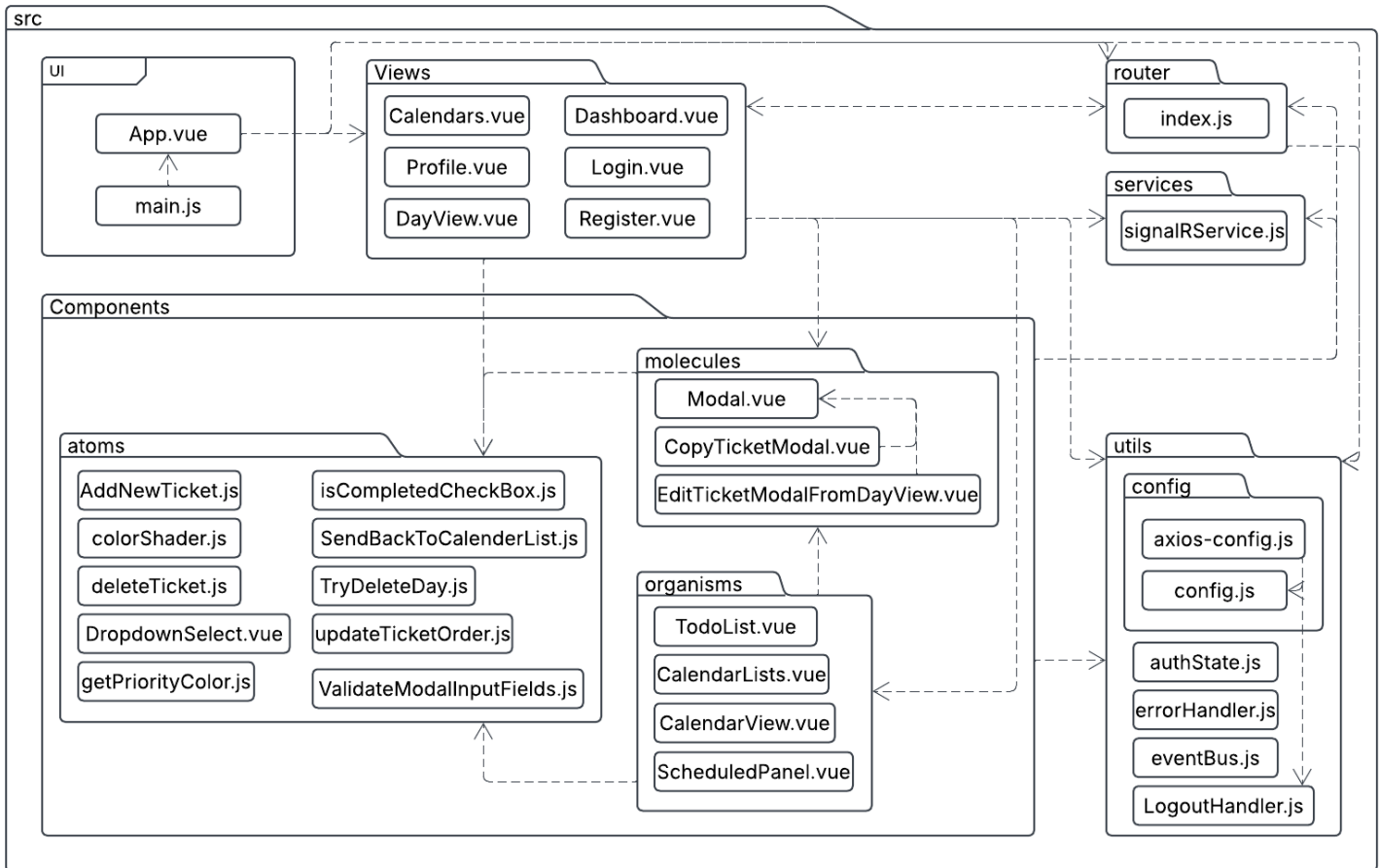
### 3.1.4. Csomag diagramok (UML)

(3.1., 3.2.) A csomag diagramokban amennyiben a mappa ikon látható, egy valós fizikai mappára utal, amennyiben a másik jelölés (a név a tárolón belül van, nem felette) akkor pedig egy nem fizikai azaz logikai egységről beszélünk. Minden egyéb jelölés valós fizikai fájlokat takar. A nyíl a importáló csomagtól közvetlenül az importált elemre mutat. Amennyiben nem egy elemre hanem egy logikai rétegre/mappára mutat az adott komponens mindegyike importálhatja azt.



3.1. ábra. Applikáció szintű csomag diagram

A program belépési pontja az index.html, amely a src/main.js mappáját importálja, a részletesebb megértés érdekében az alábbiakban<sup>(3.2.)</sup> látható az src mappa csomag diagramja is. Az api működése nem kerül bővebb kifejtésre ebben a pontban, mivel csomagok szempontjából annak működése jól leolvasható a fenti<sup>(3.1.)</sup> ábráról.



3.2. ábra. Src mappa csomag diagrammja

## 3.2. A rendszer működése

(3.3., 3.4, 3.5, 3.6) A működést 2 féle diagrammal mutatom be, az első típus<sup>(3.3., 3.4)</sup> hangsúlyt fektet a kommunikáció módjának bemutatására míg a második típus<sup>(3.5., 3.6)</sup> a kommunikáció sorrendjét, valamint hibakezeléseket hivatott szemléltetni.

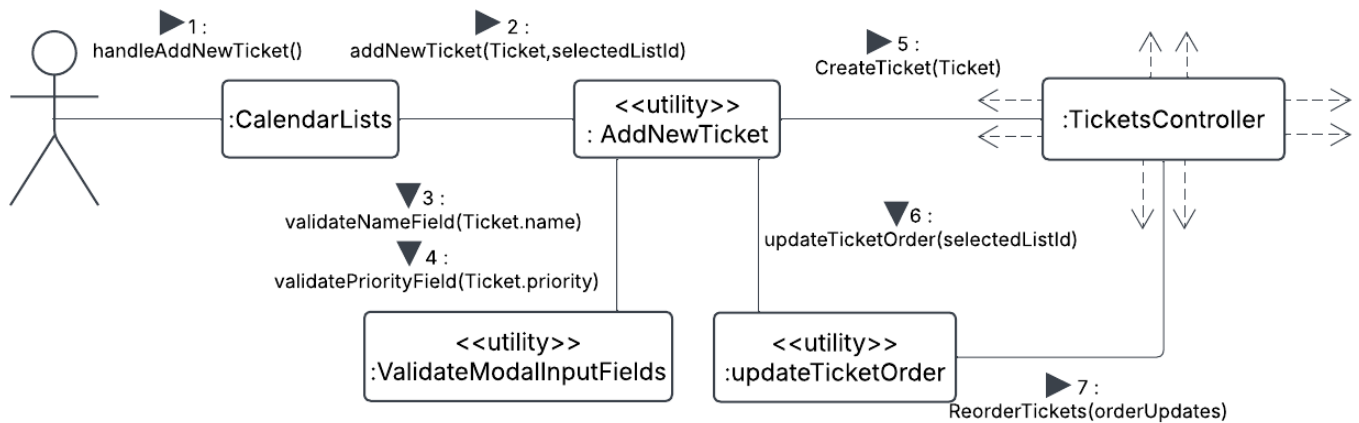
A frontend Vue 3 keretrendszeren alapuló implementációt reprezentál, amely a komponenseket JavaScript objektumokként valósítja meg. Mivel ez általában névtelen objektumokat generál ezért a komponensre annak a fájlnek a nevével fogunk hivatkozni, melyben található. A folyamatban részt vesznek fontos, objektumhoz nem kötött segédfüggvények is, melyeket «utility» sztereotípiával és a fent említett névadási konvenciókkal fogunk megjeleníteni a lent látható diagramokban.

A diagramok azt ábrázolják, hogy hogyan kezeli a program azt amikor a felhasználó egy új kártyát kíván felvenni valamelyik `calendarList`-be.

### 3.2.1. Kommunikációs diagram (UML)

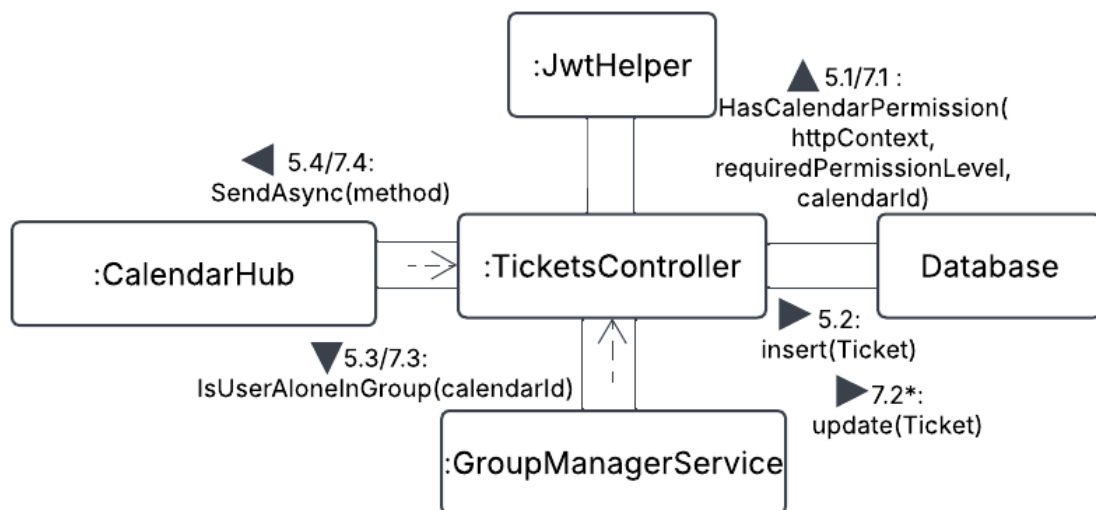
(3.3.) Az alábbi diagrammon látható egy sikeres folyamat működése, a diagramnak hála jól láthatóak lesznek a kommunikációban részt vevő objektumok kapcsolatai és leolvasható lesz a kommunikáció sorrendje is.

A függvényhívások paraméterezése során csak a sikeres hívás, valamint az adat továbbítás szempontjából releváns információkat jelenítem meg, ezzel könnyítve a diagram átláthatóságát.



3.3. ábra. Kommunikációs diagram, kliens

(3.4.) A controllerek nagy része a lent látható logikát követve épül fel. A `JwtHelper` felel a naptárral kapcsolatos engedélyek ellenőrzéséért, a `GroundManagerService` valamint a `CalendarHub` pedig a kliens értesítéséről ha változás történik egy adott naptár azonosító szerinti csoportban.

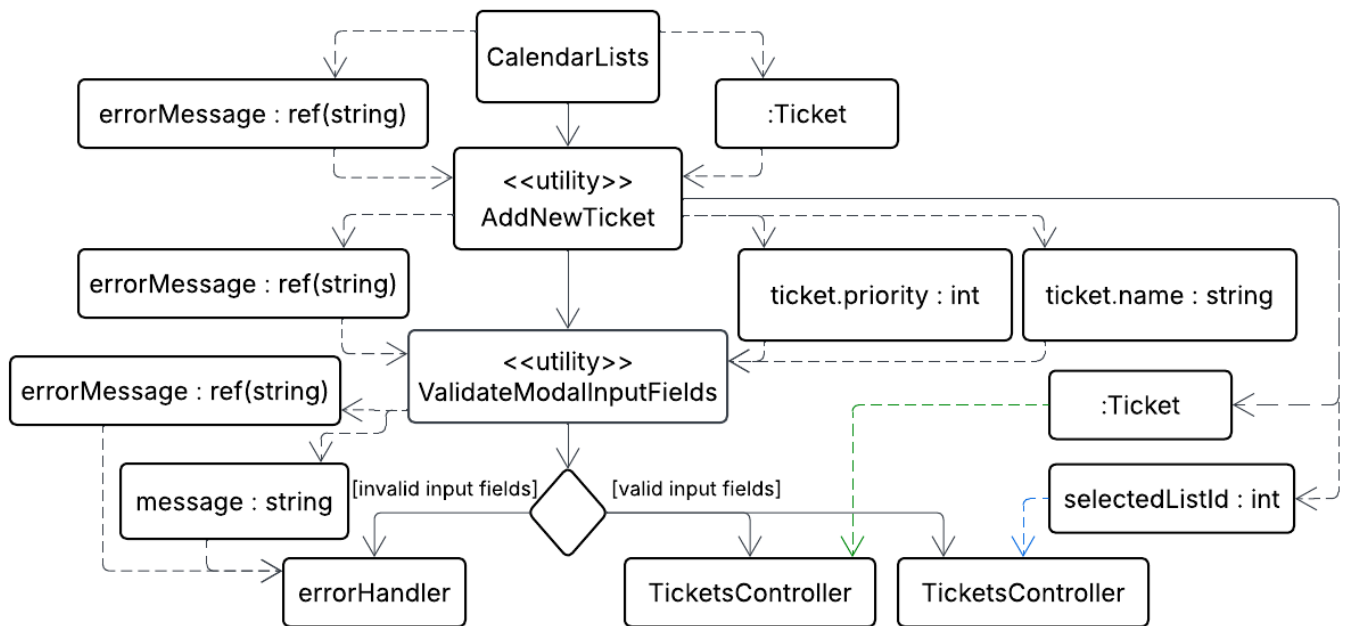


3.4. ábra. Kommunikációs diagram, szerver

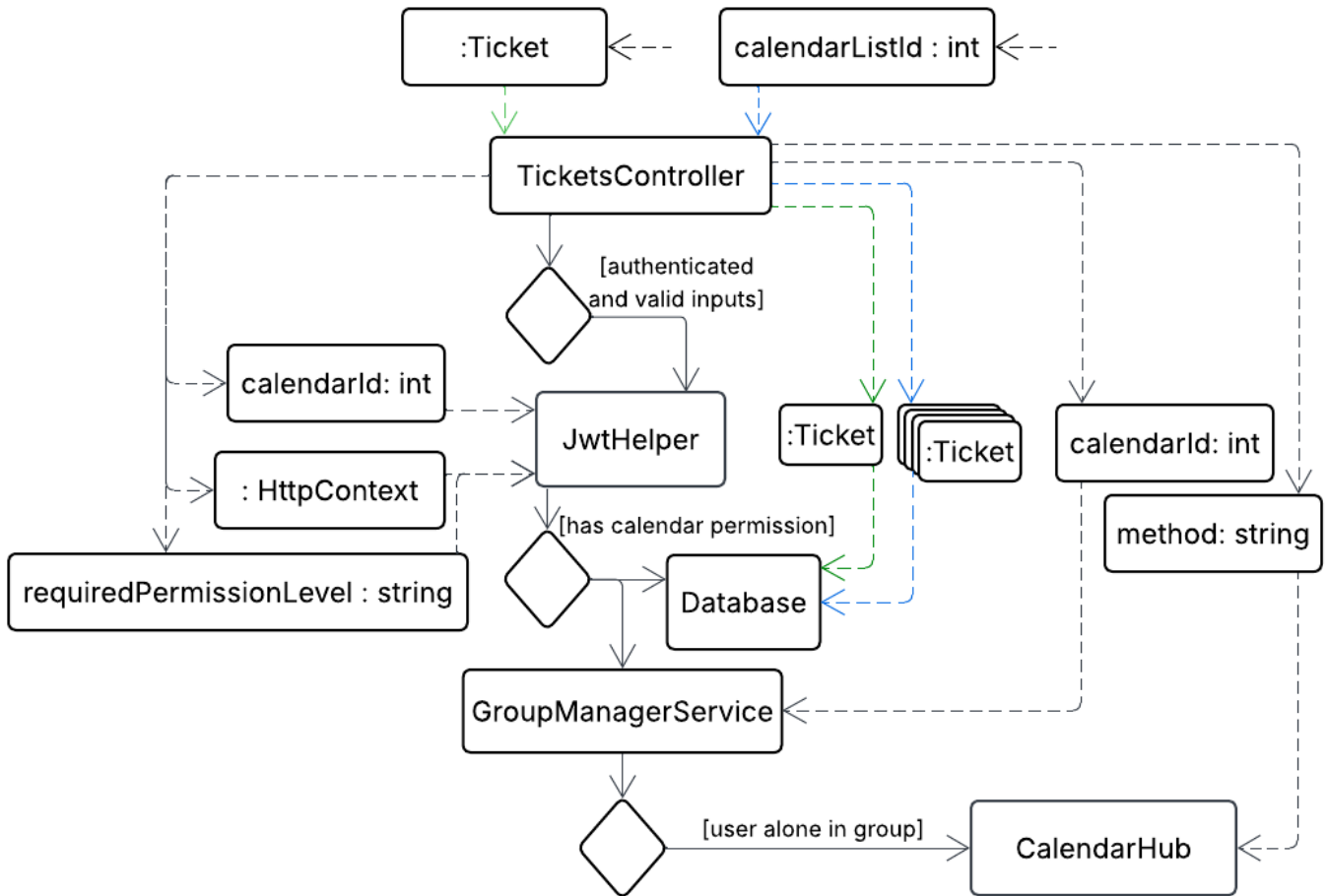
### 3.2.2. Tevékenységdiagram (UML)

(3.5.) Az alábbi diagramban `ref(...)` kulcsszó jelöli azon objektumokat melyek változása esetén frissül a felhasználói interfész is. Az olvashatóság érdekében nincs feltüntetve, de ilyen objektumok a `calendarLists` is, mely szintén frissül amikor hozzáadjuk a felhasználó által meghatározott kártyát.

Könnyebb követhetőség érdekében a `TicketsController` hívásokat kék és zöld színekkel jelöltem, hogy leolvasható legyen a későbbi diagramon<sup>(3.6.)</sup> is, hogy pontosan melyik tevékenység melyikhez tartozik.



3.5. ábra. Tevékenység diagram, kliens



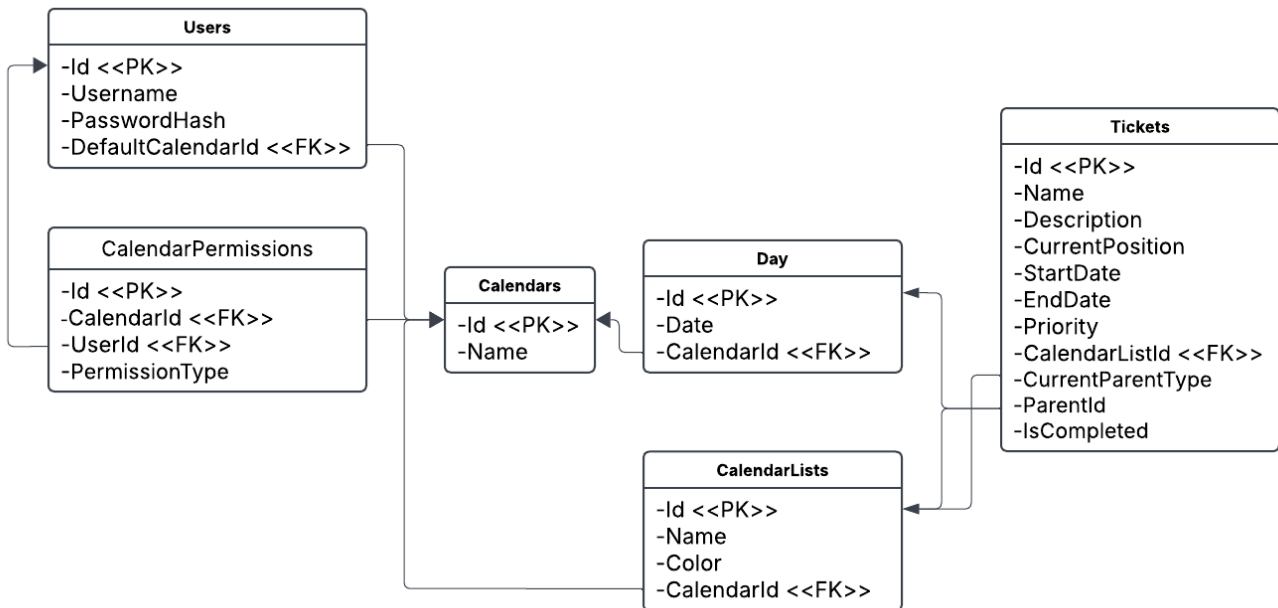
3.6. ábra. Tevékenység diagram, szerver

### 3.3. Adatbázis modell

Az adatbázis 5 táblából áll, nincs a csoporttól elkülönülő tábla, azaz az összes kapcsolatban áll legalább 1 másikkal. Úgy lett tervezve, hogy dinamikusan és hatékonyan kezelhető legyen a felhasználóhoz tartozó naptárak, listák és jegyek száma és azok tulajdonságai. Az adatbázis tervezése során fontos szempont volt a redundancia kerülésére, valamint a komplexitás és a letisztultság között meghúzódó egyensúly megtalálása.

(3.7.) Az alábbi diagramon a nyilak a táblák közötti kulcskapcsolatokat jelölik. A «PK» a Primary Key-t (elsődleges kulcsot), míg a «FK» a Foreign Key-t (idegen kulcsot) szimbolizálja.

### 3.3.1. Az adatmodell áttekintése




3.7. ábra. Adatbázis entitások és kapcsolatok

### 3.3.2. Tábla-szintű leírás

(3.8., 3.10., 3.11., 3.12., 3.13.) Az alábbiakban minden adatbázis entitást külön vizsgálunk, képeken láthatóak lesznek a tábla argumentumnevei, típusai, NULL értéket engedélyező beállításai (nullable), valamint az elsődleges és másodlagos kulcsok is. A táblákat a hivatkozási hierarchia aljától kezdve mutatjuk be, hogy az idegen kulcsok értelmezésekor a hivatkozott tábla szerkezete már ismert legyen. Az elsődleges kulcsot egy sárga kulcs, míg a másodlagosat egy kék pont jelöli.



(3.8.) A naptárakat tartalmazó tábla az egyetlen amely nem rendelkezik idegen kulccsal. Illetve jellemzően az idegenkulcsok ennek a táblának az azonosítójára hivatkoznak.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(255)	<input type="checkbox"/>

3.8. ábra. Calendars





(3.9.) A napokat reprezentáló tábla mindig egy naptárhoz van kötve, elsősorban azonosító alapján kezeljük a velük kapcsolatos tevékenységeket. De előfordul, hogy a date és calendarId-val azonosítjuk egyértelműen. A naptáraknak csak azon napjai léteznek adatbázisban eltárolt objektumként melyekhez legalább 1 jegy tartozik.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Date	date	<input type="checkbox"/>
	CalendarId	int	<input checked="" type="checkbox"/>

3.9. ábra. Days



(3.10.) A naptárlisták esetében kiválasztható a kívánt szín a felhasználó részéről, a választott megjelenést HEX kódok formájában tároljuk.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(255)	<input type="checkbox"/>
	Color	varchar(50)	<input checked="" type="checkbox"/>
	CalendarId	int	<input checked="" type="checkbox"/>

3.10. ábra. CalendarLists




(3.11.) A ticketek esetében megfigyelhető egyedül, hogy másik azonosítóra hivatkozik idegenkulcsuk. Ez azt a naptárlistát jelöli amelyhez létrehoztuk a ticketet. Innen kapja például a színét, vagy amikor egy napra kiosztott ticketet visszateszünk a naptár nézetre ennek a mezőnek köszönhetően fogja megtalálni a helyét.

A ParentId tartalmazhatja egy nap vagy egy naptár lista azonosítóját is, azt hogy pontosan melyiket azt a CurrentParentType argumentum alapján dől el. Melynek értékei lehetnek "CalendarList", "ScheduledList" és "TodoList". Utóbbi kettő esetén tartalmazza a nap azonosítóját, a megkülönböztetés azért van, hogy egyértelmű legyen, hogy melyik oldalon szeretnénk megjeleníteni a napi nézeten belül.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Name	varchar(255)	<input type="checkbox"/>
	Description	text	<input checked="" type="checkbox"/>
	CurrentPosition	int	<input checked="" type="checkbox"/>
	StartTime	time(7)	<input checked="" type="checkbox"/>
	EndTime	time(7)	<input checked="" type="checkbox"/>
	Priority	int	<input checked="" type="checkbox"/>
	CalendarListId	int	<input checked="" type="checkbox"/>
	CurrentParentType	varchar(50)	<input checked="" type="checkbox"/>
	ParentId	int	<input checked="" type="checkbox"/>
	IsCompleted	bit	<input type="checkbox"/>




3.11. ábra. Tickets

(3.12.) Az felhasználók esetén alkalmazunk egy unique (egyediségi) megszorítást az email mezőre, tekintve hogy ez alapján azonosítjuk a felhasználónkat. A jelszót hashelt formában tároljuk, melyet a szerver titkosít.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	Username	varchar(255)	<input type="checkbox"/>
	PasswordHash	varchar(255)	<input type="checkbox"/>
	Email	nvarchar(255)	<input type="checkbox"/>
	DefaultCalendarId	int	<input checked="" type="checkbox"/>

3.12. ábra. Users

(3.13.) A naptárakhoz való jogosultságok kezelését ezen a táblán keresztül végezzük, a szerveren található HasCalendarPermission függvény ezen tábla értékei alapján dolgozik.

	Column Name	Data Type	Allow Nulls
	Id	int	<input type="checkbox"/>
	CalendarId	int	<input checked="" type="checkbox"/>
	UserId	int	<input checked="" type="checkbox"/>
	PermissionType	varchar(50)	<input checked="" type="checkbox"/>

3.13. ábra. CalendarPermissions

## SQL és C# típusok megfeleltetése Entity Framework-ben

SQL típus	C# típus (EF-ben)
bit	bool
int	int
varchar(n)	string
nvarchar(n)	string
text	string
date	DateTime
time(7)	TimeSpan

## 3.4. Modul- és osztályszerkezet

### 3.4.1. Backend modulok és rétegek

Controller, Service, Repository, Model felépítése

### 3.4.2. Osztálydiagram (UML)

Főbb osztályok és kapcsolataik (öröklés, asszociáció)

### 3.4.3. Főbb metódusok leírása

Módszerek bemenő paraméterei, kimenetei, működésük

## 3.5. A felhasználói felület terve

### 3.5.1. Képernyők listája és leírása

Pl. LoginView, CalendarView, DayView, TicketModal

### 3.5.2. Navigációs térkép

Képernyők közti kapcsolatok vizuális ábrázolása

### 3.5.3. Felhasználói események kezelése

Főbb interakciók leírása (pl. gombnyomás, drag-and-drop)

## 4. fejezet

# Fejlesztői dokumentáció -Megvalósítás

johentek ilyen  
kódolási kon-  
venciók, branch  
kezelés, stb  
mint a btga-  
ben ami van

### 4.1. Fejlesztés közbeni döntések

Adatábrázolás és struktúra

Felhasznált komponensek

Nyelvi eszközök alkalmazása és indoklása

### 4.2. Kiemelt kódrészletek

Hibakezelés, validálás, adatlekérés példái

### 4.3. Komponens terv és telepítés

Fizikai komponensek (frontend, backend, adatbázis)

Telepítési folyamat leírása (lokálisan és szerveren)

## 5. fejezet

# Fejlesztői dokumentáció - Tesztelés

### 5.1. Tesztelési stratégia

Mehet egy  
techstack is ide,  
nugetekkel mint  
implMo

Modultesztek és rendszertesztek

Fekete- és fehérdoboz tesztelés

## 5.2. Tesztesetek

Bemenet, elvárt kimenet, tesztleírás

## 5.3. Tapasztalatok és módosítások

Implementációs döntések változtatása tesztek alapján

## 5.4. Nagy adatmennyiség melletti viselkedés

Teljesítménytesztek és értékelés

## 5.5. Eredmények és hatékonyság elemzése

Optimalizációs szempontok, erőforráskezelés

## 5.6. Manuális frontend teszt

Bejelentkezési oldal funkciói

### Regisztráció

<b>Given</b>	A felhasználó megnyitotta az alkalmazást
<b>When</b>	A regisztrációs lehetőséget választja
<b>Then</b>	A rendszer belépteti a főoldalra, ahol láthatja a naptárát és a feladatlistáit.

### Bejelentkezés

<b>Given</b>	A felhasználó megnyitotta az alkalmazást
<b>When</b>	A bejelentkezés lehetőséget választja, majd bejelentkezik
<b>Then</b>	A rendszer elvégzi a regisztrációt, majd a bejelentkezési oldalra irányítja a felhasználót.

**Főoldal funkciói****Új feladatlista létrehozása**

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A feladatlista hozzáadása opciót választja (+ gomb a feladatlisták felett) majd megfelelően paraméterezi a megnyíló modalt
<b>Then</b>	A feladatlista megjelenik a főoldalon.

**Feladatlista módosítása**

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	Az adott feladatlistán a manage opciót választja, majd a felnyíló modalt megfelelően paraméterezi
<b>Then</b>	A feladatlista valamint a hozzá tartozó kártyák megváltoznak (a kártyák csak akkor ha változott a feladatlista színe).

**Feladatlista törlése**

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	Az adott feladatlistán a manage opciót választja, majd a felnyíló modalon a törlés gombra nyom
<b>Then</b>	A feladatlista valamint a hozzá tartozó kártyák törlődnek.

**Kártya létrehozása az adott feladatlistához**

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A feladatlistán új kártya létrehozására kattint, majd megfelelően paraméterezi azt.
<b>Then</b>	A kártya megjelenik a kívánt oszlopban.

**Kártya törlése az adott feladatlistáról**

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A kártyát kitörli
<b>Then</b>	A kártya törlődik a feladatlistáról

### Kártya másolása saját naptárba

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A naptár másolása gombra kattint majd kiválasztja a felugró modalon melyik naptárba szeretne másolni
<b>Then</b>	A rendszer lemásolja az adott kártyát feltéve hogy egy pontosan ilyen még nem létezik az adott naptárban

### Feladatlista kártyáinak újrarendezése

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	Az adott feladatlistán megfog egy kártyát majd a listán belül mozgatva új pozícióba teszi
<b>Then</b>	A feladatlistában szereplő kártyák sorrendje frissül

### Kártya áthelyezése

<b>Given</b>	A felhasználó a főoldalon van és van egy oszlop egy meglévő kártyával.
<b>When</b>	A felhasználó a kártyát a naptár valamelyik mezőjére húzza. (balra időponthoz köti ami egy modal megfelelő paraméterezésével történik)
<b>Then</b>	A kártya átkerül a naptár adott napjára.

### Naptár napjának megnyitása

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A naptár napjára kattint
<b>Then</b>	A naphoz tartozó napi nézet megnyílik.

### További naptárak kezelése

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A naptáron lévő manage gombra kattint
<b>Then</b>	A további naptárak oldal megnyílik.

### Naptár másolása saját naptárba



<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A naptár másolása gombra kattint majd kiválasztja a felugró modalon melyik naptárba szeretne másolni
<b>Then</b>	A rendszer lemásolja a naptárhoz tartozó összes kártyát a duplikátumok szűrésére odafigyelve

## Napi nézet funkciói

### Kártya létrehozása

<b>Given</b>	A felhasználó a napi nézet oldalon van
<b>When</b>	Az új kártya létrehozása gombra (+) kattint, majd megfelelően paraméterezi azt.
<b>Then</b>	A kártya megjelenik a todo listában ha nem lett időponthoz kötve, amennyiben igen a scheduled listában lesz látható.

### Kártya törlése

<b>Given</b>	A felhasználó a napi nézet oldalon van
<b>When</b>	A kártyát kitörli
<b>Then</b>	A kártya törlődik

### Todo lista kártyáinak újrendezése

<b>Given</b>	A felhasználó a napi nézet oldalon van
<b>When</b>	A todo listán megfog egy kártyát majd a listán belül mozgatva új pozícióba teszi
<b>Then</b>	A todo listában szereplő kártyák sorrendje frissül

### Kártya megjelölése elvégzettként

<b>Given</b>	A felhasználó a napi nézet oldalán van
<b>When</b>	A kártyát megjelöli elvégzettként
<b>Then</b>	A kártya elvégzettként lesz megjelenítve

### Kártya időpontra osztása

<b>Given</b>	A felhasználó a napi nézet oldalán van és van legalább 1 kártya az időponthoz nem kötött feladatlistában
<b>When</b>	A felhasználó a kártyát szerkesztésére nyitja, majd időponthoz köti
<b>Then</b>	A kártya a megfelelő időpontban megjelenik az időponthoz kötött feladatlistában

#### Kártya visszaküldése a főoldalra

<b>Given</b>	A felhasználó a napi nézet oldalán van és van legalább 1 kártya ami a főoldal valamelyik feladatlistájában volt.
<b>When</b>	A felhasználó a kártyát visszaküldi a főoldalra
<b>Then</b>	A kártya visszakerül az eredeti feladatlistájába

#### Kártya másolása saját naptárba

<b>Given</b>	A felhasználó a főoldalon van
<b>When</b>	A naptár másolása gombra kattint majd kiválasztja a felugró modalon melyik naptárba szeretne másolni
<b>Then</b>	A rendszer lemásolja az adott kártyát feltéve hogy egy pontosan ilyen még nem létezik az adott naptárban

### Több naptár kezelése funkciói

#### Új naptár létrehozása

<b>Given</b>	A felhasználó a több naptár fülön van
<b>When</b>	Az új naptár létrehozása gombra kattint
<b>Then</b>	A rendszer létrehoz egy új naptárt, amelyhez további felhasználókat is hozzáadhat.

#### Naptár törlése vagy követésének megszüntetése

<b>Given</b>	A felhasználó a több naptár fülön van
<b>When</b>	A naptár törlése gombra kattint
<b>Then</b>	Amennyiben a felhasználó az utolsó owner a naptár és minden hozzá tartozó adat törlődik, ellenkező esetben csak az user hozzáférése

**Profil funkciói****Jelszó változtatása**

<b>Given</b>	A felhasználó a profil fölön van
<b>When</b>	A jelszó változtatása gombra kattint majd megfelelően paraméterezi a modalt
<b>Then</b>	A jelszava megváltozik

**Profil törlése**

<b>Given</b>	A felhasználó a profil fölön van
<b>When</b>	A profil törlése gombra kattint
<b>Then</b>	A felhasználó és minden hozzá kapcsolódó adat törlődik. A hozzá tartozó naptárak a naptár törlési szabályai szerint kerülnek kezelésre.

## 6. fejezet

### Összegzés